

Package: hypnoR (via r-universe)

October 9, 2026

Title Hypnogram Handling, Plotting, and Sleep Architecture Metrics

Version 0.1.1

Description Provides a staging-agnostic layer for hypnogram ingestion, sleep architecture metric computation, cycle segmentation, and transition analysis. Accepts full AASM-staged hypnograms (W / N1 / N2 / N3 / REM) from mrpheus and coarser actigraphy-derived hypnograms (W / Sleep / Quiet sleep) from zeitR; all metric functions degrade gracefully depending on staging resolution. Includes publication-ready plotting with a shared Circadia Lab colour palette. Designed as the hypnogram layer of the Circadia Lab ecosystem, feeding into syncR::sync().

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports cli (>= 3.6.0), stats, tibble (>= 3.0.0), utils

Suggests covr, dplyr (>= 1.1.0), forcats (>= 1.0.0), ggplot2 (>= 3.4.0), knitr, lubridate (>= 1.9.0), mrpheus, pkgdown, rlang (>= 1.1.0), rmarkdown, testthat (>= 3.0.0), tidyr (>= 1.3.0), zeitR

Additional_repositories <https://circadia-bio.r-universe.dev>

Config/testthat/edition 3

URL <https://hypnor.circadia-lab.uk>,
<https://github.com/circadia-bio/hypnoR>

BugReports <https://github.com/circadia-bio/hypnoR/issues>

Config/roxygen2/version 8.0.0

Repository <https://circadia-bio.r-universe.dev>

Date/Publication 2026-07-11 11:47:21 UTC

RemoteUrl <https://github.com/circadia-bio/hypnoR>

RemoteRef v0.1.1
RemoteSha 3f2d3be85ae3f38a45e77bed9c7a67eea49f1897

Contents

compute_cycles	2
compute_sleep_architecture	4
compute_transitions	5
new_hypnogram	6
plot_architecture	8
plot_hypnogram	9
plot_transition_matrix	10
read_hypnogram	11
smooth_hypnogram	12
window_hypnogram	14
Index	16

compute_cycles	<i>Detect NREM/REM sleep cycles</i>
----------------	-------------------------------------

Description

Segments a full AASM hypnogram into NREM/REM cycles. A cycle is defined as an NREM stretch followed by a qualifying REM period; only *complete* cycles are returned – a trailing NREM stretch at the end of the recording that is not followed by a qualifying REM period does not produce a final row.

Usage

```
compute_cycles(  
  hypnogram,  
  method = c("feinberg_floyd", "aasm"),  
  min_rem_epochs = 5L,  
  rem_gap_min = 15  
)
```

Arguments

- | | |
|-----------|---|
| hypnogram | A hypnor_hypnogram object as returned by new_hypnogram() or read_hypnogram() , or any data frame with at minimum epoch and stage columns – it will be passed through new_hypnogram() automatically if not already a hypnor_hypnogram. Epoch duration is read from the object’s epoch_sec attribute. |
| method | "feinberg_floyd" (default) or "aasm": |

	"feinberg_floyd" Feinberg & Floyd (1979)'s original, simplest rule: a REM period is any maximal contiguous run of REM epochs of at least min_rem_epochs. No tolerance for interruption – a single non-REM epoch ends the REM period. "aasm" A simplified AASM-compatible variant that tolerates brief non-REM interruptions within a REM period: consecutive REM runs separated by a gap of at most rem_gap_min minutes are merged into a single REM period before the min_rem_epochs threshold is applied to their combined REM epoch count. This is a practical approximation, not a citable AASM standard – the AASM manual itself does not define sleep-cycle segmentation rules.
min_rem_epochs	Minimum number of REM epochs (after any merging under method = "aasm") for a run to qualify as a REM period. Default 5 (2.5 min at 30-s epochs).
rem_gap_min	Only used when method = "aasm". Maximum gap, in minutes, of non-REM epochs between two REM runs for them to be merged into a single REM period. Default 15.

Details

Only applicable to full AASM hypnograms (coarse actigraphy-derived hypnograms have no REM stage and so cannot be cycle-segmented); this function errors on a coarse hypnogram rather than silently returning an empty result.

