Package ‘rsleep’

March 10, 2024

Type Package

Title Analysis of Sleep Data

Version 1.0.12

Description A toolbox for sleep data processing, visualization and analysis. Tools for state of the art automatic sleep stages scoring.

License MIT + file LICENSE

Encoding UTF-8

URL `https://rsleep.org/`, `https://github.com/boupetch/rsleep`

BugReports `https://github.com/boupetch/rsleep/issues`

Imports abind, dplyr, edfReader, ggplot2, jsonlite, psd, signal, xml2, readr, xts

Suggests caret, gridExtra, keras, knitr, reshape2, rmarkdown, testthat, RSQLite, DBI, SleepCycles, devtools

RoxygenNote 7.3.0

VignetteBuilder knitr

NeedsCompilation no

Author Paul Bouchequet [aut, cre] (<`https://orcid.org/0000-0002-4033-2929`>)

Maintainer Paul Bouchequet <`paul.bouchequet@frenchkpi.com`>

Repository CRAN

Date/Publication 2024-03-09 23:30:06 UTC

R topics documented:

`a7` *A7 spindle detection algorithm*

Description

A sleep spindle detection algorithm that emulates human expert spindle scoring

Usage

```
a7(
  x,
  sRate,
  window = 0.3,
  step = 0.1,
  butter_order = 5,
  A7absSigPow = 1.25,
  A7relSigPow = 1.6,
  A7sigmaCov = 1.3,
  A7sigmaCorr = 0.69
)
```

Arguments

`x`	EEG signal in uV.
`sRate`	Sample rate of the signal.
`window`	Size of the window in seconds. Default: 0.3
`step`	Size of the step between windows in seconds. Default: 0.1
`butter_order`	Order of the Butterworth filters. Default: 5
`A7absSigPow`	A7absSigPow treshold. Default: 1.25
`A7relSigPow`	A7relSigPow treshold. Default: 1.6
`A7sigmaCov`	A7sigmaCov treshold. Default: 1.3
`A7sigmaCorr`	A7sigmaCorr treshold. Default: 0.69

Details

A sleep spindle detection algorithm based on 4 features computed along segmented signal according to ‘window‘ size and ‘step‘ size parameters.

1. Absolute sigma power

$$A7absSigPow = \log_{10}\left(\sum_{i=1}^{N}\frac{EEG\sigma_i^2}{N}\right)$$

2. Relative sigma power

$$A7relSigPow = zscore\left(\log_{10}\left(\frac{PSA_{11-16Hz}}{PSA_{4.5-30Hz}}\right)\right)$$

3. Sigma covariance

$$A7sigmaCov = zscore\left(\log_{10}\left(\frac{1}{N}\sum_{i=1}^{N}\left(EEG_{bf_i} - \mu_{EEG_{bf}}\right)\left(EEG_{\sigma_i} - \mu_{EEG_\sigma}\right)\right)\right)$$

4. Sigma correlation

$$A7sigmaCor = \frac{\mathrm{cov}(EEG_{bf}, EEG_\sigma)}{sd_{EEG_{bf}} * sd_{EEG_\sigma}}$$

Value

Detected spindles and associated features.

References

Lacourse, K., Delfrate, J., Beaudry, J., Peppard, P., & Warby, S. C. (2019). A sleep spindle detection algorithm that emulates human expert spindle scoring. In Journal of Neuroscience Methods (Vol. 316, pp. 3–11). Elsevier BV. https://doi.org/10.1016/j.jneumeth.2018.08.014

Examples

```
tryCatch({
  fpath <- paste0(tempdir(),"c3m2_n2_200hz_uv.csv")

  download.file(
    url = "https://rsleep.org/data/c3m2_n2_200hz_uv.csv",
    destfile = fpath)

  # Read only a sample of the EEG signal
  s = read.csv(fpath,header = FALSE)[,1][25000:45000]

  file.remove(fpath)

  a7_results = a7(s, 200)

  # Plot the first detected spindle
  data = data.frame(x=s,index=seq_along(s))
  a = a7_results$spindles$idxStart[1]
  b = a7_results$spindles$idxEnd[1]
  data = data[(data$index <= (b+600)) & (data$index >= (a-600)), ]
  library(ggplot2)
  ggplot(data, aes(x = index, y = x)) +
    geom_line() +
   geom_line(data = subset(data, index >= a & index <= b), aes(x = index, y = x), color = "red") +
    labs(x = "Signal index", y = "C3-M2") +
    theme_minimal()

  # Visualise features distribution

  hist(a7_results$df$absSigPow,main = "A7absSigPow")

  hist(a7_results$df$relSigPow,main = "A7relSigPow")

  hist(a7_results$df$sigmaCov,main = "A7sigmaCov")

  hist(a7_results$df$sigmaCorr,main = "A7sigmaCorr")
 }, error = function(e) {
  print("Error executing this example, check your internet connection.")
  })
```

adanorm *Adaptive Normalization of a Signal*

Description

This function implements an adaptive normalization method on a given respiratory signal.

Usage

```
adanorm(x, sRate)
```

Arguments

`x`	Numeric vector representing the input signal to be normalized.
`sRate`	Integer value representing the sampling rate of the signal (number of samples per second).

Details

It is designed to preserve the parts of the signal where the amplitude of respiration is small, typically when the body maintains a sleeping posture for extended periods.

Adaptive normalization first segments the signal into 1 second window before computing $A(k)$ and is based on the following equations:

Equation (1) - Mean absolute deviation:

$$A(k) = \frac{1}{fs} \sum_{i=k \cdot fs}^{(k+1) \cdot fs - 1} |x(i)|$$

Equation (2) - Standard deviation:

$$\sigma(k) = \sqrt{\frac{1}{fs - 1} \sum_{i=k \cdot fs}^{(k+1) \cdot fs - 1} (x(i) - \bar{x}(k))^2}$$

Equation (3) - Adaptive normalization factor, initialized to 1.

$$F_{\text{norm}}(k) = \min\left(0.95 F_{\text{norm}}(k-1) + 0.05 A(k), 0.95 F_{\text{norm}}(k-1) + 0.05 \sigma(k)\right)$$

Value

Numeric vector representing the adaptively normalized signal.

