# Package: nocturn (via r-universe)

June 23, 2026

**Title** Sleep Data Filtering and Visualisation

**Version** 1.1.3

**Description** An online app and command-line utility to import, filter and visualise sleep data. Can be used with sleep data collected from any type of device (e.g. radar, sleep diary,...) as long as the data contains sleep onset and wake-up times for each sleep session.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**Suggests** bslib, devtools, DT, markdown, readr, shinyjs, shinyWidgets, svglite, testthat

**Config/testthat/edition** 3

**Imports** circular, cli, dplyr, edfReader, ggnewscale, ggplot2, lubridate, readxl, rlang, scales, shiny, shinyalert, stringr, tibble, tidyr, rsvg, svgedit, qpdf

**Depends** R (>= 4.3)

**LazyData** true

**URL** https://github.com/chronopsychiatry/AMBIENT-BD-nocturn

**BugReports** https://github.com/chronopsychiatry/AMBIENT-BD-nocturn/issues

**Config/roxygen2/version** 8.0.0

**Repository** https://chronopsychiatry.r-universe.dev

**Date/Publication** 2026-06-17 15:29:15 UTC

**RemoteUrl** https://github.com/chronopsychiatry/ambient-bd-nocturn

**RemoteRef** HEAD

**RemoteSha** 8a19c5314918adaeb8ab3589e922124c73a03900

# Contents

chronotype . . . 3
clean_epochs . . . 4
clean_sessions . . . 4
composite_phase_deviation . . . 5
example_epochs . . . 6
example_epochs_v1 . . . 7
example_sessions . . . 8
example_sessions_v1 . . . 9
filter_by_age_range . . . 10
filter_by_night_range . . . 11
filter_by_sex . . . 12
filter_epochs_from_sessions . . . 13
get_epochs_summary . . . 14
get_non_complying_sessions . . . 15
get_removed_sessions . . . 15
get_sessions_summary . . . 16
group_epochs_by_night . . . 17
group_sessions_by_night . . . 18
interdaily_stability . . . 19
load_batch . . . 19
load_epochs . . . 20
load_sessions . . . 21
max_time . . . 22
mean_time . . . 23
min_time . . . 24
nocturn . . . 24
parse_date . . . 25
parse_time . . . 25
plot_bedtimes_waketimes . . . 26
plot_hypnogram . . . 27
plot_sleep_bubbles . . . 28
plot_sleep_clock . . . 29
plot_sleep_spiral . . . 30
plot_timeseries . . . 31
plot_timeseries_sessions . . . 32
read_edf_epochs . . . 33
read_edf_sessions . . . 33
remove_sessions_no_sleep . . . 34
sd_time . . . 35
select_devices . . . 36
select_subjects . . . 37
set_min_sleep_period . . . 38
set_min_time_in_bed . . . 39
set_session_sleep_onset_range . . . 40
set_session_start_time_range . . . 41
shift_times_by_12h . . . 42

sleep_regularity_index . . . . . 43
sleep_report . . . . . 44
sleeptimes_boxplot . . . . . 45
sleeptimes_density . . . . . 45
sleeptimes_histogram . . . . . 46
social_jet_lag . . . . . 47
time_diff . . . . . 48
time_to_hours . . . . . 48
update_date . . . . . 49

**Index** **51**

---

`chronotype` *Calculate the Chronotype*

---

## Description

This function calculates the Chronotype metric based on the mid-sleep time If sleep duration on free days is greater than on workdays, it applies a correction as described in Roenneberg et al. (2019).

## Usage

```
chronotype(sessions)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions data frame |

## Details

This function uses columns:

- `time_at_midsleep`
- `sleep_period`
- `is_workday`

## Value

The Chronotype value in hours

## See Also

Other sleep metrics: `composite_phase_deviation()`, `interdaily_stability()`, `sleep_regularity_index()`, `social_jet_lag()`

## Examples

```
chronotype(example_sessions)
```

## clean_epochs *Clean epoch data*

### Description

Clean epoch data

### Usage

```
clean_epochs(epochs)
```

### Arguments

| | |
|---|---|
| epochs | A dataframe containing the epoch data |

### Value

The cleaned-up epochs dataframe

### Examples

```
epochs <- clean_epochs(nocturn::example_epochs)
clean_epochs(example_epochs)
```

## clean_sessions *Clean session data*

### Description

Clean session data

### Usage

```
clean_sessions(sessions)
```

### Arguments

| | |
|---|---|
| sessions | A dataframe containing the session data |

### Value

The cleaned-up sessions dataframe

### Examples

```
sessions <- clean_sessions(nocturn::example_sessions)
clean_sessions(example_sessions)
```

composite_phase_deviation *Calculate Composite Phase Deviation (CPD)*

## Description

This function calculates the Composite Phase Deviation (CPD) metric, used to measure the regularity of the sleep patterns.

## Usage

```
composite_phase_deviation(sessions)
```

## Arguments

| | |
|---|---|
| sessions | The sessions data frame |

## Details

This function uses columns:

- time_at_midsleep
- is_workday
- night

## Value

The Composite Phase Deviation (CPD) value

## See Also

Other sleep metrics: chronotype(), interdaily_stability(), sleep_regularity_index(), social_jet_lag()

## Examples

```
composite_phase_deviation(example_sessions)
```

`example_epochs` *Example Epoch data*

## Description

A data frame containing epoch data recorded by a Somnofy device.

## Usage

```
example_epochs
```

## Format

`example_epochs`**:**

A data frame with 18,755 rows and 15 columns. Each row represents a time-point (or epoch) in a session. Epochs are 30 seconds long. The columns are as follows:

- timestamp: The time at which the epoch was recorded in UTC.
- subject_id: The ID of the subject.
- signal_quality_mean: The mean signal quality of the epoch.
- movement_fast_mean: The mean movement detected during the epoch.
- movement_fast_nonzero_pct
- distance_mean: the distance of the subject from the device in meters.
- motion_data_count: The number of data points in the epoch (30).
- light_ambient_mean: The ambient light level during the epoch.
- sound_amplitude_mean: The sound amplitude during the epoch.
- temperature_ambient_mean: The ambient temperature during the epoch.
- humidity_mean: The ambient humidity during the epoch.
- pressure_mean: The ambient pressure during the epoch.
- indoor_air_quality_mean: The indoor air quality during the epoch.
- epoch_duration: The precise duration of the epoch (seconds).
- sleep_stage: The sleep stage as established with the VT algorithm. They are encoded as numbers 0-5

## Source

data-raw/example_epochs.csv

`example_epochs_v1` *Example Epoch data (Somnofy API v1)*

## Description

A data frame containing epoch data recorded by a Somnofy device.

## Usage

