FALCONAge

Biological age and aging-clock scoring from multiomic data, in Python and R, on CPU or GPU.

Bhagesh Hunakunti

Table of contents

Chapter 1. FALCONAge 6
Scope and limitations 7
Further reading 9
Citation 9
Chapter 2. Install 11
Installation 11
Complete walkthrough, from an empty machine 13
Chapter 3. Getting started 17
Quick start 17
Scoring 18
Interpreting the result 18
Chapter 4. The Claude skill 20
Chapter 5. Loading your data 21
What reads what 21
Raw Illumina IDATs 21
A beta matrix 22
A GEO series matrix 22
A ComputAgeBench study 23
RRBS and bisulfite coverage 23
Nanopore bedMethyl 23
A targeted assay 24
Clinical chemistry 24
Olink NPX 25
NMR metabolomics 25
Bulk RNA-seq counts 26
After loading 26
Chapter 6. Preprocessing, imputation and coverage 27
6. Preprocessing, imputation, missing features 27
Platform coverage 29
Chapter 7. Raw arrays to betas 31
Raw IDATs 31
20. From raw arrays to betas 31
Chapter 8. Choosing a clock 36
Choosing by research question 37

Scale type and permitted operations 40
Coverage and validity 40
Feature count and coefficient mass 41
Reliability 42
Specimen type 43
Availability 43
The clock catalogue 44
Chapter 9. What a score means 45
The clock catalogue 45
Training targets, and what a score means 46
Scale types and permitted operations 47
8. Output types and units 49
Chapter 10. Clock catalogue 51
Bundled: 69 clocks 52
Untraced: 52 clocks 57
Licensed: 57 clocks 61
Querying it from code 66
Chapter 11. Age acceleration 67
Age acceleration 67
9. Downstream statistics on an aging score 68
Chapter 12. Uncertainty, specimen and platform 70
Measurement uncertainty 70
19. Uncertainty, specimen and platform 70
Chapter 13. Clinical chemistry and survival 79
Clinical chemistry 79
Survival and association 80
Chapter 14. Benchmarking and validation 82
Benchmarking 82
11. Benchmarking and validation 82
Chapter 15. Beyond the clock score 85
Disorder: entropy, drift and the barometer 85
Repertoire structure 86
Where a clock's weight sits 88
Reliability: trait, state and technical 89
Mosaic ages within a tissue 89
What none of this is 90
Chapter 16. Figures and reports 91
Reporting 91
10. Plots 92
Chapter 17. Figure gallery 95
Start here 95
Is the clock calibrated 97
Age acceleration 99
Outcomes 104
How the clocks relate 105

Benchmarking 109
Data quality 112
Other 116
In total 124
Chapter 18. The science of aging clocks 125
1. Definitions and framing 125
References 127
Chapter 19. Biology and measurement 128
2. Biology of aging that the scores encode 128
3. Measurement chemistry, per omics layer 131
Chapter 20. 4. Algorithm families with full math 135
4.1 Penalised linear regression, the field's default 135
4.2 PC clocks 135
4.3 Survival models 135
4.4 Klemera–Doubal method (KDM) 136
4.5 Homeostatic dysregulation (HD) 137
4.6 Mitotic / replicative clocks 137
4.7 GrimAge architecture 138
4.8 DunedinPACE and DunedinPoAm 139
4.9 Mammalian universal clocks 139
4.10 Deconvolution as a model 139
4.11 Binarisation clocks 140
4.12 Rank-based transcriptomic clocks 141
4.13 Spatial single-cell clocks 141
4.14 Single-cell methylation clocks 141
4.15 Deep learning 142
4.16 SystemsAge 142
4.17 Ensembling and stacking 142
4.18 tAge, relative transcriptomic clocks 143
4.19 SCALE - literature-informed single-cell aging score 144
4.20 Transform reference table 144
Chapter 21. Published formulas and constants 146
5. Exact published formulas and constants 146
Appendix F - Reproducing the four reference computations by hand 148
Chapter 22. 7. The clock catalogue 150
7.1 Distributions 150
7.2 Selected clocks with their key parameters 151
7.3 Clocks outside pyaging worth supporting 154
Chapter 23. Genotype and structure above the probe 155
21. Genetic variation under the probe 155
22. Structure above the single probe 156
Chapter 24. 23. Building a clock of your own 157
23.1 The objective 157
23.2 Choosing the penalty honestly 157
23.3 The age transform 158

23.4 Feature selection is not settled ... 158
23.5 Two metrics that disagree, and both are needed ... 159
23.6 The step that gets skipped ... 159
23.7 What to publish ... 159
23.8 Fitting one here ... 159
Chapter 25. Failure modes and the frontier ... 160
15.2 Where the paper and the coefficients disagree ... 161
16. Failure modes ... 163
17. The frontier, and what FALCONAge does not yet compute ... 164
Chapter 26. 18. Datasets, registries, reference resources ... 167
Chapter 27. References ... 168
Prior software this package builds on ... 168
Foundational clocks ... 168
Non-methylation clocks ... 169
Methods, tooling, benchmarking ... 170
Biology of aging ... 171
Sources traced while implementing the clocks they describe ... 172
Mechanism, limits and translation ... 172
Beyond the clock score: disorder, repertoire and coefficient mass ... 172
Chapter 28. Architecture ... 174
What of this exists ... 174
1. Scope, decisions and repository layout ... 176
Chapter 29. Core engine and clock registry ... 184
2. Core engine ... 184
3. Clock registry ... 188
Chapter 30. 4. Algorithms, the complete operation catalogue ... 196
4.1 Preprocess ops ... 197
4.2 Postprocess transforms ... 199
4.3 Model classes ... 200
4.4 The ten special forward passes ... 201
4.5 Clinical chemistry clocks ... 203
4.6 Op composition and validation ... 204
4.7 Scaffold-only clocks ... 205
Chapter 31. Download, preprocessing, scoring and reports ... 208
5. Download module ... 208
6. Preprocess module ... 211
7. Scoring, analysis, plots, report ... 214
15. The layer that says what a score is worth ... 219
Chapter 32. 8. R package ... 221
8.1 What “identical” means here ... 221
8.2 Environment management ... 221
8.3 The bridge ... 222
8.4 Public API - one-to-one with Python ... 223
8.5 S3 classes ... 223
8.6 DESCRIPTION ... 224

8.7 Roxygen conventions 227
8.8 Command-line front end 228
8.9 pkgdown reference groups 228
Chapter 33. Docker, testing, documentation and CI 231
9. Docker 231
10. Testing 234
11. Documentation 237
12. CI/CD and publishing 240
Chapter 34. GPU support: what was tested, on what, and what it is worth 243
The machine 243
What was already present, and what had to be installed 243
1. Device and dtype resolution 244
2. Which clocks the device actually reaches 244
3. CPU and GPU do not agree bit for bit 245
4. Speed: the GPU makes the shipping clocks slower 246
A performance bug this found 247
Reproducing this 247
14. GPU computing 248
Chapter 35. Design history 250
15. Design blueprint 250
13. Build order 252
14. Open questions 254
Chapter 36. Other software 255
12. Existing software landscape 255
13. Prior implementations 257
Chapter 37. Appendices 260
Appendix A - Full pyaging clock registry (173 clocks) 261
Appendix B - tAge transcriptomic clock registry (60 models) 277
Appendix C - ComputAgeBench benchmark studies (65 datasets, blood/saliva/buccal) 282
Appendix D - organAging organ protein panels 286
Appendix E - Preprocess and postprocess operation inventory 286

Generated from the FALCONAge documentation site, https://bhagesh-h.github.io/FALCONAge/

Chapter 1. FALCONAge

FALCONAge computes biological age scores from molecular data. It reads what a laboratory actually produces, applies published aging clocks to it, and returns each score with its unit, its provenance, and an estimate of how much of it is measurement noise.

An aging clock is a supervised model mapping molecular features, most often DNA methylation at specific CpG sites, to a single number. Roughly two hundred have been published since 2013, and they were not fitted to the same target. Some were regressed on calendar age, some on a survival-weighted composite and then rescaled into years, some on the rate of change in organ-system biomarkers tracked longitudinally, some on cumulative stem-cell divisions. Those are four different quantities. Reported together as "biological age" they look comparable, and the arithmetic that mixes them has no defined meaning.

This package treats that as the problem to solve rather than a caveat to mention. Every catalogue entry declares the target its clock was trained on, the unit the output carries, and which downstream operations are consequently defined. Asking for age acceleration on a pace-of-aging clock returns an error naming the units, not a number. Scoring a placenta clock on blood is refused. Scoring saliva on a whole-blood clock warns and cites the size of the error measured in people sampled both ways. The refusals are the design, and each one names the measurement it rests on.

The second half of the package is about what a single number is worth. Technical replicates of the same DNA differ by up to nine years on prominent clocks, and only 18% of probes on the 450K array reach an intraclass correlation of 0.5 in whole blood. A score reported without that context invites a reader to believe a difference that the assay alone could have produced. FALCONAge propagates published per-probe reliability through each clock's own weights to give a standard error per sample, and calibrates distribution-free prediction intervals against known ages.

It is one numerical core in Python, called from R through reticulate, so a result obtained in either language is the same bits rather than two implementations that agree approximately. The catalogue holds 178 clocks; 69 of them score offline, and the rest are documented with the reason they cannot, which is licensing or untraced provenance rather than missing code.

These are instruments for population research, not a clinical judgement about one person. FALCONAge computes what the papers specify and records how it did so; whether a given clock is fit for a given decision is not something a scoring tool can certify, and the limits below say what else it cannot do.

Input

Input	Reader	Notes
Raw Illumina IDATs	`read_idat_dir()`	27K, 450K, EPIC v1, EPIC v2. Full chain to betas: pOOBAH detection, noob background correction, optional BMIQ
Beta matrix (CSV, Parquet)	`read_betas()`	
GEO series matrix	`read_series_matrix()`	

Input	Reader	Notes
Nanopore bedMethyl	`read_bedmethyl_dir()`	`modkit` output, with per-site coverage kept alongside; 5mC and 5hmC read separately
RRBS coverage files	`read_rrbs_dir()`	
Targeted panels	`read_panel()`	Pyrosequencing, EpiTYPER, amplicon. The declared CpG list is checked, not intersected
Olink NPX	`read_olink()`	Proteomics. Already log2 and relative; the scale is left alone
SomaScan RFU	`read_somascan()`	Proteomics, log2 by default because that is what the models were fitted on
Bulk RNA-seq counts	`read_counts()`	Transcriptomics, with the full RLE → log10 → z → YuGene chain
Clinical chemistry	`read_clinical()`	Units declared rather than guessed
ComputAgeBench	`read_computage_bench()`	65 harmonised case/control studies, pinned to a commit

Output

- One score per sample per clock, for as many of the 178 catalogued clocks as your features support
- **A standard error on every score**, propagated from published per-probe reliability through each clock's own weights, and a distribution-free prediction interval
- A coverage figure per clock, and what the probes you are missing cost in *years*
- A run manifest holding the package and registry versions, the SHA-256 of every coefficient file, the array manifest that decoded your IDATs, the reliability table behind the intervals, the device and floating-point precision, the imputation policy, and every warning raised

Execution. One numerical core in Python, called from R through reticulate, on CPU or GPU.

Refusals. A saliva sample scored on a whole-blood clock (measured at 3.83–16.46 years of error in the same people). Cell-free DNA on any array clock. Age acceleration on a pace-of-aging clock. A cohort-centred clock given one sample. Each refusal names the measurement behind it.

Scope and limitations

What the tool is for, what it declines to do, and what it cannot do yet. Three different questions with three different answers, in the order a model card puts them.

Intended use

The reason, rather than the disclaimer. Reliability varies sharply between clocks, and technical reproducibility and biological stability are separate properties that do not track together: a clock can be highly repeatable on split samples and still move with a meal or a night's sleep. So a score that survives replication is not thereby measuring a stable trait, and the two failures need different evidence to rule out. The science of aging clocks sets out, clock by clock, which is established and which is contested.

Out-of-scope use

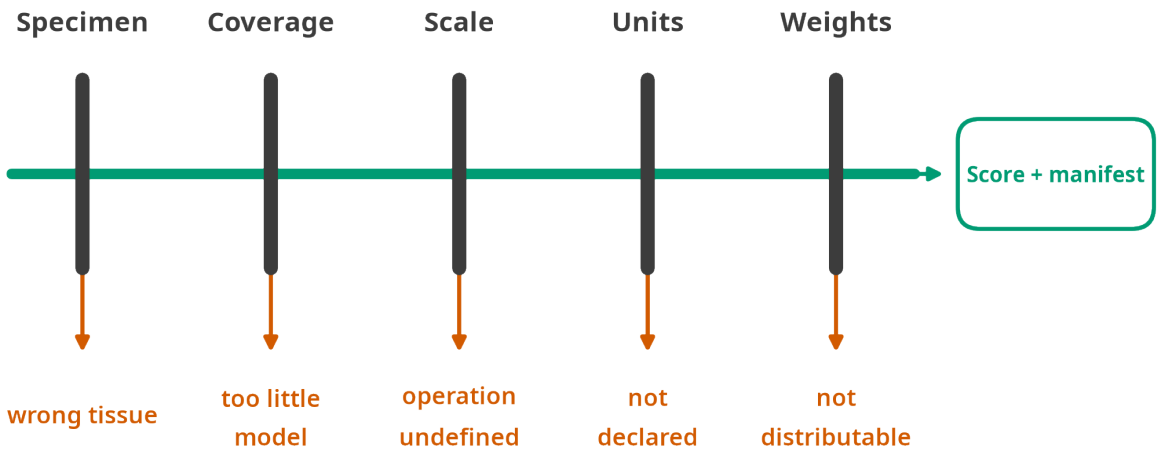


Figure 1: The gates a scoring call passes, and what stops it at each one

Each gate is a place where the wrong answer is available and cheap. A clock scored on the wrong specimen, on too little of its own model, on a scale that does not admit the operation asked for, on clinical values whose units were never stated, or from a weights file that cannot be redistributed, all return a number that looks exactly like a right one. What comes out of the last gate is a score and a manifest recording which gates it passed and on what evidence.

Each of these is a silent wrong answer somewhere else, so each is a hard failure here:

- **Fill a missing probe with zero.** In beta space zero means completely unmethylated, which is a real and extreme measurement. An absent feature takes the clock's reference value: the authors' own for Horvath 2013, DunedinPoAm38, CorticalClock and AltumAge, a healthy adult blood mean for 22 other blood clocks. Where there is none, it takes the mean of the clock's present features, and the run names that fallback with the share of the weight it touched. Every substitution is counted in the manifest.
- **Guess clinical units.** PhenoAge has SI and conventional variants that disagree by years. `units=` has no default.
- **Score below the coverage floor.** A clock whose features are largely absent is skipped and named, rather than scored on mostly-imputed input.
- **Treat probes as interchangeable.** Coverage is measured twice: by count, and by the share of the model's total |coefficient| the present features carry. A clock can clear a 92% feature floor having lost the probes that do a third of the work, and only the second number sees it. `probe_loss()` reports both before you score anything.

- **Claim coverage means validity.** High coverage says the probes are present. It says nothing about whether coefficients fitted in one tissue, species or population transfer to yours; that judgement stays with you, and Choosing a clock lays out the grounds for it.
- **Let a paper-versus-implementation disagreement pass silently.** Eleven clocks in the catalogue are scored from coefficients that differ from what their paper describes: a feature count, a changed statistic, a constant. Each carries the note, and it is raised as a warning at score time, not left on a documentation page.

Known limitations

Said here rather than left to be inferred. The science page §19 and §20 cover each in full, with the algorithms and the reason each is or is not blocked.

- **It does not repair cross-platform probe loss.** It measures it twice over: by count, by coefficient mass, and now in *years* per clock per platform. It cannot undo it. The liftover mapping tables encode concordance measured on paired samples and are not derivable from the array manifests.
- **It has no proteomic or transcriptomic clocks in the catalogue.** The readers and the full preparation chains ship; no entry declares either `data_type`, because every published model in both families is licence-restricted and would enter as a `licensed` scaffold. Scoring a proteomic matrix today refuses with "no clocks to score", which is accurate rather than broken.
- **It does not correct dye bias by default.** The step exists and is opt-in. Measured on real IDATs our implementation moves the median beta by +0.10 to +0.12, because a correct correction needs the normalisation control probes and their addresses are not in the manifest we can fetch.
- **It has no foundation-model backend.** The `NeuralClock` architecture ships, safetensors only. CpGPT and MethylGPT are interesting here for zero-shot imputation and array conversion rather than as another age predictor.
- **105 of the 178 catalogued clocks have no traced coefficient source.** A per-clock literature hunt, not a technical problem, and `primary_source_traced: false` is the honest record of it.

Removed from this list, because they now exist: array normalisation (pOOBAH, noob, BMIQ, probe masks), the raw-IDAT path, single-cell scoring, and the transcriptomic per-dataset centring refusal.

Further reading

- Getting started, install, score, interpret, in both languages.
- Choosing a clock, by the question you asked, by what the scale permits, and by whether the coefficients transfer to your data at all.
- Clock catalogue, all 178, searchable.
- Python reference · R reference
- GPU support, what was verified on real hardware, and why `device="auto"` is CPU.

The whole site is also downloadable as PDF or markdown; see the **Download** menu.

Citation

APA:

```
Hunakunti, B. (2026). FALCONAge: Multiomic biological age and aging clock scoring in Python
and R (Version 1.0.0) [Computer software]. https://github.com/bhagesh-h/FALCONAge
```

BibTeX:

```
@software{falconage2026,
  author  = {Bhagesh Hunakunti},
  title   = {FALCONAge: Multiomic Biological Age and Aging Clock Scoring in Python and R},
  year    = {2026},
  version = {1.0.0},
  url     = {https://github.com/bhagesh-h/FALCONAge},
  license = {GPL-3.0-or-later}
}
```

In R, `citation("FALCONAge")`. The machine-readable version is CITATION.cff, which is what GitHub's *Cite this repository* button reads.

The clock FALCONAge computed for you is somebody else's work, and citing FALCONAge does not cite it. Every registry entry carries its primary reference, so the list for an analysis comes out of the analysis:

Python

```
fa.registry.load().get("grimage2").cite("bibtex")
```

R

```
cite_clock("grimage2", "bibtex")
```

Chapter 2. Install

Installation

Docker is the supported path, and it is the one the rest of this page assumes. One image carries Python, R, the CLI and all 69 bundled clocks at pinned versions, so the same input gives the same numbers on your machine and on a reviewer's. Nothing else to install. Both images are published on Docker Hub and built from the Dockerfiles in this repository. Two commands from nothing to a proven install:

```
docker pull bhagesh/falconage:1.0.0-cpu
docker run --rm bhagesh/falconage:1.0.0-cpu python -m pytest /opt/falconage/src/python/
tests -q
```

The second line proves it by running the test suite inside the image, against the source it was built from.

Or build the same image from a clone, which takes about ten minutes the first time and is then cached:

```
git clone https://github.com/bhagesh-h/FALCONAge.git
cd FALCONAge
docker build -f docker/Dockerfile.cpu -t bhagesh/falconage:1.0.0-cpu .
```

Then either open a session in the image:

```
docker run --rm -it -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu python
docker run --rm -it -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu R
```

Or run a single command without opening one:

```
docker run --rm -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu \
  report betas.csv --outdir results/
```

The image's entrypoint is the `falconage` CLI, so the first argument is a verb: `report`, not `falconage report`. The exceptions are `python`, `R`, `Rscript`, `pytest` and `bash`, which are run as given.

On Windows PowerShell write `"${PWD}"` for `"$PWD"`. Getting started walks the same ground one command at a time, including the GPU image and a **native pip / `remotes::install_github` install** for anyone who already manages their own environment.

If you have never used a terminal, start at the step-by-step walkthrough below instead; it assumes nothing and covers fetching real public data.

Which shell you are typing into

`"$PWD"` means "the folder I am in now", and the spelling differs:

Where you are typing	Use
macOS Terminal, Linux, Git Bash, WSL	`"$PWD"`
Windows PowerShell	`"${PWD}"`
Windows `cmd.exe`	`"%cd%"`

Everything else is identical. A path error on Windows is almost always this.

Native install, if you manage your own environment

```
pip install "falconage @ git+https://github.com/bhagesh-h/FALCONAge.git#subdirectory=
python"
```

```
import falconage as fa
fa.config()
```

```
remotes::install_github("bhagesh-h/FALCONAge", subdir = "r")

library(FALCONAge)
falconage_install()   # builds the Python environment R calls into; a few minutes
falconage_config()    # what resolved: versions, devices, registry size
```

The R package delegates every number to the Python core, so it needs a Python environment either way. On reticulate ≥ 1.41 an ephemeral one is built on first use and `falconage_install()` is optional; use it when you want a pinned, reproducible one.

Every code example on this page runs unchanged in a native install; only the `docker run` prefix goes away.

`falconage_config()` is the first thing to run when anything looks wrong. It reports the interpreter, the registry version, and how many clocks are usable:

```
FALCONAge 1.0.0
  registry     1.1.0 (178 clocks)
  availability 69 bundled - 52 untraced - 57 licensed
  numpy        2.5.1
  torch        not installed (the CPU path needs none)
  devices      cpu
  python       /usr/local/bin/python
```

Complete walkthrough, from an empty machine

Everything above, in order, assuming no Python, no R, and no prior experience with either. Each command can be copied one at a time and each says what it does and what you should see. It ends with a scored public dataset and the figures.

Step 0: install Docker

Install Docker Desktop (Windows, macOS) or Docker Engine (Linux), start it, then run:

```
docker --version
```

You should get a version string. If instead you get "command not found" or "cannot connect to the Docker daemon", Docker is not installed or not running; fix that before going further, because nothing below will work.

If a later command fails with a path error: it is almost certainly the `"$PWD"` spelling for your shell; see the table above.

Step 1: clone the repository

```
git clone https://github.com/bhagesh-h/FALCONAge.git
cd FALCONAge
```

Every later command is run from inside this `FALCONAge` folder.

Step 2: obtain the image

One image carries Python, R, the CLI and all 69 bundled clocks. Pull the published one:

```
docker pull bhagesh/falconage:1.0.0-cpu
```

Or build the same thing from the clone, which takes about ten minutes the first time and is then cached:

```
docker build -f docker/Dockerfile.cpu -t bhagesh/falconage:1.0.0-cpu .
```

Check it either way:

```
docker run --rm bhagesh/falconage:1.0.0-cpu python -c "import falconage; print(falconage.__version__)"
```

Step 3: inspect the corpus before downloading it

A second, deliberately tiny image fetches the test corpus. It is separate on purpose: fetching a fixture is the first thing you do when the package itself is broken, so it must not need the package.

```
docker build -f docker/Dockerfile.testdata -t falconage-testdata:1.0.0 .
```

Now the dry run. **This transfers nothing.** It prints every group, file count and byte total so you can decide before committing to half a gigabyte:

```
docker run --rm -v "$PWD/test/data:/data" falconage-testdata:1.0.0 plan
```

You should see:

```
group          files        size   what it is for
------------------------------------------------------------------------------
bench             14    549.8 MB   ComputAgeBench case/control methylation
clinical           3      5.4 MB   NHANES blood chemistry with mortality linkage
gestational        1     50.8 MB   Umbilical cord blood with gestational age
epicv2             1     41.2 MB   EPIC v2 probe-suffix handling
idat               8     60.4 MB   Raw IDATs, EPIC v1 and EPIC v2
mammalian          4      9.3 MB   Mammalian methylation array, three species
mouse              5     50.2 MB   Mouse blood RRBS across the lifespan
------------------------------------------------------------------------------
selected          36    767.1 MB
```

Step 4: download and verify

```
docker run --rm -v "$PWD/test/data:/data" falconage-testdata:1.0.0 python
docker run --rm -v "$PWD/test/data:/data" falconage-testdata:1.0.0 verify
```

The first fetches 36 files from GEO and Hugging Face into `test/data/` on your own disk, never inside the image, so rebuilding never re-downloads. The second re-computes every SHA-256 against `checksums.sha256` and fails loudly on a single wrong byte.

Only smaller pieces, if 767 MB is too much:

```
docker run --rm -v "$PWD/test/data:/data" falconage-testdata:1.0.0 python --groups clinical
```

> **Fetching the corpus from R**
>
> Not decoration: the R fetcher writes a byte-identical `checksums.sha256`, so the two languages are shown to agree on the input before any clock is scored.
>
> ```
> docker run --rm -v "$PWD/test/data:/data" falconage-testdata:1.0.0 R
> ```

Step 5: score it

Python, CPU

```
docker run --rm -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu \
  python test/run_all.py
```

Every clock against every dataset. It writes one folder per input under `test/output/`, every figure under `test/output_figures/`, and rewrites the generated tables in `test/README.md`, so the numbers in that document are the numbers the code just produced and cannot drift from them.

To check the install without the corpus, run the unit suite instead, seconds, no data needed:

```
docker run --rm -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu \
  python -m pytest python/tests -q
```

R

The same image. R calls the same Python core through reticulate, so these are the same numbers, not a second implementation that agrees approximately.

```
docker run --rm -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu \
  Rscript -e 'testthat::test_local("r")'
```

That suite includes the R-versus-Python conformance tests, which assert equality at tolerance exactly zero.

Scoring from R interactively:

```
docker run --rm -it -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu R
```

```
library(FALCONAge)
falconage_config()
d   <- read_betas("test/data/bench/GSE40279.csv")
res <- score(prepare(d), clocks = "compatible")
summary(res)
```

Python, GPU

Only worth it if you have an NVIDIA card. **Read GPU support first**: for the clocks that ship today the GPU is *slower* than the CPU, which is why `device="auto"` resolves to CPU even inside this image.

Pull the CUDA image. It is 14.6 GB against the CPU image's 2.5 GB, because it carries torch built against CUDA 12.4:

```
docker pull bhagesh/falconage:1.0.0-cuda
```

Or build it:

```
docker build -f docker/Dockerfile.cuda -t bhagesh/falconage:1.0.0-cuda .
```

Check the card is visible inside a container before anything else:

```
docker run --rm --gpus all bhagesh/falconage:1.0.0-cuda nvidia-smi
```

No output or an error here means GPU passthrough is not set up, and every command below will fail. On Linux that is usually a missing `nvidia-container-toolkit`.

```
docker run --rm --gpus all -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cuda \
  python test/gpu_check.py
```

This reproduces every number on the GPU page, device resolution, CPU-versus-GPU agreement, the speed table and the profile that explains it. On a machine with no CUDA device it stops after the first step with an explanation rather than an error, so it is safe to run anywhere.

Scoring on the GPU needs `--gpus all` **and** `--device cuda`. Without the flag, `--device cuda` raises rather than falling back, so a run that asked for a GPU cannot quietly become a very slow success:

```
docker run --rm --gpus all -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cuda \
  score --input test/data/prepared.h5ad --device cuda --outdir results
```

Step 6: score your own data

Put a beta matrix or a clinical table in a folder, mount it read-only, and mount somewhere for the results to land:

```
docker run --rm \
  -v "$PWD/mydata:/data:ro" \
  -v "$PWD/results:/results" \
  -v falconage-cache:/opt/falconage/cache \
  bhagesh/falconage:1.0.0-cpu \
  score --input /data/betas.csv --clocks compatible --outdir /results
```

The named `falconage-cache` volume is worth adding from the start. Nothing bundled needs it; it keeps what FALCONAge downloads on request, the Illumina manifests `fetch_manifest()` caches (12 to 26 MB each) and data fetched with `fa.download`, so a new container does not fetch them again.

Chapter 3. Getting started

FALCONAge scores DNA methylation, clinical chemistry and NMR metabolomics data against a catalogue of 178 published aging clocks. It also reads and normalises proteomic and transcriptomic data, and `fa.fit_clock` fits a clock to them, but the catalogue holds no published clock for either yet. The numerical work happens in one Python core; the R package calls it through reticulate, so an R result and a Python result are the same bits rather than two implementations that agree approximately.

> How to read this page
>
> Blocks with a **copy button** are things to run. Blocks tagged **Output** are what you should see back. Reading them is the point; pasting one into a terminal only produces an error.
>
> ```
> this is a command: copy it
> ```
>
> ```
> this is what it printed
> ```

Quick start

Python

```python
import falconage as fa

data = fa.prepare(fa.read_betas("betas.csv"))
res  = fa.score(data, clocks="compatible")

res.summary()
fa.acceleration(res, method="both")
fa.plot.ba_vs_ca(res, "horvath2013")
fa.report.write_report(res, "report.html")
```

R

```r
library(FALCONAge)

d   <- prepare(read_betas("betas.csv"))
res <- score(d, clocks = "compatible")

summary(res)
acceleration(res, method = "both")
plot_ba_vs_ca(res, "horvath2013")
report(res, "report.html")
```

Both produce the same numbers, because both run the same code. The R package delegates all arithmetic to the Python core through reticulate, and the R suite asserts equality at tolerance exactly zero rather than approximate agreement.

Scoring

Every Python and R snippet from here on runs unchanged in three places: inside the Docker image (`docker run --rm -it -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu python`, or `... R`), in a native install, or in the CLI-free path below. Only the way you open the session differs.

Python

```
d   = fa.prepare(fa.read_betas("betas.csv"))
res = fa.score(d, clocks="compatible")

res.summary()
res.scores               # samples x clocks
res.long()               # one row per sample per clock, with the scale
res.qc()                 # per-clock feature coverage and skip reasons
res.manifest             # versions, device, SHA-256 of every coefficient file
```

R

```
d   <- prepare(read_betas("betas.csv"))
res <- score(d, clocks = "compatible")

summary(res)
as.data.frame(res)                  # samples x clocks
as.data.frame(res, form = "long")   # one row per sample per clock, with the scale
coverage(res)                       # per-clock feature coverage and skip reasons
manifest(res)                       # versions, device, and every coefficient checksum
```

`clocks = "compatible"` scores what this dataset can support and reports the rest as skipped with a reason. Naming clocks explicitly is different: then every one must work, because an explicit request should never be dropped quietly.

Interpreting the result

The scale, the unit, which arithmetic is permitted, both coverage measures, the published reliability where one exists, and any documented disagreement between the clock's paper and the coefficients in circulation, all on the object rather than only on this website:

Python

```
res.interpretation()
```

R

```
interpretation(res)
```

`technical_icc` and `biological_icc` are separate columns because they measure different things and do not track together: the same sample assayed twice, versus the same person sampled again a few days later. A clock can be excellent on the first and poor on the second, and the clocks most used to claim an intervention worked are where that gap is widest.

Chapter 4. The Claude skill

The repository ships a Claude skill at `.claude/skills/falconage/`. It is four markdown files: the workflow (`SKILL.md`), the Docker command set, the clock catalogue routed by the question each clock answers, and every refusal with the measurement behind it.

It is active already for anyone working inside a clone. To have it available in any directory, copy it into your personal skills folder with `cp -r .claude/skills/falconage ~/.claude/skills/`, or on Windows PowerShell `Copy-Item -Recurse .claude\skills\falconage $HOME\.claude\skills\`.

Then ask in plain language. “Score these IDATs and tell me which clocks you refused and why” is enough; the skill supplies the rest.

> **i** Why the skill is checked by CI, and checked harder than the prose
>
> `docs/check_api_docs.py` resolves every `fa.*` name and every `falconage <verb>` in the skill against the running package on each push, exactly as it does for these pages.
>
> The reason it matters more here is that these pages are read by a person, who notices when a function does not exist and goes looking. A skill is read by a model that then *runs* what it says. The first draft named five readers at top level that actually live under `fa.preprocess`, and invented a `falconage probe` verb outright, copied from a specification section of the architecture page that described a verb never built. The check caught all six, and then caught two more invented verbs on that page.

Chapter 5. Loading your data

Nine input types reach the same place: a `FalconData` carrying a samples-by-features matrix, the modality, the platform, and whatever sample annotation came with the file. Once you have one, scoring is the same call whatever you loaded.

Every output on this page was produced by running the code above it, against the public test corpus where the corpus has that input type and against a small file written inline where it does not. Nothing here is illustrative.

What reads what

Your data	Call	Modality
Raw Illumina IDATs	`read_idat_dir()`	`dna_methylation`
A beta matrix, CSV, TSV or parquet	`read_betas()`	`dna_methylation`
A GEO series matrix	`read_series_matrix()`	`dna_methylation`
A ComputAgeBench study	`read_computage_bench()`	`dna_methylation`
RRBS or bisulfite coverage files	`read_rrbs_dir()`	`rrbs`
Nanopore bedMethyl	`read_bedmethyl_dir()`	`dna_methylation`
A targeted assay: pyrosequencing, EpiTYPER, ddPCR	`read_panel()`	`dna_methylation`
Clinical chemistry	`read_clinical()`	`clinical_chemistry`
Olink NPX	`preprocess.read_olink()`	`proteomics`
SomaScan RFU	`preprocess.read_somascan()`	`proteomics`
Bulk RNA-seq counts	`preprocess.read_counts()`	`transcriptomics`

`fa.read(path)` dispatches on the file name when you would rather not choose. It is convenience, not magic: it says which reader it picked, and any argument that reader needs has to be passed to the reader itself.

Raw Illumina IDATs

A directory of `_Grn.idat` and `_Red.idat` pairs, one pair per sample, exactly as the scanner writes them. The array manifest is fetched once and cached.

```
import falconage as fa

d = fa.read_idat_dir("idats/")
d = fa.prepare(d)              # noob, BMIQ, probe masks, id harmonisation
```

```
INFO    fetched ebc36febf212e1d2_HumanMethylationEPIC_manifest_v2.csv.gz (23.7 MB)
FalconData(2 samples x 865918 features, dna_methylation, EPICv1, 2.0% missing)
FalconData(2 samples x 865859 features, dna_methylation, EPICv1, 2.0% missing)
```

The feature count drops by 59 across `prepare()`: those are the probes the published masks exclude, not samples lost. Platform is detected from the manifest and the probe count rather than declared.

EPIC v2 needs one more thing, and it is automatic. Its probe ids carry a suffix (`cg00000029_TC21`), and a clock fitted on 450K sees zero overlap with them until the suffixes are collapsed. `prepare()` does that by default; `aggregate_epicv2=False` turns it off, and there is no good reason to.

A beta matrix

Probes down the rows, samples across the columns, or the transpose: the orientation is detected from which axis carries probe ids.

```
d = fa.read_betas("betas.csv")
```

```
FalconData(3 samples x 3 features, dna_methylation, 0.0% missing)
```

The file that produced it:

```
probe,S001,S002,S003
cg00000029,0.512,0.488,0.559
cg00000165,0.113,0.096,0.140
cg00000236,0.842,0.871,0.836
```

Betas belong in 0 to 1. A matrix of percentages is a factor of a hundred out and every clock will return a confidently wrong number, so `prepare()` checks the range and says so.

A GEO series matrix

The `*_series_matrix.txt.gz` GEO offers for most published studies. Sample metadata comes with it.

```
d = fa.read_series_matrix("GSE330325_series_matrix.txt.gz")
d.obs.head(3)
```

```
FalconData(8 samples x 888987 features, dna_methylation, EPICv2, 0.0% missing)

                title            tissue    cell_type
sample_id
GSM9723838  GSD1a_rep1  Skin Fibroblasts  Fibroblasts
GSM9723839  GSD1a_rep2  Skin Fibroblasts  Fibroblasts
GSM9723840  GSD1a_rep3  Skin Fibroblasts  Fibroblasts
```

The `characteristics:` lines become `obs` columns. Set `obs["tissue"]` before scoring and the specimen check runs; leave it and every clock's tissue assumption goes unchecked.

A ComputAgeBench study

65 curated case-control studies with age, sex, tissue and condition already normalised.

```
fa.io.list_computage_bench().head()
d = fa.read_computage_bench("GSE107143")
```

```
             n platform tissue      conditions  age_min  age_max
dataset
GSE56046   1128     450K  Blood  AS=789, HC=339     44.0     83.0
GSE42861    689     450K  Blood  HC=335, RA=354     18.0     70.0
GSE117859   608     450K  Blood         HIV=608     25.0     75.0
GSE111629   572     450K  Blood  HC=237, PD=335     35.0     92.0
GSE117860   529   EPICv1  Blood         HIV=529     25.0     75.0

FalconData(16 samples x 460295 features, dna_methylation, 450K, 0.0% missing)

             dataset       gpl tissue    cell_type sex   age condition class
SampleID
GSM2861662  GSE107143  GPL13534  Blood  Whole blood   M  79.0        AS   CVD
GSM2861663  GSE107143  GPL13534  Blood  Whole blood   M  88.0        AS   CVD
GSM2861664  GSE107143  GPL13534  Blood  Whole blood   M  88.0        AS   CVD
```

This is the input `run_benchmark()` expects: it needs `condition`, a control group, and `age`, and this reader supplies all three.

RRBS and bisulfite coverage

One file per sample, each a table of positions with methylated and unmethylated counts. Sites are intersected across samples and thresholded on coverage.

```
d = fa.read_rrbs_dir(["20D02.overlap.txt.gz", "20D03.overlap.txt.gz",
                      "M3503.overlap.txt.gz"], min_coverage=5)
```

```
FalconData(3 samples x 1976936 features, rrbs, RRBS, 1.9% missing)
```

Two million shared sites from three files, and the sites are genomic coordinates rather than probe ids. An array clock cannot be scored on them directly; that needs a liftover this package does not ship, and it says so rather than matching on position and hoping.

Nanopore bedMethyl

`modkit` output, one row per site with the modification fraction in column 11.

```
s = fa.read_bedmethyl("sample.bed", sample_id="ONT01", min_coverage=10)
```

```
feature_id
chr1:10468    0.733
chr1:10470    0.120
chr1:10483    0.902
Name: ONT01, dtype: float64
```

One file gives a Series; `read_bedmethyl_dir()` gives a `FalconData` across the sites the files share. The fraction is converted from the percentage the format stores, so what you get back is already a beta.

A targeted assay

Pyrosequencing, EpiTYPER, ddPCR: a handful of CpGs measured directly, with samples down the rows.

```
d = fa.read_panel("panel.csv")
```

```
FalconData(2 samples x 3 features, dna_methylation, targeted panel, 0.0% missing)
```

```
sample,cg16867657,cg09809672,cg02228185
P01,0.68,0.41,0.55
P02,0.72,0.38,0.51
```

A panel almost never carries a whole clock, and the coverage floor is what stops a three-probe file being scored as though it were a 353-probe model. `probe_loss()` says what a partial set costs before you score anything.

Clinical chemistry

The clinical clocks take laboratory values, and they take them in declared units.

```
UNITS = {"albumin": "g/L", "creatinine": "umol/L", "glucose": "mmol/L",
         "crp": "mg/L", "lymphocyte_percent": "%", "mean_cell_volume": "fL",
         "red_cell_distribution_width": "%", "alkaline_phosphatase": "U/L",
         "white_blood_cell_count": "10^9/L"}

d = fa.read_clinical("labs.csv", units=UNITS)
fa.score(d, clocks=["phenoage"]).scores.round(2)
```

```
FalconData(2 samples x 10 features, clinical_chemistry, 0.0% missing)

WARNING [phenoage] Two constants circulate in the divisor of the Gompertz inversion:
0.090165 and 0.09165. FALCONAge uses 0.090165, the value BioAge corrected to in April
2026. Code still using 0.09165 differs by several years on the same input.

           phenoage
sample_id
```

```
C01          55.79
C02          69.61
```

Units are per column, not per file. A clinical table mixes them, so there is no single name to pass; `units="SI"` is refused with the mapping it wants. PhenoAge exists in an SI and a conventional variant that disagree by years, and `units=` has no default for that reason.

HbA1c in mmol/mol. Laboratories outside the US report HbA1c in IFCC units. Declare `"hba1c": "mmol/mol"` and it is converted to the NGSP percentage the reference models were fitted in, by the published master equation NGSP = 0.09148 × IFCC + 2.152 (NGSP; Hoelzel et al., *Clin Chem* 2004;50:166-174, doi:10.1373/clinchem.2003.024802). It is not a ratio, so it applies to HbA1c only.

A cohort measured unlike its reference. KDM and homeostatic dysregulation are fitted on a reference cohort, and `fit_kdm`/`fit_hd` record each marker's distribution there. When a scored cohort leaves it, `score()` says so (category `reference_range`): a median outside the reference's 1st to 99th percentile, which is what a unit mismatch looks like, or many values beyond the reference's range. The case that matters most is CRP. NHANES III's assay read nothing below 0.21 mg/dL, and 63.9% of its adults sit at that floor, so a high-sensitivity CRP cohort scored against it has most of its values where the reference has none. `fa.models.clinical.censor_to_reference(df, ref)` sets values below a reference's reporting floor to the floor, which is how the reference itself recorded them, and returns how many moved. It only touches markers whose minimum is a floor (at least 1% of reference rows sitting exactly at it); values above the reference's range are real measurements and are left alone.

Olink NPX

Olink exports long: one row per sample per assay. That is detected and pivoted.

```
from falconage import preprocess as pp

d = pp.read_olink("olink_npx.csv")
```

```
INFO    read_olink: olink_npx.csv is Olink's long export (SampleID, Assay, NPX);
        pivoted to 2 sample(s) x 2 assay(s)
FalconData(2 samples x 2 features, proteomics, olink, 0.0% missing)
```

A wide matrix of samples by assays is read as it stands. Values stay in NPX, which is already log2 and is the scale the coefficients were fitted in.

For SomaScan, `pp.read_somascan()` takes RFU and logs it by default, because every published SomaScan aging model here was fitted on logged RFU and feeding it linear values is an order-of-magnitude error that looks like a plausible number.

NMR metabolomics

A Nightingale Health export (CSV, TSV or Excel), samples down the rows:

```
d = fa.read_nightingale("nightingale.xlsx")      # Excel needs the `metabolomics` extra
(openpyxl)
res = fa.score(d, clocks=["metaboage", "metabohealth"])
```

Column names are translated to the BBMRI-NL names the two scores were fitted with, by MiMIR's table and MiMIR's rule (lower-cased, and translated only when exactly one entry lists the name); a name the table does not know is kept and listed in `obs.attrs["untranslated"]` rather than guessed. MetaboAge (van den Akker et al. 2020) and MetaboHealth (Deelen et al. 2019) are scored as MiMIR scores them, its quality control included, and match it to 1e-9 and 1e-12. MetaboHealth z-scores each measure within the cohort, so a score depends on who else is in the file: it is marked `requires_cohort` and is not comparable between files scored separately.

Bulk RNA-seq counts

Genes down the rows, samples across, raw counts rather than TPM.

```
d = pp.read_counts("counts.csv")
d = pp.prepare_transcriptomic(d)   # RLE, log10, per-sample z, YuGene
```

```
FalconData(2 samples x 3 features, transcriptomics, 0.0% missing)
```

The preparation chain ships and is tested. No transcriptomic clock has a registry entry yet: tAge's models are licence-restricted, which is a rights problem rather than a technical one, and the chain is here so that supplying one is all that stands between you and a score.

After loading

Whatever you loaded, the rest is the same:

```
res = fa.score(d, clocks="compatible")   # only the clocks this data supports
res.interpretation()                     # scale, unit, legal operations, caveats
fa.technical_se(res, d)                  # how much of each score is the assay
```

`clocks="compatible"` is the safe default: it asks the registry which clocks this platform, modality and specimen actually support, and skips the rest by name rather than scoring them on whatever features happen to match.

Chapter 6. Preprocessing, imputation and coverage

6. Preprocessing, imputation, missing features

6.1 The canonical input contract

Rows = samples, columns = features. This is the pyaging convention and it should be the new tool's too. Feature names must be the exact identifiers the clock coefficients use: `cg00000029` for human methylation, Ensembl or Entrez for expression, UniProt/Olink assay names for proteins.

pyaging wraps this in AnnData: `adata.X` is the feature matrix, `adata.obs` holds sample metadata and later the predictions, `adata.uns` holds per-clock metadata and missingness diagnostics, `adata.obsm["X_<clock>"]` is the transient per-clock feature slice (pyaging _pred.py).

6.2 EPIC v2 probe aggregation

EPIC v2 column names look like `cg00000029_TC21`. Split on _, group by CpG prefix, and average duplicated probes before anything else (pyaging `epicv2_probe_aggregation`). Skipping this means every EPIC v2 clock silently sees zero overlapping features.

6.3 Two distinct imputation stages

Stage 1 - dataset-level, before the clock. `df_to_adata(imputer_strategy=...)` with `mean`, `median`, `constant` (0), or `knn` (default). KNN is expensive on wide matrices and is the first thing to move to GPU.

Stage 2 - clock-level, for features the dataset does not contain at all. pyaging preallocates the clock's feature matrix, fills present features by index lookup, and fills absent features with the model's `reference_values` (typically the training-set mean or a gold-standard median) or with zero if no reference vector was shipped (pyaging `check_features_in_adata`). It records `adata.uns["<clock>_percent_na"]` and `adata.uns["<clock>_missing_features"]`, and raises if 100% of features are missing.

Filling with zero in β space means "fully unmethylated", which is biologically wrong for most sites, always ship a reference vector.

Third stage in tAge: pad missing genes with NA and let the model's own sklearn `SimpleImputer` insert training medians. This is the cleanest of the three because the imputation statistic travels with the model.

What FALCONAge does, and what the R packages do. For a linear clock FALCONAge fills an absent feature with the mean of the clock's observed features over the whole dataset: one constant, the same for every absent feature and every sample (`models/linear.py`). `imputation="reference"` replaces it with a per-feature value wherever a reference is supplied. The mitotic mean, percentile and transmission clocks (epiTOC1/2/3, HypoClock, EPICmitHyper/Hypo, stemTOC, stemTOCvitro) are the exception: their published statistic is taken over the sites present, so FALCONAge leaves absent sites out and records `imputation: "excluded"`. methylclock 1.16.0 leaves absent CpGs out of the sum (`predAge` in `R/internals.R`), which is a fill of 0. dnaMethyAge 0.2.0 fills each one with its per-probe mean from the package's `golden_ref` table (`meanImputation`, called from `methyAge`). Each policy adds the same amount to every sample, so on one dataset the packages differ by a constant

```
Δ = Σ_{j absent} w_j (f_j − g_j)
```

where `w_j` is the coefficient and `f_j`, `g_j` are the two fills. For Horvath's clocks the constant is multiplied by 21 above the adult age of 20, where the anti-log transform is linear.

Measured on 858 EPIC v1 whole-blood samples from one cohort, on QC-filtered betas, FALCONAge minus each package:

Clock	Absent CpGs	vs methylclock 1.16.0	vs dnaMethyAge 0.2.0	After matching the fill
DNAm PhenoAge	13	−0.50 y	−2.87 y	0.0000 y
Horvath 2013	27	−8.61 y	−6.86 y	0.0000 y
Hannum	9	+11.57 y	−1.39 y	0.0000 y
Skin & Blood	12	−2.01 y	not run	0.0000 y

With the absent CpGs filled as the other package fills them, FALCONAge reproduces every sample's score from each package it was compared with, to four decimal places. The coefficients and intercepts agree (DNAm PhenoAge's 60.664 is in all three), and the whole offset is the fill. The same betas can therefore put Hannum ages 13 years apart in two established packages with no disagreement about the clock. The offsets belong to a dataset's missing CpGs, not to the clocks, so the registry does not carry them as per-clock discrepancies. The coverage record (`res.coverage[clock_id]`) gives the number of values filled (`n_imputed`) and the method (`imputation`). Neither 0 nor a pooled mean is a published value for a CpG, which is why a per-CpG reference from each clock's own training data is the better fill.

Horvath's normalisation is optional in his own code, and the packages default differently. Horvath 2013 supplied a calibration step (Additional file 24, `BMIQcalibration`): each sample's betas over 21,368 probes are mapped onto the distribution of `goldstandard2`, the per-probe means in Additional file 22 (Genome Biology, CC BY 2.0), after absent probes are set to those means. dnaMethyAge 0.2.0 runs it by default for `HorvathS2013` (`simple_mode = TRUE` skips it), methylclock 1.16.0 does not (`normalize = FALSE`), and FALCONAge offers it as an opt-in step, `fa.preprocess.horvath_normalise()` (R: `horvath_normalise()`). Run unmodified on 120 of the same EPIC v1 blood samples, it lowered Horvath 2013 age by a mean 1.87 years (SD 1.15 across samples, −4.6 to +3.3) and left the correlation with chronological age at 0.89; raw and normalised ages correlated at 0.9993. The step moves the level and barely the ranking, so a comparison across packages has to say whether it ran.

`horvath_normalise()` is a port of `BMIQcalibration`. The R code fits its beta mixtures on 20,000 probes drawn by `set.seed(1); sample(...)` and stops its optimisers early (Nelder-Mead after 50 evaluations), so its result depends on the exact draw and the exact optimiser path; the port reproduces each piece as R computes it (the Mersenne-Twister draw, `optim`'s Nelder-Mead and BFGS, RPMM's `blc`, `density()`). Against the unmodified Additional file 24 it agrees to 3e-10 in beta on a synthetic fixture and to 2e-8 years of Horvath age on the 120 samples above, reproducing the −1.87-year shift. Two arguments follow the code people actually ran:

Argument	Default	The other value
`absent`	`"drop"`: calibrate on the gold-standard probes the data carry, as Knight et al.'s published wrapper does	`"fill"`: set absent probes to their gold-standard value first

Argument	Default	The other value
`sample_kind`	`"Rejection"`: R 3.6.0 and later	`"Rounding"`: the draw of earlier R

With `sample_kind="Rounding"`, Knight et al.'s own test dataset gives 37.3665, 38.3463 and 39.322 weeks against the 37.366, 38.346 and 39.324 their instructions state; the remaining 0.002 is their k-nearest-neighbour imputation of 51 missing values. Only the 21,368 gold-standard probes are changed, and a missing value is set to its gold-standard value first, as Horvath's code does for a single sample; both counts are recorded in `uns["horvath_normalisation"]`.

6.4 Imputation options used for benchmarking

ComputAge exposes three and treats the choice as part of the model config (ComputAge benchmarking.py:323):

- `none` - reindex and fill 0.
- `sesame_450k` - fill with the sesame 450K per-probe median.
- `average` - fill with the column mean of the current dataset (0 if the column is empty).

Reporting the imputation method alongside a score is not optional; it changes the answer.

6.5 Preprocessing steps that live inside a model

These are per-clock and must run *after* feature selection, not before:

Step	Clocks
robust scale `(x − median)/std`	AltumAge, CpGPTGrimAge3
standard scale `(x − mean)/std`	CpGPTPCGrimAge3
per-sample z-score across features	ZhangEN, ZhangBLUP
quantile normalise to gold standard	DunedinPACE, Stubbs
quantile normalise then scale	Stubbs
TPM by feature length then log1p	OcampoATAC1/2
binarise at per-row non-zero median	BiTAge
median-fill then per-sample average rank	Pasta, PastaMouse, REG
mean over non-missing (−1 sentinel)	epiTOC1, HypoClock, epiCMIT
0.95 quantile over non-missing	stemTOC, stemTOCvitro
nan ⟶ 0	epiTOC2, epiTOC3
autosomal z-score	XChrom, YChrom
NaN ⟶ reference value	~90 clocks (the `LinearReferenceClock` base)

Platform coverage

Worth running once on an array you have not used before. It reports, per clock, how many features are present **and** how much of the model's weight they carry. Two different numbers, and the second is the one that catches EPIC v2 probe loss:

Python

```
fa.probe_loss(d, clocks="scoreable")
```

R

```r
probe_loss(d, clocks = "scoreable")
```

```
                     tier  n_features  n_present  coverage  mass_coverage  bias_years
heaviest_absent
horvath2013      bundled          353        339    0.9603         0.9715     -1.2570
cg11932564 (0.8%), ...
hannum           bundled           71         64    0.9014         0.8994      1.0279
cg21139312 (2.7%), ...
zhangmortality   bundled           10          9    0.9000         0.7687      0.0124
cg08362785 (23.1%), ...
```

Three rows of the table for the EPIC v2 series in the test corpus (GSE330325). `zhangmortality` has nine of its ten CpGs, 90% by count, and the one it lacks carries 23% of its weight.

`bias_years` is filled in when the platform is one the shift has been measured on; see what probe loss costs. `NaN` there means *not measured*, never *no bias*.

A clock at 96% feature coverage and 61% weight coverage has kept nearly all its probes and lost the ones that matter. The scoring floor applies to both measures, so this run would be refused with a message naming the heaviest absent probes rather than returning a confident number.

Chapter 7. Raw arrays to betas

Raw IDATs

New in this release. Before this you had to normalise in sesame or minfi and bring a beta matrix back.

Python

```
d = fa.read_idat_dir("idats/")        # every Grn/Red pair in the folder
res = fa.score(fa.prepare(d), clocks="compatible")
```

R

```
# The raw-array chain is Python-only for now; call it through reticulate.
fa <- reticulate::import("falconage")
d  <- fa$read_idat_dir("idats/")
```

The chain runs detection first (pOOBAH), then background correction (noob), because pOOBAH's null *is* the uncorrected out-of-band signal. Undetected probes become `NA` rather than a number. The array manifest is downloaded from Illumina's public bucket the first time and cached. It is the one step in FALCONAge that needs a network.

```
FalconData(2 samples x 865918 features, dna_methylation, EPICv1, 2.0% missing)
```

20. From raw arrays to betas

Also new here, and the other half of it. Section 20 is about what a score is worth; this is about being able to produce one from what a laboratory actually has.

At first FALCONAge read beta matrices and told you to normalise elsewhere. That was accurate and it was the single largest reason somebody with their own IDATs could not use the package.

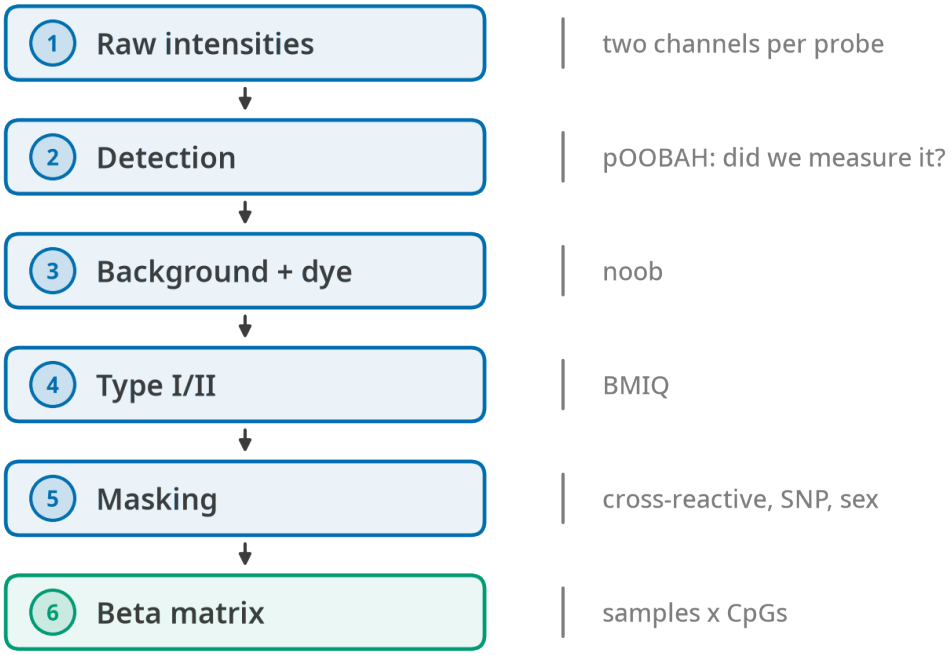


Figure 2: The chain from two channels of raw intensity to a beta matrix

Each step answers a question the next one assumes. Detection asks whether a probe was measured at all, background and dye correction asks what the intensity would have been without the array's own offset, and BMIQ asks how to put two probe chemistries on one scale. Masking removes probes that are known to be unreliable rather than merely absent. Skipping a step does not fail; it produces a beta matrix that is quietly on a different scale from the one the clock's coefficients were fitted on.

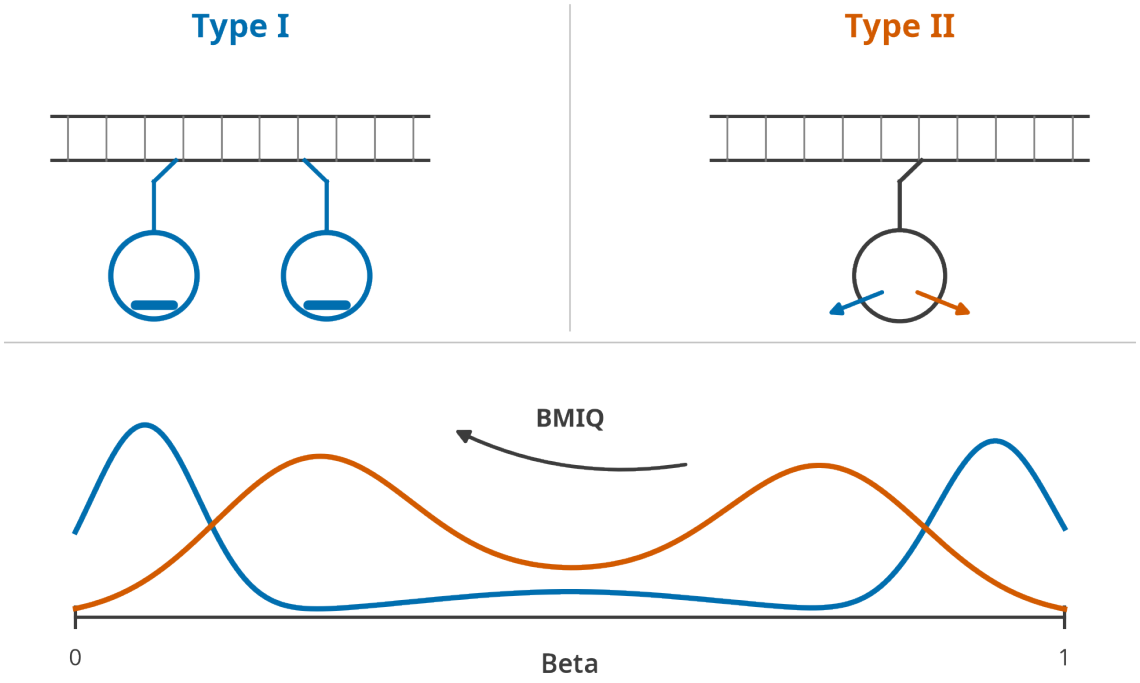


Figure 3: Why the two probe designs need reconciling before a coefficient means anything

A type I probe uses two bead addresses and one colour channel; a type II probe uses one address and both channels. The two designs give measurably different beta distributions, with type II compressed toward the middle of the range, which is what BMIQ corrects by mapping type II onto the type I reference. On EPIC arrays type II probes are the large majority, so leaving this uncorrected shifts most of the matrix a clock will read.

20.1 What an IDAT does not contain

An IDAT is keyed by **bead address**, not by probe. A type I probe has two addresses and reads both in one colour; a type II probe has one address and reads methylated in green, unmethylated in red. Which colour a type I probe uses is a property of its chemistry and is written down nowhere in the file.

So the first thing the chain needs is the array manifest, and Illumina's licence does not permit redistributing it. `methylprep` (MIT) solved this by downloading from a public S3 bucket and caching; FALCONAge uses the same bucket, the same filenames, and its own `download.fetch`. One network round trip per machine, ever. Every other path in the package works offline and this one says so.

The manifest's SHA-256 goes into the run manifest. A beta matrix is a function of which manifest resolved its addresses, in exactly the way a score is a function of which coefficient file produced it, and recording one and not the other would be an odd place to stop.

20.2 Out-of-band signal, which everything downstream rests on

A type I probe's *other* colour, at its own two addresses, measures nothing real. It is the array's background, sampled on this chip, in this run, at tens of thousands of addresses. That is what makes it a usable null, and it is why type II probes, which have no out-of-band signal, borrow the type I null.

Two steps use it and they must run in this order:

pOOBAH (Zhou et al. 2018) asks, per probe, the probability that a signal this bright could come from background alone; read straight off the empirical out-of-band distribution in the relevant channel. A probe above 0.05 becomes `NaN`, because a beta of 0.43 computed from two noise readings is indistinguishable from a real 0.43 once it is in the matrix.

noob (Triche et al. 2013) then deconvolves the observed signal into a normal background plus an exponential true signal, per channel, using the same out-of-band values.

Detection **before** correction, and `noob()` raises if asked to run first. The reason is not stylistic: pOOBAH's null is the *uncorrected* out-of-band distribution, and correcting first subtracts the background from the very numbers that define what background looks like. Every probe then clears a bar that has been lowered underneath it.

Two numerical points worth recording because both were wrong in a first draft:

- The background location and scale use **Huber's M-estimator**, as minfi does. The upper tail of the out-of-band signal is cross-hybridisation, not background. On the corpus's EPIC v1 arrays a plain mean of the contaminated green channel reads about three times the robust estimate.
- The normal-exponential posterior is evaluated through `log_ndtr`. Written directly as a density over a tail probability it is `0/0` for any probe well below background, which is *every undetected probe on the array*, not an edge case.

20.3 The validation, and why it is a real one

The corpus holds raw IDATs for GSM5548192 and GSM5548193, and the betas their authors published for those same two physical samples. The two paths share nothing.

correlation	**r = 0.99928**
median absolute difference	**0.0113**
probes within 0.05	**99.7%**
probes compared	809,400

That is a check on the address decoding, the type I/II split and the channel assignment, where a raw-array reader actually goes wrong, and where an error would be a whole-matrix error rather than a subtle one.

It is asserted on the **uncorrected** betas, deliberately. GSE182991′s published matrix agrees with our raw decode at r = 0.999, which says its own processing was minimal; comparing our background-corrected output against it would be asserting that a correction does nothing.

20.4 Two steps that ship switched off, and why

Dye bias. Green and red do not fluoresce equally, and a type II probe reads methylated in one and unmethylated in the other, so it carries a tilt that is not methylation. The standard fix matches the two channels using type I probes.

Measured on real IDATs, our implementation of it moves the median beta by **+0.10 to +0.12**: far more than a dye correction should. The cause is visible in the data: on that chip red runs about twice as hot as green (median out-of-band 1,171 against 415), so mapping red onto green halves every type II unmethylated signal.

That is a limitation of the *assumption*, not an arithmetic slip. Matching the two channels' distributions presumes type I green probes and type I red probes measure the same underlying quantity; on this array their median betas are 0.05 and 0.14. A correct correction needs the normalisation control probes, whose

addresses are not in the core-columns manifest we can fetch. So it is left in, opt-in, and documented. Turned on by default it would silently make results worse.

BMIQ. Implemented in full, a three-state beta mixture per probe type, fitted by EM, then a quantile map within each state (Teschendorff et al. 2013). It has to be asked for because clocks differ in how much they depend on it, and type I probes come out untouched, so BMIQ never moves a clock built purely on them.

20.5 Probe masks: standard for an EWAS, a decision for a clock

Zhou, Laird and Shen catalogued the probes that do not measure what their name says, multi-mapping, common SNP at the interrogated base, repeats, polymorphic extension base, and publish a recommended general mask. On 450K it flags 29,504 probes, six percent of the array.

Masking them is correct for an EWAS. **Scoring a pre-fitted clock is a different situation**: every clock in this registry was trained on unmasked data, before most of these masks existed, and its coefficients were fitted with those probes in, whatever the probes were measuring. Removing them at score time does not recover a truer number: it deletes an input the model expects and hands the gap to the imputer.

So nothing is masked automatically, and `mask_report()` prices the decision:

clock	probes masked	share of model	share of |coefficient|
dunedinpoam38	5 / 46	10.9%	**6.1%**
dnamtl	15 / 140	10.7%	5.4%
hannum	1 / 71	1.4%	1.6%
horvath2013	**1 / 353**	0.3%	**0.2%**

Two clocks, the same published mask, and opposite answers. For Horvath it is free; for DunedinPoAm38 it removes a ninth of the model. The coefficient-mass column is the one to read, and it is why this is a report rather than a step.

20.6 What the chain still does not do

- **No control-probe normalisation.** The core-columns manifest carries no control addresses, which is why dye bias is unvalidated (§20.4).
- **No cross-platform liftover.** Unchanged from v1.0, and blocked for the same reason: mLiftOver's tables encode empirical paired-sample concordance.
- **EPIC v1 and v2 cannot be told apart by address count.** They are five percent apart, so detection refuses as ambiguous rather than picking the closer one. Pass `platform=` for a chip near the boundary.

Chapter 8. Choosing a clock

There is no best clock. There is a clock that answers the question you asked, and a much larger number that answer a different one in the same units.

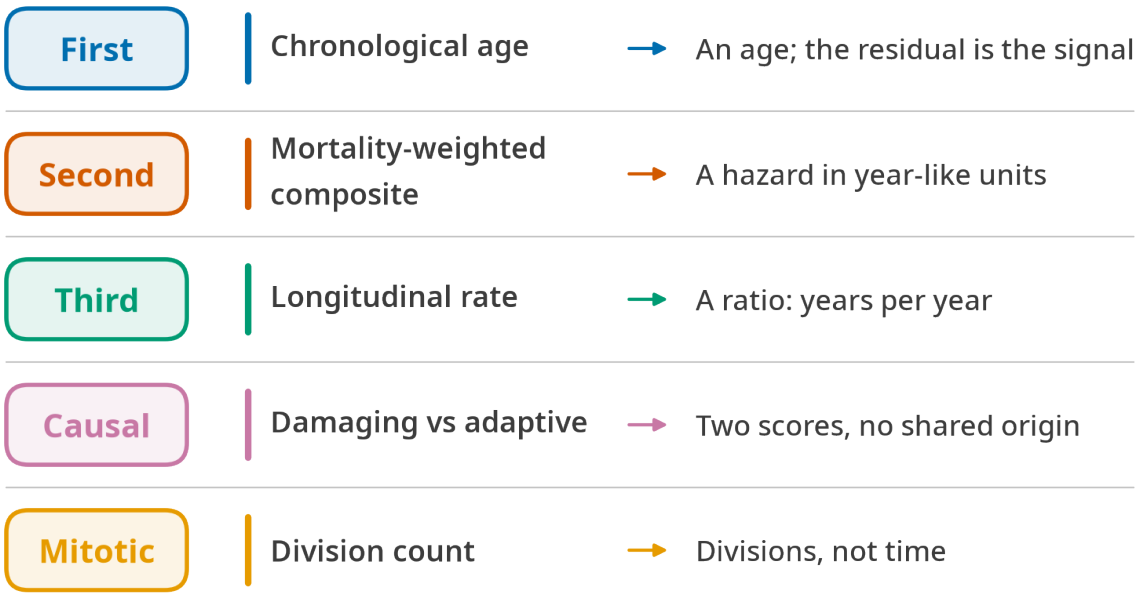


Figure 4: Five generations of clock, and what the output of each one actually is

The generations are not successive improvements on one measurement. Each was trained against a different target, so each returns a different quantity: a predicted age whose signal is the residual, a mortality composite rescaled into year-like units, a rate, a pair of scores with no shared origin, and a division count. A later generation is not a better answer to the earlier question, and the first generation is still the right choice when the question really is how old a sample looks.

Choosing by research question

Question	Ready to score	Needs a licensed file	No coefficients traced	Scale
How old does this sample look?	altumage, cage, corticalclock, hannum, horvath2013, intrinclock, lin, meer, and 11 more	pchannum, pchorvath2013, pcskinandblood, retroclock, systemsag eageprediction	abec, cabec, ctsliver, eabec, encen100, encen40, ensemblea gehumanmouse, gliasin, and 20 more	age_years, relative_score
How old is this newborn, gestationally?	bohlin, knight, leecontrol, leerefinedrobust, leerobust, mayne	—	epicga	gestational_weeks
How fast is this person aging?	dunedinpoam38	dunedinpace	ensemblea gestatic, ensemblea gestatictop	age_years, pace_ratio
Which organ system is aging fastest?	—	systemsage, systemsag eblood, systemsag ebrain, systemsag eheart, systemsag ehormone, systemsag eimmune, systemsag einflammation, systemsagekidney, and 4 more	dnamfitage	age_years
Who is at risk of dying sooner, or is frailer?	dnamfili, dnamic, dnamphenoage, hd, hrsinchphenoage, kdm, linage2, metabohealth, and 2 more	cpgptgrimage3, cpgptpcgrimage3, dnamemrage, grimage, grimage2, omicmage, pcgrimage, pcphenoage	fiage, mammalianlifespan, senmortalityage	age_years, mortality_log_hazard, relative_score
Is damage separable from adaptation?	yingadaptage, yingdamage	—	—	age_years_relative
How much has this tissue divided?	cellpopage, epicmithyper, epicmithypo, epitoc1, epitoc2, epitoc3, hypoclock, replitali, and 2 more	—	replitalinorm, senctureage	divisions, relative_score
What is the blood's cell composition?	deconvolutebloodepicbcell, deconvolutebloodepiccd4tcell, deconvolutebloodepi	twelvecelldeconvolu	—	proportion

Question	Ready to score	Needs a licensed file	No coefficients traced	Scale
	ccd8tcell, deconvolutebloodepicmonocyte, deconvolutebloodepicneutrophil, deconvolutebloodepicnkcell	tebloodepicbas, twelvecelldeconvolutebloodepicbmem, twelvecelldeconvolutebloodepicbnv, twelvecelldeconvolutebloodepiccd4mem, twelvecelldeconvolutebloodepiccd4nv, twelvecelldeconvolutebloodepiccd8mem, twelvecelldeconvolutebloodepiccd8nv, twelvecelldeconvolutebloodepiceos, and 4 more		
How long are the telomeres?	dnamtl	pcdnamtl	—	telomere_kb
What is this person exposed to, or how do they live?	dnamstress, mccartneyalcohol, mccartneybmi, mccartneybodyfat, mccartneyeducation, mccartneyhdlcholesterol, mccartneyldlcholesterol, mccartneysmoking, and 3 more	grimage2packyrs, pcgrimagepackyrs	dnamfitagegaitf, dnamfitagegaitm, dnamfitagegripf, dnamfitagegripm, dnamfitagevo2max, reedbmi	gestational_weeks, relative_score
What is a specific protein or lab value likely to be?	downsyndrome	grimage2adm, grimage2b2m, grimage2cystatinc, grimage2gdf15, grimage2leptin, grimage2loga1c, grimage2logcrp,	bocklandt, compil6, garagnani	relative_score

Question	Ready to score	Needs a licensed file	No coefficients traced	Scale
		grimage2pai1, and 8 more		
Is a named disease more likely?	depressionbarbu, prostatecancerkirby	—	adbahadosingh, cvdwesterman, hepatoxu	relative_score
What is the sample's chromosomal sex, or its species?	—	—	mammalianfemale, xchrom, ychrom	relative_score

Every one of the 178 catalogued clocks appears above, routed by its declared `predicts` field. The full table with paper links is the clock catalogue.

"Ready to score" means the coefficients ship in the package. The rest are catalogued and need a file; see Availability below.

Python

```
reg = fa.registry.load()
reg.filter(generation="second", availability="bundled")
reg.compatible_with(d)
```

R

```
list_clocks(generation = "second", tier = "bundled")
compatible_clocks(d)
```

Scale type and permitted operations

Every clock carries a `scale_type`, and it decides which downstream operations are defined.

Scale	Clocks	Permitted	Refused, and why
`age_years`	80	acceleration, correlate, difference, mean, residual	—
`relative_score`	53	correlate, rank	everything except correlation and rank: there is no external unit to difference or average
`proportion`	18	compositional, correlate, difference, mean	anything ignoring the sum-to-one constraint
`divisions`	10	correlate, difference, mean, rank	acceleration: a division count is not elapsed time
`gestational_weeks`	8	acceleration, correlate, difference, mean, residual	mixing with an age in years
`mortality_log_hazard`	3	correlate, hazard ratio, mean, rank	acceleration: a log-hazard has no zero on the age scale
`telomere_kb`	2	acceleration, correlate, difference, mean	comparison with an age in years; and higher is the *younger* direction here, unlike every age clock beside it
`pace_ratio`	2	correlate, difference, mean, rank	acceleration: it is already a rate
`age_years_relative`	2	correlate, difference, mean, residual	acceleration: the clock tracks age with a slope near one but its origin moves between cohorts, so predicted minus chronological measures the cohort

`acceleration()` enforces this. Naming a clock whose scale does not support it is an error; leaving the clock list empty quietly excludes them.

Coverage and validity

A clock trained on 450K runs perfectly well on EPIC data that carries its probes, and fails on 450K data filtered to 20,000 probes. Only the feature list can answer whether a clock is usable, which is what `compatible_clocks()` checks.

But high coverage does not make a result meaningful. The mammalian methylation array was designed around CpGs conserved across mammals, and carries 96% of Horvath2013's 353 probes, so a zebra scores at *higher* coverage than many human 450K datasets and returns a confident number from a clock fitted on people. Nothing in the arithmetic can notice. Set `species` and FALCONAge warns:

```
d <- falcon_data(betas, modality = "dna_methylation", species = "Equus grevyi")
res <- score(d, clocks = "horvath2013")
manifest(res)$warnings
#> trained on Homo sapiens, scored on Equus grevyi. Feature coverage says nothing
#> about whether the coefficients transfer across species.
```

Feature count and coefficient mass

Coverage has two forms and they disagree exactly when it matters: how many of a clock's features are present, and what share of the model's total |coefficient| those present features carry. Check both before committing to a clock on a given array.

```
fa.probe_loss(d, clocks="scoreable")
#> horvath2013   coverage 0.9632   mass_coverage 0.6117   cg16867657 (14.1%), ...
```

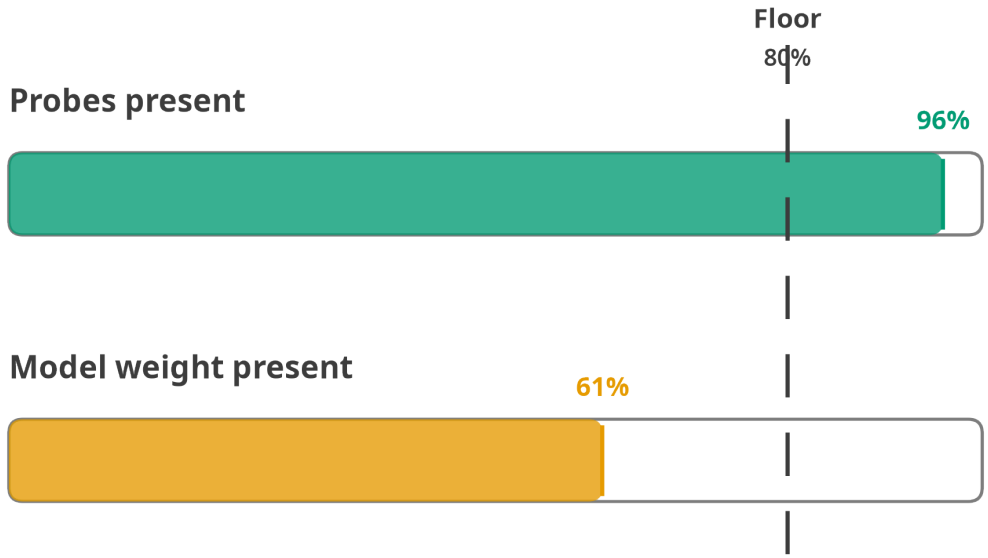


Figure 5: Horvath 2013 on one EPIC v2 array: probes present against model weight present

Ninety-six per cent of probes present, sixty-one per cent of the model. The scoring floor applies to both, so this is refused with the heaviest absent probes named rather than scored into a confident number. The gap opens because elastic-net coefficients are far from uniform: a handful of probes carry a disproportionate

share of the total weight, and losing four per cent of the probes loses thirty-five per cent of the model when the missing four per cent are those.

→ The science, §19.5 prices the same loss in years, per clock per platform, and explains why the PC clocks barely move where first-generation clocks shift by double digits.

Reliability

"Reliable" means two different things, and they do not track together:

	What is repeated	What it measures
Technical ICC	the same DNA, assayed twice	the assay, and the model's sensitivity to array noise
Biological ICC	the same person, sampled again days later	whether the number is a property of the person or of the morning

Nearly every clock is technically excellent; biological ICCs are substantially lower. **GrimAge2 and DunedinPACE are among the most biologically fragile**, and they are the two most often used to claim an intervention worked. `res.interpretation()` carries both columns, with `None` meaning "not established" rather than "fine".

The rule: measuring change within a person, prefer a clock with a published biological ICC, or a principal-component version. → The science, §17.7.

Measure it on your own cohort

A published ICC is somebody else's laboratory. Measure yours:

```
se = fa.technical_se(res, data)
se.diagnostics[["median_se", "se_over_sd", "implied_cohort_icc"]]
```

`se_over_sd`
: measurement error as a fraction of the spread between your own samples. Above 1, the clock cannot distinguish two of your samples however different their point estimates look.

`implied_cohort_icc`
: the reliability *this* cohort implies. Often below the published figure, because a narrow age range leaves less between-sample variance for the same assay noise.

For scale: on this package's test corpus Horvath 2013 comes out at ±1.58 years, DunedinPoAm38 is least repeatable at 0.72, and the 319,607-probe BLUP clock reaches 1.00. With technical duplicates, `fa.icc_from_replicates(data, "subject")` replaces the published per-probe table with your own plates. → The science, §19.2 for the propagation itself.