References

Choi, S. H., Yoon, H., Kim, H. S., Kim, H. B., Kwon, H. B., Oh, S. M., Lee, Y. J., & Park, K. S. (2018). Real-time apnea-hypopnea event detection during sleep by convolutional neural networks. In Computers in Biology and Medicine (Vol. 100, pp. 123–131). Elsevier BV. https://doi.org/10.1016/j.compbiomed.2018.06

bandpass *Bandpass Filter Function*

Description

This function applies a bandpass filter to a signal. It first normalizes the high and low frequencies based on the Nyquist frequency, then creates a Butterworth filter using the ‘signal::butter‘ function, and finally applies the filter to the signal using ‘signal::filtfilt‘.

Usage

```
bandpass(x, high, low, sRate, order = 5)
```

Arguments

`x`	A numeric vector representing the signal to be filtered.
`high`	The high cutoff frequency for the bandpass filter.
`low`	The low cutoff frequency for the bandpass filter.
`sRate`	The sampling rate of the signal.
`order`	The order of the Butterworth filter, defaulting to 5.

Value

A numeric vector representing the filtered signal.

References

If applicable, add references here.

See Also

`butter`, `filtfilt`

Examples

```
sample_signal <- sin(seq(0, 10, length.out = 1000))
filtered_signal <- bandpass(sample_signal, high = 0.3, low = 0.1, sRate = 100)
```

bands_psd — *Compute power spectral density of bands listed in the bands argument.*

Description

‘bands_psd‘ calculates power spectral densities estimates on bands. Bands are computed from spectrogram bands equal or greater than lower limit and inferior to the upper limit.

Usage

```
bands_psd(signal, sRate, bands, normalize = FALSE, method = "pwelch")
```

Arguments

`signal`	Numerical vector of the signal.
`sRate`	Signal sample rate in Hertz.
`bands`	A list of bands to compute with lower and upper limits in the form ‘list(c(0,4),c(4,8))‘
`normalize`	A band to normalize (divide) by. Defaults to ‘c(0.5,40)‘. Can be set up to FALSE for raw results. Defaults to FALSE.
`method`	Character. Method to use to compute power spectral density. "pwelch" or "psm". Defaults to "pwelch".

Value

A list of bands powers.

Examples

```
signal <- sin(seq(0,100,0.01))
bands_psd(bands = list(c(0,4),c(4,8)), signal = signal, sRate = 200)
```

chambon2018 — *Deep Learning Architecture for Temporal Sleep Stage Classification model implementation in Keras.*

Description

Keras implementation of the deep learning architecture described by Chambon & Al in "A Deep Learning Architecture for Temporal Sleep Stage Classification Using Multivariate and Multimodal Time Series". Consecutives polysomnography (PSG) epochs are supposed to be input to the model to fit on categorized stages as output. ‘write_batches_psg()‘ function writes files batches with the right format for ‘x‘ and ‘y‘ values. The model can then be trained using the ‘train_batches()‘ function. ‘score_psg()‘ uses this model to predict PSG epochs from a raw European Data Format (EDF) record.

Usage

```
chambon2018(channels = 6, samples = 6300)
```

Arguments

`channels`	Integer. Number of channels in each input.
`samples`	Integer. Number of samples in each channel.

Value

A Keras sequential model.

References

Chambon, S., Galtier, M., Arnal, P., Wainrib, G. and Gramfort, A. (2018) A Deep Learning Architecture for Temporal Sleep Stage Classification Using Multivariate and Multimodal Time Series. IEEE Trans. on Neural Systems and Rehabilitation Engineering 26:(758-769).

`check_events` *Check events dataframe format compliance.*

Description

Check events dataframe format compliance.

Usage

```
check_events(events)
```

Arguments

`events`	Events dataframe. Dataframe must contain `begin` (`POSIXt`), `end` (`POSIXt`) and `event` (`character`) columns.

Value

Boolean, according to the events dataframe format compliance.

Examples

```
events <- data.frame(begin = as.POSIXct(c(1536967800,1536967830,1536967860), origin = "1970-01-01"))
events$end <- as.POSIXct(c(1536967830,1536967860,1536967890), origin = "1970-01-01")
events$event = c("N3","N3","REM")
rsleep::check_events(events)
```

choi2018 — *Convolutional neural network for real-time apnea-hypopnea event detection during sleep*

Description

Keras implementation of the deep learning architecture described by Choi & Al in "Real-time apnea-hypopnea event detection during sleep by convolutional neural network".

Usage

```
choi2018(segment_size = 160)
```

Arguments

`segment_size`	Integer. The size of the segment to predict.

Value

A Keras sequential model.

References

Choi SH, Yoon H, Kim HS, et al. Real-time apnea-hypopnea event detection during sleep by convolutional neural networks. Computers in Biology and Medicine. 2018;100:123-131.

ckappa — *Computes Cohen's Kappa for agreement in the case of 2 raters.*

Description

Cohen's kappa coefficient value is a robust statistical measure of inter-rater agreement published in 1960 by Jacob Cohen. It has been reused by numerous studies in sleep medicine to measure the accuracy of predictions, especially for automatic sleep staging.

Usage

```
ckappa(observed, predicted)
```

Arguments

`observed`	The vector of observed values (truth).
`predicted`	The vector of predicted values.

References

Cohen J. A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement. 1960;20:37-46.

Examples

```
observed = c("AWA", "N1", "N2", "N3", "REM")
predicted = c("AWA", "AWA", "N2", "N3", "REM")
ckappa(observed, predicted)
```

clean_oximetry *Clean Oximetry Signal*

Description

This function processes an oximetry signal vector to remove values below a specified threshold. It is designed to enhance the quality of oximetry data by replacing sub-threshold impossible values with the nearest valid data points.

