Package 'par2circadian'

November 18, 2026

Title AR(2) Eigenvalue Persistence Analysis of Circadian Time Series

Version 0.1.1

Description Estimates temporal persistence in gene expression time series by fitting a second-order autoregressive model per gene and reporting the eigenvalue modulus of its characteristic equation. Conventional circadian methods answer whether a gene cycles; the eigenvalue modulus instead measures how strongly recent expression constrains future expression, which separates clock genes from their targets and from background in intact tissue. Provides single-series and matrix-wide fitting, root-type classification, half-life and intrinsic-period conversions, residual-bootstrap confidence intervals for short series, and hierarchy summaries. Numerically equivalent to the reference 'Python' implementation 'par2-circadian'. The method is described in Whiteside (2026) <doi:10.21203/rs.3.rs-9283100/v1>.

License Apache License (>= 2)

Encoding UTF-8

URL https://github.com/mickwh2764/par2circadian-R

BugReports https://github.com/mickwh2764/par2circadian-R/issues

Depends R (>= 3.6.0)

Imports stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Michael Whiteside [aut, cre, cph] (ORCID: <https://orcid.org/0009-0000-0643-5791>)

Maintainer Michael Whiteside <mickwh@msn.com>

Repository CRAN

Date/Publication 2026-11-18 10:00:08 UTC

Contents

bootstrap_ar2 . . . 2
classify_dynamics . . . 3
classify_gene_layer . . . 3
discover_hierarchy . . . 4
eigenperiod . . . 5
fit_ar2 . . . 6
fit_ar2_batch . . . 7
gearbox_gap . . . 8
gene_sets . . . 8
half_life . . . 9

Index **10**

`bootstrap_ar2` *Residual-bootstrap confidence intervals for an AR(2) fit*

Description

Resamples the fitted residuals with replacement, regenerates the series from the fitted recursion, refits, and returns the percentile interval over the refits. This is the standard interval for autoregressive coefficients and needs only the one fitted series – no biological replicates, which most circadian designs (one animal pool per timepoint) do not provide.

Usage

```
bootstrap_ar2(centred, phi1, phi2, n_bootstrap = 2000, seed = NULL, conf = 95)
```

Arguments

`centred`	The mean-centred series the coefficients were fitted to.
`phi1`, `phi2`	Fitted AR(2) coefficients.
`n_bootstrap`	Number of bootstrap draws.
`seed`	Optional integer seed. When supplied, `set.seed(seed)` is called before resampling, which resets the session's random-number stream. Leave `NULL` and call `set.seed()` yourself to keep control of the stream.
`conf`	Confidence level as a percentage.

Value

A list with `eigenvalue_ci`, `phi1_ci`, `phi2_ci` (each a length-2 numeric vector) and `n_timepoints`.

Reproducibility across implementations

Point estimates from this package agree with the Python implementation to within floating-point rounding, but bootstrap intervals do not: R and NumPy draw from different generators, so the same seed selects different resamples. Compare interval widths, not endpoints.

Examples

```
set.seed(1)
x <- as.numeric(arima.sim(list(ar = c(0.6, -0.3)), n = 48))
fit_ar2(x, n_bootstrap = 200, seed = 42)$eigenvalue_ci
```

---

classify_dynamics *Classify a gene's dynamics from its AR(2) result*

Description

The label depends on root type as well as magnitude: a high $|\lambda|$ with real roots is persistent but not oscillatory, and calling it an oscillator is the most common misreading of the metric.

Usage

```
classify_dynamics(eigenvalue, root_type)
```

Arguments

`eigenvalue`	Numeric $|\lambda|$.
`root_type`	`"Complex"` or `"Real"`.

Value

One of `"Unstable"`, `"Sustained oscillator"`, `"Damped oscillator"`, `"Overdamped decay"` or `"Rapid decay"`.

Examples

```
classify_dynamics(0.92, "Complex")
classify_dynamics(0.92, "Real")
```

---

classify_gene_layer *Assign a gene to a hierarchy layer*

Description

Assign a gene to a hierarchy layer

Usage

```
classify_gene_layer(gene_name)
```

Arguments

gene_name	Gene symbol.