Value

A tibble with one row per detected (complete) cycle:

cycle Integer cycle index.

start_epoch,end_epoch First and last epoch of the cycle, using the values in hypnogram\$epoch itself – not row position. These differ whenever hypnogram doesn't start at epoch 1 (e.g. after subsetting a larger recording to a sleep-period window).

nrem_min,rem_min Duration of the NREM and REM portions of the cycle (minutes). nrem_min is the whole non-REM portion of the cycle, including any brief interruption epochs absorbed into a merged REM period under method = "aasm"; rem_min counts only actual REM epochs. nrem_min + rem_min == cycle_min.

cycle_min Total cycle duration (minutes).

Zero rows if no qualifying REM period is found (e.g. no sleep, or no REM at all).

References

Feinberg, I., & Floyd, T. C. (1979). Systematic trends across the night in human sleep cycles. *Psychophysiology*, 16(3), 283-291.

Examples

```
## Not run:
hyp <- read_hypnogram("night_001.csv")
compute_cycles(hyp)
```

```
compute_cycles(hyp, method = "aasm", rem_gap_min = 10)

## End(Not run)
```

```
compute_sleep_architecture
```

```
Compute sleep architecture metrics
```

Description

Derives standard sleep architecture summary statistics from a staged hypnogram. All metrics are resolution-agnostic: the function computes every metric that is possible given the available staging levels and returns NA for metrics that require stages not present at the detected resolution (e.g. REM/SWS latency and AASM stage percentages require a full AASM hypnogram).

Usage

```
compute_sleep_architecture(hypnogram, lights_off = NULL, lights_on = NULL)
```

Arguments

hypnogram	A hypnor_hypnogram object as returned by new_hypnogram() or read_hypnogram() , or any data frame with at minimum epoch and stage columns – it will be passed through new_hypnogram() automatically if not already a hypnor_hypnogram. Epoch duration and staging resolution are read from the object's epoch_sec and resolution attributes.
lights_off, lights_on	Optional POSIXct timestamps for lights-off and lights-on. When both are supplied, hypnogram is first restricted to this window via window_hypnogram() – every metric (TST, SOL, WASO, stage percentages, everything) is computed relative to the window, not just TIB/SE. Otherwise TIB defaults to the full span of hypnogram as passed in (first to last epoch).

Details

Sleep onset is defined as the first epoch with any non-"W" stage – this applies uniformly to full AASM hypnograms (where N1 counts as sleep onset) and coarse hypnograms (where "Sleep" or "Quiet sleep" both count).

Value

A one-row tibble with columns:

tst_min Total sleep time (minutes).

tib_min Time in bed (minutes).

se_pct Sleep efficiency (percent) = TST / TIB * 100.

sol_min Sleep onset latency (minutes). NA if no sleep epoch is present.

waso_min Wake after sleep onset (minutes): wake epochs between the first and last sleep epoch. NA if no sleep epoch is present.

rem_lat_min REM latency from sleep onset (minutes). NA for coarse hypnograms or if no REM epoch is present.

sws_lat_min SWS (N3) latency from sleep onset (minutes). NA for coarse hypnograms or if no N3 epoch is present.

pct_n1,pct_n2,pct_n3,pct_rem Stage percentages of TST. NA for coarse hypnograms.

pct_sleep,pct_quiet_sleep Stage percentages of TST for coarse hypnograms. NA for full AASM hypnograms.

staging_resolution "aasm" or "coarse".

Examples

```
## Not run:
hyp <- read_hypnogram("night_001.csv")
compute_sleep_architecture(hyp)

## End(Not run)
```

compute_transitions	<i>Compute stage-transition statistics</i>
---------------------	--

Description

Builds a stage-to-stage transition probability matrix and derives fragmentation indices from a staged hypnogram. Works with both full AASM and coarse actigraphy-derived staging.