Then ask how many samples you need

```
fa.power("horvath2013", effect=0.5, result=pilot)
```

At ICC 0.9 a tenth of the sample size exists only to average out the instrument, and the function says so. It refuses to default the standard deviation, because n scales with its square.

Specimen type

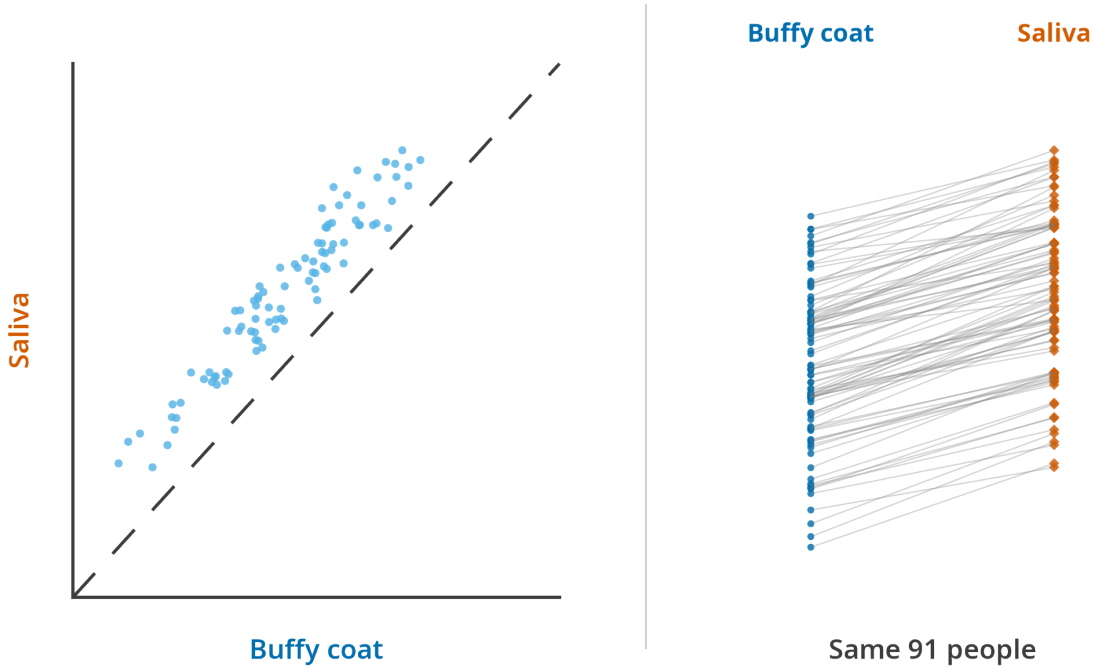


Figure 6: The same 91 donors, scored from saliva and from buffy coat

Saliva clock ages ran **3.83–16.46 years above buffy coat in the same 91 people**, while still correlating with them at Spearman 0.45–0.69. A correlation check passes and the output is a decade out, because correlation is not agreement. The left panel is the whole argument: every point sits above the line of identity, which is a bias, and the scatter around it stays tight, which is what a correlation coefficient reports. The paired lines on the right show the same donors shifting in the same direction, so no amount of averaging removes it.

The rule: set `obs["tissue"]` on every dataset. FALCONAge then compares it against what each clock was fitted on. Pan-tissue clocks say nothing, most warn with the measured discordance, and twelve refuse outright. Cell-free DNA is refused by every clock. No `tissue` column gets one warning saying the check did not run.

→ The science, §19.4 for the twelve refusals and the measurements behind each. Reliability, technical against biological, is [39].

Availability

Availability	n	What you do
`bundled`	69	Nothing. Coefficients ship inside the package, or the clock is a formula with none.
`untraced`	52	Catalogued, but no primary source has been established, so no coefficients ship.
`licensed`	57	Obtain a coefficient file and register it. The architecture is implemented and tested.

```
list_clocks(tier = "C")     # the ones needing author permission
list_clocks(untraced = TRUE)
```

The distinction between B and C matters. C is a licensing decision by the authors. B is an honesty one by us: copying a coefficient set out of another package without knowing where that package got it is how the known paper-versus-implementation discrepancies spread in the first place, and a traced extractor is a welcome contribution.

The clock catalogue

Python

```
reg = fa.registry.load()
reg.filter(availability="bundled")
reg.search("mortality")
reg.get("horvath2013")
reg.get("dnamphenoage").cite("bibtex")
```

R

```
list_clocks(tier = "bundled")
list_clocks(search = "mortality")
clock_info("horvath2013")
cite_clock("dnamphenoage", "bibtex")
```

Forty clocks are `licensed` scaffolds: the model, the feature list, the transform chain and the expected shapes all ship, but the coefficients are research-use-only or have no public source FALCONAge may redistribute. Asking about one says why, where to obtain a file, and which open clocks answer the same question.

Python

```
fa.registry.register_local_weights("grimage2", "~/licensed/grimage2_coefs.csv")
fa.score(d, clocks=["grimage2"])
```

R

```
register_local_weights("grimage2", "~/licensed/grimage2_coefs.csv")
score(d, clocks = "grimage2")
```

Chapter 9. What a score means

The clock catalogue

178 clocks, in three availability groups. Availability is about redistribution rights, not about whether the algorithm works.

Availability	n	What you do
`bundled`	69	Nothing. Sixty-six ship a weights file; PhenoAge, KDM and homeostatic dysregulation are formulas with none to ship.
`untraced`	52	Catalogued, but no primary source for the numbers has been established yet, so nothing is bundled.
`licensed`	57	Obtain a coefficient file and register it; the numbers are research-use-only. DunedinPACE and five PC clocks import from the authors' files and reproduce the authors' code; the composite architectures (GrimAge, PCGrimAge, SystemsAge) are not implemented yet.

```
fa.registry.load().filter(availability="bundled")
```

Two licensed families can be imported from the authors' own files, which then score exactly as the authors' code does: DunedinPACE from an installed DunedinPACE package, a source checkout or an `.rda` (`fa.registry.load().import_dunedinpace(path)`; R: `import_dunedinpace()`), and the PC clocks from `CalcAllPCClocks.RData` (`import_pc_clocks(path)`).

Training targets, and what a score means

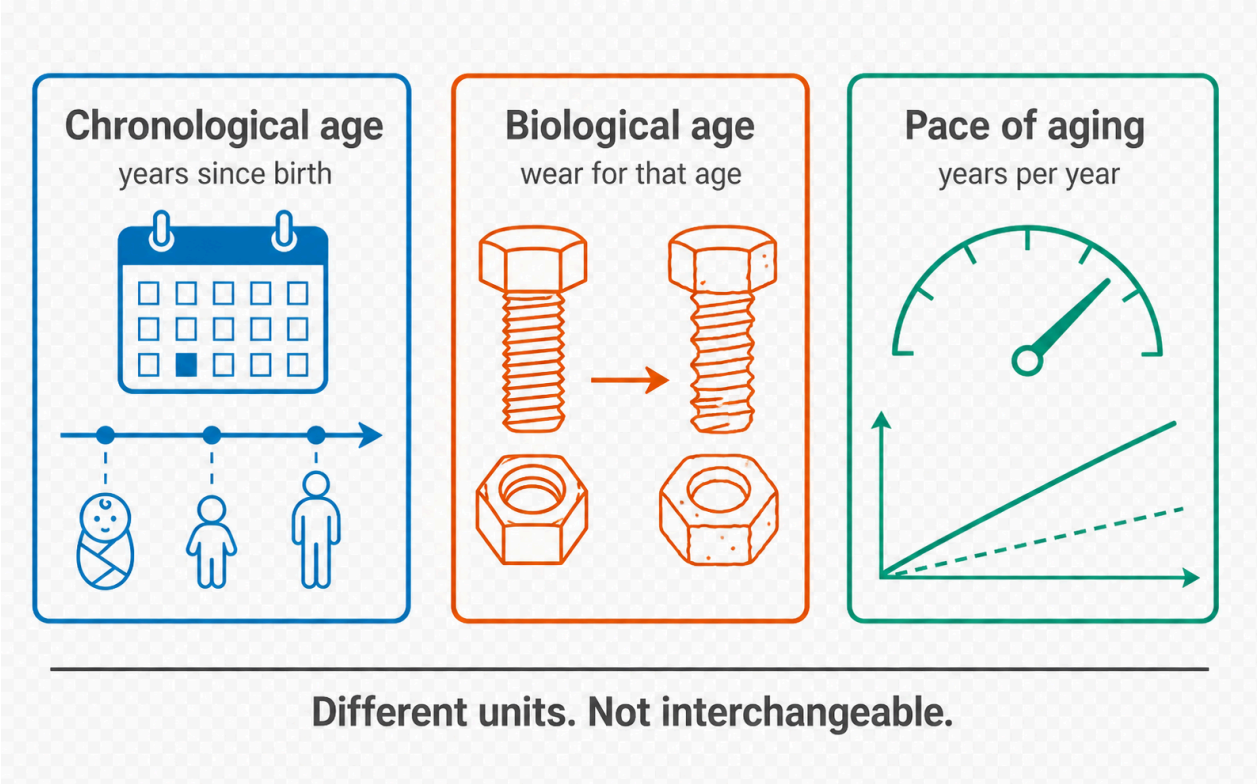


Figure 7: Three quantities that all get called "biological age", and the units that keep them apart

Chronological age is elapsed time, and it is known without measuring anything. Biological age is an estimate of how worn a body is for that elapsed time. Pace of aging is a rate, years of wear per year of calendar time, and it is the only one of the three that a single sample cannot establish on its own: the Dunedin measure was fitted against two decades of repeated biomarker readings. Because the three carry different units, none of them converts into another, which is why a registry entry declares a `scale_type` and not only a unit.

What a clock's output *means* is set by the outcome it was regressed on. Six families are in common use, and they are not variants of one quantity:

- **Chronological-age clocks** (Horvath 2013 [5], Hannum 2013 [4]) are fitted to calendar age. Their error bars are prediction error against a known truth.
- **Mortality- and morbidity-trained clocks** (PhenoAge [8], GrimAge [11]) are fitted to a survival-weighted composite and then rescaled into years. The unit is years; the target was never age.
- **Pace-of-aging clocks** (DunedinPACE [15]) are fitted to the *rate of change* in organ-system biomarkers tracked longitudinally. Belsky and colleagues draw the distinction directly:

> [Pace-of-aging measures capture] the rate of deterioration in system integrity occurring over a fixed time interval, whereas epigenetic clocks estimate the progress of aging up to a cross-section in time.
>
> Belsky et al., *eLife* 2022

- **Mitotic clocks** (epiTOC2 [20], StemTOC) estimate cumulative stem-cell divisions. The quantity is a count, not elapsed time.
- **Causality-enriched clocks** (CausAge, DamAge, AdaptAge) restrict features to CpGs with Mendelian-randomisation support, separating damaging from adaptive change (Ying et al., *Nature Aging* 2024, [19]).
- **Deconvolution models** return cell-type proportions, a composition that is not forced to sum to one; the sum shows how much of the sample the reference explains.

Reporting all six as "biological age" discards the information needed to interpret any of them.

Scale types and permitted operations

	Correlate	Difference	Residual	Acceleration
Years	✓	✓	✓	✓
Years, no fixed zero	✓	✓	✓	✗
Pace ratio	✓	✓	✗	✗
Divisions	✓	✓	✗	✗
Log hazard	✓	✗	✗	✗
Proportion	✓	✓	✗	✗

Figure 8: Six of the nine scales, and the four operations most often applied to them without checking

Reading down the acceleration column is the shortest statement of the problem. Only an age in years with a fixed origin admits it. A pace is already a rate, a division count is not elapsed time, and a log hazard is not even on an additive scale, so subtracting a chronological age from any of them produces a number with no units behind it. The matrix drawn here is the same one the code enforces, in `LEGAL_OPS`.

Every registry entry declares a `scale_type` alongside its unit, platform, species and the tissue its coefficients were fitted in. That field decides which downstream operations FALCONAge will perform:

scale_type	n	What the number is	Units in the registry	Operations permitted
age_years	80	An age. The model predicts elapsed time and is reported on the same scale as chronological age.	years, months, days	acceleration, correlate, difference, mean, residual
relative_score	53	A score with no external unit. Comparable within a cohort, meaningless as an absolute value.	unitless, picograms per milliliter, probability, and 12 more	correlate, rank
proportion	18	A composition. The estimates are fractions of a whole, not forced to sum to one; the sum shows how much the reference explains.	proportion	compositional, correlate, difference, mean
divisions	10	A count of cell divisions, not elapsed time.	beta value, proportion, cell divisions per stem cell, and 1 more	correlate, difference, mean, rank
gestational_weeks	8	An age, before birth.	weeks, days, units per week	acceleration, correlate, difference, mean, residual
mortality_log_hazard	3	A risk on the log-hazard scale. Exponentiating a difference gives a hazard ratio.	log hazard, per cohort SD, log hazard, unitless	correlate, hazard ratio, mean, rank
telomere_kb	2	An estimated physical length.	kilobases, base pairs	acceleration, correlate, difference, mean
pace_ratio	2	A rate: how much biological ageing per year of calendar time, at the moment of sampling.	biological years per chronological year	correlate, difference, mean, rank
age_years_relative	2	An age in years with a slope near one against chronological age, but no fixed origin: the offset moves between cohorts. The residual and group differences are defined; predicted minus chronological is not.	years	correlate, difference, mean, residual

Age acceleration is the residual of predicted age on chronological age. It is defined when the prediction is in years and undefined when it is a rate, so asking for it on DunedinPACE is an error with a message, not a number:

```
IllegalOperationError: 'acceleration' is not defined for dunedinpace, whose output is
pace_ratio (biological years per chronological year).
  Legal here: correlate, difference, mean, rank.
  A pace of aging is already a rate; subtracting chronological age from it is a units
  error, not a conservative choice.
```

8. Output types and units

A tool that returns a bare number is dangerous. Every prediction needs at least six attached fields.

8.1 The output record

`res.long()` returns one row per sample per clock:

```
sample_id, clock, value, unit, scale_type, generation, predicts, n_features,
coverage, mass_coverage, n_imputed, availability, registry_version, falconage_version
```

Acceleration is computed separately (`fa.acceleration`, with the convention recorded), the missing features and the fill used are in `res.coverage`, and the coefficient files' SHA-256, the device and the package versions are in `res.manifest`. pyaging writes the value into `adata.obs[clock]` and its metadata and missingness into `adata.uns`.

8.2 Scale types and what you may do with them

Scale type	Example clocks	Legal operations
Age in years	Horvath, GrimAge, PhenoAge, SystemsAge	subtract CA; residualise on CA; average across clocks only after z-scoring
Age in months / days / hours	Petkovich, Stubbs (months), Meer (days), BiTAge (hours)	convert before comparing to human clocks
Gestational age in weeks	Knight, Bohlin, Mayne, Lee, EPICGA	compare to ultrasound GA, not to adult clocks
Rate (bio-y per chron-y)	DunedinPACE, DunedinPoAm38	centred at 1; do **not** subtract CA
Log hazard	ZhangMortality, MetaboHealth, SenMortalityAge, organAging mortality, raw Cox scores	exponentiate for HR; convert to years only with an explicit Gompertz assumption
log10(hazard ratio)	tAge mortality clocks	never multiply by species lifespan
Probability	CVDWesterman, ADBahadoSingh, MammalianFemale	ROC/AUC, calibration
Log odds	ProstateCancerKirby, SenCultureAge	logistic transform before interpretation
Proportion	deconvolution clocks	not forced to sum to 1; the sum is how much the reference explains
Beta value	epiTOC1, stemTOC, HypoClock, Garagnani, Bocklandt	ordinal within a study; not comparable across platforms
Cell divisions per stem cell	epiTOC2, epiTOC3	

Scale type	Example clocks	Legal operations
Population doublings / passages	RepliTali, CellPopAge	
Physical units	DNAmTL (kb), PCDNAmTL (bp)	compare to measured values
Concentration	GrimAge2 sub-clocks (pg/mL), logCRP (ln mg/L), logA1C (ln %)	intermediate quantities, not ages
Unitless score	DNAmFILi, DNAmIC, DNAmStress, ReedBMI, DownSyndrome; the DNAmFitAge gait, grip and VO2max predictors	rank or z-score within cohort
Relative age	EnsembleAgeHumanMouse, tAge normalized age	fraction of maximum lifespan

A clock in physical units can be checked against the range its source paper reports, and a median far outside that range is a computation error rather than an unusual cohort. FALCONAge warns (category `implausible`) when a cohort's median leaves the interval. For DNAmTL the interval is 4 to 12 kb: Lu et al. 2019 (Table 2) fit DNAmTL = 8.05 − 0.018 × age kb in their test cohorts, 6.2 to 8.0 kb from birth to 100 years. The guard exists because FALCONAge 1.0.0 shipped DNAmTL with its intercept's sign reversed and returned about −8.5 kb on adult blood; nothing flagged it. Two more human scales carry an interval, each from a published fact about people:

Scale, unit	Interval	Why
`age_years`, years	−1 to 122.45	No verified human age exceeds Jeanne Calment's 122 years and 164 days (Robine et al. 2019); an age below −1 year precedes conception
`pace_ratio`, years per year	0.40 to 2.44	The slowest and fastest members of the Dunedin Study aged 0.40 and 2.44 biological years per chronological year over two decades (Belsky et al. 2022)

The intervals are wide on purpose: they catch a sign, unit or fill error, not an unusual cohort. The pace interval did so on its first real run. Forced below its 80% coverage floor on one cohort's EPIC v1 betas, DunedinPoAm38 lost 11 of its 46 CpGs, 23% of its weight, and with those filled by the dataset's mean its median was 0.38. The authors' own `PoAmProjector` fills an absent probe with its Dunedin cohort mean, which gives 0.85 on the same data. FALCONAge now fills from those means too.

8.3 Aggregate outputs

FALCONAge returns the first, second and last of these; the composite score and the per-system profile are not built.

- Per-sample × per-clock long table (`res.long()`) and wide matrix (`res.wide()`).
- Age-acceleration matrix, absolute and residual (`fa.acceleration(res, method="both")`).
- Composite score: z-score each clock within cohort, then average within generation class, and report the classes separately rather than mixing generations.
- Per-system / per-organ profile (SystemsAge, organAging, tAge tissue clocks) with a radar or heatmap layout.
- QC block: missingness per clock, platform detected, imputation used, warnings.

Chapter 10. Clock catalogue

This page is written from the registry the package ships, so it cannot describe a catalogue different from the one that scores. Every column below is a field the scoring code reads.

`scale_type` is the load-bearing one. It is not a label: it decides which downstream operations FALCONAge will perform, and asking for age acceleration on a `pace_ratio` clock is refused rather than computed.

178 clocks, registry schema `1.1.0`.

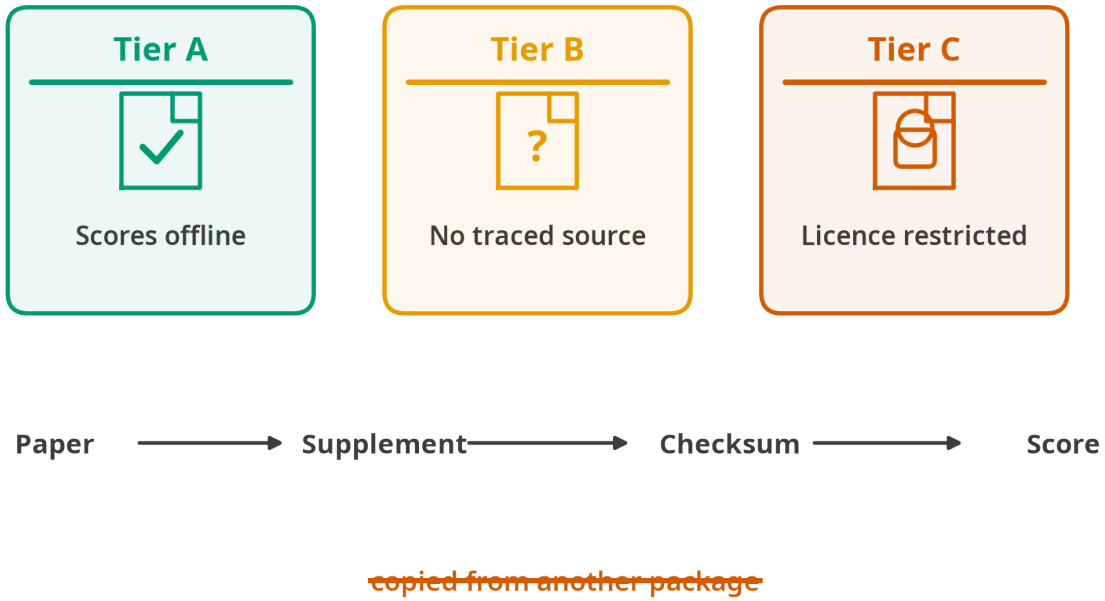


Figure 9: What separates the three availability groups, and what a traced coefficient source means

Availability is about provenance, not about quality. A `bundled` clock has a coefficient file whose origin was followed back to the paper or its supplement and checksummed, so the same numbers come out on any machine with no network. An `untraced` clock has no public source for its coefficients, and a `licensed` one has a source whose terms do not permit redistribution. Copying a coefficient vector out of another package would move clocks into the bundled column and is the one route not taken here, because a number with no source behind it cannot be checked against anything.

Bundled: 69 clocks

Coefficients ship with the package, or the clock is a formula with none to ship. Works offline, no network, no licence.

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
altumage	2022	chronological age	age_years	20318	multi-tissue	27K, 450K	de Lima Camillo 2022
bohlin	2016	gestational age	gestational_weeks	96	cord blood	450K	Bohlin 2016
cage	2023	chronological age	age_years	3225	whole blood	450K, EPICv1	Bernabeu 2023
cellpopage	2024	cell-population passage age	relative_score	42	cultured fibroblasts	EPICv1	Lujan 2024
corticalclock	2020	chronological age	age_years	347	brain cortex	450K	Shireby 2020
deconvolutebloodepicbcell	2018	B cell proportion	proportion	450	whole blood	450K, EPICv1	Salas 2018
deconvolutebloodepiccd4tcell	2018	CD4+ T cell proportion	proportion	450	whole blood	450K, EPICv1	Salas 2018
deconvolutebloodepiccd8tcell	2018	CD8+ T cell proportion	proportion	450	whole blood	450K, EPICv1	Salas 2018
deconvolutebloodepicmonocyte	2018	monocyte proportion	proportion	450	whole blood	450K, EPICv1	Salas 2018
deconvolutebloodepicneutrophil	2018	neutrophil proportion	proportion	450	whole blood	450K, EPICv1	Salas 2018
deconvolutebloodepicnkcell	2018	natural killer cell proportion	proportion	450	whole blood	450K, EPICv1	Salas 2018
depressionbarbu	2021	major depressive disorder	relative_score	196	whole blood	EPICv1	Barbu 2021
dnamfili	2022	frailty risk	relative_score	20	whole blood	EPICv1, 450K	Li 2022

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
dnamic	2025	intrinsic capacity	relative_score	91	whole blood	EPICv1	Fuentealba 2025
dnamphenoage	2018	phenotypic age	age_years	513	whole blood	27K, 450K, EPICv1	Levine 2018
dnamstress	2023	stress exposure	relative_score	211	whole blood	EPICv1	Jung 2023
dnamtl	2019	leukocyte telomere length	telomere_kb	140	whole blood	450K, EPICv1	Lu 2019
downsyndrome	2021	Down syndrome methylation score	relative_score	652	neonatal blood spots	EPICv1	Muskens 2021
dunedinpoam38	2020	pace of aging	pace_ratio	46	whole blood	450K	Belsky 2020
epicmithyper	2020	mitotic age	divisions	184	B cells	450K, EPICv1	Duran-Ferrer 2020
epicmithypo	2020	mitotic age	divisions	1164	B cells	450K, EPICv1	Duran-Ferrer 2020
epitoc1	2016	mitotic age	divisions	385	multi-tissue, whole blood	450K	Yang 2016
epitoc2	2020	mitotic age	divisions	163	whole blood	450K	Teschendorff 2020
epitoc3	2020	mitotic age	divisions	170	cultured primary human cells, whole blood, multi-tissue, cord blood	450K, EPICv1	Teschendorff 2020
hannum	2013	chronological age	age_years	71	whole blood	450K	Hannum 2013
hd	2013	physiological dysregulation	relative_score	—	blood	clinical laboratory assays	Cohen 2013
horvath2013	2013	chronological age	age_years	353	multi-tissue	27K, 450K	Horvath 2013
hrsinchphenoage	2022	phenotypic age	age_years	959	whole blood	450K, EPICv1	Higgins-Chen 2022
hypoclock	2020	mitotic age	divisions	678	multi-tissue	450K	Teschendorff 2020
intrinclock	2024	chronological age	age_years	370	multi-tissue	450K, EPICv1	Tomusiak 2024

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
kdm	2006	biological age	age_years	—	blood	clinical laboratory assays	Klemera 2006
knight	2016	gestational age	gestational_weeks	148	cord blood, neonatal blood spots	27K, 450K	Knight 2016
leecontrol	2019	gestational age	gestational_weeks	546	placenta	450K, EPICv1	Lee 2019
leerefinedrobust	2019	gestational age	gestational_weeks	395	placenta	450K, EPICv1	Lee 2019
leerobust	2019	gestational age	gestational_weeks	558	placenta	450K, EPICv1	Lee 2019
lin	2016	chronological age	age_years	99	whole blood	27K, 450K	Lin 2016
linage2	2025	mortality risk	age_years	59	blood	clinical laboratory assays	Fong 2025
mayne	2017	gestational age	gestational_weeks	62	placenta	27K, 450K	Mayne 2017
mccartneyalcohol	2018	alcohol consumption	gestational_weeks	450	whole blood	EPICv1	McCartney 2018
mccartneybmi	2018	body mass index	relative_score	1109	whole blood	EPICv1	McCartney 2018
mccartneybodyfat	2018	body fat percentage	relative_score	968	whole blood	EPICv1	McCartney 2018
mccartneyeducation	2018	educational attainment	relative_score	373	whole blood	EPICv1	McCartney 2018
mccartneyhdlcholesterol	2018	HDL cholesterol	relative_score	737	whole blood	EPICv1	McCartney 2018
mccartneyldlcholesterol	2018	LDL cholesterol	relative_score	233	whole blood	EPICv1	McCartney 2018
mccartneysmoking	2018	smoking exposure	relative_score	233	whole blood	EPICv1	McCartney 2018

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
mccartney totalchole sterol	2018	total cholesterol	relative_ score	204	whole blood	EPICv1	McCartney 2018
mccartney totalhdlra tio	2018	total-to-HDL cho- lesterol ratio	relative_ score	412	whole blood	EPICv1	McCartney 2018
mccartneywhr	2018	waist-to-hip ratio	relative_ score	226	whole blood	EPICv1	McCartney 2018
meer	2018	chronological age	age_years	435	multi-tissue	RRBS	Meer 2018
metaboage	2020	chronological age	age_years	56	blood	Nightingale Health 1H-NMR	van den Akker 2020
metabohealth	2019	mortality	mortality_ log_hazard	14	blood	Nightingale Health 1H-NMR	Deelen 2019
pedbe	2020	chronological age	age_years	94	buccal epithelium	450K, EPICv1	McEwen 2020
phenoage	2018	phenotypic age	age_years	10	blood	clinical labora- tory assays	Levine 2018
prostatec ancerkirby	2017	prostate cancer	relative_ score	3	prostate	450K	Kirby 2017
replitali	2022	replicative his- tory	divisions	87	cultured primary human cells	EPICv1	Endicott 2022
skinandblood	2018	chronological age	age_years	391	buccal epithe- lium, whole blood, epithelium, cultured fibrob- lasts, skin, cord blood	450K, EPICv1	Horvath 2018
stemtoc	2024	mitotic age	divisions	371	multi-tissue, cul- tured human cells, whole blood	450K, EPICv1	Zhu 2024
stemtocvitro	2024	mitotic age	divisions	629	multi-tissue, cul- tured human cells	450K, EPICv1	Zhu 2024
stoch	2024	chronological age	age_years	353	sorted monocytes	450K	Tong 2024
stocp	2024	chronological age	age_years	513	sorted monocytes	450K	Tong 2024
stocz	2024	chronological age	age_years	514	sorted monocytes	450K	Tong 2024
vidalbralo	2016	chronological age	age_years	8	whole blood	27K	Vidal-Bralo 2016

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
weidner	2014	chronological age	age_years	3	whole blood	27K, bisulfite sequencing	Weidner 2014
yingadaptage	2024	adaptive epigenetic age	age_years_relative	999	whole blood	450K	Ying 2024
yingcausage	2024	chronological age	age_years	585	whole blood	450K	Ying 2024
yingdamage	2024	damaging epigenetic age	age_years_relative	1089	whole blood	450K	Ying 2024
zhangblup	2019	chronological age	age_years	319607	whole blood, saliva	450K, EPICv1	Zhang 2019
zhangen	2019	chronological age	age_years	514	whole blood, saliva	450K, EPICv1	Zhang 2019
zhangmortality	2017	mortality risk	mortality_log_hazard	10	whole blood	450K	Zhang 2017

Untraced: 52 clocks

Catalogued, but no primary source for the coefficients has been established, so nothing is bundled. The metadata is real; the numbers are absent.

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
abec	2020	chronological age	age_years	—	whole blood	EPICv1	Lee 2020
adbahadosingh	2021	late-onset Alzheimer's disease	relative_score	—	whole blood	EPICv1	Bahado-Singh 2021
bocklandt	2011	EDARADD methylation	relative_score	—	saliva	27K	Bocklandt 2011
cabec	2020	chronological age	age_years	—	whole blood	450K, EPICv1	Lee 2020
compil6	2021	interleukin-6	relative_score	—	whole blood	450K	Stevenson 2021
ctsliver	2024	chronological age	age_years	—	liver	EPICv1	Tong 2024
cvdwesterman	2020	cardiovascular disease risk	relative_score	—	whole blood	450K	Westerman 2020
dnamfitage	2023	physical-fitness biological age	age_years	—	whole blood	450K	McGreevy 2023
dnamfitagegaitf	2023	gait speed	relative_score	—	whole blood	450K	McGreevy 2023
dnamfitagegaitm	2023	gait speed	relative_score	—	whole blood	450K	McGreevy 2023
dnamfitagegripf	2023	grip strength	relative_score	—	whole blood	450K	McGreevy 2023
dnamfitagegripm	2023	grip strength	relative_score	—	whole blood	450K	McGreevy 2023
dnamfitagevo2max	2023	VO2max	relative_score	—	whole blood	450K	McGreevy 2023
eabec	2020	chronological age	age_years	—	whole blood	EPICv1	Lee 2020
encen100	2023	chronological age	age_years	—	whole blood, saliva, buccal epithelium	450K, EPICv1	Dec 2023
encen40	2023	chronological age	age_years	—	whole blood, saliva, buccal epithelium	450K, EPICv1	Dec 2023

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
ensemblea gehumanmouse	2025	relative age	relative_score	—	multi-tissue, whole blood	mammalian methylation array	Haghani 2025
ensemblea gestatic	2025	intervention-responsive epigenetic age	age_years	—	multi-tissue	Horvath MammalMethylChip40, Horvath MammalMethylChip320	Haghani 2025
ensemblea gestatictop	2025	intervention-responsive epigenetic age	age_years	—	multi-tissue	Horvath MammalMethylChip40, Horvath MammalMethylChip320	Haghani 2025
epicga	2021	gestational age	gestational_weeks	—	cord blood	EPICv1	Haftorn 2021
fiage	2025	frailty index	relative_score	53	whole blood	EPICv1	no primary source traced, catalogued at
garagnani	2012	ELOVL2 methylation	relative_score	—	whole blood	450K	Garagnani 2012
gliasin	2024	chronological age	age_years	—	brain cortex	450K	Tong 2024
han	2020	chronological age	age_years	—	whole blood	450K	Han 2020
hep	2024	chronological age	age_years	—	liver	EPICv1	Tong 2024
hepatoxu	2017	hepatocellular carcinoma	relative_score	—	plasma cell-free DNA	targeted bisulfite sequencing	Xu 2017
mammalian1	2023	chronological age	age_years	—	multi-tissue	Horvath MammalMethylChip40	Lu 2023
mammalian2	2023	chronological age	age_years	—	multi-tissue	Horvath MammalMethylChip40	Lu 2023
mammalian3	2023	chronological age	age_years	—	multi-tissue	Horvath MammalMethylChip40	Lu 2023
mammalian blood2	2023	chronological age	age_years	—	blood	Horvath MammalMethylChip40	Lu 2023
mammalian blood3	2023	chronological age	age_years	—	blood	Horvath MammalMethylChip40	Lu 2023

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
mammalianfemale	2023	sex	relative_score	—	multi-tissue	mammalian methylation array	Li 2023
mammalian lifespan	2024	species maximum lifespan	age_years	—	multi-tissue	Horvath MammalMethylChip40	Li 2024
mammalian skin2	2023	chronological age	age_years	—	skin	Horvath MammalMethylChip40	Lu 2023
mammalian skin3	2023	chronological age	age_years	—	skin	Horvath MammalMethylChip40	Lu 2023
neusin	2024	chronological age	age_years	—	brain cortex	450K	Tong 2024
petkovich	2017	chronological age	age_years	—	whole blood	bisulfite sequencing	Petkovich 2017
pipekelasticnet	2022	chronological age	age_years	—	multi-tissue	27K, 450K, EPICv1	Pipek 2022
pipekfilteredh	2022	chronological age	age_years	—	multi-tissue	27K, 450K, EPICv1	Pipek 2022
pipekretrainedh	2022	chronological age	age_years	—	multi-tissue	27K, 450K, EPICv1	Pipek 2022
reedbmi	2020	BMI methylation score	relative_score	—	whole blood, cord blood	450K	Reed 2020
replitalinorm	2022	replicative history	divisions	—	cultured fibroblasts	EPICv1	Endicott 2022
retroelem entagev1	2024	chronological age	age_years	—	whole blood	EPICv1	Ndhlovu 2024
retroelem entagev2	2024	chronological age	age_years	—	whole blood	EPICv1	Ndhlovu 2024
senchronoage	2026	chronological age	age_years	—	whole blood	450K, EPICv1	Kasamoto 2026
senculture age	2026	cellular senescence	relative_score	—	cultured fibroblasts, cultured mesenchymal stromal cells	450K, EPICv1	Kasamoto 2026
senmortalityage	2026	mortality risk	mortality_log_hazard	—	whole blood	450K, EPICv1	Kasamoto 2026
stubbs	2017	chronological age	age_years	—	multi-tissue, liver, lung, heart, brain	RRBS	Stubbs 2017

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
					cortex, skeletal muscle, cerebellum, spleen		
thompson	2018	chronological age	age_years	—	adipose tissue, blood, cerebellum, brain cortex, heart, kidney, liver, lung, skeletal muscle, spleen	RRBS	Thompson 2018
wu	2019	chronological age	age_years	—	whole blood	27K, 450K	Wu 2019
xchrom	2021	X-chromosome dosage	relative_score	—	whole blood	450K, EPICv1	Wang 2021
ychrom	2021	Y-chromosome presence	relative_score	—	whole blood	450K, EPICv1	Wang 2021

Licensed: 57 clocks

The architecture is implemented and tested; the coefficients are research-use-only and are not ours to distribute. Obtain a file and register it.

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
cpgptgrimage3	2024	biological age	age_years	—	whole blood	450K	de Lima Camillo 2024
cpgptpcgrimage3	2024	biological age	age_years	—	whole blood	450K	de Lima Camillo 2024
dnamemrage	2026	electronic medical record age	age_years	—	whole blood	450K, EPICv1	Chen 2026
dunedinpace	2022	pace of aging	pace_ratio	—	whole blood	EPICv1	Belsky 2022
grimage	2019	mortality risk	age_years	—	whole blood	450K	Lu 2019
grimage2	2022	mortality risk	age_years	—	whole blood	450K	Lu 2022
grimage2adm	2022	adrenomedullin	relative_score	—	whole blood	450K	Lu 2022
grimage2b2m	2022	beta-2-microglobulin	relative_score	—	whole blood	450K	Lu 2022
grimage2cystatinc	2022	cystatin C	relative_score	—	whole blood	450K	Lu 2022
grimage2gdf15	2022	GDF-15	relative_score	—	whole blood	450K	Lu 2022
grimage2leptin	2022	leptin	relative_score	—	whole blood	450K	Lu 2022
grimage2loga1c	2022	hemoglobin A1c	relative_score	—	whole blood	450K	Lu 2022
grimage2logcrp	2022	C-reactive protein	relative_score	—	whole blood	450K	Lu 2022
grimage2packyrs	2022	smoking exposure	relative_score	—	whole blood	450K	Lu 2022
grimage2pai1	2022	PAI-1	relative_score	—	whole blood	450K	Lu 2022
grimage2timp1	2022	TIMP-1	relative_score	—	whole blood	450K	Lu 2022

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
omicmage	2026	biological age	age_years	—	whole blood	450K, EPICv1	Chen 2026
pcdnamtl	2022	leukocyte telomere length	telomere_kb	—	whole blood	450K	Higgins-Chen 2022
pcgrimage	2022	mortality risk	age_years	—	whole blood	450K	Higgins-Chen 2022
pcgrimageadm	2022	adrenomedullin	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcgrimageb2m	2022	beta-2 microglobulin	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcgrimagecystatinc	2022	cystatin C	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcgrimagegdf15	2022	growth differentiation factor 15	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcgrimageleptin	2022	leptin	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcgrimagepackyrs	2022	smoking pack-years	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcgrimagepai1	2022	plasminogen activator inhibitor 1	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcgrimagetimp1	2022	TIMP metallopeptidase inhibitor 1	relative_score	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pchannum	2022	chronological age	age_years	—	whole blood	450K	Higgins-Chen 2022
pchorvath2013	2022	chronological age	age_years	—	multi-tissue	450K, EPICv1	Higgins-Chen 2022
pcphenoage	2022	phenotypic age	age_years	—	whole blood	450K, EPICv1	Higgins-Chen 2022
pcskinandblood	2022	chronological age	age_years	—	skin, whole blood, cultured fibroblasts	450K, EPICv1	Higgins-Chen 2022
retroclock	2025	retroelement methylation age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsage	2025	multisystem biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
systemsag eagepredic tion	2025	chronological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsageblood	2025	blood-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsagebrain	2025	brain-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsageheart	2025	cardiovascular-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsag ehormone	2025	endocrine-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsag eimmune	2025	immune-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsag einflammat ion	2025	inflammatory-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsag ekidney	2025	renal-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsageliver	2025	hepatic-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsagelung	2025	pulmonary-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsag emetabolic	2025	metabolic-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
systemsag emusculosk eletal	2025	musculoskeletal-system biological age	age_years	—	whole blood	450K, EPICv1	Sehgal 2025
twelvecel ldeconvolu tebloodepi cbas	2022	basophil proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecel ldeconvolu	2022	memory B cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
tebloodepicbmem							
twelvecelldeconvolutebloodepiccbnv	2022	naive B cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolutebloodepicccd4mem	2022	memory CD4+ T cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolutebloodepicccd4nv	2022	naive CD4+ T cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolutebloodepicccd8mem	2022	memory CD8+ T cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolutebloodepicccd8nv	2022	naive CD8+ T cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolutebloodepicceos	2022	eosinophil proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolutebloodepiccmono	2022	monocyte proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolutebloodepiccneu	2022	neutrophil proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022
twelvecelldeconvolu	2022	natural killer cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022

Clock	Year	Predicts	Scale	Features	Tissue	Platform	Paper
tebloodepicnk							
twelvecelldeconvolutebloodepictreg	2022	regulatory T-cell proportion	proportion	—	purified blood leukocytes	EPICv1	Salas 2022

Querying it from code

Python

```python
reg = fa.registry.load()
reg.filter(availability="bundled", scale_type="age_years")
reg.search("mortality")
reg.compatible_with(data)
reg.get("horvath2013")
```

R

```r
list_clocks(tier = "bundled")
list_clocks(search = "mortality")
compatible_clocks(d)
clock_info("horvath2013")
```

Chapter 11. Age acceleration

Age acceleration

The conventions disagree by several years on the same data, and papers frequently do not say which they used.

Python

```
fa.acceleration(res, method="absolute")      # predicted - chronological
fa.acceleration(res, method="residual")      # residual of predicted on chronological
fa.acceleration(res, method="both")         # both, as <clock>_absolute / <clock>_residual
fa.acceleration(res, method="within_group", group="dataset")
```

R

```
acceleration(res, method = "absolute")
acceleration(res, method = "residual")
acceleration(res, method = "both")
acceleration(res, method = "within_group", group = "dataset")
```

Absolute is interpretable in years and carries the clock's own bias: a clock five years high on everybody gives everybody five years of acceleration. Residual removes that bias, and removes any real cohort-wide effect with it. Within-group is what the AA2 benchmark needs: it asks whether cases accelerate relative to *their own* controls. `"both"` returns the first two side by side, suffixed, for when you would otherwise have to pick without knowing which the comparison paper used.

Against a reference population

The residual is centred on this cohort, so it cannot say how the cohort compares with anyone else, and it absorbs any effect the whole cohort shares. `method="reference"` fits the age line in a second, scored population and applies it unchanged here:

```
ref = fa.score(reference_data, clocks=["phenoage"])
fa.acceleration(res, method="reference", reference=ref, match=("age", "sex"))
```

The line is fitted separately within each level of the categorical `match` columns (usually sex), an age outside the reference's range in its stratum is refused rather than extrapolated, and a reference scored with different coefficients is refused. `attrs["reference_fit"]` records, per clock and stratum, the reference sample size, age range, fitted line and residual SD.

Adjusting for cell composition

Blood cell composition changes with age, and with most other things that happen to a person. In more than 10,000 blood samples, immune composition was significantly associated with age acceleration for **every one** of six widely used clocks (Zhang et al., *Aging Cell* 2024;23:e14071, doi:10.1111/acel.14071), so an unadjusted acceleration is measuring two things and reporting one number.

The proportions needed to separate them are usually already in the result, computed by the deconvolution clocks in the same run:

Python

```
fa.cell_composition(res)                        # the proportions, as covariates
fa.acceleration(res, adjust="cell_composition") # net of them
fa.acceleration(res, adjust=["cd8t", "mono"])   # or measured counts from obs
```

R

```
cell_composition(res)
acceleration(res, adjust = "cell_composition")
acceleration(res, adjust = c("cd8t", "mono"))
```

An adjusted acceleration is a different quantity from an unadjusted one, so the returned frame records which covariates went in. Comparing the two as though they were the same measurement is the error this is designed to make visible.

Ask for acceleration on a pace-of-aging clock and it refuses:

```
IllegalOperationError: 'acceleration' is not defined for dunedinpoam38, whose output is
  pace_ratio (biological years per chronological year).
  Legal here: correlate, difference, mean, rank.
  A pace of aging is already a rate; subtracting chronological age from it is a units
  error, not a conservative choice.
```

That refusal is the point. A wrong number in a column of numbers is invisible.

9. Downstream statistics on an aging score

9.1 Association with chronological age

Pearson r, Spearman ρ, MAE, MedAE, RMSE, and the regression slope of BA on CA. The slope is the diagnostic: penalised regression compresses predictions toward the training mean, so slopes < 1 are normal and produce the well-documented systematic underestimation in older subjects. Report the slope, not just r.

9.2 Mortality

Cox proportional hazards with the standardised score plus CA and sex as covariates:

```
coxph(Surv(time, status) ~ scale(BA_advance) + age + gender, data)
```

Report the HR per SD with 95% CI. BioAge standardises the advance within sex so the effect size means "one sex-specific SD of biological age advancement" (BioAge table_surv.R). Observed HRs from BioAge's NHANES IV tables run 1.21–1.61 per SD across measures and subgroups, with KDM advance, PhenoAge advance and homeostatic dysregulation broadly comparable (BioAge README).

Additional model families worth supporting: time-dependent ROC/AUC, Harrell's C-index, competing risks (Fine–Gray), and SuperLearner ensembles with out-of-fold predictions (the TwinGene/UK Biobank pipeline uses 10-fold CV, optionally repeated across seeds).

9.3 Morbidity and phenotype

Logistic or Poisson models for prevalent and incident disease; multimorbidity counts; frailty index; intrinsic capacity; grip strength and gait speed; cognitive scores. Interaction tests for BA × (BMI, sex, smoking, education) are standard in the epidemiological literature.

9.4 Group comparison and intervention response

The ComputAgeBench formulation, which is the cleanest published statement of the problem (ComputAge benchmarking.py:339–397):

- **AA2** - is the age acceleration in an aging-accelerating condition (AAC) cohort greater than in healthy controls from the same study? One-sided Welch's t-test if Δ is assumed normal, else one-sided Mann–Whitney U.
- **AA1** - is the age acceleration in an AAC cohort greater than zero, with no control group? One-sided one-sample t-test, else one-sided Wilcoxon signed-rank.
- Benjamini–Hochberg FDR correction is applied **per model across datasets**, then thresholded at 0.05.

For paired intervention designs, use paired tests on within-subject deltas and report both the absolute change and the change relative to the measurement's own ICC.

9.5 Reliability

Intraclass correlation coefficient on technical replicates (same DNA, two arrays) and on biological replicates (same person, short interval). Pool across studies with Fisher-Z. The result to remember is from Sehgal et al. (bioRxiv 2025.10.13.682176), on the TranslAGE platform: **technical reproducibility does not predict biological reliability**, and PCGrimAge was the only clock reaching ICC > 0.75 for biological reliability. A tool should surface each clock's published ICC next to its prediction.

Chapter 12. Uncertainty, specimen and platform

Measurement uncertainty

Technical replicates of the *same DNA* differ by up to nine years on prominent clocks, so a bare number does not say whether a difference between two samples is real.

Python

```
se = fa.technical_se(res, d)
se.diagnostics[["median_se", "se_over_sd", "implied_cohort_icc"]]

fa.conformal_interval(res, level=0.90)     # the other, larger uncertainty
fa.power("horvath2013", effect=0.5, result=res)   # how many samples you need
```

R

```
u <- technical_se(res, d)
u$diagnostics

conformal_interval(res, level = 0.90)
power_n("horvath2013", effect = 0.5, result = res)
```

```
              method  n_features  median_se  se_over_sd  implied_cohort_icc
horvath2013    probe         353       1.58        0.14                0.98
hannum         probe          71       2.46        0.22                0.95
dunedinpoam38  probe          46       0.04        0.53                0.72
```

Two samples whose intervals overlap are not distinguishable by that clock on that assay, however different their point estimates look.

19. Uncertainty, specimen and platform

The preceding sections are about computing the number correctly. This one is about the fact that a correctly computed number is still not interpretable on its own, which is what the field's own literature says and what no implementation, including this one until now, did anything about.

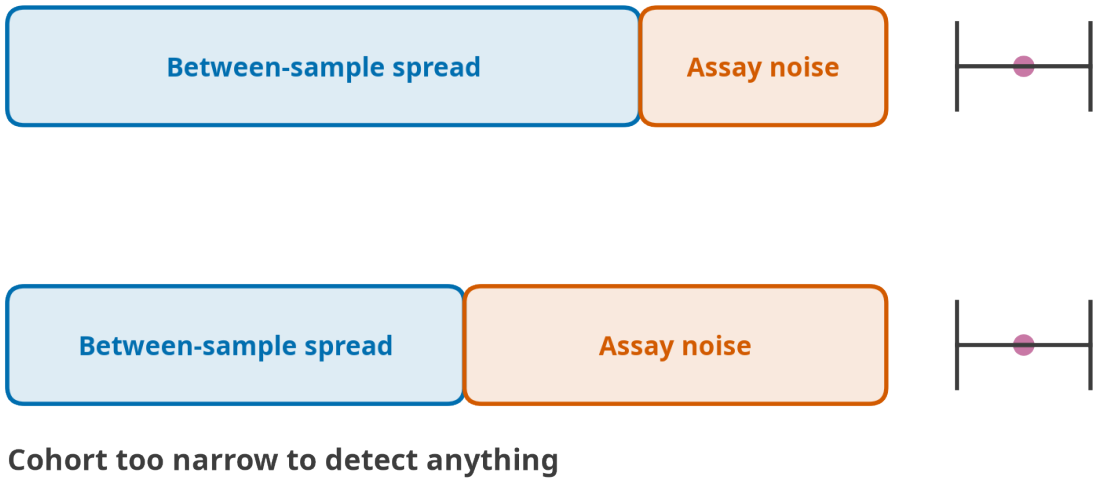


Figure 10: The same assay noise against a wide cohort and a narrow one

Reliability is not a property of a clock alone. The intraclass correlation is `1 - SE^2 / SD^2`, so the same technical error divided by a narrower between-sample spread gives a worse coefficient without anything about the assay having changed. A clock with an excellent published ICC, measured on a cohort spanning sixty years of age, can be unable to resolve anything in a trial whose participants are all within a decade of each other. That is the calculation to run before recruiting, not after.

19.1 The point estimate problem

Split one DNA sample, run both halves on the same array, and six prominent clocks disagree with themselves by **up to nine years** (Nat Aging 2022). In adult whole blood many 450K probes are measured unreliably, and the unreliable ones replicate worse (Sugden et al., *Patterns* 2020;1:100014, doi:10.1016/j.patter.2020.100014); DunedinPACE was trained only on the 81,239 probes shared by 450K and EPIC whose ICC exceeded 0.4 in that study (Belsky et al. 2022). A reader handed "DNAmAge = 54.2" cannot tell whether a 1.5-year change between two draws is biology or the assay.

Every package in the field emits that bare number. FALCONAge emits an interval, and the arithmetic behind it is short enough to check by hand.

19.2 Propagating measurement error

A linear clock is `score = f(b + wᵀβ)`. Give each probe a measurement variance `σ²ⱼ` and the score's variance follows directly:

$$\mathrm{Var} = f'(\mathrm{raw})^2 \sum_j w_j^2 \, \sigma_j^2$$

$$\sigma_j^2 = s_j^2 \, (1 - \mathrm{ICC}_j)$$

with s^2_j the spread of probe *j* across the cohort in front of you and ICC_j its published test-retest reliability. One matrix–vector product per clock.

Three parts of that are load-bearing:

The ICC table is measured, not assumed. `registry/data/probe_icc.csv.gz` carries 283,860 probes, derived from 69 blood samples each assayed twice on EPIC v1 and preprocessed with SeSAMe 2, using the two-way random-effects, absolute-agreement, single-rating model, ICC(2,1) (Epigenetics 2024, CC BY 4.0). Its citation, licence and SHA-256 go into the run manifest, because an interval whose provenance is not recorded is worse than no interval.

An imputed feature contributes its full variance. Not the reduced $s^2(1-ICC)$. An imputed probe is the cohort mean or a published reference; it carries no information about the sample in front of you. Treating it as well-measured would make *worse data produce a narrower interval*, which is the wrong direction and the part of the calculation most likely to look like a bug.

The output transform is differentiated, not linearised once. Horvath's `anti_log_linear` has slope $21 \cdot e^x$ below zero and 21 above it, so a child's uncertainty and an adult's scale differently from the same raw error. Every op in the catalogue has an analytic derivative; the four that destroy the information (`binarize`, `rank_normalize`, `quantile_normalize`, `simplex_projection`) have none, and a chain containing one **refuses** rather than skipping the step and returning a confidently small interval.

The sum assumes probe errors are independent, and they are not. Writing $\sum_j w_j^2 \sigma_j^2$ drops the cross term $2\sum_{j<k} w_j w_k \operatorname{Cov}(\varepsilon_j, \varepsilon_k)$. Chip position, plate and scanner drift move many probes together, and those covariances are positive far more often than negative, so **the SE reported here is a lower bound**, and not necessarily a tight one, since a clock with hundreds of positively correlated terms can have a true SE several times this.

The honest alternative does not exist: a $319,607 \times 319,607$ error covariance would need replicate designs nobody publishes. What exists is a cross-check. `source="clock"` derives the same quantity from a published clock-level ICC measured on real technical replicates, which *does* contain the correlated part; when the two disagree substantially, the clock-level figure is the one to believe and the gap is the correlated component the probe path cannot see. `icc_from_replicates()` closes it properly on your own duplicates. This is stated rather than buried because an interval presented without its assumption is exactly the false precision the section exists to argue against.

On the corpus, Horvath 2013 comes out at ±1.58 years of technical error with an implied cohort ICC of 0.98; DunedinPoAm38 is the least repeatable at 0.72; the 319,607-probe BLUP clock reaches 1.00, because a weight spread thinly over the whole array averages the noise away. That last result is the same mechanism that makes PC clocks robust to probe loss, arrived at from the other direction. Read all three as lower bounds, for the reason just given.

19.3 Prediction error is a different, larger question

`technical_se()` asks what a repeat of the same DNA would do. `conformal_interval()` asks how far the number is from the truth.

Split conformal: take healthy whole-blood samples with known ages, compute the absolute residuals, and the $\lceil (n+1)(1-\alpha) \rceil / n$ quantile is a half-width that covers the truth at the stated rate on any exchangeable sample. No distributional assumption is made, and the guarantee is finite-sample. Conformalised quantile regression (Genet Epidemiol 2025) is the better version, widening the interval where the training data

were sparse; it needs fitted quantile models, so split conformal ships and the calibration table reports the width per age band so a reader can see where one width is lying.

The calibration takes whole blood, buffy coat and peripheral blood leukocytes from adults, and nothing else. The corpus also labels sorted populations and PBMC as blood, and on those a whole-blood clock's residual carries the cell type: on its healthy controls Hannum ran 21 years low on CD8+ T cells, and DNAm PhenoAge 24 years low on CD14+ monocytes and 40 on CD8+ T cells. Until that was noticed they were more than half the calibration set, and the widths quoted composition as clock error.

The limitation is not small and is stated in the output: every row carries `exchangeable = False`. The calibration cohort is adult, blood, and overwhelmingly of European ancestry. On a paediatric or non-European cohort the guarantee does not transfer, and no widening factor would make it.

19.3.1 What the calibration found, and what it turned into

The first calibration put Ying's DamAge and AdaptAge 37–150 years from chronological age. The obvious reading ("these are not age predictors") turned out to be wrong, and the measurement that settled it is worth reproducing because the distinction it draws is a general one.

Regress predicted on chronological across the corpus's healthy cohorts:

clock	r	slope	offset swing between datasets
horvath2013	0.905	0.982	15 y
yingcausage	0.960	0.954	15 y
yingdamage	0.909	**0.967**	**162 y**
yingadaptage	0.745	1.170	**162 y**

DamAge's slope is 0.967, better than DNAmPhenoAge's 1.199. It tracks age. What it does not have is a **fixed origin**: its offset against chronological age is −84 years in one healthy cohort and +78 in another.

The mechanism is visible in one number. DamAge and AdaptAge move as near mirror images, **r = −0.975** across datasets, and the swing in their *sum* is only 33 years against 162 for each alone. Their published intercepts are **+543.43** and **−511.97**; a dataset-level shift in overall methylation pushes one up and the other down by nearly the same amount. One shared offset, hugely amplified, not two broken clocks.

So neither obvious label fits. `age_years` would license `predicted − chronological`, which measures the cohort. `relative_score` would forbid the residual and the group difference, the operations the paper itself reports, and would contradict a unit of years and a training target of chronological age, both of which are accurate.

They are `age_years_relative`: years, slope near one, no fixed zero. `LEGAL_OPS` admits `residual`, `difference`, `mean` and `correlate`, and refuses `acceleration`. The vocabulary had listed `acceleration` and `residual` separately since v1.0 and nothing had ever read the difference; this is the first clock that needs it.

19.4 Specimen type is not metadata

Same 91 people, three specimens (bioRxiv 2025.09.16.673560, CATSLife1, mean age 30.9):

Comparison	Mean difference in clock age	Spearman between the two
saliva vs buffy coat	**3.83 – 16.46 years**	0.45 – 0.69
saliva vs PBMC	in the same direction	0.25 – 0.55
buffy coat vs PBMC	−0.06 – 0.39 years	0.66 – 0.87

Read the two columns together, because separately each is reassuring. Saliva and buffy coat clock ages correlate at up to 0.69, a correlation most people would accept, and are still a decade apart. **Correlation is not agreement**, and the check that gets run is the correlation. The blood pair is the control: same study, same pipeline, difference under half a year, which is what makes the saliva number a specimen effect rather than noise.

Which clock matters as much as which specimen. On those samples DunedinPACE differed by −0.007 biological years per chronological year (p = 0.486) where Hannum differed by 12.5 years, so "this clock transfers" is a per-clock fact, not a per-tissue one, and FALCONAge therefore holds the policy per clock. Across unrelated tissues the published figure is 20–30 years (PMC12714307).

FALCONAge reads `obs["tissue"]` against each clock's declared training tissue and takes one of three positions:

`allow`
: the genuinely pan-tissue clocks. Nothing to say.

`warn`
: the default. Says what the clock was fitted on, what it was given, and the measured discordance where one is published.

`refuse`
: twelve clocks whose training tissue has no counterpart elsewhere: four placenta clocks, three cord-blood clocks, neonatal blood spots, buccal, and three brain clocks. A placenta clock on blood is a category error, not an offset.

Not every package holds a policy. methylclock 1.16.0's `DNAmAge()` takes no tissue argument and, with its default `clocks = "all"`, returns PedBE, a buccal clock (McEwen et al., PNAS 2020), for blood. FALCONAge refuses PedBE on blood, so a clock missing from a FALCONAge run that another package reported is sometimes the refusal working.

Cell-free DNA is refused by **every** clock regardless of policy. It is not a tissue a clock generalises to; it is a fragment population shed from many of them, and array clocks applied to it directly perform poorly (bioRxiv 2025.11.27.690895).

And when `obs` has no `tissue` column at all, the run says so once, because silence was the previous behaviour, and silence is the bug.

> **i** This check found a real error in our own test suite
>
> At first the integration tests scored the three Lee **placenta** clocks on GSE66459, which is umbilical **cord blood**, and asserted only that the answers looked like gestational weeks. They did, because gestational age is gestational age whatever tissue you read it from. That is exactly the failure the check exists to catch, and it caught it here first.

19.5 Probe loss, priced

`probe_loss()` has always said how much of a model is absent: count and coefficient mass. It could not say what that costs. Now it can, because the cost was measured: every full-450K matrix in the corpus, masked down to each platform's probe set and rescored, 206 samples across eight datasets. The two right-hand columns are the same masks under two ways of filling an absent CpG: the mean of the clock's present

CpGs in the data, which FALCONAge used until the reference values existed, and the clock's reference value, which it uses now.

clock	platform	probes kept	pooled fill	reference fill
hrsinchphenoage	EPICv2	874 / 959	+15.9 y	+0.88 y
yingcausage	EPICv2	492 / 585	+14.8 y	−6.1 y
hannum	EPICv2	64 / 71	−11.5 y	+1.03 y
lin	EPICv2	96 / 99	−9.3 y	+0.84 y
yingdamage	EPICv2	1009 / 1089	+9.1 y	+0.66 y
horvath2013	EPICv2	339 / 353	−2.5 y	−1.3 y
zhangblup	EPICv2	286,843 / 319,607	−0.46 y	−0.46 y (no fill)
yingdamage	EPICv1	1089 / 1089	0.00 y	0.00 y
vidalbralo	either	8 / 8	0.00 y	0.00 y

Most of what the pooled column prices is the fill, not the loss. A pooled value sits near 0.5, and clock CpGs are often near 0 or 1: `lin` lost three probes of ninety-nine and moved nine years. Filled from the reference, the same masks move `hannum`, `lin`, `hrsinchphenoage` and `yingdamage` by about a year or less; over the 19 rows on the years scale whose fill changed, the median absolute shift fell from 3.36 to 0.66 years, and 38 of the 45 rows with a reference moved less. Those numbers are out of sample: the blood reference is a mean over healthy controls in this corpus, so each dataset was scored against it rebuilt without that dataset (scored in-sample, `hannum` read −0.40 rather than +1.03). `yingcausage` still moves six years, its 93 lost CpGs carrying weight that no fixed value stands in for.

`zhangblup` fills nothing, because an absent probe drops out after the per-sample standardisation, and loses ten percent of a 319,607-probe model for under half a year at the median: the graceful-degradation property published for PC clocks, reproduced here in a different architecture. Its shift has two parts. Scored on the same samples, the lost probes alone move the estimate by −0.64 y at the median, and restandardising over the EPIC v2 probe set by +0.15 y. And the two rows at exactly zero are the internal check: a clock that loses nothing must shift by nothing, and does.

What the table does not cover, so it is not read as covering it. 86 rows: 43 clocks × two targets, EPIC v1 and EPIC v2. For the six deconvolution entries the shift is not probe loss alone: a 450K matrix is deconvolved with the 350-CpG legacy IDOL table and an EPIC one with the 450-CpG table, so their rows price a change of array, which is what a user moving between arrays sees (a median shift of at most 0.034 in a cell-type proportion). 450K is the *reference* the shift is measured from, so it cannot appear, its shift is zero by construction, not by measurement. 27K is absent for a different reason: masking a 450K matrix to 27,578 probes leaves most of these clocks below the coverage floor, so what would be measured is the shift of a clock that would have been refused anyway. And the reference cohort is 206 blood-derived samples of adults, whole blood and leukocytes as well as PBMC and sorted T cells; a shift measured there is a reasonable prior for another blood cohort and not a correction factor for a different tissue.

The shift is **reported and never applied**. An automatic offset would be a second number nobody can trace, which is the failure the coefficient digests exist to prevent.

Platform loss or probe QC. A clock CpG can be absent for two reasons, and only one of them is permanent. If the array does not carry it, nothing recovers it. If the array carries it and the data lacks it, it was measured and then removed, almost always by probe filters (SNP-affected, cross-reactive, failed detection) that belong to an epigenome-wide association study and not to clock scoring. With the platform's manifest

in the cache, `probe_loss()` counts the second kind in `n_removed_by_qc` and `qc_removed_mass`, and `score()` warns (category `probe_qc`) once the removed CpGs carry 1% of a clock's |coefficient|. The manifest is never downloaded during scoring; `fetch_manifest(platform)` caches it once. On one cohort's QC-filtered EPIC v1 betas this separates the two cleanly: all 30 of DNAmTL's absent CpGs were removed by QC (11.1% of its weight), while Horvath's 27 absent CpGs split into 8 removed by QC (2.7%) and 19 the array never carried. The remedy is to score clocks on normalised but unfiltered betas.

19.6 A reported score must not move when the next plate arrives

ComBat estimates its location and scale from every sample at once. Add a batch, re-run, and every previously corrected value changes, measured directly on epigenetic ages as a mean shift of 0.077 to 0.39 years with a **maximum of 2.20 years** (iComBat, PMC12495439).

A 2.2-year retrospective change in someone's reported biological age, caused by a stranger's sample being run three months later, is not a rounding difference. The individual-level critique of these measures lists "longitudinal comparability without re-running all samples" as a prerequisite for any clinical use, and this is precisely it.

`fit_batch_reference()` freezes the per-probe grand mean, covariate effects, pooled variance **and the empirical-Bayes hyperparameters** on a reference cohort. `apply_batch_reference()` standardises each later batch against those frozen values and estimates only that batch's own additive and multiplicative effects. A batch's corrected values then depend on that batch and the reference alone. There is no path through the arithmetic by which a new plate could move an old one, and the test asserts bit equality rather than closeness.

Freezing the hyperparameters is the part that is easy to get wrong. Leave them free and adding a batch re-estimates the prior, which moves every other batch's shrunk estimate, and the whole exercise is ComBat with extra steps.

Two designs are refused: a batch with fewer than eight samples, and one where the variable of interest is nested inside batch. The second matters because correcting a confounded design removes the effect along with the artefact and returns a clean-looking null, the one failure mode of batch correction that looks like success.

19.7 Before the first array, and after the last

`power()` answers "how many samples to see this effect on this clock", and needs no data file. It refuses to default the standard deviation, because n scales with its square and a guessed SD is a guessed answer reported to three significant figures. Where a reliability figure is available it splits the answer: at ICC 0.9, a tenth of the sample size exists only to average out the instrument. That is the arithmetic behind the published finding that the original clocks need 3–16 replicates per condition where their PC versions need 1–2.

`consensus()` implements the decision rule from *When to Trust Epigenetic Clocks* [77]. Re-analysing six intervention datasets, the authors found that in **five of six** exactly one clock reached significance, a first-generation clock every time, and **four of those five lost it** under multiple-testing correction. In no case did the PC version of the same clock corroborate. Their conclusion, in their words: *a single significant clock after an intervention is likely a false positive.* The same group later widened this to 16 clocks and 94 further biomarkers across the 51 longitudinal studies of TranslAGE [78], which is where the clock grouping the FALCONAge report uses comes from.

That PC clause is the sharpest test in the paper, and by default FALCONAge cannot run it. The principal-component clocks are `licensed`: the authors' data file (`CalcAllPCClocks.RData`, linked from the PC-Clocks repository) carries no licence, so it is not ours to distribute, and a corroboration check with nothing to check against would be inert. `consensus()` therefore names each clock's counterpart in `high_reliability_partner`, reports agreement in `partner_corroborates`, and where the counterpart was not scored it says so in the verdict string rather than dropping the clause. `fa.registry.load().import_pc_clocks(path)` turns the check on: it reads a downloaded copy of that file and registers PCHorvath 2013, PCSkinAndBlood, PCHannum, PCPhenoAge and PCDNAmTL, each collapsed exactly to one weight per CpG with the authors' fill values for absent CpGs, and it reproduces the authors' `run_calcPCClocks.R` to 1e-10. PCGrimAge, eight sub-models on the components plus sex and age, needs a composite model class FALCONAge does not have yet.

So `consensus()` tests every scored clock, corrects across the whole set actually run, and returns a verdict rather than a table to pick from: `supported` needs significance across at least two generations including an outcome-trained clock and survival of Bonferroni; one lit clock is `unsupported` whatever its p-value. The verdict string always carries the counts it was computed from, a verdict without its arithmetic is an oracle, and that is the opposite of the posture of this package.

Repeated measures. An intervention usually measures the same people more than once, and treating each visit as a new person lets the spread between people swamp the change within them. `design="paired"` tests each person's change from one visit to another (`subject_col` names the person); `design="mixed"` fits `y ~ visit + (1 | person)` by REML, keeps a person who missed a visit, and takes any number of visits, testing them jointly. It reproduces `nlme::lme` on a design with missing visits: estimates to 1e-7, standard errors to 1e-6, and the containment degrees of freedom exactly (Pinheiro & Bates 2000). Both add `mdc95`, the minimum detectable change $1.96 \cdot \sqrt{2} \cdot \text{SEM}$ with the SEM the within-person SD across the visits (Weir 2005), and flag a significant mean change that is smaller than it: twelve people can make a group change of 1.6 SD of the change significant while no one of them could be shown to have changed. That SEM holds day-to-day biology and assay noise together; `technical_se()` gives the assay part alone, and `variance_components()` separates the two when there are technical replicates.

19.8 Evidence, attached to the score

`result.evidence()` returns what a value on this clock has been *shown* to predict, with effect sizes and DOIs. Seeded from studies that tested many clocks on one cohort under one protocol, so the rows are comparable to each other: GrimAge v2 at hazard ratio 1.54 per SD for all-cause mortality and 1.86 for cirrhosis, DunedinPACE at 1.44 for diabetes, and the observation that first-generation clocks account for about 5 % of significant disease associations across 174 outcomes in 18,859 people (Nat Commun 2025).

A row without a DOI is rejected at load time. An uncheckable effect size looks identical to a checkable one in a table, and is worth less than nothing.

The most useful row is not about any clock. Across 39 biomarkers in more than 20,000 people, a clock's chronological-age accuracy and its mortality prediction are **uncorrelated: R = 0.12, P = 0.67** (Nat Aging 2025). That is the empirical justification for `scale_type` and `LEGAL_OPS`, and it is the reason "which clock is most accurate" is the wrong question.

19.9 What was deliberately not built

Recorded so it does not appear by accident.

- **No silent correction.** Batch reference, platform offset and interval are all reported. A score adjusted by an untraceable factor destroys the provenance discipline that is the reason to use this package.
- **No percentile against an unnamed population.** The same person reads as accelerated or decelerated depending on the comparator; a percentile without a cited reference cohort means different things to different readers.
- **No diagnostic thresholds.** None exist in the literature. Nothing here is coloured red above a cut-off.
- **No LLM dependency.** A scoring library whose output depends on a model that changes weekly is not reproducible.

Chapter 13. Clinical chemistry and survival

Clinical chemistry

```
d <- read_clinical("nhanes.csv", units = list(
  albumin = "g/L", creatinine = "umol/L", glucose = "mmol/L", crp = "mg/dL",
  lymphocyte_percent = "%", mean_cell_volume = "fL",
  red_cell_distribution_width = "%", alkaline_phosphatase = "U/L",
  white_blood_cell_count = "10^3/uL", age = "years"))

score(d, clocks = "phenoage")
```

Units are required and never guessed. Albumin at 4.2 is g/dL and albumin at 42 is g/L; both are clinically normal, and PhenoAge fitted on one returns nonsense for the other. No range check can separate them, so the only correct behaviour is to ask.

KDM and homeostatic dysregulation have no coefficients of their own: each is defined relative to a reference cohort. FALCONAge ships one such fit, BioAge's NHANES III KDM (`fa.models.clinical.kdm_bioage`); for any other reference, fit it:

```
ref <- fit_kdm(nhanes3, markers = markers)
score(d, clocks = "kdm", reference = ref)
```

LinAge2

LinAge2 (Fong et al. 2025) is trained on survival rather than age: examination, blood and urine inputs in NHANES variable names and units, three scores from health questionnaire items, and a Cox model on principal components, one per sex, returned as the age at which a model on age alone gives the same risk. The inputs are named as in the authors' codebook, with age and sex in `obs`:

```
d = fa.FalconData(X=nhanes_style_inputs, obs=obs[["age", "sex"]],
modality="clinical_chemistry")
fa.score(d, clocks=["linage2"])
```

Cotinine (`LBXCOT`) in ng/mL is binned into the authors' four smoking categories. A missing lipid sets LDL to 0, as the authors' code does, and the result names the samples that happened to; any other missing input leaves that sample without a score. The parameters are what the authors' `linAge2.R` fits from the NHANES file it ships with, and FALCONAge reproduces its output to 5e-10 years on the 2,344 subjects of their test wave.

Blood-count ratios

```
fa.blood_count_ratios(cbc, units={"neutrophils": "10^9/L", "lymphocytes": "10^9/L",
                                  "monocytes": "10^9/L", "platelets": "10^9/L"})
```

NLR, LMR, PLR and SII from a complete blood count. Units are declared, as everywhere; counts are converted to 10^9/L. Percentages give the two ratios that do not depend on the total (NLR, LMR) and not the two that involve platelets, and a zero lymphocyte count gives no ratio rather than infinity.

Which marker carries an association

A composite clinical clock associated with an outcome does not say which of its markers carries the association. `fa.leave_one_marker_out(res, "kdm", data=d, reference=ref, test="outcome")` recomputes KDM, HD or PhenoAge without each marker in turn (KDM and HD are refitted; for PhenoAge the marker is held at the reference mean) and reports how far the score and the association moved.

Survival and association

Two figures that most aging-clock papers carry, and that need the analysis functions above.

Survival

Python

```
fa.cox_hazard(res, time_col="time", event_col="status")      # HR per unit (per year for
an age clock)
fa.plot.kaplan_meier(res, "horvath2013",
                     time_col="time", event_col="status")    # the figure
```

R

```
cox_hazard(res, time_col = "time", event_col = "status")
plot_kaplan_meier(res, "horvath2013", time_col = "time", event_col = "status")
```

The curves compare the fastest-ageing tenth with the slowest, not two halves. The middle of the acceleration distribution is where a clock discriminates least, so a median split dilutes the comparison; the tails are the published convention. The log-rank p is in the subtitle.

If the curves cross: the hazards are not proportional and the single hazard ratio beside them is an average over a period during which the ordering changed. That is worth saying in a caption rather than leaving for a reader to notice.

Association

Python

```
assoc = fa.associate(res, "crp", covariates=("age", "sex"))
fa.plot.volcano(assoc, fdr=0.05)
```

R

```
assoc <- associate(res, "crp", covariates = c("age", "sex"))
plot_volcano(assoc, fdr = 0.05)
```

The dashed line is the Benjamini-Hochberg threshold at the stated FDR, read off the `q` column rather than recomputed. This matters more than it looks: across a few hundred tests a raw 0.05 admits roughly one in twenty of the nulls, and drawing that line instead labels them significant.

Chapter 14. Benchmarking and validation

Benchmarking

```
res <- combine(list(r1, r2, r3), keys = c("GSE1", "GSE2", "GSE3"))
b   <- run_benchmark(res, condition_col = "condition", control = "HC",
                     dataset_col = "dataset")
b$summary
```

Median absolute error against chronological age is reported but never ranked on: a perfect chronological oracle would score best on it and be useless, because it would have no age acceleration left to detect anything with.

11. Benchmarking and validation

11.1 ComputAgeBench, the reference methodology

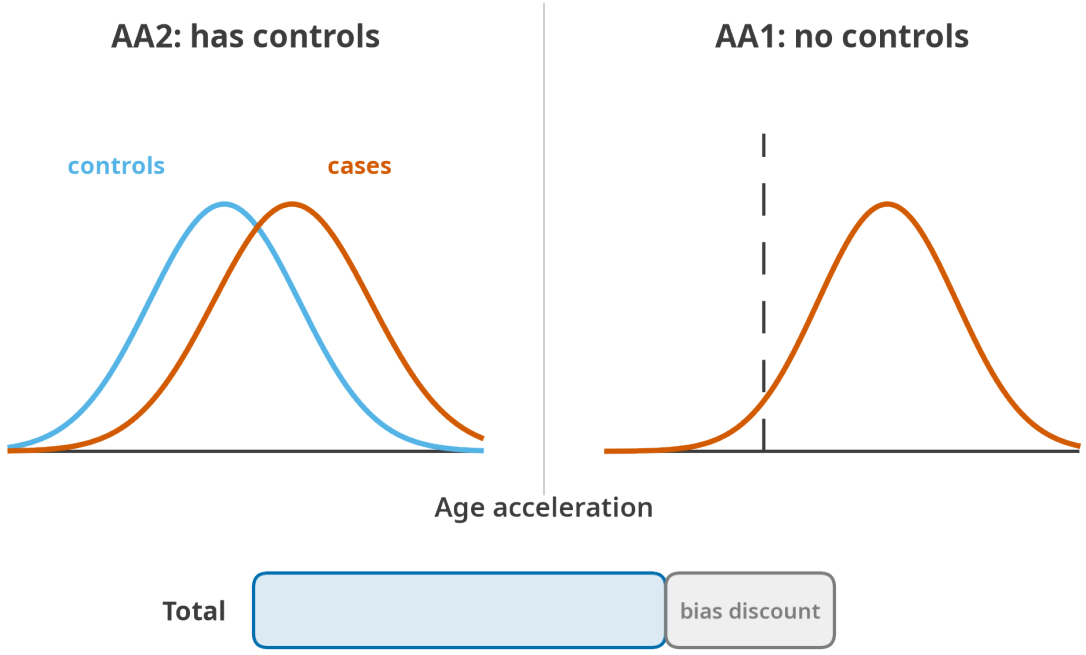


Figure 11: The two tests, and why the one without controls has its credit discounted

AA2 compares a condition group against its own healthy controls from the same study, so a clock that over-predicts everybody gains nothing from it. AA1 has no controls to compare against and asks only whether the condition group's acceleration sits above zero, which a uniformly over-predicting clock passes everywhere. The total discounts the AA1 credit by the clock's own median signed error on healthy

controls, measured on the same data, which is the correction that separates a clock that detects disease from a clock with a broken intercept.

`computage` (Kriukov, Efimov, Kuzmina, Khrameeva, Dylov; bioRxiv 2024.06.06.597715; KDD '25) [36].

Premise. MAE against chronological age is the wrong metric because a perfect chronological oracle is useless. Instead, a good clock must show higher age acceleration in people with *pre-defined aging-accelerating conditions* (AACs) than in healthy controls.

Data. Benchmark split: **10,404 samples, 900,449 CpGs, 65 studies**. Training split: **7,419 samples, 907,766 CpGs, 46 studies** (healthy people only). Hosted on Hugging Face at `computage/computage_bench` as per-study parquet plus a single `computage_bench_meta.tsv`.

Condition classes (ComputAge data_utils.py):

Class	Conditions
HC	healthy control
CVD	AS (atherosclerosis), IHD, CVA (stroke)
ISD	IBD, HIV, HIV_TB
MBD	XOB (extreme obesity), T1D, T2D
MSD	OP (osteoporosis), RA
NDD	MS, PD, AD, CJD
RSD	TB, COPD
PGS	CGL (congenital generalised lipodystrophy), WS (Werner), HGPS (progeria)

The full study table with GSE accessions, PMIDs, first authors, tissues, sample counts and HC availability is ComputAge BenchBSB_table.csv - 66 rows. Highlights: GSE56046 (n = 1,202, atherosclerosis), GSE42861 (n = 689, RA), GSE117859 (n = 608, HIV), GSE111629 (n = 572, PD), GSE59685 (n = 531, AD), GSE131752 (Werner, n = 48), GSE182991 (HGPS, n = 27), GSE214297 (CGL, n = 16).