Usage

```
clean_oximetry(oximetry, threshold = 70)
```

Arguments

oximetry	A numeric vector representing the oximetry signal. Each element corresponds to an oximetry reading.
threshold	A numeric value setting the minimum acceptable oximetry value. Default is 70. Values in ‘oximetry‘ below this threshold will be replaced with the nearest value above the threshold or an average of the nearest valid values on either side.

Details

The function iterates through the ‘oximetry‘ vector. For each value below the ‘threshold‘, it searches for the nearest valid value (above the threshold) to the left and right. If both neighbors are found, it replaces the sub-threshold value with their average. If only one valid neighbor is found, it uses that value.

The algorithm ensures that the processed signal retains the general pattern of the original data while mitigating the impact of anomalously low readings.

Value

A numeric vector of the same length as ‘oximetry‘. Sub-threshold values are replaced based on nearby valid readings.

Examples

```
oximetry_data <- c(91, 92, 91, 34, 92, 93, 91)
clean_oximetry(oximetry_data)
```

`create_xts` *Create an XTS Object from Resampled Signals*

Description

This function takes multiple signals and their corresponding sample rates, resamples the signals to the highest sample rate among them, and creates an xts (eXtensible Time Series) object with the resampled signals aligned according to a provided start time.

Usage

```
create_xts(signals, sample_rates, start_time)
```

Arguments

`signals`	A list of numeric vectors representing the signals. Each signal in the list should correspond to one sample rate in the 'sample_rates' argument.
`sample_rates`	A numeric vector containing the sample rates for each signal in 'signals'. The length of 'sample_rates' must match the length of 'signals'.
`start_time`	The start time for the xts object. This can be a character string or an object that can be converted to POSIXct. The time is assumed to be in UTC.

Value

An xts object containing the resampled signals, with each column representing one of the original signals, resampled to the highest sample rate among them. The xts object's index starts from 'start_time' and increments at a rate of 1 divided by the maximum sample rate.

Examples

```
signals <- list(rnorm(100), rnorm(100))
sample_rates <- c(1, 2)
start_time <- "2020-01-01 00:00:00"
xts_data <- create_xts(signals, sample_rates, start_time)
plot(xts_data)
```

`detect_apneic_events` *Detect Apneic Events in SpO2 Signal Data*

Description

This function implements the algorithm described by Jund & Al in "Real-Time Automatic Apneic Event Detection Using Nocturnal Pulse Oximetry", 2018. It analyzes a given SpO2 signal to detect apneic events. It works by resampling the input signal and applying a series of checks to identify potential apnea instances. The algorithm uses a state machine with different blocks representing various stages of detection.

Usage

```
detect_apneic_events(spo2, sRate)
```

Arguments

`spo2`	A numeric vector representing the SpO2 signal data.
`sRate`	The original sampling rate of the SpO2 signal.

Value

A list of numeric vectors. Each vector represents a detected apneic event, containing the start and end indices of the event in the resampled signal.

References

Jung, D. W., Hwang, S. H., Cho, J. G., Choi, B. H., Baek, H. J., Lee, Y. J., Jeong, D.-U., & Park, K. S. (2018). Real-Time Automatic Apneic Event Detection Using Nocturnal Pulse Oximetry. In IEEE Transactions on Biomedical Engineering (Vol. 65, Issue 3, pp. 706–712). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/tbme.2017.2715405

Examples

```
# Example usage
spo2_sample <- c(98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88)
sample_rate <- 1  # Assuming 1 Hz sampling rate
detected_apneas <- detect_apneic_events(spo2_sample, sample_rate)
print(detected_apneas)
```

`detect_rem` *Detection of Rapid-Eye Movements (REMs)*

Description

Implements the algorithm detailed in Agarwal &Al. "Detection of Rapid-Eye Movements in Sleep Studies." This function processes electrooculography (EOG) signals to detect rapid-eye movements (REMs) characteristic of REM sleep, applying filters, artifact detection, and angle-based inclusion criteria.

Usage

```
detect_rem(
  roc,
  loc,
  sRate,
  l = 0.5,
  art = 500,
  nip = 120,
  cc = -0.2,
  da = 45,
  dadiff = 15
)
```

Arguments

`roc`	Right outer canthus EOG signal vector.
`loc`	Left outer canthus EOG signal vector.
`sRate`	Sampling rate of the EOG signals in Hz.
`l`	Window length in seconds for REM detection (default is 0.5).
`art`	Artifact threshold, max amplitude allowed in EOG signals to consider the data valid (default is 500).
`nip`	Negative Inflexion Point threshold for REM detection (default is 120).
`cc`	Correlation coefficient threshold for inclusion of a REM event. Negative correlation indicates potential REM (default is -0.2).
`da`	Desired angle for REM detection (default is 45 degrees).
`dadiff`	Acceptable deviation from the desired angle for one eye, if the other eye compensates with a larger deviation (default is 15 degrees).

Details

The function processes the EOG signals by applying a band-pass Butterworth filter to isolate frequencies between 1 and 5 Hz, typical for REMs. It then computes the artifact measure and evaluates the signal for REM events based on the slope of the EOG signal segments, correlation between left and right signals, and other criteria derived from the REM detection algorithm described by Agarwal et al. The function returns a list containing filtered EOG signals, artifact measures, and detected REM events with their characteristics and validity based on the algorithm's criteria.

Value

A list with the following elements: - ‘rocf‘: Filtered right outer canthus EOG signal. - ‘locf‘: Filtered left outer canthus EOG signal. - ‘block_art‘: Maximum absolute amplitude in the EOG channels for the detection block, used for artifact measurement. - ‘cpm‘: Product of inverted left and right EOG signals, part of REM detection criteria. - ‘cn‘: Conditioned signal based on ‘cpm‘, with values below a threshold set to 0. - ‘crems‘: Data frame of candidate REMs with indices, characteristics, and validity flag. - ‘rems‘: Subset of ‘crems‘ containing only the valid REM events.