Usage

```
compute_transitions(hypnogram, normalise = TRUE, include_wake = TRUE)
```

Arguments

hypnogram	A hypnor_hypnogram object as returned by new_hypnogram() or read_hypnogram() , or any data frame with at minimum epoch and stage columns – it will be passed through new_hypnogram() automatically if not already a hypnor_hypnogram.
normalise	If TRUE (default), each row of the transition count matrix is divided by its row sum to give transition probabilities. Rows for a from-stage that is never visited are returned as NA rather than NaN. If FALSE, raw transition counts are returned.
include_wake	If TRUE (default), "W" is included as a state in the transition matrix like any other stage (including W->W self-transitions). If FALSE, the matrix is restricted to sleep-stage transitions only: any transition into or out of "W" is excluded before the matrix is built.

Details

Fragmentation metrics (`n_transitions`, `fragmentation_index`, `wake_transitions`) are always computed from the full epoch sequence, wake included, regardless of `include_wake` – that argument only controls the shape of the returned matrix.

Value

A list with two elements:

matrix A tibble with one row per *from* stage: a *from* column plus one numeric column per *to* stage (transition probabilities or counts).

fragmentation A one-row tibble with:

n_transitions Number of epoch-to-epoch stage changes (self-transitions do not count).

fragmentation_index Proportion of epochs that are followed by a different stage.

wake_transitions Number of transitions into Wake (proxy for arousal burden).

Examples

```
## Not run:
hyp <- read_hypnogram("night_001.csv")
trans <- compute_transitions(hyp)
trans$matrix
trans$fragmentation

# Sleep-stage transitions only, excluding Wake
compute_transitions(hyp, include_wake = FALSE)

## End(Not run)
```

new_hypnogram	<i>Construct a hypnoR hypnogram object</i>
---------------	--

Description

`new_hypnogram()` is the low-level constructor for the `hypnor_hypnogram` class used throughout `hypnoR`. It normalises hypnogram-shaped input – whether a bare tibble (as produced by `zeitR::export_hypnogram()` or `read_hypnogram()`) or a specially classed object (currently: `mrpheus::export_hypnogram()`'s `mrpheus_hypnogram`) – into the single internal representation that every other `hypnoR` function expects.

Usage

```
new_hypnogram(x, ...)

## Default S3 method:
new_hypnogram(
  x,
```

```

    epoch_sec = NULL,
    resolution = NULL,
    subject_id = NULL,
    source = NULL,
    ...
)

## S3 method for class 'mrpheus_hypnogram'
new_hypnogram(
  x,
  epoch_sec = NULL,
  resolution = NULL,
  subject_id = NULL,
  source = NULL,
  ...
)
```

Arguments

<code>x</code>	A data frame with at minimum epoch and stage columns, or an object with a dedicated <code>new_hypnogram()</code> method (currently: <code>mrpheus_hypnogram</code>). Recognised optional columns: <code>time</code> (POSIXct), <code>subject_id</code> , <code>source</code> .
<code>...</code>	Passed to methods; currently unused.
<code>epoch_sec</code>	Epoch duration in seconds. When <code>x</code> carries its own epoch duration (e.g. an <code>mrpheus_hypnogram</code> 's <code>epoch_s</code> attribute) that value is used unless <code>epoch_sec</code> is explicitly supplied here. Falls back to 30L when no value can be determined from either source.
<code>resolution</code>	"aasm", "coarse", or NULL (default). When NULL, resolution is auto-detected from the stage labels present in <code>x</code> (or, for <code>mrpheus_hypnogram</code> input, taken from its <code>resolution</code> attribute).
<code>subject_id</code>	Character or NULL. Overrides any subject/participant identifier carried by <code>x</code> .
<code>source</code>	Character or NULL. Overrides any source/scorer label carried by <code>x</code> .

Value

A tibble of class `hypnor_hypnogram` with columns `epoch` (integer), `time` (POSIXct, NA if unknown), `stage` (ordered factor), `subject_id` (character, NA if unknown), and `source` (character, NA if unknown). The detected `epoch_sec` and `resolution` are attached as attributes.

Examples

```

## Not run:
# From a bare tibble (e.g. zeitR::export_hypnogram() output)
new_hypnogram(zeitr_hyp)

# From mrpheus::export_hypnogram() output
new_hypnogram(mrpheus_hyp)

## End(Not run)
```

plot_architecture	<i>Plot sleep architecture as a bar chart</i>
-------------------	---

Description

Renders stage durations or percentages as a horizontal bar chart using ggplot2 and a Circadia Lab colour palette.