```
example_epochs_v1
```

## Format

`example_epochs_v1`:

A data frame with 1,373 rows and 16 columns. The corresponding session ID is contained in the file name. Each row represents a time-point (or epoch) in a session. Epochs are 30 seconds long. The columns are as follows:

- timestamp: The time at which the epoch was recorded in UTC.
- signal_quality_mean: The mean signal quality of the epoch.
- movement_fast_mean: The mean movement detected during the epoch.
- movement_fast_nonzero_pct
- distance_mean: the distance of the subject from the device in meters.
- motion_data_count: The number of data points in the epoch (30).
- light_ambient_mean: The ambient light level during the epoch.
- sound_amplitude_mean: The sound amplitude during the epoch.
- temperature_ambient_mean: The ambient temperature during the epoch.
- humidity_mean: The ambient humidity during the epoch.
- pressure_mean: The ambient pressure during the epoch.
- indoor_air_quality_mean: The indoor air quality during the epoch.
- epoch_duration: The precise duration of the epoch (seconds).
- sleep_stage: The sleep stage as established with the VT algorithm. They are encoded as numbers 0-5

## Source

data-raw/SEtXSxcMEhYXKQAA.example_epochs_v1.csv

`example_sessions` *Example Sessions data*

## Description

A data frame containing sessions recorded by a Somnofy device.

## Usage

```
example_sessions
```

## Format

`example_sessions`**:**

A data frame with 124 rows and 60 columns. Each row represents a session. Columns contain metadata about the session, including:

- session_start: The start time of the session in UTC.
- session_end: The end time of the session in UTC.
- subject_id: The ID of the subject.
- device_serial_number: The serial number of the device used.
- time_at_sleep_onset: The time at which the subject fell asleep.
- time_at_wakeup: The time at which the subject woke up. Columns also include various metrics averaged over the session, such as:
- mean heart rate
- mean respiration rate Finally, some columns contain environmental parameters, such as:
- Temperature
- Humidity
- Light intensity
- Noise level
- Atmospheric pressure

## Source

data-raw/example_sessions.csv

`example_sessions_v1` *Example Sessions data (Somnofy API v1)*

## Description

A data frame containing sessions recorded by a Somnofy device.

## Usage

```
example_sessions_v1
```

## Format

`example_sessions_v1`:

A data frame with 87 rows and 70 columns. Each row represents a session. Columns contain metadata about the session, including:

- user_id: The ID of the recorded subject.
- sex: The sex of the recorded subject.
- birth_year: The year of birth of the recorded subject.
- session_start: The start time of the session in UTC.
- session_end: The end time of the session in UTC.
- time_at_sleep_onset: The time at which the subject fell asleep.
- time_at_wakeup: The time at which the subject woke up. Columns also include various metrics averaged over the session, such as:
- mean heart rate
- mean respiration rate Finally, some columns contain environmental parameters, such as:
- Temperature
- Humidity
- Light intensity
- Noise level
- Atmospheric pressure

## Source

data-raw/example_sessions_v1.csv

`filter_by_age_range` *Filter sessions by age range*

## Description

Filter sessions by age range

## Usage

```
filter_by_age_range(
  sessions,
  min_age = NULL,
  max_age = NULL,
  return_mask = FALSE
)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `min_age` | The minimum age of the subjects (inclusive) |
| `max_age` | The maximum age of the subjects (inclusive) |
| `return_mask` | If TRUE, returns a logical vector indicating which sessions belong to subjects within the specified age range |

## Details

This function uses columns:

- `birth_year`
- `night`

## Value

The sessions dataframe with only the sessions that belong to subjects within the specified age range, or a logical vector if `return_mask` is TRUE

## See Also

Other filtering: `filter_by_night_range()`, `filter_by_sex()`, `filter_epochs_from_sessions()`, `remove_sessions_no_sleep()`, `select_devices()`, `select_subjects()`, `set_min_sleep_period()`, `set_min_time_in_bed()`, `set_session_sleep_onset_range()`, `set_session_start_time_range()`

## Examples

```
filtered_sessions <- filter_by_age_range(example_sessions_v1, min_age = 11, max_age = 18)
```

`filter_by_night_range` *Filter sessions for nights within a night range*

## Description

Filter sessions for nights within a night range

## Usage

```
filter_by_night_range(sessions, from_night, to_night, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `from_night` | The start night of the range (inclusive) in YYYY-MM-DD format |
| `to_night` | The end night of the range (inclusive) in YYYY-MM-DD format |
| `return_mask` | If TRUE, returns a logical vector indicating which sessions meet the night range requirement |

## Details

This function uses columns:

- `night`

## Value

The sessions dataframe with only the sessions that fall within the specified night range, or a logical vector if `return_mask` is TRUE

## See Also

Other filtering: `filter_by_age_range()`, `filter_by_sex()`, `filter_epochs_from_sessions()`, `remove_sessions_no_sleep()`, `select_devices()`, `select_subjects()`, `set_min_sleep_period()`, `set_min_time_in_bed()`, `set_session_sleep_onset_range()`, `set_session_start_time_range()`

## Examples

```
filtered_sessions <- filter_by_night_range(example_sessions, "2025-04-07", "2025-04-10")
```

filter_by_sex *Filter by sex*

## Description

Filter by sex

## Usage

```
filter_by_sex(sessions, sex, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| sessions | The sessions dataframe |
| sex | The sex to filter for (M, F, or NULL for both) |
| return_mask | If TRUE, returns a logical vector indicating which sessions belong to the specified sex |

## Details

This function uses columns:

- sex

## Value

The sessions dataframe with only the sessions that belong to the specified sex, or a logical vector if return_mask is TRUE

## See Also

Other filtering: filter_by_age_range(), filter_by_night_range(), filter_epochs_from_sessions(), remove_sessions_no_sleep(), select_devices(), select_subjects(), set_min_sleep_period(), set_min_time_in_bed(), set_session_sleep_onset_range(), set_session_start_time_range()

## Examples

```
filtered_sessions <- filter_by_sex(example_sessions_v1, "M")
```

`filter_epochs_from_sessions` *Filter epochs based on session IDs*

## Description

Filter epochs based on session IDs

## Usage

```
filter_epochs_from_sessions(epochs, sessions, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| `epochs` | The epochs dataframe |
| `sessions` | The sessions dataframe |
| `return_mask` | If TRUE, returns a logical vector indicating which epochs belong to the specified sessions |

## Details

This function uses sessions columns:

- `id` And epoch columns:
- `session_id`

## Value

The epochs dataframe with only the epochs that belong to the specified sessions, or a logical vector if `return_mask` is TRUE

## See Also

`filter_by_night_range()` to filter sessions by night range.

Other filtering: `filter_by_age_range()`, `filter_by_night_range()`, `filter_by_sex()`, `remove_sessions_no_sleep()`, `select_devices()`, `select_subjects()`, `set_min_sleep_period()`, `set_min_time_in_bed()`, `set_session_sleep_onset_range()`, `set_session_start_time_range()`

## Examples