References

Agarwal, R., Takeuchi, T., Laroche, S., & Gotman, J. (2005). Detection of Rapid-Eye Movements in Sleep Studies. IEEE Transactions on Biomedical Engineering, 52(8), 1390–1396. https://doi.org/10.1109/TBME.2005.85151

`detect_rpeaks` *Detect R peaks in a raw ECG signal.*

Description

‘detect_rpeaks‘ implements the first part of the Pan & Tompkins algorithms to detect R peaks from a raw ECG signal.

Usage

```
detect_rpeaks(
  signal,
  sRate,
  lowcut = 0,
  highcut = 15,
  filter_order = 1,
  integration_window = 15,
  refractory = 200,
  return_index = FALSE
)
```

Arguments

`signal`	Numerical vector of ECG signal.
`sRate`	ECG signal sample rate.
`lowcut`	Butterworth bandpass filter low cut value.
`highcut`	Butterworth bandpass filter high cut value.
`filter_order`	Butterworth bandpass filter order value.

`integration_window`	Convolution window size.
`refractory`	Minimal space between peaks in milliseconds.
`return_index`	If TRUE, the index for each R peak is returned instead of the timing.

Value

A numeric vector of detected R peaks, expressed in seconds* from the start of the signal. This vector can be used in RHRV using ‘RHRV::LoadBeatVector()‘.

*(or samples if return_index is TRUE)

References

Pan, Jiapu, and Willis J. Tompkins. "A real-time QRS detection algorithm." IEEE Trans. Biomed. Eng 32, no. 3 (1985): 230-236.

Examples

```
tryCatch({
  path <- paste0(tempdir(),"rec_1.dat")
  download.file("https://rsleep.org/data/rec_1.sdat",path)
  ecg <- readBin(path,integer(),500*30)
  peaks <- detect_rpeaks(ecg, sRate = 500)
  unlink(path)
  print(peaks)
  ecg.df <- data.frame(ECG = ecg,Seconds = c(1:length(ecg))/500)
  library(ggplot2)
  ggplot(ecg.df,aes(x = Seconds,y = ECG)) +
    geom_line() +
    theme_bw() +
    geom_vline(
      data.frame(p = peaks),
      mapping = aes(xintercept = p),
      linetype="dashed",
      color = "red")
}, error = function(e) {
  print("Error executing this example, check your internet connection.")
  })
```

`epochs` *Split signals into consecutive, non-overlaping epochs according to an events dataframe or an epoch duration.*

Description

Split long signals into a list of consecutive epochs according to an events dataframe or an epoch duration.

Usage

```
epochs(
  signals,
  sRates,
  resample = max(sRates),
  epoch = 30,
  startTime = 0,
  padding = 0
)
```

Arguments

`signals`	A list of numeric vectors containing signals, or a single vector containing one signal.
`sRates`	A vector or list of integer values of the signals sample rates.
`resample`	The sample rate to resample all signals. Defaults to to the max of the provided sample rates.
`epoch`	Epochs reference. Can be an events dataframe or the number of seconds of each epoc Defaults to 30.
`startTime`	The start timestamp of the signal, used to join events to epoch.
`padding`	Number of previous and next epochs to pad the current epoch with. This functionnality is mostly used to enrich deep learning datasets. Defaults to 0.

Value

A list of signal chunks

Examples

```
epochs(list(rep(c(1,2,3,4),100),rep(c(5,6,7,8),100)),4,4,1,padding = 2)
```

`hypnogram` *Filter and reorder an events dataframe or a hypnodensity to keep only sleep stages related-events.*

Description

Remove non-sleep stages events and reorder dataframe rows using the `begin` column.

Usage

```
hypnogram(
  events,
  labels = c("N3", "N2", "N1", "REM", "AWA"),
  startTime = 946681200,
  epoch_duration = 30,
  plot = FALSE
)
```

Arguments

`events`	Events dataframe. Dataframe must have `begin` (`POSIXt`), `end` (`POSIXt`) and `event`
`labels`	Sleep stages labels. Defaults to `c("N3","N2","N1","REM","AWA")`.
`startTime`	Hypnogram start time. Used when a hypnodensity dataframe is passed as events. Defaults to 946681200.
`epoch_duration`	Epoch duration in seconds. Used when a hypnodensity dataframe is passed as events. Defaults to 30.
`plot`	Plot the hypnogram or in not using `ggplot2`.

Value

Hypnogram dataframe or plot.

Examples

```
tryCatch({
  fpath <- paste0(tempdir(),"/15012016HD.csv")

  download.file("https://rsleep.org/data/15012016HD.csv",fpath, method="curl")

  events <- read_events_noxturnal(fpath)

  unlink(fpath)

  hypnogram(events)
 }, error = function(e) {
  print("Error executing this example, check your internet connection.")
  })
```

`normalize_cycles` *Normalize sleep cycles scored on Noxturnal software from start and stop flags to unique events.*

Description

Normalize sleep cycles scored on Noxturnal software from start and stop flags to unique events.

Usage

```
normalize_cycles(events)
```

Arguments

`events`	Events dataframe. Dataframe must have `begin` (`POSIXt`), `end` (`POSIXt`) and `event`. Cycles flags must be named `Activity-CLASSICstart`, `Activity-BNstart`, `Activity-BNend`, `Activity-REMstart`, `Activity-REMend`, `Activity-ENstart` or `Activity-ENend`.