Filters. Tissues restricted to Blood, Saliva, Buccal ("BSB"). Age limited to [18, 90], except progeria (PGS) which is exempt so that young progeria patients are retained.

Scoring.

```
AA2_score = number of AA2 datasets where FDR-adjusted p < 0.05
AA1_score = number of AA1 datasets where FDR-adjusted p < 0.05
MedAE     = median |prediction − age| over all healthy-control samples
MedE      = median (prediction − age) over healthy controls          # the bias
total     = AA2_score + AA1_score × (1 − max(0, MedE) / MedAE)
```

The bias penalty is the clever part: a clock that simply over-predicts everyone's age will pass AA1 trivially, so AA1 credit is discounted by how much of the clock's error is systematic positive bias (ComputAge benchmarking.py:427). GrimAge2 leads AA1 but carries a large positive HC bias, which the formula penalises.

Running it. ComputAge's `run_benchmark(models_config, experiment_prefix, output_folder)`, not `fa.run_benchmark`, where `models_config` has an `in_library` dict (clock name → params, notably `imputation`) and a `new_models` dict (name → path to a pickled sklearn estimator with `feature_names_in_`).

Outputs: AA2 and AA1 p-value CSVs, FDR-corrected boolean CSVs, MedAE CSV, bias CSV, four PDFs, and a pickled benchmark object.

11.2 Biolearn and the Biomarkers of Aging Consortium

Open-source Python library providing a unified interface over aging biomarkers plus a data library that loads GEO, NHANES and Framingham. Reference implementations for 39 biomarkers evaluated over >20,000 individuals; the framework paper reports integration of 17 additional GEO datasets covering 12,463 samples across tissues and age ranges. Version 0.7.0 released January 2025. The consortium runs a Synapse challenge series (chronological age, mortality, multimorbidity) with >$200k in prizes. ComputAge's `model_definitions` are explicitly ported from biolearn (ComputAge definitions.py).

11.3 TranslAGE - reliability validation

TranslAGE (Borrus et al., bioRxiv 2025.10.16.682960) harmonises 179 blood methylation datasets and precomputes 41 biomarker scores. On it, Sehgal et al. (bioRxiv 2025.10.13.682176) measured technical and biological ICCs for 18 clocks, pooled with Fisher-Z: technical reproducibility did not predict biological reliability, PCGrimAge was the only clock with biological ICC > 0.75, and the PC-based clocks, PCGrimAge and SystemsAge in particular, stayed technically reliable across slide positions and DNA extraction protocols.

11.4 Intervention responsiveness

In CALERIE, two years of calorie restriction (about 12% below baseline intake in practice) slowed DunedinPACE by 2 to 3% against controls (Waziry et al., *Nature Aging* 2023, doi:10.1038/s43587-022-00357-y), the first randomised trial to move a methylation clock. Ordering: DunedinPACE responds fastest, GrimAge needs longer or larger interventions, first-generation Horvath barely moves.

11.5 What a benchmarking module in the new tool should do

1. Chronological accuracy on healthy controls: MedAE, MAE, RMSE, r, ρ, slope, bias (MedE).
2. AA1 and AA2 with configurable distributional assumption and FDR correction.
3. Mortality: HR per SD, C-index, time-dependent AUC.
4. Reliability: ICC from user-supplied replicate structure, with Fisher-Z pooling.
5. Responsiveness: paired-design effect sizes with the measurement ICC reported alongside.
6. Cross-clock agreement: Spearman matrix, hierarchical clustering, UMAP.
7. A composite leaderboard using the ComputAgeBench total-score formula, with the components shown, never just the total.

Chapter 15. Beyond the clock score

Every clock in the catalogue models the *mean* beta at a set of sites and returns one number. That is not the only thing a methylation matrix can be asked, and on the field's own evidence it may not be the most informative thing.

Tong et al. [65] quantified how much of a clock is entropy rather than biology: roughly 66 to 75 per cent of Horvath2013′s accuracy against chronological age is reproducible by a purely stochastic model of methylation change, about 90 per cent for Zhang, and about 63 per cent for PhenoAge. The gradient is the interesting part. The more accurately a clock predicts chronological age, the more of it is drift.

This page covers five things FALCONAge computes that are not clocks: the disorder readouts, repertoire structure, where a clock's coefficient mass sits, reliability components, and mosaic ages within a tissue. None of them returns an age, none is gated by `scale_type`, and none is comparable between datasets processed differently.

Disorder: entropy, drift and the barometer

`falconage.disorder` holds four functions, and the division between them is about *what carries the number*, which is the distinction most easily lost.

Function	Returns	Per what
`fa.entropy`	how far the methylome sits from fully committed	sample
`fa.drift`	how far a sample sits from the cohort centroid	sample
`fa.noise_barometer`	summed per-site SD over age-variable sites	**group**
`fa.variable_sites`	which cytosines widen with age	site

Entropy

Normalised Shannon entropy over the beta matrix:

$$S = \frac{1}{N \log \frac{1}{2}} \sum_{i=1}^{N} \Big[\beta_i \log \beta_i + (1-\beta_i) \log(1-\beta_i) \Big]$$

Bounded 0 to 1. A site at beta 0 or 1 is decided and contributes nothing; a site at 0.5 is maximally uncertain and contributes its full share. So $S = 0$ is a methylome where every site is committed and $S = 1$ is one where none of them is.

```
import falconage as fa

data = fa.read("betas.csv")
ent = fa.entropy(data)        # entropy, n_sites
```

`n_sites` is not decoration. Entropy is a *mean over sites*, so a mean over different sites is not a comparison. The default restricts to sites covered in every sample for exactly that reason; set `complete_sites=False` to use each sample's own coverage and then read `n_sites` before comparing any two rows.

On the test corpus this lands at 0.695 with an SD of 0.007 across 825,177 complete sites, which is what a clean bimodal EPIC distribution looks like once it is summarised to one number.

Drift

The barometer is a property of a group, which leaves nothing to regress against an outcome. `fa.drift` is the per-sample analogue: the distance from the cohort centroid, computed **leave-one-out** so that a sample is never scored against a mean it helped compute.

```
dri = fa.drift(data)                          # drift from this cohort
dri = fa.drift(data, reference=young_mean)    # drift from a young reference
```

The leave-one-out default matters more than it looks. Scoring a sample against a centroid that includes it shrinks its own distance, and in a small cohort that shrinkage is large enough to invert a group difference.

Which sites actually widen

The barometer of Mei et al. [70] is defined over "cytosines whose variance rises with age", and that clause is the whole statistic. Summed over every site instead, the sum is dominated by sites that are merely variable, most of which have nothing to do with age.

`fa.variable_sites` is that selection step, split out so it can be inspected rather than assumed. The test is Brown-Forsythe, which is Levene's test using the median, across age bins. Bartlett's test is the more common choice in this literature and is the wrong one here: it assumes normality within each group, and beta values near 0 or 1 are visibly not normal, so Bartlett reports variance differences that are distributional artefacts.

A significant result says the variances *differ* between bins. It does not say variance *rises*, and a site whose variance falls is just as significant, so `direction` carries the sign of the trend and `rising` is the conjunction the barometer wants.

> ! What this looks like on a cohort that is too small
>
> Run on the test corpus, `sites_rising` under Benjamini-Hochberg is **zero**. Twenty-seven samples in three bins of nine, against roughly 820,000 tests, has almost no power once the correction is applied.
>
> That is the correct answer, not a gap in the implementation. A barometer that means anything needs a cohort in the hundreds. `fa.noise_barometer` accepts an explicit `sites=` list so a selection made elsewhere can be reused, which is also the only way the statistic transfers between cohorts.

Repertoire structure

`falconage.immune` addresses a covariate no published blood clock carries.

Deconvolution answers *how many cells of each type*. It says nothing about *how many distinct clones those cells represent*, and the two are different measurements: two donors with an identical CD8 memory fraction can differ by orders of magnitude in clone count.

Why clone count should move a clock at all

Take one cell type holding fraction f of the sample, split into clones of relative size w_k with $\sum_k w_k = 1$. Give clone k a somatic offset δ_k from its lineage mean at some site, with mean zero and variance σ^2. The compartment's bulk beta is the size-weighted mean, so

$$\operatorname{Var}\left[\sum_k w_k \delta_k\right] = \sigma^2 \sum_k w_k^2 = \frac{\sigma^2}{N_{\text{eff}}}, \qquad N_{\text{eff}} = \left(\sum_k w_k^2\right)^{-1}$$

and $\sum_k w_k^2$ is exactly the Simpson index. The variance a clonal compartment contributes to bulk methylation is inversely proportional to its **effective number of clones**, and clone count enters through that quantity and no other. Three equal clones carry one cell's somatic state at full weight; ten thousand clones average it to nothing.

This is why the effect is invisible to composition adjustment: f can be identical between two donors while N_{eff} differs thousand-fold, and every published blood clock sees only f.

The myeloid version of the question is settled. Nachun et al. [71] measured clonal hematopoiesis against six clocks and found 1.31 years of acceleration on GrimAge to 3.08 on extrinsic EEAA. The lymphoid analogue has not been asked.

Measuring it

```
div = fa.repertoire_diversity(clones, sample_col="sample_id", count_col="count")
data.obs = data.obs.join(div)

res = fa.score(data, clocks="compatible")
acc = fa.acceleration(res, adjust=["clonality", "cd8mem_fraction"])
```

`fa.acceleration` already accepts arbitrary `obs` columns as covariates, so the incremental test is one call. The question is entirely whether clonality adds variance beyond cell fraction.

Depth is the trap. Richness and Shannon both rise with sequencing depth, so an unrarefied comparison between a deeply and a shallowly sequenced donor measures the library rather than the donor. Pass `rarefy="min"` to subsample every donor to a common depth by multivariate hypergeometric draw, which is what a shallower run would actually have produced; sampling with replacement inflates richness at low depth, in the same direction as the bias being corrected. `clonality`, `simpson` and `effective_clones` are far more stable and are the ones to trust when rarefaction is not possible.

Simulating it

The mechanism arm needs no paired cohort. `fa.simulate_clonality` generates bulk methylomes at **fixed** cell fractions and varying clone structure, so anything a clock does in response is clone structure alone.

```
sim = fa.simulate_clonality(
    reference, {"CD8mem": 0.3, "Neu": 0.7},
    clonal_types=["CD8mem"],
    clone_sizes=[fa.immune.zipf_clone_sizes(n, alpha=1.0)
                 for n in (2, 20, 200, 2000)],
    n_replicates=20)
res = fa.score(sim, clocks="compatible")
```

`reference` is sorted-cell mean beta, features by cell type. The six-cell IDOL library that the bundled deconvolution entries use (FlowSorted.Blood.EPIC, GPL-3) is one such table: `fa.registry.load().deconvolution_table("deconvolutebloodepicbcell", "EPICv1")` returns it, 450 CpGs by six cell types.

> **i** What the simulation returns on the test corpus
>
> Across 42 clocks scored on 48 simulated samples at identical cell fractions, the median log-log slope of score spread against effective clone count is **-0.494**, interquartile range −0.559 to −0.415. The derivation predicts −0.5 and was not fitted to the simulation.
>
> Among the clocks reporting years, `dnamphenoage` moves **4.35 years** between an effectively-two-clone compartment and a diverse one, and `horvath2013` moves 1.54 years. Both fall below a year once the compartment holds a few hundred effective clones.
>
> Cell-type fractions were identical for every one of those samples, so composition adjustment cannot remove any of it.

Where a clock's weight sits

A clock is a weight vector over CpGs, and two questions about that vector need no samples at all.

Is this clock exposed to genotype? If a clock CpG is under strong meQTL control and the allele frequency at that variant differs between ancestral populations, a systematic difference in clock output between ancestry groups follows from genotype frequency alone, with no difference in aging. Cruz-Gonzalez et al. [75] established that the major clocks lose accuracy in admixed individuals; whether the size of that loss tracks each clock's meQTL exposure is open, and half of that measurement is a property of the coefficients.

Is this clock regulatory? Patel et al. [76] report that most CpGs used by established clocks do *not* overlap known transcription factor binding sites. Reproducing that against any annotation class is the same computation with a different list.

```
mass = fa.coefficient_mass({"meqtl": godmc_cpgs, "enhancer": enhancer_cpgs})
```

Why mass and not count

Counting annotated sites answers the wrong question. A clock's behaviour is driven by its heavy features: a 353-CpG clock with two enormous weights and 351 small ones behaves like a two-CpG clock. The statistic is therefore the share of $\sum_j |w_j|$ on annotated sites, with the plain count returned beside it. **When the two diverge, the annotated sites are the heavy ones, and that divergence is the finding.**

Run on the registry with each clock's own CpG set as the annotation, which needs no external file, `horvath2013` shares 11.6 per cent of its *sites* with `dnamphenoage` but 19.6 per cent of its *mass*. `weidner` shares two of its three sites with `dnamphenoage` and 89 per cent of its mass. Two clocks that share their heaviest features will agree with each other for reasons that have nothing to do with either being right.

FALCONAge bundles no annotation list. meQTL catalogues and regulatory builds are large, versioned and separately licensed, and freezing one into a wheel would make it silently stale. GoDMC and the EPIGEN meQTL database are the usual sources for the first question, the Ensembl Regulatory Build or

a Roadmap chromatin-state track for the second. `.attrs["unmatched"]` reports how many supplied ids matched nothing, which is the check that catches an EPIC v2 list with `_TC21` suffixes meeting bare `cg` probe ids.

Reliability: trait, state and technical

A published ICC answers "how reproducible is this clock across a cohort". Someone asking "has my number moved" needs a different quantity, and the gap between the two is why clock ICCs in the literature read as reassuring while replicate disagreement of up to nine years is also true [68].

`fa.variance_components` separates them, given repeated measurement:

$$y_{ijk} = \mu + a_i + s_{ij} + \varepsilon_{ijk}$$

for person i, occasion j, replicate k, fitted by Henderson's method of moments. Two ICCs come back and the gap between them is the point:

`icc`
: On raw scores. Comparable to what the literature publishes.

`icc_age_adjusted`
: The same after chronological age is regressed out. A cohort spanning decades holds most of its between-person variance in the age term, so a raw ICC largely reports that the clock tracks age, which was never in question. The adjusted figure is between-person reliability among people of the same age, which is what any individual-level claim rests on.

A clock whose raw ICC is 0.98 and whose adjusted ICC is 0.4 is not a precise instrument. It is a cohort with a wide age range.

```
vc = fa.variance_components(res, subject_col="donor", occasion_col="visit")
vc.table                       # var_trait, var_state, var_tech, icc, icc_age_adjusted
vc.replicates_needed(0.9)      # per occasion, to reach ICC 0.9
```

With one draw per visit and no technical replicates, which is the usual longitudinal design, day-to-day change and assay noise cannot be told apart: `var_state` then holds both, `var_tech` is NaN, `icc` is the test-retest reliability across visits, and `vc.design["within_person"]` says so.

`replicates_needed` returns `inf` where no number of replicates gets there, and that is the informative case rather than an error. Replicates average away the *technical* term only, so a clock whose day-to-day within-person variance already exceeds the budget cannot be rescued by running the array again. The honest answer is then a second draw, not a second scan.

Components are not clipped at zero. A negative one means the within-level spread exceeded the between-level spread, which is real for a small design or a clock measuring nothing in this cohort, and rounding it to zero disguises both.

Mosaic ages within a tissue

`fa.models.single_cell.scage` gives an age per cell [28]. A bulk clock already reports the mean, so the quantity worth having is the *shape* of the distribution underneath it: a tissue whose cells are uniformly

middle-aged and one holding a mixture of young and very old cells have the same bulk methylation and, plausibly, very different biology.

The obstacle is that single-cell methylation coverage is sparse, so each per-cell age rests on few sites and carries large error. Observed spread is therefore always positive, and reporting it as heterogeneity is the mistake `fa.mosaic` exists to avoid.

The null is stated plainly: **every cell in the group has the same true age, and all observed spread is estimation error**. It is simulated by drawing each cell from a normal centred on the group mean with that cell's own standard error, taken from the profile-likelihood width `scage` already returns.

```
ages = fa.models.single_cell.scage(cells, reference)
mos  = fa.mosaic(ages, group="tissue")
```

A small `p_excess` is evidence of genuine mosaicism. A large one says the data cannot distinguish a mosaic tissue from a uniform one measured badly, which at typical single-cell coverage is the usual answer and is worth reporting as such. `n_at_grid_edge` counts cells whose estimate landed on the boundary of the age grid, where the likelihood was truncated rather than maximised; those pile up and inflate the tails, so a group with many of them should have its skew and kurtosis ignored.

What none of this is

None of these functions is a clock, and none returns an age. They carry no coefficients, no `scale_type` gate applies, and their outputs move with normalisation, probe panel and detection-p filtering. The only defensible comparison is within one processed matrix.

That is a real constraint rather than a disclaimer. Entropy computed on noob-normalised betas and entropy computed on raw betas are different numbers for the same DNA, and neither is more correct than the other.

Chapter 16. Figures and reports

Reporting

Two documents, each one self-contained HTML file.

The one-page report is the scores, the run and a few figures, small enough to send:

```
write_results(res, "results/")
report(res, "report.html", group = "condition")
```

It is one file, inlined CSS, base64 figures, the table embedded, so it still works after somebody emails it, which is the only thing anybody does with a report.

The full run, step by step. `run_report()` runs quality control, scoring, technical error, prediction intervals, age acceleration, a consensus test across clocks when a group column is named, and the interpretation tables, and writes every deliverable into one directory: the same directory the `report` command writes. With `quarto = TRUE` it adds `falconage_report.qmd`, in which every file of that directory sits under the step that produced it, in the order a reader has to take them, because each step is only as reliable as the ones before it:

Step	What it holds
0. Provenance	versions, the coefficient file and SHA-256 behind each clock, device, warnings
1. Input and quality control	per-sample missingness and beta distribution; the QC figures
2. Coverage	per-clock coverage by feature count and by coefficient weight; clocks not scored, and why
3. Scores	the scores, what each category of clock measures, and how the clocks relate
4. Agreement with chronological age	error against age and whether it depends on age
5. Uncertainty	technical standard error and conformal prediction intervals
6. Age acceleration	residuals on age, for the scales where they are defined
7. Group comparison	the consensus verdict and the test for every clock
8. Interpretation	what each score means and the published evidence, with DOIs
9. Further outputs	every other file in the directory, named and sized

A file the report does not recognise still appears, in step 9, so nothing in the directory is left out. Tables expand, search, page and download as CSV; figures enlarge on click and download at full resolution. Rendered, it is one HTML file with every figure and table embedded: 3.7 MB for 16 samples and three clocks.

Python

```
fa.report.run_report(d, "results/", group_col="condition", quarto=True)
```

R

```
run_report(d, "results/", group = "condition", quarto = TRUE)
```

Command line

```
docker run --rm -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cpu \
  report betas.csv --outdir results/ --group-col condition --quarto
```

Rendering it. Quarto is not in the FALCONAge image. Installed it takes 423 MB, which would add about 17% to the 2.5 GB image for an optional document, so inside the image the source is written and the run says so. Render it with the Quarto container, from the output directory, and open `falconage_report.html`:

```
cd results
docker run --rm --user "$(id -u):$(id -g)" -e HOME=/tmp -v "$PWD:/w" -w /w \
  ghcr.io/quarto-dev/quarto quarto render falconage_report.qmd
```

Where Quarto is installed, `run_report()` renders it directly, and `quarto_report("results/")` (R) or `fa.report.write_quarto_report("results/", render=True)` (Python) rewrites the report for a directory that already exists.

10. Plots

10.1 The standard set

1. **BA vs CA scatter, faceted by clock, coloured by sex**, with a fitted line, the identity line, and Pearson r annotated in the corner. This is `plot_ba` (BioAge plot_ba.R), ggplot2, `geom_point(shape = 1)`, `geom_smooth(method = lm)`, `facet_wrap(~ method, scales = "free")`, colours `#7294D4` (men) and `#E6A0C4` (women).
2. **Pairs / correlation matrix of age accelerations.** Scatter plots below the diagonal, Pearson r above the diagonal colour-mapped, clock labels on the diagonal. `plot_baa` (BioAge plot_baa.R) implements this in base graphics with a custom `layout()` matrix so the diagonal cells can hold text.
3. **Hierarchical clustering + Spearman heatmap of clocks.** pyaging Figure 1a - 38 methylation clocks over ~13,000 samples; telomere predictors cluster together, chronological predictors separate from mortality predictors.
4. **UMAP of samples in clock space.** pyaging Figures 1b–e; reveals "ageotypes" where AltumAge (chronological) and GrimAge2 (mortality) disagree.
5. **Reprogramming / intervention timecourse.** pyaging Figure 1f, GSE54848, 39 clocks over OSKM days; most rejuvenate between days 10 and 20, telomere predictors go *up*, ENCen100 is flat. Any tool should make "run N clocks across a timepoint variable and plot trajectories" a one-liner.
6. **Benchmark grids.** ComputAgeBench `plot_class_bench` - a boolean matrix of clock × (dataset:condition:test) with per-condition-class column groups and a total column; `plot_medae` (MedAE bar chart, upper bound 18 y); `plot_bias` (median error, x-limits ±20 y). Saved as PDF at 180 dpi (ComputAge benchmarking.py:451).
7. **Forest plot of hazard ratios** per clock, per outcome, per cohort.
8. **Density / violin of age acceleration by group** (disease vs control, treated vs untreated).
9. **Organ or system radar/heatmap.** organAging uses heatmaps of organ ages against traits, medications, foods, professions, diseases; SystemsAge gives an 11-spoke profile.

10. **Expression-density QC plots at each preprocessing stage.** tAge's `plot_eset_density` is called after raw, RLE, log, scale, YuGene and centring - a cheap way to catch a broken normalisation.
11. **Volcano / GSEA plots** when linking clock features to biology.
12. **Spatial maps** of predicted age or age acceleration over tissue coordinates, plus proximity effect matrices (effector cell type × target cell type, coloured by t-statistic).
13. **Calibration plot** for probability-output clocks; **precision–recall curve** for classifiers on imbalanced test sets (CellBiAge's choice of AUPRC over AUROC).
14. **Missingness bar chart** per clock, percent of features absent, drawn before the results so the user sees which clocks are untrustworthy for their platform.

10.2 Plotting conventions worth adopting

- Always draw the identity line on BA-vs-CA plots.
- Never plot rate clocks (DunedinPACE) on the same axis as age clocks.
- Facet by clock generation, not alphabetically.
- Annotate every panel with n, r and the missingness fraction.
- Colour by group, shape by platform; keep the palette stable across the whole report.

10.3 What FALCONAge draws, and against which of the above

Thirty-one figures in Python, twenty of them also in R, drawn alike because both languages read their colours, titles, subtitles and one-line descriptions from the same `colorscheme.yaml`. The table lists those that answer the principles above; the gallery shows every one. Every function returns its data frame as well as its figure, which is how two rendering engines draw the same numbers.

Figure	Answers	From §10.1
`ba_vs_ca`	does this clock track age here, and with what bias	1
`bland_altman`	is the error age-dependent (regression dilution made visible)	-
`calibration`	is the predicted age calibrated across the age range	13
`acceleration_by_group`	do cases accelerate relative to controls	8
`acceleration_density`	the same, as distributions rather than summaries	8
`acceleration_heatmap`	clock × study, the whole pooled picture in one panel	-
`forest`	effect size with an interval, per clock	7
`clock_corr`	which clocks agree, clustered	3
`clock_chord`	how much of that agreement is shared CpGs rather than shared signal	-
`clock_radar`	one profile per group across clocks, on a common z scale	9
`clock_pca`, `clock_embedding`	sample structure in clock space; ageotypes	4
`clock_scatter_matrix`	pairwise acceleration, the `plot_baa` layout	2
`coverage_bar`, `missingness`	which clocks are scoring imputed values, read before the scores	14

Figure	Answers	From §10.1
`beta_density`	is the beta distribution bimodal, or did normalisation break	10
`sex_check`	does the X/Y dosage match the recorded sex	-
`platform_comparison`, `study_comparison`	is the batch bigger than the biology	-
`benchmark_bars`, `benchmark_error_bias`, `benchmark_heatmap`	AA1/AA2, the MedAE–MedE trade-off, clock × dataset	6
`timecourse`, `trajectory`	clocks across a timepoint variable	5
`clock_atlas`	every clock, every pooled study, one figure	-

`clock_chord` is the one with no precedent in the list above and it earns its place: the correlation heatmap says two clocks agree, and the chord says how much of that agreement was built in by construction. A pair with a thick chord and a high correlation has told you much less than a pair with a high correlation and no chord at all.

A volcano plot of clock associations is implemented (`fa.plot.volcano`, R `plot_volcano`). GSEA (11) belongs to feature-level biology rather than to scoring, and spatial maps (12) wait on the spatial modality.

Two rules the code enforces rather than recommends. A figure whose data is empty, flat, or all zeros raises `NothingToPlot` and `save_all` records the reason, because an axis-free rectangle in a report reads as a measurement of nothing rather than as an absence of measurement. And the scale-type gate applies to figures too: a `pace_ratio` clock never reaches an acceleration panel.

Chapter 17. Figure gallery

Each of these is produced by `test/run_all.py` against the public test corpus and copied into `test/output_figures/gallery/`. They are regenerated from the code, so a figure here is what the current version actually draws – not a screenshot taken once.

The caption under each image is the same text the figure prints beneath its own axes. There is one copy of it, in `colorscheme.yaml`, so the website and the PNG cannot drift apart. The same file's `ui` block sets the colours and fonts of both HTML reports and of this site, so a figure, a report and a page of the documentation share one palette: Okabe-Ito for data, because group membership is carried by hue alone and has to survive colour-vision deficiency, and the logo orange, in a darker shade where it is text, only for links and controls.

ℹ Reproducing these

```
docker run --rm -v "$PWD:/work" -w /work falconage:1.0.0-cpu \
  python test/run_all.py
```

Which dataset each figure comes from, and why that one was chosen, is recorded in SOURCES.md.

Start here

One figure for the question everyone actually has: which of these algorithms detects anything.

Clock atlas: every algorithm across every pooled study

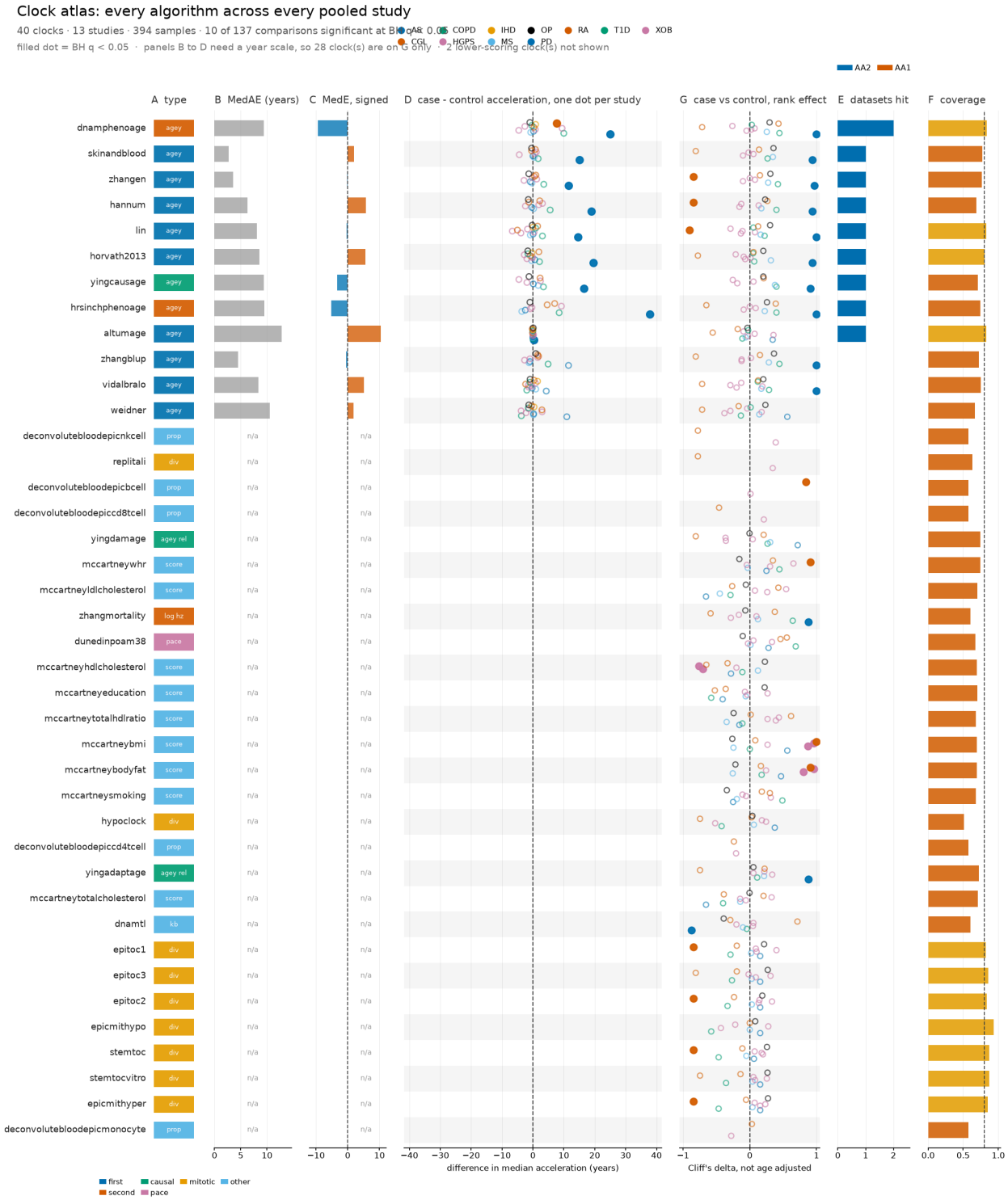


Figure 12: Clock atlas: every algorithm across every pooled study

Read panel D down the page. A row of hollow circles on zero is an algorithm that detected nothing in any cohort; filled dots pushed right are the ones that did, and a row with several says the effect held across studies rather than in one. Panels B and C are diagnostics for reading D, not scores: a clock that merely returned chronological age would be perfect in B, empty in D, and useless. Panel F separates the two ways of finding nothing – a clock that saw its features and detected no difference, from one that never had the features to look with. Panel G is for the clocks whose scale admits no acceleration at all – a pace, a telomere length, a division count – and asks the question their scale does allow: were cases and controls separated. Cliff's delta, so 0 is no separation and 1 is complete, and unlike D it is not adjusted for age, which matters when a cohort's cases are older than its controls.

`plot.clock_atlas()` in Python.

Is the clock calibrated

Before any biology: does the prediction track chronological age, and where does it drift.

Predicted against chronological age

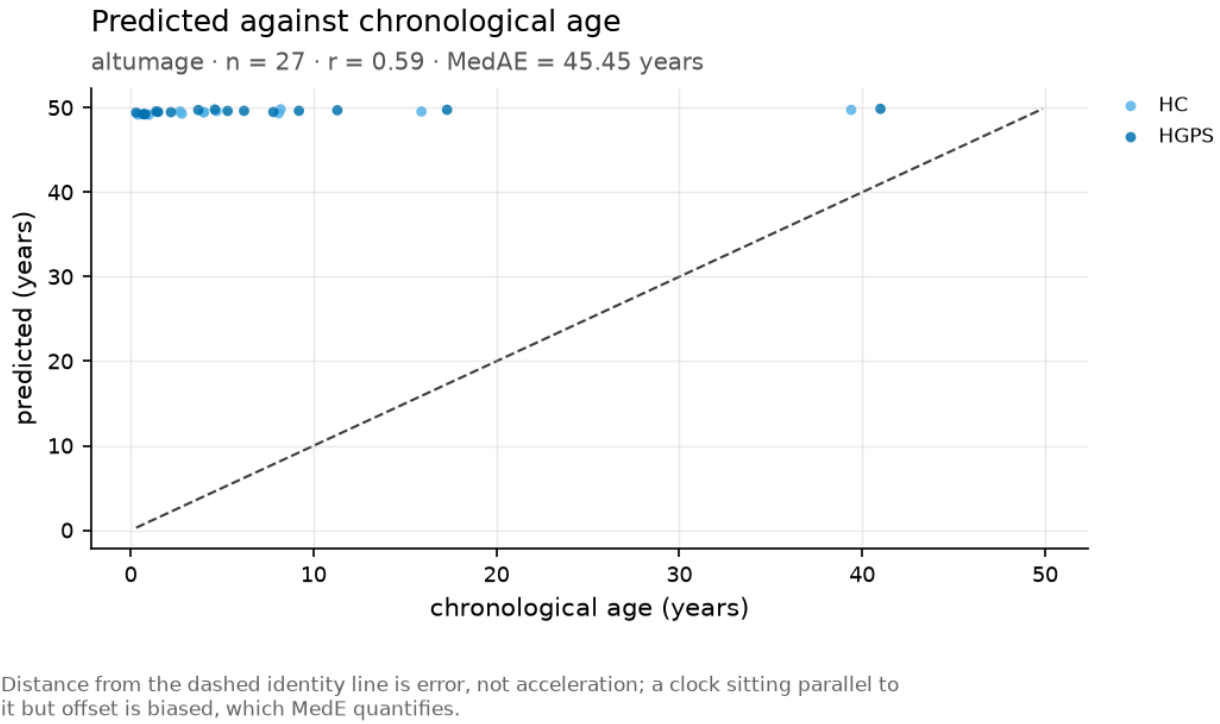


Figure 13: Predicted against chronological age

Distance from the dashed identity line is error, not acceleration; a clock sitting parallel to it but offset is biased, which MedE quantifies.

`plot.ba_vs_ca()` in Python.

Agreement across the age range

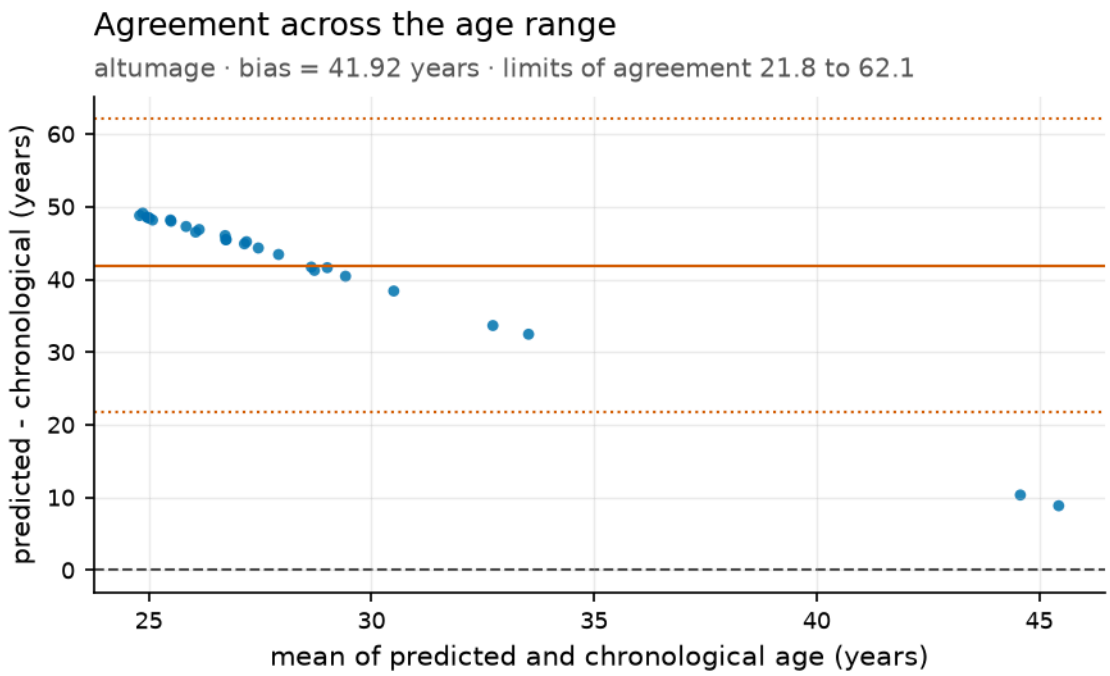


Figure 14: Agreement across the age range

A trend in this cloud means the clock's error depends on age, which a correlation coefficient cannot show and which invalidates a single MedAE.

`plot.bland_altman()` in Python.

Residual against chronological age

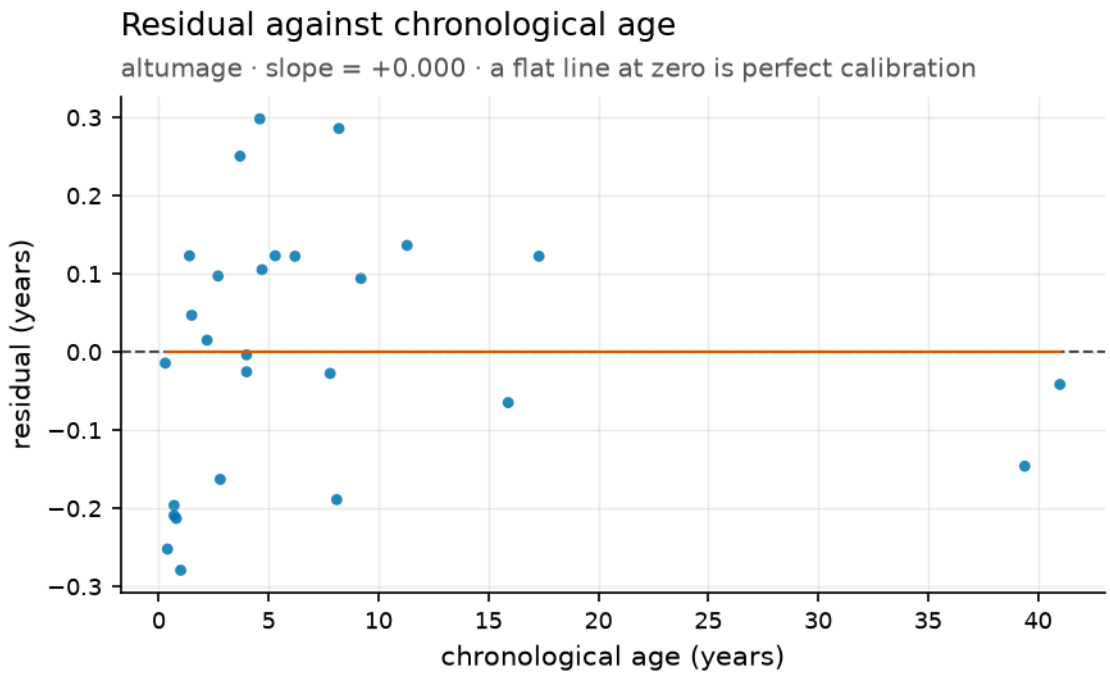


Figure 15: Residual against chronological age

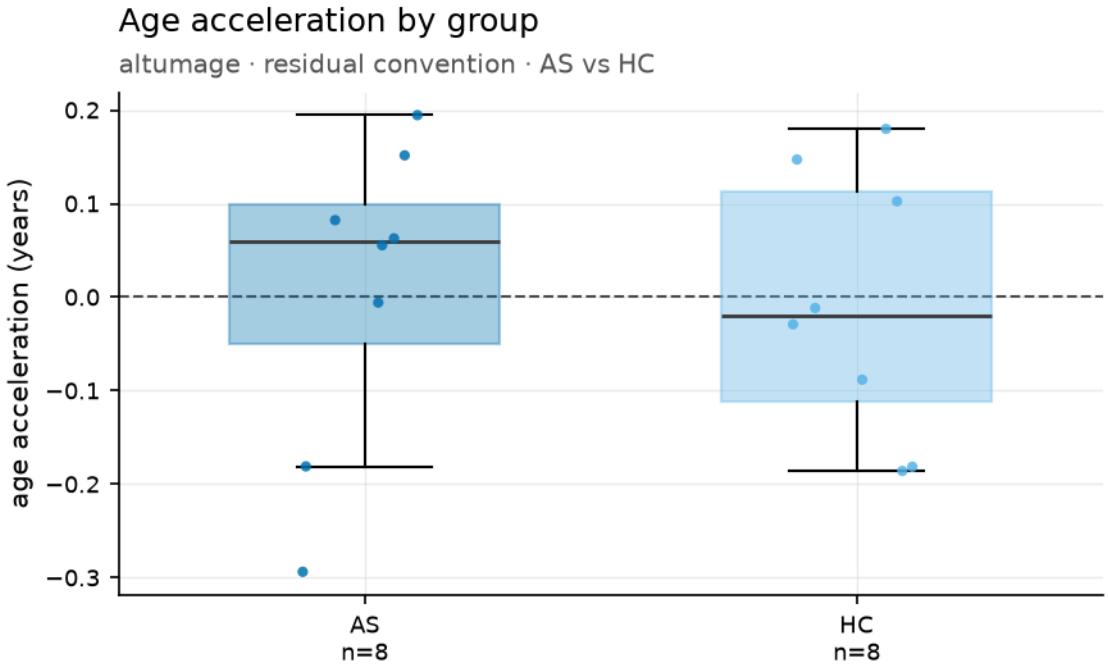
A negative slope is regression to the mean: the clock over-ages the young and under-ages the old, which is the most common artefact in this field.

`plot.calibration()` in Python.

Age acceleration

The residual after age is removed, by group and across studies.

Acceleration group

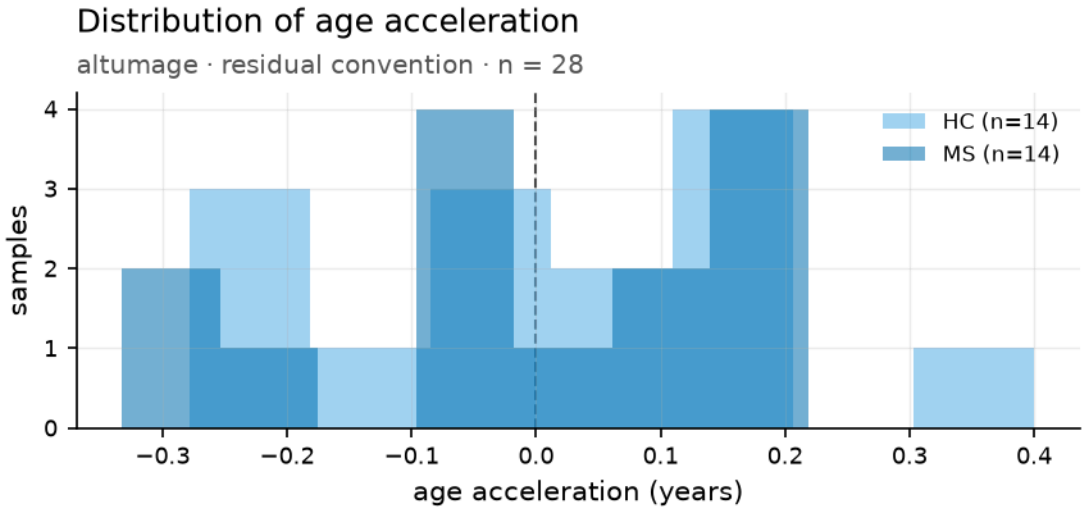


Each point is one sample; the box is the median and interquartile range. Read the separation between groups, not the distance from zero.

Figure 16: Acceleration group

`plot.acceleration_group()` in Python.

Distribution of age acceleration



Counted, not smoothed: a kernel density on a dozen cases invents structure that is not in the data.

Figure 17: Distribution of age acceleration

Counted, not smoothed: a kernel density on a dozen cases invents structure that is not in the data.

`plot.acceleration_density()` in Python.

Age acceleration across clocks and samples

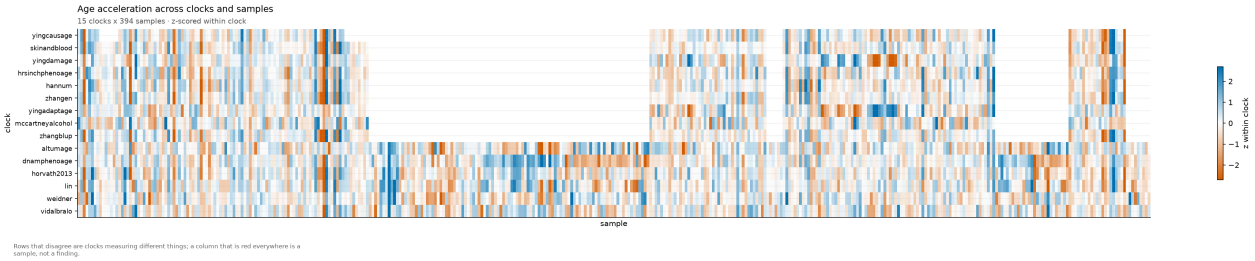


Figure 18: Age acceleration across clocks and samples

Rows that disagree are clocks measuring different things; a column that is red everywhere is a sample, not a finding.

`plot.acceleration_heatmap()` in Python.

Effect of condition on age acceleration

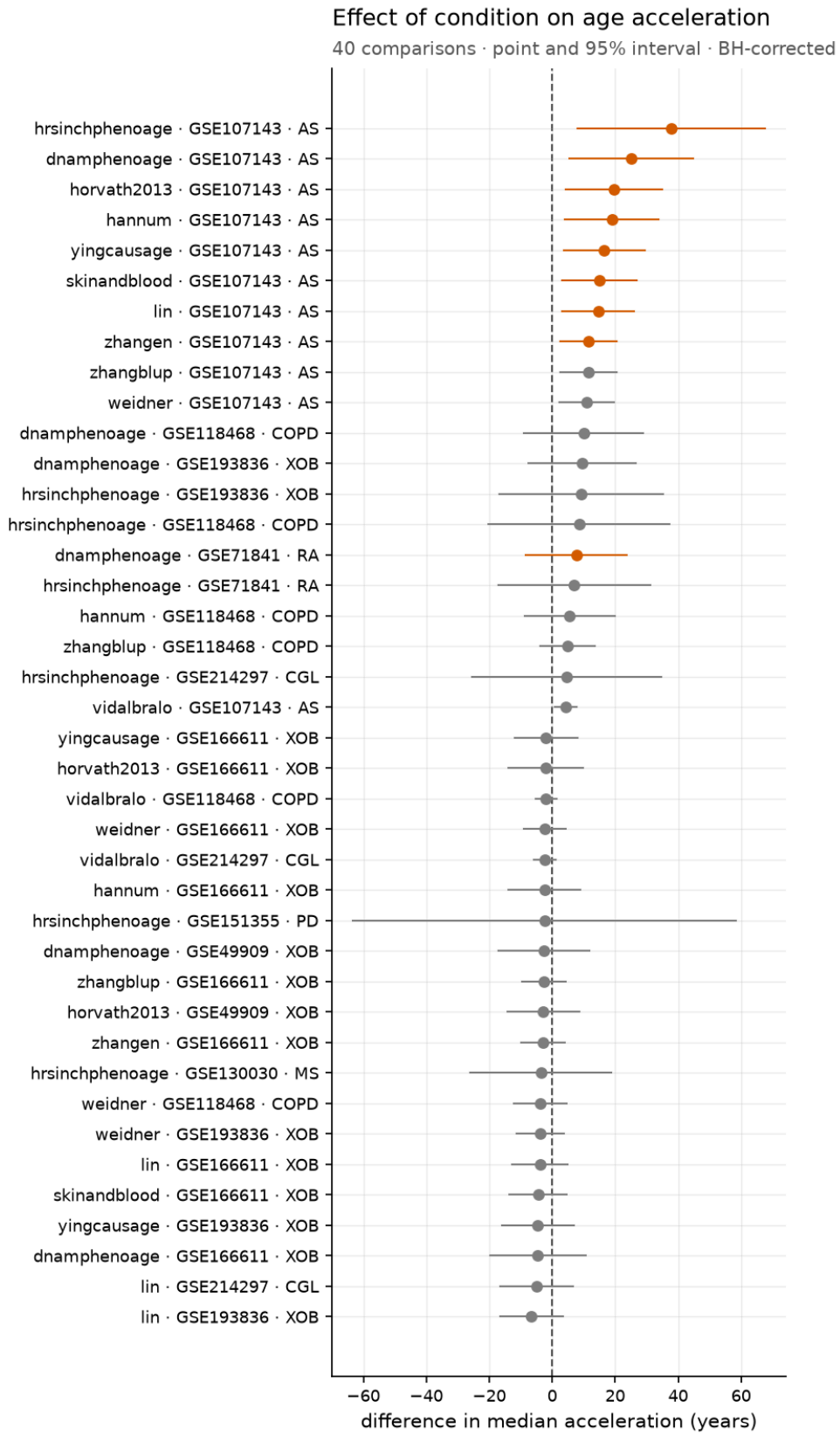


Figure 19: Effect of condition on age acceleration

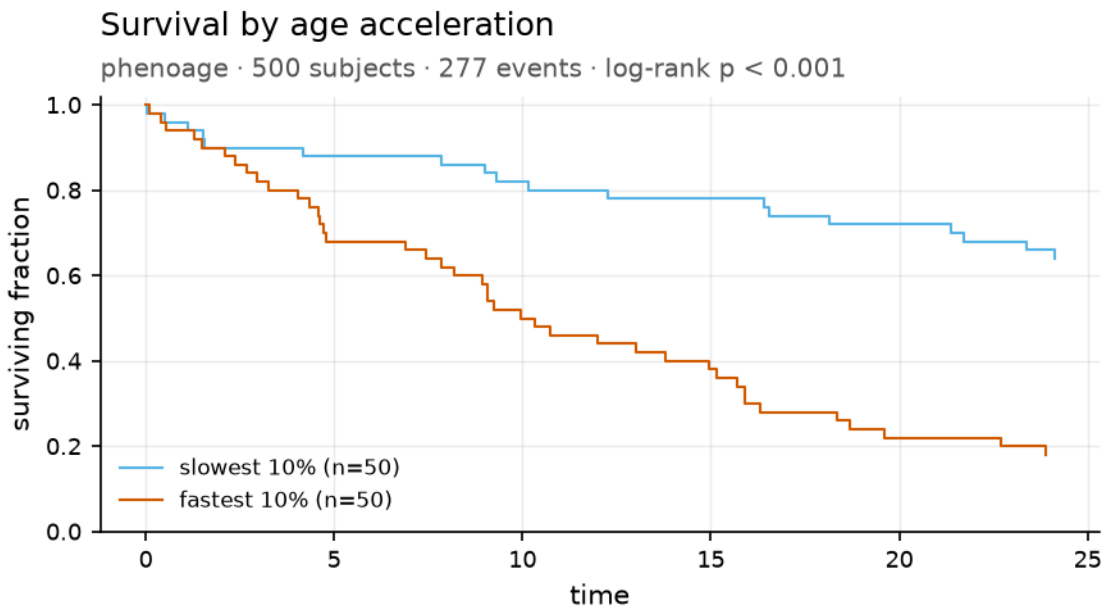
Intervals crossing zero are the honest majority. Width tracks sample size, so a wide interval far from zero is a small study, not a large effect.

`plot.forest()` in Python.

Outcomes

Whether acceleration predicts anything that happens to a person.

Survival by age acceleration



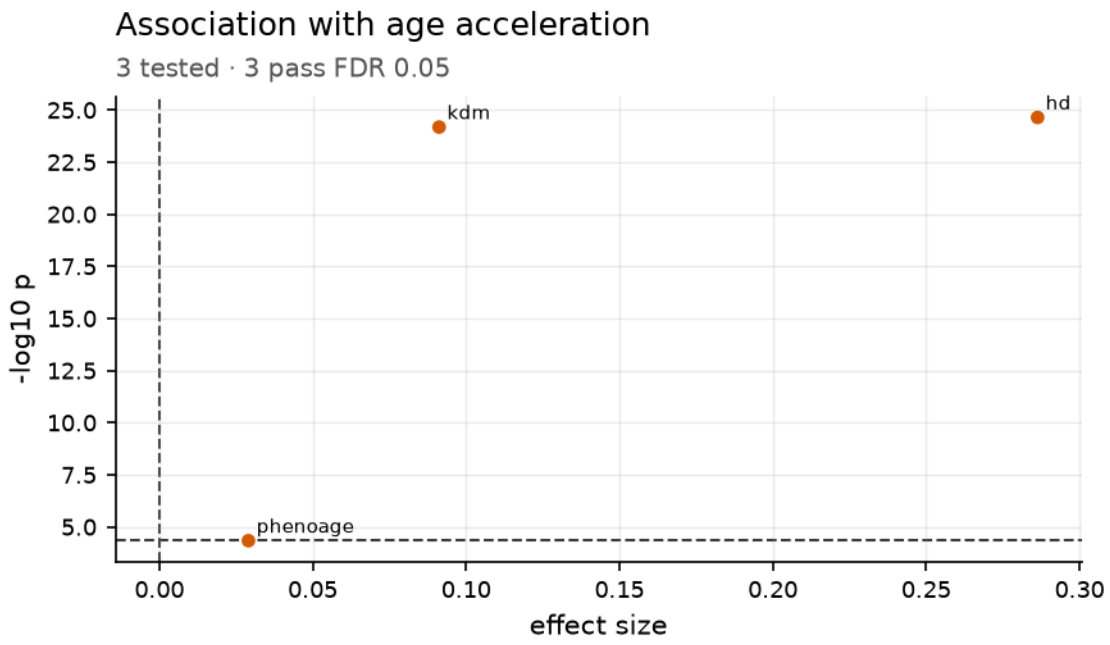
Strata are the tails of the acceleration distribution, not a median split: the published figure compares the fastest-ageing tenth with the slowest, because the middle of the distribution is where a clock has least to say. Curves that cross mean the hazard is not proportional, and the hazard ratio beside them is then an average over a period during which the ordering changed.

Figure 20: Survival by age acceleration

Strata are the tails of the acceleration distribution, not a median split: the published figure compares the fastest-ageing tenth with the slowest, because the middle of the distribution is where a clock has least to say. Curves that cross mean the hazard is not proportional, and the hazard ratio beside them is then an average over a period during which the ordering changed.

`plot.kaplan_meier()` in Python.

Association with age acceleration



Effect size against evidence. The dashed line is the Benjamini-Hochberg threshold at the stated FDR, not a raw p-value cut -- with thousands of tests the two differ by orders of magnitude. Points far right with small -log10 p are the usual trap: a large effect measured in few samples.

Figure 21: Association with age acceleration

Effect size against evidence. The dashed line is the Benjamini-Hochberg threshold at the stated FDR, not a raw p-value cut – with thousands of tests the two differ by orders of magnitude. Points far right with small -log10 p are the usual trap: a large effect measured in few samples.

`plot.volcano()` in Python.

How the clocks relate

Twenty clocks are not twenty measurements. These show how much they share – of their features, and of their answers.

Agreement between clocks

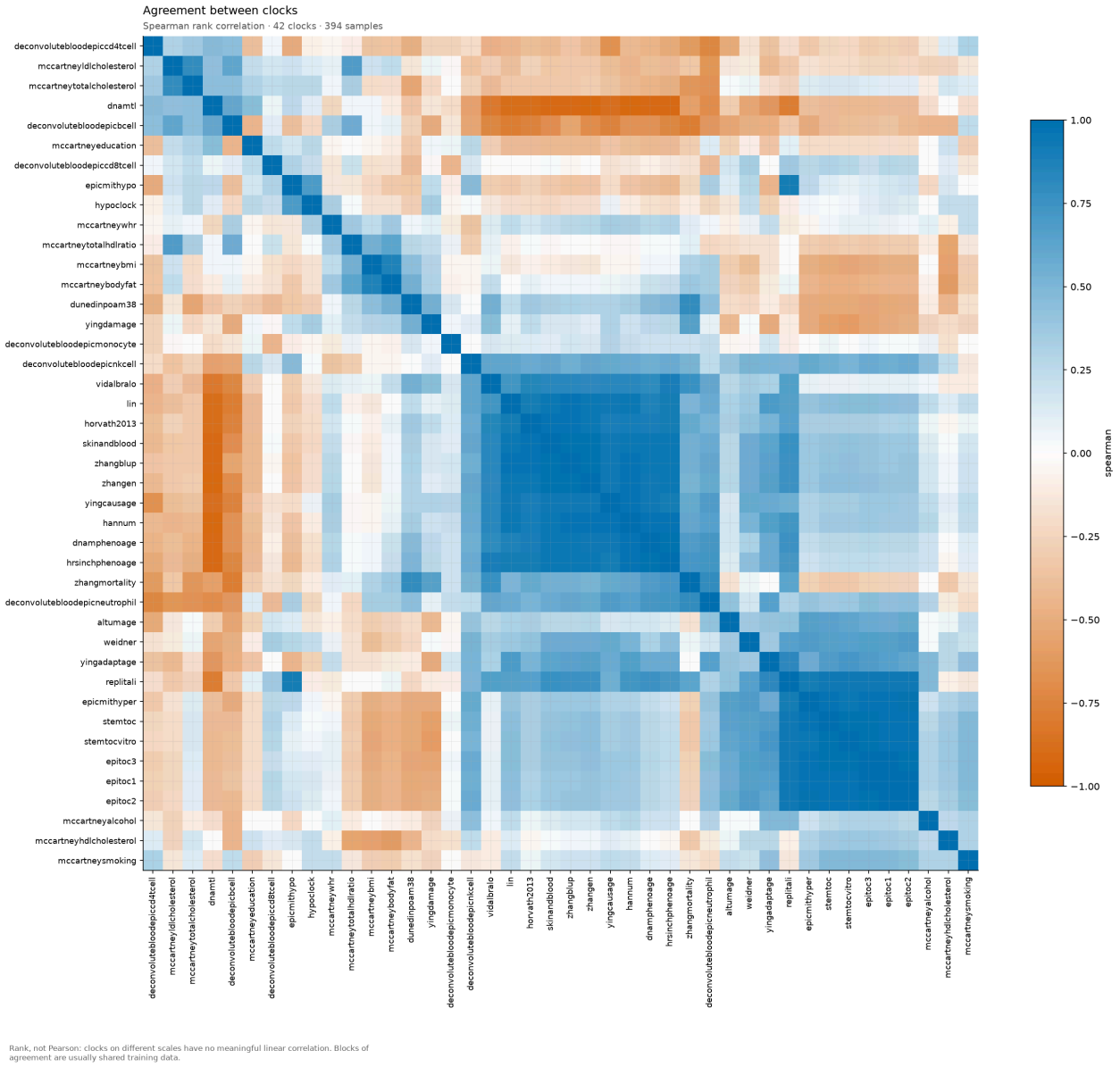


Figure 22: Agreement between clocks

Rank, not Pearson: clocks on different scales have no meaningful linear correlation. Blocks of agreement are usually shared training data.

`plot.clock_corr()` in Python.

Feature sharing between clocks

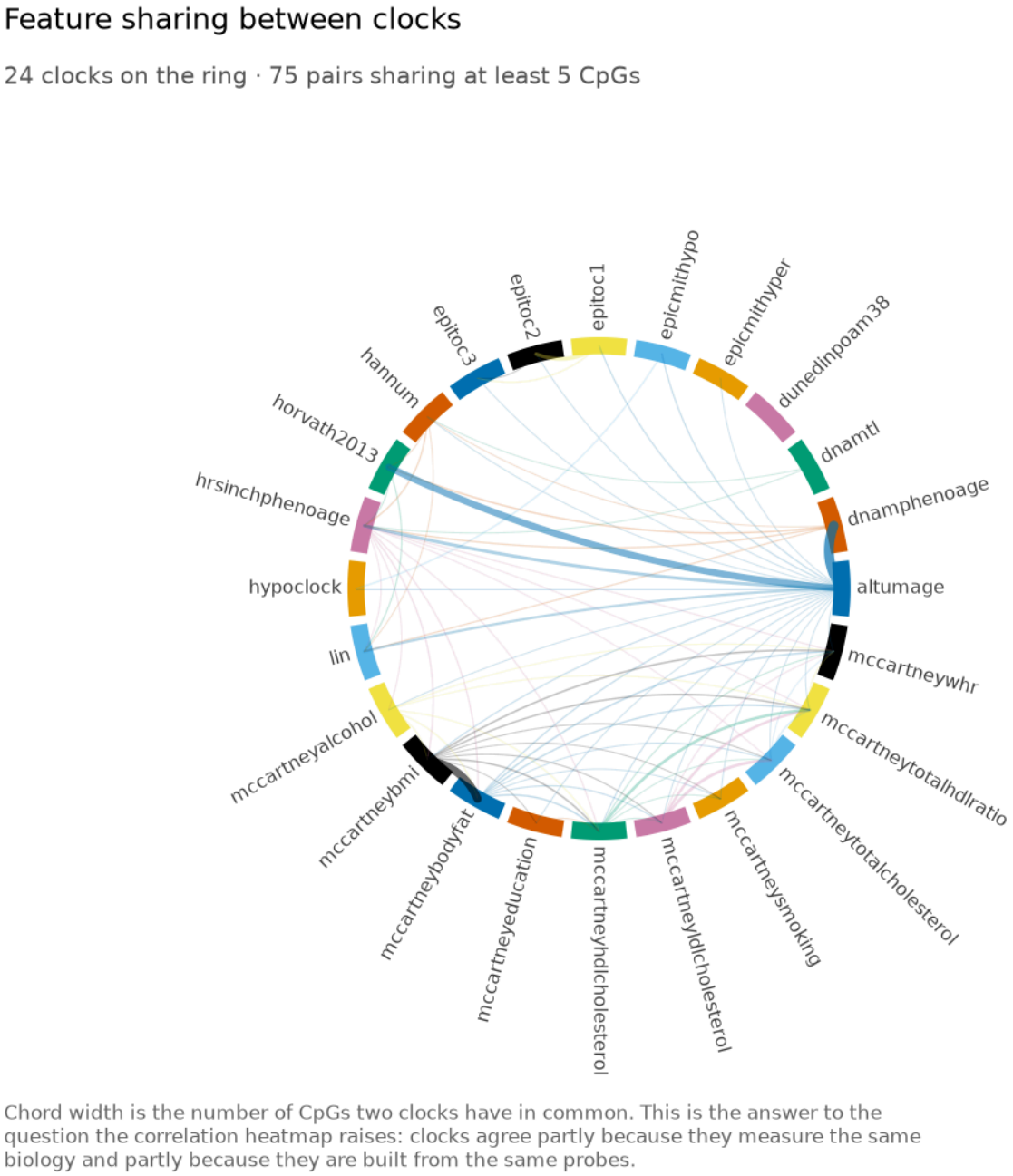


Figure 23: Feature sharing between clocks

Chord width is the number of CpGs two clocks have in common. This is the answer to the question the correlation heatmap raises: clocks agree partly because they measure the same biology and partly because they are built from the same probes.

`plot.clock_chord()` in Python.

Multi-clock profile

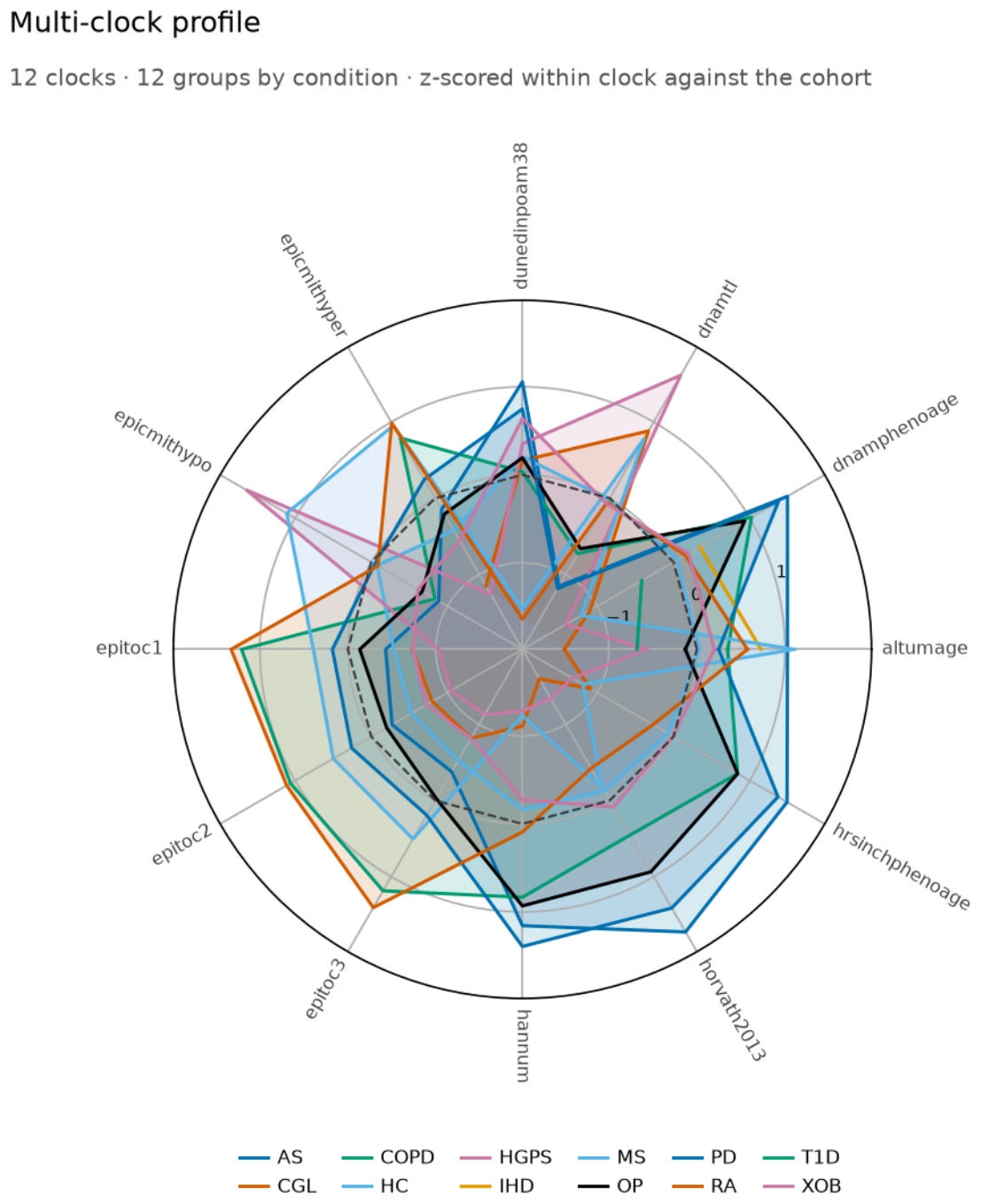
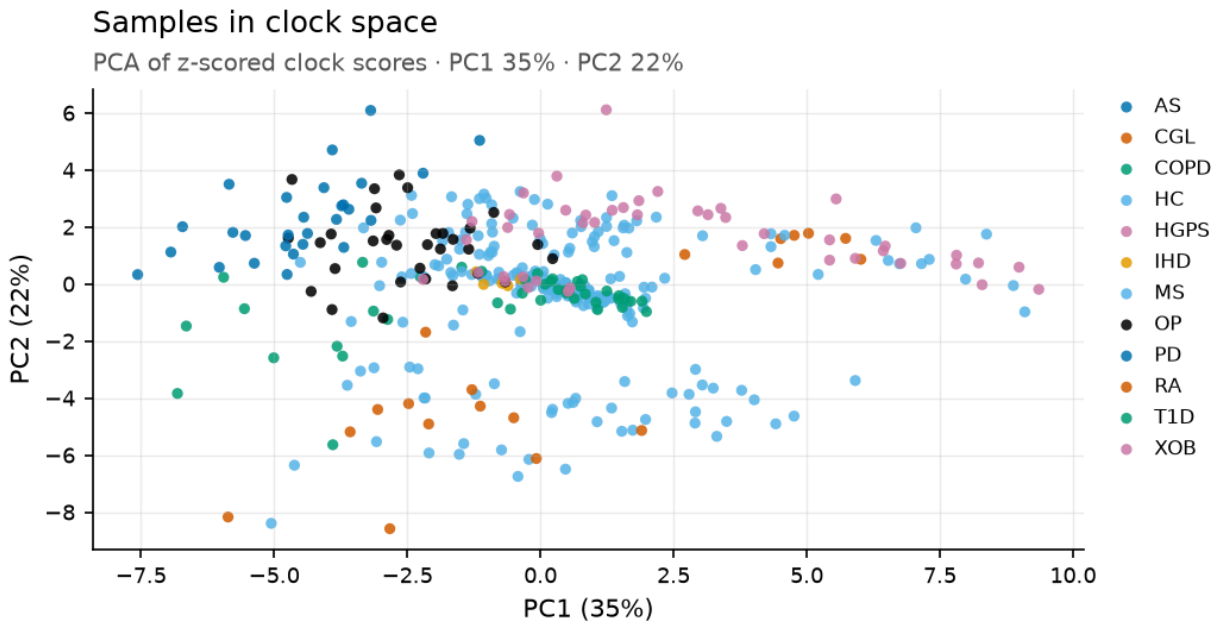


Figure 24: Multi-clock profile

One closed polygon per group, on a common z scale. A spike on one axis is one clock disagreeing with the rest, which is a finding about the clock at least as often as about the sample.

`plot.clock_radar()` in Python.

Samples in clock space



PC1 is almost always chronological age. Structure on PC2 that follows a batch and not a phenotype is a batch effect.

Figure 25: Samples in clock space

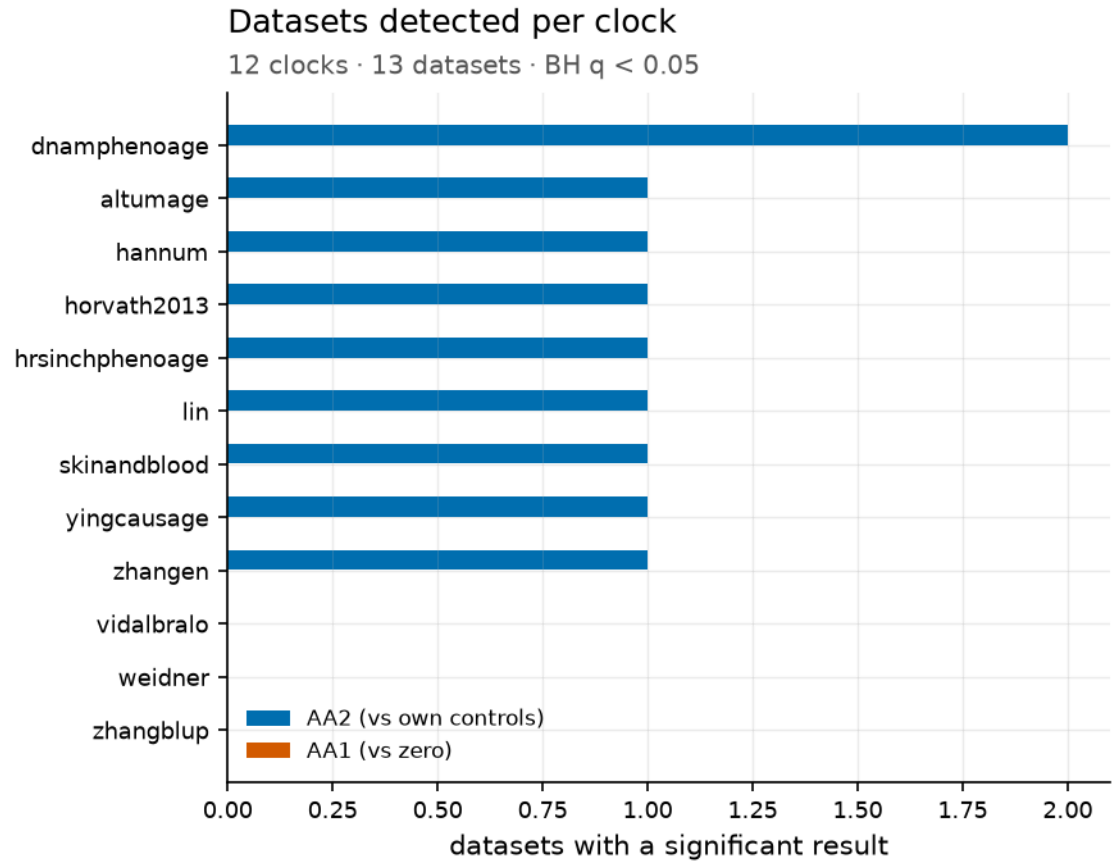
PC1 is almost always chronological age. Structure on PC2 that follows a batch and not a phenotype is a batch effect.

`plot.clock_pca()` in Python.

Benchmarking

AA1 and AA2 against the ComputAgeBench conditions, and the error-versus-bias tradeoff that decides the ranking.

Datasets detected per clock



AA2 compares cases with their own controls; AA1 tests case-only cohorts against zero. Ordered by the combined total, not by either alone.

Figure 26: Datasets detected per clock

AA2 compares cases with their own controls; AA1 tests case-only cohorts against zero. Ordered by the combined total, not by either alone.

`plot.benchmark_bars()` in Python.

Error against bias on healthy controls

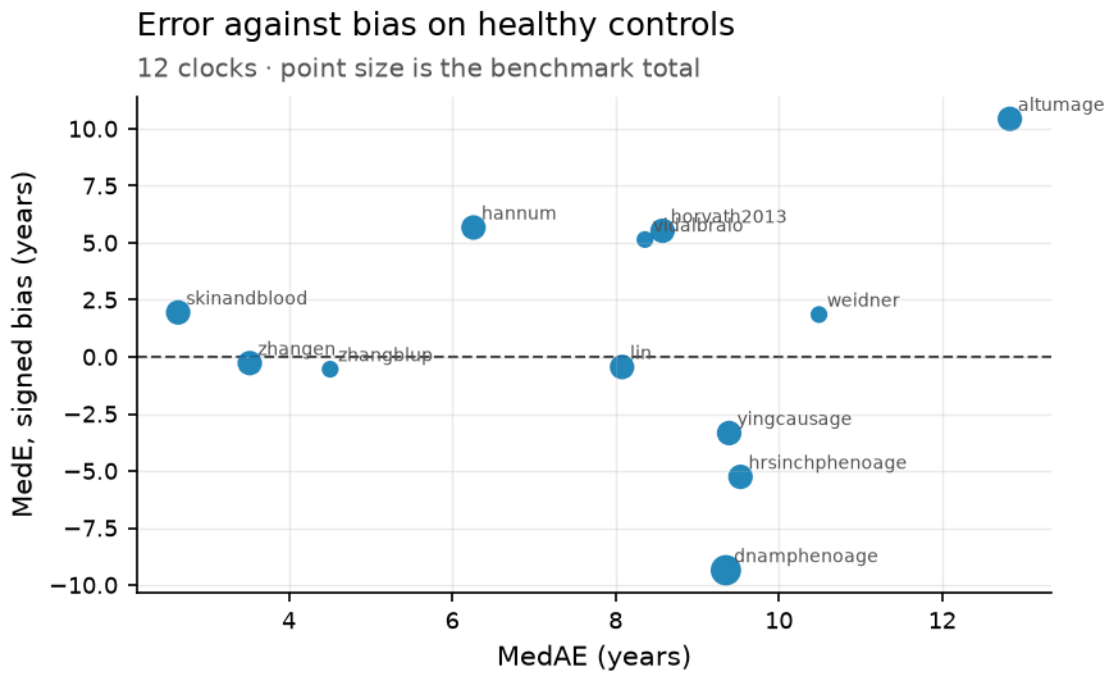


Figure 27: Error against bias on healthy controls

Neither axis is a ranking. A clock at the origin merely predicts chronological age well, which is the one thing a useful clock need not do.

`plot.benchmark_error_bias()` in Python.

Effect size per clock and dataset

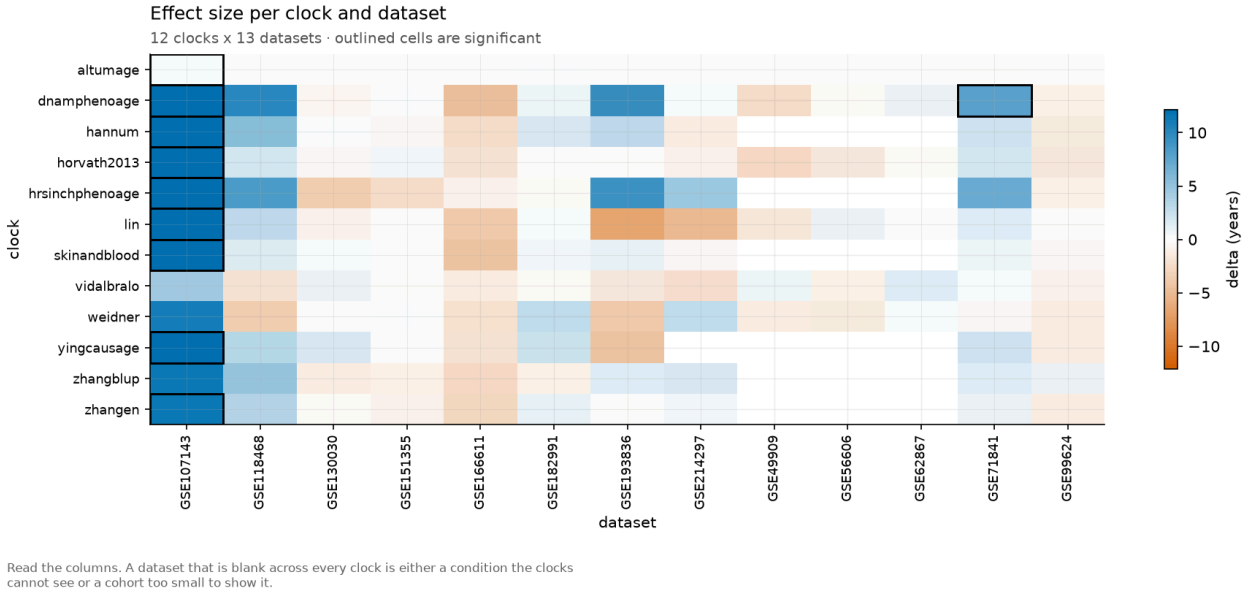


Figure 28: Effect size per clock and dataset

Read the columns. A dataset that is blank across every clock is either a condition the clocks cannot see or a cohort too small to show it.

