Package ‘rsleep’

December 22, 2022

Type Package

Title Analysis of Sleep Data

Version 1.0.6

Description Provides users functions for sleep data management and analysis such as European Data Format (EDF) to Morpheo Data Format (MDF) conversion: P.Bouchequet, D.Jin, G.Solelhac, M.Chennaoui, D.Leger (2018) <doi:10.1016/j.msom.2018.01.130> ``Morpheo Data Format (MDF), un nouveau format de donnees simple, robuste et performant pour stocker et analyser les enregistrements de sommeil''. Provides hypnogram statistics computing and visualisation functions from the American Academy of Sleep Medicine (AASM) manual ``The AASM Manual for the Scoring of Sleep and Associated Events'' <`https://aasm.org/clinical-resources/scoring-manual/`>.

License MIT + file LICENSE

Encoding UTF-8

URL `https://rsleep.org/`

Imports abind, edfReader, ggplot2, jsonlite, psd, signal, xml2

Suggests caret, gridExtra, keras, knitr, reshape2, rmarkdown, testthat

RoxygenNote 7.2.2

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 2022-12-22 19:50:02 UTC

R topics documented:

`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)
```

`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

```
path <- paste0(tempdir(),"rec_1.dat")
download.file("https://physionet.org/files/ecgiddb/1.0.0/Person_01/rec_1.dat?download",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")
```

`epochs` *Split signals into 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)
```

`get_sleep_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

```
get_sleep_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

```
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 <- get_sleep_periods(events, mode = "continuous")

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

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

ggplot(periods_stages, aes(x=event,y=duration,color=event)) + geom_boxplot()
```

`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

```
fpath <- paste0(tempdir(),"SC4001EC-Hypnogram.edf")

furl <- paste0("https://www.physionet.org/files/sleep-edfx/1.0.0/",
 "sleep-cassette/SC4001EC-Hypnogram.edf?download")

download.file(furl,fpath)

events <- read_events_sleepedfx(fpath)

unlink(fpath)

hypnogram(events, plot = TRUE)
```

`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)
```

`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"))
```

Arguments

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

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

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

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

unlink("hypnodensity.csv")

plot_hypnodensity(hypnodensity)
```

`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)

fpath <- paste0(tempdir(),"SC4001EC-Hypnogram.edf")
furl <- paste0("https://www.physionet.org/files/sleep-edfx/1.0.0/",
"sleep-cassette/SC4001EC-Hypnogram.edf?download")
```

```
download.file(furl,fpath)
hypnogram <- read_events_sleepedfx(fpath)
unlink(fpath)
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)
```

Arguments

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

Value

A dataframe of stages.

`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.

`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 events related stats in a named vector.*

Description

`stages_stats` computes stages related stats.

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` *Compute a matrix of stages transitions counts.*

Description

Compute a matrix of stages transitions counts.

Usage

```
transitions(
  hypnogram,
  stages = c("AWA", "REM", "N1", "N2", "N3"),
  plot = FALSE
)
```

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")`.
`plot`	Plot the transition matrix or not. Takes `"heatmap"` to plot a heatmap of transitions or `FALSE` to return the matrix as a dataframe without plotting it.

Value

A matrix with count of stages transitions, or a plot.

Examples

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

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

unlink("hypnodensity.csv")

events <- hypnogram(hypnodensity)

transitions(events, plot = "heatmap")
```

`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