Examples

```
cycles <- data.frame(event = c("Activity-CLASSICstart","Activity-CLASSICend"))
cycles$begin <- as.POSIXct(c("2016-01-16 01:13:30","2016-01-16 01:15:30"))
cycles$end <- as.POSIXct(c("2016-01-16 01:13:30","2016-01-16 01:15:30"))
normalize_cycles(cycles)
```

`periods` *Get a dataframe of sleep periods from a hypnogram, continuous or by stages.*

Description

Get a dataframe of sleep periods from a hypnogram, continuous or by stages.

Usage

```
periods(
  hypnogram,
  mode = "continuous",
  stages = c("N1", "N2", "N3", "N4", "REM")
)
```

Arguments

`hypnogram`	A hypnogram dataframe. Dataframe must contain `begin` (`POSIXt`), `end` (`POSIXt`) and `event` (`character`) columns.
`mode`	Period mode. `"continuous"` computes periods of N1, N2, N3 or REM sleep, regardless of stage. `"stages"` computes periods of sleep by stage.
`stages`	Stages to include in periods. Defaults to ‘c("N1", "N2", "N3", "N4", "REM")‘.

Value

A dataframe of periods with their begin and stop times, duration and stages for stage mode.

Examples

```
tryCatch({
library(ggplot2)

download.file(
 "https://rsleep.org/data/hypnodensity.csv",
 "hypnodensity.csv")

hypnodensity <- read.csv2("hypnodensity.csv")

unlink("hypnodensity.csv")

events <- hypnogram(hypnodensity)
periods_continuous <- periods(events, mode = "continuous")

ggplot(periods_continuous, aes(x=duration)) + geom_histogram(bins = 30)

periods_stages <- periods(events, mode = "stages")

ggplot(periods_stages, aes(x=event,y=duration,color=event)) + geom_boxplot()
}, error = function(e) {
  print("Error executing this example, check your internet connection.")
  })
```

`plot_event` *Highlight a scored event over a signal.*

Description

Highlight a scored event over a signal.

Usage

```
plot_event(signal, sRate, sig_start, event_start, event_end, window = 10)
```

Arguments

`signal`	The signal vector.
`sRate`	Sample rate of the signal.
`sig_start`	Date-Time value of the signal start.
`event_start`	Date-Time value of the event start.
`event_end`	Date-Time value of the event end.
`window`	Number of seconds of signal to plot before, and after.

Value

A plot of the highlighted event over the signal.

`plot_hypnodensity` *Plot a hypnodensity graph.*

Description

Plot a hypnodensity graph using ‘ggplot2‘. Hypnodensity can be read from file or returned by the ‘score_stages_edf‘ function.

Usage

```
plot_hypnodensity(
  hypnodensity,
  stages = c("AWA", "REM", "N1", "N2", "N3"),
  colors = c("#5BBCD6", "#FF0000", "#00A08A", "#F2AD00", "#F98400")
)
```

Arguments

`hypnodensity`	A hypnodensity dataframe as returned by the ‘score_stages_edf‘ function.
`stages`	Vector of stages labels to plot.
`colors`	Vector of colors to use.

Value

A ‘ggplot2‘ hypnodensity graph.

References

Stephansen, J.B., Olesen, A.N., Olsen, M., Ambati, A., Leary, E.B., Moore, H.E., Carrillo, O., Lin, L., Han, F., Yan, H. and Sun, Y.L., 2018. Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy. Nature communications, 9(1), p.5229.

Examples

```
tryCatch({
download.file("https://rsleep.org/data/hypnodensity.csv", "hypnodensity.csv")

hypnodensity <- read.csv2("hypnodensity.csv")

unlink("hypnodensity.csv")

plot_hypnodensity(hypnodensity)
}, error = function(e) {
  print("Error executing this example, check your internet connection.")
  })
```

plot_hypnogram *Plot a hypnogram from an events dataframe.*

Description

Plot a hypnogram from an events dataframe.

Usage

```
plot_hypnogram(events, labels = c("N3", "N2", "N1", "REM", "AWA"))
```

Arguments

events	Events dataframe. Dataframe must have `begin` (`POSIXt`), `end` (`POSIXt`) and `event`
labels	Sleep stages labels. Defaults to `c("N3","N2","N1","REM","AWA")`.

Value

a ggplot object.

Examples

```
hypnogram <- data.frame(begin = as.POSIXlt(
c(1536967800,1536967830,1536967860),origin = "1970-01-01"))
hypnogram$end <- as.POSIXlt(c(1536967830,1536967860,1536967890),
origin = "1970-01-01")
hypnogram$event = c("N3","N3","REM")
plot_hypnogram(hypnogram)
```

psm *Power spectral density using adaptive sine multitaper.*

Description

Power spectral density using adaptive sine multitaper.

Usage

```
psm(x, sRate, length = 0, show = TRUE)
```

Arguments

`x`	Signal vector.
`sRate`	Sample rate of the signal.
`length`	periodogram resolution. 0 default to not resize.
`show`	todo

Value

peridodogram plotted or raw.

References

Barbour, A. J. and R. L. Parker (2014), psd: Adaptive, sine multitaper power spectral density estimation for R, Computers & Geosciences, Volume 63, February 2014, Pages 1-8, ISSN 0098-3004, http://dx.doi.org/10.1016/j.cageo.2013.09.015

Examples

```
x <- sin(c(1:10000))
psd <- psm(x, 200, 100)
head(psd)
```

`pwelch` *Power spectral density using Welch's method.*

Description

Power spectral density using Welch's method.

Usage

```
pwelch(x, sRate, points = 0, overlap = 0, padding = 0, show = TRUE)
```

Arguments

`x`	Signal vector.
`sRate`	Sample rate of the signal.
`points`	todo
`overlap`	todo
`padding`	todo
`show`	todo

Value

peridodogram plotted or raw

References

Welch, P. “The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short, Modified Periodograms.” IEEE Transactions on Audio and Electroacoustics 15, no. 2 (June 1967): 70–73. https://doi.org/10.1109/TAU.1967.1161901.