`plot.benchmark_heatmap()` in Python.

Data quality

Drawn before the results, because a coverage problem invalidates everything after it.

Feature coverage per clock

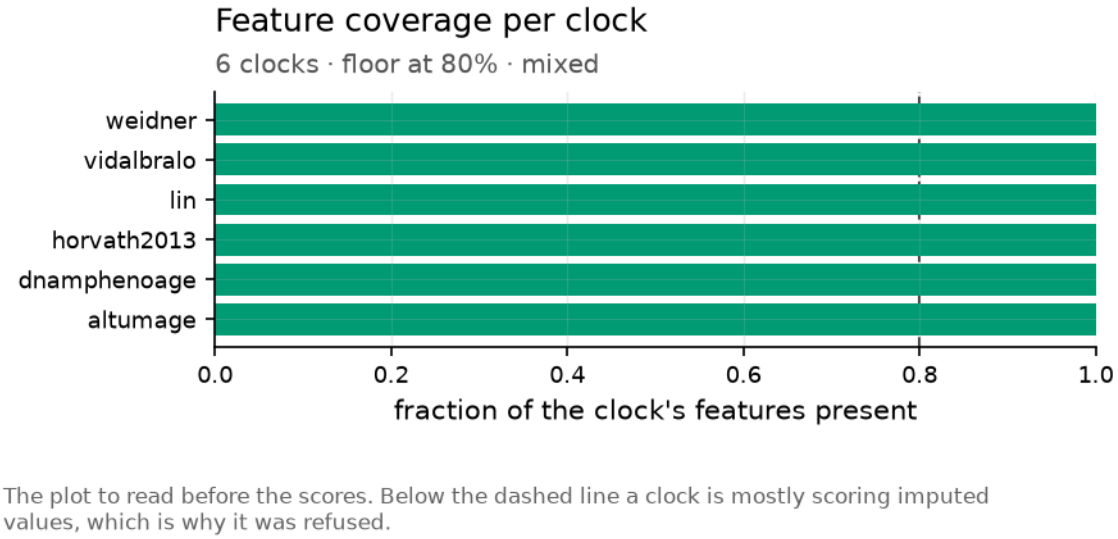
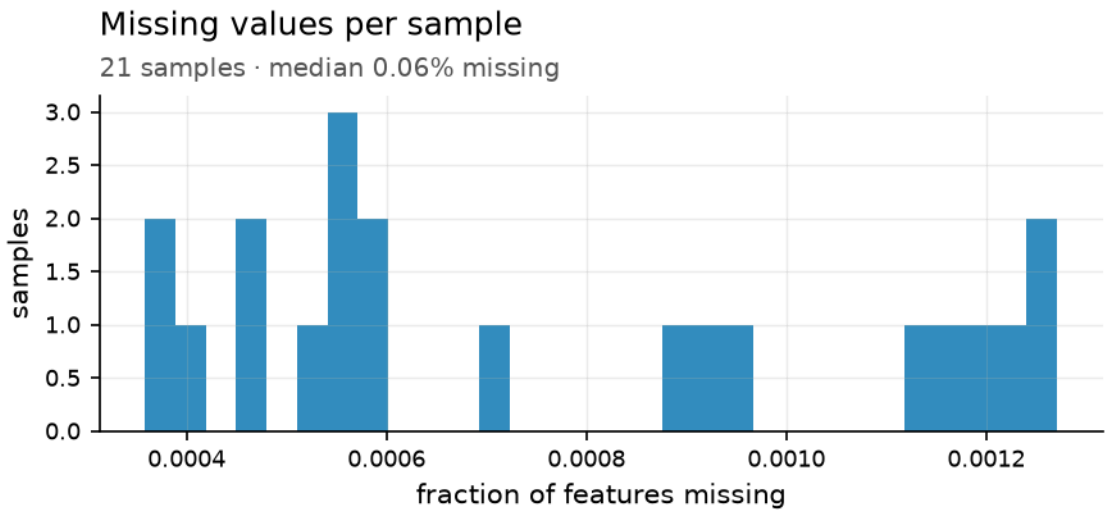


Figure 29: Feature coverage per clock

The plot to read before the scores. Below the dashed line a clock is mostly scoring imputed values, which is why it was refused.

`plot.coverage_bar()` in Python.

Missing values per sample



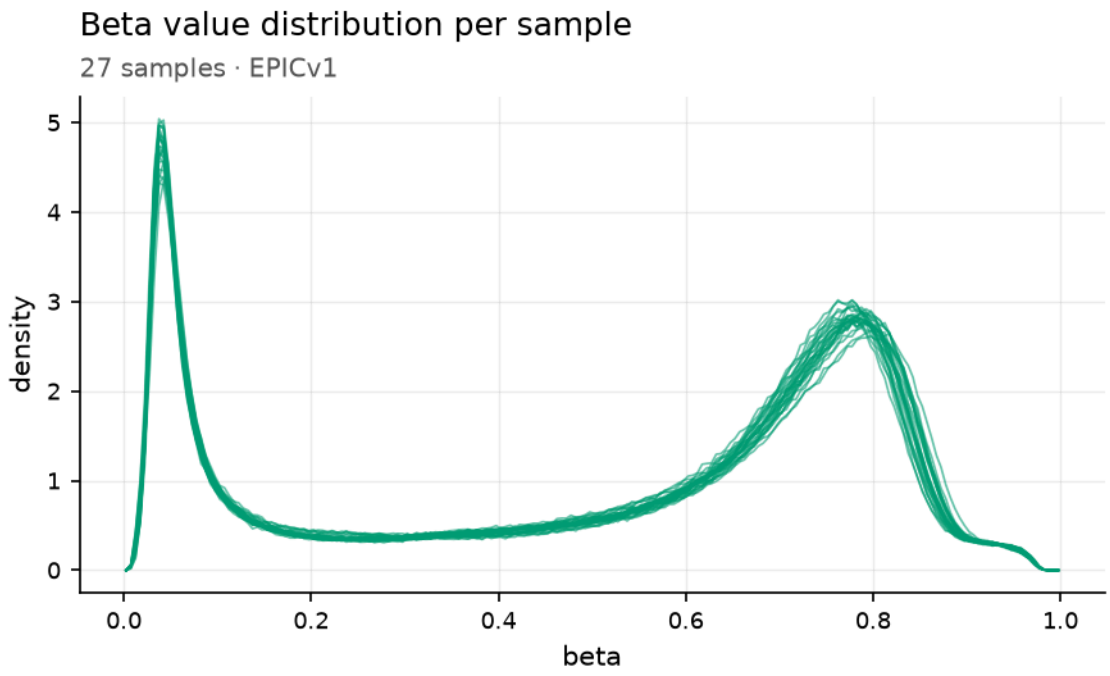
One sample far to the right is a failed array; a shifted whole cohort is a probe filter someone applied upstream.

Figure 30: Missing values per sample

One sample far to the right is a failed array; a shifted whole cohort is a probe filter someone applied upstream.

`plot.missingness()` in Python.

Beta value distribution per sample



Two peaks near 0 and 1 is what a working array looks like. A sample with a filled middle failed, and no downstream normalisation will fix it.

Figure 31: Beta value distribution per sample

Two peaks near 0 and 1 is what a working array looks like. A sample with a filled middle failed, and no downstream normalisation will fix it.

`plot.beta_density()` in Python.

Platform

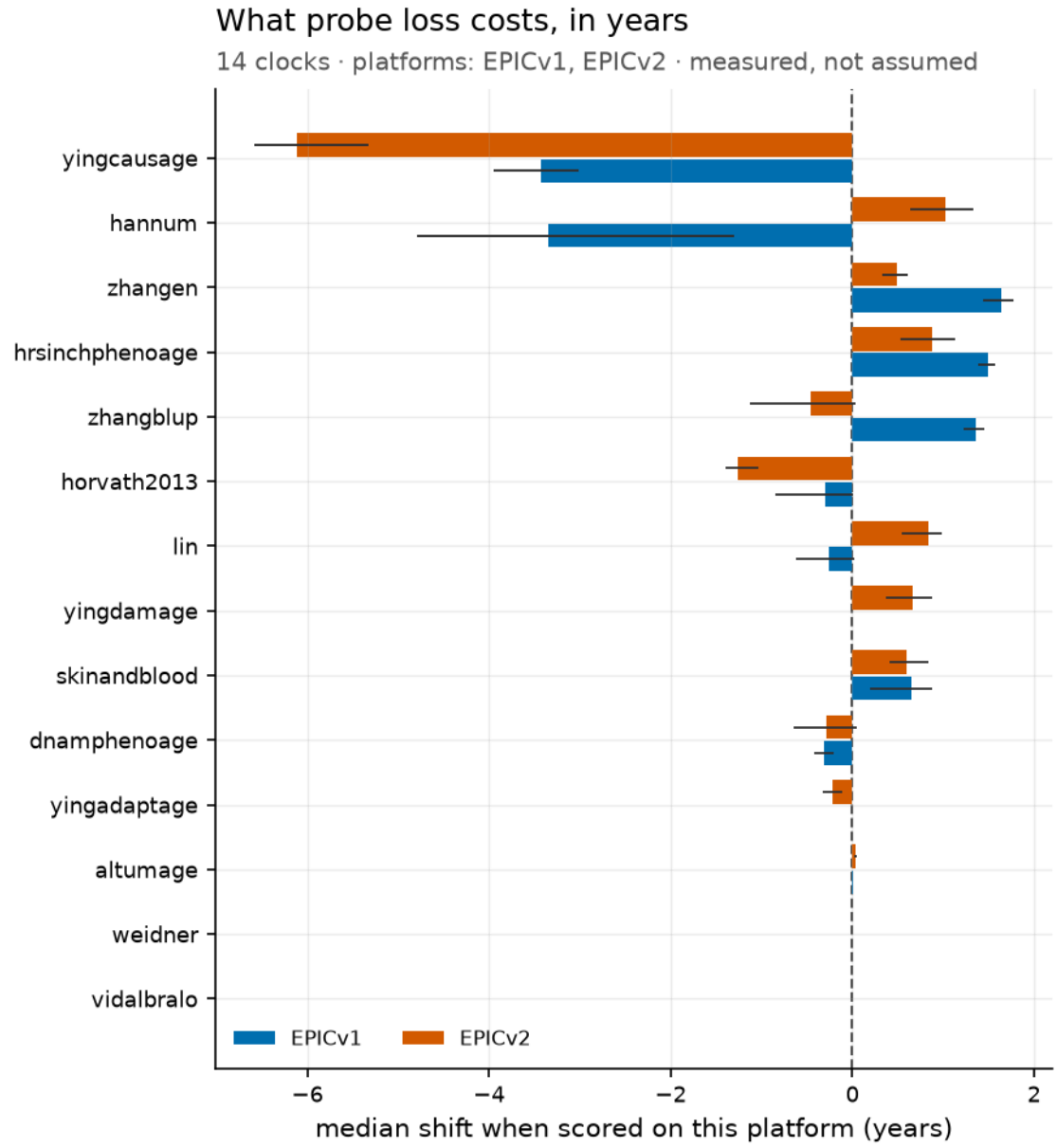


Figure 32: Platform

`plot.platform()` in Python.

Study

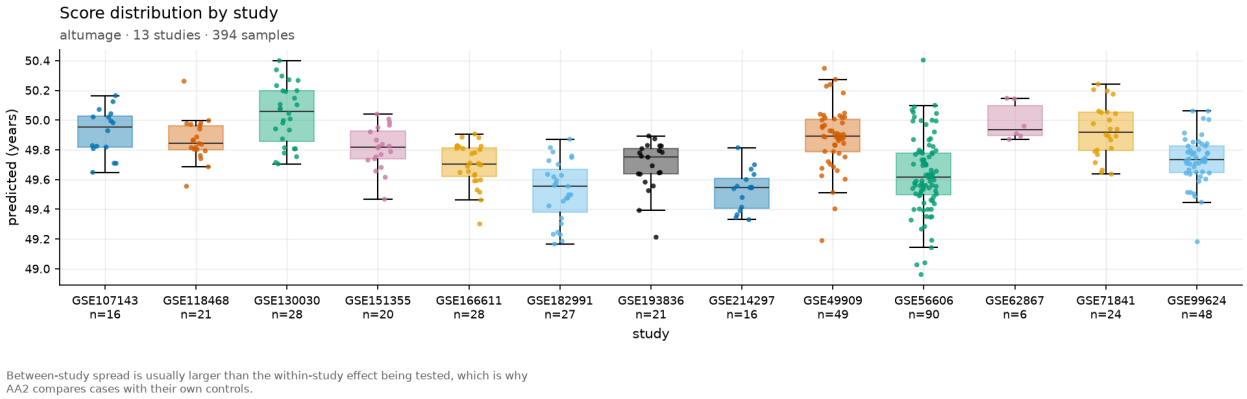


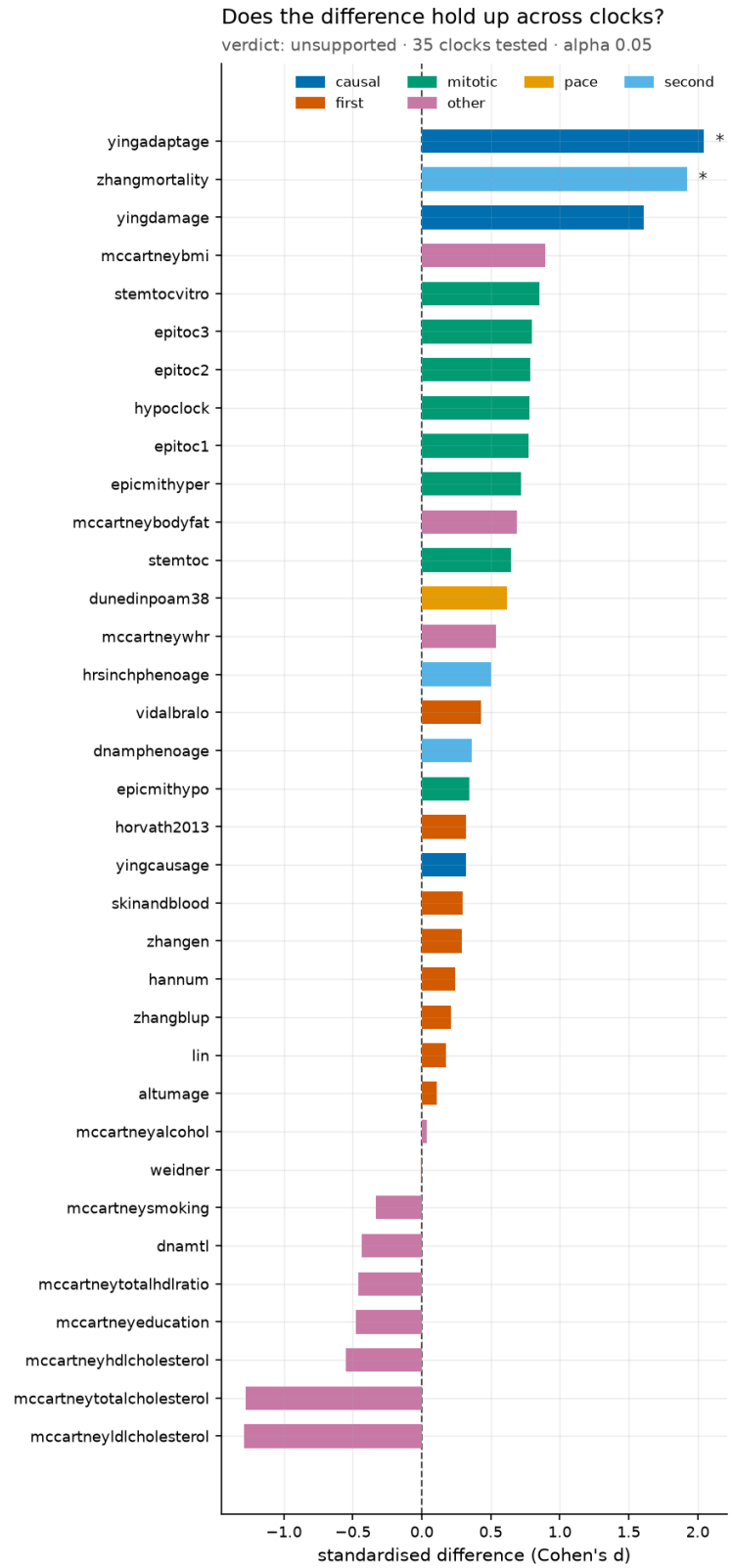
Figure 33: Study

`plot.study()` in Python.

Other

Figures not yet placed in a section above.

Does the difference hold up across clocks?



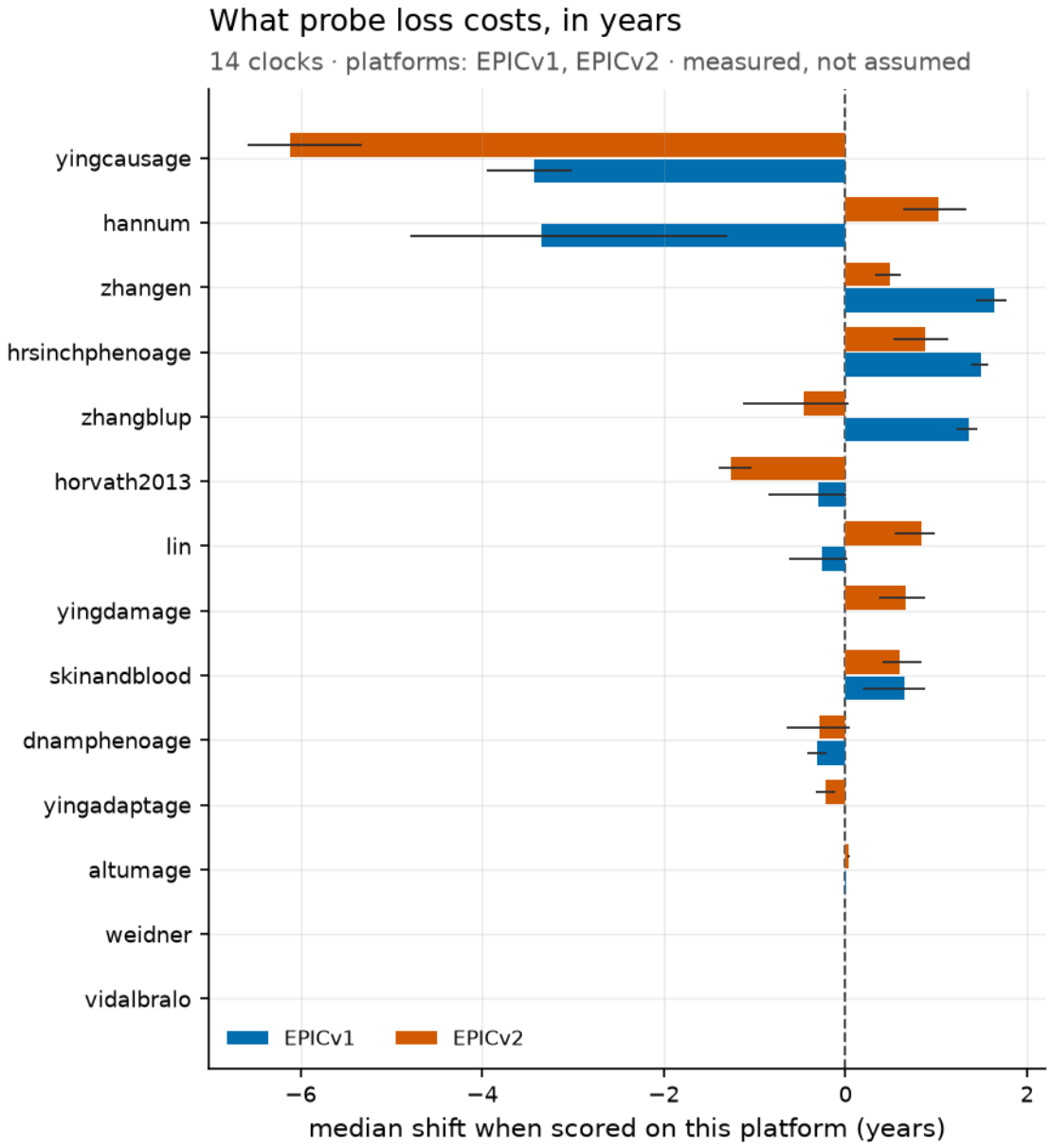
One bar per clock, coloured by generation; * survives Benjamini-Hochberg and ** survives Bonferroni across the tests actually run. The shape is the finding, not any one bar. A real change lights up across generations; a single lit bar among twenty dark ones is the published signature of a false positive, and five of six re-analysed intervention datasets looked exactly like that.

Figure 34: Does the difference hold up across clocks?

One bar per clock, coloured by generation; * survives Benjamini-Hochberg and ** survives Bonferroni across the tests actually run. The shape is the finding, not any one bar. A real change lights up across generations; a single lit bar among twenty dark ones is the published signature of a false positive, and five of six re-analysed intervention datasets looked exactly like that.

`plot.consensus_plot()` in Python.

What probe loss costs, in years



Full 450K matrices masked down to each platform's probe set and rescored, absent probes filled from each clock's reference values as score() fills them; the bar is the median shift and the whisker a bootstrap interval. A bar at zero is a clock that lost nothing. A long bar is a clock whose lost probes carry weight no fill value stands in for, which is why a correlation check passes while the intercept moves. Reported, never subtracted.

Figure 35: What probe loss costs, in years

Full 450K matrices masked down to each platform's probe set and rescored, absent probes filled from each clock's reference values as score() fills them; the bar is the median shift and the whisker a bootstrap interval. A bar at zero is a clock that lost nothing. A long bar is a clock whose lost probes carry weight no fill value stands in for, which is why a correlation check passes while the intercept moves. Reported, never subtracted.

`plot.platform_bias()` in Python.

How much of each score is the assay

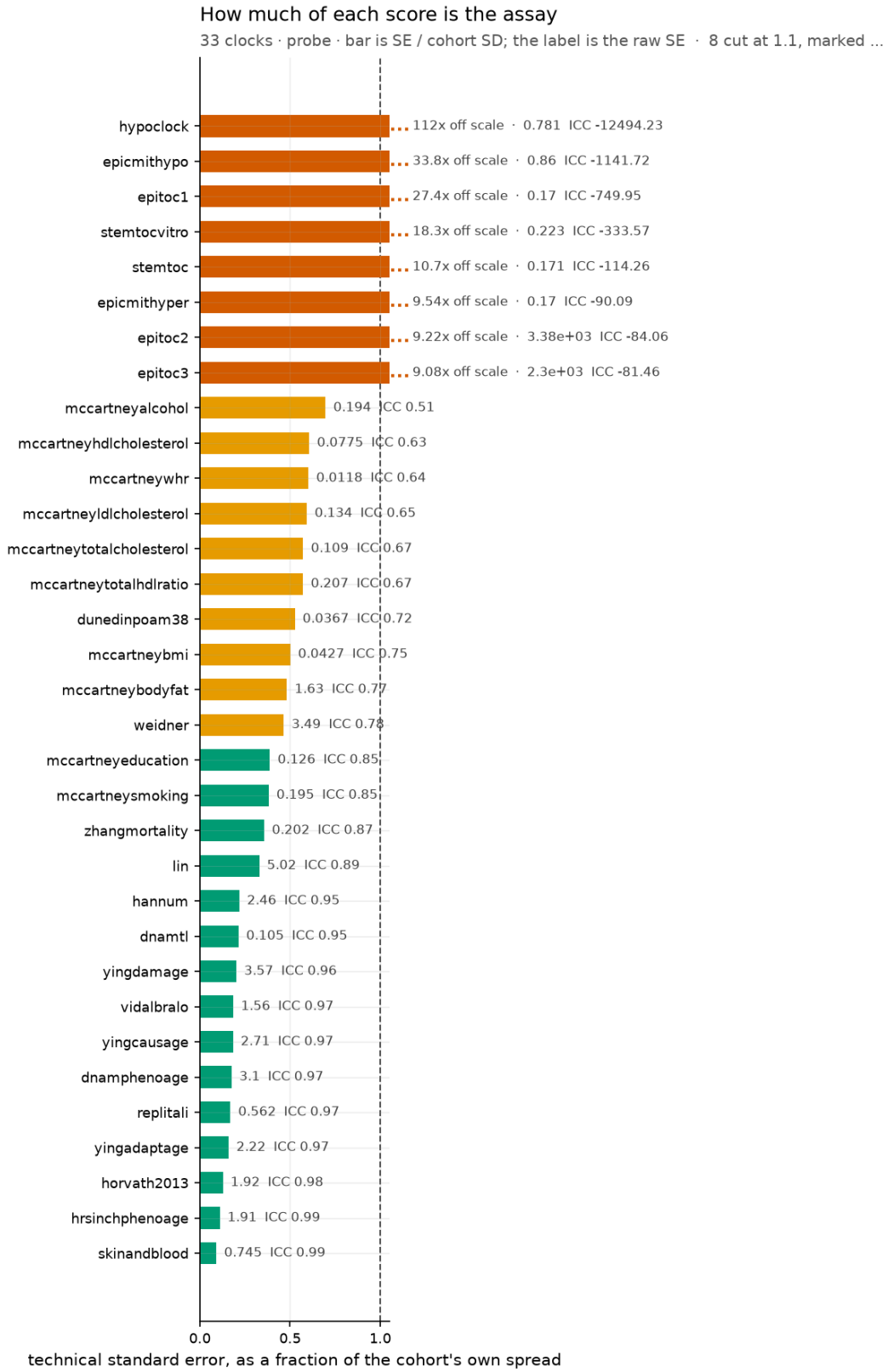
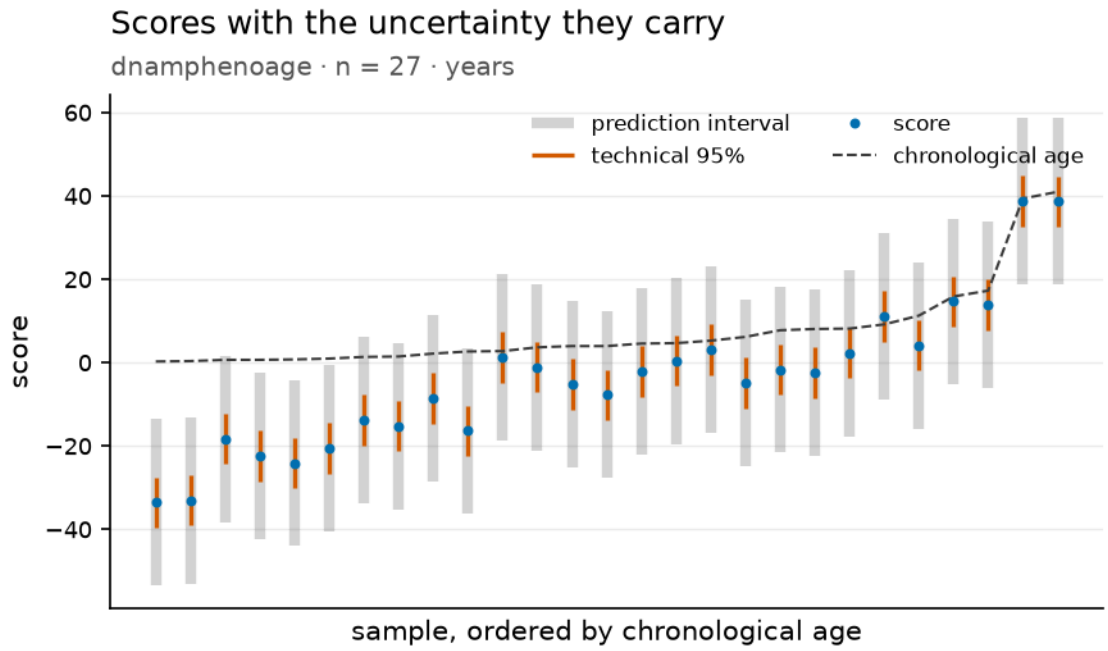


Figure 36: How much of each score is the assay

Measurement error propagated from published per-probe reliability through each clock's own weights. The ICC beside each bar is what this cohort implies, not a published figure: green above 0.8, amber below it, red where the assay noise is as large as the spread between these samples – a clock that cannot tell them apart, however confident its point estimate.

`plot.reliability_forest()` in Python.

Scores with the uncertainty they carry



Two samples whose bars overlap are not distinguishable by this clock on this assay. The narrow bar is measurement error -- what a repeat of the same DNA would do. The wide grey bar, where present, is the conformal prediction interval against chronological age, which is a larger question and has a larger answer. Neither is a confidence interval on a diagnosis.

Figure 37: Scores with the uncertainty they carry

Two samples whose bars overlap are not distinguishable by this clock on this assay. The narrow bar is measurement error – what a repeat of the same DNA would do. The wide grey bar, where present, is the conformal prediction interval against chronological age, which is a larger question and has a larger answer. Neither is a confidence interval on a diagnosis.

`plot.score_interval()` in Python.

In total

26 figures. Every one is also written per dataset under `test/output_figures/<group>/<dataset>/` when the full corpus run is executed; the gallery keeps one representative of each type so a reader does not have to fetch 586 MB to see them.

Chapter 18. The science of aging clocks

Every number FALCONAge returns comes from somebody's paper. This is the record of which paper, what it actually says, and where the published description and the circulating implementation disagree, which they do in eleven documented cases.

It covers what an aging score *is*, the chemistry and biology behind each input layer, every algorithm family with its equations, the exact transformations and constants published clocks use, the output types and their units, the figures the field reads, and the benchmarking machinery used to judge a clock. It runs to the state of the field in 2026.

Read it as the substrate rather than as the manual. Architecture says which file computes what; this says why that is the right computation, and it is the document to check before trusting a number. Where a section describes something FALCONAge does not implement, it says so, the catalogue here is deliberately wider than v1.0, because the point of writing the field down is to know what is missing.

1. Definitions and framing

1.1 The three quantities

- **Chronological age (CA)** - time since birth. Years for humans, months or days for rodents, hours for *C. elegans*.
- **Biological age (BA)** - the output of a model that maps molecular or clinical features to a number on an age-like scale. In practice BA is whatever a fitted model predicts; its meaning comes entirely from what the model was trained on.
- **Age acceleration (AA)** - the residual. Two conventions, and they are not interchangeable:
  - **Absolute age acceleration (AAA)**, also called delta: `Δ = BA − CA`. This is what ComputAgeBench tests (ComputAge benchmarking.py:355).
  - **Relative age acceleration (RAA)**: residual of `lm(BA ~ CA)` in the analysis sample. This removes the systematic regression-to-the-mean bias that all penalised regressions carry. Used by the TwinGene/UKB pipeline as `BA_res = residuals(lm(BA ~ age))` (Biological-Aging-and-PRS README).
  - A third convention, used by spatial single-cell clocks, replaces CA with the per-animal per-cell-type mean prediction, so acceleration is measured within an animal rather than against the calendar (SpatialAgingClock deploy.py `get_age_acceleration`).
- **Pace of aging** - a rate, not a level. Units are biological years per chronological year. DunedinPACE and DunedinPoAm are the two published examples; both are centred near 1.0.

1.2 Clock generations

This taxonomy governs everything downstream, what the score means, whether it responds to interventions, and how it must be validated.

Generation	Training target	Examples	Behaviour
First	Chronological age	Horvath 2013, Hannum, Lin, Zhang EN/BLUP, AltumAge, PedBE, SkinAndBlood	High age correlation, weak or absent mortality signal. Zhang BLUP has RMSE ≈ 2.0 y yet its acceleration no longer predicts mortality, the "oracle paradox".
Second	Mortality or a composite phenotype	PhenoAge, DNAmPhenoAge, GrimAge, GrimAge2, CpGPTGrimAge3, SystemsAge, OMICmAge	Lower CA accuracy, much stronger mortality/morbidity association.
Third ("pace")	Longitudinal rate of change across many organ systems	DunedinPoAm38, DunedinPACE	Rate, responds fastest to interventions.
Causal	CA, but with features filtered by Mendelian-randomisation evidence of causality	YingCausAge, YingDamAge, YingAdaptAge	Separates damaging from adaptive/protective methylation.
Mitotic	Stem-cell division count / replicative history	epiTOC1, epiTOC2, epiTOC3, stemTOC, HypoClock, epiCMIT-hyper/hypo, RepliTali, MiAge	Cancer risk, proliferative history.
System- and organ-specific	Mortality or CA, restricted to organ-enriched features	SystemsAge (11 systems), organAging (24 organ protein panels), MEAT (muscle), CorticalClock (brain), scImmuAging (5 PBMC cell types)	Reveals within-person heterogeneity.
Foundation-model	Self-supervised pretraining, then fine-tuned	CpGPT, methylGPT	Cross-platform imputation, SOTA mortality prediction.

1.3 Biomarker validation taxonomy (Biomarkers of Aging Consortium)

Moqri et al. classify aging biomarkers as **predictive** (associates with future outcomes), **responsive** (moves under an intervention), and **biological** (has a mechanistic, causal claim). A clock that is only predictive is a risk score; only responsive is a surrogate endpoint; a biological biomarker needs causal evidence. The consortium's Biolearn library and the 2024–25 open competition (chronological age, mortality, multimorbidity; >$200k prizes) operationalise this. Any new tool should record which of the three claims each clock supports.

1.4 Why this matters commercially and scientifically

The geroscience hypothesis holds that aging is the dominant risk factor common to cancer, cardiovascular disease, T2D, Alzheimer's and Parkinson's, so slowing aging delays all of them. A validated surrogate endpoint shortens trials that would otherwise need decades of mortality follow-up. TAME (metformin) has raised ~$11M of $40M over seven years with a six-year observation window; a responsive clock is the alternative (computational_aging_course/ml/aging_clocks.md).

Butler's criteria for a usable aging biomarker: changes with age; predicts mortality better than CA alone; predicts early-stage age-related disease; minimally invasive. Moskalev adds: sensitive early, predictable in a practical time window, robust.

References

The bibliography moved to its own page, where it is numbered once and cited by number from every page of this site. It reads the same in the PDF, where it is bound as the last chapter.

Citations in this document are the bracketed numbers, for example [5] for Horvath 2013. The numbering is global, so a source keeps its number wherever it appears.

Chapter 19. Biology and measurement

2. Biology of aging that the scores encode

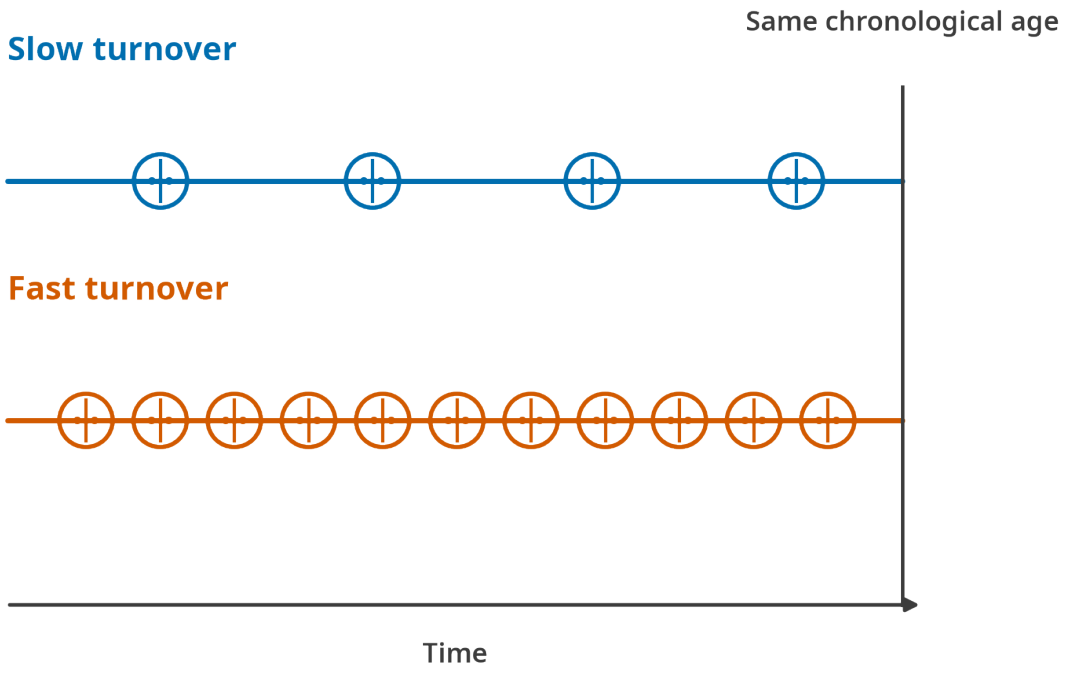


Figure 38: Two tissues of the same chronological age that have divided a different number of times

Elapsed time and accumulated cell divisions are separate axes, and a mitotic clock reads the second one. Colon epithelium turns over in days and brain tissue barely at all, so two samples taken from the same donor on the same day carry the same chronological age and very different division counts. This is why a division count is not reported in years and why subtracting a chronological age from one is refused: the quantity was never elapsed time.

The twelve hallmarks of López-Otín et al. 2023 [56], which extend the nine of 2013 [55], are the biological vocabulary the scores are trying to summarise. Grouped as primary (damage), antagonistic (compensatory responses that turn harmful), integrative (system-level consequence). A hallmark must show age-associated manifestation, experimental acceleration of aging when worsened, and therapeutic reversibility.

Primary

1. **Genomic instability** - point mutations, indels, single/double-strand breaks (local); translocations, chromosomal rearrangement, nuclear-architecture defects, transposon and viral integration (global). Species somatic mutation rate is inversely correlated with lifespan (Cagan 2022). Pathogenic mechanisms: clonal expansion, somatic evolution, disruption of gene regulatory networks producing transcriptional noise. Mitochondrial DNA is separately unstable

- many copies, high replicative index, oxidative microenvironment, no histones; most aged-cell mtDNA mutations come from POLG replication error rather than oxidative damage. DNA damage feeds inflammation via cytoplasmic chromatin fragments (CCF), escaped mtDNA, and transposon cDNA. Progeroid syndromes from repair defects: Werner, Bloom, xeroderma pigmentosum, trichothiodystrophy, Cockayne, Seckel.

2. **Telomere attrition** - TTAGGG tandem repeats, several to tens of kb, with a 75–300 nt G-rich 3′ overhang forming the protective cap. Telomere dysfunction signals to mitochondria through p53 → PGC-1α suppression. Measured molecularly (qPCR T/S ratio, Southern TRF, flow-FISH) or estimated from methylation (DNAmTL, PCDNAmTL).
3. **Epigenetic alterations** - histone PTMs, DNA methylation, ncRNA, chromatin remodelling. The direction of travel in aging is de-repression: global hypomethylation, heterochromatin relaxation, transposon (LINE-1/SINE) reactivation, active histone acetylation. Age-related methylation change concentrates in introns, intergenic regions, tumour-suppressor and Polycomb-target promoters. Cross-talk with known longevity pathways: histone demethylases act on IGF-1 signalling; sirtuins are NAD^+-dependent deacetylases.
4. **Loss of proteostasis** - mistranslation, misfolding, incomplete products; slowed elongation; cumulative oxidative protein damage saturating chaperones. Drives ALS, Alzheimer's.
5. **Disabled macroautophagy** - autophagosome capture then lysosomal digestion. Selective subtypes: glycophagy, lipophagy, mitophagy, ER-phagy, nucleophagy, xenophagy, lysophagy. Autophagy-gene expression falls with age; genetic inhibition accelerates aging in models; CD4+ T cells from exceptionally long-lived people show enhanced autophagic flux. Reduced flux permits aggregate accumulation, dysfunctional organelles, uncleared pathogens, and inflammasome activation. NAD^+ precursors have clinical-trial evidence in non-melanoma skin cancer chemoprevention, prediabetic insulin resistance, and Parkinson's neuroinflammation.

Antagonistic

6. **Cellular senescence** - stable proliferative arrest via TP53→CDKN1A/p21 and CDKN2A/p16→RB1. SASP effects: chemokine/cytokine recruitment (CCL2, CXCL2, CXCL3, IL-18, IL-2, IL-6, IL-8); immune suppression (TGF-β); fibroblast activation and collagen deposition (TGF-β, IL-11, PAI-1); ECM remodelling (MMPs); progenitor activation (EGF, PDGF); paracrine senescence (TGF-β, TNF-α, IL-8). Lamin B1 depletion → SAHF (H3K9me3, HP1α foci) → LINE-1 de-repression → cytosolic dsDNA → cGAS/STING and TLR activation.
7. **Deregulated nutrient sensing** - insulin/IGF-1 → PI3K-AKT and Ras-MEK-ERK; FOXO and E26 transcription factors; MTORC1 responding to glucose, amino acids (leucine), hypoxia, low energy, acting on SREBP and TFEB. Overnutrition activates MTORC1 and EP300 and inhibits AMPK, SIRT1, SIRT3; fasting and dietary restriction do the reverse and extend lifespan across models including primates.
8. **Mitochondrial dysfunction** - falling ATP output, rising ROS. POLG-mutator mice show fragmented cristae, disrupted outer membrane, and premature aging (alopecia, kyphosis, low body weight and subcutaneous fat, low BMD, anaemia, cardiomyopathy) mirroring telomerase-deficient, p53-hyperactive, and PGC1α/β-null mice.

Integrative

9. **Stem cell exhaustion** - reduced steady-state renewal and injury repair. OSKM (OCT4, SOX2, KLF4, MYC) full reprogramming resets p16, telomere length, and the DNA methylation clock; partial/transient reprogramming rejuvenates clock, DNA damage, epigenetic patterns and transcriptome without identity loss.

10. **Altered intercellular communication** - pro-aging circulating factors: CCL11/eotaxin and β2-microglobulin (reduce neurogenesis), IL-6 and TGF-β (impair haematopoiesis), complement C1q (compromises muscle repair). A single old-blood transfusion ages young mice within days; dilution of old mouse blood with saline + 5% albumin rejuvenates multiple tissues.
11. **Chronic inflammation (inflammaging)** - rising CRP, IL-6; plasma IL-6 predicts all-cause mortality. Fed by CHIP, cytosolic DNA, SASP, uncleared debris, circadian disruption, gut barrier failure.
12. **Dysbiosis** - gut microbiota diversity declines; SCFA, vitamin, secondary bile acid and amino-acid-derivative production shifts; gut–brain and gut–organ signalling degrades.

PIP framing - each hallmark is Part of a complex process, Interdependent with the others, and a Point-of-entry for intervention. No single hallmark accounts for aging.

What the clocks actually pick up

- Most age-related DNAm changes are small: ~1.5–5% absolute beta shift over decades. Clock signal is an aggregate of many tiny shifts, not a few large ones.
- ~20% of 450K loci can support *some* chronological clock (Porter 2021), so CpG identity across published clocks overlaps very little. Liu 2020 argues the clocks capture genuinely different aging axes rather than being noisy versions of one thing.
- CTCF-binding sites are enriched among top-ranked CpGs in AltumAge SHAP/DeepPINK analyses.
- Epigenetic drift: interindividual methylation variance rises with age (Hannum 2013).
- Men age ~4% faster than women on the Hannum clock.
- Epigenetic age is ~40% heritable (Horvath & Raj 2018).
- Clocks work in immortalised non-senescent B cells and embryonic cells, so they are not simply senescence detectors; they work in non-proliferating brain cells, so they are not simply division counters. Multi-tissue clocks trained on one tissue transfer to others; single-tissue and second-generation clocks may partly reflect cell composition.

3. Measurement chemistry, per omics layer

3.1 DNA methylation

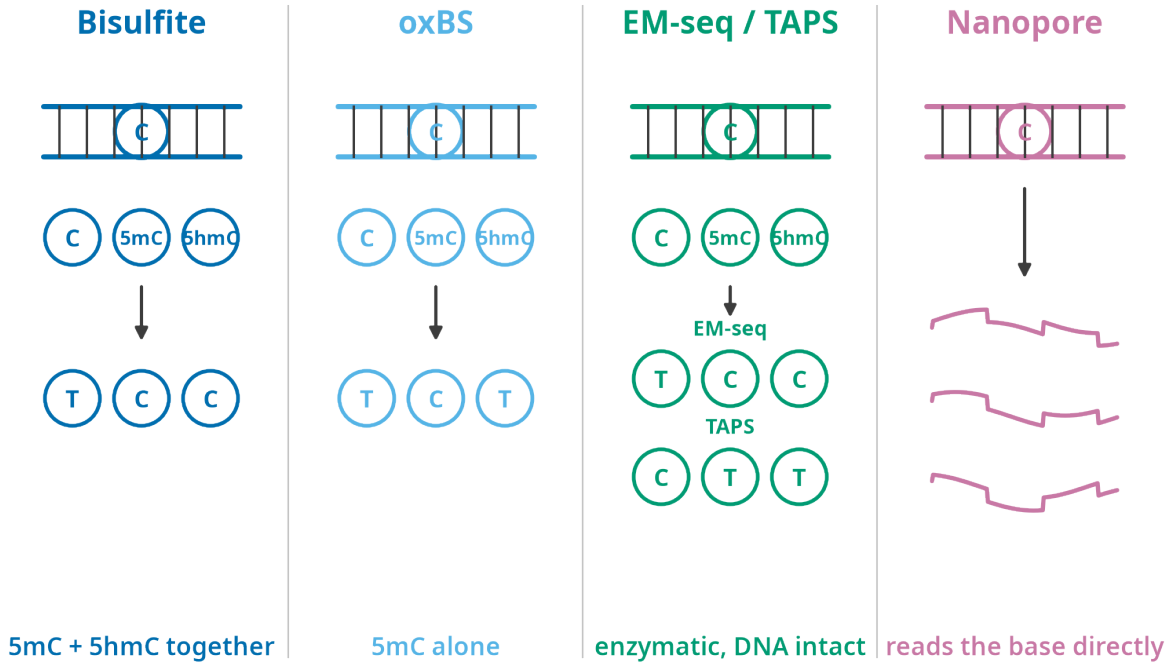


Figure 39: What each conversion chemistry does to an unmodified C, a 5mC and a 5hmC

The four columns answer the same question differently, and the differences are what a clock's coefficients were fitted through. Bisulfite protects 5mC and 5hmC alike, so a bisulfite beta is their sum and every array clock is trained on that sum. Oxidative bisulfite converts 5hmC to a form that then reads as T, which isolates 5mC. EM-seq and TAPS are both enzymatic and both leave the DNA intact, but they are inverses of each other at the readout: EM-seq reads a methylated cytosine as C, TAPS reads it as T. Nanopore skips conversion and calls the base from the current trace. A coefficient fitted on a bisulfite array does not transfer to a TAPS library without accounting for that.

Chemistry. 5-methylcytosine at CpG dinucleotides. In mammals 70–80% of CpG cytosines are methylated. Bisulfite treatment deaminates unmethylated cytosine to uracil (read as thymine) and leaves 5mC intact, creating a C/T polymorphism that every downstream assay queries. Enzymatic conversion (EM-seq, TAPS) is the low-damage alternative and is displacing bisulfite for sequencing.

Beta value. For a CpG, `β = M / (M + U + α)` where M and U are methylated and unmethylated intensities and α is a stabilising offset (Illumina default 100). β ∈ [0,1]. Nearly every clock consumes β. The M-value `M = log2(β / (1 − β))` is preferred for differential testing because it is homoscedastic, but clock coefficients are almost always in β space, do not silently swap.

Array platforms.

Platform	Probes	Notes
Illumina 27K (HumanMethylation27)	27,578	Horvath 2013 and Bocklandt-era clocks
Illumina 450K	485,512	The workhorse; most published coefficient sets
EPIC v1 (850K)	865,918 (manifest B4)	Keeps over 90% of 450K probes; adds about 413,000 CpGs, many in enhancers
EPIC v2	937,690 probes / 923,452 unique sites	Probe IDs carry suffixes (`cg00000029_TC21`); replicate probes must be aggregated before applying any clock
Illumina MSA (Methylation Screening Array)	~270K	Targeted, cheaper; emerging
HorvathMammalMethylChip40	~37K conserved loci	128+ mammalian species, 59 tissues; basis of the universal mammalian clocks
HorvathMammalMethylChip320	expanded	Used by EnsembleAge

Probe chemistry inside the array. Infinium Type I uses two beads per CpG (one for the methylated allele, one for unmethylated) in a single colour channel; Type II uses one bead with two-colour single-base extension. Type II is ~85% of probes on EPIC and has a compressed dynamic range and different beta distribution. This design bias is what BMIQ corrects.

Sequencing assays. - **RRBS** - MspI digestion enriches CpG-dense regions; sparse, integer-count coverage. Petkovich (90 CpGs, mouse blood), Stubbs (multi-tissue mouse), Meer, Thompson all use RRBS. - **WGBS** - full coverage, expensive; the substrate for scAge/scEpiAge single-cell work. - **Targeted bisulfite sequencing** - HepatoXu (10 loci, plasma cfDNA HCC detection). - **scM&T-seq / snmC-seq** - single-cell methylation, binary and extremely sparse (often <10% of CpGs covered per cell), forcing entirely different algorithms.

Normalisation pipeline (array). IDAT → background correction and dye bias (`noob`) → between-array (`funnorm`, quantile, or `SeSAMe`) → probe-type bias (`BMIQ` or `SWAN`) → detection-p filtering, cross-reactive/SNP probe masking, sex-chromosome handling. Consensus from Illumina EPIC comparisons: `noob + BMIQ` and `SeSAMe` perform well; funnorm is preferred when global methylation differences are expected between groups. `sesame_450k_median.csv` in ComputAge is the field's standard NA-fill vector (ComputAge clocks dir).

Cell composition. Whole blood is a mixture; a clock trained on bulk blood partly reads cell proportion. Two standard reference libraries: Reinius/Houseman 6-cell (450K, `450K_reinius_12_reference.csv`) and Salas/IDOL 12-cell EPIC (`EPIC_salas_18_reference.csv`). Deconvolution is a constrained regression onto a reference signature matrix; pyaging exposes 6-cell and 12-cell EPIC deconvolution as clocks in their own right, returning proportions.

Sex estimation from methylation. X and Y chromosome probe z-scores against the autosomal mean and SD, projected on PCA loadings, implemented as `XChrom` (4,047 probes) and `YChrom` (284 probes) in pyaging e7267e9, classes `XChrom` and `YChrom` in `_models.py`. Needed because GrimAge, DNAmFitAge and the mammalian clocks take sex as an input.

3.2 Transcriptomics

- **Bulk RNA-seq.** Raw counts → library-size normalisation. Three conventions appear in published clocks:
  - **TPM then log1p** - divide by transcript length, rescale to 1e6, `log1p`. Used by OcampoATAC1/2 in pyaging.
  - **CPM then binarise** - BiT age (*C. elegans*).
  - **RLE (edgeR `calcNormFactors(method="RLE")`) scaled to 1e7, then log10(x+1)** - tAge.
- **Microarray expression** - treat as already normalised; rank or quantile align to the RNA-seq reference distribution.
- **Gene identifier space.** tAge maps everything into **mouse Entrez** space via ortholog tables for human, rat, macaque (tAge inst/extdata/metadata/). Pasta/REG use human Ensembl and derive `PastaMouse` by selecting `ENSMUSG*` features from an 8,113-gene joint space and filling human-only genes with reference values.
- **Single cell.** Sparsity and dropout break bulk assumptions. Three strategies in the literature:
  - **Pseudobulk** - average n cells per (donor, cell type), repeated B times. scImmuAging uses size 15, n = 100 repeats (scImmuAging/R/main.R); SpatialAgingClock uses n = 20, B = 100.
  - **Binarisation** - CellBiAge: after log-normalisation, CCA batch integration and scaling, set entry = 1 if scaled value > 0 else 0, on the top 2,000 HVGs. This is the single largest performance lever in that paper (see §4.11).
  - **Spatial smoothing** - propagate expression over a spatial neighbour graph before prediction (see §4.13).

3.3 Chromatin

- **Histone-mark ChIP-seq.** Signal is read from bigWig files, averaged over each Ensembl gene body (chr 1–22, X), then `arcsinh`-transformed (pyaging `bigwig_to_df`). Eight marks have published clocks: H3K4me1, H3K4me3, H3K9ac, H3K9me3, H3K27ac, H3K27me3, H3K36me3, plus a pan-histone model. All are PCA + elastic net + ARD.
- **ATAC-seq.** Counts per open chromatin region (OCR) → length-normalised TPM → `log1p` → subset to the model's OCRs. OcampoATAC1 (228 OCRs) and OcampoATAC2 (380 OCRs), PBMC.

3.4 Proteomics

- **Olink Explore** (proximity extension assay). Output is **NPX**, a log2-scaled relative quantification. Explore 3072 vs Explore 1536 have different panels; organAging ships separate full and feature-reduced models for each. Application is a plain dot product of NPX with coefficients, plus intercept for first-generation models (organAging README).
- **SomaScan** (aptamer). Used by Oh et al. 2023 organ clocks (~5,000 proteins, ~5,700 people).
- **Mass spectrometry.** For any non-Olink platform, organAging's recommendation is: transform to a roughly symmetric distribution (log intensities), z-score each protein, then multiply by the Olink NPX standard deviations published in their Table S3 before applying coefficients.
- **Organ-enrichment definition.** A protein is assigned to an organ if its GTEx expression in that tissue is ≥ 4× the mean across all other tissues. organAging's feature sets from organAging `GTEx_4x_FC_genes.json`: Adipose 5, Adrenal 4, Artery 14, Brain 118, Esophagus 5, Female 6, Heart 7, Immune 124, Intestine 31, Kidney 9, Liver 79, Lung 11, Male 55, Muscle 20, Pancreas 29, Pituitary 12, Salivary 10, Skin 17, Stomach 12, Thyroid 2, Organismal 2,347, Multi-organ 569, Conventional 2,916.

3.5 Clinical chemistry and haematology

The PhenoAge / KDM / HD family. Canonical nine PhenoAge markers plus age:

Marker	Unit (Levine)	Alternative unit
Albumin	g/L	g/dL
Creatinine	μmol/L	mg/dL
Glucose	mmol/L	mg/dL
C-reactive protein	ln(mg/dL)	ln(mg/L)
Lymphocyte percent	%	
Mean cell volume	fL	
Red cell distribution width	%	
Alkaline phosphatase	U/L	
White blood cell count	10^3 cells/μL	

Unit mismatch is the number-one source of wrong PhenoAge values in the wild, see §16.

The BioAge R package extends this to a larger marker menu with clinically-healthy windows used to build the HD reference cohort (BioAge hd_nhanes.R): albumin, ALP, ln(CRP) with CRP < 2, total cholesterol < 200, ln(creatinine), HbA1c 4–5.6, SBP < 120, DBP < 80, mean BP < 93.33, BUN, uric acid (sex-specific 2.7–6.3 F / 3.7–8.0 M), lymphocyte % 20–40, MCV, WBC 4.5–11, RBC (sex-specific), RDW 11.5–14.5, monocytes 3–10, neutrophils 45–74, cystatin C 0.51–0.98, GGT (sex-specific), insulin 2.52–24.1, HDL (sex-specific), LDL 80–130, triglycerides 54–110, pulse 60–100, FEV1 ≥ 0.8 × mean, cadmium 2.7–10.7, vitamins A/E/B12/C.

3.6 Other layers

- **Glycomics.** IgG N-glycan composition. GlycanAge from 5,117 European individuals; three glycans (GP6 non-galactosylated, GP14 and GP15 bis-galactosylated) explain ~60% of CA variance; prediction error ~9.7 years. Notable as the first clock class with repeated demonstrations of reversal by lifestyle intervention.
- **Metabolomics.** MetaboAge and the metabolite arm of GOLD BioAge / StackAge / OMICmAge. Nightingale NMR and LC-MS panels dominate; features are usually log-transformed and z-scored.
- **Microbiome.** Deep-learning microbiome clocks exist (Galkin 2018) but predictive accuracy is poor relative to methylation; treat as exploratory.
- **Imaging.** MRI-based multi-organ clocks (Nature Medicine 2025) and retinal/face age. Out of scope for a molecular tool but worth a plug-in slot.
- **EHR / clinical composite.** EMRAge (the target OMICmAge is trained on), frailty index (FI), intrinsic capacity (dnamic clock predicts it from methylation).
- **Telomere length.** Direct assays (qPCR T/S, TRF Southern, flow-FISH, TelSeq from WGS) and DNAm estimators (DNAmTL 140 CpGs ⟶ kb; PCDNAmTL ⟶ base pairs).

Chapter 20. 4. Algorithm families with full math

4.1 Penalised linear regression, the field's default

Objective for elastic net over samples i and features j:

```
min_β  (1/2n) Σ_i (y_i − β₀ − Σ_j x_ij β_j)²  +  λ [ (1−α)/2 · Σ_j β_j²  +  α · Σ_j |β_j| ]
```

α = 1 is LASSO, α = 0 is ridge. Horvath used α = 0.5. Application at inference is a dot product:

```
score = β₀ + Σ_j β_j x_j
```

then a clock-specific output transform (§5). Of the 177 clocks in pyaging's registry at commit e7267e9, 66 are elastic net, 18 LASSO, 5 ordinary linear regression, 1 ridge and 1 BLUP.

BLUP (Zhang 2019) treats all 319,607 probes as random effects with a common variance, `β = (X'X + λI)⁻¹X'y` in the mixed-model sense. It spreads small weights over all 319,607 probes, so it degrades less than elastic net when many probes are missing, as Zhang's script notes (line 91, DNAm-based-age-predictor pred.R). Crucially, both Zhang models **z-score within each sample across probes** before the dot product:

```
dataNona.norm <- apply(dataNona, 1, scale)   # per-individual scaling, probes x individuals
```

4.2 PC clocks

Higgins-Chen 2022. Instead of regressing age on individual CpGs, regress on the top principal components of a large CpG block (78,464 probes for most PC clocks, 125,175 for SystemsAge). This averages away probe-level technical noise and raises test-retest ICC dramatically.

```
score = ((x − μ) W) β + β₀
```

where μ is the training centre vector, W the p × k rotation (1,933 PCs for PCGrimAge), β the regression weights on PCs. Implemented as `PCLinearModel` (pyaging _base_models.py:72). The storage cost is the rotation matrix - 78,464 × k floats, which is why PC clocks are the memory-heavy members of any registry.

4.3 Survival models

Cox proportional hazards. `h(t|x) = h₀(t) exp(x'β)`. The linear predictor `x'β` is a log-hazard, unbounded, not in years. FALCONAge fits it by Newton–Raphson on the partial likelihood with Breslow's handling of tied event times, where the risk set at a time includes every subject still followed at that time, tied subjects included, and matches R's `survival::coxph(ties = "breslow")` to six decimals in its tests. Follow-up recorded in whole months, as NHANES is, makes ties the usual case rather than the exception; FALCONAge 1.0.0 left tied subjects out of each other's risk sets and was 1.1% off R on such data. `cox_hazard()` reports the hazard ratio per unit of the score, per year for an age clock. SenMortalityAge and

organAging's mortality models report this linear predictor; GrimAge, GrimAge2 and CpGPTGrimAge3 compute it internally and rescale it to years, as below.

Cox → years. GrimAge standardises the Cox score against training-set moments then rescales to the training age distribution:

```
years = ((cox − cox_mean) / cox_std) × age_std + age_mean
```

GrimAge v1: cox_mean 13.20127, cox_std 1.086805, age_mean 59.63951, age_std 9.049608. GrimAge2: cox_mean 15.370829484122, cox_std 1.09534876966487, age_mean 66.0943807965085, age_std 9.05974444998421 (pyaging _models.py:1683,1769).

Gompertz mortality → years. The PhenoAge route (§5.1) and the organAging route. organAging's published human Gompertz fit on the log scale: intercept −9.94613787413831, slope 0.0897860500778604, mean cohort age 57.29426 y. Convert a relative log-hazard `r` to years:

```
avg_r      = intercept + slope × mean_age            # = −4.801912
years      = (avg_r + r − intercept) / slope
```

That inverts the Gompertz model as stated. organAging's README (commit 5147b03, line 100) writes `(−avg_r + r) / slope − intercept`, which returns 63.4 rather than the cohort mean of 57.29 years at `r = 0`; a comparison with README-converted values has to say which expression it used.

Accelerated failure time. organAging also fits `WeibullAFTFitter(penalizer=0.01)` (lifelines) for time-to-death models (organAging clock_utils.py).

Cox elastic net. `CoxnetSurvivalAnalysis(alpha_min_ratio="auto", n_alphas=50)` from scikit-survival, grid-searched over `l1_ratio ∈ {0.5, 1.0}` with 3-fold CV - organAging's mortality models.

4.4 Klemera–Doubal method (KDM)

The most statistically principled of the clinical-chemistry methods. Fit one univariate regression of each biomarker on age, then invert them all jointly with precision weights.

For biomarker j: $x_j = q_j + k_j \cdot CA + \varepsilon_j$, with residual SD s_j and R^2 r_j.

$$BA_E = \Sigma_j (x_j - q_j)(k_j / s_j^2) \;/\; \Sigma_j (k_j / s_j)^2$$

KDM corrects `BA_E` toward CA using the residual variance:

$$BA = [\, \Sigma_j (x_j - q_j)(k_j/s_j^2) + CA / s_BA^2 \,] \;/\; [\, \Sigma_j (k_j/s_j)^2 + 1/s_BA^2 \,]$$

s_BA^2 is estimated as $s^2 - s_r$ where s^2 is the observed variance of `(BA_E − CA)` and

$$r_char = \Sigma_j |(k_j/s_j)\sqrt{r_j}| \;/\; \Sigma_j |k_j/s_j|$$
$$s_r = ((1 - r_char^2)/r_char^2) \times ((age_max - age_min)^2 / (12\ m))$$

with m the number of biomarkers. Verbatim in BioAge kdm_calc.R. BioAge trains separately by sex on NHANES III aged 30–75, non-pregnant, and projects into NHANES IV. Samples with more than 2 missing biomarkers are set to NA. Output columns: `kdm`, `kdm_advance = kdm − age`.

How FALCONAge follows it. `fa.models.fit_kdm` and `fa.models.clinical.kdm` reproduce the `kdm0` column BioAge ships with its NHANES data: on NHANES III to 0.0007 years mean absolute difference (maximum 0.007 years, n = 9,583), and projected into NHANES IV to 0.0007 years (maximum 0.005 years, n = 8,243). Three details carry that agreement. Each biomarker is regressed on the rows where that biomarker is present; `s_j` is the residual SD with divisor n − 1, as BioAge's `get_effs` takes it; and `s_BA²` is estimated on the reference rows that carry every biomarker, then reused when projecting into another cohort. That last choice follows the shipped `kdm0` rather than today's `kdm_calc.R`, which also averages over partially observed rows and reproduces `kdm0` only to r = 0.94. The shipped `kdm0` is complete-case (every row with a value carries all nine markers); `kdm(..., max_missing=0)` reproduces it exactly, including which rows are NA, and the default `max_missing=2` is the rule in today's code. FALCONAge 1.0.0 took the square root of the correlation in `r_char`, where the formula takes the square root of R^2, and was 1.77 years out on the same rows. The regression test runs on a fixture of those NHANES III rows (`python/tests/data/`). `fa.models.clinical.validate_panel(reference, full, reduced)` asks whether a reduced panel is still the same clock: both are fitted on the same reference rows, and their agreement and mortality hazard ratios are reported, the latter in BioAge's `table_surv` form (advance standardised within sex, Cox with age and sex, HR per SD). On NHANES III, a nine-marker panel against the twelve-marker one gives r = 0.978 for biological age, r = 0.895 for advance, and HR per SD of 1.353 against 1.349. `fa.models.clinical.kdm_bioage(df)` scores a cohort on the `kdm0` scale from a packaged fit of those rows (`registry/data/kdm_bioage_nhanes3.json`, built by `python/tools/build_kdm_bioage.py`), which is what most NHANES papers mean by KDM biological age. It takes BioAge's column names and units: FEV1 in mL, SBP in mmHg, total cholesterol in mg/dL, HbA1c in %, albumin in g/dL, creatinine in mg/dL, `lncrp` = ln(1 + CRP in mg/dL), ALP in U/L, BUN in mg/dL.

ComputAge also ships a *de novo* KDM estimator (ComputAge deage/kdm.py) with forward feature selection, LASSO preselection, 10-fold CV, RSE weighting, and stability testing, useful as a template for letting users fit their own KDM on arbitrary biomarker panels.

4.5 Homeostatic dysregulation (HD)

Mahalanobis distance from a young reference cohort. Standardise both the reference and target matrices using the reference means and SDs, then

$$HD_i = \sqrt{(x_i - \mu_{ref})' \, \Sigma_{ref}^{-1} \, (x_i - \mu_{ref})}$$

Reported both raw (`hd`, scaled by its own SD) and logged (`hd_log = log(HD)/sd(log HD)`). Both divisors are the scored cohort's own spread, so BioAge's values are not comparable between cohorts. FALCONAge's `hd()` returns the unscaled distance, which is comparable between cohorts scored against one reference, and `fa.models.clinical.bioage_hd_scale()` gives BioAge's two forms for comparison. The reference must be young (typically 20–30), non-pregnant, and within clinically normal biomarker windows; train separately by sex. BioAge hd_calc.R. Unlike KDM and PhenoAge, HD has no age units, it is a dimensionless distance, and it does *not* need to be residualised against CA the same way.

4.6 Mitotic / replicative clocks

- **epiTOC1** - plain mean β over 385 "ground-state hypomethylated" Polycomb-target CpGs. Missing sites encoded as −1 and excluded from the mean (pyaging _models.py:1988).

- **epiTOC2** - an explicit stem-cell division model. Given per-CpG methylation-transmission rate δ_c and ground-state β_0_c:

  ```
  tnsc = (2/k) Σ_c (β_c − β₀_c) / (δ_c (1 − β₀_c))
  ```

  k CpGs (163 for epiTOC2, 170 for epiTOC3). Units: estimated cell divisions per stem cell. pyaging _models.py:2008. The sum and k run over the sites the data carries, as in the authors' own code (EpiMitClocks, `colMeans` over the matched rows) and in dnaMethyAge. FALCONAge matches both to a relative 1e-9 on a test matrix with sites absent and values missing (`python/tests/data/SOURCE.md`). The authors also return a simplified estimate with every β_0 set to 0 (`tnsc2`), which dnaMethyAge reports under `simple_mode = TRUE`. It is not a rescaling of the full one: on 858 EPIC v1 blood samples it was a median 1.22 times larger. FALCONAge returns the full estimate, so a comparison has to state which one the other package reported.
- **stemTOC** - 0.95 quantile of β over 371 (or 629 in-vitro) CpGs rather than the mean, which makes it robust to cell-type dilution.
- **HypoClock** - mean β over 678 solo-WCGW hypomethylating sites, returned as `1 − mean`.
- **epiCMIT-hyper / epiCMIT-hypo**: the mean over 184 hypermethylating, and one minus the mean over 1,164 hypomethylating, B-cell CpGs.
- **RepliTali** - elastic net on 87 EPIC CpGs predicting population doublings in cultured cells; `RepliTaliNorm` is its 218-CpG upstream normaliser exposed separately.

4.7 GrimAge architecture

Two-stage. Stage 1: eight (v1) or ten (v2) independent elastic nets predict plasma protein concentrations and smoking pack-years from methylation. Stage 2: a Cox model on those DNAm surrogates plus age and sex.

v1 components: PACKYRS, ADM (adrenomedullin), B2M, CystatinC, GDF15, Leptin, PAI-1, TIMP-1. v2 adds logCRP and logA1C. Feature counts (pyaging registry): ADM 187, B2M 92, CystatinC 88, GDF15 138, Leptin 187, logA1C 87, logCRP 132, PACKYRS 173, PAI-1 211, TIMP-1 43; the composite model touches 1,032 CpGs plus age and female indicator.

Forward pass (pyaging _models.py:1656), concat order matters and is fixed: `[GDF15, B2M, CystatinC, TIMP1, ADM, PAI1, Leptin, PACKYRS, (LogCRP, A1C), Age, Female]`.

PCGrimAge inserts a PCA step: centre 78,464 CpGs, rotate to 1,933 PCs, append Female and Age, then run eight PC-based sub-clocks and the Cox layer.

CpGPTGrimAge3 / CpGPTPCGrimAge3 (2024) replace most sub-clocks with CpGPT-derived protein proxies. Required inputs: age, seven GrimAge2 proxies (`grimage2timp1`, `grimage2packyrs`, `grimage2logcrp`, `grimage2adm`, `grimage2leptin`, `grimage2gdf15`, `grimage2pai1`) and 16 CpGPT protein proxies (`cpgpt_s100a9`, `tnfrsf13c`, `tgfb1`, `tek`, `ccl14`, `tnfsf15`, `lilrb2`, `tnf`, `chit1`, `postn`, `il34`, `pdcd1`, `cst3`, `cxcl2`, `gzma`, `il5`). 24 features total for the non-PC variant, 31 for the PC variant (which rotates 30 → 29 PCs). Trained on Framingham Heart Study whole blood, Illumina 450K, mortality endpoint (author clarification recorded in pyaging audit_report.md).

4.8 DunedinPACE and DunedinPoAm

DunedinPoAm38 (2020) - 46 CpGs, elastic net on the 12-year, 3-timepoint Pace of Aging from the Dunedin cohort. In-sample r = 0.6.

DunedinPACE (2022) - 173 scoring CpGs, elastic net on the 20-year, 4-timepoint Pace of Aging; CpGs restricted to probes with test-retest ICC > 0.4 (n = 81,239 candidates). In-sample r = 0.78. The published algorithm quantile-normalises the sample against a **gold-standard mean vector** before scoring. pyaging implements the official 20,000-probe panel: 173 scoring CpGs plus 19,827 background probes used only to define the sample's rank distribution (pyaging _models.py:238):

```
sorted_gold = sort(reference_values)                        # length 20000
q_idx       = linspace(0, len(sorted_gold)-1, x.shape[1])  # rank → gold quantile map
for each sample row:
    normalized[argsort(row)] = sorted_gold[q_idx]
return normalized[:, scoring_indices]
```

This is a per-sample rank transform, so **the background probes must be present** - dropping them silently changes the answer.

4.9 Mammalian universal clocks

Three variants, all elastic net on the HorvathMammalMethylChip40, each with a different age parameterisation and a species lookup table (AnAge: gestation time, average age at sexual maturity, maximum lifespan).

- **Clock 1** - predicts `log(age + 2)`; inverse `age = exp(ŷ) − 2`.
- **Clock 2** (relative age), predicts a log-log-transformed relative age. Inverse:

```
rel   = exp(−exp(−ŷ))
age   = rel × (max_age + gestation) − gestation
```

- **Clock 3** (log-linear age relative to maturity), with `m̂ = 5 (gestation / maturity)^0.38`:

```
ŷ ≥ 0 : age = m̂ (maturity + gestation)(ŷ + 1) − gestation
ŷ < 0 : age = m̂ (maturity + gestation) exp(ŷ) − gestation
```

Species is passed as a one-hot block appended to the CpG vector (1,756 species for clock-2 variants, 1,707 for clock-3 variants); the model splits the input tensor by these widths (pyaging _models.py:513–771). Also in this family: MammalianLifespan (predicts species maximum lifespan, `exp` link) and MammalianFemale (sex classifier, sigmoid).

4.10 Deconvolution as a model

Reference-based, constrained. Given a reference table X (CpGs × cell types) of sorted-cell mean methylation and a sample's betas y, the proportions p minimise $\|Xp - y\|^2$ with $p \geq 0$ (Houseman et al. 2012). FALCONAge solves it as the authors' FlowSorted.Blood.EPIC does in `projectCellType_CP`, with the arguments its documentation gives for a beta matrix (`nonnegative = TRUE, lessThanOne = FALSE`): per sample, over the CpGs that sample observes, rounded to four decimals. With non-negativity the only constraint,

the problem is non-negative least squares, whose solution is unique, and `scipy.optimize.nnls` returns it. The proportions are not forced to sum to one; the sum is how much of the sample the reference explains. On 858 whole-blood EPIC samples from one cohort, the six proportions equal minfi's `estimateCellCounts2` output for every sample and cell type, to the four decimals it reports.

pyaging solves a different problem: an unconstrained fit through the pseudo-inverse, then a **Euclidean projection onto the probability simplex** (pyaging _models.py:1269):

```
p_raw = x @ pinv(S).T
sorted_v = sort(p_raw, descending)
cssv     = cumsum(sorted_v)
rho      = max{ i : sorted_v[i] − (cssv[i] − 1)/(i+1) > 0 }
theta    = (cssv[rho] − 1) / (rho + 1)
p        = clamp(p_raw − theta, min=0)
```

It vectorises across a batch, but it is not the constrained fit: the projection forces the sum to one and is taken in the geometry of p rather than of the least-squares problem. On the same 858 samples and the same reference it adds 0.0058 to every cell type on average (at most 0.0099), the 3.6% the reference leaves unexplained spread evenly, which is a relative 11% for B cells at a mean of 0.053. pyaging's libraries are its own 600-CpG 6-cell and 240-CpG 12-cell selections. FALCONAge ships the published IDOL libraries of Salas et al. 2018, 450 CpGs on EPIC and 350 on 450K (FlowSorted.Blood.EPIC, GPL-3). The 12-cell IDOL-Ext library (Salas et al. 2022) is under a Dartmouth research-use licence, so those entries are `licensed` and read a table the user registers.

4.11 Binarisation clocks

Two independent lines converged on the same idea: with sparse count data, the *identity* of expressed genes carries more robust age signal than their levels.

- **BiT age** (*C. elegans*, Meyer & Schumacher 2021). Per sample: mask zeros, compute the median of non-zero CPMs, set gene = 1 if CPM > median else 0. Score = Σ (binary × elastic-net coef)
  - intercept. Intercept 103.54631743289005 for *C. elegans* (hours); 59.626 for human fibroblasts. 576 predictor genes. AgingClock biological_age_prediction.py.
- **CellBiAge** (mouse brain single cell, Yu et al., *Cell Reports* 42:113500, Dec 2023). Pipeline: log-normalise → CCA batch integration → scale (mean 0, SD 1 per gene) → **binarise at 0** (scaled value > 0 → 1). Feature space: top 2,000 highly variable genes (HVGs beat highly expressed genes decisively; 1,413 shared between train/test batches in the hypothalamus data). The mean threshold beat BiT age's median threshold on their test data.

CellBiAge numbers worth carrying into a benchmark suite. Metric is AUPRC (test sets are class-imbalanced), models are elastic-net logistic regression, random forest, gradient boosting, SVM, XGBoost and an MLP, tuned with GridSearchCV / KerasTuner under group CV:

Setting	Before binarisation	After
Hypothalamus, ElasticNet	0.60	0.94
Hypothalamus, all five models	~0.65	~0.96
SVZ control, MLP and ELN	0.67	0.90
Train/test batch swapped	-	0.94

Setting	Before binarisation	After
Top 80 genes only	-	~0.95
Top 40 genes only	-	0.90

Data: female C57BL/6 hypothalamus snRNA-seq, 40,064 nuclei, 3 vs 24 months, two batches on different chemistry (10x v3 + NovaSeq for train, v2 + HiSeq for test); male SVZ scRNA-seq, 79,123 cells, 15 animals, 6 vs 23 months, sedentary vs 5 weeks voluntary wheel running. Exercise significantly reduced predicted-aged probability in young-animal neuroblasts ($p = 0.027$) and activated NSC/NPCs ($p = 0.031$); no effect in aged animals or in the all-cell pooled analysis. Only 4 genes (*Nes*, *Ly6a*, *Slc5a4b*, *B230312C02Rik*) were shared among the top 200 features across all cell types. CellBiAge slightly outperformed the Buckley 2023 bootstrap clock on the three shared cell types while avoiding the 100-pseudo-cell bootstrap.

4.12 Rank-based transcriptomic clocks

Pasta and REG (2025) fill NaNs with the global median, then rank-normalise **within each sample** with average ranks for ties, then apply a ridge-logistic (Pasta) or ridge (REG) linear layer over 8,113 genes. Pasta's output transform is `x·scale + offset·scale`; REG's is `x + intercept`. Rank normalisation makes the model invariant to any monotone per-sample transformation, which is why these clocks work across RNA-seq and microarray without re-normalising. pyaging _models.py:1037–1189.