```
# Apply filtering to sessions to keep specific nights, and filter epochs accordingly
filtered_sessions <- filter_by_night_range(example_sessions, "2025-04-07", "2025-04-10")
filtered_epochs <- filter_epochs_from_sessions(example_epochs, filtered_sessions)
```

`get_epochs_summary` *Summarise epoch information*

## Description

Display the number of sessions in the epoch data, as well as the start and end dates of the epoch data

## Usage

```
get_epochs_summary(epochs)
```

## Arguments

| | |
|---|---|
| `epochs` | The epochs dataframe |

## Details

This function uses columns:

- `timestamp`
- `session_id`

## Value

A single-row dataframe summarising epoch information

## See Also

`get_sessions_summary()` to summarise session information.

Other data tables: `get_non_complying_sessions()`, `get_removed_sessions()`, `get_sessions_summary()`

## Examples

```
get_epochs_summary(example_epochs)
```

`get_non_complying_sessions` *Get non-complying sessions (i.e. where there is more than one session on the same day)*

## Description

Get non-complying sessions (i.e. where there is more than one session on the same day)

## Usage

```
get_non_complying_sessions(sessions)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |

## Details

This function uses columns:

- `night`

## Value

The sessions dataframe with only the sessions that are non-complying

## See Also

Other data tables: `get_epochs_summary()`, `get_removed_sessions()`, `get_sessions_summary()`

## Examples

```
duplicate_sessions <- get_non_complying_sessions(example_sessions)
```

`get_removed_sessions` *Get a table of sessions that were removed during filtering*

## Description

Get a table of sessions that were removed during filtering

## Usage

```
get_removed_sessions(sessions, filtered_sessions)
```

### Arguments

| | |
|---|---|
| `sessions` | The original sessions dataframe |
| `filtered_sessions` | The filtered sessions dataframe |

### Details

This function uses columns:

- `id`
- `sleep_period`

### Value

The sessions dataframe with only the sessions that were removed during filtering

### See Also

Other data tables: `get_epochs_summary()`, `get_non_complying_sessions()`, `get_sessions_summary()`

### Examples

```
filtered_sessions <- set_session_start_time_range(example_sessions, "22:00", "06:00")
removed_sessions <- get_removed_sessions(example_sessions, filtered_sessions)
```

---

## `get_sessions_summary` *Make a summary of session information*

---

### Description

Summarise session information, including the number of sessions, mean session length, mean time at sleep onset and wakeup, subject and device ID.

### Usage

```
get_sessions_summary(sessions)
```

### Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe. |

### Details

This function uses columns:

- `time_at_sleep_onset`
- `time_at_wakeup`
- `time_in_bed`
- `sleep_period`

## Value

A single-row dataframe summarizing session information.

## See Also

`get_epochs_summary()` to summarise epoch information.

Other data tables: `get_epochs_summary()`, `get_non_complying_sessions()`, `get_removed_sessions()`

## Examples

```
get_sessions_summary(example_sessions)
```

---

# group_epochs_by_night *Create a grouping by night for epoch data*

---

## Description

The function creates a new column `night` that groups the epochs by night. Timepoints before 12 PM are considered part of the previous night.

## Usage

```
group_epochs_by_night(epochs)
```

## Arguments

| | |
|---|---|
| epochs | The epochs dataframe |

## Details

This function uses columns:

- `timestamp`

## Value

The epochs dataframe with the `night` column added

## See Also

`group_sessions_by_night()` to group session data by night.

Other time processing: `group_sessions_by_night()`, `max_time()`, `mean_time()`, `min_time()`, `parse_date()`, `parse_time()`, `sd_time()`, `shift_times_by_12h()`, `time_diff()`, `time_to_hours()`, `update_date()`

## Examples

```
epochs <- group_epochs_by_night(example_epochs)
```

group_sessions_by_night *Create a grouping by night for session data*

## Description

The function creates a new column `night` that groups the sessions by night depending on their start time. Sessions that start before 12 PM are considered part of the previous night.

## Usage

```
group_sessions_by_night(sessions)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |

## Details

This function uses columns:

- `session_start`

## Value

The sessions dataframe with the `night` column added

## See Also

`group_epochs_by_night()` to group epoch data by night.

Other time processing: `group_epochs_by_night()`, `max_time()`, `mean_time()`, `min_time()`, `parse_date()`, `parse_time()`, `sd_time()`, `shift_times_by_12h()`, `time_diff()`, `time_to_hours()`, `update_date()`

## Examples

```
sessions <- group_sessions_by_night(example_sessions)
```

## interdaily_stability *Calculate Interdaily Stability (IS)*

### Description

This function calculates the Interdaily Stability (IS) metric from a binary awake/asleep variable

### Usage

```
interdaily_stability(epochs)
```

### Arguments

| | |
|---|---|
| epochs | The epochs data frame |

### Details

This function uses columns:

- timestamp
- is_asleep

### Value

The Interdaily Stability (IS) value

### See Also

Other sleep metrics: chronotype(), composite_phase_deviation(), sleep_regularity_index(), social_jet_lag()

### Examples

```
interdaily_stability(example_epochs)
```

## load_batch *Load session or epoch data in batch mode*

### Description

Load session or epoch data in batch mode

### Usage

```
load_batch(
  folder_path = NULL,
  file_list = NULL,
  file_names = NULL,
  pattern = NULL,
  type = "sessions"
)
```

### Arguments

| | |
|---|---|
| folder_path | The path to the folder containing session files (do not use with file_list) |
| file_list | The list of file paths to load (do not use with folder_path) |
| file_names | An optional vector of file names corresponding to the files in file_list |
| pattern | An optional pattern to filter files in the folder |
| type | The type of data to load: "sessions" or "epochs" |

### Value

A dataframe containing the combined session data from all matching files in the folder

### See Also

Other data loading: load_epochs(), load_sessions(), read_edf_epochs(), read_edf_sessions()

### Examples

```
tf1 <- tempfile(fileext = ".csv")
tf2 <- tempfile(fileext = ".csv")
utils::write.csv(data.frame(id = 1), tf1, row.names = FALSE)
utils::write.csv(data.frame(id = 1), tf2, row.names = FALSE)
load_batch(file_list = c(tf1, tf2))
```

---

## load_epochs *Load epoch data*

---

### Description

Load epoch data

### Usage

```
load_epochs(epochs_file, file_name = NULL)
```

### Arguments

| | |
|---|---|
| epochs_file | The path to the epochs file |
| file_name | An optional file name to be recorded in the epochs table |

### Details

The function loads the epoch data from a file and groups the epochs by night. Supported formats: CSV, Excel, EDF.

### Value

A dataframe containing the epoch data

### See Also

Other data loading: load_batch(), load_sessions(), read_edf_epochs(), read_edf_sessions()

### Examples