Usage

```
plot_architecture(  
  architecture,  
  metric = c("duration", "percentage"),  
  colours = NULL,  
  title = NULL  
)
```

Arguments

architecture	A tibble returned by compute_sleep_architecture() – either a single row, or multiple rows for comparing several nights, in which case a night or id column (if present) is used to facet the plot into one panel per night.
metric	"duration" (minutes, default) or "percentage" of TST.
colours	Named character vector of stage colours. See plot_hypnogram() for defaults.
title	Optional plot title.

Value

A ggplot object.

Examples

```
## Not run:  
hyp <- read_hypnogram("night_001.csv")  
arch <- compute_sleep_architecture(hyp)  
plot_architecture(arch)  
plot_architecture(arch, metric = "percentage")  
  
## End(Not run)
```

plot_hypnogram	<i>Plot a hypnogram</i>
----------------	-------------------------

Description

Renders a hypnogram using ggplot2 and the Circadia Lab colour palette. Accepts both full AASM and coarse staging; the lane order (deepest sleep at the bottom, Wake at the top) and colour mapping are set automatically from the staging levels present in hypnogram.

Usage

```
plot_hypnogram(
  hypnogram,
  style = c("step", "capsule"),
  x_axis = c("auto", "time", "hours"),
  date_breaks = "2 hours",
  cycles = NULL,
  colours = NULL,
  title = NULL,
  corner_min = 9
)
```

Arguments

hypnogram	A hypnor_hypnogram object as returned by new_hypnogram() or read_hypnogram() , or any data frame with at minimum epoch and stage columns – it will be passed through new_hypnogram() automatically if not already a hypnor_hypnogram. Epoch duration is read from the object's epoch_sec attribute.
style	"step" (default) or "capsule": "step" Classic clinical step-plot: one line tracing the stage at every epoch. "capsule" Rounded-pill bars per contiguous stage run, one lane per stage.
x_axis	"auto" (default), "time", or "hours": "auto" Uses actual clock time (from the time column) if hypnogram carries any non-NA timestamps, otherwise falls back to elapsed hours since the first epoch. "time" Forces clock time; errors if time is entirely NA (no start_time was supplied to new_hypnogram() or <code>mrpheus::export_hypnogram()</code>). "hours" Forces elapsed hours since the first epoch, regardless of whether real timestamps are available.
date_breaks	Only used when plotting clock time. Passed to <code>ggplot2::scale_x_datetime()</code> 's date_breaks. Default "2 hours".
cycles	Optional: the tibble returned by compute_cycles() . When supplied, a dashed vertical line is drawn at the start of each cycle.

colours	Named character vector mapping stage labels to hex colours. Defaults to a built-in palette drawn from the Circadia Lab colours. Pass your own named vector to override.
title	Optional plot title.
corner_min	Only used by style = "capsule". Maximum pill corner radius, in minutes. Runs shorter than 2 * corner_min get a proportionally smaller radius (so very brief runs render as fully rounded capsule/stadium shapes rather than having oversized corners), longer runs are capped at this radius. Default 9.

Value

A ggplot object.

Examples

```
## Not run:
hyp <- read_hypnogram("night_001.csv")
plot_hypnogram(hyp)
plot_hypnogram(hyp, cycles = compute_cycles(hyp))
plot_hypnogram(hyp, x_axis = "hours")
plot_hypnogram(hyp, style = "capsule")

## End(Not run)
```

plot_transition_matrix

Plot a stage-transition heatmap

Description

Renders the transition probability (or count) matrix returned by `compute_transitions()` as a heatmap using ggplot2.

Usage

```
plot_transition_matrix(
  transitions,
  label_values = TRUE,
  digits = 2L,
  title = NULL
)
```

Arguments

transitions	The matrix element of the list returned by <code>compute_transitions()</code> : a tibble with a from column plus one numeric column per <i>to</i> stage.
label_values	If TRUE (default), cell values are printed inside each tile. NA cells (unvisited from-stages) are left blank.

digits Number of decimal places for cell labels. Default 2.
 title Optional plot title.

Value

A ggplot object.