Examples

```
x <- sin(c(1:10000))
psd <- pwelch(sin(c(1:10000)), 200)
head(psd)
```

`read_events_compumedics` *Read a stages export from Compumedics software in .txt format.*

Description

Read a stages export from Compumedics software in .txt format.

Usage

```
read_events_compumedics(
  txt,
  startTime = as.POSIXlt("2000-01-01"),
  labels = c(AWA = 0, N1 = 1, N2 = 2, N3 = 3, REM = 5)
)
```

Arguments

`txt`	txt file path.
`startTime`	Character string or date object of the hypnogram start.
`labels`	Labels and values as a named list. Defaults to c("AWA" = 0, "N1" = 1, "N2" = 2, "N3" = 3, "REM" = 5).

Value

A dataframe of stages.

`read_events_ndb` *Read events from a Resmed Noxturnal .ndb file.*

Description

Read events from a Resmed Noxturnal .ndb file.

Usage

```
read_events_ndb(data_file)
```

Arguments

`data_file`	.ndb file path.

Value

An events dataframe.

`read_events_noxturnal` *Read a Noxturnal events file (Unicode CSV format)*

Description

Read a Noxturnal events file (Unicode CSV format)

Usage

```
read_events_noxturnal(dir)
```

Arguments

`dir`	Noxturnal events file path.

Value

A dataframe of scored events.

`read_events_profusion` *Read a annotation file from Compumedics Profusion software in XML format.*

Description

Read a annotation file from Compumedics Profusion software in XML format.

Usage

```
read_events_profusion(xml, startTime = as.POSIXlt("1970-01-01 00:00:00"))
```

Arguments

`xml`	XML file path.
`startTime`	Character string or date object of the hypnogram start.

Value

A dataframe of stages and events.

`read_events_sleepedfx` *Read a SleepEDFX events file EDF+*

Description

Read a SleepEDFX events file EDF+

Usage

```
read_events_sleepedfx(dir, update = TRUE)
```

Arguments

`dir`	EDF+ path
`update`	merge N3 and N4 or not

Value

A dataframe of scored events.

`read_mdf` *Read a Morpheo Data Format (MDF) directory to a list.*

Description

Read a Morpheo Data Format (MDF) directory to a list.

Usage

```
read_mdf(mdfPath, channels = c(NA), metadata = TRUE)
```

Arguments

`mdfPath`	character. MDF path.
`channels`	character. Channels to read.
`metadata`	boolean. Read or not the metadata.

Value

A list.

References

P. Bouchequet, D. Jin, G. Solelhac, M. Chennaoui, D. Leger, "Morpheo Data Format (MDF), un nouveau format de données simple, robuste et performant pour stocker et analyser les enregistrements de sommeil", Médecine du Sommeil, vol. 15, n 1, p. 48/49, march 2018.

`schwabedal2018` *Automated Classification of Sleep Stages in Mice with Deep Learning model implementation in Keras.*

Description

Model inspired by the article "Automated Classification of Sleep Stages and EEG Artifacts in Mice with Deep Learning". Implemented using Keras. Adapted to use minimum 2 channels and to not score artifact epochs.

Usage

```
schwabedal2018(channels = 2, samples = 8000)
```

Arguments

`channels`	Number of channels in each input.
`samples`	Number of samples in each channel.

Value

A Keras sequential model.

References

Schwabedal, Justus T. C., Daniel Sippel, Moritz D. Brandt, and Stephan Bialonski. "Automated Classification of Sleep Stages and EEG Artifacts in Mice with Deep Learning." ArXiv:1809.08443 [Cs, q-Bio], September 22, 2018. http://arxiv.org/abs/1809.08443.

`score_mice` *Score mice sleep from European Data Format (EDF) files.*

Description

Score mice sleep from European Data Format (EDF) files.

Usage

```
score_mice(edf, model, verbose = TRUE)
```

Arguments

`edf`	Character. European Data Format (EDF) file path.
`model`	model
`verbose`	Boolean. Display or not status messages.

Value

A dataframe containing predicted hypnodensity values of the record.

`score_psg` *Score 30 seconds epochs from European Data Format (EDF) files.*

Description

Score 30 seconds epochs from European Data Format (EDF) files.

Usage

```
score_psg(
  edf,
  channels = c("C3-M2", "C4-M1", "O1-M2", "E1-M2", "E2-M1", "1-2"),
  model = chambon2018(6, 3 * 30 * 70),
  verbose = TRUE
)
```

Arguments

edf	Character. European Data Format (EDF) file path.
channels	A vector containing the channels names if names differ from ‘c("C3-M2","C4-M1","O1-M2","E1-M2","E2-M1","1-2")‘.
model	The Keras model.
verbose	Boolean. Display or not status messages.

Value

A dataframe containing predicted hypnodensity values of the record.

References

Chambon, S., Galtier, M., Arnal, P., Wainrib, G. and Gramfort, A. (2018) A Deep Learning Architecture for Temporal Sleep Stage Classification Using Multivariate and Multimodal Time Series. IEEE Trans. on Neural Systems and Rehabilitation Engineering 26:(758-769).

Kemp, B., Värri, A., Rosa, A.C., Nielsen, K.D. and Gade, J., 1992. A simple format for exchange of digitized polygraphic recordings. Electroencephalography and clinical neurophysiology, 82(5), pp.391-393.

segmentation *Split signals into consecutive, overlapping segments.*

Description

Split signals into consecutive, overlapping segments.

Usage

```
segmentation(
  signals,
  sRates,
  segments_size = 10,
  step = 1,
  padding = 0,
  resample = max(sRates),
  return_index = FALSE
)
```

Arguments

signals	A list of numeric vectors containing signals, or a single vector containing one signal.
sRates	A vector or list of integer values of the signals sample rates.
segments_size	The size of segments, in seconds.