Value

"Clock", "Target" or "Background".

Examples

```
classify_gene_layer("Per2")
classify_gene_layer("Actb")
```

---

discover_hierarchy *Recover the three-layer persistence hierarchy from a batch fit*

Description

Stratifies fitted genes into clock, target and background layers and reports their median $|\lambda|$. The clock-to-target difference is the gearbox gap: in intact tissue clock genes are the most persistent layer, and the gap collapses when the feedback loop is disrupted.

Usage

```
discover_hierarchy(results, clock_genes = NULL, target_genes = NULL)
```

Arguments

results	A par2_batch data frame from fit_ar2_batch, or any data frame with gene and eigenvalue columns.
clock_genes	Optional character vector replacing CORE_CLOCK_GENES, for other species or custom definitions.
target_genes	Optional character vector replacing KNOWN_TARGET_GENES.

Value

A list with the three layer medians, gearbox_gap, hierarchy_preserved, the layer sizes, the clock and target genes with their eigenvalues, and health_grade.

Interpreting the grade

The A-F grade is a convenience banding of the gap (A at 0.15 and above, then 0.10, 0.05, 0.02, F below). The thresholds are descriptive, taken from the ranges seen in intact mouse liver, and are not a validated diagnostic.

Examples

```
set.seed(1)
m <- t(replicate(6, as.numeric(arima.sim(list(ar = c(0.5, -0.2)), n = 24))))
rownames(m) <- c("Per2", "Arntl", "Nr1d1", "Myc", "Wee1", "Actb")
discover_hierarchy(fit_ar2_batch(m))$gearbox_gap
```

`eigenperiod` *Intrinsic eigenperiod in time units*

Description

The oscillation period implied by a complex-root AR(2) fit. This is the period the dynamics prefer, which need not equal the period of any imposed driver.

Usage

```
eigenperiod(phi1, phi2, sampling_interval = 2)
```

Arguments

`phi1`, `phi2`	AR(2) coefficients.
`sampling_interval`	Time between consecutive samples.

Value

The period in the units of `sampling_interval`, or `NA` for real roots, where there is no oscillation to have a period.

Examples

```
eigenperiod(0.6, -0.3)
eigenperiod(1.2, -0.2)
```

`fit_ar2` *Fit an AR(2) model to one expression time series*

Description

Fits $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \epsilon_t$ to a mean-centred series by ordinary least squares and returns the eigenvalue modulus $|\lambda|$ of the characteristic equation $r^2 - \phi_1 r - \phi_2 = 0$, together with the derived persistence measures.

Usage

```
fit_ar2(x, n_bootstrap = 0, seed = NULL, conf = 95)

## S3 method for class 'par2_fit'
print(x, ...)
```

Arguments

`x`	Numeric vector of expression values, at least 6 timepoints.
`n_bootstrap`	Number of residual-bootstrap draws. When greater than zero, percentile confidence intervals are added; 2000 is ample.
`seed`	Optional integer seed for reproducible bootstrap intervals. When supplied, `set.seed(seed)` is called before resampling, which resets the session's random-number stream. Leave `NULL` and call `set.seed()` yourself to keep control of the stream.
`conf`	Confidence level for the bootstrap intervals, as a percentage.
`...`	Ignored.

Details

$|\lambda|$ measures how strongly recent history constrains future values. It is not a test of rhythmicity: a non-oscillatory red-noise series can score high, correctly, and a clean fixed-period cosine need not. Use it alongside a rhythm-detection method, not instead of one.

Roots are complex when $\phi_1^2 + 4\phi_2 < 0$, which indicates oscillatory dynamics; real roots indicate overdamped, non-oscillatory decay.

Value

An object of class `par2_fit`: a list with elements `eigenvalue` ($|\lambda|$), `phi1`, `phi2`, `r2`, `root_type` (`"Complex"` or `"Real"`), `half_life` in samples, and `eigenperiod` in samples (`NA` for real roots). With `n_bootstrap > 0` it also carries `eigenvalue_ci`, `phi1_ci`, `phi2_ci` and `n_timepoints`.