Examples

```
## Not run:
hyp  <- read_hypnogram("night_001.csv")
trans <- compute_transitions(hyp)
plot_transition_matrix(trans$matrix)

## End(Not run)
```

read_hypnogram	<i>Read a hypnogram from file</i>
----------------	-----------------------------------

Description

Imports a staged hypnogram from common formats and returns a tidy tibble with one row per epoch.
 The resulting object is the standard input for all other hypnoR functions.

Usage

```
read_hypnogram(
  path,
  format = "auto",
  epoch_sec = 30L,
  tz = "UTC",
  quiet = FALSE
)
```

Arguments

path Path to the hypnogram file.
 format One of "csv", "edf_annotations", "yasa", "compumedics", "nox", or "auto" (default). When "auto", the format is inferred from the file extension and header.
 epoch_sec Epoch duration in seconds. Default 30.
 tz Time zone string passed to [lubridate::as_datetime\(\)](#). Default "UTC".
 quiet Suppress informational messages. Default FALSE.

Details

hypnoR distinguishes two staging resolutions:

- **Full AASM** (5-state): supplied by `mrpheus`.
- **Coarse** (3-state): supplied by `zeitR`.

All downstream metric functions are resolution-agnostic: they compute what is possible given the available stages and document which metrics require full AASM staging.

Value

A tibble with columns:

epoch Integer epoch index (1-based).

time POSIXct timestamp for the start of each epoch.

stage Ordered factor. Level set depends on staging resolution: full AASM (W, N1, N2, N3, REM) or coarse actigraphy-derived (W, Sleep, Quiet sleep).

source Character — the originating device / scorer label, if available in the file header.

Examples

```
## Not run:
hyp <- read_hypnogram("night_001.csv")
hyp <- read_hypnogram("night_001.edf", format = "edf_annotations")

## End(Not run)
```

smooth_hypnogram

Smooth a hypnogram by reassigning short, likely-spurious runs

Description

Raw automatic sleep staging is typically a per-epoch classification with no temporal continuity constraint (e.g. `mrpheus::stage_epochs()` is a pure per-epoch argmax over posterior probabilities), which can produce brief, isolated stage flips inconsistent with sleep physiology – a single REM epoch nested inside an N2 run, for instance. `smooth_hypnogram()` offers two independent, rule-based cleanup passes that operate purely on stage labels (no posterior probabilities required, and none are carried through by [new_hypnogram\(\)](#) in any case).

Usage

```
smooth_hypnogram(hypnogram, method = "aasm_isolated", min_run_epochs = 2L)
```

Arguments

hypnogram	A hypnor_hypnogram object as returned by <code>new_hypnogram()</code> or <code>read_hypnogram()</code> , or any data frame with at minimum epoch and stage columns – it will be passed through <code>new_hypnogram()</code> automatically if not already a hypnor_hypnogram.
method	One or more of "aasm_isolated" (default) and "min_run". When both are requested, "aasm_isolated" is always applied first, then "min_run" – regardless of the order the two are given in method: "aasm_isolated" A single epoch whose stage differs from both of its immediate neighbours is reassigned to the neighbours' stage, but only when both neighbours share the <i>same</i> stage. This mirrors the conservative manual-scoring convention of resolving an ambiguous single epoch bounded identically on both sides. It never touches a run of 2+ epochs, and never touches an isolated epoch whose two neighbours disagree with each other. "min_run" Any run shorter than min_run_epochs is reassigned to whichever flanking run is longer (ties favour the preceding run). Broader than "aasm_isolated": it does not require the two flanking runs to agree with each other, and (with min_run_epochs > 2) can merge runs longer than a single epoch. Applied as a single left-to-right pass over the run-length structure computed at the start of this step – it is not iterated to a fixed point, so a pathological hypnogram may still contain runs shorter than min_run_epochs afterwards. Call <code>smooth_hypnogram()</code> again on the result for a second pass if needed. Note that if "aasm_isolated" ran first, it may already have merged some runs together, which changes the flanking-run lengths "min_run" sees – the two methods are genuinely sequential, not independent.
min_run_epochs	Only used by "min_run". Minimum run length (in epochs) to leave untouched. Default 2L (merges only single-epoch runs).