4.13 Spatial single-cell clocks

Sun et al., *Nature* 638:160–171 (2025). Per cell type (18 brain cell types), a LASSO model with 5-fold CV over 20 alphas, fitted on spatially smoothed expression.

SpatialSmooth. Build a spatial graph per animal (fixed k = 20 neighbours, binary adjacency S), then iterate to convergence:

```
X_{t+1} = (1 − α) X₀ + α S X_t ,   α = 0.8, max_iter = 30, tol = 1e-2
```

This is label propagation applied to expression. Missing genes are imputed with per-gene training means (or SpaGE, which projects from a reference scRNA-seq atlas via shared principal vectors). Age acceleration is defined **within animal and cell type**, as prediction minus the mean prediction for that (animal, cell type) pair, which cancels the animal-level batch and calendar age. The package also computes **cell proximity effects**: for each effector cell type, split target cells into "near" and "far" sets by distance cutoff (optionally a ring of given width), then t-test the age acceleration difference. Outputs a data frame with `Near Cell`, `AgeAccel Cell`, cutoff multiplier `n`, `t`, `p`, `Aging Effect`, `Near Freq`, `Near Num`.

4.14 Single-cell methylation clocks

- **scAge** (Trapp 2021, Python). Uses the bulk CpG–age linear relationships as priors and computes, for each cell, the age maximising the product of per-CpG likelihoods given the binary methylation call. Handles arbitrary, non-overlapping CpG coverage per cell.
- **scEpiAge** (Bonder 2024, *Nat Commun* 15:7567). Inverts the regression: predicts methylation *from* age, so cells need not share CpGs. Mouse blood 10–101 weeks; DNAm-age distribution is wider than technical noise predicts, and CD4+/CD8+ T cells read younger than B cells. Cell-type effect exceeds age effect, a warning for any single-cell aging score.

4.15 Deep learning

- **AltumAge** (2022) - MLP over 20,318 CpGs shared across 27K/450K/EPIC. Architecture from pyaging _base_models.py:89: BatchNorm(20318) ⟶ Linear(20318,32) ⟶ SELU ⟶ [BatchNorm(32) ⟶ Linear(32,32) ⟶ SELU] × 4 ⟶ BatchNorm(32) ⟶ Linear(32,1). BatchNorm eps 0.001, momentum 0.99 (Keras defaults, preserved on port). Preprocessing is scikit-learn's **RobustScaler**, `(x − median) / IQR` with the training set's median and interquartile range; pyaging stores the IQR under the name `std`.
- **CpGPT** (de Lima Camillo et al., bioRxiv 2024.10.24.619766), a transformer pretrained on more than 2,000 datasets and over 150,000 samples. Encodes sequence context, genomic position and epigenetic state per CpG; imputes and reconstructs genome-wide profiles from partial input; transfers across platforms, tissues, mammalian species and single-cell data. Fine-tuned heads give SOTA mortality and chronological-age prediction. Code: github.com/lucascamillomd/CpGPT.
- **GraphAge** (PNAS Nexus 2025) - GNN over a CpG regulatory graph; interprets which CpG interactions matter and how their importance shifts with age.
- **DeepStrataAge** (npj Aging 2026) - DNN on 12,234 CpGs, MAE 1.89 y across >20,000 blood methylomes, with SHAP attributions revealing sex-specific "epigenetic waves".
- **EpInflammAge** - epigenetic + inflammatory markers, MAE ~7 y, r = 0.85 in healthy controls.
- **XAI-AGE** - pathway-structured network giving pathway-level activity as an interpretable intermediate.
- **PointNet** - used by scImmuAging alongside LASSO and random forest for PBMC cell-type clocks.

4.16 SystemsAge

Sehgal et al., *Nature Aging* 5:1880-1896 (2025). One blood methylome ⟶ 11 system-specific ages plus a composite. Forward pass (pyaging _models.py:2617):

1. Centre 125,175 CpGs and rotate to DNAm PCs.
2. Project PCs onto a `system_vector` to obtain system components.
3. Per system, a linear module over its component indices ⟶ 11 system scores.
4. Separately, a PC ⟶ predicted-age model, with a quadratic polynomial correction $a + b\cdot\hat{y} + c\cdot\hat{y}^2$, then divide by 12 (months ⟶ years).
5. Concatenate the 11 system scores with predicted age; for the composite, run a second PCA-based model over that 12-vector.
6. Affine calibration per output: $((\text{raw} - c_0)/c_1) \times c_3 + c_2$, then /12.

Systems: Blood, Brain, Inflammation, Heart, Hormone, Immune, Kidney, Liver, Metabolic, Lung, MusculoSkeletal (indices 0–10 in the pyaging classes), with index −1 for the composite. In a reliability study of 18 clocks (Sehgal et al., bioRxiv 2025.10.13.682176), the PC-based clocks, PCGrimAge and SystemsAge in particular, stayed technically reliable across slide positions and DNA extraction protocols.

4.17 Ensembling and stacking

- **EnsembleAgeStatic / EnsembleAgeStaticTop** (mouse, 2025), elastic net over 288 / 431 CpGs designed to be responsive to lifespan-modulating interventions.
- **EnsembleAgeHumanMouse** - 100 CpGs on the mammalian array, cross-species relative age.
- **StackAge** (2026), ensemble over UK Biobank plasma proteomics + metabolomics, n = 30,376; improves risk prediction for 12 chronic diseases over single-omic clocks.

- **DNAmFitAge** - a Klemera–Doubal composite over four DNAm-predicted fitness measures. It fits sex-specific models. The four inputs are standardised with published slopes and intercepts (pyaging _models.py:1875):

Input	Female (offset, slope)	Male (offset, slope)
VO2max	46.825091, −0.13620215	49.836389, −0.141862925
Grip	39.857718, −0.22074456	57.514016, −0.253179827
Gait	2.508547, −0.01245682	2.349080, −0.009380061
GrimAge	7.978487, 0.80928530	9.549733, 0.835120557

  Each is transformed `(value − offset)/slope`, then a sex-specific KDM layer produces the age.

4.18 tAge, relative transcriptomic clocks

Tyshkovskiy et al., *Nature* 654:173–188 (2026), from >11,000 transcriptomes across >25 tissues in mouse, rat, macaque and human. 60 distributed models: {EN, BR} × {Chronological, Mortality, Normalized age} × {Mouse, Multispecies, Rodents} × {Multi-Tissue, Brain, Kidney, Liver, Skeletal Muscle} × {Scaled, YuGene}.

Preprocessing chain (tAge preprocessing.R):

1. `filter_genes` - keep genes with count ≥ 10 in ≥ 20% of samples.
2. `map_genes` - to mouse Entrez space via ortholog tables.
3. `RLE_normalization` - `sweep(counts, 2, libsize × calcNormFactors(RLE), "/") × 1e7`.
4. `log_transform` - `log10(x + 1)`.
5. `scale_eset` - **column-wise** z-score, i.e. per sample across genes.
6. `YuGene` - per sample: shift to non-negative, sort descending, cumulative proportion, carry forward for ties, value = `1 − cumprop`. A non-parametric rank/uniformisation.
7. `.align_to_gene_list` - reorder to the reference gene list, pad missing genes with **NA** (not zero) so the model's own `SimpleImputer` fills them with training medians.
8. `control_subtraction` - subtract the per-gene median of a reference group, or of all samples when no reference is given. **All distributed clocks are relative (`_diff`) models; centring is never skipped.**

Output units, from the clock registry (tAge clocks_metadata.csv):

Outcome	Units	Rescaled by max lifespan?
Chronological	years (human) / months (rodents)	Yes - × species max lifespan
Mortality	log10(hazard ratio)	No
Normalized age	fraction of max lifespan	No

Species maximum-lifespan factors in tAge 87352dd: human 122 and monkey 39 (years), mouse 48 and rat 45.6 (months) (tage_predict.py). Mortality clocks must never be multiplied by these, a mistake the registry flag `lifespan_scaled` exists to prevent.

The paper's biology: genes for senescence, inflammation and apoptosis go up with age across species and cell types; wound healing, differentiation and ECM synthesis go down. CDKN1A and LGALS3 are universal markers whose protein levels also track mortality and multimorbidity in UK Biobank. Modules (inflammation, energy production, ECM organisation) get their own clocks so an intervention's primary axis of action can be read off.

4.19 SCALE - literature-informed single-cell aging score

Mao et al. A weighted, expression-detection-frequency-scaled z-score over a curated aging gene set (4_SCALE_score.R):

```
weights = c(rep(1,50), rep(-1,50)) *
          rowSums(counts[aging_genes,] > 0) / ncol(counts)   # detection frequency
z       = t(scale(t(logdata[aging_genes,])))                  # gene-wise z across cells
score   = t(z) %*% weights
```

The 50 up / 50 down genes per tissue come from glmnet selection over a union of curated aging resources: GenAge, CSGene, AgeFactDB, Aging Atlas, GO:aging, GO:CellAge, plus manual curation. Applied to Tabula Muris Senis across 20+ tissues. Validated against single-cell entropy and somatic mutation accumulation, and against transcriptomic and methylation clocks in muscle stem cells, Alzheimer's, and heterochronic parabiosis. Correlation with age is assessed four ways: Pearson, distance correlation (`energy::dcor`), MIC (`minerva`), and nonlinear correlation (`nlcor`), a good template for reporting monotone-but-nonlinear age relationships.

4.20 Transform reference table

Every clock's output transform, extracted from pyaging _postprocessing.py and pyaging _models.py:

Name	Formula	Used by
identity	y	most elastic-net clocks
`anti_log_linear` (Horvath)	$y<0 : 21\cdot e^{y} - 1 ; y\geq 0 : 21\cdot y + 20$	Horvath2013, SkinAndBlood, PedBE, Han, PCHorvath2013, PCSkinAndBlood, IntrinClock, CorticalClock, Pipek×3
Horvath with adult_age 48 ⟶ years	same with 49/48, then /12	Wu (children)
`anti_logp2`	$e^{y} - 2$	Mammalian1
`anti_log`	e^{y}	MammalianLifespan
`anti_log_log`	`e^(-e^(-y))`	Mammalian2 relative-age step
`mortality_to_phenoage`	see §5.1	PhenoAge
`cox_to_years`	$((y-\text{cox_mean})/\text{cox_std})\cdot \text{age_std} + \text{age_mean}$	GrimAge, GrimAge2, CpGPTGrimAge3, CpGPTPCGrimAge3
`petkovichblood`	$\text{age} = ((y + 1.712)/0.1666)^{(1/0.4185)} / 30.5$	Petkovich (mouse, months)
`stubbsmultitissue`	$\text{age} = (\exp(0.1207y^{2} + 1.2424y + 2.5440) - 3) \times 7/30.5$	Stubbs (mouse, months)
`days_to_months`	$y / 30.5$	Meer
`days_to_weeks`	$y / 7$	Bohlin, EPICGA
`sigmoid`	`1/(1+e^(-y))`	MammalianFemale, CVDWesterman, ADBahadoSingh; pyaging also applies it to the McCartney predictors, which the paper defines without a link and FALCONAge scores as the plain weighted sum (§15.2)

Name	Formula	Used by
`one_minus`	`1 − y`	HypoClock, epiCMIT-hypo
`add_constant`	`y + c`	REG, Lin (+12.2169841), PhenoAgeV1 in ComputAge (+60.664), YingCausAge (+86.80816381), YingDamAge (+543.4315887), YingAdaptAge (−511.9742762)
`scale_and_shift`	`y·s + o·s`	Pasta, PastaMouse

Note the intercept convention differs between packages: ComputAge/biolearn carry the intercept in the `transform` lambda, pyaging bakes it into the `nn.Linear` bias. A merged tool must pick one and normalise on import.

Chapter 21. Published formulas and constants

5. Exact published formulas and constants

5.1 PhenoAge (Levine 2018), clinical-biomarker version

The paper prints the model to four decimals in Table 1 and Supplementary Table S1, with γ and the inversion to years in Supplement 1. BioAge carries the same fit at full precision. **Both are in the paper's units: SI, except CRP, which enters as ln(CRP) with CRP in mg/dL.**

As printed (Levine 2018 Table 1 and Supplement 1), FALCONAge's default:

```
xb = −19.9067
     − 0.0336 · albumin          (g/L)
     + 0.0095 · creatinine       (μmol/L)
     + 0.1953 · glucose          (mmol/L)
     + 0.0954 · ln(CRP)          (CRP in mg/dL)
     − 0.0120 · lymphocyte%      (%)
     + 0.0268 · MCV              (fL)
     + 0.3306 · RDW              (%)
     + 0.0019 · ALP              (U/L)
     + 0.0554 · WBC              (10³/μL)
     + 0.0804 · age              (years)

γ        = 0.0076927
m        = 1 − exp( −exp(xb) · (exp(120γ) − 1) / γ )
PhenoAge = 141.50225 + ln( −0.00553 · ln(1 − m) ) / 0.090165
```

Full precision (BioAge `phenoage0`, BioAge phenoage_calc.R), `coefficients="bioage"`:

```
xb = −19.90667
     − 0.03359355 · albumin      (g/L)
     + 0.009506491 · creatinine  (μmol/L)
     + 0.1953192  · glucose      (mmol/L)
     + 0.09536762 · lncrp        (BioAge: lncrp = ln(1 + CRP), CRP in mg/dL)
     − 0.01199984 · lymphocyte%  (%)
     + 0.02676401 · MCV          (fL)
     + 0.3306156  · RDW          (%)
     + 0.001868778 · ALP         (U/L)
     + 0.05542406 · WBC          (10³/μL)
     + 0.08035356 · age          (years)

m        = 1 − exp( (−1.51714 · exp(xb)) / 0.007692696 )
PhenoAge = 141.50225 + ln( −0.0055305 · ln(1 − m) ) / 0.090165
```

Every full-precision weight rounds to its Table 1 value, so these are the fit the table was printed from. On BioAge's 8,924 NHANES III rows the printed table puts PhenoAge a mean 0.074 years above the full-precision fit (at most 0.14 years). With `coefficients="bioage"` and `crp_transform="log1p"` FALCONAge reproduces `phenoage0` on those rows to below 0.00001 years.

Four transcription errors circulate, and all four appear in the equation printed in Liu et al. 2018 (PLoS Med 15:e1002718): alkaline phosphatase 0.00188 (Table 1: 0.0019), glucose 0.0195 (0.1953), RDW 0.3356 (0.3306), and the divisor 0.09165 (Supplement 1: 0.090165). FALCONAge carried 0.00188 until this was checked. The divisor is the costly one: BioAge corrected it in its April 2026 commit ("Update PhenoAge constant to 0.090165 and recompute data"), and code still using 0.09165 shifts results by several years. Guard `m` away from 0 and 1 or `ln(1 − m)` blows up.

The units are part of the formula. The weights are per SI unit. An implementation that applies them to albumin in g/dL, creatinine and glucose in mg/dL and CRP in mg/L without converting, as mcp-phenoage-clock does (phenoage.ts), returns nonsense: for the healthy 50-year-old of Appendix F it gives xb = 8.36 and a ten-year mortality of 1, where the converted inputs give xb = −9.07 and a PhenoAge of 41.85 years.

The CRP term is where BioAge departs from the paper. Levine 2018 enters ln(CRP). BioAge's `lncrp`, the variable its NHANES PhenoAge (`phenoage0`) is computed from, is ln(1 + CRP) (`data-raw/nhanes_all.R`, `lncrp = log(1 + crp)`). The difference is not small. On NHANES III (n = 8,924) FALCONAge with ln(CRP) sits a mean 1.50 years below `phenoage0` (SD 0.45), and not by a constant: ln(1 + CRP) − ln(CRP) = ln(1 + 1/CRP), which is largest at low CRP. With ln(1 + CRP) the gap falls to the 0.074 years of coefficient rounding, and with `coefficients="bioage"` as well it closes. FALCONAge follows the paper, records BioAge's choice as a known discrepancy, and offers `phenoage(df, crp_transform="log1p", coefficients="bioage")` for comparison with BioAge output. Every PhenoAge computed through BioAge carries the shift.

DNAm PhenoAge is the methylation surrogate of this quantity: 513 CpGs, elastic net trained to predict the clinical PhenoAge. pyaging exposes both, `phenoage` (clinical, blood chemistry) and `dnamphenoage` (methylation).

5.2 pyaging generalised PhenoAge postprocess

The torch implementation with λ = 0.0192 (pyaging's `PhenoAge.postprocess` in `src/pyaging/models/_models.py`, commit `e7267e9`):

```
mortality = 1 − exp( −exp(x) · (exp(120λ) − 1) / λ )
age       = 141.50225 + ln( −0.00553 · ln(1 − mortality) ) / 0.090165
```

5.3 BiT age (*C. elegans*) second correction

The published biological ages are rescaled to account for truncation of the lifespan distribution. With median lifespan 372 h (15.5 d) and SD 192/3 h (8/3 d) (AgingClock biological_age_correction.py):

```
corrected(y) = y − erfinv( 0.5 − ( ( erf(((372 − y)/sd)/√2)/2 ) + 0.5 ) / 2 ) · √2 · 2 · sd
```

Lifespan fold-change with error propagation uses `uncertainties.ufloat` with a 5% lifespan-assay bias and 0.5 d age-reporting bias; biological age is rescaled as `chronological_age × 15.5 / reported_median_lifespan`.

5.4 Zhang 2019 predictors

514 elastic-net probes or 319,607 BLUP probes, trained on 13,566 samples. Per-sample z-scoring of probes is mandatory (§4.1). Missing values are filled with the per-probe mean across individuals; probes that are NA across all individuals are dropped (DNAm-based-age-predictor pred.R). BLUP degrades more gracefully with missing probes than EN. RMSE reported as 2.04 y, the most accurate chronological clock, and the one whose acceleration stopped predicting mortality.

FALCONAge standardises over every probe the data carries, exactly as `pred.R` does, before the weights are applied, and leaves an absent model probe at 0 (the sample's own mean) as `pred.R` does by restricting the sum to the probes present. It refuses when the data carries less than `min_coverage` of the 319,607 probes the models were fitted on, because a subset's per-sample mean and SD are not the array's. On the authors' example (10 samples, all 485,512 probes of the 450K) it matches `pred.R` to 1e-13 years for the elastic net and 5e-12 for BLUP.

The scope of that standardisation is the part implementations disagree on. pyaging (commit `e7267e9`) takes the row mean and SD over the model's own inputs, 514 probes for the elastic net. On the same 10 samples that places the elastic-net estimate 9.9 to 22.7 years below `pred.R` (mean 17.7), and its mean absolute error against chronological age rises from 4.7 to 21.3 years. The BLUP estimate moves by 0.6 to 2.0 years, because its model already spans 319,607 of the 485,512 probes.

Appendix F - Reproducing the four reference computations by hand

Small worked checks to use as smoke tests while building.

PhenoAge, from conventional units. Inputs: age 50, albumin 4.5 g/dL, creatinine 0.9 mg/dL, glucose 90 mg/dL, CRP 1.0 mg/L, lymphocyte 30%, MCV 90 fL, RDW 13%, ALP 70 U/L, WBC 6.0 ×10³/µL. Convert first, to the units of Table 1: albumin 45 g/L, creatinine 79.5615 µmol/L, glucose 4.9950 mmol/L, CRP 0.1 mg/dL. Then

```
xb = −19.9067 − 0.0336(45) + 0.0095(79.5615) + 0.1953(4.9950) + 0.0954·ln(0.1)
     − 0.0120(30) + 0.0268(90) + 0.3306(13) + 0.0019(70) + 0.0554(6.0) + 0.0804(50)
   = −9.0718
m        = 1 − exp( −exp(xb) · (exp(120·0.0076927) − 1) / 0.0076927 )
PhenoAge = 141.50225 + ln(−0.00553·ln(1 − m)) / 0.090165  =  41.85 years
```

Put the unconverted numbers into the same weights and xb is 8.36, m is 1, and there is no age.

Horvath output transform. A raw linear score of −0.5 becomes `21·exp(−0.5) − 1 = 11.74` years; a score of 1.2 becomes `21·1.2 + 20 = 45.2` years.

GrimAge2 Cox → years, with the constants of pyaging's `GrimAge2.postprocess` (FALCONAge ships GrimAge2 as a licensed scaffold, so these are not its numbers to publish). A Cox linear predictor of 16.0 becomes `((16.0 − 15.370829484122)/1.09534876966487) × 9.05974444998421 + 66.0943807965085 = 71.30` years.

tAge chronological, mouse. The chronological clocks predict age divided by the species' maximum recorded lifespan, and the mortality clocks predict log10 of the expected mortality rate (Tyshkovskiy et al., Nature 2026); the mouse's maximum recorded lifespan is 4 years in AnAge. A model output of 0.35 (fraction of maximum lifespan) becomes `0.35 × 48 = 16.8` months. The same 0.35 from a mortality clock stays as `log10(HR) = 0.35`, i.e. HR ≈ 2.24 - never 16.8 anything.

Checked against the sources (2026-09-25). 5.1 against Levine et al. 2018, Table 1, Supplementary Table S1 and Supplement 1, and BioAge's `phenoage_calc.R`; 5.2 and the GrimAge2 constants against pyaging at commit `e7267e9`; 5.3 against `src/biological_age_correction.py` and `src/lifespan_prediction_with_error_propagation.py` in Meyer-DH/AgingClock; 5.4 against Zhang et al. 2019 (the 2.04-year RMSE and the absent mortality association in both Lothian cohorts) and their `pred.R`; the Horvath transform against dnaMethyAge's `HorvathS2013_transform`; the tAge scales against Tyshkovskiy et al. 2026 and the AnAge entry for *Mus musculus*.

Chapter 22. 7. The clock catalogue

pyaging's registry is the most complete machine-readable inventory in existence: at commit e7267e9, **177 clocks** with curated provenance (pyaging clock_metadata.json), backed by an evidence ledger of 6 author-confirmed, 332 code-confirmed, 2,435 paper-confirmed and 59 supplement-confirmed field assignments. This chapter describes that registry; FALCONAge's own catalogue of 178 clocks is in the clock catalogue.

7.1 Distributions

Counted from pyaging's `clock_metadata.json` at commit e7267e9 (177 clocks).

Data type. DNA methylation 156, histone modification 8, transcriptomics 6, clinical biomarkers 5, chromatin accessibility 2.

Species. *Homo sapiens* 157, multiple species 11, *Mus musculus* 7, *C. elegans* 1, human and mouse 1.

Platform. Illumina 450K 92, EPIC 87, 27K 13, MammalMethylChip40 10, ChIP-seq 8, RNA-seq 6, clinical laboratory assays 5, RRBS 3, expression microarray 3, mammalian methylation array 2, MammalMethylChip320 2, ATAC-seq 2, bisulfite sequencing 2, targeted bisulfite sequencing 1.

Model type. Elastic net 66, LASSO 18, reference-based constrained deconvolution 18, PCA + elastic net 18, PCA + elastic net + ARD 8, linear regression 5, weighted linear score 3, causality-weighted elastic net 3, and two each of logistic regression, single-CpG score, PCA + Cox, Klemera–Doubal composite, mean methylation aggregation, dynamic methylation transmission, elastic net Cox, 95th-percentile aggregation and PCA. One each of: deep neural network, Cox proportional hazards, elastic net Cox ensemble, bootstrap-stabilised elastic net, complement of mean methylation, two-stage elastic net + Cox, feature-selected logistic, Mahalanobis distance composite, mean aggregation, two-stage elastic net, ridge logistic, orthologue-transferred ridge logistic, penalised hazards with Gompertz calibration, Gompertz hazards with age calibration, weighted methylation aggregation, ridge, elastic net logistic, quadratically calibrated elastic net, screened elastic net and BLUP.

Unit. Years 86, proportion 20, unitless 18, pg/mL 7, beta value 6, weeks 6, months 4, probability 3, and two each of biological years per chronological year, days, cell divisions per stem cell, pack-years, ratio, log odds, population doublings and log hazard. One each of hours, passages, m/s, kg, mL/kg/min, kb, relative age, ln(%), ln(mg/L), units per week and bp.

Population. Adults 78, older adults 29, all ages 27, multiple mammalian species 11, human age unspecified 7, mice 6, newborns 4, human cell cultures 4, pregnancies 4, and one each of *C. elegans*, centenarians, humans and mice, children and adolescents, adult men, prenatal and newborn, and children.

7.2 Selected clocks with their key parameters

The full 177-row table is in the registry; this is the subset a new tool should support first, sorted by likely usage.

Clock	Year	Layer	Features	Model	Predicts	Unit	Notes
horvath2013	2013	DNAm multi-tissue	353	elastic net	CA	years	anti_log_linear; the reference clock
hannum	2013	DNAm blood	71	elastic net	CA	years	
lin	2016	DNAm blood	99	linear	CA	years	intercept 12.2169841
skinandblood	2018	DNAm skin/blood/ fibroblast	391	elastic net	CA	years	most accurate first-gen across tissues
phenoage	2018	clinical chem	10	Gompertz-calibrated penalised hazards	phenotypic age	years	see §5.1
dnamphenoage	2018	DNAm blood	513	elastic net	phenotypic age	years	
zhangen	2019	DNAm blood/saliva	514	elastic net	CA	years	per-sample z-score
zhangblup	2019	DNAm	319,607	BLUP	CA	years	per-sample z-score
grimage	2019	DNAm blood	1,032	2-stage EN + Cox	mortality	years	8 sub-clocks
grimage2	2022	DNAm blood	1,032	EN Cox	mortality	years	10 sub-clocks (+logCRP, +logA1C)
dunedinpoam38	2020	DNAm blood	46	elastic net	pace	bio-y/chron-y	
dunedinpace	2022	DNAm blood	20,000 (173 scoring)	elastic net	pace	bio-y/chron-y	quantile norm required
altumage	2022	DNAm multi-tissue	20,318	DNN	CA	years	5 hidden layers, SELU
pchorvath2013 / pchannum / pcphenoage / pcgrimage / pcskinandblood / pcdnamtl	2022	DNAm	78,464 (78,466 for PCGrimAge)	PCA + EN	various	years / bp	high test-retest ICC
hrsinchphenoage	2022	DNAm blood	959	weighted linear score	phenotypic age	years	
yingcausage / yingdamage / yingadaptage	2024	DNAm blood	585 / 1,089 / 999	causality-weighted EN	CA / damage / adaptation	years	MR-filtered CpGs

Clock	Year	Layer	Features	Model	Predicts	Unit	Notes
dnamfitage	2023	DNAm blood	630	Klemera–Doubal composite	fitness bio-age	years	sex-specific
dnamtl	2019	DNAm blood	140	elastic net	telomere length	kb	
stemtoc	2024	DNAm multi-tissue	371	0.95-quantile aggregation	mitotic age	beta	
epitoc1 / epitoc2 / epitoc3	2016/2020/2020	DNAm	385 / 163 / 170	mean / transmission model	mitotic age	beta / divisions per stem cell	
systemsage + 11 system variants	2025	DNAm blood	125,175	PCA + EN	11 system ages + composite	years	
cpgptgrimage3	2024	DNAm-derived proxies	24	Cox	mortality/bio-age	years	needs CpGPT + GrimAge2 proxies
cpgptpcgrimage3	2024	DNAm-derived proxies	31	PCA + Cox	mortality/bio-age	years	
intrinclock	2024	DNAm multi-tissue	380	2-stage EN	CA	years	stable across immune cell types
encen40 / encen100	2023	DNAm blood/ saliva/buccal	559 / 198	elastic net	CA	years	centenarian-tuned
retroelementagev1/ v2	2024	DNAm blood	1,317 / 1,378	elastic net	CA	years	retroelement-focused
senchronoage / senscultureage / senmortalityage	2026	DNAm	187 / 142 / 91	EN / EN-logistic / EN-Cox	CA / senescence / mortality	years / log-odds / log-hazard	newest human clocks in the registry
pasta / reg	2025	transcriptomics multi-tissue	8,113	ridge-logistic / ridge	transcriptomic age	years	rank-normalised
bitage	2021	transcriptomics *C. elegans*	576	elastic net	bio age	hours	binarised
ocampoatac1/2	2023	ATAC-seq PBMC	228 / 380	elastic net	CA	years	
camillo* (8 marks)	2025	histone ChIP-seq	102–3,739	PCA + EN + ARD	CA	years	
mammalian1/2/3 + blood/skin variants	2023	DNAm mammalian array	335–2,572	elastic net	CA	years	species lookup required
petkovich / stubbs / meer / thompson	2017–18	DNAm mouse	90 / 17,992 / 435 / 582	elastic net	CA	months / days	RRBS or bisulfite-seq
deconvolute* (6 + 12 cell)	2018 / 2022	DNAm EPIC	600 / 240	constrained deconvolution	cell proportions	proportion	
xchrom / ychrom	2021	DNAm	4,047 / 284	PCA	sex chromosome dosage	unitless	

Clock	Year	Layer	Features	Model	Predicts	Unit	Notes
mccartney* (10)	2018	DNAm blood	204–1,109	LASSO	lifestyle/clinical traits	various	smoking, BMI, alcohol, lipids, education, WHR, body fat
knight / bohlin / mayne / lee×3 / epicga	2016–2021	DNAm cord blood/ placenta	62–558	EN / LASSO	gestational age	weeks	
pedbe / wu	2019–2020	DNAm buccal/ blood	94 / 111	EN	CA in children	years	
corticalclock / neusin / gliasin	2020 / 2024	DNAm brain cortex	347 / 672 / 220	EN	CA	years	
ctsliver / hep / hepatoxu	2017–2024	DNAm liver / cfDNA	90 / 70 / 10	LASSO / logistic	CA / HCC	years / unitless	
cvdwesterman / adbahadosingh / prostatecancerkirby / depressionbarbu / downsyndrome	2017–2021	DNAm blood	235 / 4 / 3 / 196 / 652	disease risk models	disease probability or score	probability / log odds / unitless	
dnamfili / dnamic / dnamstress	2022–2025	DNAm blood	20 / 91 / 211	LASSO / EN	frailty / intrinsic capacity / stress	unitless	
replitali / replitalinorm / cellpopage	2022–2024	DNAm cultured cells	87 / 218 / 42	EN	replicative history	population doublings / passages	

7.3 Clocks outside pyaging worth supporting

- **organAging** - 24 organ panels × {chronological, mortality} × {full Olink 3072, feature-reduced Olink 1536, CSF}. Coefficients ship as CSVs in the repo.
- **tAge** - 60 transcriptomic models (§4.18).
- **SpatialAgingClock** - 18 mouse brain cell types (§4.13).
- **scImmuAging** - 5 human PBMC cell types (CD4T, CD8T, MONO, NK, B), glmnet objects with `lambda.min`, applied to pseudocells.
- **SCALE** - >20 mouse tissues, Tabula Muris Senis.
- **CellBiAge** - mouse brain snRNA/scRNA binary classifiers.
- **scAge / scEpiAge** - single-cell methylation.
- **OMICmAge** - 421 epigenetic biomarker proxies (110 protein, 286 metabolite, 25 clinical) distilled into a DNAm-only predictor of EMRAge. Discovery MGB Aging Biobank n = 3,451; validation TruDiagnostic n = 14,213 and Generation Scotland n = 18,672.
- **GOLD BioAge / GOLD ProtAge / GOLD MetAge / Light BioAge** - Gompertz-law clinical, proteomic, metabolomic, and 3-marker variants (Adv Sci 12(32), 2025); benchmarked on NHANES and UK Biobank.
- **StackAge** - UKB proteomics + metabolomics ensemble, n = 30,376.
- **Oh 2023 organ clocks** - SomaScan, ~5,700 people, `organage` Python package.
- **GP-age** - Gaussian process regression on 30 CpGs selected by k-means over 80 age-correlated sites; MedAE 2.1 y, 11,910 blood samples.
- **RNAAgeCalc** - cross-tissue and tissue-specific GTEx transcriptomic ages (R/Bioconductor, Python `racpy`, Shiny).
- **MEAT** - muscle epigenetic age test (Bioconductor).
- **MiAge** - mitotic age; coefficients ship in ComputAge.
- **GlycanAge**, **IMM-AGE**, microbiome clocks.

Of the clocks outside pyaging, FALCONAge now scores MetaboAge and MetaboHealth (NMR metabolomics), LinAge2 and cAge; OMICmAge is catalogued as licensed.

Chapter 23. Genotype and structure above the probe

21. Genetic variation under the probe

Every section above treats a beta value as a measurement of methylation. It is also, in part, a measurement of genotype, and the two are not separable by looking at the number.

21.1 A probe sits at a genomic coordinate

Each probe has a chromosome and a base position, so the probe set can be intersected with any variant catalogue: ClinVar for clinically annotated variants, dbSNP for the general catalogue. The intersection is an ordinary range join, which `GenomicRanges` or `bedtools` will do, and it answers a question that matters before scoring: does this clock depend on probes whose signal a common variant can move.

21.2 Three ways a variant reaches the beta value

A SNP at the interrogated base. Single-base extension reads the variant rather than the methylation state, so the beta becomes a genotype readout. This is the population of probes the published masks remove. FALCONAge ships the Zhou masks (`fa.preprocess.mask_report`, `apply_mask`) but applies none automatically, because every clock was fitted on unmasked data; `mask_report` shows how much of each clock's coefficient mass a mask would remove.

A SNP under the probe body. Hybridisation is weakened, intensity falls, and the beta drifts toward the middle of the range without any obvious signature of failure.

Cross-reactivity. A 50-mer against a three-base genome maps to more than one locus more often than the design intends. Chen and colleagues catalogued these for the 450K, and the affected probes carry a mixture of signals from several places in the genome.

21.3 meQTL: the variant is somewhere else entirely

A methylation quantitative trait locus is a variant associated with methylation at a CpG, usually nearby but sometimes far away. GoDMC and the EPIGEN meQTL resource map these genome-wide. The consequence for a clock is direct: if a clock CpG is under strong meQTL control, part of the between-person variance in that clock is genetic and fixed at conception rather than acquired, and no intervention will move it.

> **i** What this changes about a score
>
> A clock's between-person variance splits into an acquired component, which is what the clock is meant to measure, and a genetic component, which is not. The split is not visible in the score. Two people with the same clock age can differ because one has lived differently and the other inherited a different genotype at a handful of high-weight probes.

21.4 The arrays carry genotype probes on purpose

450K and EPIC include rsID-labelled SNP probes, 65 on the 450K, for sample identity. They confirm that two arrays came from the same person, which is the check that catches a swapped sample. `MethylToSNP`

goes further and infers genotypes from the methylation pattern itself, which is useful and is also a reminder that array data is identifiable.

22. Structure above the single probe

22.1 A clock is a combination score

A clock sums hundreds of weighted features, which makes it the same kind of object as a polygenic risk score, and methylation risk scores for individual traits are built the same way: many published methylation biomarkers are proxies for proteins, metabolites and clinical values rather than clocks.

The shared property matters more than the shared arithmetic. A combination score is robust where a single probe is noisy, and it is opaque in the same measure: a change in the total does not say which component moved. That is the argument for clocks that report their components separately, as SystemsAge does for eleven systems.

22.2 Co-methylation modules

CpGs covary. Weighted correlation network analysis groups them into modules and summarises each by its first principal component, and the module can then be tested as one unit rather than as several thousand correlated tests. This is where the multiple-testing burden of an epigenome-wide scan becomes tractable, and it recovers structure that a per-probe analysis cannot see.

22.3 Allele-specific methylation

The two alleles at a locus can carry different methylation, and the pattern can follow a haplotype block. Array data cannot resolve this because a beta value averages both alleles into one number. Sequencing can, and tools such as MethHaplo combine allele-specific methylation with genotype to identify the regions involved. Imprinted loci are the familiar case and not the only one.

22.4 Interactions between variants are real and small

Genome-wide tests for SNP by SNP effects on methylation find them, and find that they explain little variance. The practical reading is that additive models are not leaving much on the table at this level.

Chapter 24. 23. Building a clock of your own

The catalogue is other people's clocks. This is the arithmetic of fitting one, with the numbers the published clocks actually used.

23.1 The objective

With n samples and p CpGs, $p \gg n$ always. Horvath had 8,000 samples against 21,369 candidate probes; a modern EPIC study has 850,000 columns and a few hundred rows. An unpenalised least-squares fit is not identifiable, because $X^\top X$ is singular whenever $p > n$. Penalisation is what makes the problem have an answer, not a refinement on one.

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - x_i^\top \beta\right)^2 + \lambda \left[\frac{1-\alpha}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right] \right\}$$

The two penalties do different work. The ℓ_1 term has a corner at zero, so its subgradient can hold a coefficient exactly at zero, which is what performs selection. The ℓ_2 term is smooth and shrinks correlated predictors together rather than choosing between them, which matters because neighbouring CpGs are strongly correlated and pure LASSO picks among them close to arbitrarily.

α	Name	Behaviour on correlated CpGs	Used by
1	LASSO	Selects one per correlated group, close to arbitrarily	eFRS, the McCartney EpiScores
0.5	Elastic net	Keeps correlated groups together while still zeroing most coefficients	Horvath 2013
0	Ridge	Shrinks everything, selects nothing	Rarely used alone

Horvath fitted $\alpha = 0.5$ with $\lambda = 0.0226$ chosen by cross-validation, over the 21,369 probes shared by the 27K and 450K platforms that had fewer than ten missing values. That produced the 353 sites.

> ! The selected set is not the biology
>
> Rerun the same fit with a slightly different λ, a different fold split, or a different subset of samples, and a different CpG set comes out with almost identical accuracy. The correlated groups are interchangeable to the optimiser. This is one mechanism behind the discrepancies listed in section 15.2, and it is why two faithful implementations of the same clock can disagree on membership.

23.2 Choosing the penalty honestly

λ comes from k-fold cross-validation: fit on $k-1$ folds, measure on the held-out one, take the λ minimising mean cross-validated error, or the largest λ within one standard error of it for a sparser model.

Any step that looked at the outcome must sit inside the loop. Selecting CpGs by their correlation with age on the full dataset and then cross-validating the fit on those CpGs leaks the held-out fold into the selection, and the reported error is optimistic by an amount that cannot be recovered afterwards.

23.3 The age transform

Methylation changes quickly in childhood and slowly afterwards, so a model linear in age fits adults and misses the first two decades. Horvath regresses a transformed age and inverts afterwards, with $A = 20$. Up to A,

$$F(\text{age}) = \log(\text{age} + 1) - \log(A + 1),$$

and above it,

$$F(\text{age}) = \frac{\text{age} - A}{A + 1}.$$

Both branches give 0 at age $= A$ and both have derivative $1/(A + 1)$ there, so F is continuous and differentiable at the join. The model predicts F; the reported age is F^{-1} of the linear predictor. Applying the inverse twice, or omitting it, is invisible in an adult cohort because the branches nearly agree there, and badly wrong in a paediatric one.

23.4 Feature selection is not settled

Elastic net is the default rather than the optimum. Benchmarked on the Hannum blood data (GSE40279, 656 samples, 473,035 CpGs, 10-fold cross-validation, 80/20 split):

Method	CpGs kept	R^2	Median error
SelectKBest (2,000) then Boruta	35	0.873	3.08 years
Boruta alone, or KBest alone	varies	0.861 to 0.862	
Recursive elimination, 1% per iteration	varies	≥ 0.81	
Genetic algorithm, most threshold methods	varies	0.77 to 0.81	
Neural network on 65 features	65	0.76	found 61 CpGs no other method did
Elastic net, no prior selection	276	below five of the above	
The five most-selected CpGs overall	5	0.83	3.79 years
cg16867657 (ELOVL2) alone	1	0.73	

Two rows there are worth sitting with. A single CpG in ELOVL2 reaches $R^2 = 0.73$, so most of what a 353-CpG clock knows about chronological age is available from a handful of sites. And an unfiltered elastic net over all 473,035 probes was beaten by five selection strategies, which is the opposite of the usual intuition: past a point the extra probes contribute noise rather than signal.

23.5 Two metrics that disagree, and both are needed

$$\mathrm{MedAE} = \mathrm{median}_i |\hat{y}_i - y_i|, \qquad r = \mathrm{corr}(\hat{y}, y)$$

Correlation is inflated by a wide age range and says nothing about bias. Median absolute error is in years and says nothing about ordering. A clock at r = 0.95 on a cohort spanning 20 to 90 may have an MAE of six years; the same clock on a cohort spanning 55 to 65 may have r near zero and the same MAE. Neither number is wrong and neither alone is informative.

Expect the fitted slope of $\hat{y}$ on y to fall below one. Penalisation shrinks coefficients toward zero, which compresses predictions toward the training mean, so the old read young and the young read old. That is regression to the mean, a property of the estimator rather than a defect of the cohort, and it leaves the residual correlated with age unless the slope is corrected. It is why age acceleration is a residual on chronological age.

23.6 The step that gets skipped

A model fitted to chronological age will correlate with chronological age. Reporting that as validation is circular. The claim worth making is about the residual: whether age acceleration associates with disease, mortality or an intervention, on a cohort that was not used to fit the model. Everything in section 11 applies to a clock you fitted exactly as it applies to a published one.

23.7 What to publish

Feature list, weights, intercept, the preprocessing chain and the postprocess, together and in the order they apply. A clock published without its postprocess is the commonest source of the disagreements in section 15, because the next implementer has to guess whether an anti-log was already applied, and a plausible wrong guess produces plausible wrong ages.

23.8 Fitting one here

`fa.fit_clock(data, target="age")` fits an elastic net to one cohort and reports only predictions made out of fold: the cohort is split into outer folds, lambda is chosen by cross-validation inside each training part, and each held-out fold is predicted by a model that saw neither its samples nor their role in choosing lambda. The estimator is glmnet's for a gaussian outcome, including its internal scaling of the outcome, its lambda grid and `cv.glmnet`'s `lambda.1se` or `lambda.min`. It reproduces glmnet 5.1′s path to within 1e-6 and chooses exactly the lambdas that `cv.glmnet`'s rules give on glmnet's own fold fits; `cv.glmnet`'s returned lambdas can differ slightly, because its internal fold fits differ from direct `glmnet` calls by about 1e-4 in MSE. A missing value is filled with its feature's mean in the training part it falls in. The model fitted on everyone is returned for new samples, and its accuracy is the out-of-fold one, not its fit. The result is a model of this cohort and is labelled as one.

Chapter 25. Failure modes and the frontier

15.2 Where the paper and the coefficients disagree

22 of the 178 catalogued clocks carry a `known_discrepancies` note: a place where the paper, its coefficients and the implementations in circulation disagree. Each is raised as a warning at score time.

Clock	The disagreement
`bocklandt`	The paper describes a 2-CpG model; the coefficient set in circulation carries 1. Which probe was dropped is not documented upstream, so this is not the published model.
`bohlin`	The paper reports 96 CpGs; the coefficient set in circulation carries 251. Scores here will match other packages and not the figures printed in the paper.
`cvdwesterman`	The paper reports 1,305 CpGs; the coefficient set in circulation carries 235. Expect disagreement with any value quoted from the paper.
`mccartneyalcohol`, `mccartneybmi`, `mccartneybodyfat`, `mccartney education`, `mccartneyhdlcholest erol`, `mccartneyldlcholesterol`, `mccartneysmoking`, `mccartney totalcholesterol`, `mccartney totalhdlratio`, `mccartneywhr`	The paper defines this predictor as a weighted sum of beta values and applies no link function; the ROC analysis it reports dichotomises the phenotype, not the score. Packages that pass the sum through a sigmoid report a different number. The scale is relative_score, whose legal operations are correlate and rank, and a sigmoid is monotone, so no permitted result changes either way.
`phenoage`	Two constants circulate in the divisor of the Gompertz inversion: 0.090165, printed in Levine 2018 Supplement 1, and 0.09165, printed in Liu et al. 2018 (PLoS Med). FALCONAge uses 0.090165, the value BioAge corrected to in April 2026. Code still using 0.09165 differs by several years on the same input.
`phenoage`	Levine 2018 Table 1 prints alkaline phosphatase as 0.0019; Liu et al. 2018 print 0.00188, and BioAge carries the fit at full precision (0.001868778, every weight rounding to Table 1). FALCONAge uses Table 1 as printed, a mean 0.074 years above the full-precision fit on NHANES III; phenoage(df, coefficients='bioage') uses BioAge's.
`phenoage`	BioAge enters CRP as log(1 + CRP), its lncrp, where the paper uses log(CRP). On NHANES III that puts BioAge's phenoage0 a mean 1.50 years above the paper's formula (SD 0.45, more at low CRP; n = 8,924). FALCONAge follows the paper; phenoage(df, crp_transform='log1p', coefficients='bioage') reproduces phenoage0 exactly.
`senchronoage`, `senculture age`, `senmortalityage`	The upstream README's feature count disagrees with the coefficient object it ships. The registry follows the coefficient file, because that is what scores.
`weidner`	The third site is a substitution. The paper's gamma is not cg17861230 but an unnamed CpG upstream of it, measured by pyrosequencing and better correlated with age; no array id was ever published for it. Every array implementation of this clock, including this one, puts cg17861230 in its place, and gamma carries 164.7 of the model against 26.4 and 23.7 for the two genuine probes. Treat the result as the published model evaluated on a near neighbour.
`yingadaptage`, `yingdamage`	Scored across three healthy cohorts in the FALCONAge corpus, this clock's median offset against chronological age swings 162 years between datasets while Horvath2013's swings 15; DamAge and AdaptAge move as near mirror images (r = −0.975) and the swing in their sum is only 33, so the cause is one dataset-level shift amplified by their opposing intercepts of +543.43 and

Clock	The disagreement
	−511.97 rather than a fault in either. The slope against age is sound (0.967 pooled for DamAge). The scale is therefore age_years_relative: the residual and group differences are defined, predicted-minus-chronological is not.
yingadaptage	The paper reports 1,000 CpGs; the coefficient set carries 999. The same off-by-one appears in all three Ying clocks, consistent with the paper's total counting an intercept row as a feature.
yingcausage	The paper reports 586 CpGs; the coefficient set carries 585. The same off-by-one appears in all three Ying clocks, consistent with the paper's total counting an intercept row as a feature.
yingdamage	The paper reports 1,090 CpGs; the coefficient set carries 1,089. The same off-by-one appears in all three Ying clocks, consistent with the paper's total counting an intercept row as a feature.
zhangmortality	The paper defines this as an integer count of markers past quartile thresholds; the implementations in circulation compute a continuous weighted sum. They are different statistics, and the continuous form has no published cut-points. FALCONAge scores the continuous sum of the LASSO Cox weights, a log hazard (scale_type mortality_log_hazard), not the count.

The table is generated from the registry by `docs/build_catalogue.py`: `reg.get(id).known_discrepancies` returns the full text, and `res.manifest["warnings"]` carries whichever of them fired on your run. The Ying clocks additionally carry a measured cohort-offset note, which is why they sit on `age_years_relative` rather than `age_years`, §19.3.1.

16. Failure modes

Units. PhenoAge in SI vs conventional units gives different answers with the same biomarker draw. Albumin g/L vs g/dL is a 10× error; creatinine μmol/L vs mg/dL is ~88×; CRP mg/L vs mg/dL is 10× inside a log. FALCONAge requires declared units when clinical data are read and refuses to guess them.

The 0.090165 constant. Some published code uses 0.09165. BioAge fixed this in April 2026. Check any inherited implementation.

EPIC v2 suffixes. Aggregate before scoring or every clock sees zero features; `prepare()` does it by default.

Dropping DunedinPACE background probes. The 19,827 non-scoring probes define the quantile mapping. Subsetting to the 173 scoring CpGs produces a plausible-looking wrong number.

Zero-filling missing methylation features. $\beta = 0$ means fully unmethylated. Use reference values. pyaging warns. FALCONAge fills an absent feature from the clock's reference value (the present features' mean, with a warning, where it has none) and refuses a clock below 80% of its features or of its coefficient mass (`min_coverage`).

Mixing generations in a composite. Averaging Horvath (chronological) with GrimAge (mortality) produces a number with no interpretation. Composite only within generation, after z-scoring.

Subtracting CA from a rate clock. DunedinPACE is not an age, and FALCONAge refuses the subtraction (`IllegalOperationError`).

Multiplying a mortality clock by species maximum lifespan. tAge guards this with a registry flag precisely because it is an easy mistake.

Per-sample vs per-feature scaling. Zhang z-scores across probes *within* a sample; AltumAge scales each feature by the training median and interquartile range *across* samples; tAge's `scale_eset` is column-wise, i.e. per sample. Getting the axis wrong is silent.

Regression dilution. All penalised clocks compress toward the training mean, so BA is biased low in the old and high in the young, and the naive `BA − CA` correlates with CA. Report the RAA residual too.

Cell composition confounding. Second-generation and single-tissue clocks partly read cell proportion. Run deconvolution alongside and either adjust or report both.

Cell type dominates age in single cells. scEpiAge found cell-type effect > age effect. Always model within cell type.

Batch and array position. Chip position and DNA extraction protocol move clock values by years. PC clocks and SystemsAge are the mitigations; randomising sample layout is the prevention.

Technical ICC is not biological ICC. A clock can be perfectly reproducible on a re-run of the same DNA and still be unstable across two blood draws a week apart.

Age-range extrapolation. Models trained on adults 20–70 do not extrapolate; gestational and paediatric clocks are separate for a reason. FALCONAge does not check the registry `population` field at scoring time; check it before scoring outside a clock's training age range.

`torch.load(weights_only=False)`. Arbitrary code execution from a downloaded file. FALCONAge loads network weights from safetensors only.

Sex encoding. GrimAge takes `Female` as 1/0; ComputAge maps unknown sex to 0.5, which is a silent half-male half-female fudge. Make unknown sex an explicit user decision.

Intercept placement. Some packages put the clock intercept in the transform lambda, others in the linear layer bias. Normalise at import or double-count.

FP32 on PC clocks. The 78,464-dimensional centring and rotation is ill-conditioned; validate before defaulting to FP32.

Sample size for HD. The Mahalanobis covariance needs more reference samples than biomarkers, well-conditioned. BioAge warns on single-row inputs. FALCONAge refuses a reference with fewer than five complete rows more than it has markers, and inverts the covariance with a pseudo-inverse, which tolerates near-collinear markers; it does not report the condition number.

17. The frontier, and what FALCONAge does not yet compute

Reviewed to August 2026. This section exists because a tool that silently omits half a field reads as though the field ends where it does. Each item states the science, the blocker, and whether the blocker is technical or legal.

17.1 Array normalisation, and what it corrects

FALCONAge reads raw IDATs (`fa.idat_to_betas`, `fa.read_idat_dir`) and runs the chain below; each step exists to correct a specific instrument artefact rather than to tidy data.

An Infinium array carries two probe chemistries. **Type I** uses two beads per CpG read in one colour channel, which leaves the other channel at that address carrying no designed signal – the **out-of-band** readings. **Type II** uses one bead, methylated in green and unmethylated in red, and makes up 72% of 450K probes and 84% of EPIC v1 probes. The two designs have different beta distributions, so a raw array is bimodal for reasons that are chemistry rather than biology.

Step	What it removes	How
noob	background fluorescence, cross-channel bleed	normal-exponential deconvolution using the out-of-band Type I readings as the background distribution – the background estimate comes free from probes never meant to fluoresce in that channel
dye bias	red/green intensity difference	quantile-matching the channels on Type I probes measured in both
pOOBAH	measurements indistinguishable from background	detection p-value from the empirical out-of-band distribution; failures are masked to NA, not deleted
BMIQ	the Type I/Type II design difference	three-state beta mixture fitted to Type II, Type I quantile-mapped onto it within each state

Step	What it removes	How
probe masks	probes untrustworthy anywhere	cross-reactive probes, probes overlapping common SNPs at the interrogated base, sex chromosomes

The order is load-bearing (pOOBAH on the raw out-of-band signal, then noob, then dye-bias correction, then BMIQ), and it matters because **clock coefficients were fitted on the output of a particular chain**. Applying Horvath's weights to raw betas is not a smaller version of the right answer; it is the right weights on the wrong scale.

`fa.preprocess.bmiq` and `fa.preprocess.horvath_normalise` provide the last step; normalising with sesame or minfi and reading the beta matrix remains supported.

17.2 Cross-platform harmonisation

Probe content changed across 27K, 450K, EPIC v1 and EPIC v2. EPIC v2 both dropped probes earlier clocks depend on and renamed others with replicate suffixes. FALCONAge handles the suffixes and *measures* the damage from the drops, by count and by coefficient mass, and fills an absent probe from the clock's reference value, which recovers most of the shift for most clocks. It does not impute a dropped probe from its neighbours.

`mLiftOver` is the published answer, and it is four things rather than an ID lookup: probe-ID conversion, replicate management, optional imputation with a confidence filter, and – the part that cannot be reconstructed – a mapping table carrying empirical evidence of whether each probe's cross-platform mapping is reliable. Those tables are measured on paired samples and distributed through Bioconductor's ExperimentHub, so this is blocked on data rather than effort.

Principal-component clocks absorb probe loss because a missing probe perturbs a component instead of deleting a term, which is why EPIC v2 disrupts the first-generation clocks and barely moves the PC ones.

17.3 Proteomic organ clocks

If a protein is enriched at least fourfold in one organ relative to all others, its plasma level proxies that organ's state – so one blood draw supports an age estimate per organ. Ten organ clocks plus an organismal one were trained in UK Biobank (n = 43,616) and validated in China (n = 3,977) and the USA (n = 800) at cross-cohort r = 0.98 and 0.93, with brain age most strongly linked to mortality.

Two properties matter for anyone implementing this. **Olink NPX is log2 and relative already**, so re-logging it is an error and comparing across panel versions without bridging is another. And these clocks are **z-scored against a training cohort, not against your own samples** – using your cohort's mean silently recentres everyone to zero acceleration, the same class of mistake as filling an absent probe with zero. About half the ageing predictors in Olink-based clocks are absent from SomaScan-based ones, so the platforms are not interchangeable.

17.4 Transcriptomic clocks

Six clocks from two normalisations (YuGene-diff, Scaling-diff) and three targets (chronological age, normalised age, mortality), trained as multi-species multi-tissue elastic nets over more than 11,000 transcriptomes.

The chain is: filter low-count genes, map to one-to-one orthologues only, RLE-normalise, log10, per-sample z-score or YuGene, then **median-centre across all samples within each dataset**.

That last step is why these are not simply another modality. A transcriptomic clock is **undefined for a single sample in isolation** – the centring is a property of the cohort. An implementation has to refuse a one-sample run rather than warn about it.

Accuracy runs 5-7 years MAE against methylation's 3-5. The honest reading is that they answer a different question, not that they are worse.

17.5 Methylation foundation models

CpGPT (pretrained on 2,000+ datasets, 150,000+ samples) and **MethylGPT** (226,555 profiles from 5,281 datasets) are transformers over CpG sites taking sequence, position and epigenetic context, so they can reason about a site they have not seen on an array.

The interesting capability is not another age predictor. It is **zero-shot imputation and cross-array conversion**, which attacks EPIC v2 probe loss, platform harmonisation and single-cell sparsity at once. CpGPT placed second for age and first for methylation-based mortality in the Biomarkers of Aging Challenge, and its attention weights give per-sample interpretability – which sites drove *this* prediction.

This would also be the first model here where a GPU is the right default. Every bundled clock is either linear in its inputs or, for AltumAge, a small network, and the largest reads 319,607 CpGs; GPU support measures CUDA as *slower* for that reason. A transformer inverts the arithmetic.

17.6 Single-cell

Single-cell methylation is binary and sparse – a site is covered or not, methylated or not – so a linear clock cannot be applied directly. FALCONAge implements **scAge**'s per-cell maximum likelihood over the covered subset (`fa.models.single_cell.scage`), closer in spirit to conditioning on coverage than to taking a dot product, and tests the spread of per-cell ages against estimation error (`fa.mosaic`); what remains outside it is the transcriptomic side. On the transcriptomic side **scAgeClock** reaches about 2 years MAE for the best cell types using gated multi-head attention, and single-cell profiles are pooled to pseudobulk before a bulk transcriptomic clock is applied.

17.7 Reliability, and why it governs everything above

Technical and biological reliability come apart. Nearly every clock is technically excellent – the same sample assayed twice agrees closely – while biological ICCs, the same person sampled again days later, are substantially lower, with no clear relationship between the two. **GrimAge2 and DunedinPACE are among the most biologically fragile**, and they are the two most often used to claim an intervention worked.

The registry carries both figures where a source exists and `None` where none does, because an unpublished ICC is not a good ICC. The Biomarkers of Aging Consortium's own stated bottleneck is not the models but the absence of standard procedures for collecting specimens and applying biomarkers.

Chapter 26. 18. Datasets, registries, reference resources

Benchmark and training data - `computage/computage_bench` on Hugging Face - 10,404 benchmark samples / 65 studies and 7,419 training samples / 46 studies, parquet + TSV metadata. - `lucascamillomd/pyaging-data` on Hugging Face, clock weights, reference vectors, Ensembl gene annotation, example datasets per modality. - GEO series listed in ComputAge BenchBSB_table.csv. - GSE54848 - OSKM reprogramming timecourse used in the pyaging figure. - NHANES III and NHANES IV - shipped as `.rda` in BioAge with a codebook. - UK Biobank, Framingham Heart Study, Generation Scotland, MESA, Health and Retirement Study, Dunedin Study, Tabula Muris Senis, GTEx.

Gene and biomarker resources - Open Genes (2,402 genes), GenAge, AnAge (the mammalian clock lifespan lookup), CellAge, CSGene, AgeFactDB, Aging Atlas, GO aging terms, Digital Ageing Atlas, AGEMAP.

Reference matrices - `sesame_450k_median.csv` - NA-fill medians. - `450K_reinius_12_reference.csv`, `EPIC_salas_18_reference.csv` - deconvolution signatures. - `biolearn_averages_450k.csv` - alternative fill vector. - `DunedinPACE_Gold_Means.csv` - the 20,000-probe gold standard.

Model registries - pyaging `clock_metadata.json` (177 clocks at e7267e9) + `evidence_ledger.jsonl` (930 kB of per-field provenance) + `controlled_vocabulary.json` + `validate_metadata.py`. - tAge `clocks_metadata.csv` (60 models) + Zenodo record 18763485. - ComputAge `model_definitions` + `library.yaml` (GEO dataset parsers).

Chapter 27. References

Citations across this site are numbers in square brackets and every one of them points here. The numbering is global rather than per page, so a source keeps its number wherever it is cited, and the same list is bound into the PDF as its own chapter.

Where a source is available at a stable identifier the entry links to it. Some of these are preprints and are cited as preprints on purpose: the field moves faster than journal review, and a claim resting on one is worth knowing about.

Prior software this package builds on

1. de Lima Camillo LP. pyaging: a Python-based compendium of GPU-optimized aging clocks. *Bioinformatics* 40(4):btae200 (2024). https://doi.org/10.1093/bioinformatics/btae200

2. Yu D, Li M, Linghu G, Hu Y, Hajdarovic KH, Wang A, Singh R, Webb AE. CellBiAge: Improved single-cell age classification using data binarization. *Cell Reports* 42(12):113500 (2023). - PMC10791072. https://doi.org/10.1016/j.celrep.2023.113500

Foundational clocks

3. Bocklandt S et al. Epigenetic predictor of age. *PLoS ONE* 6(6):e14821 (2011). https://doi.org/10.1371/journal.pone.0014821

4. Hannum G et al. Genome-wide methylation profiles reveal quantitative views of human aging rates. *Mol Cell* 49(2):359–367 (2013). https://doi.org/10.1016/j.molcel.2012.10.016

5. Horvath S. DNA methylation age of human tissues and cell types. *Genome Biol* 14:R115 (2013). https://doi.org/10.1186/gb-2013-14-10-r115

6. Weidner CI et al. Aging of blood can be tracked by DNA methylation changes at just three CpG sites. *Genome Biol* 15:R24 (2014). https://doi.org/10.1186/gb-2014-15-2-r24

7. Yang Z et al. Correlation of an epigenetic mitotic clock with cancer risk. *Genome Biol* 17:205 (2016). - epiTOC https://doi.org/10.1186/s13059-016-1064-3

8. Levine ME et al. An epigenetic biomarker of aging for lifespan and healthspan. *Aging* 10(4):573–591 (2018). https://doi.org/10.18632/aging.101414

9. Liu Z, Kuo P-L, Horvath S, Crimmins E, Ferrucci L, Levine M. A new aging measure captures morbidity and mortality risk across diverse subpopulations from NHANES IV. *PLoS Med* 15(12):e1002718 (2018). https://doi.org/10.1371/journal.pmed.1002760

10. Horvath S et al. Epigenetic clock for skin and blood cells. *Aging* 10(7):1758–1775 (2018). https://doi.org/10.18632/aging.101508

11. Lu AT et al. DNA methylation GrimAge strongly predicts lifespan and healthspan. *Aging* 11(2):303–327 (2019). https://doi.org/10.18632/aging.101684

12. Lu AT et al. DNA methylation GrimAge version 2. *Aging* 14(23):9484–9549 (2022). https://doi.org/10.18632/aging.204434

13. Zhang Q et al. Improved precision of epigenetic clock estimates across tissues and its implication for biological ageing. *Genome Med* 11:54 (2019). https://doi.org/10.1186/s13073-019-0667-1

14. Belsky DW et al. Quantification of the pace of biological aging in humans through a blood test, the DunedinPoAm DNA methylation algorithm. *eLife* 9:e54870 (2020). https://doi.org/10.7554/elife.54870

15. Belsky DW et al. DunedinPACE, a DNA methylation biomarker of the pace of aging. *eLife* 11:e73420 (2022). https://doi.org/10.7554/elife.73420

16. Higgins-Chen AT et al. A computational solution for bolstering reliability of epigenetic clocks. *Nat Aging* 2:644–661 (2022). - PC clocks https://doi.org/10.1038/s43587-022-00248-2

17. de Lima Camillo LP, Lapierre LR, Singh R. A pan-tissue DNA-methylation epigenetic clock based on deep learning. *npj Aging* 8:4 (2022). - AltumAge https://doi.org/10.1038/s41514-022-00085-y

18. Lu AT et al. Universal DNA methylation age across mammalian tissues. *Nat Aging* 3:1144–1166 (2023). https://doi.org/10.1038/s43587-023-00462-6

19. Ying K et al. Causality-enriched epigenetic age uncouples damage and adaptation. *Nat Aging* 4:231–246 (2024). - CausAge, DamAge, AdaptAge https://doi.org/10.1038/s43587-023-00557-0

20. Teschendorff AE. A comparison of epigenetic mitotic-like clocks for cancer risk prediction. *Genome Med* 12:56 (2020). The paper that introduces epiTOC2 and HypoClock, which is what this entry is cited for. https://doi.org/10.1186/s13073-020-00752-3

21. Higgins-Chen AT, Levine ME et al. Systems Age: a single blood methylation test to quantify aging heterogeneity across 11 physiological systems. *Nat Aging* (2025). https://doi.org/10.1038/s43587-025-00958-3

Non-methylation clocks

22. Meyer DH, Schumacher B. BiT age: a transcriptome-based aging clock near the theoretical limit of accuracy. *Aging Cell* 20:e13320 (2021). https://doi.org/10.1111/acel.13320

23. Oh HS-H et al. Organ aging signatures in the plasma proteome track health and disease. *Nature* 624:164–172 (2023). https://doi.org/10.1038/s41586-023-06802-1

24. Goeminne LJE et al. Plasma protein-based organ-specific aging and mortality models unveil diseases as accelerated aging of organismal systems. *Cell Metab* 37(1):205–222.e6 (2025). https://doi.org/10.1016/j.cmet.2024.10.005

25. Tyshkovskiy A, Kholdina D, Davitadze M et al. Universal transcriptomic hallmarks of mammalian ageing and mortality. *Nature* 654(8117):173–188 (2026). https://doi.org/10.1038/s41586-026-10542-3

26. Sun ED, Zhou OY, Hauptschein M, Rappoport N, Xu L, Navarro Negredo P, Ling L, Rando TA, Zou J, Brunet A. Spatial transcriptomic clocks reveal cell proximity effects in brain ageing. *Nature* 638:160–171 (2025). https://doi.org/10.1038/s41586-024-08334-8

27. Li W et al. Single-cell immune aging clocks reveal inter-individual heterogeneity during infection and vaccination. *Nat Aging* (2025). https://doi.org/10.1038/s43587-025-00819-z

28. Trapp A, Kerepesi C, Gladyshev VN. Profiling epigenetic age in single cells. *Nat Aging* 1:1189–1201 (2021). - scAge https://doi.org/10.1038/s43587-021-00134-3

29. Bonder MJ et al. scEpiAge: an age predictor highlighting single-cell ageing heterogeneity in mouse blood. *Nat Commun* 15:7567 (2024). https://doi.org/10.1038/s41467-024-51833-5

30. Ren X, Kuan PF. RNAAgeCalc: a multi-tissue transcriptional age calculator. *PLoS ONE* 15(8):e0237006 (2020). https://doi.org/10.1371/journal.pone.0237006

31. Krištić J et al. Glycans are a novel biomarker of chronological and biological ages. *J Gerontol A* 69(7):779–789 (2014). https://doi.org/10.1093/gerona/glt202

32. Putin E et al. Deep biomarkers of human aging. *Aging* 8(5):1021–1033 (2016). https://doi.org/10.18632/aging.100968

33. Hao M et al. Gompertz law-based biological age (GOLD BioAge). *Adv Sci* 12(32) (2025). https://doi.org/10.1002/advs.202501765

34. Chen Q, Dwaraka VB, Carreras-Gallo N et al. OMICmAge quantifies biological age by integrating multi-omics with electronic medical records. *Nat Aging* (2026). https://doi.org/10.1038/s43587-026-01073-7

Methods, tooling, benchmarking

35. Kwon D, Belsky DW. A toolkit for quantification of biological age from blood chemistry and organ function test data: BioAge. *GeroScience* 43:2795–2808 (2021). https://doi.org/10.1007/s11357-021-00480-5

36. Kriukov D, Efimov E, Kuzmina EA, Khrameeva EE, Dylov DV. ComputAgeBench: epigenetic aging clocks benchmark. bioRxiv 2024.06.06.597715; KDD '25. https://doi.org/10.1101/2024.06.06.597715

37. Ying K et al. Biolearn, an open-source library for biomarkers of aging / A unified framework for systematic curation and evaluation of aging biomarkers. *Nat Aging* (2025). https://doi.org/10.1038/s43587-025-00987-y

38. TranslAGE: a comprehensive platform for systematic validation of epigenetic aging biomarkers. bioRxiv 2025.10.16.682960. https://doi.org/10.1101/2025.10.16.682960

39. Biological versus technical reliability of epigenetic clocks and implications for disease prognosis and intervention response. bioRxiv 2025.10.13.682176. https://doi.org/10.1101/2025.10.13.682176

40. Teschendorff AE. Epigenetic ageing clocks: statistical methods and emerging computational challenges. *Nat Rev Genet* 26 (2025). https://doi.org/10.1038/s41576-024-00807-w

41. Zoller JA, Horvath S. MammalMethylClock R package. bioRxiv 2023.09.06.556506. https://doi.org/10.1101/2023.09.06.556506

42. Thrush KL, Higgins-Chen AT et al. R methylCIPHER: a methylation clock investigational package. bioRxiv 2022.07.13.499978. https://doi.org/10.1101/2022.07.13.499978

43. de Lima Camillo LP, Horvath S, Wang B et al. CpGPT: a foundation model for DNA methylation. bioRxiv 2024.10.24.619766. https://doi.org/10.1101/2024.10.24.619766

44. Varshavsky M et al. Accurate age prediction from blood using a small set of DNA methylation sites and a cohort-based machine learning algorithm. *Cell Rep Methods* 3(9):100567 (2023). - GP-age https://doi.org/10.1016/j.crmeth.2023.100567

45. Liu Z et al. Underlying features of epigenetic aging clocks in vivo and in vitro. *Aging Cell* 19(10):e13229 (2020). https://doi.org/10.1111/acel.13229

46. Porter HL et al. Many chronological aging clocks can be found throughout the epigenome. *Aging Cell* 20:e13492 (2021). https://doi.org/10.1111/acel.13492

47. Nelson PG, Promislow DEL, Masel J. Biomarkers for aging identified in cross-sectional studies tend to be non-causative. *J Gerontol A* 75(3):466–472 (2020). https://doi.org/10.1093/gerona/glz174

48. Ying K et al. ClockBase: a comprehensive platform for biological age profiling in human and mouse. bioRxiv 2023.02.28.530532. https://doi.org/10.1101/2023.02.28.530532

49. Vasileiou et al. Multi-omics signatures of organ clocks in biological aging and disease. *Aging Cell* (2026). https://doi.org/10.1111/acel.70518

50. Liu et al. The evolving landscape of clinical aging clocks: from epigenetic to multi-omics integration. *Aging Cell* (2026). https://doi.org/10.1111/acel.70579

51. Turning back time: a comprehensive list of interventions that decrease next-generation epigenetic aging clocks in humans. *Front Genet* (2026). https://doi.org/10.3389/fgene.2026.1836446

52. Waziry R et al. Effect of long-term caloric restriction on DNA methylation measures of biological aging in healthy adults from the CALERIE trial. *Nat Aging* 3:248–257 (2023). https://doi.org/10.1038/s43587-022-00357-y

53. DeepStrataAge: an interpretable deep-learning clock. *npj Aging* (2026). https://doi.org/10.1038/s41514-026-00358-w

54. GraphAge: unleashing the power of graph neural networks to decode epigenetic aging. *PNAS Nexus* 4(6):pgaf177 (2025). https://doi.org/10.1093/pnasnexus/pgaf294

Biology of aging

55. López-Otín C, Blasco MA, Partridge L, Serrano M, Kroemer G. The hallmarks of aging. *Cell* 153(6):1194–1217 (2013). https://doi.org/10.1016/j.cell.2013.05.039

56. López-Otín C, Blasco MA, Partridge L, Serrano M, Kroemer G. Hallmarks of aging: an expanding universe. *Cell* 186(2):243–278 (2023). https://doi.org/10.1016/j.cell.2022.11.001

57. Vijg J, Dong X. Pathogenic mechanisms of somatic mutation and genome mosaicism in aging. *Cell* 182(1):12–23 (2020). https://doi.org/10.1016/j.cell.2020.06.024

58. Cagan A et al. Somatic mutation rates scale with lifespan across mammals. *Nature* 604:517–524 (2022). https://doi.org/10.1038/s41586-022-04618-z

59. Horvath S, Raj K. DNA methylation-based biomarkers and the epigenetic clock theory of ageing. *Nat Rev Genet* 19:371–384 (2018). https://doi.org/10.1038/s41576-018-0004-3

60. Field AE et al. DNA methylation clocks in aging: categories, causes, and consequences. *Mol Cell* 71(6):882–895 (2018). https://doi.org/10.1016/j.molcel.2018.08.008

61. Moqri M et al. Biomarkers of aging for the identification and evaluation of longevity interventions. *Cell* 186(18):3758–3775 (2023). https://doi.org/10.1016/j.cell.2023.08.003

Sources traced while implementing the clocks they describe

62. Shireby GL, Davies JP, Francis PT et al. Recalibrating the epigenetic clock: implications for assessing biological age in the human cortex. *Brain* 143(12):3763-3775 (2020). Supplementary Table 3 carries the 347 weights the cortical clock uses here. https://doi.org/10.1093/brain/awaa334

63. McCartney DL, Hillary RF, Stevenson AJ et al. Epigenetic prediction of complex traits and death. *Genome Biology* 19:136 (2018). Additional file 1 carries the ten EpiScore weight sets. https://doi.org/10.1186/s13059-018-1514-1

64. Horvath S. Putting epigenetic aging clocks on trial. *Nature Medicine* (2026). Asks whether a methylation clock behaves as a surrogate endpoint should, which is the question a score has to survive before it is used to judge an intervention. https://doi.org/10.1038/s41591-026-04524-1

Mechanism, limits and translation

65. Tong H, Dwaraka VB, Chen Q et al. Quantifying the stochastic component of epigenetic aging. *Nature Aging* 4(6):886-901 (2024). Across 22,770 blood samples from 25 cohorts, 66 to 75 per cent of Horvath's accuracy is reproducible by drift alone, rising to about 90 per cent for Zhang and falling to about 63 per cent for PhenoAge. https://doi.org/10.1038/s43587-024-00600-8

66. Tarkhov AE, Denisov KA, Fedichev PO. Aging clocks based on accumulating stochastic variation. *Nature Aging* 4(6):871-885 (2024). https://doi.org/10.1038/s43587-024-00619-x

67. Meyer DH, Schumacher B. Epigenetic drift underlies epigenetic clock signals, but displays distinct responses to lifespan interventions, development, and cellular dedifferentiation. *Aging* 16(2) (2024). https://doi.org/10.18632/aging.205503

68. From population science to the clinic? Limits of epigenetic clocks as personal biomarkers. *Epigenomics* 17:1447-1461 (2025). PMC12714307. The source of the cross-tissue 20 to 30 year error figure and the replicate spread this package cites at score time. https://doi.org/10.1080/17501911.2025.2603880

69. Fahy GM et al. Reversal of epigenetic aging and immunosenescent trends in humans. *Aging Cell* 18(6):e13028 (2019). The TRIIM trial: nine men, no control arm. https://doi.org/10.1111/acel.13028

Beyond the clock score: disorder, repertoire and coefficient mass

70. Mei X, Blanchard J, Luellen C, Conboy MJ, Conboy IM. Fail-tests of DNA methylation clocks, and development of a noise barometer for measuring epigenetic pressure of aging and disease. *Aging* 15:8552-8575 (2023). The barometer implemented as `fa.noise_barometer`: a summed per-site standard deviation over cytosines whose variance rises with age. https://doi.org/10.18632/aging.205046

71. Nachun D, Lu AT, Bick AG, Natarajan P, Weinstock J, Szeto MD et al. Clonal hematopoiesis associated with epigenetic aging and clinical outcomes. *Aging Cell* 20:e13366 (2021). The myeloid clonality result: 1.31 years of acceleration on GrimAge to 3.08 on extrinsic EEAA, with DNMT3A and TET2 producing opposing genome-wide patterns. https://doi.org/10.1111/acel.13366

72. Salas LA, Zhang Z, Koestler DC, Butler RA, Hansen HM, Molinaro AM, Wiencke JK, Kelsey KT, Christensen BC. Enhanced cell deconvolution of peripheral blood using DNA methylation for high-resolution immune profiling. *Nature Communications* 13:761 (2022). The twelve-subset EPIC IDOL-Ext reference that resolves naive and memory CD4T and CD8T. https://doi.org/10.1038/s41467-021-27864-7

73. Tomusiak A, Floro A, Tiwari R, Riley R, Matsui H, Andrews N, Kasler H, Verdin E. Development of an epigenetic clock resistant to changes in immune cell composition. *Communications Biology* 7:934 (2024). IntrinClock. Composition-resistant is not clonality-resistant, and the two invariances are separate claims. https://doi.org/10.1038/s42003-024-06609-4

74. Zhang Z, Reynolds SR, Stolrow HG, Chen JQ, Christensen BC, Salas LA. Deciphering the role of immune cell composition in epigenetic age acceleration: insights from cell-type deconvolution applied to human blood epigenetic clocks. *Aging Cell* 23:e14071 (2024). Nine or more of twelve cell types associate significantly with age acceleration in most of six widely used clocks. https://doi.org/10.1111/acel.14071

75. Cruz-Gonzalez A, Okpala U, Gu C, Gomez A, Mews P, Vance JM et al. Methylation clocks fail to generalize across genetically admixed individuals. *eLife* (2026). The accuracy loss that motivates measuring each clock's meQTL exposure from its coefficients. https://doi.org/10.7554/eLife.105343

76. Patel S, Schwarz M, Riege K, Varshavsky M, Kestler HA et al. An improved epigenetic age estimation with TFMethyl Clock reveals DNA methylation changes during aging in transcription factor binding sites. bioRxiv 2025.10.07.680024. A preprint. Its central finding is that most CpGs used by established clocks do *not* overlap known transcription factor binding sites, which is the motivation for measuring coefficient mass against an annotation rather than assuming it. https://doi.org/10.1101/2025.10.07.680024

77. Borrus DS, Sehgal R, Armstrong JF, Kasamoto J, Gonzalez J, Higgins-Chen A. When to Trust Epigenetic Clocks: Avoiding False Positives in Aging Interventions. bioRxiv 2024.10.22.619720 (2024). PMC11526921. The decision rule `consensus()` implements: across six intervention datasets, five had exactly one significant clock, a first-generation clock every time, four of those five lost it under multiple-testing correction, and in no case did the principal-component version of the same clock corroborate. https://doi.org/10.1101/2024.10.22.619720

78. Sehgal R, Borrus DS, Armstrong JF, Gonzalez J, Kasamoto J, Markov Y et al. Responsiveness of epigenetic aging biomarkers to longevity interventions in humans. *Nature Medicine* (2026). The published successor to [77] by the same group, widened to 16 clocks and 94 further biomarkers across the 51 longitudinal studies of the TranslAGE database. The source of the clock grouping the FALCONAge report uses. https://doi.org/10.1038/s41591-026-04562-9

Chapter 28. Architecture

Framework for **A**ging **CL**ocks, **O**mics **N**ormalisation and **Age** scoring.

A single-stop toolkit for computing aging scores from multiomic data, usable identically from Python and R, on CPU or GPU, with a download module that takes public-database accessions and a preprocessing module that turns raw or public data into a scoreable matrix.