```
tf <- tempfile(fileext = ".csv")
utils::write.csv(data.frame(id = 1), tf, row.names = FALSE)
load_epochs(tf)
```

---

## load_sessions *Load session data*

---

### Description

Load session data

### Usage

```
load_sessions(sessions_file)
```

### Arguments

| | |
|---|---|
| sessions_file | The path to the sessions file |

### Details

The function loads the session data from a file Supported formats: CSV, Excel, EDF.

### Value

A dataframe containing the session data

### See Also

Other data loading: load_batch(), load_epochs(), read_edf_epochs(), read_edf_sessions()

## Examples

```
tf <- tempfile(fileext = ".csv")
utils::write.csv(data.frame(id = 1), tf, row.names = FALSE)
load_sessions(tf)
```

---

`max_time` *Calculate the maximum time from 12pm to 12pm*

---

## Description

This function calculates the maximum time from a vector of time strings in the format "YYYY-MM-DD HH:MM:SS". It considers a time window from 12pm to 12pm the next day, so 11:00 is considered later than 13:00.

## Usage

```
max_time(time_vector)
```

## Arguments

| | |
|---|---|
| `time_vector` | A vector of time strings in the format "YYYY-MM-DD HH:MM:SS". |

## Value

A string representing the maximum time in the format "HH:MM".

## See Also

`min_time()` to calculate the minimum time in the same format.

Other time processing: `group_epochs_by_night()`, `group_sessions_by_night()`, `mean_time()`, `min_time()`, `parse_date()`, `parse_time()`, `sd_time()`, `shift_times_by_12h()`, `time_diff()`, `time_to_hours()`, `update_date()`

## Examples

```
max_time(c("2025-04-08 23:00:00", "2025-04-09 01:00:00", "2025-04-09 02:30:00"))
```

mean_time *Calculate the mean time from a vector of time strings*

## Description

This function calculates the mean time from a vector of time strings in the format "YYYY-MM-DD HH:MM:SS".

## Usage

```
mean_time(time_vector, unit = "HH:MM")
```

## Arguments

| | |
|---|---|
| time_vector | A vector of time strings in format "YYYY-MM-DD HH:MM:SS", "HH:MM:SS" or "HH:MM". |
| unit | The unit of time for the result. Can be "HH:MM" (default), "hour", "minute" or "second". |

## Value

A string representing the mean time in the format "HH:MM".

## See Also

Other time processing: group_epochs_by_night(), group_sessions_by_night(), max_time(), min_time(), parse_date(), parse_time(), sd_time(), shift_times_by_12h(), time_diff(), time_to_hours(), update_date()

## Examples

```
# Use on a vector of time strings representing full dates
time_vector <- c("2025-04-08 23:00:00", "2025-04-09 01:00:00")
mean_time(time_vector)

# Use on time-only strings
time_vector <- c("22:56", "01:32")
mean_time(time_vector)

# Use on a dataframe column
mean_time(example_sessions$time_at_sleep_onset)
```

## min_time *Calculate the minimum time from 12pm to 12pm*

### Description

This function calculates the minimum time from a vector of time strings in the format "YYYY-MM-DD HH:MM:SS". It considers a time window from 12pm to 12pm the next day, so 11:00 is considered later than 13:00.

### Usage

```
min_time(time_vector)
```

### Arguments

| | |
|---|---|
| time_vector | A vector of time strings in the format "YYYY-MM-DD HH:MM:SS". |

### Value

A string representing the minimum time in the format "HH:MM".

### See Also

max_time() to calculate the maximum time in the same format.

Other time processing: group_epochs_by_night(), group_sessions_by_night(), max_time(), mean_time(), parse_date(), parse_time(), sd_time(), shift_times_by_12h(), time_diff(), time_to_hours(), update_date()

### Examples

```
min_time(c("2025-04-08 23:00:00", "2025-04-09 01:00:00", "2025-04-09 02:30:00"))
```

## nocturn *nocturn app*

### Description

This function launches the nocturn app, a Shiny application for visualizing and analyzing sleep data.

### Usage

```
nocturn()
```

### Value

No return value, called for side-effects

## Examples

```
if(interactive()){nocturn()}
```

---

`parse_date` *Parse a vector of date strings into Date objects*

---

## Description

This function parses a vector of date strings into Date objects. All formats containing year, month, and day information are supported.

## Usage

```
parse_date(date_vector)
```

## Arguments

| | |
|---|---|
| `date_vector` | A vector of date strings |

## Value

A vector of Date objects

## See Also

Other time processing: `group_epochs_by_night()`, `group_sessions_by_night()`, `max_time()`, `mean_time()`, `min_time()`, `parse_time()`, `sd_time()`, `shift_times_by_12h()`, `time_diff()`, `time_to_hours()`, `update_date()`

## Examples

```
parse_date("2026-01-02T14:33:09")
```

---

`parse_time` *Parse a vector of time strings into POSIXct objects*

---

## Description

This function parses a vector of time strings into POSIXct objects. Supported formats include "YYYY-MM-DD HH:MM:SS", "YYYY-MM-DD HH:MM", "HH:MM:SS", and "HH:MM". Time-zone information is ignored.

## Usage

```
parse_time(time_vector)
```

### Arguments

| | |
|---|---|
| time_vector | A vector of time strings |

### Value

A vector of POSIXct objects

### See Also

Other time processing: group_epochs_by_night(), group_sessions_by_night(), max_time(), mean_time(), min_time(), parse_date(), sd_time(), shift_times_by_12h(), time_diff(), time_to_hours(), update_date()

### Examples

```
parse_time("2026-01-01T14:34:09")
```

---

## plot_bedtimes_waketimes

*Plot bedtimes and waketimes*

---

### Description

Plot bedtimes and waketimes

### Usage

```
plot_bedtimes_waketimes(sessions, groupby = "night", color_by = "default")
```

### Arguments

| | |
|---|---|
| sessions | The sessions dataframe |
| groupby | The grouping variable for the plot. Can be "night", "workday", or "weekday". |
| color_by | The variable to color the bars by. Can be "default" or any other column name in the sessions dataframe. Note that if color_by is anything else than "default", groupby will be set to "night". |

### Details

This function uses columns:

- night
- time_at_sleep_onset
- time_at_wakeup
- is_workday

## Value

A ggplot graph showing the bedtimes and waketimes

## Examples

```
plot_bedtimes_waketimes(example_sessions)
```

---

`plot_hypnogram` *Plot Hypnogram*

---

## Description

Plot Hypnogram

## Usage

```
plot_hypnogram(epochs)
```

## Arguments

| | |
|---|---|
| `epochs` | The epochs dataframe |

## Details

This function uses columns:

- `timestamp`
- `sleep_stage`

## Value

A ggplot object showing the hypnogram as bars

## See Also

Other plot epochs: `plot_sleep_spiral()`, `plot_timeseries()`

## Examples

```
plot_hypnogram(example_epochs)
```

`plot_sleep_bubbles` *Plot Sleep Bubbles*

## Description

This function creates a bubble plot of sleep sessions, where the size and colour of the bubbles represents the sleep duration.

## Usage

```
plot_sleep_bubbles(sessions, color_by = "default", bubble_size = 10)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe. |
| `color_by` | The variable to color the bubbles by. Can be "default" or any other column name in the sessions dataframe. |
| `bubble_size` | The size of the bubbles. Default is 10. |

## Details

This function uses columns:

- `sleep_period`
- `night`

## Value

A ggplot object containing the sleep bubbles graph.