`step`	The step between segments, in seconds.
`padding`	umber of previous and next epochs to pad the current epoch with. Defaults to 0.
`resample`	The sample rate to resample all signals. Defaults to to the max of the provided sample rates.
`return_index`	If TRUE, the index of segments is returned instead of the segments.

Value

A matrix of segments.

References

Choi SH, Yoon H, Kim HS, et al. Real-time apnea-hypopnea event detection during sleep by convolutional neural networks. Computers in Biology and Medicine. 2018;100:123-131.

Examples

```
computed_segments = segmentation(
  signals = list(c(sin(1:1000)),c(cos(1:1000))),
  sRates = c(1, 1),
  segments_size = 5,
  resample = 1)
dim(computed_segments)
plot(computed_segments[1,,1], type = "l")
plot(computed_segments[2,,1], type = "l")
```

`smooth_hypnogram` *Smooth hypnogram epoch, simulating human scorers behaviour.*

Description

Smooth hypnograms epoch, simulating human scorers behaviour.

Usage

```
smooth_hypnogram(hypnogram, event = "N2", neighbors = "REM", count = 2)
```

Arguments

`hypnogram`	A hypnogram dataframe.
`event`	Central stage label.
`neighbors`	Extremities stages labels.
`count`	Number of consecutive central stages.

Value

A hypnogram dataframe.

References

Liang, Sheng-Fu, Chin-En Kuo, Yu-Han Hu, Yu-Hsiang Pan, and Yung-Hung Wang. "Automatic stage scoring of single-channel sleep EEG by using multiscale entropy and autoregressive models." IEEE Transactions on Instrumentation and Measurement 61, no. 6 (2012): 1649-1657.

Examples

```
hypnogram <- data.frame(begin = as.POSIXlt(
c(1536967800,1536967830,1536967860),origin = "1970-01-01"))
hypnogram$end <- as.POSIXlt(c(1536967830,1536967860,1536967890),
origin = "1970-01-01")
hypnogram$event = c("REM","N2","REM")
smooth_hypnogram(hypnogram, "N2","REM",1)
```

`smooth_liang2012` *Smooth hypnogram according to the 11 rules described by Liang & Al.*

Description

Smooth hypnogram according to the 11 rules described by Liang & Al.

Usage

```
smooth_liang2012(hypnogram)
```

Arguments

`hypnogram`	A hypnogram dataframe.

Value

A smoothed hypnogram dataframe.

References

Liang, Sheng-Fu, Chin-En Kuo, Yu-Han Hu, and Yu-Shian Cheng. “A Rule-Based Automatic Sleep Staging Method.” Journal of Neuroscience Methods 205, no. 1 (March 2012): 169–76. https://doi.org/10.1016/j.jneumeth.2011.12.022.

spectrogram *Plot the spectrogram of signal.*

Description

'spectrogram' resamples signal and use the 'specgram' function from the 'signal' library to compute the spectrogram. Results resolution can be then reduced to quickly plot large signals.

Usage

```
spectrogram(
  signal,
  sRate,
  maxFreq = 25,
  n = 1024,
  window = n,
  overlap = 0,
 cols = c(rep("#3B9AB2", 9), "#78B7C5", "#EBCC2A", "#E1AF00", rep("#F21A00", 6)),
  freq = 4,
  plot = TRUE,
  startTime = as.POSIXct("1970/01/01 00:00:00")
)
```

Arguments

`signal`	Numerical vector of the signal.
`sRate`	Signal sample rate in Hertz.
`maxFreq`	Maximal frequency to plot in Hertz. Signal will be resampled at maxFreq*2 sample rate.
`n`	The size of the Fourier transform window.
`window`	Shape of the fourier transform window, defaults to n.
`overlap`	Overlap with previous window, defaults to 0.
`cols`	Color scale used for the underlying plot function.
`freq`	Aggregate frequency used to lower spectrogram resolution. Defaults to 4.
`plot`	Boolean, plot or not the spectrogram.
`startTime`	Posixct of the signal start. Adjust the x axis labels accordingly.

Value

A spectrogram.

Examples

```
library(signal)
spectrogram(chirp(seq(-2, 15, by = 0.001), 400, 10, 100, 'quadratic'),20,n=1024)
```

`stages_stats` *Get stages related statistics in a named vector.*

Description

`stages_stats` computes stages related statistics.

Usage

```
stages_stats(e)
```

Arguments

`e`	Events dataframe. Dataframe must have `begin` (`POSIXt`), `end` (`POSIXt`) and `event` (`character`) columns.

Value

stages vector

Examples

```
e <- data.frame(begin = as.POSIXlt(seq(from = 0, to = 30*10, by = 30),origin = "1970-01-01"))
e$end <- as.POSIXlt(seq(from = 30, to = 30*11, by = 30), origin = "1970-01-01")
e$event = c("AWA","N1","N2","N3","N3","REM","N2","REM","N2","REM","AWA")
stages_stats(e)
```

`train_batches` *Trains a model from files batches.*

Description

Trains a model from files batches.

Usage

```
train_batches(model, batches, epochs = 10)
```

Arguments

`model`	Keras model.
`batches`	Character vector of batches files.
`epochs`	Integer. Number of epochs to train the model.

Value

A trained and serialized Keras model.

`transitions` *Count and format stages transitions.*

Description

Count and format stages transitions.

Usage

```
transitions(
  hypnogram,
  stages = c("AWA", "REM", "N1", "N2", "N3", "NREM"),
  format = "vector"
)
```

Arguments

`hypnogram`	A hypnogram dataframe. Dataframe must contain `begin` (`POSIXt`), `end` (`POSIXt`) and `event` (`character`) columns.
`stages`	Stages to include in transitions Defaults to `c("N1", "N2", "N3", "N4", "REM")`.
`format`	Set the return format. 'vector', 'dataframe' or 'heatmap'.

Value

Count of stages transitions in selected format.