This document is the design record. It says which file computes what, what each function takes and returns, and why each decision went the way it did. The science it implements is in The science of aging clocks; every algorithmic claim here cites a section there rather than restating it.

What of this exists

> ! Read this before believing any section below
>
> This document was written **before** the code and is deliberately wider than what ships. Every section whose specification outruns the implementation says so at its head. The table here is the reconciliation, and it is checked against the running package rather than remembered, the counts below come from `len(ops.PREPROCESS)`, `registry.load()` and the container's own `Modality` type.

Current as of v1.0.0.

Area	Shipped in v1.0.0	Specified, not built
Clock catalogue	178 entries. **69 `bundled` score offline**: 66 ship a coefficient file, 3 (PhenoAge, KDM, homeostatic dysregulation) are formulas with none to ship. 57 `licensed` are tested scaffolds	The 52 `untraced` entries carry metadata but no traced coefficients
Modalities	DNA methylation, clinical chemistry, RRBS, **nanopore**, **transcriptomics**, **proteomics**	Single-cell as a *modality* (scAge is a model, not a container type); spatial
Inputs	Beta matrix, GEO series matrix, **raw IDATs**, RRBS, **bedMethyl**, **targeted panels**, **Olink NPX**, **SomaScan RFU**, **bulk counts**, clinical tables, ComputAgeBench	—
Normalisation	**pOOBAH, noob, BMIQ, probe masks** (§20 of the science page)	Control-probe dye correction; mLiftOver cross-platform harmonisation
Model classes	`LinearClock`, `ClinicalClock`, `ScaffoldClock`, **`PCLinearClock`**, **`AggregationClock`**, **`DivisionClock`**, **`NeuralClock`**, **`DeconvolutionClock`**, **`ScAgeReference`**	`MultiStageCoxClock`, `CompositeClock`
Preprocess ops	7: `beta_to_m`, `m_to_beta`, `scale`, `rank_normalize`, `quantile_normalize`, `binarize`, `simplex_projection`; and `standardise_within_sample` on the whole dataset	The other seven in §4.1
Postprocess ops	**21**, plus **24 analytic derivatives** for uncertainty propagation and 4 ops explicitly marked non-differentiable	—
Uncertainty	**Technical SE, conformal prediction intervals, per-probe ICC table, power, consensus**	Conformalised quantile regression (split conformal ships instead)
Batch	**Frozen-reference ComBat**: adding a plate cannot move an earlier one	—
Download	GEO series and sample, ArrayExpress, SRA/ENA, Zenodo by DOI, Hugging Face, **Figshare, PRIDE, MetaboLights, GDC**	Credentialed archives (Synapse, EGA, dbGaP, UK Biobank): documented, deliberately not automated
Registry storage	One `clocks.yaml` inside the package, loaded into frozen dataclasses	The per-clock file tree and pydantic schema of §3
Devices	CPU (numpy) and CUDA (torch), fp64 default	MPS is resolved and refused rather than supported
R interface	S3 classes, ggplot2 figures, bit-identical to Python, **including the uncertainty and batch verbs**	The raw-array chain and the model architectures are Python-only

Two gaps are worth naming rather than leaving in a cell.

Tier B coefficient provenance is not an engineering task. It is a per-clock literature hunt, find the paper, find the supplement, check the coefficient count against what circulates, record the discrepancy where they differ, and some of the 110 have no public supplement at all. `primary_source_traced: false` is the honest record of that debt.

The proteomic and transcriptomic chains ship without registry entries. Every published model in both families is licence-restricted, so each would enter as a licensed scaffold. Scoring a proteomic matrix today refuses with "no clocks to score", which is accurate rather than broken.

1. Scope, decisions and repository layout

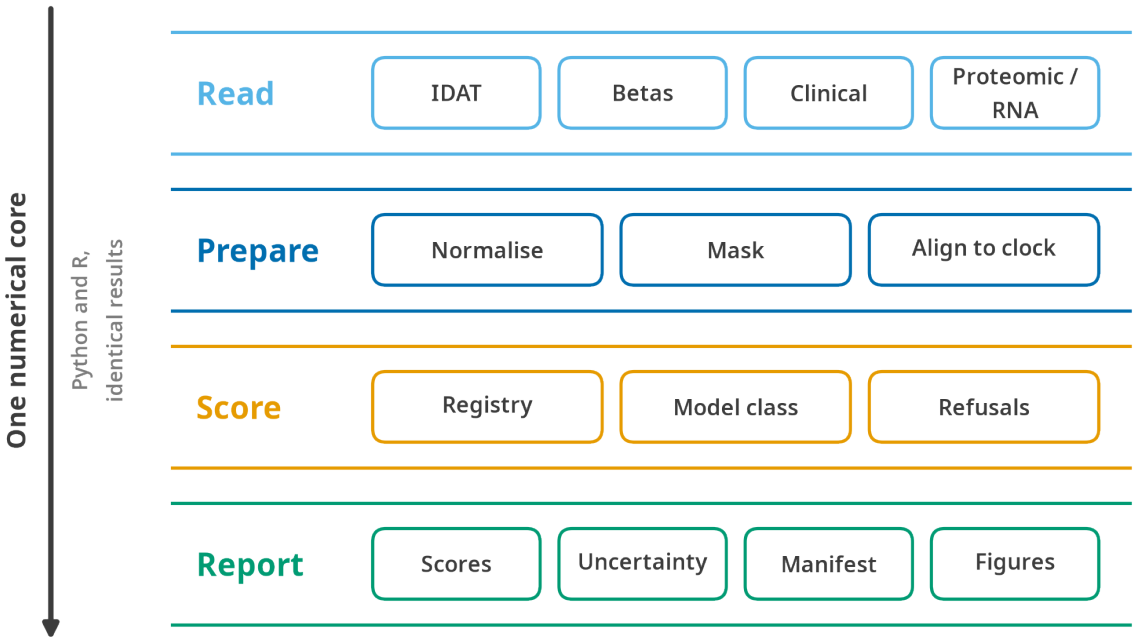


Figure 40: The four stages a dataset passes through, and the one numerical core beneath them

The R package does not reimplement any of this. It calls the same Python functions through reticulate, which is why the two languages return identical numbers rather than numbers that agree to a tolerance. The stages are separable: a beta matrix that is already normalised enters at Prepare, and a clinical table skips the methylation path entirely. What every route shares is the last stage, where a score is never emitted without the manifest recording how it was produced.

1.1 What each release delivered

Capability	Ships in 1.0.0	Later
DNA methylation clocks	172 catalogued: linear, PC, GrimAge family, mitotic, deconvolution, SystemsAge, AltumAge	—

Capability	Ships in 1.0.0	Later
Clinical chemistry	PhenoAge (both unit variants), KDM, homeostatic dysregulation	—
Clock coefficients	69 bundled, 57 scaffold-only, 52 untraced (§3.7, §4.7)	the untraced count narrows as sources are traced
Raw arrays	full chain: manifest fetch, pOOBAH, noob, BMIQ, probe masks	control-probe dye correction
Uncertainty	technical SE, conformal intervals, power, consensus	conformalised quantile regression
Batch	frozen-reference ComBat	—
Specimen and platform checks	species, specimen type, cell-free DNA refusal, platform bias in years	—
Transcriptomics	the tAge chain ships; no registry entries	tAge's 60 models, licence permitting
Proteomics	Olink and SomaScan readers ship; no registry entries	organAging, Oh 2023 organ clocks
Single cell	scAge (per-cell likelihood)	pseudobulk, CellBiAge, SpatialAgingClock
Foundation models	`NeuralClock` architecture, safetensors only	CpGPT as an *imputation* backend, not a clock
Download	GEO, ArrayExpress, SRA/ENA, Zenodo, Figshare, Hugging Face, GDC, PRIDE, MetaboLights; ComputAgeBench loader	Metabolomics Workbench
Preprocess	IDAT → beta, series matrix, beta CSV/parquet, clinical chemistry, the RNA-seq and Olink/SomaScan chains	—
Compute	CPU and CUDA, FP64 default, FP32 opt-in	MPS validated, ROCm
Interfaces	Python API, R API, CLI, Docker (CPU + CUDA)	-

The v1.0 line is drawn where it is because methylation plus clinical chemistry covers the overwhelming majority of real use, is fully specified in the science notes down to the constants, and has published values available to test against.

Every **architecture** is written from scratch, with the equations taken from the science notes, §4 rather than from anybody's source: 15 preprocess ops, 15 postprocess transforms and 10 special forward passes are specified, of which v1.0 implements the seven, nine and zero that its 23 scoring clocks reach for. No clock implementation is imported from another package.

Coefficients are a different matter, because they are fitted data rather than a procedure and cannot be written at all. §3.6 surveys where each set actually comes from and §3.7 sorts them into three tiers by what FALCONAge may distribute. Forty clocks (the GrimAge, PC GrimAge and SystemsAge families, DunedinPACE and a few others) ship as a complete, tested scaffold with no numbers, because their coefficients are research-use-only or have no traceable public source. §4.7 covers what that means in practice.

1.2 The four architectural decisions

One numerical core, in Python. The R package is a native-feeling front end that calls the Python core through `reticulate`. This is the design `tAge` uses (`tAge/R/predict.R`), and the reason is that "identical

results in both languages" is either guaranteed by construction or it is a permanent testing liability. Two native implementations means writing 15 preprocess ops, 15 postprocess transforms and 10 special forward passes twice, then proving on every release that they still agree. One core means the R and Python answers are the same bytes because they are the same computation. GPU support in R comes free. The cost is that R users need a Python environment, which `falconage_install()` and the Docker image handle.

Quarto for both documentation sites. `quartodoc` renders the Python API, `pkgdown` renders the R API, and one Quarto site wraps both with the same theme and navigation. Tutorials are `.qmd` with Python and R tabs showing the same analysis. The risk is that quartodoc is younger than Sphinx and Quarto 2 (a Rust rewrite) lands late 2026, so the Quarto version is pinned in CI and in the Docker image. Docstrings are numpydoc-format, which means the fallback to Sphinx + `pydata-sphinx-theme` requires no source changes if the toolchain becomes a problem.

FALCONAge.

Context	Name
PyPI distribution	`falconage`
Python import	`import falconage as fa`
R package	`FALCONAge`
R library call	`library(FALCONAge)`
CLI	`falconage <verb>`
Docker image	`falconage:VERSION-cpu`, `falconage:VERSION-cuda`
Docs	`https://bhagesh-h.github.io/FALCONAge/`
Registry namespace	`falconage-registry` on Hugging Face

PyPI lowercase follows PEP 8 and PyPI normalisation; R CamelCase follows the `cyRAVEN` convention. `falconage` and `FALCONAge` were both free on PyPI at the time of writing; CRAN availability must still be confirmed with `available::available("FALCONAge")` before the first submission.

Versioning. One version number governs `python/pyproject.toml`, `r/DESCRIPTION`, the Docker tag and the git tag; CI refuses a release when they disagree (§12). The clock registry carries its own independent `schema_version` and `registry_version`, because adding a clock should not force a package release and a package release should not silently change which weights a user gets. Every result records both.

1.3 Repository layout

```
FALCONAge/
├── README.md
├── PUBLISHING.md
├── CITATION.cff
├── LICENSE                          # GPL-3
├── CHANGELOG.md                     # shared; r/NEWS.md is generated from it
│
├── python/
│   ├── pyproject.toml
│   ├── README.md                    # symlink or short pointer to the root README
│   ├── src/falconage/
│   │   ├── __init__.py              # public API re-exports, __version__
```

```
│   │   ├── _version.py
│   │   │
│   │   ├── core/
│   │   │   ├── device.py          # resolve_device, resolve_dtype, DeviceSpec
│   │   │   ├── container.py       # FalconData
│   │   │   ├── manifest.py        # RunManifest
│   │   │   ├── logging.py         # get_logger, verbosity, progress
│   │   │   ├── config.py          # FalconConfig, from_yaml, from_env
│   │   │   ├── units.py           # UnitRegistry, convert, require_units
│   │   │   └── errors.py          # exception hierarchy
│   │   │
│   │   ├── io/
│   │   │   ├── methylation.py     # read_betas, read_idat_dir, read_series_matrix
│   │   │   ├── clinical.py        # read_clinical
│   │   │   ├── expression.py      # read_counts
│   │   │   ├── proteomics.py      # read_olink, read_somascan (v1.2)
│   │   │   ├── anndata_io.py      # read_h5ad, write_h5ad
│   │   │   └── tables.py          # read_sample_table, write_results
│   │   │
│   │   ├── download/
│   │   │   ├── base.py            # Downloader ABC, DownloadResult
│   │   │   ├── resolve.py         # accession → Downloader dispatch
│   │   │   ├── cache.py           # content-addressed cache
│   │   │   ├── transfer.py        # resumable HTTP/FTP, checksum, retry
│   │   │   ├── geo.py             # GSE/GSM
│   │   │   ├── arrayexpress.py    # E-MTAB
│   │   │   ├── sra.py             # SRP/SRR/PRJNA
│   │   │   ├── zenodo.py          # DOI
│   │   │   ├── figshare.py
│   │   │   ├── huggingface.py
│   │   │   ├── gdc.py             # TCGA
│   │   │   ├── pride.py           # PXD (v1.2)
│   │   │   ├── metabolights.py    # MTBLS, ST (v1.2)
│   │   │   ├── credentialed.py    # Synapse/EGA/dbGaP/UKB guidance + hooks
│   │   │   └── metadata.py        # source metadata → common sample table
│   │   │
│   │   ├── preprocess/
│   │   │   ├── methylation/
│   │   │   │   ├── idat.py        # IDAT parse
│   │   │   │   ├── noob.py
│   │   │   │   ├── bmiq.py
│   │   │   │   ├── sesame.py
│   │   │   │   ├── quality.py     # detection p, probe masks
│   │   │   │   ├── platform.py    # detect_platform, harmonise_probe_ids
│   │   │   │   └── epicv2.py      # aggregate_replicate_probes
│   │   │   ├── clinical.py        # validate_ranges, convert_units
│   │   │   ├── expression.py
│   │   │   ├── proteomics.py      # v1.2
```

```
│   │   │   ├── impute.py          # dataset-level imputation
│   │   │   └── qc.py              # QCReport
│   │   │
│   │   ├── registry/
│   │   │   ├── schema.py          # pydantic models
│   │   │   ├── registry.py        # ClockRegistry: list/search/get/filter
│   │   │   ├── weights.py         # HF resolution, safetensors load, cache
│   │   │   ├── vocabulary.py      # controlled vocabulary
│   │   │   └── data/              # the YAML files (packaged)
│   │   │
│   │   ├── models/
│   │   │   ├── base.py            # FalconClock ABC
│   │   │   ├── ops/
│   │   │   │   ├── preprocess.py  # the 15 preprocess ops
│   │   │   │   ├── postprocess.py # the 15 postprocess transforms
│   │   │   │   └── functional.py  # shared kernels (rank, quantile, simplex)
│   │   │   ├── linear.py          # LinearClock, PCLinearClock
│   │   │   ├── cox.py             # MultiStageCoxClock (GrimAge family)
│   │   │   ├── neural.py          # NeuralClock (AltumAge)
│   │   │   ├── aggregation.py     # AggregationClock (mitotic clocks)
│   │   │   ├── deconvolution.py   # DeconvolutionClock
│   │   │   ├── composite.py       # CompositeClock (SystemsAge, DNAmFitAge)
│   │   │   ├── clinical.py        # PhenoAge, KDM, HD
│   │   │   └── species.py         # mammalian species-lookup transforms
│   │   │
│   │   ├── score/
│   │   │   ├── predict.py         # score()
│   │   │   ├── features.py        # feature alignment, clock-level imputation
│   │   │   └── result.py          # FalconResult
│   │   │
│   │   ├── analysis/
│   │   │   ├── acceleration.py    # absolute, residual, within-group
│   │   │   ├── survival.py        # Cox, C-index, time-dependent AUC
│   │   │   ├── association.py     # logistic, linear, interactions
│   │   │   ├── reliability.py     # ICC, Fisher-Z pooling
│   │   │   ├── benchmark.py       # AA1/AA2, MedAE, bias, total score
│   │   │   └── agreement.py       # cross-clock correlation, clustering, UMAP
│   │   │
│   │   ├── plot/
│   │   │   ├── theme.py           # shared palette and rcParams
│   │   │   ├── age.py             # ba_vs_ca, acceleration_density
│   │   │   ├── agreement.py       # clock_corr, clock_umap
│   │   │   ├── qc.py              # missingness, beta_distribution
│   │   │   ├── benchmark.py       # class_bench, medae, bias
│   │   │   ├── survival.py        # forest
│   │   │   ├── systems.py         # organ radar, system heatmap
│   │   │   └── trajectory.py      # timecourse
│   │   │
```

```
│   │   ├── report/
│   │   │   ├── html.py            # self-contained HTML report
│   │   │   └── templates/
│   │   │
│   │   └── cli/
│   │       ├── __main__.py
│   │       ├── app.py             # typer app
│   │       └── commands/          # download, preprocess, score, analyse,
│   │                              # bench, clocks, cache, config, report
│   └── tests/
│       ├── unit/  integration/  property/  conformance/
│       └── conftest.py
│
├── r/
│   ├── DESCRIPTION  NAMESPACE  NEWS.md  _pkgdown.yml  .Rbuildignore
│   ├── R/
│   │   ├── FALCONAge-package.R    # @keywords internal "_PACKAGE" + overview
│   │   ├── zzz.R                  # .onLoad delayed import, options
│   │   ├── install.R              # falconage_install, falconage_config
│   │   ├── bridge.R               # py handle, error translation, marshalling
│   │   ├── download.R  preprocess.R  score.R  analysis.R  plot.R  report.R
│   │   ├── classes.R              # falcon_data, falcon_result S3
│   │   ├── clocks.R               # list_clocks, clock_info, cite_clock
│   │   ├── cli-options.R          # option parser for inst/scripts
│   │   └── utils.R
│   ├── man/                       # roxygen-generated
│   ├── vignettes/
│   │   ├── FALCONAge.Rmd  methylation.Rmd  clinical.Rmd
│   │   ├── download.Rmd  gpu.Rmd  benchmarking.Rmd  clocks.Rmd
│   ├── tests/testthat/
│   └── inst/
│       ├── CITATION
│       ├── scripts/falconage.R    # CLI front end
│       └── extdata/               # small example inputs
│
├── registry/                      # source of truth, packaged into both
│   ├── schema.yaml
│   ├── vocabulary.yaml
│   ├── clocks/*.yaml              # one file per clock
│   └── validate.py
│
├── docs/                          # Quarto site
│   ├── _quarto.yml
│   ├── index.qmd  installation.qmd  quickstart.qmd
│   ├── guide/*.qmd                # narrative, Python+R tabs
│   ├── reference/                 # quartodoc output (Python)
│   ├── r/                         # pkgdown output (R), built in
│   └── _extensions/
```

```
│
├── docker/
│   ├── Dockerfile.cpu  Dockerfile.cuda
│   ├── entrypoint.sh  install_deps.R
│   └── docker-compose.yml
│
├── tests/vectors/                  # shared gold-standard IO pairs
│   ├── manifest.yaml
│   └── <clock>/{input.parquet,expected.parquet}
│
└── .github/workflows/
    ├── python-test.yaml  r-cmd-check.yaml  docs.yaml
    ├── docker.yaml  release-pypi.yaml  release-r.yaml
    └── registry-validate.yaml
```

1.4 What actually got laid down

Flatter than the plan in three places, each for the same reason: a directory with one file in it is a directory you have to open to find out it has one file in it.

```
python/src/falconage/
├── core/       backend.py config.py container.py errors.py
│               logging.py manifest.py units.py
│               tissue.py preanalytical.py
├── io/         __init__.py methylation.py            # readers, platform detection
│               bench.py  # ComputAgeBench loader
├── preprocess/ __init__.py methylation.py            # prepare, qc, EPIC v2, units
│               manifest.py idat.py bmiq.py masks.py  # raw-array chain
│               batch.py  # frozen-reference ComBat
│               proteomic.py transcriptomic.py  # modality chains
├── uncertainty/ __init__.py  # SE, conformal, ICC
├── registry/   registry.py  data/clocks.yaml         # all 178 entries, one file
│               data/probe_icc.csv.gz  # reliability table
│               data/platform_bias.csv conformal.csv  # derived artefacts
│               data/references/  # fill values for absent CpGs
│               data/evidence.yaml  # curated associations
├── models/     linear.py clinical.py ops.py pc.py
│               aggregation.py division.py neural.py single_cell.py
├── score/      __init__.py                           # score(), combine()
├── analysis/   __init__.py                           # acceleration, cox, icc,
│                                                     # benchmark, power, consensus
├── plot/       __init__.py                           # public surface + save_all
│               _common.py spec.py colorscheme.yaml   # canvas, theme, captions
│               accuracy.py acceleration.py agreement.py
│               qc.py benchmark.py outcomes.py atlas.py
│               uncertainty.py
├── report/     html.py
└── cli/        app.py __main__.py                    # + report/power/consensus verbs
```

```
python/tools/   build_probe_icc.py build_platform_bias.py  # derivations
                build_conformal.py build_references.py
r/R/            FALCONAge-package.R zzz.R install.R bridge.R
                data.R score.R analysis.R clocks.R download.R
                plot.R atlas.R uncertainty.R
docs/           _quarto.yml reference-groups.yml build_docs.py
                build_downloads.py build_catalogue.py build_gallery.py
                index.qmd clocks.qmd gallery.qmd gpu.md guide/
test/           run_all.py gpu_check.py responsive_check.py
                check_versions.py data/ output/ output_figures/
```

Two collapses and one reversal. `models/ops/{preprocess,postprocess,functional}.py` became one `ops.py`, because the shared kernels the third file was for are four functions. `registry/{schema,weights,vocabulary}.py` became `registry.py` plus the data file: the vocabulary is a frozenset: the schema is the dataclass, and weight resolution is one function because nothing is fetched at run time.

`plot/` went the other way, and the original reasoning was wrong. It was kept as one module plus `atlas.py` on the argument that splitting the figures by subject would make the shared helpers a circular import waiting to happen. It grew to 1,523 lines, an implementation living in a package initialiser, which is the thing an initialiser should not be. The circularity worry was avoidable by naming the problem: the helpers are not shared *between* figure families: they are shared *by* them, so they belong in a `_common.py` that imports from nobody in the package. Each family module imports `_common`, `__init__` imports the families and re-exports, and the dependency graph is a tree. `save_all` stays in `__init__` because orchestrating the families is the one job that legitimately knows about all of them. No public name moved; `fa.plot.ba_vs_ca` is where it was.

There is no top-level `registry/` directory and no `tests/vectors/` tree; the catalogue ships inside the package and the gold-standard vectors live beside the tests that assert on them.

Chapter 29. Core engine and clock registry

2. Core engine

2.1 Device and dtype

`falconage/core/device.py`

```
@dataclass(frozen=True)
class DeviceSpec:
    device: torch.device          # cpu | cuda:N | mps
    dtype:  torch.dtype           # float64 | float32
    batch_size: int
    backend: Literal["torch", "numpy"]

def resolve_device(
    device: str = "auto",         # "auto" | "cpu" | "cuda" | "cuda:1" | "mps"
    dtype:  str = "float64",      # "float64" | "float32"
    batch_size: int | None = None,
) -> DeviceSpec
```

`auto` picks CUDA when `torch.cuda.is_available()`, then MPS, then CPU, and logs the choice, mirroring pyaging's `set_torch_device` (`pyaging/predict/_pred_utils.py:398`). `batch_size` defaults to 1024 on CPU and is raised to fill available VRAM on CUDA, capped so that the largest PC rotation (125,175 × k for SystemsAge) fits.

> **FALCONAge diverges here in four ways, three of them deliberate.** The module is `core/backend.py`, the function is `resolve()`, and `DeviceSpec` carries plain strings with no `batch_size` field: nothing shipping is large enough to need batching, and a field that is always its default is a field that is wrong the first time it matters.
>
> The substantive difference is `auto`. It resolves to **CPU even when a CUDA device is present**, because on the clocks that ship the card loses: a 2,340-feature dot product is a rounding error against the pandas alignment that feeds it, and the transfer is pure cost. Measured at 4.6× slower on an RTX 4060 at 16,384 samples. Picking CUDA because one exists would make the common case several times worse on every machine that has one, silently, so the GPU is opt-in via `device="cuda"` or `FALCONAGE_DEVICE=cuda`. The numbers are in GPU support.
>
> `resolve()` also refuses a device it cannot provide rather than falling back. A run asked for a GPU and quietly given a CPU looks like a very slow success, and the user finds out hours later.

FP64 is the default because that is what pyaging uses and what the gold-standard vectors are generated at. FP32 roughly doubles throughput on consumer GPUs, which have poor FP64 rates, and is opt-in. Clocks whose registry entry sets `requires_fp64: true` are promoted back to FP64 for their own forward pass and a warning is emitted; this covers the PC clocks, whose 78,464- and 125,175-dimensional centring and rotation is ill-conditioned (the science notes, §14.3, §16).

```
def check_dtype_safety(clock: ClockMeta, spec: DeviceSpec) -> DeviceSpec:
    """Promote to FP64 for clocks that need it. Returns a possibly-modified spec."""
```

Memory guard: before a clock runs, estimate `n_samples × n_features × itemsize` plus the rotation matrix, and either lower the batch size or raise `InsufficientMemoryError` with the required figure, rather than letting CUDA OOM mid-run.

2.2 The data container

`falconage/core/container.py`

`FalconData` wraps `anndata.AnnData` rather than replacing it, so the on-disk artefact is a plain `.h5ad` that both languages and every scverse tool can read.

```
class FalconData:
    adata: AnnData                  # X = primary modality, obs = samples, var = features
     modality: str                          # "methylation" | "clinical" | "expression"
| "proteomics"
    platform: str | None             # "illumina_450k" | "illumina_epic_v2" | ...
    species: str                     # "homo_sapiens" | ...
    units: dict[str, str]            # feature → unit, for clinical chemistry
```

Conventions, matching pyaging's AnnData slot contract and extending it:

Slot	Content
`X`	samples × features, the primary modality
`obs`	`age`, `sex`, `tissue`, `condition`, `subject_id`, `timepoint`, `time`, `status`, plus predictions after scoring
`var`	`feature_id`, `id_type`, `platform`, `chromosome`, `position`
`obsm["X_<clock>"]`	transient per-clock feature slice, deleted after scoring unless `keep=True`
`layers["<modality>"]`	a second modality on the same features
`uns["falconage"]`	the run manifest
`uns["<clock>_percent_na"]`, `uns["<clock>_missing_features"]`, `uns["<clock>_metadata"]`	per-clock diagnostics

Multi-modality with different feature spaces (850K methylation probes alongside 3,000 proteins) uses `MuData`, with `FalconData.from_mudata()` / `.to_mudata()`. A single `FalconData` always has one primary feature space.

Constructors:

```
FalconData.from_dataframe(df, modality, platform=None, metadata_cols=(), units=None)
FalconData.from_anndata(adata, modality, ...)
FalconData.read_h5ad(path)
FalconData.write_h5ad(path)
```

`from_dataframe` takes samples in rows and features in columns, splits `metadata_cols` into `obs`, and preserves `df.index` as sample identifiers. This is the same signature as pyaging's `df_to_adata` minus the imputation, which moves to the preprocess module where it belongs.

2.3 Run manifest

`falconage/core/manifest.py`

Everything needed to reproduce a run, written into `uns["falconage"]` and into a sidecar `run_manifest.json` next to the results.

```
@dataclass
class RunManifest:
    falconage_version:  str
    registry_version:   str
    registry_schema:    str
    python_version:     str
    r_version:          str | None      # set when the call came through reticulate
    platform:           str             # OS, arch
    device:             str
    dtype:              str
    torch_version:      str
    cuda_version:       str | None
    timestamp:          str             # ISO 8601, UTC
    command:            str             # the resolved call
    config:             dict            # fully resolved, defaults expanded
    inputs:             list[InputRecord]   # path, sha256, n_samples, n_features
   clocks:         list[ClockRun]    # name, version, weights_sha256, percent_na, warnings
    warnings:           list[str]
```

`ClockRun.weights_sha256` is what makes a result auditable: two runs that report the same score for the same clock either used the same weights or the manifest says they did not.

2.4 Logging and verbosity

`falconage/core/logging.py`

Three levels, matching cyRAVEN's `cyRAVEN.verbose` option so the two front ends behave the same:

Level	Python	R
`none`	`logging.CRITICAL`	`options(FALCONAge.verbose = "none")`
`inform` (default)	`logging.INFO`, indented progress	`"inform"`
`debug`	`logging.DEBUG`, per-op timings and shapes	`"debug"`

Progress bars via `rich` on a TTY, plain line output otherwise, suppressed entirely under `none`. Every message goes through the logger, so `logging.disable()` in Python and `suppressMessages()` in R both work.

2.5 Configuration

`falconage/core/config.py`

Precedence, lowest to highest: packaged defaults → user config file (`~/.config/falconage/config.yaml`) → environment (`FALCONAGE_*`) → explicit function arguments.

```yaml
# ~/.config/falconage/config.yaml
device: auto
dtype: float64
batch_size: 1024
verbose: inform
cache_dir: ~/.cache/falconage
registry_version: "1.1.0"        # pin to make a project reproducible
hf_token: null
max_download_gb: 50
```

2.6 Units

`falconage/core/units.py`

The single most common source of wrong answers in this field is a clinical biomarker in the wrong unit (the science notes, §16). The unit registry makes that a hard error.

```python
class UnitRegistry:
    def convert(self, value: ArrayLike, marker: str, frm: str, to: str) -> ArrayLike
    def canonical(self, marker: str, variant: str) -> str
   def require(self, markers: Sequence[str], units: dict[str, str] | None) -> dict[str, str]
```

`require` raises `MissingUnitsError` naming every marker whose unit was not declared. There is no default and no inference from magnitude, a creatinine of 80 is plausible as both µmol/L and an absurd mg/dL, and guessing is how the 88× error happens. Conversions carried:

Marker	Units	Factor
albumin	g/L ↔ g/dL	×10
creatinine	µmol/L ↔ mg/dL	×88.42
glucose	mmol/L ↔ mg/dL	×18.016
CRP	mg/L ↔ mg/dL	×10
total/HDL/LDL cholesterol	mmol/L ↔ mg/dL	×38.67
triglycerides	mmol/L ↔ mg/dL	×88.57
urea/BUN	mmol/L ↔ mg/dL	×2.801
uric acid	µmol/L ↔ mg/dL	×59.48
bilirubin	µmol/L ↔ mg/dL	×17.1

2.7 Error hierarchy

`falconage/core/errors.py`

```
FalconError
├── ConfigurationError
│   ├── MissingUnitsError
│   └── InvalidDeviceError
```

```
├── DataError
│   ├── FeatureMismatchError        # 100% of a clock's features absent
│   ├── ExcessiveMissingnessError   # above the configured threshold
│   ├── PlatformMismatchError       # EPIC v2 probes not aggregated, etc.
│   ├── MissingCovariateError       # clock needs age/sex and it is not there
│   └── OutOfRangeError             # beta outside [0,1], negative counts
├── RegistryError
│   ├── UnknownClockError
│   ├── WeightsUnavailableError
│   └── SchemaValidationError
├── DownloadError
│   ├── AccessionNotFoundError
│   ├── ChecksumMismatchError
│   └── CredentialsRequiredError
└── ComputeError
    ├── InsufficientMemoryError
    └── NumericalInstabilityError
```

Every error carries `.remedy`, a one-sentence statement of what to do about it, which the R bridge surfaces as the condition message and the CLI prints after the traceback.

3. Clock registry

3.1 Why the registry is a separate artefact

The registry is versioned independently of the package, packaged into both language distributions, and validated in its own CI job. That separation exists because adding a clock and releasing software are different events with different risk profiles: a user pinning `registry_version: "1.0.0"` gets the same coefficients in a year regardless of package upgrades, and a registry fix does not require a PyPI release.

3.2 Schema

> **v1.0 stores this differently.** All 178 entries live in one file, `python/src/falconage/registry/data/clocks.yaml`, loaded into frozen dataclasses by `registry/registry.py`. There is no `registry/` directory at the repository root, no file per clock, and no pydantic.
>
> The per-clock tree below is the right shape for a registry that outside contributors edit concurrently, one file per pull request, no merge conflicts. At 178 entries maintained by one author it bought nothing and cost a directory walk on every import, plus 178 files to keep a schema version in step across. The single file is 178 entries the loader validates in one pass, and it travels inside the wheel so a clock catalogue is never a download. Pydantic went the same way: the fields are flat, the loader coerces and defaults them in about forty lines, and a runtime dependency that exists to check a file the package itself ships is a dependency earning nothing.
>
> The field vocabulary below is otherwise accurate: it is what the loader reads.

```yaml
# one entry in falconage/registry/data/clocks.yaml
id: grimage2
display_name: GrimAge2
schema_version: "1.0"
clock_version: "1.0.0"

provenance:
  year: 2022
  citation: "Lu AT, Binder AM, Zhang J, et al. DNA methylation GrimAge version 2. Aging.
2022;14(23):9484-9549."
  doi: "10.18632/aging.204434"
  last_author: "Steve Horvath"
  journal: "Aging"
  approved_by_author: false
  evidence: paper-confirmed        # author | code | paper | supplement-confirmed

license:
  spdx: null
  research_only: true
  redistributable: true
  notes: "Research use only; commercial use requires a licence from UCLA."

biology:
  data_type: dna_methylation
  species: homo_sapiens
  tissue: [whole_blood]
  platform: [illumina_450k]
  population: older_adults
  generation: second               # first|second|pace|causal|mitotic|system|foundation
  predicts: [mortality_risk]
  training_target: [mortality]

output:
  unit: years
  scale_type: age_years            # governs which downstream operations are legal
  plausible_range: [0, 150]        # a value outside this is flagged, not clipped

features:
  n_features: 1032
  id_type: cpg_illumina
  requires_covariates: [age, sex]
  reference_values: true

architecture:
  kind: multi_stage_cox
  sub_models:                      # order is load-bearing, see §4.3
    - {name: GDF15,     n_features: 138}
    - {name: B2M,       n_features: 92}
```

```
    - {name: CystatinC, n_features: 88}
    - {name: TIMP1,     n_features: 43}
    - {name: ADM,       n_features: 187}
    - {name: PAI1,      n_features: 211}
    - {name: Leptin,    n_features: 187}
    - {name: PACKYRS,   n_features: 173}
    - {name: LogCRP,    n_features: 132}
    - {name: A1C,       n_features: 87}
  covariate_order: [age, female]
  requires_fp64: false

preprocess:
  - op: fill_nan_with_reference

postprocess:
  - op: cox_to_years
    params:
      cox_mean: 15.370829484122
      cox_std:  1.09534876966487
      age_mean: 66.0943807965085
      age_std:  9.05974444998421

weights:
  uri: "hf://bhagesh-h/falconage-registry/clocks/grimage2/weights.safetensors"
  sha256: "..."
  size_bytes: 8421376
         reference_uri:    "hf://bhagesh-h/falconage-registry/clocks/grimage2/reference.
safetensors"
  reference_sha256: "..."

reliability:
  technical_icc:  {value: 0.83, source: "Higgins-Chen 2022", n: null}
  biological_icc: {value: 0.61, source: "TranslAGE 2025", n: null}

benchmark:
  computage:
    aa2_passed: null
    aa1_passed: null
    medae: null
    median_bias: null
    total_score: null

gold_standard:
  input:     "tests/vectors/grimage2/input.parquet"
  expected:  "tests/vectors/grimage2/expected.parquet"
  tolerance: 1.0e-6
  source:    "pyaging 0.3.0, FP64, CPU"
```

```
known_discrepancies: []
```

3.3 The discrepancy field is load-bearing

The science, §15.2 documents the eleven clocks where the published paper and the coefficient set in circulation disagree. Every implementation in the field silently inherits one side of each disagreement. FALCONAge records both and warns at score time:

```
known_discrepancies:
  - field: n_features
    paper_says: 96
    implementation_has: 251
    source: "pyaging audit_report.md"
    resolution: implementation
    note: >
      The paper and supplement report a 96-CpG primary ultrasound model. The
      third-party coefficient file everyone uses carries 251 CpGs and cannot be
      mapped to any of the paper's six LASSO variants. No author-hosted 251-CpG
      model was located.
```

The nine to encode at minimum: `bohlin` (96 vs 251), `cvdwesterman` (1,305 vs 235), `zhangmortality` (integer quartile count vs continuous weighted sum), `yingcausage` (586 vs 585), `yingdamage` (1,090 vs 1,089), `yingadaptage` (1,000 vs 999), `senchronoage` (188 vs 187), `sencultureage` (141 vs 142), `senmortalityage` (89 vs 91), plus `bocklandt` (2 CpGs vs 1) and `downsyndrome` (EWAS estimates repurposed as a zero-intercept score).

Warning behaviour is a config knob: `discrepancy_warnings: once | always | never`, defaulting to `once` per clock per session.

3.4 Controlled vocabulary

`registry/vocabulary.yaml` fixes the allowed values for `data_type`, `species`, `tissue`, `platform`, `population`, `generation`, `predicts`, `training_target`, `unit`, `scale_type` and `architecture.kind`. This is the mechanism that kept pyaging's 173 entries coherent through an audit that collapsed 94 free-text tissue strings to 30 controlled terms (`pyaging/clocks/metadata/audit_report.md`). `registry/validate.py` fails CI on any value outside the vocabulary, any missing required field, any weight URI without a checksum, and any clock without a gold-standard vector.

3.5 Registry API

`falconage/registry/registry.py`

```
class ClockRegistry:
    @classmethod
    def load(cls, version: str | None = None) -> ClockRegistry

    def get(self, clock_id: str) -> ClockMeta
    def list(self) -> list[str]
     def search(self, query: str) -> list[ClockMeta]            # free text over name,
```

```
citation, notes
     def filter(self, **criteria) -> list[ClockMeta]            # data_type=, species=,
generation=, ...
    def by_doi(self, doi: str) -> list[ClockMeta]
    def compatible_with(self, data: FalconData,
                        min_coverage: float = 0.8) -> list[ClockMeta]
    def cite(self, clock_id: str, style: str = "plain") -> str
```

`compatible_with` is the function that makes a 173-clock registry usable: hand it a loaded dataset and it returns the clocks whose species, platform and feature coverage actually match, sorted by coverage. This replaces the current field practice of running everything and reading the missingness column afterwards.

3.6 Coefficient provenance and the three availability tiers

A clock is two things: an **architecture**, which is a procedure described in a paper, and a **coefficient set**, which is data produced by fitting that procedure to a cohort. FALCONAge writes every architecture from scratch. It cannot write a coefficient set from scratch, because there is no procedure that regenerates Horvath's 353 weights without Horvath's training data.

So the question for each clock is only ever *where do its numbers come from and may we ship them.* Surveying pyaging's 173 notebooks for the URL each one fetches gives the answer for the 159 methylation and clinical clocks in v1:

Provenance	n	Traceable to the authors
Publisher supplementary file	31	yes, the paper's own Additional file
Author's own code repository	48	yes
Author-hosted file share (Box, Google Drive)	24	yes, on fragile links
Copied package-to-package, primary source unrecorded	37	not without tracing
No public primary source	13	no
Embedded inline, small enough to read off the paper	6	yes

The 37 in the fourth row are the reason the eleven discrepancies in §3.3 exist. epiTOC2′s coefficients in pyaging are a file lifted from biolearn's repository; the McCartney scores and most of the deconvolution panels are the same story. Copying a copy is how a transcription error becomes a field-wide convention, and tracing each back to its supplement is the only way to find out which reading is right.

3.7 The three tiers

Every registry entry carries `availability`, one of three values. This governs what ships, what is fetched, and what the user has to do.

Tier	n	Architecture	Coefficients	User action
A - open	38	ships	ship, extracted from the primary source and redistributed	none
B - fetch on first use	95	ships	fetched from the author's URL at first	none, but needs network once

Tier	n	Architecture	Coefficients	User action
			use, cached, licence printed once	
C - permission required	28	ships	never fetched; the user obtains them and points FALCONAge at a local file	obtain a licence or file

As shipped in v1.0.0. The registry names the tiers by what they mean: `bundled` (A), `untraced` (B) and `licensed` (C). The letters are still accepted by `filter`, `list_clocks` and `falconage clocks list --tier`. The counts are 69 bundled, 52 untraced and 57 licensed. Fetching on first use was not built: an untraced clock carries its metadata and no coefficients, and scoring it raises with the reason.

Tier A is where the coefficient set came from a publisher supplement or a permissively licensed author repository and may be redistributed. Includes PhenoAge, KDM and homeostatic dysregulation, which need no network at all.

Tier B is where the coefficients are public but FALCONAge has no clear right to re-host them, or where the primary source has not yet been traced. The scaffold ships; the numbers arrive on first use. Forty-eight of these are author repositories whose licences have not been individually reviewed and will move to A as each is checked.

Tier C is the subject of §4.7.

```yaml
# in every registry entry
availability: B
coefficient_source:
  tier: T2
  url: "https://github.com/rsinghlab/AltumAge/raw/main/example_dependencies/..."
  format: csv
  sha256: "..."
  extractor: extractors/altumage.py
  licence: "MIT (repository)"
  redistributable: true
  retrieved: "2026-08-07"
  primary_source_traced: true
```

`primary_source_traced: false` is a visible debt, counted in the registry validation report, and `falconage clocks list --untraced` prints them.

3.8 Coefficient extraction

`registry/extractors/`. One small script per clock, or per family where the format is shared. Each reads the primary source in whatever shape it was published, xlsx, csv, a zip of csvs, an RData object, a supplementary PDF table, and emits a normalised two-column file plus a provenance record.

```
registry/extractors/
├── _base.py                # fetch, checksum, normalise, write, record
├── horvath2013.py          # Springer MOESM3 csv
```

```
├── hannum.py               # Elsevier mmc2.xlsx, sheet 2, cols B:C
├── ying.py                 # Nature Aging MOESM6 zip, three clocks in one archive
└── ...
```

```python
class Extractor(ABC):
    clock_id: str
    source_url: str
    expected_sha256: str
    expected_n_features: int

    @abstractmethod
    def parse(self, raw: bytes) -> pd.DataFrame:      # feature_id, coefficient
        ...

    def run(self) -> ExtractionResult:
        # fetch -> verify checksum -> parse -> assert n_features -> assert no
        # duplicate feature ids -> compare against any prior extraction -> write
```

Two assertions do most of the work. `expected_n_features` catches a supplement whose layout changed or a sheet index that was off by one. The comparison against the prior extraction turns an upstream file being silently revised into a failed CI run rather than a changed result.

Diffing against the field. For every clock where pyaging, biolearn, computage or methylCIPHER also carries coefficients, extraction runs a comparison and writes the result to `registry/diffs/<clock>.md`: features present in one and not the other, coefficients differing beyond 1e-9, and a verdict. Disagreements are published rather than resolved silently. This is what turns the eleven known discrepancies from inherited folklore into a documented finding, and it is expected to surface more.

3.9 Weight loading

`falconage/registry/weights.py`

```python
def resolve_weights(meta: ClockMeta, cache_dir: Path | None = None) -> Path
def load_weights(meta: ClockMeta, spec: DeviceSpec) -> dict[str, torch.Tensor]
def register_local_weights(clock_id: str, path: Path, sha256: str | None = None) -> None
```

Rules:

- **safetensors only.** `torch.load(weights_only=False)`, which pyaging uses (`pyaging/predict/_pred_utils.py:90`), executes arbitrary code from a downloaded file. Architecture lives in the registry YAML and in `models/`; tensors live in safetensors; nothing pickled is ever loaded.
- **Checksum verified** against the registry before use; a mismatch raises rather than warns.
- **Hugging Face cache** for tier A, the same cache for tier B, so weights are shared with other tools and survive a reinstall.
- **Offline mode**: `FALCONAGE_OFFLINE=1` uses only what is cached and raises `WeightsUnavailableError` naming the missing clock rather than attempting a network call.
- **Bundling**: nothing over 1 MB ships inside the PyPI or CRAN artefact. The clinical-chemistry sets are the exception. PhenoAge is ten numbers, so that path works with no network at all.

- **Licence surfacing**: tier B prints the source licence once per clock per session on first fetch. Tier C prints the obtain-it-yourself instructions.

  As shipped. Bundled coefficients ship inside the package and are checked against the registry's SHA-256 before use; untraced clocks are not fetched (§3.7), so the cache and the licence line for tier B were not built. Licensed clocks print the obtain-it-yourself instructions, as above.

Chapter 30. 4. Algorithms, the complete operation catalogue

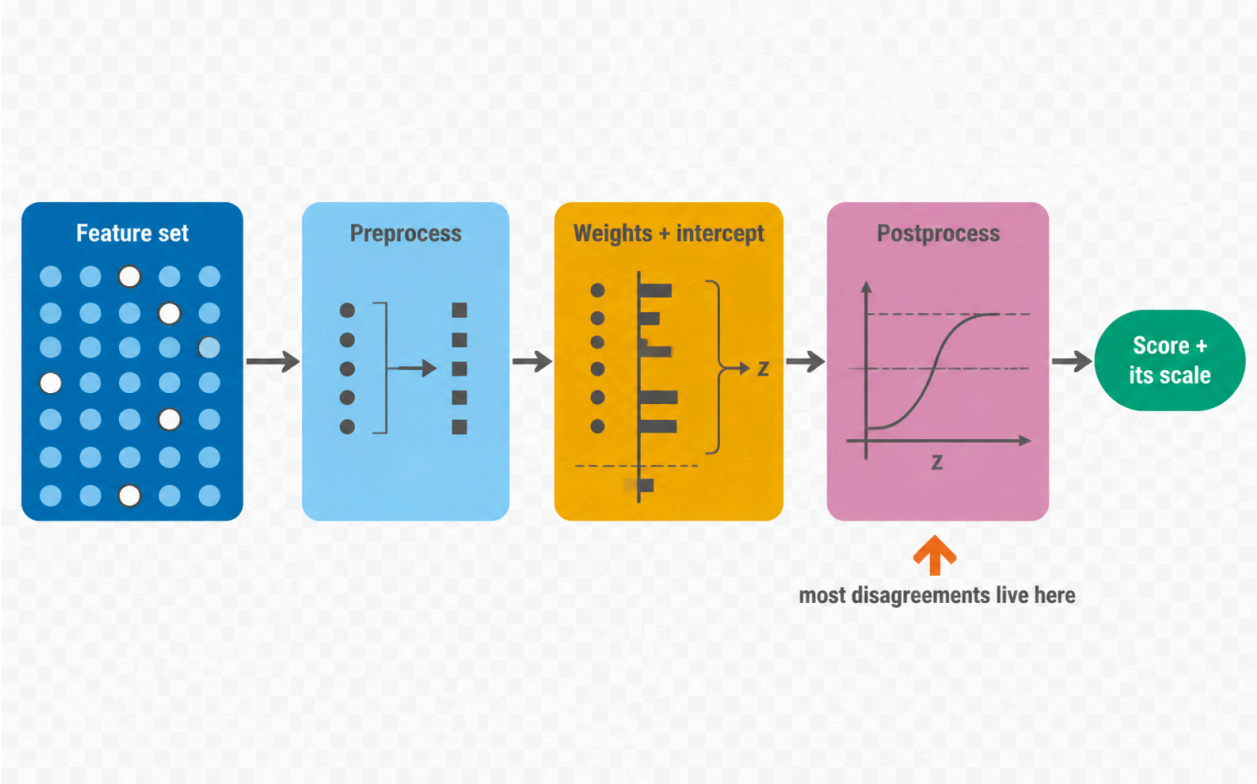


Figure 41: The four parts every clock has, and where implementations disagree

The shape is the field's, not this package's: the reviews that survey it describe the same four parts [59] [60].

Every clock in the registry is expressible as `select features → preprocess ops → architecture forward → postprocess ops`, with ten exceptions that need their own forward pass (§4.3). The ops are the vocabulary; implementing all of them is what makes adding a clock a YAML edit rather than a code change.

The weights are the part everybody agrees on, because they are published. The disagreements between implementations of the same clock are almost all in the last box: whether an anti-log is applied, which of two published transforms was meant, whether an intercept was already folded into the coefficients. That is why the registry stores the postprocess chain as declared data rather than burying it in each model class, and why the clocks whose packaged coefficients contradict their own paper carry the discrepancy as a warning at score time.

v1.0 implements seven of the fifteen preprocess ops below, one dataset-level op and 21 postprocess transforms; 35 of the 69 bundled clocks declare at least one. The rest are not ops here. A bundled clock that needs one does the step in its own model class (AltumAge's per-feature scaler in `neural.py`, the mitotic statistics in `aggregation.py` and `division.py`), and the others wait

on coefficients that are licensed, not yet traced to a primary source, or absent from the registry. An op with no reachable test vector is an op that is wrong in a way nobody will notice. They are kept here because the vocabulary is the point: when a coefficient set does arrive, adding the clock stays a registry edit.

4.1 Preprocess ops

`falconage/models/ops.py`, dispatched through `PREPROCESS` and applied by `apply_chain()`.

They are plain functions of `(x, xp, **params)`, not `nn.Module` subclasses as originally specified. `xp` is the array module the device resolver handed back, numpy or torch, so one implementation serves both backends and the CPU path never imports torch. An `nn.Module` would have forced torch on every install for a 500-CpG dot product that numpy finishes before torch has finished importing. All are batched and vectorised; none contains a Python loop over samples. Source for each is Appendix E.

```
def apply_chain(x, chain: Sequence[dict], table: dict, *, xp) -> Any:
    """Run an ordered op list from a registry entry. Order is load-bearing:
    beta_to_m then scale is not scale then beta_to_m."""
```

Shipped: `beta_to_m`, `m_to_beta`, `scale`, `rank_normalize`, `quantile_normalize`, `binarize`, `simplex_projection`. The full specified catalogue follows. Under more general names the shipped seven cover `identity`, `scale_median`, `scale_mean`, `quantile_normalize_gold`, `quantile_normalize_scale_gold` (as `quantile_normalize` then `scale`), `binarize_row_median` and `median_fill_rank`. `scale_row` is not `scale`, which applies statistics stored with the model; it is the dataset-level op below.

One op runs on the dataset rather than on a clock's slice: `standardise_within_sample`, in `DATASET_PREPROCESS`, which the Zhang et al. 2019 elastic-net and BLUP predictors name in their chains. It needs every probe the data carries, so `LinearClock` runs it before alignment, from two passes over column blocks that never hold the standardised matrix, and refuses when the data holds less than `min_coverage` of the 319,607 probes the models were fitted on. The per-sample mean and SD of a subset are not the array's. It is the `scale_row` row of the table below, with the row statistics taken over the array as the authors' `pred.R` takes them, not over the model's inputs; formulas §5.4 gives what the narrower scope does to the elastic-net estimate.

Op	Computation	Stored params	Clocks
`identity`	`x`	-	default
`fill_nan_with_reference`	`where(isnan(x), ref, x)`	`ref[n_features]`	~90, the `LinearReferenceClock` family
`scale_median`	`(x − median) / std`	`median[]`, `std[]`	AltumAge, CpGPTGrimAge3
`scale_mean`	`(x − mean) / std`	`mean[]`, `std[]`	CpGPTPCGrimAge3
`scale_row`	`(x − rowmean) / rowstd`, guarding zero SD	-	ZhangEN, ZhangBLUP
`quantile_normalize_gold`	rank map onto sorted gold means, then column subset	`gold[20000]`, `idx[]`	DunedinPACE
`quantile_normalize_scale_gold`	as above, then `(x − gold_mean)/gold_std`, then subset	`gold[]`, `gold_std[]`, `idx[]`	Stubbs

Op	Computation	Stored params	Clocks
`tpm_log1p`	`log1p(1e6 · (1000·x/len) / rowsum)`, then subset	`lengths[]`, `idx[]`	OcampoATAC1/2
`binarize_row_median`	mask zeros, threshold at each row's non-zero median	-	BiTAge
`median_fill_rank`	fill NaN with global median, then per-row average ranks	-	Pasta, PastaMouse, REG
`mean_over_present`	mean of entries `!= -1` per row	-	epiTOC1, HypoClock, epiCMIT×2
`quantile_over_present`	q-quantile of entries `!= -1` per row	`q=0.95`	stemTOC, stemTOCvitro
`nan_to_zero`	`nan_to_num(x, 0)`	-	epiTOC2, epiTOC3
`autosomal_zscore`	z-score every probe against that sample's autosomal mean and SD	`autosome_idx[]`	XChrom, YChrom
`expand_with_reference`	scatter the provided features into a wider reference vector	`full_ref[]`, `idx[]`	PastaMouse

Three of these need care.

`quantile_normalize_gold`. The DunedinPACE gold standard is 20,000 probes, of which only 173 are scored; the other 19,827 exist to define the sample's rank distribution (the science notes, §4.8). Dropping them produces a plausible wrong number, so the op raises `FeatureMismatchError` when background coverage falls below a configurable 90%. Vectorised form:

```
sorted_gold = torch.sort(gold)[0]                                  # [20000]
q_idx = torch.linspace(0, len(sorted_gold) - 1, x.shape[1]).long()
order = torch.argsort(x, dim=1)                                    # [B, F]
out = torch.empty_like(x)
out.scatter_(1, order, sorted_gold[q_idx].expand_as(x))
return out[:, self.scoring_idx]
```

One `argsort` and one `scatter_` per batch, replacing pyaging's per-row Python loop (`pyaging/models/_models.py:255`).

`median_fill_rank`. Average ranks with tie handling. pyaging's implementation is a nested `while` loop per row and is the slowest path in that package. The vectorised form uses the double-argsort identity for ordinal ranks and a segment-mean over `unique_consecutive` for ties:

```
order = torch.argsort(x, dim=1)
ordinal = torch.empty_like(order)
ordinal.scatter_(1, order, torch.arange(x.shape[1], device=x.device).expand_as(order))
# tie correction: for each run of equal sorted values, replace with the run's mean rank
```

`autosomal_zscore`. Mean and SD over autosomal probes only, computed per sample with NaN masking, then applied to all probes including the sex chromosomes (the science notes, §3.1).

4.2 Postprocess transforms

`falconage/models/ops.py`, dispatched through `POSTPROCESS`. Scalar in, scalar out, applied elementwise over the batch. Constants come from the science notes §4.20 and §5.

Op	Formula	Params	Clocks
`identity`	`y`	-	most
`anti_log_linear`	`y<0: (1+A)eʸ − 1`; `y≥0: (1+A)y + A`	`adult_age=20`	Horvath2013, SkinAndBlood, PedBE, Han, PCHorvath2013, PCSkinAndBlood, IntrinClock, CorticalClock, Pipek×3
`anti_log_linear_months`	as above with `A=48`, then `/12`	`adult_age=48`	Wu
`anti_logp2`	`eʸ − 2`	-	Mammalian1
`anti_log`	`eʸ`	-	MammalianLifespan
`anti_log_log`	`e^(−e^(−y))`	-	Mammalian2 relative-age step
`mortality_to_phenoage`	`1 − exp(−exp(y)(e^{120λ}−1)/λ)` then `141.50225 + ln(−0.00553·ln(1−m))/0.090165`	`lambda=0.0192`	PhenoAge
`cox_to_years`	`((y−μ_c)/σ_c)·σ_a + μ_a`	four constants	GrimAge, GrimAge2, CpGPTGrimAge3, CpGPTPCGrimAge3
`petkovich_blood`	`((y+1.712)/0.1666)^(1/0.4185) / 30.5`	-	Petkovich
`stubbs_multitissue`	`(exp(0.1207y² + 1.2424y + 2.5440) − 3) · 7/30.5`	-	Stubbs
`divide`	`y / d`	`d`	Meer (30.5), Bohlin (7), EPICGA (7)
`sigmoid`	`1/(1+e^{−y})`	-	McCartney ×6, MammalianFemale, CVDWesterman, ADBahadoSingh
`one_minus`	`1 − y`	-	HypoClock
`add_constant`	`y + c`	`c`	REG, Lin (+12.2169841), Ying ×3
`scale_and_shift`	`y·s + o·s`	`s`, `o`	Pasta, PastaMouse

`mortality_to_phenoage` clamps the mortality score to `[1e-12, 1 − 1e-12]` before the logarithm, following mcp-phenoage-clock (`mcp-phenoage-clock/src/phenoage.ts`), and records `clamped: true` in the result when it fires, an extreme biomarker combination that saturates the Gompertz CDF is information, not an implementation detail.

Intercept convention. ComputAge and biolearn carry the clock intercept inside the transform lambda; pyaging bakes it into the `nn.Linear` bias. FALCONAge normalises on import: the intercept always lives in the linear bias, and `add_constant` is reserved for genuine post-hoc offsets that the source publishes

separately. The registry records `intercept_source: bias | transform` for each imported clock so the normalisation is auditable.

4.3 Model classes

`falconage/models/`

```
class FalconClock(nn.Module, ABC):
    meta: ClockMeta
    features: list[str]
    reference_values: Tensor | None

    def preprocess(self, x: Tensor) -> Tensor      # composed from meta.preprocess
    @abstractmethod
    def core(self, x: Tensor, ctx: ClockContext) -> Tensor
    def postprocess(self, y: Tensor) -> Tensor     # composed from meta.postprocess
    def forward(self, x: Tensor, ctx: ClockContext) -> Tensor
```

`ClockContext` carries what a clock may need beyond its feature matrix, `age`, `sex`, `species`, and the outputs of other clocks it depends on:

```
@dataclass
class ClockContext:
    age:      Tensor | None
    female:   Tensor | None          # 1.0 / 0.0; unknown is an error, never 0.5
    species:  Tensor | None          # one-hot for mammalian clocks
    upstream: dict[str, Tensor]      # e.g. grimage for DNAmFitAge
```

Sex encoding deserves its own note. ComputAge maps unknown sex to 0.5 (`ComputAge/computage/models_library/model.py`), which silently produces a half-male half-female prediction. FALCONAge raises `MissingCovariateError` and offers `sex_unknown="error" | "impute_from_methylation" | "half"`, where the middle option runs `XChrom`/`YChrom` first and the last must be chosen deliberately.

Class	Core computation	Clocks	v1.0
`LinearClock`	`xW + b`	34 bundled	**shipped**
`ClinicalClock`	PhenoAge / KDM / HD on tabular biomarkers	3	**shipped**
`ScaffoldClock`	refuses, and says where the coefficients come from	the 57 `licensed` clocks	**shipped**
`PCLinearClock`	`((x − μ)R)W + b`	PCHorvath2013, PCHannum, PCPhenoAge, PCSkinAndBlood, PCDNAmTL, PCGrimAge	implemented; no weights ship, so `build()` never reaches it
`MultiStageCoxClock`	sub-linear-models → ordered concat with covariates → Cox layer	GrimAge, GrimAge2, CpGPTGrimAge3, CpGPTPCGrimAge3	specified
`NeuralClock`	dense layers from a safetensors file	AltumAge	**shipped**

Class	Core computation	Clocks	v1.0
`AggregationClock`	mean or 95th percentile over the sites present	epiTOC1, stemTOC×2, HypoClock, epiCMIT×2	**shipped**
`DivisionClock`	methylation transmission model, averaged over the sites present	epiTOC2, epiTOC3	**shipped**
`DeconvolutionClock`	per-sample non-negative least squares on a reference table (`projectCellType_CP`) → column select	6-cell IDOL (EPIC, 450K); 12-cell IDOL-Ext with a registered table	**shipped**
`CompositeClock`	multiple sub-clocks combined by a declared rule	SystemsAge ×12, DNAmFitAge	specified
`SpeciesClock`	split CpG block from species one-hot, species-dependent inverse transform	Mammalian ×9	specified
`SexScoreClock`	autosomal z-score → sex-chromosome projection	XChrom, YChrom	specified

`ScaffoldClock` is the class that is not in the original design and turned out to matter most. Every "specified" row above is needed by a clock family whose coefficients cannot be redistributed, so writing the class first would produce code with no real answer to test against. Instead the registry entry, the feature list, the transform chain and the expected shapes all ship, `build()` returns a `ScaffoldClock`, and calling it raises `WeightsUnavailableError` naming the laboratory to ask, the licence that applies, and the open clocks that answer the same question. A silent skip would leave a results table with a column quietly missing.

4.4 The ten special forward passes

Each gets an explicit implementation and its own gold-standard vector.

1. GrimAge / GrimAge2 (`models/cox.py`). Ten (v2) or eight (v1) sub-linear-models on disjoint feature index sets, then a fixed concat order, then the Cox layer, then `cox_to_years`. The order is `[GDF15, B2M, CystatinC, TIMP1, ADM, PAI1, Leptin, PACKYRS, (LogCRP, A1C), Age, Female]` and is load-bearing, the Cox coefficients are positional (`pyaging/models/_models.py`:1742). The registry declares the order; a mismatch between declared and stored sub-model count fails validation.

2. PCGrimAge. Centre 78,464 CpGs, rotate to 1,933 PCs, append `Female` and `Age`, run eight PC-space sub-clocks, then the Cox layer. `requires_fp64: true`.

3. CpGPTPCGrimAge3. Scale → split age from the 30 proxies → rotate 30 → 29 PCs → concat → rescale the PC block with a second mean/std pair → linear → `cox_to_years`. The two-stage scaling is not a mistake in the source; both scalers are stored.

4. DNAmFitAge. Split the batch by sex, run `GaitF/GaitM`, `GripF/GripM` and a shared `VO2Max`, standardise each with the published sex-specific offset and slope, then a sex-specific Klemera–Doubal layer, then recombine in the original row order. The eight constants (the science notes, §4.17) live in the registry, not in code.

5. Mammalian clocks (`models/species.py`). Input is `[CpGs | species_one_hot]` with the one-hot width declared per clock (1,756 for clock-2 variants, 1,707 for clock-3). The species index selects gestation time,

average maturity age and maximum lifespan from a packaged AnAge table, and the inverse transform is clock-specific:

```
clock 1:  age = exp(ŷ) − 2
clock 2:  age = e^(−e^(−ŷ)) · (max_age + gestation) − gestation
clock 3:  m̂ = 5 (gestation / maturity)^0.38
          ŷ ≥ 0 : age = m̂ (maturity + gestation)(ŷ + 1) − gestation
          ŷ < 0 : age = m̂ (maturity + gestation) e^ŷ − gestation
```

A species not in the AnAge table raises rather than falling back to a default.

6. SystemsAge (`models/composite.py`). Seven steps (the science notes, §4.16): centre and rotate 125,175 CpGs → project onto the system vector → 11 per-system linear modules → a separate PC→age model with a quadratic correction then `/12` → concat the 12-vector → optional composite PCA layer → affine calibration `((raw − c₀)/c₁)·c₃ + c₂` → `/12`. Twelve registry entries share one weight file and differ only in `prediction_index` (0–10, or −1 for the composite), so the weights are loaded once and cached across the twelve.

7. Deconvolution (`models/deconvolution.py`). The design was pyaging's: pseudo-inverse of the signature matrix, then a Euclidean projection onto the probability simplex. What shipped is the authors' own, `projectCellType_CP` from FlowSorted.Blood.EPIC with `nonnegative = TRUE, lessThanOne = FALSE`: per sample, non-negative least squares over the CpGs it observes (`scipy.optimize.nnls`), rounded to four decimals, with no sum-to-one constraint. The simplex projection is a different estimator, and on real blood it moved every cell type by about 0.006 (methods §4.10 gives the comparison).

```
table = registry.deconvolution_table(cid, data.platform, present=data.X.columns)
al = align_present(data, list(table.index))            # absent CpGs left out, not filled
props = project(al.matrix, table.to_numpy())           # samples x cell types
return props[:, list(table.columns).index(cell_type)]
```

The six 6-cell entries share one table per array (EPIC, 450K legacy) and differ by `deconvolution.cell_type` in the registry; the twelve 12-cell entries read a table the user registers under its licence. `fa.cell_composition(result)` returns every proportion scored in a run as one frame.

8. epiTOC2 / epiTOC3. `tnsc = (2/k) Σ_c (β_c − β₀_c) / (δ_c(1 − β₀_c))`, with per-CpG `δ` and `β₀` vectors stored as weights and a guard against `δ(1−β₀) == 0`.

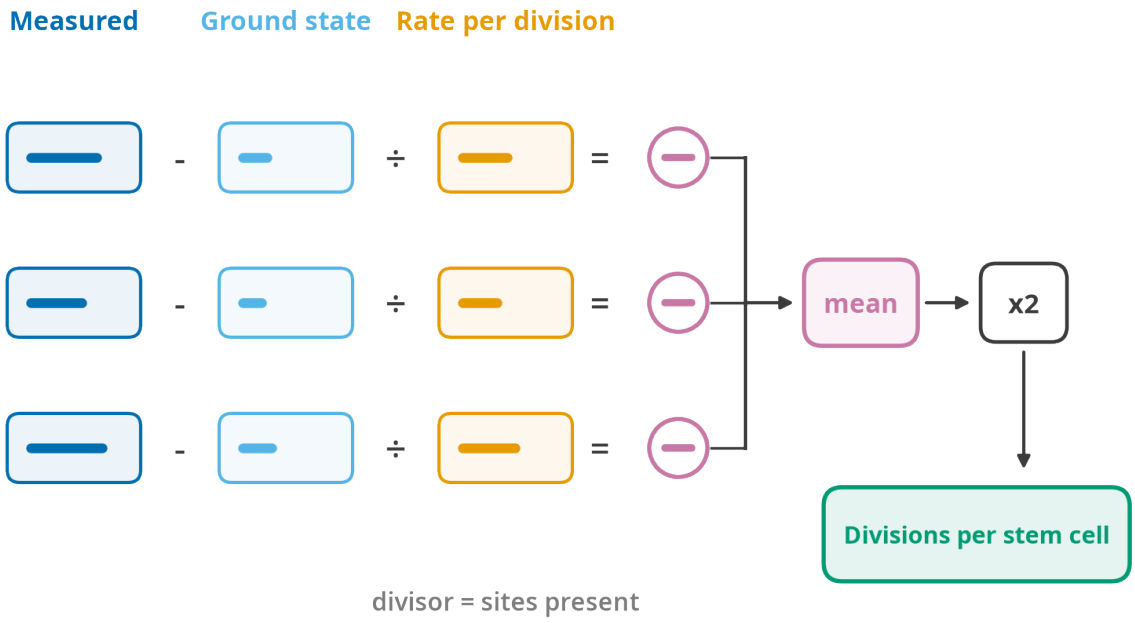


Figure 42: The division model inverted per site, then averaged

This is not a weighted sum with a different name. Each site is inverted on its own: the drift observed since the ground state is divided by that site's drift per division, which turns a methylation fraction into a division count. Only then are the per-site estimates averaged, and `k` is the number of sites actually present rather than the number the clock declares, so a missing site drops out of both the sum and the divisor instead of contributing a zero. The factor of two converts from the per-allele rate to divisions per stem cell.

9. XChrom / YChrom. Autosomal z-score, then `Σ (z_sex − mean) · coef` over the sex-chromosome probe subset. Exposed both as clocks and through `estimate_sex(data) -> DataFrame[x_score, y_score, predicted_sex, confidence]`, since sex estimation is a preprocessing need as much as a clock.

10. PastaMouse. Expand the mouse feature vector into the full 8,113-gene human space using reference values for human-only genes, then run the shared Pasta forward.

4.5 Clinical chemistry clocks

`falconage/models/clinical.py`. These are not methylation clocks and do not share the tensor path; they operate on a tabular biomarker frame and are implemented in a way that mirrors `BioAge/R/` so the numbers can be checked against it directly.

PhenoAge. Two variants, selected explicitly.

```
def phenoage(
    data: pd.DataFrame,
    units: Literal["si", "conventional"],       # no default
    variant: Literal["levine2018", "liu2019"] = "levine2018",
```

```
) -> pd.DataFrame                                    # phenoage, mortality_score,
phenoage_advance, clamped
```

The two coefficient sets and their Gompertz constants are in the science notes §5.1 and are stored in the registry, not in code. `units` has no default because the SI and conventional parameterisations are not interconvertible by scaling the inputs, the coefficients differ too. The `0.090165` constant is the corrected one; BioAge's April 2026 commit fixed a widespread `0.09165` and the test suite pins the corrected value.

KDM. Fit-then-project, with the fit reusable:

```
def fit_kdm(reference: pd.DataFrame, markers: list[str], age_col: str = "age",
            s_ba2: float | None = None) -> KDMReference
def kdm(df: pd.DataFrame, ref: KDMReference, age_col: str = "age",
        max_missing: int | None = 2) -> pd.Series
```

The math is the science notes §4.4, transcribed from `BioAge/R/kdm_calc.R` including the details that matter: per-marker regressions on the rows carrying each marker, `r_char` from the ratio of two weighted sums, `s_r` from the age range and biomarker count, `s_BA²` estimated on the reference and kept with the fit, and the rule that a sample with more than two missing biomarkers returns NA rather than a partial score. Fitting by sex is the caller's choice: fit one reference per sex, as BioAge does. The result reproduces BioAge's shipped `kdm0` to 0.0007 years (§4.4), and `kdm_bioage()` scores on that scale from a packaged NHANES III fit.

Homeostatic dysregulation. Mahalanobis distance from a young reference cohort:

```
def hd_fit(reference: pd.DataFrame, biomarkers: list[str]) -> HDFit
def hd(data, biomarkers, fit=None, log=True, by_sex=True) -> pd.DataFrame
```

`hd_fit` standardises the reference by its own mean and SD, then stores the covariance. It checks the condition number and raises when the covariance is near-singular. BioAge only warns on a single-row reference, which is the degenerate case rather than the common one. The reference cohort must be young, non-pregnant and within the clinical windows in the science notes §3.5; the packaged NHANES III fit satisfies this and is the default.

4.6 Op composition and validation

At load time the registry entry's `preprocess` and `postprocess` lists are compiled into an `nn.Sequential`, and the composition is validated:

- an op requiring stored params must find them in the weight file
- `quantile_normalize_gold` must be followed by a column subset
- a clock with `scale_type: age_years` may not carry `sigmoid` as its final transform
- the declared `n_features` must match the weight tensor's leading dimension

These checks run in `registry/validate.py` in CI and again at load time, so a malformed registry entry fails before it produces a number.

4.7 Scaffold-only clocks

Forty clocks are `licensed` (tier C): their architecture is described in a published paper and is implemented here in full, but their coefficients are not ours to distribute. For these FALCONAge ships **the scaffold and nothing else** - the model class, the feature list, the preprocess and postprocess chain, the shape and dtype of every expected tensor, and a loader that accepts a coefficient file the user supplies.

Everything about these clocks except the numbers is written from scratch, and all of it is testable. What cannot be done is invent the numbers.

Which they are, and why

Family	n	Why permission is needed
GrimAge, GrimAge2 and its 10 sub-clocks, CpGPTGrimAge3, CpGPTPC-GrimAge3	14	Coefficients were never published in the papers. UCLA licenses the clock; what circulates in other packages has no traceable primary source.
SystemsAge and its 11 system variants	12	Flagged research-use-only by the authors. Coefficients are on a lab Google Drive, published in Nature Aging 2025 but not under a redistribution licence.
PCGrimAge	1	A principal-component reparameterisation of GrimAge; inherits its restriction.
DunedinPACE	1	Research-use-only. Distributed through the authors' R package under terms the user accepts on install.

The GrimAge family is the sharp case. GrimAge2 is the most-used mortality clock in the field, and its coefficient set has no public primary source at all, pyaging re-hosts a copy whose origin its own audit could not establish. FALCONAge will not launder that. The scaffold is complete and correct; the numbers must come from UCLA or from the user's own licensed copy.

What ships

```yaml
# registry/clocks/grimage2.yaml  (excerpt)
availability: C
coefficient_source:
  tier: T5
  url: null
  primary_source_traced: false
  licence: "Research use only. Commercial licensing via UCLA TDG."
  obtain: >
    GrimAge2 coefficients are not publicly redistributable. Request access from
    the Horvath laboratory, or use the authors' online calculator at
    dnamage.clockfoundation.org. Once you hold a coefficient file, register it
    with falconage.registry.register_local_weights("grimage2", path).
  contact: "https://dnamage.clockfoundation.org"
scaffold:
  complete: true
  expects:
```

```
    - {name: "sub_models", shape: "10 x (n_features_i,)", dtype: float64}
    - {name: "cox", shape: "(12,)", dtype: float64}
  validated_against: "published worked example, Lu 2022 Table 2"
```

Supplying coefficients

```python
import falconage as fa

fa.registry.register_local_weights(
    "grimage2",
    "~/licensed/grimage2_coefficients.csv",
)
res = fa.score(data, clocks=["grimage2"])
```

```r
register_local_weights("grimage2", "~/licensed/grimage2_coefficients.csv")
res <- score(data, clocks = "grimage2")
```

`register_local_weights` parses the file, checks it against the scaffold's declared shapes and feature count, warns on any feature the scaffold does not expect, converts to safetensors, and caches it. A file that does not match the published architecture is rejected with the mismatch named, which also makes this the mechanism for validating a coefficient set someone hands you.

The registration is recorded in the run manifest as `weights_source: user_supplied, sha256: ...`, so a result computed from a licensed copy is distinguishable from one computed from a redistributed set.

Behaviour without coefficients

Scoring a licensed clock with nothing registered raises rather than silently skipping:

```
WeightsUnavailableError: grimage2 is a scaffold-only clock.

  Its architecture is implemented and tested, but its coefficients are
  not distributed with FALCONAge.

  Why: Research use only; coefficients were never published in the paper and the copies
  circulating in other packages have no traceable primary source.

  Obtain them from: https://dnamage.clockfoundation.org (Horvath laboratory); commercial
  licensing via UCLA TDG
  Then: falconage.registry.load().register_local_weights('grimage2', <path>)

  The question it answers: Who is at risk of dying sooner, or is frailer?
  Bundled clocks that answer it too:
    dnamphenoage     dna methylation     513 features, bundled
    hrsinchphenoage  dna methylation     959 features, bundled
    zhangmortality   dna methylation     10 features, bundled
    hd               clinical chemistry  markers chosen by the user, bundled
```

```
    kdm              clinical chemistry  markers chosen by the user, bundled
    phenoage         clinical chemistry  10 features, bundled
```

Naming the open alternatives matters. A user who wanted a mortality clock and hit a wall should leave with one that works, not with an error. The list is derived, not kept by hand: it is every bundled clock that `falconage.registry.questions` routes to the same question as the licensed one, the same routing the choosing a clock page uses. A hand-kept list offered Horvath and Hannum, both fitted to chronological age, in place of GrimAge. Where no bundled clock answers the question, as for SystemsAge, the message says so. The registry's `coefficient_source.alternatives` follow as "Related entries", each with its availability, so a sibling that is itself licensed is never presented as something to run instead.

`clocks="compatible"` excludes unregistered licensed clocks by default and reports the exclusion. `clocks="all"` includes them and fails loudly. `falconage clocks list --tier licensed` shows the whole set with its obtain instructions.

Testing a clock whose weights you do not have

The scaffold is still fully tested, three ways:

1. **Synthetic weights.** Generate a coefficient set of the right shape; run it, and assert the forward pass produces the algebraically expected value. This tests the architecture, the sub-model ordering, the concat order, the Cox transform constants, without needing the real numbers. A GrimAge2 scaffold given a coefficient vector of all zeros except one must return the value that hand-computation says it should.
2. **Published worked examples.** Several papers report a reference sample's inputs and its resulting age. Where one exists it is the gold-standard vector, and it is a better oracle than another package's output because it comes from the authors.
3. **Shape and invariant contracts.** Feature count, dtype, sub-model count, concat order, output range and scale type are all asserted from the registry entry regardless of which weights are loaded.

A licensed clock without a published worked example carries `gold_standard.source: "synthetic weights, architecture only"` and the docs say so on its page. It is honest about being an untested-against-truth implementation of a correctly-described architecture.

Could they be reimplemented from scratch instead?

For the architecture, yes, and that is what §4.4 already specifies, the GrimAge two-stage structure, the PC rotation and SystemsAge's seven-step pass are all described well enough in their papers to write independently, and are written here.

For the coefficients, no. They are the output of an elastic net fit to cohorts that are not public. Framingham, the Women's Health Initiative, the Dunedin Study. Refitting the same architecture on different data produces a different clock that must not carry the same name. That path is real and is scoped as a v1.5 milestone (§13), where a FALCONAge-native mortality clock trained on the open ComputAgeBench split would be a new contribution rather than a substitute.

Chapter 31. Download, preprocessing, scoring and reports

5. Download module

5.1 Contract

One verb, one accession, files plus a normalised sample table.

```
def download(
    accession: str | Sequence[str],
    dest: Path | None = None,          # None -> the managed cache
    what: Literal["raw", "processed", "metadata", "auto"] = "auto",
    credentials: Path | None = None,
    dry_run: bool = False,
    max_size_gb: float | None = None,
    resume: bool = True,
    n_jobs: int = 4,
) -> DownloadResult
```

```
@dataclass
class DownloadResult:
    accession:  str
    source:     str                # "geo" | "arrayexpress" | ...
    files:      list[FileRecord]   # path, role, sha256, size, url
    samples:    pd.DataFrame       # the normalised sample table
    raw_metadata: dict             # whatever the source gave, unmodified
    manifest_path: Path
```

`what="auto"` prefers raw over processed when raw exists, because a beta matrix someone else normalised is not reproducible. `dry_run=True` returns the same object with `files` populated and sizes filled but nothing transferred, which is how a user finds out that a series is 12 GB before committing to it.