## See Also

Other plot sessions: `plot_sleep_clock()`, `plot_timeseries_sessions()`

## Examples

```
plot_sleep_bubbles(example_sessions)
```

plot_sleep_clock *Plot Sleep Clock*

## Description

Plot Sleep Clock

## Usage

```
plot_sleep_clock(sessions, color_by = "default")
```

## Arguments

| | |
|---|---|
| sessions | The sessions dataframe |
| color_by | The variable to color the segments by. Can be "default" or any other column name in the sessions dataframe. |

## Details

This function uses columns:

- time_at_sleep_onset
- time_at_wakeup
- night

## Value

A ggplot object showing the sleep clock

## See Also

Other plot sessions: plot_sleep_bubbles(), plot_timeseries_sessions()

## Examples

```
plot_sleep_clock(example_sessions)
```

`plot_sleep_spiral` *Plot Sleep Spiral*

## Description

Plot Sleep Spiral

## Usage

```
plot_sleep_spiral(epochs, color_by = "default")
```

## Arguments

| | |
|---|---|
| `epochs` | The epochs dataframe |
| `color_by` | The variable to color the spiral by. Can be "default" or any other column name in the epochs dataframe. |

## Details

This function uses columns:

- `timestamp`
- `is_asleep`

## Value

A ggplot object showing the sleep spiral

## See Also

Other plot epochs: `plot_hypnogram()`, `plot_timeseries()`

## Examples

```
plot_sleep_spiral(example_epochs)
```

`plot_timeseries` *Plot epoch time series data for a given variable*

## Description

Plot epoch time series data for a given variable

## Usage

```
plot_timeseries(epochs, variable, color_by = "default", exclude_zero = FALSE)
```

## Arguments

| | |
|---|---|
| `epochs` | The epochs dataframe |
| `variable` | The variable to plot (e.g., "temperature_ambient_mean") |
| `color_by` | The variable to color the points by. Can be "default" or any other column name in the epochs dataframe. |
| `exclude_zero` | Logical, whether to exclude zero values from the plot (default: FALSE) |

## Details

This function uses columns:

- `timestamp`
- `night`

## Value

A ggplot object

## See Also

`plot_timeseries_sessions()` to plot session data.

Other plot epochs: `plot_hypnogram()`, `plot_sleep_spiral()`

## Examples

```
plot_timeseries(example_epochs, variable="signal_quality_mean")
```

`plot_timeseries_sessions` *Plot session time series data for a given variable*

## Description

Plot session time series data for a given variable

## Usage

```
plot_timeseries_sessions(
  sessions,
  variable,
  color_by = "default",
  exclude_zero = FALSE
)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `variable` | The variable to plot (e.g., "time_at_sleep_onset") |
| `color_by` | The variable to color the points by. Can be "default" or any other column name in the sessions dataframe. |
| `exclude_zero` | Logical, whether to exclude zero values from the plot (default: FALSE) |

## Details

This function uses columns:

- `night`

## Value

A ggplot object

## See Also

`plot_timeseries()` to plot epoch data.

Other plot sessions: `plot_sleep_bubbles()`, `plot_sleep_clock()`

## Examples

```
plot_timeseries_sessions(example_sessions, variable="time_at_midsleep")
```

## read_edf_epochs *Read EDF Epochs*

### Description

Read EDF Epochs

### Usage

```
read_edf_epochs(file)
```

### Arguments

| | |
|---|---|
| file | The path to the EDF file |

### Details

The function reads the signals of the EDF file to extract epoch information. It must contain a column for timestamps and a column for sleep stage annotations.

### Value

A dataframe containing the epoch data extracted from the EDF file signals

### See Also

Other data loading: load_batch(), load_epochs(), load_sessions(), read_edf_sessions()

### Examples

```
edf <- system.file("extdata", "mini.edf", package = "nocturn")
read_edf_epochs(edf)
```

## read_edf_sessions *Read EDF Sessions*

### Description

Read EDF Sessions

### Usage

```
read_edf_sessions(file)
```

### Arguments

| | |
|---|---|
| file | The path to the EDF file |

## Details

The function reads the header of the EDF file to extract session information such as start time, duration, and calculates end time and midsleep time.

## Value

A dataframe containing the session data extracted from the EDF file header

## See Also

Other data loading: `load_batch()`, `load_epochs()`, `load_sessions()`, `read_edf_epochs()`

## Examples

```
edf <- system.file("extdata", "mini.edf", package = "nocturn")
read_edf_sessions(edf)
```

---

# remove_sessions_no_sleep

*Remove sessions with no sleep*

---

## Description

Remove sessions with no sleep

## Usage

```
remove_sessions_no_sleep(sessions, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `return_mask` | If TRUE, returns a logical vector indicating which sessions have a sleep period greater than 0 |

## Details

This function uses columns:

- `sleep_period`

## Value

The sessions dataframe with only the sessions that have a sleep period greater than 0, or a logical vector if `return_mask` is TRUE

## See Also

Other filtering: filter_by_age_range(), filter_by_night_range(), filter_by_sex(), filter_epochs_from_sessio
select_devices(), select_subjects(), set_min_sleep_period(), set_min_time_in_bed(),
set_session_sleep_onset_range(), set_session_start_time_range()

## Examples

```
filtered_sessions <- remove_sessions_no_sleep(example_sessions)
```

---

# sd_time *Calculate the circular standard deviation of a vector of times*

---

## Description

This function calculates the standard deviation of a vector of time strings, accounting for the circular nature of time (e.g., 23:59 is close to 00:00).

## Usage

```
sd_time(time_vector, unit = "hour")
```

## Arguments

| | |
|---|---|
| time_vector | A vector of time strings in format "YYYY-MM-DD HH:MM:SS", "HH:MM:SS" or "HH:MM". |
| unit | The unit of time for the result. Can be "second", "minute", or "hour". Default is "hour". |

## Value

A numeric value representing the standard deviation in the specified unit.