Precision

A point estimate from a short series is imprecise: at 24 evenly sampled timepoints the 95 is biased upward below roughly 24 points. Report the interval alongside any per-gene value and do not interpret differences smaller than it.

See Also

fit_ar2_batch for a whole expression matrix, classify_dynamics, half_life.

Examples

```
set.seed(1)
x <- as.numeric(arima.sim(list(ar = c(0.6, -0.3)), n = 48))
fit <- fit_ar2(x)
fit$eigenvalue
fit$root_type
```

---

fit_ar2_batch *Fit AR(2) to every gene in an expression matrix*

Description

Fit AR(2) to every gene in an expression matrix

Usage

```
fit_ar2_batch(expression, gene_names = NULL)
```

Arguments

expression	A numeric matrix of genes (rows) by timepoints (columns), or anything coercible by as.matrix. Missing values are dropped per gene; genes left with fewer than 6 timepoints are skipped rather than returned with a meaningless fit.
gene_names	Optional character vector of gene identifiers. Defaults to the matrix rownames, or Gene_1 ... Gene_n when there are none.

Value

A data frame of class par2_batch, one row per fitted gene, ordered by decreasing $|\lambda|$, with columns gene, eigenvalue, phi1, phi2, r2, root_type, half_life and eigenperiod.

Examples

```
set.seed(1)
m <- t(replicate(5, as.numeric(arima.sim(list(ar = c(0.5, -0.2)), n = 24))))
rownames(m) <- paste0("Gene", 1:5)
head(fit_ar2_batch(m))
```

gearbox_gap *Gearbox gap between two sets of eigenvalues*

Description

Gearbox gap between two sets of eigenvalues

Usage

```
gearbox_gap(clock_eigenvalues, target_eigenvalues)
```

Arguments

`clock_eigenvalues`, `target_eigenvalues`
: Numeric vectors of $|\lambda|$.

Value

`median(clock) - median(target)`, or `0` if either set is empty.

Examples

```
gearbox_gap(c(0.72, 0.70), c(0.61, 0.59))
```

gene_sets *Core clock and known circadian target gene symbols*

Description

Mouse and human symbols for the transcriptional-translational feedback loop (`CORE_CLOCK_GENES`) and for well-established clock-controlled targets (`KNOWN_TARGET_GENES`), used to stratify a batch fit into layers. Matching is by exact symbol, so supply your own set for other species or nomenclatures.

Usage

```
CORE_CLOCK_GENES

KNOWN_TARGET_GENES
```

Format

Character vectors.

Value

Two character vectors of gene symbols: `CORE_CLOCK_GENES` and `KNOWN_TARGET_GENES`, each listing mouse and human forms. They are the default sets that `classify_gene_layer` matches against.

`half_life` *Persistence half-life in time units*

Description

The time for a perturbation to decay to half its amplitude, given the eigenvalue modulus and the sampling interval.

Usage

```
half_life(eigenvalue, sampling_interval = 2)
```

Arguments

`eigenvalue`	Numeric $\lvert\lambda\rvert$.
`sampling_interval`	Time between consecutive samples, in whatever units the result should be reported in. Defaults to 2 hours, the interval of the reference liver series.

Value

Half-life in the units of `sampling_interval`: `0` when $\lvert\lambda\rvert \leq 0$, and `Inf` at or above 1, where the process is non-stationary and nothing decays.

Examples

```
half_life(0.8)
half_life(0.8, sampling_interval = 1)
```

Index

bootstrap_ar2, 2

classify_dynamics, 3, *7*
classify_gene_layer, 3, *8*
CORE_CLOCK_GENES *(*gene_sets*)*, 8

discover_hierarchy, 4

eigenperiod, 5

fit_ar2, 6
fit_ar2_batch, *4*, *7*, 7

gearbox_gap, 8
gene_sets, 8

half_life, *7*, 9

KNOWN_TARGET_GENES *(*gene_sets*)*, 8

print.par2_fit *(*fit_ar2*)*, 6