5.2 Source dispatch

`falconage/download/resolve.py` recognises the accession by pattern and returns the right `Downloader`. Unrecognised input raises with the list of recognised patterns rather than guessing.

Pattern	Source	Backend	Ships
`GSE\d+`	GEO series	GEOparse plus supplementary-file fetch; `methylprep` when the series has IDATs	now
`GSM\d+`	GEO sample	same, single sample	now
`GPL\d+`	GEO platform	manifest only	now

Pattern	Source	Backend	Ships
`E-\w+-\d+`	ArrayExpress / BioStudies	BioStudies REST v1	now
`SR[PRXS]\d+`, `PRJNA\d+`, `ERP\d+`, `DRP\d+`	SRA / ENA	`pysradb` for metadata, ENA FTP or `fasterq-dump` for reads	now
`10\.\d{4,}/.*`	Zenodo, Figshare, Dryad	DOI resolution then the host REST API	now
numeric with `source="zenodo"`	Zenodo record	Zenodo REST	now
`hf://owner/repo[/path]`	Hugging Face	`huggingface_hub`	now
`TCGA-\w+`, GDC UUID	GDC	GDC files/data API	now
`PXD\d+`	PRIDE	PRIDE REST	later
`MTBLS\d+`	MetaboLights	MetaboLights REST	later
`ST\d{6}`	Metabolomics Workbench	MW REST	later
`syn\d+`	Synapse	`synapseclient`, credentialed	later
`EGAD\d+`, `EGAS\d+`	EGA	documented, credentialed	later
`phs\d+`	dbGaP	documented, not automated	-

5.3 GEO, in detail

GEO is where most of the traffic goes, and it has three shapes that need different handling.

```
class GEODownloader(Downloader):
    def probe(self, gse: str) -> GEOSeriesInfo
    def fetch(self, gse: str, what: str, dest: Path) -> DownloadResult
```

`probe` reads the MINiML metadata and reports what is actually available before anything is transferred:

```
@dataclass
class GEOSeriesInfo:
    gse: str
    title: str
    n_samples: int
    platforms: list[str]              # GPL ids, mapped to falconage platform names
    has_idats: bool
    has_series_matrix: bool
    supplementary_types: list[str]
    total_size_bytes: int
    characteristics_fields: list[str] # the free-text keys that might hold age and sex
```

The three shapes:

1. **IDATs present** - roughly 40% of methylation series. Download the `_Grn.idat.gz` and `_Red.idat.gz` pairs plus the platform manifest, and hand off to the preprocess module. This is the only path that yields a reproducible beta matrix.

2. **Series matrix only.** Parse the `!Sample_` header block for metadata and the tab-delimited body for the value matrix. The values are whatever the submitter normalised, which is recorded in the manifest as `normalisation: submitter-reported` and carried into the result, so a downstream comparison across series knows the matrices are not commensurable.
3. **Supplementary processed files.** A CSV, TSV or RData of betas with no fixed convention. `read_betas` sniffs orientation, delimiter, and whether values are betas in `[0,1]` or M-values, and asks rather than guesses when it cannot tell.

5.4 Metadata normalisation

`falconage/download/metadata.py`. The hardest part of the download module is not transfer: it is that the GEO `characteristics_ch1` field is free text and every submitter writes age differently.

```
def normalise_samples(raw: pd.DataFrame, source: str,
                      mapping: dict[str, str] | None = None) -> pd.DataFrame
```

Target schema, which is what the rest of the package consumes:

Column	Type	Notes
`sample_id`	str	index; GSM id for GEO
`title`	str	submitter sample title
`age`	float	years; months and weeks converted, source unit recorded
`sex`	category	`male`, `female`, `unknown`
`tissue`	str	controlled vocabulary where mappable, raw string otherwise
`condition`	str	`HC` for controls where identifiable
`platform`	str	falconage platform name
`subject_id`	str	for repeated measures
`timepoint`	float	for longitudinal series
`raw_characteristics`	str	the original text, always kept

Extraction is rule-based and conservative: regex patterns for the common age spellings (`age: 54`, `age (years): 54`, `agey: 54`, `54 yr`), a sex vocabulary, and a tissue lexicon. Every extracted value keeps its source string in `raw_characteristics`, and the fraction of samples where extraction succeeded is reported. When it falls below a threshold the module says so loudly rather than returning a half-empty age column:

```
WARNING  age extracted for 41/120 samples (34%) from field 'characteristics_ch1'
         unparsed examples: "age at diagnosis: 54y", "AGE=54", "54 years old"
         supply mapping={"age": "characteristics_ch1"} with a custom parser, or
         edit the returned sample table directly
```

No LLM, no fuzzy matching that might silently mis-assign. A wrong age is worse than a missing one because age acceleration is computed against it.

5.5 Transfer, cache, credentials

`transfer.py` - HTTP range requests for resume, parallel connections capped at `n_jobs`, SHA-256 verification against the published checksum where one exists and against a recorded one on re-download, exponential backoff on 429 and 5xx, and a per-source rate limit (NCBI allows 3 requests per second without an API key, 10 with one).

`cache.py` - content-addressed under `FALCONAGE_CACHE`, default `~/.cache/falconage`, laid out as `<source>/<accession>/<sha256-prefix>/<filename>` with a `manifest.json` per accession.

```
falconage cache ls              # accession, source, size, last used
falconage cache size
falconage cache rm GSE40279
falconage cache verify          # re-check every stored checksum
falconage cache gc --older-than 90d --keep-under 20GB
```

`credentialed.py` covers the sources that need authorisation. Synapse and EGA get working implementations behind an explicit credentials file; dbGaP and UK Biobank get documentation and a clear error explaining that access is granted per project and cannot be automated:

```
raise CredentialsRequiredError(
    "dbGaP accession phs000424 requires an approved data access request. "
    "FALCONAge cannot download it for you.",
    remedy="Download via the dbGaP repository key and SRA Toolkit, then point "
           "falconage preprocess at the resulting files.",
)
```

Credentials come from a file or the environment, never from a function argument that would land in a shell history or a notebook.

5.6 CLI

```
falconage download GSE40279 --dest data/ --what raw
falconage download GSE40279 --dry-run              # sizes and file list only
falconage download GSE40279 GSE42861 --n-jobs 8
falconage download 10.5281/zenodo.18763485
```

As shipped, `--dry-run` is what answers "what is in this series?": it prints every URL and byte count and transfers nothing. The specification also had a separate `probe` verb for that; it was not built, because it would have been the same code behind a second name.

6. Preprocess module

6.1 Contract

Raw or public data in, a `FalconData` the scoring module accepts out. One entry point per modality, all converging on samples by features with exact feature identifiers and declared units.

```
def preprocess_methylation(
    source: Path | pd.DataFrame | DownloadResult,
```

```
    platform: str | None = None,        # None -> auto-detect
    pipeline: str = "noob_bmiq",        # "noob_bmiq" | "sesame" | "noob" | "raw"
    detection_p: float = 0.01,
    mask_probes: bool = True,           # cross-reactive, SNP, sex-chromosome
    aggregate_epicv2: bool = True,
    imputation: str = "none",           # dataset-level; clock-level is separate
    sample_table: pd.DataFrame | None = None,
    device: str = "auto",
) -> FalconData
```

Analogous `preprocess_clinical`, plus `preprocess_expression` and `preprocess_proteomics`.

6.2 Methylation

IDAT to beta. The chain is the science notes section 3.1:

```
IDAT pair (Grn, Red)
  -> parse binary IDAT, match to platform manifest
  -> out-of-band background correction and dye bias (noob)
  -> probe-type bias correction (BMIQ), or SeSAMe as a single-step alternative
  -> detection p-value filter
  -> cross-reactive / SNP / sex-chromosome probe masking
  -> beta = M / (M + U + 100)
  -> beta matrix, samples by probes
```

Implementation route: `methylprep` for IDAT parsing and noob, a native BMIQ port (beta-mixture quantile normalisation is around 150 lines, and having it in-tree avoids an R dependency inside the Python core), and SeSAMe through `rpy2` only when the user asks for it. The pipeline choice is recorded in the manifest because it changes the answer, the science notes section 3.1 notes that `noob + BMIQ` and SeSAMe both perform well but are not interchangeable, and funnorm is preferred when global methylation differs between groups.

Platform detection.

```
def detect_platform(feature_ids: Sequence[str]) -> PlatformInfo
```

Decides from probe-ID pattern and count: EPIC v2 if any ID carries a `_TC21`-style suffix, then by overlap against packaged manifest probe sets for 27K, 450K, EPIC v1, EPIC v2, MSA, MammalMethylChip40 and MammalMethylChip320. Returns the best match with its overlap fraction, and raises `PlatformMismatchError` when the best overlap is under 50% rather than proceeding with the wrong manifest.

EPIC v2 aggregation. Split each column name on `_`, group by the CpG prefix, average replicates. Mandatory, the science notes section 16 records that skipping it means every clock sees zero overlapping features. When `detect_platform` says EPIC v2 and `aggregate_epicv2=False`, the module raises rather than producing a silently empty result.

Cross-platform harmonisation. A 450K clock applied to EPIC data uses the probes present on both. The module reports the intersection size per clock before scoring, and records `platform_native` alongside `platform_applied`, so a result computed on 340 of Horvath's 353 probes says so.

Series matrix and beta CSV paths run the same masking, aggregation and detection steps, minus the normalisation the submitter already applied, with `normalisation: submitter-reported` in the manifest.

6.3 Clinical chemistry

```
def preprocess_clinical(
    data: pd.DataFrame,
    units: dict[str, str],              # required, see core/units.py
    validate_ranges: bool = True,
    range_action: str = "warn",         # "warn" | "na" | "error"
) -> FalconData
```

Three steps: map column names to canonical marker names through a synonym table, so `ALB`, `albumin` and `Albumin (g/dL)` all resolve to `albumin`; convert every marker to the canonical unit the clocks expect; validate against physiological ranges. The ranges are the BioAge clinically-healthy windows (the science notes section 3.5) for the reference-cohort case, and wider physiological bounds for the general case, a CRP of 40 mg/L is outside the healthy window but is a real measurement, so it warns rather than nulls by default.

6.4 Imputation, kept in three distinct places

the science notes section 6.3 separates three stages that are routinely conflated. Keeping them separate is what makes the missingness report meaningful.

Stage	Where	Options	Recorded as
Dataset-level	`preprocess.impute`	`none`, `mean`, `median`, `constant`, `knn`	`imputation.dataset`
Clock-level	`models.linear.align`	the clock's `reference_values` for an absent feature; the mean of its present features where there is none; a present feature's own cohort mean for its missing values	`coverage[clock]`: `n_from_reference`, `n_pooled`, `reference`
Model-internal	inside the clock	the model's own stored statistic	`imputation.model`

Dataset-level KNN is the expensive one and is GPU-accelerated when a device is available: `cuml.KNNImputer` when RAPIDS is installed, a chunked CuPy distance otherwise, sklearn on CPU.

Clock-level filling uses `reference_values` from the registry, built by `python/tools/build_references.py`: the authors' published per-CpG values for Horvath 2013, DunedinPoAm38, CorticalClock and AltumAge, and a healthy adult whole-blood mean from the test corpus for 22 other blood clocks. Zero is never a fill: in beta space it means fully unmethylated. A feature the reference lacks, or any absent feature of a clock with no reference, takes the mean of the clock's present features, and the run warns (category `imputation`) with the count and the share of the model's |coefficient| involved. Below `min_coverage`, default 0.8, of the features or of the coefficient mass, scoring raises `FeatureCoverageError` rather than returning a number nobody should use.

6.5 QC report

```
@dataclass
class QCReport:
    n_samples: int
    n_features: int
    platform: PlatformInfo
    missingness_overall: float
    missingness_per_sample: pd.Series
    beta_distribution: dict            # quantiles, bimodality statistic
    failed_detection: pd.Series        # per sample
    predicted_sex: pd.DataFrame        # XChrom/YChrom, against declared sex
    sex_mismatches: list[str]
    age_available: float               # fraction of samples with age
    clock_coverage: pd.DataFrame       # clock, n_present, n_total, pct
    warnings: list[str]
```

`clock_coverage` prints before scoring, not after. A user who can see that GrimAge2 will run on 61% of its features gets to decide whether to proceed; a user who discovers it afterwards has already written the number down.

6.6 CLI

```
falconage preprocess methylation data/GSE40279/ --out prepared.h5ad --pipeline noob_bmiq
falconage preprocess methylation betas.csv --platform illumina_450k --out prepared.h5ad
falconage preprocess clinical labs.csv --units units.yaml --out prepared.h5ad
```

The specification had a separate `qc` verb writing its own HTML. Not built: `preprocess` already prints the QC summary, and `report` embeds the full set of QC figures, so a third path to the same numbers was a third thing to keep in step.

7. Scoring, analysis, plots, report

7.1 score()

```
def score(
    data: FalconData,
    clocks: str | Sequence[str] | Literal["all", "compatible"] = "compatible",
    device: str = "auto",
    dtype: str = "float64",
    batch_size: int | None = None,
    max_missing: float = 0.30,
    imputation: str = "reference",
    sex_unknown: str = "error",
    keep_features: bool = False,
    progress: bool = True,
) -> FalconResult
```

`clocks="compatible"` is the default and calls `registry.compatible_with(data)`, which is the behaviour a first-time user wants. `"all"` runs everything and warns about the ones that will be mostly imputed.

Loop, per clock:

1. Resolve metadata; warn on `research_only` and on any `known_discrepancies`.
2. Check required covariates; raise `MissingCovariateError` naming the clock and the covariate.
3. Align features into `obsm[f"X_{clock}"]`, filling absences from `reference_values`. Record `percent_na` and the missing-feature list. Raise at 100% missing, or above `max_missing`.
4. Promote dtype if `requires_fp64`.
5. Batch through the model under `torch.inference_mode()`.
6. Write predictions to `obs[clock]` and metadata to `uns[f"{clock}_metadata"]`.
7. Range-check against `plausible_range`; flag, never clip.
8. Free the `obsm` slice unless `keep_features`, then `gc.collect()` and `torch.cuda.empty_cache()`.

7.2 FalconResult

```
class FalconResult:
    data: FalconData
    manifest: RunManifest

    def table(self, long: bool = True) -> pd.DataFrame
    def wide(self) -> pd.DataFrame
    def acceleration(self, method: str = "both",
                     within: str | None = None) -> pd.DataFrame
    def qc(self) -> pd.DataFrame
    def to_csv(self, path) -> None
    def to_h5ad(self, path) -> None
```

The long table carries every provenance field from the science notes section 8.1:

```
sample_id, clock, clock_version, registry_version, value, unit, scale_type,
chronological_age, aa_absolute, aa_relative, percent_features_missing,
n_features_missing, imputation, preprocess_ops, postprocess_ops,
species, tissue, platform_native, platform_applied, research_only,
flags, citation, doi
```

`flags` is a comma-separated list drawn from a fixed vocabulary: `out_of_range`, `high_missingness`, `clamped`, `dtype_promoted`, `platform_mismatch`, `known_discrepancy`, `extrapolated_age_range`, `sex_imputed`.

7.3 Age acceleration

```
def acceleration(result, method="both", within=None, covariates=None) -> pd.DataFrame
```

Three conventions (the science notes section 1.1), all guarded by `scale_type`:

- `absolute` - `value − chronological_age`. Legal only for `scale_type` in `{age_years, age_months, age_days, age_hours, gestational_weeks}`. Attempting it on `rate`, `log_hazard`, `probability` or `proportion` raises `IncompatibleScaleError`: DunedinPACE is not an age, and subtracting CA from it is a category error.

- `relative` - residual of `lm(value ~ age)`, optionally with covariates. Legal for any continuous scale. This is the convention that removes the regression-dilution bias every penalised clock carries.
- `within` - residual against a group mean, for the single-cell and spatial case where the reference is the animal rather than the calendar, following SpatialAgingClock.

`method="both"` returns absolute and relative side by side, and omits absolute for scales where it is illegal rather than emitting NaN.

7.4 Survival, association, reliability, benchmark

```
# survival.py
def cox(result, time, status, clocks=None, covariates=("age", "sex"),
        standardise_within=("sex",)) -> pd.DataFrame     # HR, CI, p, n, events
def c_index(result, time, status, clocks=None) -> pd.DataFrame
def time_dependent_auc(result, time, status, horizons=(5, 10, 15)) -> pd.DataFrame

# association.py
def associate(result, outcome, family="gaussian", covariates=("age", "sex")) ->
pd.DataFrame
def interactions(result, outcome,
                 modifiers=("bmi", "sex", "smoking", "education")) -> pd.DataFrame

# reliability.py
def icc(result, subject_col, replicate_col, kind="ICC2_1") -> pd.DataFrame
def pool_icc(iccs: Sequence[pd.DataFrame]) -> pd.DataFrame        # Fisher-Z

# benchmark.py
def aa2_test(result, condition_col, control="HC", assumption="normal") -> pd.DataFrame
def aa1_test(result, condition_col, control="HC", assumption="normal") -> pd.DataFrame
def ca_accuracy(result, control="HC") -> pd.DataFrame    # MedAE, MAE, RMSE, r, rho,
slope, MedE
def total_score(aa2, aa1, accuracy) -> pd.DataFrame
def run_benchmark(result, condition_col, ...) -> BenchmarkResult
```

`cox` standardises the acceleration within sex before fitting, so the hazard ratio is per sex-specific SD - the BioAge convention (BioAge table_surv.R), which is what makes effect sizes comparable across clocks with different variances.

The benchmark functions implement the ComputAgeBench methodology (the science notes section 11.1):

```
AA2:   one-sided Welch t-test (normal) or Mann-Whitney U (none), AAC against HC
AA1:   one-sided one-sample t-test or Wilcoxon signed-rank, AAC against 0
FDR:   Benjamini-Hochberg per model across datasets, threshold 0.05
total = aa2_passed + aa1_passed * (1 - max(0, MedE) / MedAE)
```

The bias penalty in the total is what stops a clock that simply over-predicts everyone from sweeping AA1. `total_score` returns the components alongside the total, and the report never prints the total on its own.

`run_benchmark(..., dataset="computage")` fetches the 65-study benchmark set from Hugging Face and runs the whole thing, so a user can place their own clock on the published leaderboard.

7.5 Agreement

```
def clock_correlation(result, method="spearman", cluster=True) -> pd.DataFrame
def clock_embedding(result, method="umap", n_neighbors=15) -> pd.DataFrame
def concordance(result, reference_clock: str) -> pd.DataFrame
```

`clock_correlation` with `cluster=True` returns the matrix plus a dendrogram order, which is the pyaging Figure 1a analysis: telomere predictors cluster together, and chronological predictors separate from mortality predictors.

7.6 Plots

Every plotting function returns the figure and has a parallel `*_data()` function returning the frame it plotted, so the R side renders the same numbers with ggplot2 rather than passing matplotlib figures across the bridge. The fourteen from the science notes section 10:

Function	Figure
`ba_vs_ca`	BA against CA, faceted by clock, coloured by sex, identity line, r annotated
`clock_corr`	correlation matrix: scatter below the diagonal, r above, labels on it
`clock_dendrogram`	hierarchical clustering of clocks
`clock_umap`	samples embedded in clock space
`trajectory`	prediction across a timepoint variable, one line per clock
`class_bench`	boolean clock by dataset:condition:test grid with per-class column groups
`medae_bar`	median absolute error per clock, healthy controls only
`bias_bar`	median error per clock, symmetric x-limits
`forest`	hazard ratios per clock per outcome
`acceleration_density`	violin or density of age acceleration by group
`system_profile`	radar or heatmap over SystemsAge systems and organAging organs
`qc_density`	beta distribution per sample at each preprocessing stage
`calibration`	predicted against observed, for probability outputs
`missingness`	percent of features absent per clock, drawn before the results
`kaplan_meier`	survival of the fastest-ageing tail against the slowest, with the log-rank p
`volcano`	effect size against evidence, thresholded at the BH cut rather than a raw p

The last two close the gap against the figure conventions in current papers. Both estimators are written out rather than imported: the product-limit form and the two-sample log-rank statistic are about forty lines together, and `survival` and `lifelines` are heavier dependencies than that deserves. The R side calls the Python core for the log-rank p and redraws the curve in ggplot2, so the statistic cannot differ between the languages while the figure still carries the R theme.

`volcano` draws the Benjamini-Hochberg threshold, taken from the `q` column that `associate()` already returns rather than recomputed. Drawing a raw p-value cut is the usual error and, across thousands of tests, calls noise significant.

Conventions enforced in `theme.py`: the identity line is always drawn on BA-against-CA; rate clocks never share an axis with age clocks; facets are ordered by clock generation rather than alphabetically; every panel is annotated with n, r and missingness; colour encodes group and shape encodes platform, with one palette across a whole report.

7.7 Report

```
def report(result, path: Path, sections="all", embed_data=True) -> Path
```

One self-contained HTML file, inlined CSS, base64 figures, no external requests, carrying the run manifest, the QC block, the results table, the standard figures, and the caveats that apply to the clocks that actually ran. `embed_data=True` attaches the results CSV as a data URI so the report is a complete record.

7.8 CLI

> **The command lines in this section, in 5.5, 5.6 and 6.6, are the specified interface and several of them do not parse against the shipped one.** The verbs are right; the argument shapes are not. `score`, `bench` and `preprocess` take `--input` rather than a positional, `preprocess` has no `methylation` or `clinical` sub-verb, `clocks` requires an action (`list`, `info`, `cite`) before any filter, `power` takes `--clock`, and `cache` has `ls` and `rm` but not `size`, `verify` or `gc`.
>
> Left as written because this is the design record and the spec is what it records. The shipped interface is in Getting started and in `--help`, which is generated from the parser and cannot drift. `docs/check_api_docs.py` parses every command in every other document against that parser and skips this page for exactly this reason.

```
falconage score prepared.h5ad --clocks horvath2013,grimage2,dunedinpace --out results/
falconage score prepared.h5ad --clocks compatible --device cuda --dtype float32
falconage bench results/ --condition-col condition --control HC
falconage report results/ --out report.html
falconage clocks --tier bundled --search mortality
falconage power horvath2013 --effect 0.5 --sd 6
falconage consensus results/ --group condition
```

Two departures from the specification. `analyse --cox` was not built as a verb: survival and association are two functions with several arguments each, and a command line is a poor place to specify a model formula, so they stay in `score()`'s Python and R surfaces where the arguments can be named. And `clocks` took `list`/`info`/`cite` sub-verbs in the specification; as shipped it is one verb with flags, because the three did the same lookup and differed only in what they printed. `power` and `consensus` are new since the specification was written.

15. The layer that says what a score is worth

Sections 1–14 describe the architecture that computes a score. This one describes the layer that says how much that score is worth. It is deliberately a *layer*: nothing below changed, no score moved, and every new artefact is reported rather than applied.

15.1 New modules

Module	What it holds
`falconage.uncertainty`	`technical_se`, `interval`, `conformal_interval`, `icc_from_replicates`, and the loaders for the two calibration tables
`falconage.core.tissue`	the specimen alias table, families, and the published discordance between them
`falconage.core.preanalytical`	the recognised `obs` fields, their thresholds, and the manifest block
`falconage.preprocess.batch`	`fit_batch_reference`, `apply_batch_reference`, `BatchReference`
`falconage.models.ops.DERIVATIVE`	an analytic derivative per transform, and an explicit refusal for the four that destroy the information
`python/tools/`	the three derivation scripts for the shipped data artefacts

15.2 New registry artefacts

Each is derived by a script in `python/tools/`, digest-checked, and carries its provenance into the run manifest. None can be regenerated in CI, two need the 586 MB corpus, one needs a 72 MB published spreadsheet, so CI checks that the shipped files parse and cover what they claim instead, and the local `--check` catches staleness before a commit.

File	Rows	Source	Built by
`probe_icc.csv.gz`	283,860	ADNI duplicates, EPIC v1, CC BY 4.0	`build_probe_icc.py`
`platform_bias.csv`	86	this corpus, masked and rescored	`build_platform_bias.py`
`conformal.csv`	54	healthy adult whole-blood controls in this corpus	`build_conformal.py`
`references/*.csv`, `*.csv.gz`	29,149 in five files	the authors' published per-CpG values (Horvath 2013, DunedinPoA m38, CorticalClock, AltumAge); healthy adult whole blood in this corpus	`build_references.py`
`evidence.yaml`	18	multi-clock comparison studies	hand-curated, DOI enforced at load

15.3 Where the checks fire

The registry already declared three properties that scoring enforced. Two more were added late, and they use the same shape, a declared property of the clock, checked in `score()`, refusing when the substitution is a category error and warning when it is an offset.

Declared	Refuses when
`scale_type` → `LEGAL_OPS`	an operation is undefined for the output's scale
coverage / coefficient mass	too much of the model is absent
`availability` tier	the coefficients are not distributable
`requires_cohort`	a cohort-centred clock is given one sample
`tissue_policy`	a clock is scored off a tissue with no counterpart, or on cell-free DNA
**`age_years_relative`	the absolute acceleration convention is asked of a clock whose origin moves between cohorts

The last one also fixed a latent gap rather than only adding a rule. `LEGAL_OPS` has listed `acceleration` and `residual` as separate entries since v1.0, but `acceleration()` checked `"acceleration"` whatever convention was asked for, so the distinction in the data model was never read. It is now: `absolute` and `both` need `acceleration`, `residual` and `within_group` need `residual`.

15.4 Figure inventory, updated

Twenty-six figure types, four of them new:

Figure	Answers
`reliability_forest`	which of these clocks can be trusted to move
`score_interval`	which two samples are actually distinguishable
`platform_bias`	what this array costs each clock, in years
`consensus_plot`	did the difference hold up across clocks, or is it one clock

`save_all()` takes `se=`, `conformal=` and `consensus=` and emits them when given; `test/run_all.py` computes all three per dataset, so the gallery figures come from real public data rather than a synthetic demonstration.

15.5 What did not change

No score. The earlier numbers are bit-identical for any dataset whose clocks were legal to score in the first place, the two integration tests that changed did so because they were scoring placenta clocks on cord blood and buccal clocks on adult blood, and those results were wrong before the check existed rather than different after it.

Chapter 32. 8. R package

8.1 What “identical” means here

`FALCONAge` is not a reimplementation. Every number it returns is computed by the Python core through `reticulate`, so the R and Python answers are the same bytes because they are the same computation. What the R package adds is an interface an R user would have written: S3 classes with `print`, `summary`, `plot` and `as.data.frame`; `data.frame` in and `data.frame` out; roxygen2 documentation; a pkgdown site; and errors that arrive as R conditions rather than Python tracebacks.

This is the design tAge uses (`tAge/R/predict.R`), with one difference: tAge asks the user to set `RETICULATE_PYTHON` by hand, and FALCONAge manages the environment itself.

8.2 Environment management

`R/install.R`

```
falconage_install(method = c("auto", "uv", "virtualenv", "conda"),
                  envname = "r-falconage",
                  version = NULL,        # NULL = the version matching this R package
                  gpu = FALSE,
                  force = FALSE)

falconage_config()      # what Python, what version, what device, what registry
falconage_available()   # logical, does a working core exist
falconage_update()
```

`method = "auto"` prefers `uv` when it is on the PATH, because it resolves in seconds rather than minutes, and falls back to `virtualenv`. `gpu = TRUE` installs the CUDA torch build.

`version = NULL` pins the Python core to the exact version that matches the R package, read from `DESCRIPTION`’s `Config/falconage/python-version` field. This is the mechanism that stops an R package from silently driving a Python core it was never tested against.

`falconage_config()` prints the same block the CLI prints, so a bug report from an R user and one from a Python user carry comparable information:

```
FALCONAge 1.0.0 (R)
  python core   1.0.0  /home/x/.virtualenvs/r-falconage/bin/python
  torch         2.6.0  CUDA 12.4
  device        cuda:0  NVIDIA RTX A4000, 16 GB
  dtype         float64
  registry      1.1.0  (178 clocks)
  cache         /home/x/.cache/falconage  (2.1 GB)
```

8.3 The bridge

`R/zzz.R` and `R/bridge.R`

```
.falconage <- new.env(parent = emptyenv())

.onLoad <- function(libname, pkgname) {
  .falconage$py <- reticulate::import("falconage", delay_load = list(
    environment = "r-falconage",
    on_load  = function() check_version_compatibility(),
    on_error = function(e) NULL          # defer the error to first use
  ))
  set_default_options()
}
```

`delay_load` matters: without it, `library(FALCONAge)` fails on a machine with no Python environment yet, which breaks `R CMD check` on CRAN's build farm. With it, the package loads, `falconage_available()` returns `FALSE`, and every exported function that needs the core calls `require_core()`, which raises a condition telling the user to run `falconage_install()`.

Type marshalling. Explicit, never implicit.

R	Python	Direction
`data.frame`	`pandas.DataFrame`	both, via reticulate
`falcon_data`	`.h5ad` path	both: the object is passed as a file path, never marshalled cell by cell
`falcon_result`	`.h5ad` path plus a results `data.frame`	both
`character(1)`	`str`	both
`NULL`	`None`	both
`NA`	`float('nan')` for numerics, `None` for strings	R to Python
`matrix`	`numpy.ndarray`	both

The `falcon_data` rule is the important one. Passing an 850,000-column matrix through reticulate cell by cell is slow and copies twice. Instead the R object holds a path to an `.h5ad` file plus a cached summary (dimensions, platform, sample table), and both languages read the same file. This is also what makes a mixed-language workflow possible: preprocess in R, score in Python, plot back in R, one artefact throughout.

Error translation. `bridge.R` wraps every call:

```
py_call <- function(fn, ...) {
  tryCatch(fn(...), error = function(e) {
    info <- parse_python_error(e)      # class, message, .remedy
    rlang::abort(
      message = info$message,
      class   = paste0("falconage_", info$class),
```

```
      remedy  = info$remedy,
      parent  = e
    )
  })
}
```

A `MissingUnitsError` from Python becomes an R condition of class `falconage_missing_units_error`, so R users can `tryCatch` on it, and the message is the one-sentence remedy rather than a 40-line traceback. The Python traceback stays available as `parent`.

8.4 Public API - one-to-one with Python

Every Python verb has an R counterpart with the same name in snake_case and the same argument names. This table is the contract the conformance tests enforce (§10.5).

Python	R	Returns
`fa.download()`	`download()`	`falcon_download`
`fa.download(dry_run=True)`	`download(dry_run = TRUE)`	`data.frame`
`fa.prepare()`	`prepare()`	`falcon_data`
`fa.prepare_clinical()`	`preprocess_clinical()`	`falcon_data`
`fa.qc()`	`qc()`	`falcon_qc`
`fa.score()`	`score()`	`falcon_result`
`fa.acceleration()`	`acceleration()`	`data.frame`
`fa.cox_hazard()`	`cox()`	`data.frame`
`fa.icc()`	`icc()`	`data.frame`
`fa.run_benchmark()`	`run_benchmark()`	`falcon_benchmark`
`fa.registry.list()`	`list_clocks()`	`data.frame`
`fa.registry.get()`	`clock_info()`	`list`
`fa.registry.cite()`	`cite_clock()`	`character`
`fa.registry.compatible_with()`	`compatible_clocks()`	`data.frame`
`fa.plot.ba_vs_ca()`	`plot_ba_vs_ca()`	`ggplot`
`fa.report()`	`report()`	path, invisibly

Plotting is the one place the two diverge deliberately. Python returns matplotlib figures; R calls the Python `*_data()` function to get the frame and renders it with ggplot2, so an R user gets a `ggplot` object they can modify. The data is identical; the rendering is native to each language.

8.5 S3 classes

`R/classes.R`

```
# falcon_data
print.falcon_data       # dimensions, modality, platform, species, age availability
```

```
summary.falcon_data     # plus per-clock coverage
dim.falcon_data
as.data.frame.falcon_data

# falcon_result
print.falcon_result     # clocks run, samples, warnings
summary.falcon_result   # per-clock mean, SD, missingness, flags
as.data.frame.falcon_result   # the long table
plot.falcon_result      # dispatches to plot_ba_vs_ca()
`[.falcon_result`       # subset by clock or sample

# falcon_qc
print.falcon_qc
plot.falcon_qc
```

`print.falcon_result` shows what a user needs before trusting the numbers:

```
<falcon_result>  120 samples x 5 clocks

  clock          unit     mean     sd   missing  flags
  horvath2013    years    54.2    9.1      0.8%  -
  hannum         years    51.7    8.4      0.0%  -
  phenoage       years    56.9   11.2      0.0%  -
  grimage2       years    61.4    8.8     12.4%  -
  dunedinpace    ratio     1.03   0.11     2.1%  -

  2 warnings. See warnings(x).
  grimage2 is research-use-only.
```

8.6 DESCRIPTION

The file as it ships, not as it was specified: the differences are noted below it:

```
Package: FALCONAge
Title: Multiomic Biological Age and Aging Clock Scoring
Version: 1.0.0
Authors@R:
    person("Bhagesh", "Hunakunti", , "bhunakun@uni-bonn.de", role = c("aut", "cre"),
           comment = c(ORCID = "0000-0002-5957-8005"))
Description: Scores DNA methylation and clinical chemistry data against a
    catalogue of 178 published aging clocks, and provides the downstream
    statistics the field expects: age acceleration in its three conventions,
    survival and association models, technical reliability, and the AA1/AA2
    benchmark. Numerical work is delegated to the 'falconage' Python package
    through 'reticulate', so results from R and from Python are the same bits
    rather than two implementations that agree approximately. Twenty-three
    clocks carry coefficients inside the package and run offline; twenty-eight
```

```
    are implemented as scaffolds whose research-use-only coefficients the user
    supplies.
License: GPL (>= 3)
URL: https://github.com/bhagesh-h/FALCONAge,
    https://bhagesh-h.github.io/FALCONAge/,
    https://bhagesh-h.github.io/FALCONAge/r
BugReports: https://github.com/bhagesh-h/FALCONAge/issues
Depends:
    R (>= 4.1)
Imports:
    grDevices,
    reticulate (>= 1.34),
    stats,
    utils,
    yaml
Suggests:
    ggplot2,
    knitr,
    optparse,
    patchwork,
    rmarkdown,
    testthat (>= 3.0.0)
VignetteBuilder: knitr
biocViews: Epigenetics, DNAMethylation, DifferentialMethylation,
    MethylationArray, Software
Config/testthat/edition: 3
Encoding: UTF-8
Language: en-GB
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.3.2
```

Four departures from the specification, each deliberate:

Specified	Shipped	Why
`ggplot2`, `cli`, `rlang`, `jsonlite` in `Imports`	`ggplot2` in `Suggests`; the other three absent	Plotting returns a data frame and draws with base graphics, so `grDevices` is the hard dependency and `ggplot2` is optional. Four fewer required packages is four fewer ways for a CRAN check to fail on a machine we do not control.
`Methods implemented include ... <doi:...>` in the description	Absent	CRAN wants DOIs for methods the package *implements in R*. Every method here is implemented in the Python core; the per-clock references live in the registry and come back from `cite()`, which is a truer place for them than a fixed list of five.
`SystemRequirements: Python (>= 3.10) ...`	Absent	On reticulate ≥ 1.41 the environment is provisioned automatically on first use, so Python is not a system requirement the user must satisfy. Still to be added before a CRAN submission, since the field is also documentation.
`Config/falconage/python-version`	Absent	The R package pins the Python core by git tag inside `falconage_install()`, so the field would be a second copy of a number that already exists in one place.

Every example and every test that needs the core is wrapped in `skip_if_not(falconage_available())`, so `R CMD check` passes on a machine with no Python.

8.7 Roxygen conventions

Following the cyRAVEN register, see `R/thresholds.R` in that package for the model. Every exported function gets:

- an opening paragraph saying **what** it does, then **why** it does it that way when the choice is not obvious
- `@param` on every argument, naming the unit or the allowed values
- `@return` on every export, describing the object rather than naming its class
- `@examples`, runnable, wrapped in `\donttest{}` when they need the core or the network
- `@seealso` linking the Python equivalent by name
- `@references` with the DOI for anything implementing a published method

Worked example:

```
#' Score biological age with one or more aging clocks
#'
#' WHAT: applies published aging-clock definitions to a prepared dataset and
#' returns one value per sample per clock, with the provenance needed to
#' interpret it.
#'
#' WHY THE DEFAULT IS `"compatible"` RATHER THAN `"all"`: the registry holds
#' clocks for several species, platforms and molecular layers, and most of them
#' do not apply to any given dataset. Running everything produces a table whose
#' majority of rows were computed almost entirely from imputed reference values,
#' which looks like a result and is not one. `"compatible"` selects the clocks
#' whose species, platform and feature coverage actually match the data, and
#' reports what it excluded.
#'
#' @param data A `falcon_data` object from [preprocess_methylation()] or
#'   [preprocess_clinical()].
#' @param clocks Clock identifiers, or `"compatible"` (default) to select
#'   automatically, or `"all"`. See [list_clocks()].
#' @param device `"auto"`, `"cpu"`, `"cuda"`, or a specific device such as
#'   `"cuda:1"`. `"auto"` uses CUDA when it is available.
#' @param dtype `"float64"` (default) or `"float32"`. Clocks whose registry
#'   entry requires double precision are promoted regardless, with a warning.
#' @param max_missing Largest fraction of a clock's features that may be absent
#'   before scoring that clock is an error rather than a warning. Default 0.30.
#' @param sex_unknown What to do when a clock needs sex and it is not recorded:
#'   `"error"` (default), `"impute_from_methylation"`, or `"half"`.
#'
#' @return A `falcon_result`. [as.data.frame()] gives one row per sample per
#'   clock with the value, its unit, its scale type, the fraction of features
#'   that had to be imputed, and any flags raised. The underlying dataset,
#'   including the run manifest, is reachable through the object.
#'
```

```
#' @examples
#' \donttest{
#' if (falconage_available()) {
#'   data <- preprocess_methylation(
#'     system.file("extdata", "example_betas.csv", package = "FALCONAge"),
#'     platform = "illumina_450k"
#'   )
#'   res <- score(data, clocks = c("horvath2013", "phenoage"))
#'   head(as.data.frame(res))
#' }
#' }
#'
#' @seealso [acceleration()] for age acceleration, [list_clocks()] to browse the
#'   registry, and the Python equivalent `falconage.score()`.
#' @references
#' Horvath S (2013). DNA methylation age of human tissues and cell types.
#' Genome Biology 14:R115. \doi{10.1186/gb-2013-14-10-r115}
#' @export
score <- function(data, clocks = "compatible", device = "auto",
                  dtype = "float64", max_missing = 0.30,
                  sex_unknown = "error", ...) { }
```

8.8 Command-line front end

`inst/scripts/falconage.R`, invoked the cyRAVEN way:

```
Rscript "$(Rscript -e 'cat(system.file("scripts","falconage.R",package="FALCONAge"))')" \
  score --input prepared.h5ad --clocks horvath2013,grimage2 --outdir results/
```

Options are built in `R/cli-options.R` with `optparse`, and the flag names match the Python CLI exactly, so a script written against one runs against the other with only the interpreter changed.

8.9 pkgdown reference groups

`r/_pkgdown.yml`. Group titles are identical to the quartodoc sections in §11, which is what makes the two sites navigable as one.

```
url: https://bhagesh-h.github.io/FALCONAge

template:
  bootstrap: 5
  light-switch: true

reference:
- title: Setup
  desc: >
    FALCONAge computes through a Python core. These functions create and inspect
```

```
    that environment; everything else assumes it exists.
  contents: [falconage_install, falconage_config, falconage_available, falconage_update]

- title: Download
  desc: >
    Public data by accession. One verb covers GEO, ArrayExpress, SRA, Zenodo,
    Figshare, Hugging Face and GDC, and returns files alongside a sample table
    with age and sex extracted from whatever free-text field held them.
  contents: [download, probe, cache_ls, cache_rm, cache_verify]

- title: Preprocess
  desc: >
    Raw intensities or a public matrix into a scoreable one. Normalisation is
    selectable and recorded, because it changes the answer.
  contents: [preprocess_methylation, preprocess_clinical, detect_platform,
             aggregate_epicv2, estimate_sex, qc]

- title: Score
  desc: One call runs any number of clocks. Every value carries its provenance.
  contents: [score, acceleration]

- title: The clock registry
  desc: >
    What is available, what it predicts, in what unit, on what platform, and
    where the published definition and the circulating coefficients disagree.
  contents: [list_clocks, clock_info, compatible_clocks, cite_clock,
             registry_version]

- title: Analysis
  desc: >
    Tests on per-sample values. Age acceleration is standardised within sex
    before any hazard ratio, so effect sizes compare across clocks.
  contents: [cox, c_index, associate, interactions, icc, run_benchmark]

- title: Figures
  contents: [starts_with("plot_"), theme_falconage]

- title: Reporting
  contents: [report, run_manifest]

articles:
- title: Get started
  navbar: ~
  contents: [FALCONAge, methylation]
- title: Working with data
  navbar: Working with data
  contents: [download, clinical, clocks]
- title: Going further
```

```
  navbar: Going further
  contents: [gpu, benchmarking]
```

Chapter 33. Docker, testing, documentation and CI

9. Docker

Two images from one Dockerfile family, following the cyRAVEN pattern in its `inst/scripts/Dockerfile`: pinned base, dependency layer copied before source, and dependencies read from the declaration rather than a second list. The images carry no build-time checks, to keep them light; the test suite runs in the image on demand (`python -m pytest /opt/falconage/src/python/tests`).

9.1 Images

Tag	Base	Contains
`falconage:1.0.0-cpu`	`python:3.12-slim-bookworm`	Python core, R, `FALCONAge`, CLI
`falconage:1.0.0-cuda`	`nvidia/cuda:12.4.1-runtime-ubuntu22.04`	the same plus the CUDA torch build
`falconage:1.0.0-slim`	`python:3.12-slim-bookworm`	Python core and CLI only, no R

9.2 Dockerfile.cpu, annotated

```
# ==============================================================================
# FALCONAge - reproducible CPU environment
# ==============================================================================
#
# WHY A PINNED BASE. Clock coefficients are fixed, but the arithmetic around
# them is not: a different BLAS changes the last bits of a 78,464-dimensional
# PC rotation, and a different scikit-learn changes what SimpleImputer stores.
# Neither moves a published age by a year, but both make two runs of the same
# data disagree in the seventh decimal, which is exactly what the gold-standard
# tests assert on. `latest` would silently re-roll that on a rebuild.
#
# Versions this image reproduces:
#   Python 3.12.8 . torch 2.6.0 (CPU) . numpy 2.2.1 . pandas 2.2.3
#   anndata 0.11.3 . scikit-learn 1.6.1 . R 4.4.3 . reticulate 1.40.0
#
# BUILD, from the repository root:
#   docker build -f docker/Dockerfile.cpu -t falconage:1.0.0-cpu .
#
# RUN. Nothing is baked in: no data, no cohort, no credentials.
#   docker run --rm \
#     -v "$PWD/data:/data:ro" -v "$PWD/results:/results" \
#     falconage:1.0.0-cpu \
#     score --input /data/prepared.h5ad --clocks compatible --outdir /results
#
#   On Windows PowerShell replace "$PWD" with "${PWD}".
```

```
#
# ITERATING WITHOUT REBUILDING. Mount a checkout and set FALCONAGE_SOURCE; the
# entrypoint runs that tree instead of the installed copy. A rebuild is still
# needed when the source starts requiring a package the image does not have.
#     -v "$PWD:/src:ro" -e FALCONAGE_SOURCE=/src
# ------------------------------------------------------------------------
FROM python:3.12-slim-bookworm AS base

LABEL org.opencontainers.image.title="FALCONAge" \
        org.opencontainers.image.description="Multiomic biological age and aging clock
scoring" \
      org.opencontainers.image.source="https://github.com/bhagesh-h/FALCONAge" \
      org.opencontainers.image.licenses="GPL-3.0-or-later"

ENV PYTHONUNBUFFERED=1 \
    PIP_NO_CACHE_DIR=1 \
    # torch and numpy are OpenMP-threaded. Unset they take every core on the
    # host, which is antisocial on a shared machine and causes thermal
    # throttling on a laptop. Override with -e OMP_NUM_THREADS=8.
    OMP_NUM_THREADS=4 \
    # Weights land here. Mount a host directory over it to keep them between
    # runs instead of re-downloading a gigabyte each time.
    HF_HOME=/opt/falconage/cache \
    FALCONAGE_CACHE=/opt/falconage/cache \
    LANG=C.UTF-8

# Each library is here because a specific wheel needs it, not as a blanket
# build-essential block.
#   libhdf5      -> h5py, which anndata writes .h5ad through
#   libcurl/ssl  -> the download module
#   libxml2      -> lxml, used by the GEO MINiML parser
#   libgomp1     -> OpenMP runtime for torch and numpy
RUN apt-get update && apt-get install -y --no-install-recommends \
        libhdf5-dev libcurl4-openssl-dev libssl-dev libxml2-dev \
        libgomp1 zlib1g-dev libbz2-dev liblzma-dev ca-certificates curl \
    && rm -rf /var/lib/apt/lists/*

WORKDIR /opt/falconage

# ---- dependency layer -------------------------------------------------------
# Only the declaration, before any source. This layer costs several minutes and
# rebuilding it on every edit to src/ is the single biggest waste in a naive
# Dockerfile.
COPY python/pyproject.toml python/uv.lock /opt/falconage/python/
RUN pip install --no-cache-dir uv \
 && cd python && uv export --frozen --no-dev --format requirements-txt > /tmp/req.txt \
 && pip install --no-cache-dir -r /tmp/req.txt
```

```
# ---- R layer ----------------------------------------------------------------
FROM base AS with-r
RUN apt-get update && apt-get install -y --no-install-recommends \
        r-base r-base-dev libfontconfig1-dev libfreetype6-dev libpng-dev \
    && rm -rf /var/lib/apt/lists/*

# install_deps.R parses r/DESCRIPTION instead of repeating the package list, so
# the image can never install a different set than the package declares.
COPY r/DESCRIPTION docker/install_deps.R /opt/falconage/r/
RUN Rscript /opt/falconage/r/install_deps.R /opt/falconage/r/DESCRIPTION

# ---- source -----------------------------------------------------------------
FROM with-r AS final
COPY . /opt/falconage/src
RUN pip install --no-cache-dir --no-deps /opt/falconage/src/python \
 && R CMD INSTALL --no-multiarch --with-keep.source /opt/falconage/src/r \
 && cp /opt/falconage/src/docker/entrypoint.sh /opt/falconage/entrypoint.sh \
 && chmod +x /opt/falconage/entrypoint.sh

# reticulate must find the interpreter that already has the core installed,
# rather than trying to build an environment inside a read-only container.
ENV RETICULATE_PYTHON=/usr/local/bin/python

VOLUME ["/results", "/opt/falconage/cache"]
WORKDIR /work
ENTRYPOINT ["/opt/falconage/entrypoint.sh"]
CMD ["--help"]
```

`Dockerfile.cuda` differs only in the base image and the torch index URL.

9.3 entrypoint.sh

Same indirection as cyRAVEN's: run the installed copy by default, or a mounted source tree when `FALCONAGE_SOURCE` is set, so routine edits need no rebuild.

```
#!/bin/sh
set -e

if [ -n "${FALCONAGE_SOURCE}" ]; then
  # -e is deliberately editable-install-free: prepending to PYTHONPATH is
  # instant and leaves the image's site-packages untouched, which matters
  # because most run configurations mount the filesystem read-only.
  PYTHONPATH="${FALCONAGE_SOURCE}/python/src:${PYTHONPATH}"
  export PYTHONPATH
fi

# `falconage` with no recognised verb is a help request, not an error.
case "$1" in
  ""|-h|--help) exec falconage --help ;;
```

```
esac

exec falconage "$@"
```

9.4 docker-compose.yml

```yaml
services:
  falconage:
    image: falconage:1.0.0-cpu
    volumes:
      - ./data:/data:ro
      - ./results:/results
      - falconage-cache:/opt/falconage/cache
    environment:
      OMP_NUM_THREADS: 8

  falconage-gpu:
    image: falconage:1.0.0-cuda
    deploy:
      resources:
        reservations:
          devices: [{driver: nvidia, count: all, capabilities: [gpu]}]
    volumes:
      - ./data:/data:ro
      - ./results:/results
      - falconage-cache:/opt/falconage/cache

volumes:
  falconage-cache:
```

The named cache volume is not cosmetic: the PC clocks are several hundred megabytes each and re-downloading them on every container start is the difference between a two-minute run and a twenty-minute one.

10. Testing

10.1 Gold-standard vectors

The load-bearing test. For every clock: an input matrix and the output the reference implementation produces, stored under `tests/vectors/<clock>/`, asserted to `1e-6` in FP64, and consumed by both languages.

```
tests/vectors/
├── manifest.yaml           # clock -> {input, expected, tolerance, source, generated}
├── horvath2013/
│   ├── input.parquet       # 20 synthetic samples x 353 CpGs, seeded
│   ├── expected.parquet    # sample_id, value
│   └── provenance.json     # pyaging 0.3.0, FP64, CPU, commit hash, date
└── ...
```

Inputs are synthetic and seeded rather than real data: no consent question, no file-size problem, and a seeded generator gives coverage of the edge cases real data rarely contains, all-zero rows, a constant feature, values at exactly 0 and 1, and a controlled missingness pattern.

Generation is a one-off script per source implementation (`scripts/generate_vectors_pyaging.py`, `scripts/generate_vectors_biolearn.py`, `scripts/generate_vectors_bioage.R`), run in a pinned environment and committed with its provenance. Regenerating is a deliberate act recorded in the changelog, never a fix for a failing test.

`registry/validate.py` fails when a clock has no vector, so the registry cannot grow past the test suite.

Which oracle

Validating against pyaging makes pyaging the arbiter of correctness, and its bugs become ours. Oracles are therefore ranked, and each vector records which tier it used:

Rank	Oracle	Available for
1	A worked example published by the clock's own authors, reference inputs and the age they report	a minority of clocks, but the only truth that is not somebody's code
2	The authors' own reference implementation, run in a pinned environment	most clocks with an author repository
3	Hand computation from the published equation, for clocks small enough	PhenoAge, Weidner, Vidal-Bralo, Bocklandt, Garagnani, ZhangMortality
4	An independent third-party package, with at least two agreeing	the remainder
5	Synthetic weights, architecture only	licensed clocks with no published worked example (§4.7)

A rank-4 vector where pyaging and biolearn disagree is not a vector; it is an open question, and it goes to `registry/diffs/` rather than to `tests/vectors/`.

10.2 Cross-implementation comparison

A separate, non-blocking job compares FALCONAge against pyaging, biolearn, computage, methylCIPHER and BioAge on shared clocks, and writes a table of agreements and disagreements to `docs/comparison.md`. Disagreements are published rather than hidden, the science notes §15.2 shows the field already carries several, and a user comparing a FALCONAge number to a pyaging number deserves to know which ones differ and why.

10.3 Property tests

Hypothesis in Python, `testthat` with generated inputs in R.

Property	Holds for
deconvolution proportions sum to 1 and are non-negative	all deconvolution clocks
probability outputs lie in `[0, 1]`	`scale_type: probability`
permuting feature column order does not change the result	every clock
duplicating a sample does not change either copy's score	every clock except per-sample-normalised ones
per-sample-normalised clocks DO change under sample duplication	ZhangEN, ZhangBLUP, DunedinPACE, Pasta, REG - this is a property, not a bug, and the test pins it

Property	Holds for
a synthetic monotone age gradient produces a monotone prediction	chronological clocks
`score(x)[i] == score(x[i])` for a single sample	every clock without per-sample normalisation
FP32 and FP64 agree within the registry's per-clock tolerance	every clock
scale-type guards reject illegal operations	`acceleration(absolute)` on a rate clock

10.4 Unit and integration

Unit tests per op with hand-computed expected values, the 15 preprocess ops and 15 postprocess transforms each get one, because these are where an off-by-one in a quantile index or a wrong constant hides. Integration tests run the full path (download a small cached fixture, preprocess, score, analyse, report) on both a methylation and a clinical fixture.

10.5 R/Python conformance

A CI gate, not an aspiration.

```
# tests/conformance/test_r_python_parity.py
@pytest.mark.parametrize("clock", REGISTRY.list())
def test_identical_scores(clock, fixture_h5ad, tmp_path):
    py = falconage.score(fa.FalconData.read_h5ad(fixture_h5ad), clocks=[clock])
    r  = run_r(f'''
        library(FALCONAge)
        d <- read_falcon_data("{fixture_h5ad}")
        write.csv(as.data.frame(score(d, clocks = "{clock}")), "{tmp_path}/r.csv")
    ''')
    assert_frame_equal(py.table(), pd.read_csv(tmp_path / "r.csv"), atol=0, rtol=0)
```

`atol=0, rtol=0` - exact equality, not approximate. The two languages run the same computation, so anything other than bit equality means the bridge is doing something it should not, such as round-tripping a float through a string.

A second conformance test asserts API symmetry: every function in the Python public namespace has an R export with the matching name and argument set, checked against the table in §8.4.

10.6 Performance regression

Benchmarks on a fixed synthetic dataset, run nightly, with results committed so a slowdown is visible in the history rather than discovered later. Tracked: per-clock scoring throughput at 1,024 / 8,192 / 32,768 samples on CPU and CUDA; KNN imputation on a 500 by 400,000 matrix; IDAT parsing throughput; peak memory for the largest PC clock.

10.7 Coverage targets

90% line coverage on `models/`, `score/` and `registry/`, which are where a wrong number comes from; 75% elsewhere. Coverage on `download/` is deliberately lower because most of it is network error handling exercised by mocks rather than by real transfers.

11. Documentation

11.1 One site, two languages

A single Quarto site at `docs/`, with the Python API rendered by `quartodoc` and the R API by `pkgdown`, sharing a theme and navigation. Section titles are identical on both sides (§8.9), so a reader moving between them sees the same structure.

```
docs/
├── _quarto.yml
├── index.qmd                    # what it is, in four paragraphs
├── installation.qmd             # pip, R, Docker, GPU
├── quickstart.qmd               # the same analysis in both languages, side by side
├── guide/
│   ├── concepts.qmd             # BA, AA, generations, units - from [the science notes]
(science.qmd) 1 and 8
│   ├── download.qmd
│   ├── preprocess.qmd
│   ├── scoring.qmd
│   ├── choosing-a-clock.qmd
│   ├── analysis.qmd
│   ├── benchmarking.qmd
│   ├── gpu.qmd
│   ├── reproducibility.qmd
│   └── caveats.qmd              # [the science notes](science.qmd) 16, compressed
├── clocks/
│   ├── index.qmd                # searchable registry table, generated from YAML
│   └── <clock>.qmd              # one page per clock, generated
├── reference/                   # quartodoc, Python
├── r/                           # pkgdown output, mounted into the site
└── _extensions/
```

11.2 _quarto.yml

```yaml
project:
  type: website
  output-dir: _site
  pre-render:
    - python scripts/build_clock_pages.py     # registry YAML -> clocks/*.qmd
    - Rscript scripts/build_pkgdown.R         # pkgdown -> docs/r/

website:
  title: "FALCONAge"
  favicon: assets/logo.png
  navbar:
    left:
      - text: Get started
        href: quickstart.qmd
      - text: Guide
```

```
        menu: [guide/concepts.qmd, guide/download.qmd, guide/preprocess.qmd,
               guide/scoring.qmd, guide/choosing-a-clock.qmd, guide/analysis.qmd]
      - text: Clocks
        href: clocks/index.qmd
      - text: Reference
        menu:
          - {text: "Python API", href: reference/index.qmd}
          - {text: "R API",      href: r/reference/index.html}
    right:
      - icon: github
        href: https://github.com/bhagesh-h/FALCONAge

format:
  html:
    theme: {light: cosmo, dark: darkly}
    toc: true
    code-copy: true
    code-annotations: hover

quartodoc:
  package: falconage
  dir: reference
  style: pkgdown                    # section-grouped, matching _pkgdown.yml
  sections:
    - title: Download
      desc: >
        Public data by accession. One verb covers GEO, ArrayExpress, SRA,
        Zenodo, Figshare, Hugging Face and GDC, and returns files alongside a
        sample table with age and sex extracted from whatever free-text field
        held them.
      contents: [download, probe, cache.ls, cache.rm, cache.verify]
    # ... the remaining sections carry the same titles and desc text as
    # r/_pkgdown.yml, which is what makes the two sites read as one.
```

`quartodoc`'s `style: pkgdown` produces a grouped reference index with the same shape pkgdown produces, which is why the two sides can share section titles and `desc` paragraphs verbatim. The Quarto version is pinned in CI and in the Docker image, because Quarto 2 is a full rewrite due late 2026 and a documentation build should not break on a toolchain upgrade nobody asked for.

11.3 Bilingual examples

Every guide page shows the same operation in both languages with Quarto tabsets:

````
::: {.panel-tabset}

## Python

```python
````
````

````
import falconage as fa

data = fa.prepare(fa.read_betas("betas.csv"))
res  = fa.score(data, clocks=["horvath2013", "phenoage"])
res.table().head()
```

## R

```r
library(FALCONAge)

data <- preprocess_methylation("betas.csv", platform = "illumina_450k")
res  <- score(data, clocks = c("horvath2013", "phenoage"))
head(as.data.frame(res))
```

:::
````

A CI job extracts both blocks from every page and runs them, so an example that has drifted from the API fails the docs build rather than misleading a reader.

11.4 Generated clock pages

`scripts/build_clock_pages.py` turns each registry YAML into a page carrying: what it predicts and in what unit, the platform and species it applies to, the feature count, the architecture, the preprocess and postprocess chain, the reliability ICCs, the benchmark scores, the citation and DOI, any known discrepancies, and a runnable example. The index is a searchable, filterable table over all of them. This is generated, never hand-written, so the docs cannot drift from the registry.

11.5 Docstring conventions

numpydoc in Python, roxygen2 in R, deliberately parallel:

Concept	numpydoc	roxygen2
Summary	first line	first line
Rationale	`Notes` section	body paragraph after the title
Argument	`Parameters`	`@param`
Return	`Returns`	`@return`
Example	`Examples`	`@examples`
Cross-reference	`See Also`	`@seealso`
Citation	`References`	`@references`

Both carry the same rationale text where a function exists in both languages, so a reader gets the same explanation whichever site they land on.
````

12. CI/CD and publishing

12.1 Workflows

FALCONAge runs four workflows, not nine, and the difference is a decision rather than a backlog. The table below is the specification. What `.github/workflows/` actually holds:

Shipped	Covers	Specified rows it replaces
`python-test.yaml`	tests, lint, nine guards, the unpinned-resolution job	`python-test.yaml`, `registry-validate.yaml`
`R-CMD-check.yaml`	`R CMD check --as-cran` with the Python core, plus a no-Python job	`r-cmd-check.yaml`, `conformance.yaml`
`docs.yaml`	Quarto site with the embedded pkgdown tree, deployed to gh-pages	`docs.yaml`
`release.yaml`	one workflow, a `python` job and an `r` job, both behind the version guard	`release-pypi.yaml`, `release-r.yaml`

Registry validation and R/Python conformance are **jobs inside the two check workflows**, not workflows of their own: they share a checkout and an install with the suite they belong to, and splitting them would pay for that twice to gain a second badge saying the same thing.

`docker.yaml` and `benchmark.yaml` do not exist. Nothing publishes an image to a registry; `README.md` tells a user to build `docker/Dockerfile.cpu` locally, and the `falconage:1.0.0-cpu` tag it names is the one that build produces. The nightly benchmark would commit timings from a shared runner whose neighbours decide the numbers, which is a regression detector that cries wolf; the measurements that exist were taken on known hardware and are in GPU support.

The `python-test.yaml` row below is the one to disregard outright. There is no matrix and no Codecov upload, and the workflow's own header explains at length why: the lock file pins one resolution, `uv export --frozen` reproduces it, and on any other interpreter the pinned wheels either do not exist or do not match, so a matrix job fails before reaching a test. Three red badges where one would do teaches people to ignore the badge. Coverage was dropped after printing sixty lines of site-packages paths that nobody read.

Workflow	Trigger	Does
`python-test.yaml`	push, PR	pytest on 3.10–3.13 × {ubuntu, macos, windows}; gold-standard vectors; property tests; coverage to Codecov
`r-cmd-check.yaml`	push, PR	`R CMD check --as-cran` on ubuntu-release with the Python core installed; a second job with no Python, asserting the package still loads and checks
`conformance.yaml`	push, PR	R/Python parity across every clock; API symmetry
`registry-validate.yaml`	push to `registry/**`	schema, vocabulary, checksums, gold-standard presence

Workflow	Trigger	Does
`docs.yaml`	push to main	Quarto site including pkgdown, examples executed, deployed to gh-pages
`docker.yaml`	push to main, tags	build and push cpu/cuda/slim to GHCR, multi-arch on cpu
`release-pypi.yaml`	tag `v*`	version guard, build sdist and wheel, publish via trusted publishing
`release-r.yaml`	tag `v*`	version guard, `R CMD build`, `--as-cran`, attach tarball to the release
`benchmark.yaml`	nightly	performance regression, results committed

12.2 The version guard

Borrowed directly from cyRAVEN's `release.yaml`, extended to four files. A tag and a version that disagree cannot be repaired afterwards, because the tag is what a citation points at.

```
- name: Versions must agree
  run: |
    TAG="${GITHUB_REF#refs/tags/v}"
    PY=$(grep -m1 '^version' python/pyproject.toml | sed 's/.*"\(.*\)"/\1/')
    R=$(grep -m1 '^Version:' r/DESCRIPTION | awk '{print $2}')
    PYREQ=$(grep -m1 'Config/falconage/python-version:' r/DESCRIPTION | awk '{print $2}')
    for pair in "python:$PY" "R:$R" "R-python-pin:$PYREQ"; do
      name="${pair%%:*}"; val="${pair#*:}"
      [ "$val" = "$TAG" ] || { echo "::error::$name is $val, tag is $TAG"; exit 1; }
    done
    echo "all versions agree at $TAG"
```

12.3 R CMD check without Python

The second job in `r-cmd-check.yaml` is the one that matters for CRAN. It runs on a machine with no Python core at all and asserts that `library(FALCONAge)` succeeds, `falconage_available()` returns `FALSE`, every example is skipped rather than erroring, and the check reports no ERROR or WARNING. The `delay_load` in `.onLoad` (§8.3) is what makes this possible, and this job is what stops it from silently regressing.

12.4 Publishing

`PUBLISHING.md` carries the same honest gap table cyRAVEN uses, per target.

PyPI - trusted publishing, no long-lived token. `sdist` plus a pure-Python wheel; no compiled extensions, so no cibuildwheel. Test on TestPyPI first for the initial release.

r-universe - the first R route, no review. Add the repository to a `packages.json` in `bhagesh-h.r-universe.dev`, and binaries for Windows, macOS and Linux appear within minutes.

CRAN - needs, beyond the above: `cran-comments.md` naming platforms checked and explaining every NOTE; `\value` on every exported function; runnable examples; total runtime under 10 minutes; no writing outside `tempdir()`; win-builder and macOS results. The reticulate dependency is the reviewable point, and the answer is the no-Python check job in §12.3 plus `SystemRequirements`.

Bioconductor - the better long-term home given the `biocViews` and the audience. Needs version `0.99.x` at submission, `BiocCheck` clean, a `BiocStyle` vignette, package under 5 MB (satisfied, since no weights ship), no `sapply` or `1:n` in package code, and a support-site subscription.

Docker - GHCR on every main push and tag; Docker Hub mirror for the release tags.

Zenodo - DOI minted from each GitHub release, which is what a methods section cites.

Chapter 34. GPU support: what was tested, on what, and what it is worth

Verified on real hardware, re-measured in full on 2026-08-10. The short version: the GPU path works, it is numerically sound, and for the clocks that ship today it is **slower than the CPU**, so `device="auto"` resolves to CPU and the GPU is opt-in.

Every figure below comes from `bhagesh/falconage:1.0.0-cuda`, built from the shipping `docker/Dockerfile.cuda`. The 2026-08-08 run of this page reported the same shape on an earlier build; where a number moved, the current one is here and the old one is named next to it.

The machine

GPU	NVIDIA GeForce RTX 4060 Laptop, 8 GB, compute capability sm_89 (Ada)
Driver	610.88, CUDA UMD 13.3
Container runtime	Docker 29.6.2, GPU passthrough working via `--gpus all`
Second adapter	AMD Radeon 780M (integrated; not used - FALCONAge has no ROCm path)

What was already present, and what had to be installed

Nothing was installed on the host. Everything went into a throwaway container.

Component	State	Action
NVIDIA driver 610.88	already installed	none
CUDA user-mode driver 13.3	already installed	none
Docker GPU passthrough	already working	none, `docker run --gpus all nvidia/cuda:12.4.1-base nvidia-smi` returned the card first try
CUDA toolkit on the host	**not installed, and not needed**	none, the torch wheel bundles its own runtime; only the driver has to be on the host
`torch` with CUDA	**missing** - the CPU image ships without it on purpose	installed `torch 2.6.0+cu124` inside a container, from `docker/Dockerfile.cuda`. Nothing was added to the host
`nvidia-container-toolkit`	already configured	none

The driver reports CUDA 13.3 and the wheel is built against 12.4. That is the supported direction: a newer driver runs an older runtime. The reverse does not work, which is why the shipping `docker/Dockerfile.cuda` pins the runtime rather than tracking the driver.

1. Device and dtype resolution

```
resolve('auto',    None,     fp64=False) -> numpy:cpu/float64
resolve('cuda',    None,     fp64=False) -> torch:cuda/float64
resolve('cuda',    float32,  fp64=False) -> torch:cuda/float32
resolve('cuda',    float32,  fp64=True)  -> torch:cuda/float64   + warning
resolve('mps')                            -> refused: no MPS device
```

The `requires_fp64` override fires as designed: a clock the registry flags as ill-conditioned gets double precision back with a warning, rather than silently honouring a `float32` request that would move its answer.

2. Which clocks the device actually reaches

`device="cuda"` is a request, and until 2026-08-10 it was granted more often in the manifest than in the arithmetic. All six model classes take a `DeviceSpec`. Of the five that can produce a score, three computed in numpy regardless, and the scoring loop then recorded the request rather than the fact.

Model class	Reaches the device	Registry entries it serves
`LinearClock`	yes	20 of the 23 that score offline
`PCLinearClock`	yes	19, none shipping weights
`AggregationClock`	yes, since 2026-08-10	6, none shipping weights
`NeuralClock`	yes, since 2026-08-10	AltumAge
`ClinicalClock`	**no, by declaration**	PhenoAge, KDM, homeostatic dysregulation
`ScaffoldClock`	not applicable	the licensed entries (28 when measured, 40 now); it raises rather than scoring

Why the clinical three decline. A methylation clock reduces thousands of probes. A clinical clock reduces nine markers: PhenoAge sums ten terms, KDM fits one univariate regression per marker, HD inverts a 9×9 covariance. That is less arithmetic than a CUDA kernel launch costs to dispatch, so a device implementation would be slower, and it would pull torch into the one modality that otherwise needs nothing beyond numpy. They set `CPU_ONLY = True`, which `falconage.models.effective_spec()` reads, so the refusal is a declared property rather than an argument quietly dropped.

Why the other two were routed anyway. `NeuralClock` is the one architecture here where a device should pay, for the reason section 4 makes measurable: a linear clock is a single dot product over a few thousand features and loses to the CPU because the transfer costs more than the multiply, while AltumAge is dense layers over 20,318 inputs with depth to parallelise. `AggregationClock` takes a mean or a 95th percentile over a probe set and will not repay a device on its own; it was routed so that what the manifest records is what happened.

What the manifest says now

Per clock, not per run. This is a real run on the card: twenty methylation clocks and one clinical clock, both scored with `device="cuda"` and combined.

```
device / backend : mixed / mixed
device_requested : cuda
compute_summary  : torch:cuda/float64 (20 clocks), numpy:cpu/float64 (1 clock)
compute["METH:horvath2013"] : {device: cuda, dtype: float64, backend: torch}
compute["CLIN:phenoage"]    : {device: cpu,  dtype: float64, backend: numpy}
```

Before this, that run reported `device="cuda"` flat, for a manifest in which one of the twenty-one clocks had never touched the card.

The three scalar fields are derived from `compute`: the shared value when every clock agreed, and `"mixed"` when they did not. Before this they were assigned once per clock inside the scoring loop, so a run reported whichever clock ran last. The same overwrite made `dtype` wrong for any run combining a `requires_fp64` clock with a `float32` request, which is all six PC clocks.

Two things make a run legitimately mixed. A `CPU_ONLY` model where the others reached the device, as above; and `combine()` across datasets scored on different machines, which is ordinary in a benchmark. The combined manifest merges the per-clock records rather than copying the first run's device onto all of them.

This changes no score. It changes what the run says about itself, which is the whole of the reproducibility claim the next section rests on.

3. CPU and GPU do not agree bit for bit

This is the finding that matters most, and it qualifies a claim made elsewhere in the documentation.

clock	max abs difference	ulps
dnamphenoage	9.9e-14	14
yingcausage	8.5e-14	3
horvath2013	5.7e-14	4
hannum	4.3e-14	3
lin	4.3e-14	6
skinandblood	4.3e-14	3
zhangen	1.4e-14	1
dnamtl	1.8e-15	2

Worst case across eight clocks: **9.9e-14 years**, about three femtoseconds of biological age.

The cause is ordinary: numpy's BLAS and cuBLAS sum a dot product in different orders, and floating-point addition is not associative. It is not a defect and it is not fixable without giving up the vendor kernels.

The per-clock figures moved between the two runs, worst case 1.3e-13 on 2026-08-08 against 9.9e-14 now, horvath2013 the largest mover at 1.3e-13 to 5.7e-14. No code changed for these eight, which are all `LinearClock`; the rebuilt image carries a different numpy and therefore a different BLAS, which sums in a different order again. Numbers that move when the linear algebra library moves are what this section is about, not an exception to it.

What this means for the conformance guarantee. FALCONAge claims R and Python return the same bits. That claim is intact, both go through the same Python core, and the R suite asserts equality at tolerance

exactly zero. It is a statement about the two *languages*, not about two *devices*. A CPU result and a GPU result agree to about 1e-13 and are not bit-identical. The run manifest records `device` and `dtype` for exactly this reason: a number is reproducible against the manifest that produced it, not against every possible manifest.

Single precision costs far more: **7e-5 years** between `float32` and `float64` on the same device. Still negligible against effects measured in single-digit years, but four orders of magnitude worse than the device difference, which is the right way round.

4. Speed: the GPU makes the shipping clocks slower

Eight clocks, 2,340 distinct features, RTX 4060, best of three timings inside a run and the best of three runs, measured in `bhagesh/falconage:1.0.0-cuda` - the image the command at the foot of this page builds. These are numbers you can reproduce, not numbers from a throwaway environment.

samples	CPU float64	CUDA float64	CUDA float32	verdict
128	**0.007 s**	0.013 s	0.011 s	CPU wins by 1.9x
1,024	**0.026 s**	0.048 s	0.048 s	CPU wins by 1.8x
4,096	**0.109 s**	0.333 s	0.216 s	CPU wins by 3.1x
16,384	**0.445 s**	3.275 s	2.397 s	CPU wins by 7.4x

The gap *widens* with size, which is the opposite of the usual shape. Profiling says why:

```
4096 samples x 2340 features, 8 clocks
  feature alignment (pandas, CPU only)    0.132 s
  the dot products, on the GPU            0.010 s
  the dot products, on the CPU            0.005 s
```

Alignment is 28x the arithmetic it feeds, and at this size that arithmetic is *faster on the CPU*: the GPU's 10 ms is mostly transfer, not multiply. At 16,384 samples it is 2.4 GB moved across PCIe to save nothing. A linear clock over a few thousand features is simply too small a matrix multiplication to be worth a device.

Read the CUDA columns as a range, not a figure. Three consecutive runs of the same script on the same idle machine gave 3.275, 3.482 and 3.664 s at 16,384 samples, a 12% spread, while the CPU column moved by under 7% (0.445 to 0.477). This is a laptop card that clocks down as it warms. Each column above is the best of the three, so the columns are not necessarily from one run; computed within a run instead, the CPU margin at 16,384 was 6.9x, 7.7x and 7.8x. A desktop card with a power budget would be steadier and would still lose, because what it is losing to is the transfer.

Two earlier revisions of this page reported 6.5x and then 4.6x at 16,384. The gap has widened again to 7.4x, and the reason is not that the GPU got worse: the CPU column improved from 0.506 s to 0.445 s on a rebuilt image with a newer numpy, while the CUDA column got slower on the same hardware. The shape of the answer has never changed across three measurements on two image builds, which is worth more than any one of the figures.

So `device="auto"` resolves to **CPU**, even when a GPU is present. Picking CUDA because a card exists would make the common case several times slower on every machine that has one, silently. The GPU is opt-in: `device="cuda"` or `FALCONAGE_DEVICE=cuda`.

When the GPU will be worth it

Not for the linear clocks that ship (35 of them bundled when this was measured). It should pay for itself on:

- **the PC clocks** - 78,464 features and coefficient tensors of 78 MB to 1.2 GB. Thirty times the feature count of the current largest, so the matmul stops being a rounding error.
- **neural architectures** - AltumAge is a multilayer perceptron, not a dot product, and has real depth to parallelise.
- **CpGPT** - a transformer, where the GPU is not optional.

None of those shipped when this was measured, which is why this document can only report that the path is correct rather than that it is fast. AltumAge has been bundled since; the PC clocks and CpGPT are licensed. Its speed on a GPU has not been measured.

Both of the first two now route through the device (section 2). That is a correctness statement, not a speed one: no weights ship for either, so there is nothing here to time. The point of routing them before the weights arrive is that the alternative is discovering on the day somebody supplies a `.safetensors` file that `device="cuda"` had been doing nothing all along.

A performance bug this found

Profiling for the GPU comparison turned up something unrelated to GPUs. Feature alignment was looping `X.iloc[:, i]` once per feature - 2,666 pandas indexing calls for one eight-clock run, 76% of total runtime, against 0.5% for the arithmetic they fed. Replacing the loop with a single `reindex` made the **CPU path 2.1× faster** at 4,096 samples (0.358 s to 0.171 s), which benefits every user whether or not they own a GPU.

It also means the earlier GPU figures were flattering: they were measured against a slow CPU baseline. The table above is after the fix.

Reproducing this

Every number on this page comes from one script, and it is in the repository:

```
docker pull bhagesh/falconage:1.0.0-cuda
docker run --rm --gpus all -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cuda \
  python test/gpu_check.py
```

Or build the image instead of pulling it, which is what produced the numbers above:

```
docker build -f docker/Dockerfile.cuda -t bhagesh/falconage:1.0.0-cuda .
```

`test/gpu_check.py` runs the five steps in the order above, what torch sees, device and dtype resolution, CPU-versus-GPU agreement, the speed and precision table, and the profile that explains it. Each step is meaningless if the one before it failed, so it stops rather than continuing. On a machine with no CUDA device it stops after step 1 with an explanation, which makes it safe to run anywhere and safe to put in a bug report.

The coverage table in section 2 is not from that script, because it needs no hardware. It is asserted by `python/tests/unit/test_device_contract.py`, which hands every model class a spec that counts how often it is reached and requires either a non-empty count or a `CPU_ONLY` declaration. The torch arithmetic

is checked against numpy in the same file, on the CPU torch backend, so a runner with no card still verifies everything except the transfer:

```
docker run --rm -v "$PWD:/work" -w /work bhagesh/falconage:1.0.0-cuda \
  python -m pytest python/tests/unit/test_device_contract.py -q
```

If the build fails at the dependency layer, the pinned uv is older than the `revision` field in `python/uv.lock` and refuses to parse it. That is what broke both images between the 2026-08-08 and 2026-08-10 runs of this page. `test/check_docker_lock.py` catches it in a second, and CI runs it, but the symptom is worth recognising: `error: Failed to parse `uv.lock``, six minutes into a build.

`--max-samples` caps the speed table for a card with less memory than the 8 GB this was measured on; `--skip-profile` drops the last step.

If your card disagrees with the tables above: the tables are what should change.

14. GPU computing

14.1 What actually benefits

Profile before optimising. The expensive operations, in order:

1. **KNN imputation** on wide matrices (n_samples × 800K probes). Dominant cost for a single dataset. `cuML.KNNImputer` or a chunked CuPy distance computation.
2. **PC-clock rotation** - a dense 78,464 × 1,933 matmul per clock, 125,175 × k for SystemsAge. Trivially GPU-bound and highly parallel across samples.
3. **Quantile normalisation** for DunedinPACE - a sort per sample over 20,000 probes. `torch.sort` over a batch is one kernel.
4. **Rank normalisation** for Pasta/REG - average ranks over 8,113 genes per sample. pyaging's current implementation is a Python `while` loop per row and is the slowest path in the package; vectorise with `torch.argsort` twice plus a scatter-mean for ties.
5. **Deconvolution simplex projection** - already vectorised in pyaging; keep that pattern.
6. **Running many clocks over many samples** - the embarrassingly parallel case, and the one pyaging's paper benchmarks.
7. **Spatial smoothing propagation** - sparse matmul iterated to convergence; cuSPARSE.
8. **Elastic net *training*** (for users fitting their own clocks), `cuML.ElasticNet`, `cuML.LogisticRegression`, or a torch implementation with proximal gradient.