## See Also

Other time processing: group_epochs_by_night(), group_sessions_by_night(), max_time(), mean_time(), min_time(), parse_date(), parse_time(), shift_times_by_12h(), time_diff(), time_to_hours(), update_date()

## Examples

```
sd_time(c("23:59", "00:01"))
```

`select_devices` *Select devices by ID*

## Description

Select devices by ID

## Usage

```
select_devices(sessions, device_ids, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `device_ids` | The device IDs to select |
| `return_mask` | If TRUE, returns a logical vector indicating which sessions were recorded by the specified devices |

## Details

This function uses columns:

- `device_id`

## Value

The sessions dataframe with only the sessions recorded by the specified devices, or a logical vector if `return_mask` is TRUE

## See Also

`select_subjects()` to select sessions by subject ID.

Other filtering: `filter_by_age_range()`, `filter_by_night_range()`, `filter_by_sex()`, `filter_epochs_from_sessio`
`remove_sessions_no_sleep()`, `select_subjects()`, `set_min_sleep_period()`, `set_min_time_in_bed()`,
`set_session_sleep_onset_range()`, `set_session_start_time_range()`

## Examples

```
filtered_sessions <- select_devices(example_sessions, c("VTGVSRTHCA"))
```

`select_subjects` *Select subjects by ID*

## Description

Select subjects by ID

## Usage

```
select_subjects(sessions, subject_ids, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `subject_ids` | The subject IDs to select |
| `return_mask` | If TRUE, returns a logical vector indicating which sessions belong to the specified subjects |

- `subject_id`

## Details

This function uses columns:

## Value

The sessions dataframe with only the sessions that belong to the specified subjects, or a logical vector if `return_mask` is TRUE

## See Also

`select_devices()` to select sessions by device ID.

Other filtering: `filter_by_age_range()`, `filter_by_night_range()`, `filter_by_sex()`, `filter_epochs_from_sessio`
`remove_sessions_no_sleep()`, `select_devices()`, `set_min_sleep_period()`, `set_min_time_in_bed()`,
`set_session_sleep_onset_range()`, `set_session_start_time_range()`

## Examples

```
filtered_sessions <- select_subjects(example_sessions, c("sub_01JNDH3Z5NP0PSV82NFBGPV31X"))
```

`set_min_sleep_period` *Set minimum sleep period*

## Description

Set minimum sleep period

## Usage

```
set_min_sleep_period(sessions, min_sleep_period, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `min_sleep_period` | The minimum sleep period in hours |
| `return_mask` | If TRUE, return a logical vector indicating which sessions meet the minimum sleep period requirement |

## Details

This function uses columns:

- `sleep_period`

## Value

The sessions dataframe with only the sessions that meet the minimum sleep period requirement, or a logical vector if `return_mask` is TRUE

## See Also

Other filtering: `filter_by_age_range()`, `filter_by_night_range()`, `filter_by_sex()`, `filter_epochs_from_sessio`
`remove_sessions_no_sleep()`, `select_devices()`, `select_subjects()`, `set_min_time_in_bed()`,
`set_session_sleep_onset_range()`, `set_session_start_time_range()`

## Examples

```
filtered_sessions <- set_min_sleep_period(example_sessions, 2)
```

`set_min_time_in_bed` *Set minimum time in bed*

## Description

Set minimum time in bed

## Usage

```
set_min_time_in_bed(sessions, min_time_in_bed, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `min_time_in_bed` | The minimum time in bed in hours |
| `return_mask` | If TRUE, return a logical vector indicating which sessions meet the minimum time in bed requirement |

## Details

This function uses columns:

- `time_in_bed`

## Value

The sessions dataframe with only the sessions that meet the minimum time in bed requirement, or a logical vector if `return_mask` is TRUE

## See Also

Other filtering: `filter_by_age_range()`, `filter_by_night_range()`, `filter_by_sex()`, `filter_epochs_from_sessio`
`remove_sessions_no_sleep()`, `select_devices()`, `select_subjects()`, `set_min_sleep_period()`,
`set_session_sleep_onset_range()`, `set_session_start_time_range()`

## Examples

```
filtered_sessions <- set_min_time_in_bed(example_sessions, 2)
```

`set_session_sleep_onset_range` *Set sleep onset time range*

## Description

Set sleep onset time range

## Usage

```
set_session_sleep_onset_range(
  sessions,
  from_time,
  to_time,
  return_mask = FALSE
)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `from_time` | Include sessions where sleep started after this time (in format HH:MM) |
| `to_time` | Include sessions where sleep started before this time (in format HH:MM) |
| `return_mask` | If TRUE, returns a logical vector indicating which sessions meet the sleep onset time range requirement |

## Details

This function uses columns:

- `time_at_sleep_onset`

## Value

The sessions dataframe with only the sessions where sleep started within the specified time range, or a logical vector if `return_mask` is TRUE

## See Also

`set_session_start_time_range()` to filter sessions based on start time.

Other filtering: `filter_by_age_range()`, `filter_by_night_range()`, `filter_by_sex()`, `filter_epochs_from_sessio`
`remove_sessions_no_sleep()`, `select_devices()`, `select_subjects()`, `set_min_sleep_period()`,
`set_min_time_in_bed()`, `set_session_start_time_range()`

## Examples

```
filtered_sessions <- set_session_sleep_onset_range(example_sessions, "22:00", "06:00")
```

set_session_start_time_range *Set session start time range*

## Description

Set session start time range

## Usage

```
set_session_start_time_range(sessions, from_time, to_time, return_mask = FALSE)
```

## Arguments

| | |
|---|---|
| sessions | The sessions dataframe |
| from_time | Include sessions that started after this time (in format HH:MM) |
| to_time | Include sessions that started before this time (in format HH:MM) |
| return_mask | If TRUE, returns a logical vector indicating which sessions meet the time range requirement |

## Details

This function uses columns:

- session_start

## Value

The sessions dataframe with only the sessions that started within the specified time range, or a logical vector if return_mask is TRUE

## See Also

set_session_sleep_onset_range() to filter sessions based on sleep onset time.

Other filtering: filter_by_age_range(), filter_by_night_range(), filter_by_sex(), filter_epochs_from_sessio
remove_sessions_no_sleep(), select_devices(), select_subjects(), set_min_sleep_period(),
set_min_time_in_bed(), set_session_sleep_onset_range()

## Examples

```
filtered_sessions <- set_session_start_time_range(example_sessions, "22:00", "06:00")
```

`shift_times_by_12h` *Shift times to break at 12 pm*

## Description

This function shifts times so that the day starts at 12 PM. This is useful for plotting night data

## Usage

```
shift_times_by_12h(times)
```

## Arguments

| | |
|---|---|
| `times` | A vector of times in POSIXct format, character convertible to POSIXct, or numerical (in hours). |

## Value

A vector of times in POSIXct format (or numerical if numerical provided as input) shifted to start at 12 PM

## See Also

Other time processing: `group_epochs_by_night()`, `group_sessions_by_night()`, `max_time()`, `mean_time()`, `min_time()`, `parse_date()`, `parse_time()`, `sd_time()`, `time_diff()`, `time_to_hours()`, `update_date()`

## Examples