Value

The input hypnor_hypnogram, with two changes: stage now holds the smoothed labels, and a new stage_raw column preserves the original, unsmoothed labels for comparison/audit. All other columns and attributes (epoch_sec, resolution) are unchanged.

Examples

```
## Not run:
hyp      <- new_hypnogram(mrpheus_hyp)
hyp_smooth <- smooth_hypnogram(hyp)
mean(hyp_smooth$stage != hyp_smooth$stage_raw) # proportion of epochs changed

# Broader cleanup: also merge non-isolated short runs
smooth_hypnogram(hyp, method = c("aasm_isolated", "min_run"), min_run_epochs = 3)

## End(Not run)
```

window_hypnogram	<i>Restrict a hypnogram to a time or epoch window</i>
------------------	---

Description

Filters a hypnogram down to a sub-range, either by clock time (`lights_off/lights_on`) or by epoch number (`from_epoch/to_epoch`), re-attaching `epoch_sec` and resolution correctly on the result. This is the single place windowing logic lives in `hypnoR` – `compute_cycles()` and `compute_transitions()` have no `lights_off/lights_on` arguments of their own; window first, then pass the windowed hypnogram in.

Usage

```
window_hypnogram(
  hypnogram,
  lights_off = NULL,
  lights_on = NULL,
  from_epoch = NULL,
  to_epoch = NULL
)
```

Arguments

<code>hypnogram</code>	A <code>hypnor_hypnogram</code> object as returned by <code>new_hypnogram()</code> or <code>read_hypnogram()</code> , or any data frame with at minimum epoch and stage columns – it will be passed through <code>new_hypnogram()</code> automatically if not already a <code>hypnor_hypnogram</code> .
<code>lights_off, lights_on</code>	POSIXct or NULL. Both must be supplied together. Requires hypnogram to carry real timestamps (i.e. <code>start_time</code> was supplied to <code>new_hypnogram()</code> or <code>mrpheus::export_hypnogram()</code>) – errors if time is entirely NA. Mutually exclusive with <code>from_epoch/to_epoch</code> .
<code>from_epoch, to_epoch</code>	Integer or NULL. Either may be supplied alone (an open-ended window); defaults to the first/last epoch in hypnogram respectively. Mutually exclusive with <code>lights_off/lights_on</code> .

Details

`compute_sleep_architecture()`'s own `lights_off/lights_on` arguments are sugar for calling this first: passing them restricts TST, SOL, WASO, and every other metric to the window, not just the TIB/SE denominator.

Value

A `hypnor_hypnogram`, filtered to the window, with `epoch_sec` and resolution carried over unchanged from the input – resolution is *not* re-detected from the (possibly much smaller) windowed subset, since a short window could plausibly contain no REM/N3 epochs and get misdetected as "coarse" otherwise.

Examples

```
## Not run:
hyp <- new_hypnogram(mrpheus_hyp)

# By clock time
hyp_night <- window_hypnogram(
  hyp,
  lights_off = as.POSIXct("2024-01-01 23:00:00", tz = "UTC"),
  lights_on  = as.POSIXct("2024-01-02 07:00:00", tz = "UTC")
)

# By epoch range (e.g. isolating a sleep period out of a longer recording)
hyp_sleep <- window_hypnogram(hyp, from_epoch = 1001L, to_epoch = 1950L)

compute_cycles(hyp_sleep)
compute_transitions(hyp_sleep)

## End(Not run)
```

Index

`compute_cycles`, 2
`compute_cycles()`, 9, 14
`compute_sleep_architecture`, 4
`compute_sleep_architecture()`, 8, 14
`compute_transitions`, 5
`compute_transitions()`, 10, 14

`ggplot2::scale_x_datetime()`, 9

`lubridate::as_datetime()`, 11

`new_hypnogram`, 6
`new_hypnogram()`, 2, 4, 5, 9, 12–14

`plot_architecture`, 8
`plot_hypnogram`, 9
`plot_hypnogram()`, 8
`plot_transition_matrix`, 10

`read_hypnogram`, 11
`read_hypnogram()`, 2, 4–6, 9, 13, 14

`smooth_hypnogram`, 12

`window_hypnogram`, 14
`window_hypnogram()`, 4