14.2 Published speedups, and what we measured instead

pyaging vs biolearn on CPU (Bioinformatics 2024, Figure 1g–h):

Samples	pyaging (s)	biolearn CPU (s)	Ratio
1,024	0.233	0.386	1.7×
32,768	0.642	76.608	~120×
65,536	runs on GPU	out of memory	-

That comparison is GPU-pyaging against CPU-biolearn, so it measures two packages as much as two devices. Measuring one package on both devices gives the opposite answer for the clocks that ship here, eight bundled linear clocks over 2,340 features, on an RTX 4060, best of three:

Samples	CPU float64	CUDA float64	CUDA float32
128	**0.009 s**	0.011 s	0.013 s
1,024	**0.032 s**	0.053 s	0.051 s
4,096	**0.143 s**	0.307 s	0.190 s
16,384	**0.506 s**	2.328 s	1.726 s

The gap widens with size, which is the reverse of the usual shape, and profiling says why: at 4,096 samples the aligned matrix takes 134 ms to build and the dot products take 5 ms on the CPU against 10 ms on the GPU, most of which is transfer rather than multiply. A dot product over a few thousand features is too small a matrix multiplication to be worth a device, and the list in §14.1 is a list of things that are *not* what a first-generation linear clock does.

So the expected pattern holds only where the list in §14.1 does: the PC rotations (item 2, thirty times the feature count), neural architectures, and the single-cell paths where rapids-singlecell reports ~15× over the best CPU path and ScaleSC 20–100× over Scanpy. For the 2,340-feature case, `device="auto"` resolving to CUDA would make the common case several times slower on every machine that has a card.

Two other findings worth recording. **CPU and GPU do not agree bit for bit** - up to 1.3e-13 years, 14 ulps, because numpy's BLAS and cuBLAS sum in different orders and floating-point addition is not associative. And **single precision costs 7e-5 years** on the same device: four orders of magnitude more than the device difference, still negligible against effects measured in single-digit years, and the right way round.

14.3 Implementation pattern from pyaging

- Detect once: `torch.device("cuda" if torch.cuda.is_available() else "cpu")`.
- Optional CuPy for the AnnData matrix itself: `adata.X = cp.array(adata.X) if CUPY_AVAILABLE else np.asfortranarray(adata.X)`.
- Load every model in `float64` and `.eval()`; run under `torch.inference_mode()`.
- Batch with `anndata.experimental.pytorch.AnnLoader`, default `batch_size = 1024`.
- After each clock: `del adata.obsm[f"X_{clock}"]`, `gc.collect()`, `torch.cuda.empty_cache()`.
- Preallocate the per-clock feature matrix with CuPy when available, NumPy Fortran-order otherwise.

Caveats to fix in the new tool: `float64` doubles memory and halves throughput on consumer GPUs with poor FP64 rates, offer `float32` with a documented numerical tolerance, and keep FP64 for the PC clocks where the rotation is ill-conditioned. Several pyaging preprocess functions loop over rows in Python (`stemTOC`, `epiTOC1`, `HypoClock`, `BiTAge`, `Pasta`, `DunedinPACE`, `Stubbs`); all are vectorisable.

14.4 R side

R has no comparable GPU story for this workload. The pragmatic answer, which tAge already uses, is **reticulate**: the R package is the interface and the numerics happen in Python. Alternatives if a pure-R path is needed: `gpuR`/`torch for R` for matmuls, or shell out to a small Python worker over Arrow IPC. Whatever the mechanism, the R and Python front ends must call the same weights and the same transform code so results are bit-comparable.

Chapter 35. Design history

15. Design blueprint

The requirement was one tool, Python and R, multiomic input, optional GPU, and correct, meaning it reproduces published clock values on published test vectors.

This section was the sketch that turned that into a design. It is now documented properly elsewhere, and the sketch has been cut rather than kept in parallel: it used the abandoned working name `agingscore` and an API that no longer exists, so anyone who copied from it got nothing that runs.

Architecture is the design record for the built system, and it is checked against the running package rather than remembered. Where to look:

What the blueprint specified	Where it is documented now
Layered architecture, module boundaries	Architecture §1.4, the file tree
Data model (`X`, `obs`, `var`, `uns`, multi-modality)	Architecture §2, the shipped `FalconData`, AnnData-backed
Clock registry schema	Architecture §3, the schema as validated, field by field
API surface, Python and R	Getting started, and the Python and R references
Correctness strategy	§15.1 below, which is the part that did not reduce to an implementation detail
Packaging, distribution, CI	Architecture §9, §12
Build order and milestones	Architecture §13, with the exit criterion each milestone was held to

Two decisions from the blueprint are worth restating because they are about the field rather than about this package. Weights belong outside the git repository, in `safetensors` rather than pickles, `torch.load(weights_only=False)` executes arbitrary code while unpickling, and it is what several packages still accept. And a `license` field has to gate use and print at score time: tAge models are MGB Open Access (non-commercial), organAging is Academic Non-Commercial, and a tool that redistributes weights must surface that rather than bury it.

15.1 Correctness strategy

This is the part that decides whether a clock implementation is trusted, and it is the reason the next subsection exists at all.

1. **Gold-standard vectors.** For every clock, an input matrix and the expected output from the original implementation, asserted to 1e-6 in FP64.
2. **Cross-implementation agreement.** Where two packages implement the same clock, compare, and record disagreements in the registry rather than picking a side silently. §15.2 is that record.
3. **Registry validation** against a controlled vocabulary for tissue, platform, predicts, training target, unit, data type, species, model type and population. With 161 entries there is no other way to keep them coherent.

4. **Property tests.** Proportions sum to 1; probabilities in [0,1]; monotone response to a synthetic age gradient; permuting feature order must not change results; adding a duplicated sample must not change either sample's score.
5. **Numerical dtype tests.** FP32-versus-FP64 divergence held under a documented per-clock tolerance. The PC clocks are the ones to watch.

13. Build order

Fourteen milestones. Each has an exit criterion that is testable, not a feeling.

#	Milestone	Exit criterion
1	Repo scaffold: monorepo layout, `pyproject.toml`, `DESCRIPTION`, licences, CI skeletons	`pip install -e python/` and `R CMD INSTALL r/` both succeed; empty test suites pass on CI
2	Core engine: device, dtype, `FalconData`, manifest, logging, config, units, errors	Round-trip a `FalconData` through `.h5ad` unchanged; unit conversion table tested both directions; `resolve_device` correct on CPU and CUDA
3	Registry: schema, validator, vocabulary, weight resolution, 5 hand-written clock YAMLs	`registry-validate` green; `resolve_weights` fetches, verifies and caches; offline mode raises correctly
4	Op catalogue: all 15 preprocess ops, all 15 postprocess transforms, vectorised	Each op has a hand-computed unit test; no Python loop over samples anywhere in `ops/`
4b	Coefficient extraction: one extractor per primary source, tier assignment, diffs against pyaging/biolearn	Every tier A and B clock extracted from its own source with a checksum; `registry/diffs/` written; untraced count reported
5	Linear and PC clocks + `score()`: Horvath, Hannum, Lin, PCHorvath2013, PCPhenoAge	Gold-standard vectors pass at 1e-6 FP64; missingness accounting correct
6	Clinical clocks: PhenoAge both variants, KDM with fit/project, HD	Matches BioAge on NHANES to 1e-6; `units` omission raises; packaged NHANES fits load
7	Remaining architectures: GrimAge family, AltumAge, mitotic, deconvolution, SystemsAge, mammalian, sex	Every tier A and B clock has a passing gold-standard vector; every tier C clock has a passing synthetic-weight architecture test and a working `register_local_weights`; registry covers 161 clocks
8	Preprocess: IDAT chain, platform detection, EPIC v2 aggregation, clinical, QC	Beta matrix from a known GEO series matches the published one within noise; EPIC v2 without aggregation raises
9	Download: GEO, ArrayExpress, SRA, Zenodo, Figshare, HF, GDC; cache; metadata normalisation	`download("GSE40279", dry_run=True)` reports correct sizes; age extracted for a known series matches its published table
10	Analysis and plots: acceleration, Cox, ICC, AA1/AA2, the 14 figures	Cox HRs reproduce the BioAge NHANES IV table; AA1/AA2 reproduce a published ComputAgeBench row
11	R package: bridge, S3 classes, all exports, roxygen, vignettes	Conformance suite green at `atol=0`; `R CMD check --as-cran` clean with and without Python

#	Milestone	Exit criterion
12	Docker: cpu, cuda, slim; entrypoint; compose	Build-time smoke test passes; a GPU run on a CUDA host produces the same numbers as CPU within FP tolerance
13	Documentation: Quarto site, quartodoc, pkgdown, generated clock pages, bilingual examples	Docs build green with every example executed; both reference trees render with matching sections
14	Release 1.0.0	Version guard passes on all four files; PyPI, r-universe, GHCR and Zenodo all carry 1.0.0
15	*(v1.5)* Native clock training: elastic net, Cox elastic net and KDM fitted on the open ComputAgeBench split	A FALCONAge-native chronological and mortality clock, benchmarked against the registry, distributed under GPL with no permission constraint

Milestones 1–7 are the critical path; 4b gates 5 and 7, because a clock cannot be tested before its coefficients have a traced source. 8–10 can proceed in parallel once 2 and 3 land; 11 needs 7; 12 and 13 need 11. Milestone 15 is post-1.0 and is the answer to "can the permission-gated clocks be replaced rather than licensed", not with the same clocks, but with open ones.

14. Open questions

Weight redistribution, resolved into policy, but the letters still need writing. §3.7 sorts every clock into tier A, B or C, and §4.7 specifies what a tier C clock ships. That closes the design question. What remains is correspondence: the maintainers of the GrimAge, SystemsAge and DunedinPACE coefficient sets should be asked directly whether FALCONAge may redistribute them, and each answer recorded in the registry entry. A yes moves a clock from C to A and is worth having; a no costs nothing, because the scaffold already works with a user-supplied file.

The same question arrives again for the transcriptomic and proteomic layers: tAge models are MGB Open Access non-commercial, and organAging is an Academic Non-Commercial License. Both are likely tier B or C.

Whether the 37 package-to-package clocks can all be traced. §3.8 asserts that every tier B clock will get a primary source. Some may not have one, a coefficient file that entered circulation through a personal communication and was never published leaves no trail. Those end up as tier C by default, and the count in §3.7 could grow. The honest position is that the number is not yet known, and `primary_source_traced: false` makes the debt visible rather than hidden.

CRAN name. `available::available("FALCONAge")` has not been run. PyPI is clear for both `falconage` and `FALCONAge`. If CRAN objects, the Python name stays and only the R name moves.

How much benchmark data to vendor. ComputAgeBench is 10,404 samples across 65 studies, several gigabytes. Fetching on demand from Hugging Face keeps the package small but makes the benchmark untestable offline and in CI. Likely answer: vendor a small deterministic subset (3 studies, around 200 samples) for CI and tests, fetch the full set on demand.

IDAT parsing dependency. `methylprep` is the only maintained pure-Python IDAT reader, and it is a heavy dependency with an opinionated pipeline of its own. The alternatives are vendoring the IDAT binary parser (roughly 300 lines, well-specified format) or requiring R and `minfi` through `rpy2`. Leaning toward vendoring the parser and implementing noob and BMIQ natively, with `methylprep` as an optional accelerated path.

Which normalisation is the default. `noob + BMIQ` and SeSAMe both perform well and are not interchangeable. Defaulting to either makes results from the two paths incomparable while looking identical. Current plan is to require the choice explicitly on the first call and record it everywhere, with `noob_bmiq` as the documented default only because it is what most published clock coefficients were derived under.

Mammalian species table. The AnAge lookup that the mammalian clocks need covers 1,756 species, and its licence and redistribution terms need checking before it is packaged.

Chapter 36. Other software

12. Existing software landscape

12.1 Python

Package	Scope	GPU	Notes
pyaging	173 clocks, 5 data types, 5 species	Yes. PyTorch, CUDA auto-detect, optional CuPy	The most complete. Weights on Hugging Face `lucascamillomd/pyaging-data`, one `.pt` per clock. AnnData-native.
biolearn	39 biomarkers + data library (GEO/NHANES/ Framingham)	No	The consortium's reference implementation; strongest on data harmonisation.
computage	Benchmarking + de novo clock training (KDM, PLS)	No	The AA1/AA2 methodology and the 10,404-sample benchmark set.
organage	Oh 2023 organ proteomic clocks (SomaScan)	No	
spatial-aging-clock	18 mouse brain cell-type spatial clocks	No	SquidPy-based.
scAge	Single-cell methylation age	No	
CpGPT	Foundation model, imputation + fine-tuned heads	Yes	
racpy	RNAAgeCalc port	No	

12.2 R

Package	Scope	Notes
methylclock (Bioconductor)	Chronological clocks, gestational clocks, DNAmTL, PhenoAge	Limited scope but well-maintained.
methylCIPHER (HigginsChenLab)	PC clocks, SystemsAge, CausalAge, DunedinPACE and many more	The broadest R clock library.
dnaMethyAge	Common clocks with a tidy interface	
BioAge (dayoonkwon)	KDM, PhenoAge, homeostatic dysregulation on NHANES III/IV	The reference clinical-chemistry implementation, with plotting and survival tables.
MEAT (Bioconductor)	Muscle epigenetic age test	
**MammalMethylClock	Mammalian clock construction and application; SeSaMe normalisation, filtering, annotation, age/sex/species predictors; 11K+ samples, 59 tissues, 185 species, 19 orders	
CorticalClock	Brain cortex DNAm age	
EpiMitClocks	epiTOC1/2/3, epiCMIT, stemTOC, HypoClock	The authoritative mitotic clock source.

Package	Scope	Notes
tAge (Gladyshev-Lab)	60 transcriptomic clocks; calls Python through reticulate	MGB Open Access License, non-commercial.
scImmuAging	5 PBMC cell-type clocks	
RNAAgeCalc (Bioconductor)	GTEx transcriptomic age	

12.3 Web tools and databases

- **ClockBase** (clockbase.org) - 11 clock models reprocessed over >300,000 GEO samples, with an R package and interactive analysis.
- **TACO** (app.gladyshevlab.org/TACO). Transcriptomic Age Calculator Online; the reference application whose behaviour tAge mirrors.
- **DNA Methylation Age Calculator** (dnamage.clockfoundation.org). Horvath, SkinBlood, GrimAge.
- **Open Genes** - 2,402 age-related genes from >1,700 publications, 6 study types, 12 criteria, API + download.
- **Aging Atlas**, **AGEMAP**, **Digital Ageing Atlas**, **Human Ageing Genomic Resources** (GenAge, AnAge, LongevityMap, DrugAge, GenDR), **JenAge**.
- **aging.ai** - blood-biomarker DNN age.
- **Longevity World Cup** - open-source PhenoAge calculator and leaderboard.

12.4 The gap this project fills

No package today does all of: (a) methylation *and* transcriptomics *and* proteomics *and* clinical chemistry in one API; (b) Python *and* R with identical numerics; (c) GPU acceleration beyond pyaging's methylation focus; (d) built-in benchmarking against ComputAgeBench-style AAC data; (e) reliability-aware reporting. pyaging is closest on (a) and (c); computage owns (d); methylCIPHER and BioAge own the R side.

13. Prior implementations

Sixteen public codebases were read against the papers while this package was written. They are listed because a citation to a paper says what a method *is*, and a citation to an implementation says what its authors actually *did*, and where the two disagree, which they do in eleven documented cases: the disagreement is itself the finding.

None of it is vendored. Every clock architecture in FALCONAge is written from its published description. What these settled was the question a paper cannot answer on its own, when the text is ambiguous, which reading did the authors implement, together with the coefficient sets whose provenance could be traced back to a primary source.

Project	Origin	Language	What it settles
pyaging	lucascamillomd	Python	The broadest existing clock registry (173 entries with curated metadata and an evidence ledger), the model base classes, the preprocess and postprocess transform set, the AnnData contract, weight distribution through Hugging Face, GPU dispatch.
ComputAge	ComputationalAgingLab	Python	AA1/AA2 benchmarking, the 65-study AAC dataset table, sesame imputation vector, `LinearMethylationModel` / `GrimAgeModel`, de novo KDM and PLS estimators, biolearn-derived model definitions.
computational_aging_course	ComputationalAgingLab	Jupyter Book	Teaching-grade chapters on aging biology (12 hallmarks), aging clocks, methylomics, survival analysis, complex systems and resilience.
BioAge	dayoonkwon	R	KDM, PhenoAge (original and modified), homeostatic dysregulation, NHANES III/IV data, plotting and survival/health/SES tables. Its correction to the PhenoAge 0.090165 constant is the reading FALCONAge follows.
tAge	Gladyshev-Lab	R + Python	The 2026 Nature transcriptomic clocks, full preprocessing chain, YuGene, ortholog mapping, pseudobulk, outlier removal, relative/differential clock semantics, unit discipline for mortality vs chronological.

Project	Origin	Language	What it settles
organAging	ludgergoeminne	R + Python	Organ-specific plasma proteomic clocks (Cell Metabolism 2025), GTEx 4× fold-change organ definitions, Cox elastic net and Weibull AFT training code, log-hazard → years Gompertz conversion, ~50 plotting scripts.
SpatialAgingClock	sunericd	Python	Spatial smoothing, cell-type-specific LASSO clocks, within-animal age acceleration, cell proximity effect analysis (Nature 2025).
scImmuAging	CiiM-Bioinformatics-group	R	PBMC cell-type clocks from 1,081 donors aged 18–97 (Nat Aging 2025); pseudocell construction; LASSO/RF/PointNet training scripts.
SCALE	ChengLiLab	R	Literature-informed aging gene sets, detection-frequency-weighted z-score, single-cell entropy, four correlation measures (Pearson/dcor/MIC/nlcor).
AgingClock	Meyer-DH	Python	BiT age binarisation, the truncated-Gaussian second correction, lifespan fold-change with error propagation, v2 coefficients.
DNAm-based-age-predictor	qzhang314	R	Zhang 2019 EN (514 probes) and BLUP (319,607 probes) coefficients and the exact per-sample standardisation.
mcp-phenoage-clock	199-mcp	TypeScript	A clean, validated PhenoAge implementation with reference ranges, input bounds, mortality clamping, and interpretation strings. Good template for user-facing validation.
phenotypic-age-calc	KyteProject	Python	Minimal PhenoAge with explicit unit conversions.
Biological-Aging-and-PRS-for-Mortality-Prediction	shayanmostafaei	R	Multi-cohort mortality analysis design: TwinGene n = 9,617 (16.7 y median follow-up) discovery, UK Biobank n = 179,504 (11.83 y) validation; PhenoAge, KDM, HD, frailty index, telomere length, mvAge PRS; Cox + ROC +

Project	Origin	Language	What it settles
			SuperLearner + subgroup + interaction analyses.
Aging_clock	mdozmorov	Markdown + data	Curated literature index with clock CpG data (Levine 2018, Petkovich 2017, Wang 2017, Yang 2016 epiTOC) and Horvath's original R tutorial.
Ageing-analysis-using-R	SWAROOP006	R	Telomere repeat (TTAGGG and variants) detection across 28 genomes with seqinr. Peripheral to clock work.

Chapter 37. Appendices

Appendix A - Full pyaging clock registry (173 clocks)

Generated from pyaging clock_metadata.json. Columns: clock id, year, data type, species, tissue, predicts, training target, unit, model type, platform, n features, population, in-model preprocess, in-model postprocess, ships reference values.

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
abec	2020	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	elastic net regression	Illumina EPIC	1695	adults			
adbahadosingh	2021	DNA methylation	Homo sapiens	whole blood	late-onset Alzheimer's disease	late-onset Alzheimer's disease	probability	logistic regression	Illumina EPIC	4	older adults		sigmoid	
altumage	2022	DNA methylation	Homo sapiens	multi-tissue	chronological age	chronological age	years	deep neural network	Illumina 27K; Illumina 450K	20318	all ages	scale		ref
bitage	2021	transcriptomics	Caenorhabditis elegans	whole organism	biological age	biological age	hours	elastic net regression	RNA-seq	576	Caenorhabditis elegans	binarize		
bocklandt	2011	DNA methylation	Homo sapiens	saliva	EDARADD methylation	chronological age	beta value	single-CpG score	Illumina 27K	1	adults			
bohlin	2016	DNA methylation	Homo sapiens	cord blood	gestational age	gestational age	weeks	LASSO regression	Illumina 450K	251	newborns		days_to_weeks	
cabec	2020	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	elastic net regression	Illumina 450K; Illumina EPIC	1892	adults			
camilloh3k27ac	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net + ARD regression	ChIP-seq	1275	all ages			
camilloh3k27me3	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net + ARD regression	ChIP-seq	922	all ages			
camilloh3k36me3	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net + ARD regression	ChIP-seq	870	all ages			
camilloh3k4me1	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net + ARD regression	ChIP-seq	892	all ages			
camilloh3k4me3	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net	ChIP-seq	1240	all ages			

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
								+ ARD regression						
camilloh3k9ac	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net + ARD regression	ChIP-seq	102	all ages			
camilloh3k9me3	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net + ARD regression	ChIP-seq	341	all ages			
camillopanhistone	2025	histone modification	Homo sapiens	multi-tissue	chronological age	chronological age	years	PCA + elastic net + ARD regression	ChIP-seq	3739	all ages			
cellpopage	2024	DNA methylation	Homo sapiens	cultured fibroblasts	cell-population passage age	cell passage number	passages	elastic net regression	Illumina EPIC	42	human cell cultures			
compil6	2021	DNA methylation	Homo sapiens	whole blood	interleukin-6	interleukin-6	unitless	elastic net regression	Illumina 450K	35	older adults			
corticalclock	2020	DNA methylation	Homo sapiens	brain cortex	chronological age	chronological age	years	elastic net regression	Illumina 450K	347	human, age unspecified		anti_log_linear	ref
cpgptgrimage3	2024	DNA methylation	Homo sapiens	whole blood	biological age; mortality risk	mortality	years	Cox proportional hazards regression	Illumina 450K	24	adults	scale	cox_to_years	
cpgptpcgrimage3	2024	DNA methylation	Homo sapiens	whole blood	biological age; mortality risk	mortality	years	PCA + Cox regression	Illumina 450K	31	adults	scale	cox_to_years	
ctsliver	2024	DNA methylation	Homo sapiens	liver	chronological age	chronological age	years	LASSO regression	Illumina EPIC	90	adults			
cvdwesterman	2020	DNA methylation	Homo sapiens	whole blood	cardiovascular disease risk	cardiovascular disease	probability	elastic net Cox ensemble	Illumina 450K	235	adults		sigmoid	
deconvolutebloodepicbcell	2018	DNA methylation	Homo sapiens	purified blood leukocytes	B cell proportion	cell-type-specific methylation contrast	proportion	reference-based constrained deconvolution	Illumina EPIC	600	adults	fill_with_reference_means		ref
deconvolutebloodepic cd4tcell	2018	DNA methylation	Homo sapiens	purified blood leukocytes	CD4+ T cell proportion	cell-type-specific methyla-	proportion	reference-based constrained	Illumina EPIC	600	adults	fill_with_reference_means		ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
						tion contrast		deconvolution						
deconvoluteblood-epic cd8tcell	2018	DNA methylation	Homo sapiens	purified blood leukocytes	CD8+ T cell proportion	cell-type-specific methylation contrast	proportion	reference-based constrained deconvolution	Illumina EPIC	600	adults	fill_with_reference_means		ref
deconvolut ebloode-picmonocyte	2018	DNA methylation	Homo sapiens	purified blood leukocytes	monocyte proportion	cell-type-specific methylation contrast	proportion	reference-based constrained deconvolution	Illumina EPIC	600	adults	fill_with_reference_means		ref
deconvoluteblood-epicneutrophil	2018	DNA methylation	Homo sapiens	purified blood leukocytes	neutrophil proportion	cell-type-specific methylation contrast	proportion	reference-based constrained deconvolution	Illumina EPIC	600	adults	fill_with_reference_means		ref
deconvoluteblood-epicnkcell	2018	DNA methylation	Homo sapiens	purified blood leukocytes	natural killer cell proportion	cell-type-specific methylation contrast	proportion	reference-based constrained deconvolution	Illumina EPIC	600	adults	fill_with_reference_means		ref
depressionbarbu	2021	DNA methylation	Homo sapiens	whole blood	major depressive disorder	major depressive disorder	unitless	elastic net regression	Illumina EPIC	196	adults			
dnamfili	2022	DNA methylation	Homo sapiens	whole blood	frailty risk	frailty	unitless	LASSO regression	Illumina EPIC; Illumina 450K	20	older adults			
dnamfitage	2023	DNA methylation	Homo sapiens	whole blood	physical-fitness biological age	biological age	years	Klemera–Doubal composite	Illumina 450K	630	adults			ref
dnamfitagegaitf	2023	DNA methylation	Homo sapiens	whole blood	gait speed	gait speed	meters per second	LASSO regression	Illumina 450K	53	adult women			ref
dnamfitagegaitm	2023	DNA methylation	Homo sapiens	whole blood	gait speed	gait speed	meters per second	LASSO regression	Illumina 450K	59	adult men			ref
dnamfitagegripf	2023	DNA methylation	Homo sapiens	whole blood	grip strength	grip strength	kilograms	LASSO regression	Illumina 450K	91	adult women			ref
dnamfitagegripm	2023	DNA methylation	Homo sapiens	whole blood	grip strength	grip strength	kilograms	LASSO regression	Illumina 450K	93	adult men			ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
dnamfitage vo2max	2023	DNA methylation	Homo sapiens	whole blood	VO2max	VO2max	milliliters per kilogram per minute	LASSO regression	Illumina 450K	41	adults			ref
dnamic	2025	DNA methylation	Homo sapiens	whole blood	intrinsic capacity	intrinsic capacity	unitless	elastic net regression	Illumina EPIC	91	older adults			
dnamphenoage	2018	DNA methylation	Homo sapiens	whole blood	phenotypic age	phenotypic age	years	elastic net regression	Illumina 27K; Illumina 450K; Illumina EPIC	513	adults			
dnamstress	2023	DNA methylation	Homo sapiens	whole blood	stress exposure	stress exposure	unitless	bootstrap-stabilized elastic net regression	Illumina EPIC	211	adults			
dnamtl	2019	DNA methylation	Homo sapiens	whole blood	leukocyte telomere length	leukocyte telomere length	kilobases	elastic net regression	Illumina 450K; Illumina EPIC	140	adults			
downsyndrome	2021	DNA methylation	Homo sapiens	neonatal blood spots	Down syndrome methylation score	not applicable	unitless	weighted linear score	Illumina EPIC	652	newborns			
dunedinpace	2022	DNA methylation	Homo sapiens	whole blood	pace of aging	pace of aging	biological years per chronological year	elastic net regression	Illumina EPIC	20000	adults	quantile_normalization_with_gold_standard		ref
dunedinpoam38	2020	DNA methylation	Homo sapiens	whole blood	pace of aging	pace of aging	biological years per chronological year	elastic net regression	Illumina 450K	46	adults			
eabec	2020	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	elastic net regression	Illumina EPIC	1791	adults			
encen100	2023	DNA methylation	Homo sapiens	whole blood; saliva; buccal epithelium	chronological age	chronological age	years	elastic net regression	Illumina 450K; Illumina EPIC	198	centenarians			
encen40	2023	DNA methylation	Homo sapiens	whole blood; saliva;	chronological age	chronological age	years	elastic net regression	Illumina 450K; Illumina EPIC	559	older adults			

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
				buccal epithelium										
ensembleagehumanmouse	2025	DNA methylation	Homo sapiens and Mus musculus	multi-tissue; whole blood	relative age	relative age	relative age	elastic net regression	mammalian methylation array	100	humans and mice			
ensembleagestatic	2025	DNA methylation	Mus musculus	multi-tissue	interventi on-responsive epigenetic age	interventi on-responsive epigenetic age	years	elastic net regression	Horvath Mammalmeth ylChip40; Horvath Mammalmeth ylChip320	288	mice			
ensembleagestatictop	2025	DNA methylation	Mus musculus	multi-tissue	interventi on-responsive epigenetic age	interventi on-responsive epigenetic age	years	elastic net regression	Horvath Mammalmeth ylChip40; Horvath Mammalmeth ylChip320	431	mice			
epicga	2021	DNA methylation	Homo sapiens	cord blood	gestational age	gestational age	days	LASSO regression	Illumina EPIC	176	newborns		days_to_weeks	
epicmithyper	2020	DNA methylation	Homo sapiens	B cells	mitotic age	replicative history	proportion	mean methylation aggregation	Illumina 450K; Illumina EPIC	184	human, age unspecified	mean		ref
epicmithypo	2020	DNA methylation	Homo sapiens	B cells	mitotic age	replicative history	proportion	complement of mean methylation	Illumina 450K; Illumina EPIC	1164	human, age unspecified	mean		ref
epitoc1	2016	DNA methylation	Homo sapiens	multi-tissue; whole blood	mitotic age	chronological age	beta value	mean methylation aggregation	Illumina 450K	385	all ages	mean		ref
epitoc2	2020	DNA methylation	Homo sapiens	whole blood	mitotic age	chronological age	cell divisions per stem cell	dynamic methylation transmission model	Illumina 450K	163	adults	nan_to_zero		ref
epitoc3	2020	DNA methylation	Homo sapiens	cultured primary human cells; whole blood; multi-tis-	mitotic age	population doublings	cell divisions per stem cell	dynamic methylation transmission model	Illumina 450K; Illumina EPIC	170	all ages	nan_to_zero		ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
				sue; cord blood										
garagnani	2012	DNA methylation	Homo sapiens	whole blood	ELOVL2 methylation	not applicable	beta value	single-CpG score	Illumina 450K	1	all ages			
gliasin	2024	DNA methylation	Homo sapiens	brain cortex	chronological age	chronological age	years	elastic net regression	Illumina 450K	220	adults			
grimage	2019	DNA methylation	Homo sapiens	whole blood	mortality risk	mortality	years	two-stage elastic net + Cox regression	Illumina 450K	1032	adults		cox_to_years	ref
grimage2	2022	DNA methylation	Homo sapiens	whole blood	mortality risk	mortality	years	elastic net Cox regression	Illumina 450K	1032	older adults		cox_to_years	ref
grimage2adm	2022	DNA methylation	Homo sapiens	whole blood	adrenomedullin	adrenomedullin	picograms per milliliter	elastic net regression	Illumina 450K	187	older adults		cox_to_years	ref
grimage2b2m	2022	DNA methylation	Homo sapiens	whole blood	beta-2-microglobulin	beta-2-microglobulin	picograms per milliliter	elastic net regression	Illumina 450K	92	older adults		cox_to_years	ref
grimage2cystatinc	2022	DNA methylation	Homo sapiens	whole blood	cystatin C	cystatin C	picograms per milliliter	elastic net regression	Illumina 450K	88	older adults		cox_to_years	ref
grimage2gdf15	2022	DNA methylation	Homo sapiens	whole blood	GDF-15	GDF-15	picograms per milliliter	elastic net regression	Illumina 450K	138	older adults		cox_to_years	ref
grimage2leptin	2022	DNA methylation	Homo sapiens	whole blood	leptin	leptin	picograms per milliliter	elastic net regression	Illumina 450K	187	older adults		cox_to_years	ref
grimage2loga1c	2022	DNA methylation	Homo sapiens	whole blood	hemoglobin A1c	hemoglobin A1c	natural-log percent	elastic net regression	Illumina 450K	87	older adults		cox_to_years	ref
grimage2logcrp	2022	DNA methylation	Homo sapiens	whole blood	C-reactive protein	C-reactive protein	natural-log milligrams per liter	elastic net regression	Illumina 450K	132	older adults		cox_to_years	ref
grimage2packyrs	2022	DNA methylation	Homo sapiens	whole blood	smoking exposure	smoking exposure	pack-years	elastic net regression	Illumina 450K	173	older adults		cox_to_years	ref
grimage2pai1	2022	DNA methylation	Homo sapiens	whole blood	PAI-1	PAI-1	picograms per milliliter	elastic net regression	Illumina 450K	211	older adults			ref
grimage2timp1	2022	DNA methylation	Homo sapiens	whole blood	TIMP-1	TIMP-1	picograms per milliliter	elastic net regression	Illumina 450K	43	older adults		cox_to_years	ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
han	2020	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	linear regression	Illumina 450K	65	all ages		anti_log_linear	
hannum	2013	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	elastic net regression	Illumina 450K	71	adults			
hep	2024	DNA methylation	Homo sapiens	liver	chronological age	chronological age	years	LASSO regression	Illumina EPIC	70	adults			
hepatoxu	2017	DNA methylation	Homo sapiens	plasma cell-free DNA	hepatocellular carcinoma	hepatocellular carcinoma	unitless	feature-selected logistic regression	targeted bisulfite sequencing	10	adults			
horvath2013	2013	DNA methylation	Homo sapiens	multi-tissue	chronological age	chronological age	years	elastic net regression	Illumina 27K; Illumina 450K	353	all ages		anti_log_linear	ref
hrsinch-phenoage	2022	DNA methylation	Homo sapiens	whole blood	phenotypic age	phenotypic age	years	weighted linear score	Illumina 450K; Illumina EPIC	959	adults			
hypoclock	2020	DNA methylation	Homo sapiens	multi-tissue	mitotic age	not applicable	beta value	mean aggregation	Illumina 450K	678	human, age unspecified	mean	one_minus	ref
intrinclock	2024	DNA methylation	Homo sapiens	multi-tissue	chronological age	chronological age	years	two-stage elastic net regression	Illumina 450K; Illumina EPIC	380	all ages		anti_log_linear	
knight	2016	DNA methylation	Homo sapiens	cord blood; neonatal blood spots	gestational age	gestational age	weeks	elastic net regression	Illumina 27K; Illumina 450K	148	newborns			ref
leecontrol	2019	DNA methylation	Homo sapiens	placenta	gestational age	gestational age	weeks	elastic net regression	Illumina 450K; Illumina EPIC	546	pregnancies			
leerefinedrobust	2019	DNA methylation	Homo sapiens	placenta	gestational age	gestational age	weeks	elastic net regression	Illumina 450K; Illumina EPIC	395	pregnancies			
leerobust	2019	DNA methylation	Homo sapiens	placenta	gestational age	gestational age	weeks	elastic net regression	Illumina 450K; Illumina EPIC	558	pregnancies			

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
lin	2016	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	linear regression	Illumina 27K; Illumina 450K	99	adults			
mammalian1	2023	DNA methylation	multiple species	multi-tissue	chronological age	chronological age	years	elastic net regression	Horvath MammalMethylChip40	335	multiple mammalian species		anti_logp2	
mammalian2	2023	DNA methylation	multiple species	multi-tissue	chronological age	relative age	years	elastic net regression	Horvath MammalMethylChip40	2572	multiple mammalian species		mammalian2	ref
mammalian3	2023	DNA methylation	multiple species	multi-tissue	chronological age	chronological age	years	elastic net regression	Horvath MammalMethylChip40	2467	multiple mammalian species		mammalian3	ref
mammalianblood2	2023	DNA methylation	multiple species	blood	chronological age	relative age	years	elastic net regression	Horvath MammalMethylChip40	2257	multiple mammalian species		mammalian2	ref
mammalianblood3	2023	DNA methylation	multiple species	blood	chronological age	chronological age	years	elastic net regression	Horvath MammalMethylChip40	2097	multiple mammalian species		mammalian3	ref
mammalianfemale	2023	DNA methylation	multiple species	multi-tissue	sex	sex	probability	elastic net regression	mammalian methylation array	101	multiple mammalian species		sigmoid	
mammalian-lifespan	2024	DNA methylation	multiple species	multi-tissue	species maximum lifespan	species maximum lifespan	years	elastic net regression	Horvath MammalMethylChip40	152	multiple mammalian species		anti_log	ref
mammalianskin2	2023	DNA methylation	multiple species	skin	chronological age	relative age	years	elastic net regression	Horvath MammalMethylChip40	2240	multiple mammalian species		mammalian2	ref
mammalianskin3	2023	DNA methylation	multiple species	skin	chronological age	chronological age	years	elastic net regression	Horvath MammalMethylChip40	2055	multiple mammalian species		mammalian3	ref
mayne	2017	DNA methylation	Homo sapiens	placenta	gestational age	gestational age	weeks	elastic net regression	Illumina 27K; Illumina 450K	62	pregnancies			
mccartneyalcohol	2018	DNA methylation	Homo sapiens	whole blood	alcohol consumption	alcohol consumption	units per week	LASSO regression	Illumina EPIC	450	adults			

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
mccartneybmi	2018	DNA methylation	Homo sapiens	whole blood	body mass index	body mass index	unitless	LASSO regression	Illumina EPIC	1109	adults		sigmoid	
mccartneybodyfat	2018	DNA methylation	Homo sapiens	whole blood	body fat percentage	body fat percentage	unitless	LASSO regression	Illumina EPIC	968	adults		sigmoid	
mccartneyeducation	2018	DNA methylation	Homo sapiens	whole blood	educational attainment	educational attainment	unitless	LASSO regression	Illumina EPIC	373	adults		sigmoid	
mccartneyhdlcholesterol	2018	DNA methylation	Homo sapiens	whole blood	HDL cholesterol	HDL cholesterol	unitless	LASSO regression	Illumina EPIC	737	adults		sigmoid	
mccartneyldlcholesterol	2018	DNA methylation	Homo sapiens	whole blood	LDL cholesterol	LDL cholesterol	unitless	LASSO regression	Illumina EPIC	233	adults		sigmoid	
mccartneysmoking	2018	DNA methylation	Homo sapiens	whole blood	smoking exposure	smoking exposure	pack-years	LASSO regression	Illumina EPIC	233	adults			
mccartneytotalcholesterol	2018	DNA methylation	Homo sapiens	whole blood	total cholesterol	total cholesterol	unitless	LASSO regression	Illumina EPIC	204	adults		sigmoid	
mccartneytotalhdlratio	2018	DNA methylation	Homo sapiens	whole blood	total-to-HDL cholesterol ratio	total-to-HDL cholesterol ratio	ratio	LASSO regression	Illumina EPIC	412	adults			
mccartneywhr	2018	DNA methylation	Homo sapiens	whole blood	waist-to-hip ratio	waist-to-hip ratio	ratio	LASSO regression	Illumina EPIC	226	adults			
meer	2018	DNA methylation	Mus musculus	multi-tissue	chronological age	chronological age	days	elastic net regression	RRBS	435	mice			
neusin	2024	DNA methylation	Homo sapiens	brain cortex	chronological age	chronological age	years	elastic net regression	Illumina 450K	672	adults			
ocampoatac1	2023	chromatin accessibility	Homo sapiens	peripheral blood mononuclear cells	chronological age	chronological age	years	elastic net regression	ATAC-seq	228	adults	tpm_norm_log1p		
ocampoatac2	2023	chromatin accessibility	Homo sapiens	peripheral blood mononuclear cells	chronological age	chronological age	years	elastic net regression	ATAC-seq	380	adults	tpm_norm_log1p		
pasta	2025	transcriptomics	Homo sapiens	multi-tissue	transcriptomic age	age ordering	years	ridge logistic regression	RNA-seq; gene expression microarray	8113	human, age unspecified	median_fill_and_rank_normalization	scale_and_shift	ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
pasta-mouse	2025	transcriptomics	Mus musculus	multi-tissue	transcriptomic age	age ordering	years	orthologue-transferred ridge logistic regression	RNA-seq; gene expression microarray	1600	human, age unspecified	median_fill_and_rank_normalization	scale_and_shift	ref
pcdnamtl	2022	DNA methylation	Homo sapiens	whole blood	leukocyte telomere length	DNAmTL output	base pairs	PCA + elastic net regression	Illumina 450K	78464	adults			ref
pcgrimage	2022	DNA methylation	Homo sapiens	whole blood	mortality risk	DNAm GrimAge output	years	PCA + elastic net regression	Illumina 450K	78466	adults			ref
pchannum	2022	DNA methylation	Homo sapiens	whole blood	chronological age	Hannum clock output	years	PCA + elastic net regression	Illumina 450K	78464	adults			ref
pchorvath2013	2022	DNA methylation	Homo sapiens	multi-tissue	chronological age	Horvath clock output	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	78464	all ages		anti_log_linear	ref
pcphenoage	2022	DNA methylation	Homo sapiens	whole blood	phenotypic age	phenotypic age	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	78464	adults			ref
pcskinandblood	2022	DNA methylation	Homo sapiens	skin; whole blood; cultured fibroblasts	chronological age	skin-and-blood clock output	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	78464	all ages		anti_log_linear	ref
pedbe	2020	DNA methylation	Homo sapiens	buccal epithelium	chronological age	chronological age	years	elastic net regression	Illumina 450K; Illumina EPIC	94	children and adolescents		anti_log_linear	
petkovich	2017	DNA methylation	Mus musculus	whole blood	chronological age	chronological age	months	elastic net regression	bisulfite sequencing	90	mice		petkovich	
phenoage	2018	clinical biomarkers	Homo sapiens	blood	phenotypic age	mortality	years	penalized hazards regression with Gompertz calibration	clinical laboratory assays	10	adults		mortality_to_phenoage	
pipekelasticnet	2022	DNA methylation	Homo sapiens	multi-tissue	chronological age	chronological age	years	elastic net regression	Illumina 27K; Illumina 450K; Il-	239	all ages		anti_log_linear	

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
									lumina EPIC					
pipekfilteredh	2022	DNA methylation	Homo sapiens	multi-tissue	chronological age	chronological age	years	elastic net regression	Illumina 27K; Illumina 450K; Illumina EPIC	272	all ages		anti_log_linear	
pipekretrainedh	2022	DNA methylation	Homo sapiens	multi-tissue	chronological age	chronological age	years	linear regression	Illumina 27K; Illumina 450K; Illumina EPIC	308	all ages		anti_log_linear	
prostatecancerkirby	2017	DNA methylation	Homo sapiens	prostate	prostate cancer	prostate cancer	log odds	logistic regression	Illumina 450K	3	adult men			
reedbmi	2020	DNA methylation	Homo sapiens	whole blood; cord blood	BMI methylation score	body mass index	unitless	weighted methylation aggregation	Illumina 450K	135	all ages			
reg	2025	transcriptomics	Homo sapiens	multi-tissue	chronological age	chronological age	years	ridge regression	RNA-seq; gene expression microarray	8113	human, age unspecified	median_fill_and_rank_normalization	add_constant	ref
replitali	2022	DNA methylation	Homo sapiens	cultured primary human cells	replicative history	population doublings	population doublings	elastic net regression	Illumina EPIC	87	human cell cultures			
replitalinorm	2022	DNA methylation	Homo sapiens	cultured fibroblasts	replicative history	population doublings	population doublings	elastic net regression	Illumina EPIC	218	human cell cultures			
retroelementagev1	2024	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	elastic net regression	Illumina EPIC	1317	all ages			
retroelementagev2	2024	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	elastic net regression	Illumina EPIC	1378	all ages			
senchronoage	2026	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	elastic net regression	Illumina 450K; Illumina EPIC	187	adults			
senculture age	2026	DNA methylation	Homo sapiens	cultured fibroblasts; cultured mesenchymal	cellular senescence	cellular senescence	log odds	elastic net logistic regression	Illumina 450K; Illumina EPIC	142	human cell cultures			

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
				stromal cells										
senmortalityage	2026	DNA methylation	Homo sapiens	whole blood	mortality risk	mortality	log hazard	elastic net Cox regression	Illumina 450K; Illumina EPIC	91	adults			
skinandblood	2018	DNA methylation	Homo sapiens	buccal epithelium; whole blood; epithelium; cultured fibroblasts; skin; cord blood	chronological age	chronological age	years	elastic net regression	Illumina 450K; Illumina EPIC	391	all ages		anti_log_linear	
stemtoc	2024	DNA methylation	Homo sapiens	multi-tissue; cultured human cells; whole blood	mitotic age	population doublings; chronological age	beta value	95th-percentile methylation aggregation	Illumina 450K; Illumina EPIC	371	all ages	0.95 quantile		ref
stemtocvitro	2024	DNA methylation	Homo sapiens	multi-tissue; cultured human cells	mitotic age	population doublings	beta value	95th-percentile methylation aggregation	Illumina 450K; Illumina EPIC	629	prenatal and newborn	0.95 quantile		ref
stoch	2024	DNA methylation	Homo sapiens	sorted monocytes	chronological age	chronological age	years	elastic net regression	Illumina 450K	353	adults			
stocp	2024	DNA methylation	Homo sapiens	sorted monocytes	chronological age	chronological age	years	elastic net regression	Illumina 450K	513	adults			
stocz	2024	DNA methylation	Homo sapiens	sorted monocytes	chronological age	chronological age	years	elastic net regression	Illumina 450K	514	adults			
stubbs	2017	DNA methylation	Mus musculus	multi-tissue; liver; lung; heart; brain cortex; skeletal muscle; cerebellum; spleen	chronological age	chronological age	months	quadratically calibrated elastic net regression	RRBS	17992	mice	quantile_normalization_and_scale_with_gold_standard	stubbs	ref
systemsage	2025	DNA methylation	Homo sapiens	whole blood	multisystem biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
system-sageblood	2025	DNA methylation	Homo sapiens	whole blood	blood-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sagebrain	2025	DNA methylation	Homo sapiens	whole blood	brain-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sageheart	2025	DNA methylation	Homo sapiens	whole blood	cardiovascular-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sagehormone	2025	DNA methylation	Homo sapiens	whole blood	endocrine-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sageimmune	2025	DNA methylation	Homo sapiens	whole blood	immune-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sageinflammation	2025	DNA methylation	Homo sapiens	whole blood	inflammatory-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sagekidney	2025	DNA methylation	Homo sapiens	whole blood	renal-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sageliver	2025	DNA methylation	Homo sapiens	whole blood	hepatic-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sagelung	2025	DNA methylation	Homo sapiens	whole blood	pulmonary-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sagemetabolic	2025	DNA methylation	Homo sapiens	whole blood	metabolic-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref
system-sagemusculoskeletal	2025	DNA methylation	Homo sapiens	whole blood	musculoskeletal-system biological age	mortality	years	PCA + elastic net regression	Illumina 450K; Illumina EPIC	125175	older adults			ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
thompson	2018	DNA methylation	Mus musculus	adipose tissue; blood; cerebellum; brain cortex; heart; kidney; liver; lung; skeletal muscle; spleen	chronological age	chronological age	months	elastic net regression	RRBS	582	mice			
twelvecell deconvolutebloodepicbas	2022	DNA methylation	Homo sapiens	purified blood leukocytes	basophil proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepicbmem	2022	DNA methylation	Homo sapiens	purified blood leukocytes	memory B cell proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepicbnv	2022	DNA methylation	Homo sapiens	purified blood leukocytes	naive B cell proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepic cd4mem	2022	DNA methylation	Homo sapiens	purified blood leukocytes	memory CD4+ T cell proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepiccd4nv	2022	DNA methylation	Homo sapiens	purified blood leukocytes	naive CD4+ T cell proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepic cd8mem	2022	DNA methylation	Homo sapiens	purified blood leukocytes	memory CD8+ T cell proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepiccd8nv	2022	DNA methylation	Homo sapiens	purified blood leukocytes	naive CD8+ T cell proportion	cell-type proportions	proportion	reference-based constrained	Illumina EPIC	240	adults	fill_with_reference_means		ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
								deconvolution						
twelvecell deconvolutebloodepiceos	2022	DNA methylation	Homo sapiens	purified blood leukocytes	eosinophil proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepicmono	2022	DNA methylation	Homo sapiens	purified blood leukocytes	monocyte proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepicneu	2022	DNA methylation	Homo sapiens	purified blood leukocytes	neutrophil proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepicnk	2022	DNA methylation	Homo sapiens	purified blood leukocytes	natural killer cell proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
twelvecell deconvolutebloodepictreg	2022	DNA methylation	Homo sapiens	purified blood leukocytes	regulatory T-cell proportion	cell-type proportions	proportion	reference-based constrained deconvolution	Illumina EPIC	240	adults	fill_with_reference_means		ref
vidalbralo	2016	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	linear regression	Illumina 27K	8	adults			
weidner	2014	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	linear regression	Illumina 27K; bisulfite sequencing	3	adults			
wu	2019	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	screened elastic net regression	Illumina 27K; Illumina 450K	111	children		anti_log_linear	
xchrom	2021	DNA methylation	Homo sapiens	whole blood	X-chromosome dosage	X-chromosome dosage	unitless	principal component analysis	Illumina 450K; Illumina EPIC	4047	adults	sex_estimation *autosomal* zscore		ref

clock	yr	data	species	tissue	predicts	trained_on	unit	model	platform	n_feat	pop	pre	post	ref
ychrom	2021	DNA methylation	Homo sapiens	whole blood	Y-chromosome presence	Y-chromosome presence	unitless	principal component analysis	Illumina 450K; Illumina EPIC	284	adults	sex_estimation *autosomal* zscore		ref
yingadaptage	2024	DNA methylation	Homo sapiens	whole blood	adaptive epigenetic age	chronological age	years	causality-weighted elastic net regression	Illumina 450K	999	adults			
yingcausage	2024	DNA methylation	Homo sapiens	whole blood	chronological age	chronological age	years	causality-weighted elastic net regression	Illumina 450K	585	adults			
yingdamage	2024	DNA methylation	Homo sapiens	whole blood	damaging epigenetic age	chronological age	years	causality-weighted elastic net regression	Illumina 450K	1089	adults			
zhangblup	2019	DNA methylation	Homo sapiens	whole blood; saliva	chronological age	chronological age	years	best linear unbiased prediction	Illumina 450K; Illumina EPIC	319607	all ages	scale_row		ref
zhangen	2019	DNA methylation	Homo sapiens	whole blood; saliva	chronological age	chronological age	years	elastic net regression	Illumina 450K; Illumina EPIC	514	all ages	scale_row		ref
zhangmortality	2017	DNA methylation	Homo sapiens	whole blood	mortality risk	mortality	unitless	weighted linear score	Illumina 450K	10	older adults			

Appendix B - tAge transcriptomic clock registry (60 models)

From tAge clocks_metadata.csv. Weights on Zenodo record 18763485. `lifespan_scaled = TRUE` means the raw prediction is multiplied by the species maximum lifespan (human 122.5, mouse 48, rat 50.4, monkey 39) to reach age units; FALSE means the output is already on its native scale, log10(hazard ratio) for mortality clocks, or a fraction of maximum lifespan for normalized-age clocks. `BR` is Bayesian Ridge, `EN` is Elastic Net. `Scaled` clocks take per-sample z-scored log10 RLE expression; `YuGene` clocks take the YuGene uniformised version. All are relative models and require reference centring.

filename	type	outcome	species	tissue	scaling	lifespan_scaled
BR_Chronoage_Mouse_Multitissue_scaleddiff.pkl	BR	Chronological	Mouse	Multi-Tissue	Scaled	TRUE
BR_Chronoage_Mouse_Multitissue_yugenediff.pkl	BR	Chronological	Mouse	Multi-Tissue	YuGene	TRUE
BR_Chronoage_Multispecies_Multitissue_scaleddiff.pkl	BR	Chronological	Multispecies	Multi-Tissue	Scaled	TRUE
BR_Chronoage_Multispecies_Multitissue_yugenediff.pkl	BR	Chronological	Multispecies	Multi-Tissue	YuGene	TRUE
BR_Chronoage_Rodents_Brain_WT_scaleddiff.pkl	BR	Chronological	Rodents	Brain	Scaled	TRUE
BR_Chronoage_Rodents_Brain_WT_yugenediff.pkl	BR	Chronological	Rodents	Brain	YuGene	TRUE
BR_Chronoage_Rodents_Kidney_WT_scaleddiff.pkl	BR	Chronological	Rodents	Kidney	Scaled	TRUE
BR_Chronoage_Rodents_Kidney_WT_yugenediff.pkl	BR	Chronological	Rodents	Kidney	YuGene	TRUE
BR_Chronoage_Rodents_Liver_scaleddiff.pkl	BR	Chronological	Rodents	Liver	Scaled	TRUE
BR_Chronoage_Rodents_Liver_yugenediff.pkl	BR	Chronological	Rodents	Liver	YuGene	TRUE

filename	type	outcome	species	tissue	scaling	lifespan_scaled
BR_Chronoage_ Rodents_Multitissue_ scaleddiff.pkl	BR	Chronological	Rodents	Multi-Tissue	Scaled	TRUE
BR_Chronoage_ Rodents_Multitissue_ yugenediff.pkl	BR	Chronological	Rodents	Multi-Tissue	YuGene	TRUE
BR_Chronoage_ Rodents_Skeletal muscle_WT_yugened- iff.pkl	BR	Chronological	Rodents	Skeletal Muscle	YuGene	TRUE
BR_Chronoage_ Rodents_Skeletal_ muscle_WT_scaleddiff. pkl	BR	Chronological	Rodents	Skeletal Muscle	Scaled	TRUE
BR_Mortality_Mouse_ Multitissue_scaleddiff. pkl	BR	Mortality	Mouse	Multi-Tissue	Scaled	FALSE
BR_Mortality_Mouse_ Multitissue_yugenediff .pkl	BR	Mortality	Mouse	Multi-Tissue	YuGene	FALSE
BR_Mortality_Multi- species_Multitissue_ scaleddiff.pkl	BR	Mortality	Multispecies	Multi-Tissue	Scaled	FALSE
BR_Mortality_Multi- species_Multitissue_ yugenediff.pkl	BR	Mortality	Multispecies	Multi-Tissue	YuGene	FALSE
BR_Mortality_ Rodents_Liver_scaled- diff.pkl	BR	Mortality	Rodents	Liver	Scaled	FALSE
BR_Mortality_ Rodents_Liver_yu- genediff.pkl	BR	Mortality	Rodents	Liver	YuGene	FALSE
BR_Mortality_ Rodents_Multitissue_ scaleddiff.pkl	BR	Mortality	Rodents	Multi-Tissue	Scaled	FALSE
BR_Mortality_ Rodents_Multitissue_ yugenediff.pkl	BR	Mortality	Rodents	Multi-Tissue	YuGene	FALSE

filename	type	outcome	species	tissue	scaling	lifespan_scaled
BR_NormalizedAge_Mouse_Multitissue_scaleddiff.pkl	BR	Normalized age	Mouse	Multi-Tissue	Scaled	FALSE
BR_NormalizedAge_Mouse_Multitissue_yugenediff.pkl	BR	Normalized age	Mouse	Multi-Tissue	YuGene	FALSE
BR_NormalizedAge_Multispecies_Multitissue_scaleddiff.pkl	BR	Normalized age	Multispecies	Multi-Tissue	Scaled	FALSE
BR_NormalizedAge_Multispecies_Multitissue_yugenediff.pkl	BR	Normalized age	Multispecies	Multi-Tissue	YuGene	FALSE
BR_NormalizedAge_Rodents_Liver_scaleddiff.pkl	BR	Normalized age	Rodents	Liver	Scaled	FALSE
BR_NormalizedAge_Rodents_Liver_yugenediff.pkl	BR	Normalized age	Rodents	Liver	YuGene	FALSE
BR_NormalizedAge_Rodents_Multitissue_scaleddiff.pkl	BR	Normalized age	Rodents	Multi-Tissue	Scaled	FALSE
BR_NormalizedAge_Rodents_Multitissue_yugenediff.pkl	BR	Normalized age	Rodents	Multi-Tissue	YuGene	FALSE
EN_Chronoage_Mouse_Multitissue_scaleddiff.pkl	EN	Chronological	Mouse	Multi-Tissue	Scaled	TRUE
EN_Chronoage_Mouse_Multitissue_yugenediff.pkl	EN	Chronological	Mouse	Multi-Tissue	YuGene	TRUE
EN_Chronoage_Multispecies_Multitissue_scaleddiff.pkl	EN	Chronological	Multispecies	Multi-Tissue	Scaled	TRUE
EN_Chronoage_Multispecies_Multitissue_yugenediff.pkl	EN	Chronological	Multispecies	Multi-Tissue	YuGene	TRUE
EN_Chronoage_Rodents_Brain_scaleddiff.pkl	EN	Chronological	Rodents	Brain	Scaled	TRUE

filename	type	outcome	species	tissue	scaling	lifespan_scaled
EN_Chronoage_Rodents_Brain_yugenediff.pkl	EN	Chronological	Rodents	Brain	YuGene	TRUE
EN_Chronoage_Rodents_Kidney_scaleddiff.pkl	EN	Chronological	Rodents	Kidney	Scaled	TRUE
EN_Chronoage_Rodents_Kidney_yugenediff.pkl	EN	Chronological	Rodents	Kidney	YuGene	TRUE
EN_Chronoage_Rodents_Liver_scaleddiff.pkl	EN	Chronological	Rodents	Liver	Scaled	TRUE
EN_Chronoage_Rodents_Liver_yugenediff.pkl	EN	Chronological	Rodents	Liver	YuGene	TRUE
EN_Chronoage_Rodents_Multitissue_scaleddiff.pkl	EN	Chronological	Rodents	Multi-Tissue	Scaled	TRUE
EN_Chronoage_Rodents_Multitissue_yugenediff.pkl	EN	Chronological	Rodents	Multi-Tissue	YuGene	TRUE
EN_Chronoage_Rodents_Skeletal_muscle_scaleddiff.pkl	EN	Chronological	Rodents	Skeletal Muscle	Scaled	TRUE
EN_Chronoage_Rodents_Skeletal_muscle_yugenediff.pkl	EN	Chronological	Rodents	Skeletal Muscle	YuGene	TRUE
EN_Mortality_Mouse_Multitissue_scaleddiff.pkl	EN	Mortality	Mouse	Multi-Tissue	Scaled	FALSE
EN_Mortality_Mouse_Multitissue_yugenediff.pkl	EN	Mortality	Mouse	Multi-Tissue	YuGene	FALSE
EN_Mortality_Multispecies_Multitissue_scaleddiff.pkl	EN	Mortality	Multispecies	Multi-Tissue	Scaled	FALSE
EN_Mortality_Multispecies_Multitissue_yugenediff.pkl	EN	Mortality	Multispecies	Multi-Tissue	YuGene	FALSE

filename	type	outcome	species	tissue	scaling	lifespan_scaled
EN_Mortality_Rodents_Liver_scaled-diff.pkl	EN	Mortality	Rodents	Liver	Scaled	FALSE
EN_Mortality_Rodents_Liver_yu-genediff.pkl	EN	Mortality	Rodents	Liver	YuGene	FALSE
EN_Mortality_Rodents_Multitissue_scaleddiff.pkl	EN	Mortality	Rodents	Multi-Tissue	Scaled	FALSE
EN_Mortality_Rodents_Multitissue_yugenediff.pkl	EN	Mortality	Rodents	Multi-Tissue	YuGene	FALSE
EN_NormalizedAge_Mouse_Multitissue_scaleddiff.pkl	EN	Normalized age	Mouse	Multi-Tissue	Scaled	FALSE
EN_NormalizedAge_Mouse_Multitissue_yugenediff.pkl	EN	Normalized age	Mouse	Multi-Tissue	YuGene	FALSE
EN_NormalizedAge_Multispecies_Multi-tissue_scaleddiff.pkl	EN	Normalized age	Multispecies	Multi-Tissue	Scaled	FALSE
EN_NormalizedAge_Multispecies_Multi-tissue_yugenediff.pkl	EN	Normalized age	Multispecies	Multi-Tissue	YuGene	FALSE
EN_NormalizedAge_Rodents_Liver_scaled-diff.pkl	EN	Normalized age	Rodents	Liver	Scaled	FALSE
EN_NormalizedAge_Rodents_Liver_yu-genediff.pkl	EN	Normalized age	Rodents	Liver	YuGene	FALSE
EN_NormalizedAge_Rodents_Multitissue_scaleddiff.pkl	EN	Normalized age	Rodents	Multi-Tissue	Scaled	FALSE
EN_NormalizedAge_Rodents_Multitissue_yugenediff.pkl	EN	Normalized age	Rodents	Multi-Tissue	YuGene	FALSE

Appendix C - ComputAgeBench benchmark studies (65 datasets, blood/saliva/buccal)

From ComputAge BenchBSB_table.csv. `HC content = HC` means the study contains its own healthy controls and can run the AA2 test; `No HC` means only the AA1 test is possible. Condition abbreviations: AS atherosclerosis, IHD ischaemic heart disease, CVA cerebrovascular accident, IBD inflammatory bowel disease, CD Crohn's, UC ulcerative colitis, HIV, TB tuberculosis, XOB extreme obesity, OBS obesity, T1D/T2D diabetes, ASD, RA rheumatoid arthritis, PA psoriatic arthritis, UA undifferentiated arthritis, OA osteoarthritis, OP osteoporosis, MS multiple sclerosis, AD Alzheimer's, PD Parkinson's, CJD Creutzfeldt–Jakob, COPD, WS Werner, aWS atypical Werner, MDPS mandibular dysplasia, HGPS Hutchinson–Gilford progeria, ncLMNA non-classical laminopathy, CGL congenital generalised lipodystrophy.

Class	Condition	Conditions	Dataset ID	PMID	Cited	First author, year	Tissue	N samples	HC content	Comment
CVD	AS	AS	GSE56046	25404168	y	Reynolds, 2014	Blood	1202	HC	
CVD	AS	AS	GSE56581	25404168	y	Reynolds, 2014	Blood	214	HC	
CVD	IHD	IHD	GSE62867	25856389	y	Nazarenko, 2014	Blood, vessels	24	No HC	
CVD	CVA	CVA	GSE69138	26643952	y	Soriano-Tárraga, 2016	Blood	185 w/ages	No HC	
CVD	AS	AS	GSE107143	28698603	y	Istas, 2017	Blood	16 (8 HC)	HC	
CVD	CVA	CVA	GSE203399	36180927	y	Cullell, 2022	Blood	121	No HC	
ISD	IBD	IBD	GSE32148	22467598	y	Harris, 2012	Blood	48 (20 HC)	HC	
ISD	HIV	HIV	GSE53840	25969563	y	Horvath, 2015a	Blood	120	No HC	
ISD	HIV	HIV	GSE53841	25969563	y	Horvath, 2015a	Blood	24	No HC	
ISD	HIV	HIV	GSE67705	27105112	y	Gross, 2016	Blood	284	HC	
ISD	HIV	HIV	GSE67751	25969563	y	Horvath, 2015a	Blood	92 (69 HC)	HC	
ISD	HIV	HIV	GSE77696	27672717	y	Zhang, 2016	Blood	378 (117 HC)	HC	
ISD	IBD	IBD (CD)	GSE81961	27279921	y	Li Yim, 2016	Blood	40 (25 HC)	HC	
ISD	IBD	IBD (CD, UC)	GSE87640	27886173	y	Ventham, 2016	Blood	240 (84 HC)	HC	
ISD	IBD	IBD (CD, UC)	GSE87648	27886173	y	Ventham, 2016	Blood	382 (190 HC)	HC	
ISD	HIV	HIV	GSE100264	29269866	v	Zhang, 2017	Blood	386	No HC	
ISD	HIV	HIV	GSE107080	29269866	v	Zhang, 2017	Blood	405	No HC	
ISD	HIV	HIV	GSE117859	30545403	v	Zhang, 2018	Blood	608	No HC	
ISD	HIV	HIV	GSE117860	30545403	v	Zhang, 2018	Blood	529	No HC	
ISD	HIV	HIV	GSE140800	32760119	v	Oriol-Tordera, 2020	Blood	70	No HC	
ISD	HIV	HIV	GSE143942	Unpublished		Unpublished	Blood	85	HC	
ISD	HIV_TB	HIV, TB	GSE145714	32125282	v	DiNardo, 2020	Blood	32	HC	
ISD	HIV	HIV	GSE185389	35325780	v	Oriol-Tordera, 2022	Blood	56	No HC	
ISD	HIV	HIV	GSE185390	35325780	v	Oriol-Tordera, 2022	Blood	30	No HC	
ISD	HIV	HIV	GSE217633	36640455	v	Esteban-Cantos, 2023	Blood	413 (44 HC)	HC	

Class	Condition	Conditions	Dataset ID	PMID	Cited	First author, year	Tissue	N samples	HC content	Comment
MBD	XOB	XOB, OBS, ASD	GSE49909	24034465	v	Day, 2013	Blood, brain	283	HC	
MBD	T1D	T1D	GSE56606	21980303	v	Rakyan, 2011	Blood	100 (68 HC)	HC	MZ twins
MBD	T2D	T2D	GSE62003	25585531	v	Lunnon, 2015	Blood	70	No HC	
MBD	XOB	XOB	GSE131461	31333715	v	Ramos-Molina, 2019	Blood, adipose	40	No HC	
MBD	XOB	XOB, OBS	GSE166611	Unpublished		Unpublished	Blood	32 (17 HC)	HC	
MBD	XOB	XOB	GSE193836	35369101	v	Noronha, 2022	Blood	21 (11 HC)	HC	
MSD	RA	RA	GSE42861	23334450	v	Liu, 2013	Blood	689	HC	
MSD	RA	RA	GSE71841	Unpublished		Unpublished	Blood	24 (12 HC)	HC	Unpublished
MSD	OP	OP	GSE99624	28926142	v	Fernandez-Rebollo, 2018	Blood	48	HC	
MSD	RA	RA	GSE131989	27723282	v	Rhead, 2017	Blood	371	HC	
MSD	RA	RA	GSE134429	Unpublished		Unpublished	Blood	64 (17 HC)	HC	Unpublished
MSD	RA	RA, PA, UA, OA	GSE137593	31945409	v	Clark, 2020	Blood	103	No HC	Early arthritis
MSD	RA	RA, PA, UA, OA	GSE137594	31945409	v	Clark, 2020	Blood	119	No HC	Early arthritis
MSD	RA	RA	GSE138653	32909363	v	Tao, 2021	Blood	80	No HC	Before medication
MSD	RA	RA, UA	GSE175364	34105306	v	de la Calle-Fabregat, 2021	Blood	101	HC	UA -> RA/OA
MSD	RA	RA	GSE176168	35576644	v	Julià, 2022	Blood	113	No HC	Longitudinal (0,12 w), anti-TNF
MSD	RA	RA	GSE228104	37744384	v	Chen, 2023	Blood	40	No HC	Leflunomide
NDD	MS	MS	GSE43976	23422812	v	Marabita, 2013	Blood	95 (20 HC)	No HC	
NDD	AD	AD	GSE59685	25129077	v	Lunnon, 2014	Brain, blood	531	HC	
NDD	PD	PD	GSE72774	26655927	v	Horvath, 2015b	Blood	508 (219 HC)	HC	
NDD	PD	PD	GSE72776	26655927	v	Horvath, 2015b	Blood	84 (38 HC)	HC	
NDD	MS	MS, OBS	GSE103929	30981648	v	Castro, 2019	Blood	49	No HC	BMI
NDD	MS	MS	GSE106648	29921915	v	Kular, 2019	Blood	279 (139 HC)	HC	
NDD	PD	PD	GSE111223	28851441	v	Chuang, 2017	Saliva	259 (131 HC)	HC	

Class	Condition	Conditions	Dataset ID	PMID	Cited	First author, year	Tissue	N samples	HC content	Comment
NDD	PD	PD	GSE111629	30958317	v	Chuang, 2017	Blood	572 (237 HC)	HC	
NDD	MS	MS	GSE112596	30698680	v	Ntranos, 2019	Blood	112	No HC	
NDD	PD	PD	GSE122244	Unpublished		Unpublished	Blood	70	HC	Non-Neurological controls
NDD	MS	MS	GSE130029	31053557	v	Ewing, 2019	Blood	31 (11 HC)	HC	
NDD	MS	MS	GSE130030	31053557	v	Ewing, 2019	Blood	28 (14 HC)	HC	
NDD	MS	MS	GSE130491	31300673	v	Carlström, 2019	Blood	82 (19x3x2)	No HC	
NDD	AD	AD	GSE144858	32745807	v	Roubroeks, 2020	Blood	300	HC	
NDD	PD	PD	GSE151355	32650713	v	Go, 2020	Blood, brain	40	No HC	
NDD	CJD	CJD	GSE156994	32918118	v	Dabin, 2020	Blood	219 (106 HC)	HC	
NDD	MS	MS	GSE219293	36685876	v	Bingen, 2023	Blood	47 (18 HC)	HC	
RSD	TB	TB	GSE72338	26374119	v	Esterhuyse, 2015	Blood	38	HC	HC = LTBI household concats
RSD	COPD	COPD	GSE118468	33658578	v	Chen, 2021	Blood	21 (6 HC)	HC	
RSD	TB	TB	GSE118469	32365959	v	Chen, 2020	Blood	21 (6 HC)	HC	
RSD	TB	HIV, TB	GSE145714	32125282	v	DiNardo, 2020	Blood	32	HC	
PGS	WS	WS, aWS, MDPS	GSE131752	31259468	v	Maierhofer, 2019	Blood	48 (24 HC)	HC	
PGS	HGPS	HGPS, ncLMNA	GSE182991	35045206	v	Bejaoui, 2022	Blood	27 (12 HC)	HC	
PGS	CGL	CGL	GSE214297	36715159	v	Qannan, 2022	Blood	16 (9 HC)	HC	

Appendix D - organAging organ protein panels

Organ assignment rule: a plasma protein belongs to an organ if its GTEx expression in that tissue is at least 4× its mean expression across all other tissues. Counts from organAging GTEx_4x_FC_genes.json. Three model families ship per organ: full (Olink Explore 3072), feature-reduced (Olink Explore 1536), and CSF. Each family has first-generation (chronological age, with intercept) and mortality (Cox elastic net, relative log hazard, no intercept) variants, each fitted over five folds, use fold 1 unless working in UK Biobank itself.

Organ panel	Proteins
Adipose	5
Adrenal	4
Artery	14
Bladder	0 (excluded from training)
Brain	118
Esophagus	5
Female (reproductive)	6
Heart	7
Immune	124
Intestine	31
Kidney	9
Liver	79
Lung	11
Male (reproductive)	55
Muscle	20
Pancreas	29
Pituitary	12
Salivary	10
Skin	17
Stomach	12
Thyroid	2
Organismal (any organ-enriched protein)	2,347
Multi-organ (enriched in >1 organ)	569
Conventional (all measured proteins)	2,916

Applying them: multiply Olink NPX values by the coefficients and sum; add the intercept for first-generation models only. For non-Olink platforms, log-transform to symmetry, z-score each protein, then multiply by the published Olink NPX standard deviations (Table S3 of the paper) before scoring. The authors ignored missing values rather than imputing, and warn that performance degrades with the number of missing high-coefficient proteins.

Appendix E - Preprocess and postprocess operation inventory

Every distinct operation a clock can request, with the clocks that use it and the shape of the computation. This is the complete op set a new engine must implement to cover the pyaging registry.

Preprocess (runs on the clock's feature slice, after feature selection)

Op	Signature	Dependencies stored with model	Clocks
`identity`	`x → x`	-	most
`fill_nan_with_reference`	`where(isnan(x), ref, x)`	`reference_values` (len = n_features)	~90 (`LinearReferenceClock` and subclasses)
`scale`	`(x − median) / std`	median vector, std vector	AltumAge, CpGPTGrimAge3
`scale_mean`	`(x − mean) / std`	mean vector, std vector	CpGPTPCGrimAge3
`scale_row`	`(x − rowmean) / rowstd`	-	ZhangEN, ZhangBLUP
`quantile_normalize_with_gold_standard`	rank map onto sorted gold means, then subset	20,000-length gold mean vector + scoring index list	DunedinPACE
`quantile_normalize_and_scale`	as above, then `(x − gold_mean)/gold_std`, then subset	gold means, gold stds, index list	Stubbs
`tpm_norm_log1p`	`log1p(1e6 · (1000·x/len) / rowsum)`, then subset	region lengths, index list	OcampoATAC1/2
`binarize`	per row, mask zeros, threshold at non-zero median	-	BiTAge
`median_fill_and_rank`	fill NaN with global median, then per-row average ranks	-	Pasta, PastaMouse, REG
`mean_over_present`	mean of entries ≠ −1 per row	-	epiTOC1, HypoClock, epiCMITHyper, epiCMITHypo
`quantile95_over_present`	0.95 quantile of entries ≠ −1 per row	-	stemTOC, stemTOCvitro
`nan_to_zero`	`nan_to_num(x, 0)`	-	epiTOC2, epiTOC3
`autosomal_zscore`	z-score all probes against the autosomal mean/SD of that sample	autosome index list	XChrom, YChrom
`expand_with_reference`	insert reference values for out-of-species features, then run the shared preprocess	full feature list, full reference vector, species index list	PastaMouse

Postprocess (runs on the scalar model output)

Listed in the transform table in §4.20. Fifteen distinct forms, of which `identity`, `anti_log_linear` and `cox_to_years` cover the large majority.

What FALCONAge implements of this

Seven per-clock preprocess ops, one dataset-level op and 21 postprocess transforms, in `models/ops.py`; 35 of the 69 bundled clocks declare at least one. Several are more general than their row above: `scale` takes centre and spread vectors stored with the model and so covers `scale` and `scale_mean`; `quantile_normalize` takes the gold vector and the scoring index, covering the DunedinPACE and Stubbs rows. `scale_row` needs statistics from the sample itself, over every probe the array carries, so it runs on the dataset before alignment.

Shipped	Covers
`beta_to_m`, `m_to_beta`	the β ↔ M logit pair, which no row above names because pyaging folds it into each clock
`scale`	`scale`, `scale_mean`
`standardise_within_sample` (dataset-level)	`scale_row`, over the whole array as the Zhang 2019 `pred.R` does
`rank_normalize`	`median_fill_and_rank`
`quantile_normalize`	`quantile_normalize_with_gold_standard`, `quantile_normalize_and_scale`
`binarize`	`binarize`
`simplex_projection`	the deconvolution row of the special passes
`add`, `add_constant`, `multiply`, `divide_by`, `days_to_months`, `days_to_weeks`, `clip`, `exp`, `expit`, `sigmoid`, `one_minus`, `log_linear`, `anti_log`, `anti_log_linear`, `anti_log_log`, `anti_logp2`, `scale_and_shift`, `cox_to_years`, `mortality_to_phenoage`, `petkovich_blood`, `stubbs_multitissue`	the postprocess set

Reference filling is not an op here. `fill_nan_with_reference` is the largest row in the table and FALCONAge does it in `align()` instead, before any op runs, because the imputation choice is a property of the *dataset* rather than of the clock and has to be recorded in the manifest either way. Zero is never a fill value: in β space zero means completely unmethylated, which is a real and extreme measurement, and a clock with a large positive coefficient on a probe the array does not carry will read "totally unmethylated" and shift the prediction by whole years.

Not ops, because the statistic is the whole model: `mean_over_present` and `quantile95_over_present` are `AggregationClock`, and the epiTOC2 / epiTOC3 row of the special passes below is `DivisionClock`, which averages over the sites present, as the published implementation does, where the table has `nan_to_zero`. The deconvolution row is `DeconvolutionClock`, by the authors' non-negative least squares rather than the pseudo-inverse and simplex projection the row describes. Unbuilt: `tpm_norm_log1p`, `autosomal_zscore`, `expand_with_reference`, and the other eight special forward passes. Each belongs to a clock family whose coefficients are licensed, not yet traced to a primary source, or absent from the registry, so building the op first would produce code with no real answer to test against, see the tier table in Architecture.

Special forward passes (not expressible as preprocess → linear → postprocess)

Clock family	Structure
GrimAge / GrimAge2	8 or 10 sub-linear-models on disjoint feature index sets → concat with Age and Female in a fixed order → Cox linear layer → `cox_to_years`
PCGrimAge	centre + rotate 78,464 → 1,933 PCs → concat Female, Age → 8 PC sub-models → Cox layer
CpGPTPCGrimAge3	scale → split age from proxies → rotate 30 → 29 PCs → concat → rescale PCs → linear → `cox_to_years`
DNAmFitAge	split by sex → 5 sub-models (GaitF/M, GripF/M, VO2Max) → affine standardisation with sex-specific constants → sex-specific KDM layer → recombine
Mammalian 2/3 and blood/skin variants	split CpG block from species one-hot (1,756 or 1,707 wide) → linear on CpGs → species-dependent inverse age transform using AnAge gestation / maturity / max lifespan
SystemsAge	centre + rotate 125,175 CpGs → project to system components → 11 system linear modules + a PC→age model with quadratic correction → optional composite PCA layer → affine calibration → /12
Deconvolution	pseudo-inverse of the signature matrix → simplex projection → select one cell-type column
epiTOC2 / epiTOC3	$2/k \cdot \Sigma\ (\beta - \beta_0)/(\delta(1-\beta_0))$ with per-CpG δ and β_0 vectors
XChrom / YChrom	autosomal z-score → select sex-chromosome probes → Σ `(z − mean) · coef`
PastaMouse	expand mouse features into the 8,113-gene human space with reference values → shared Pasta forward