```
# Shift a vector of times in HH:MM format
shift_times_by_12h(c("02:30", "16:00"))
#> "14:30" "04:00"

# Shift times in YYYY-MM-DD HH:MM:SS format
shift_times_by_12h(c("2025-04-08 23:00:00", "2025-04-09 01:00:00"))
#> "2025-04-08 11:00" "2025-04-09 13:00"

# Shift sessions start times to start at 12 PM
shifted_times <- shift_times_by_12h(example_sessions$session_start)

# Use dplyr::mutate to dicrectly add the shifted times to a dataframe
epochs <- example_epochs |>
  dplyr::mutate(shifted_time = shift_times_by_12h(timestamp))
```

`sleep_regularity_index` *Calculate the Sleep Regularity Index (SRI)*

## Description

The Sleep Regularity Index (SRI) is a measure of the regularity of sleep patterns. It is calculated as the percentage of epochs where the sleep state remains the same after 24 hours.

## Usage

```
sleep_regularity_index(epochs)
```

## Arguments

| | |
|---|---|
| `epochs` | The epochs data frame |

## Details

This function uses columns:

- `timestamp`
- `is_asleep`

## Value

The Sleep Regularity Index (SRI) value

## See Also

Other sleep metrics: `chronotype()`, `composite_phase_deviation()`, `interdaily_stability()`, `social_jet_lag()`

## Examples

```
sleep_regularity_index(example_epochs)
```

`sleep_report` *Generate a patient sleep report in PDF format*

## Description

This function generates a sleep report in PDF format using an SVG template. It is designed to work with Somnofy data. Other data types may be supported in the future.

## Usage

```
sleep_report(sessions, title = "", output_file = "Sleep_report.pdf")
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions dataframe |
| `title` | The title of the report. Default is an empty string. |
| `output_file` | Path for the output PDF. Default is "Sleep_report.pdf" |

## Details

This function uses columns:

- `night`
- `time_at_sleep_onset`
- `time_at_wakeup`
- `time_at_midsleep`
- `sleep_onset_latency`
- `sleep_period`
- `time_in_bed`

## Value

No return value. Called for side-effects

## Examples

```
sleep_report(example_sessions)
```

## sleeptimes_boxplot *Plot boxplots for sleep onset, midsleep, and wakeup times*

### Description

Plot boxplots for sleep onset, midsleep, and wakeup times

### Usage

```
sleeptimes_boxplot(sessions, circular = FALSE)
```

### Arguments

| | |
|---|---|
| sessions | The sessions dataframe |
| circular | Whether to output a circular plot (default FALSE) |

### Details

This function uses columns:

- time_at_sleep_onset
- time_at_wakeup
- time_at_midsleep

### Value

A ggplot object with three horizontal boxplots (onset, midsleep, wakeup)

### Examples

```
sleeptimes_boxplot(example_sessions)
```

## sleeptimes_density *Plot density curves for sleep onset, midsleep, and wakeup times with a dashed line showing the median*

### Description

Plot density curves for sleep onset, midsleep, and wakeup times with a dashed line showing the median

### Usage

```
sleeptimes_density(sessions, adjust = 1, circular = FALSE)
```

### Arguments

| | |
|---|---|
| sessions | The sessions dataframe |
| adjust | The bandwidth adjustment for the density estimate (default 1) |
| circular | Whether to output a circular plot (default FALSE) |

### Details

This function uses columns:

- time_at_sleep_onset
- time_at_wakeup
- time_at_midsleep

### Value

A ggplot object with three overlaid density curves (sleep onset, midsleep, wakeup)

### Examples

```
sleeptimes_density(example_sessions)
```

---

## sleeptimes_histogram *Plot histograms for sleep onset, midsleep, and wakeup times*

---

### Description

Plot histograms for sleep onset, midsleep, and wakeup times

### Usage

```
sleeptimes_histogram(sessions, binwidth = 0.25, circular = FALSE)
```

### Arguments

| | |
|---|---|
| sessions | The sessions dataframe |
| binwidth | The width of the bins for the histogram (default 0.25) |
| circular | Whether to output a circular plot (default FALSE) |

### Details

This function uses columns:

- time_at_sleep_onset
- time_at_wakeup
- time_at_midsleep

## Value

A ggplot object with three overlaid histograms (sleep onset, midsleep, wakeup)

## Examples

```
sleeptimes_histogram(example_sessions)
```

---

`social_jet_lag` *Calculate Social Jet Lag*

---

## Description

This function calculates the Social Jet Lag (SJL) metric as the difference in mid-sleep times between workdays and free days.

## Usage

```
social_jet_lag(sessions)
```

## Arguments

| | |
|---|---|
| `sessions` | The sessions data frame |

## Details

This function uses columns:

- `time_at_midsleep`
- `is_workday`

## Value

The Social Jet Lag (SJL) value in hours

## See Also

Other sleep metrics: `chronotype()`, `composite_phase_deviation()`, `interdaily_stability()`, `sleep_regularity_index()`

## Examples

```
social_jet_lag(example_sessions)
```

## time_diff *Compute the forward time difference from t1 to t2 (wrapping at 24)*

### Description

This function returns the time from t1 to t2, always moving forward on the clock. For example, from 07:00 to 22:00 is 15 hours, from 22:00 to 07:00 is 9 hours.

### Usage

```
time_diff(t1, t2, unit = "hour")
```

### Arguments

| | |
|---|---|
| t1 | First time (character, POSIXct, or numeric hour) |
| t2 | Second time (character, POSIXct, or numeric hour) |
| unit | The unit of time. Can be "second", "minute", or "hour". Default is "hour". |

### Value

The forward difference in the specified unit (numeric, always positive, 0 <= x < 24)

### See Also

Other time processing: group_epochs_by_night(), group_sessions_by_night(), max_time(), mean_time(), min_time(), parse_date(), parse_time(), sd_time(), shift_times_by_12h(), time_to_hours(), update_date()

### Examples

```
time_diff("07:00", "22:00") # 15
time_diff("22:00", "07:00") # 9
time_diff("07:00", "22:00", unit = "minute") # 540
```

## time_to_hours *Convert time vector to numeric hours*

### Description

This function converts a vector of time strings or POSIXct objects to numeric hours.

### Usage

```
time_to_hours(time_vector)
```

### Arguments

| | |
|---|---|
| time_vector | A vector of time strings |

### Details

See parse_time() for supported time formats.