References

Swihart BJ, Punjabi NM, Crainiceanu CM. Modeling sleep fragmentation in sleep hypnograms: An instance of fast, scalable discrete-state, discrete-time analyses. Comput Stat Data Anal. 2015 Sep;89:1-11. doi: 10.1016/j.csda.2015.03.001. PMID: 27182097; PMCID: PMC4865264.

Examples

```
tryCatch({
download.file("https://rsleep.org/data/hypnodensity.csv", "hypnodensity.csv")

hypnodensity <- read.csv2("hypnodensity.csv")

unlink("hypnodensity.csv")

events <- hypnogram(hypnodensity)

transitions(events)

transitions(events, format = "dataframe")

transitions(events, format = "heatmap")
# 3 Dimensions sleep transitions
levels(events$event)[levels(events$event)=="N1"] <- "NREM"
levels(events$event)[levels(events$event)=="N2"] <- "NREM"
levels(events$event)[levels(events$event)=="N3"] <- "NREM"

round(
  transitions(
    events,
    format = "dataframe")/(
      sum(transitions(events)))*100,2)
}, error = function(e) {
  print("Error executing this example, check your internet connection.")
  })
```

tst90 — *Compute TST90, the percentage of time during sleep with an oxygen saturation below 90.*

Description

Compute TST90, the percentage of time during sleep with an oxygen saturation below 90.

Usage

```
tst90(spo2_signal, sRate, startTime, hypnogram)
```

Arguments

`spo2_signal`	The SpO2 signal vector.
`sRate`	The SpO2 signal vector sample rate.
`startTime`	The SpO2 signal start time.
`hypnogram`	Events dataframe containing hypnogram.

write_batches_mice — *Write batches from mice records*

Description

Write batches from mice records

Usage

```
write_batches_mice(
  records,
  events,
  batches_path = "./",
  batch_size = 128,
  classes_nb = 3,
  padding = 2,
  resample = 400,
  verbose = TRUE
)
```

Arguments

`records`	Character. Records paths
`events`	List of events
`batches_path`	Path to write batches
`batch_size`	size of each batch
`classes_nb`	number of classes
`padding`	consecutive epochs to add
`resample`	resample rate
`verbose`	Boolean. Display or not status messages.

`write_batches_psg` *Generates files batches from PSG data.*

Description

Generates train batches from PSG data to be used by the 'train_batches()' function.

Usage

```
write_batches_psg(
  records,
  events,
  batches_path = tempdir(),
  channels = c("C3-M2", "C4-M1", "O1-M2", "E1-M2", "E2-M1", "1-2"),
  resample = 70,
  padding = 1,
  batches_size = 1024,
  verbose = TRUE
)
```

Arguments

`records`	Character vector of EDF files paths to be included in the train batches.
`events`	List of events dataframes containing hypnograms corresponding to EDF records in ‘records‘ parameter.
`batches_path`	Character. Path where batches files will be saved.
`channels`	Character vector. Channels labels to include in the dataset.
`resample`	Integer. Sample rate to resample selected signals.
`padding`	Epochs added before and after each epoch.
`batches_size`	Number of epochs in each batch file.
`verbose`	Boolean, display status messages or not.

References

Chambon, S., Galtier, M., Arnal, P., Wainrib, G. and Gramfort, A. (2018) A Deep Learning Architecture for Temporal Sleep Stage Classification Using Multivariate and Multimodal Time Series. IEEE Trans. on Neural Systems and Rehabilitation Engineering 26:(758-769).

`write_channel` *Write a timeserie to disk using Morpheo Data Format (MDF) guidelines.*

Description

Write a timeserie to disk using Morpheo Data Format (MDF) guidelines.

Usage

```
write_channel(channel, signals, headers, mdfPath, endian = .Platform$endian)
```

Arguments

`channel`	character. Channel name.
`signals`	list. European Data Format (EDF) signals list.
`headers`	list. European Data Format (EDF) file headers.
`mdfPath`	character. Morpheo Data Format (MDF) directory path.
`endian`	character. Endianess. `"big"` or `"little"`. Defaults to platform endian.

References

P. Bouchequet, D. Jin, G. Solelhac, M. Chennaoui, D. Leger, "Morpheo Data Format (MDF), un nouveau format de données simple, robuste et performant pour stocker et analyser les enregistrements de sommeil", Médecine du Sommeil, vol. 15, n 1, p. 48-49, march 2018.

`write_hypnogram_compumedics` *Write a XML file containing scored stages for Compumedics software.*

Description

Write a XML file containing scored stages for Compumedics software.

Usage

```
write_hypnogram_compumedics(hypnogram, filename)
```

Arguments

`hypnogram`	A rsleep hypnogram dataframe.
`filename`	character File name to write on disk.

`write_mdf` *Write a European Data Format (EDF) record file to disk using Morpheo Data Format (MDF) guidelines*

Description

Write a European Data Format (EDF) record file to disk using Morpheo Data Format (MDF) guidelines. Target directory is erased if it already exists. Signals are stored in binary file, events and metadata in JavaScript Object Notation (JSON) files.

Usage

```
write_mdf(
  edfPath,
  mdfPath,
  channels = c(NA),
  events = c(),
  endian = .Platform$endian
)
```

Arguments

`edfPath`	character. European Data Format (EDF) file path.
`mdfPath`	character. Morpheo Data Format (MDF) directory path.
`channels`	character. Vector of channels labels to write.
`events`	dataframe. Events dataframe to write. Events dataframe. Dataframe must contain `begin` (`POSIXt`), `end` (`POSIXt`) and `event` (`character`) columns.
`endian`	character. Endianess. `"big"` or `"little"`. Defaults to platform endian.

References

P. Bouchequet, D. Jin, G. Solelhac, M. Chennaoui, D. Leger, "Morpheo Data Format (MDF), un nouveau format de données simple, robuste et performant pour stocker et analyser les enregistrements de sommeil", Médecine du Sommeil, vol. 15, n 1, p. 48/49, march 2018.

Index