### Value

A numeric vector representing the time in hours

### See Also

Other time processing: group_epochs_by_night(), group_sessions_by_night(), max_time(), mean_time(), min_time(), parse_date(), parse_time(), sd_time(), shift_times_by_12h(), time_diff(), update_date()

### Examples

```
time_to_hours(c("2026-10-12T14:20:09"))
```

---

## update_date *Update the date component of a POSIXct time object*

---

### Description

This function updates the date component of a POSIXct time object while preserving the time component.

### Usage

```
update_date(time, date)
```

### Arguments

| | |
|---|---|
| time | A POSIXct time object or a character string convertible to POSIXct |
| date | A Date object or a character string convertible to Date |

### Value

A POSIXct time object with the updated date

### See Also

Other time processing: group_epochs_by_night(), group_sessions_by_night(), max_time(), mean_time(), min_time(), parse_date(), parse_time(), sd_time(), shift_times_by_12h(), time_diff(), time_to_hours()

## Examples

```
update_date("2026-01-01T14:09:09", "0000-01-01")
```

# Index


∗ **data loading**
load_batch, 19
load_epochs, 20
load_sessions, 21
read_edf_epochs, 33
read_edf_sessions, 33
∗ **data tables**
get_epochs_summary, 14
get_non_complying_sessions, 15
get_removed_sessions, 15
get_sessions_summary, 16
∗ **datasets**
example_epochs, 6
example_epochs_v1, 7
example_sessions, 8
example_sessions_v1, 9
∗ **filtering**
filter_by_age_range, 10
filter_by_night_range, 11
filter_by_sex, 12
filter_epochs_from_sessions, 13
remove_sessions_no_sleep, 34
select_devices, 36
select_subjects, 37
set_min_sleep_period, 38
set_min_time_in_bed, 39
set_session_sleep_onset_range, 40
set_session_start_time_range, 41
∗ **plot epochs**
plot_hypnogram, 27
plot_sleep_spiral, 30
plot_timeseries, 31
∗ **plot sessions**
plot_sleep_bubbles, 28
plot_sleep_clock, 29
plot_timeseries_sessions, 32
∗ **sleep metrics**
chronotype, 3
composite_phase_deviation, 5
interdaily_stability, 19
sleep_regularity_index, 43
social_jet_lag, 47
∗ **time processing**
group_epochs_by_night, 17
group_sessions_by_night, 18
max_time, 22
mean_time, 23
min_time, 24
parse_date, 25
parse_time, 25
sd_time, 35
shift_times_by_12h, 42
time_diff, 48
time_to_hours, 48
update_date, 49

chronotype, 3
chronotype(), *5*, *19*, *43*, *47*
clean_epochs, 4
clean_sessions, 4
composite_phase_deviation, 5
composite_phase_deviation(), *3*, *19*, *43*, *47*

example_epochs, 6
example_epochs_v1, 7
example_sessions, 8
example_sessions_v1, 9

filter_by_age_range, 10
filter_by_age_range(), *11–13*, *35–41*
filter_by_night_range, 11
filter_by_night_range(), *10*, *12, 13*, *35–41*
filter_by_sex, 12
filter_by_sex(), *10, 11*, *13*, *35–41*
filter_epochs_from_sessions, 13
filter_epochs_from_sessions(), *10–12*, *35–41*

get_epochs_summary, 14
get_epochs_summary(), *15–17*
get_non_complying_sessions, 15
get_non_complying_sessions(), *14*, *16, 17*
get_removed_sessions, 15
get_removed_sessions(), *14, 15*, *17*
get_sessions_summary, 16
get_sessions_summary(), *14–16*
group_epochs_by_night, 17
group_epochs_by_night(), *18*, *22–26*, *35*, *42*, *48, 49*
group_sessions_by_night, 18
group_sessions_by_night(), *17*, *22–26*, *35*, *42*, *48, 49*

interdaily_stability, 19
interdaily_stability(), *3*, *5*, *43*, *47*

load_batch, 19
load_batch(), *21*, *33, 34*
load_epochs, 20
load_epochs(), *20, 21*, *33, 34*
load_sessions, 21
load_sessions(), *20, 21*, *33, 34*

max_time, 22
max_time(), *17, 18*, *23–26*, *35*, *42*, *48, 49*
mean_time, 23
mean_time(), *17, 18*, *22*, *24–26*, *35*, *42*, *48, 49*
min_time, 24
min_time(), *17, 18*, *22, 23*, *25, 26*, *35*, *42*, *48, 49*

nocturn, 24

parse_date, 25
parse_date(), *17, 18*, *22–24*, *26*, *35*, *42*, *48, 49*
parse_time, 25
parse_time(), *17, 18*, *22–25*, *35*, *42*, *48, 49*
plot_bedtimes_waketimes, 26
plot_hypnogram, 27
plot_hypnogram(), *30, 31*
plot_sleep_bubbles, 28
plot_sleep_bubbles(), *29*, *32*
plot_sleep_clock, 29
plot_sleep_clock(), *28*, *32*
plot_sleep_spiral, 30
plot_sleep_spiral(), *27*, *31*
plot_timeseries, 31
plot_timeseries(), *27*, *30*, *32*
plot_timeseries_sessions, 32
plot_timeseries_sessions(), *28, 29*, *31*

read_edf_epochs, 33
read_edf_epochs(), *20, 21*, *34*
read_edf_sessions, 33
read_edf_sessions(), *20, 21*, *33*
remove_sessions_no_sleep, 34
remove_sessions_no_sleep(), *10–13*, *36–41*

sd_time, 35
sd_time(), *17, 18*, *22–26*, *42*, *48, 49*
select_devices, 36
select_devices(), *10–13*, *35*, *37–41*
select_subjects, 37
select_subjects(), *10–13*, *35, 36*, *38–41*
set_min_sleep_period, 38
set_min_sleep_period(), *10–13*, *35–37*, *39–41*
set_min_time_in_bed, 39
set_min_time_in_bed(), *10–13*, *35–38*, *40, 41*
set_session_sleep_onset_range, 40
set_session_sleep_onset_range(), *10–13*, *35–39*, *41*
set_session_start_time_range, 41
set_session_start_time_range(), *10–13*, *35–40*
shift_times_by_12h, 42
shift_times_by_12h(), *17, 18*, *22–26*, *35*, *48, 49*
sleep_regularity_index, 43
sleep_regularity_index(), *3*, *5*, *19*, *47*
sleep_report, 44
sleeptimes_boxplot, 45
sleeptimes_density, 45
sleeptimes_histogram, 46
social_jet_lag, 47
social_jet_lag(), *3*, *5*, *19*, *43*

time_diff, 48
time_diff(), *17, 18*, *22–26*, *35*, *42*, *49*
time_to_hours, 48
time_to_hours(), *17, 18*, *22–26*, *35*, *42*, *48, 49*

update_date, 49

update_date(), *17, 18*, *22–26*, *35*, *42*, *48, 49*