SleepPy

Release 1.0.0

Yiorgos Christakis

Dec 18, 2019

CONTENTS:

CHAPTER

ONE

SLEEPPY PACKAGE

1.1 Submodules

1.2 sleeppy.colekripke module

class sleeppy.colekripke.**ColeKripke**(*activity_index*, *sf=0.243*)

Bases: object

Runs sleep wake detection on epoch level activity data. Epochs are 1 minute long and activity is represented by an activity index.

Methods

apply_kernel(self)	Runs the prediction of sleep wake states based on activity index data.
predict(self)	Runs full algorithm:
rescore(self)	Application of Webster's rescoring rules as described in the Cole-Kripke paper.

apply_kernel(*self*)

Runs the prediction of sleep wake states based on activity index data.

Returns

predictions [array-like] Sleep/wake predictions from a Cole-Kripke based algorithm.

predict(*self*)

Runs full algorithm: Predict sleep/wake states using Cole-Kripke's optimal kernel with specified scale factor. Rescore predictions based on Webster's rules.

Returns

rescored_predictions [array-like]

rescore(*self*)

Application of Webster's rescoring rules as described in the Cole-Kripke paper.

Returns

rescored [array-like] Array of rescored sleep/wake predictions

1.3 sleeppy.majorrest module

class sleeppy.majorrest.**MajorRestPeriod**(*df*, *fs*, *temperature_threshold=25.0*, *minimum_rest_block=30*, *allowed_rest_break=60*, *minimum_rest_threshold=0.1*, *maximum_rest_threshold=1000.0*)

Bases: object

Calculates the major rest period on a single day (24 hours) of geneactiv accelerometer data.

Methods

run(self)	Runs the algorithm to detect major rest period.

run(*self*)

Runs the algorithm to detect major rest period.

Returns

mrp [list of timestamp] Major rest period for the given 24 hour period.

1.4 sleeppy.sleep module

class sleeppy.sleep.**SleepPy**(*input_file*, *results_directory*, *sampling_frequency*, *aws_object=None*, *start_buffer='0s'*, *stop_buffer='0s'*, *start_time=''*, *stop_time=''*, *temperature_threshold=25.0*, *minimum_rest_block=30*, *allowed_rest_break=60*, *minimum_rest_threshold=0.1*, *maximum_rest_threshold=1000.0*, *minimum_hours=6*, *downsample=True*, *plot=True*)

Bases: object

Processes raw GeneActiv accelerometer data from the wrist to determine various sleep metrics and endpoints.

*** Support for data from other wrist worn accelerometers will be added in the future. To maximize generality an input format will be specified. As long as the input specifications are met, the package should produce the proper results. ***

The sleep window detection of this package are based off of the following papers, as well as their implementation in the R package GGIR:

van Hees V, Fang Z, Zhao J, Heywood J, Mirkes E, Sabia S, Migueles J (2019). GGIR: Raw Accelerometer Data Analysis. doi: 10.5281/zenodo.1051064, R package version 1.9-1, https://CRAN.R-project.org/package=GGIR.

van Hees V, Fang Z, Langford J, Assah F, Mohammad Mirkes A, da Silva I, Trenell M, White T, Wareham N, Brage S (2014). 'Autocalibration of accelerometer data or free-living physical activity assessment using local gravity and temperature: an evaluation on four continents.' Journal of Applied Physiology, 117(7), 738-744. doi: 10.1152/japplphysiol.00421.2014, https://www.physiology.org/doi/10.1152/japplphysiol.00421.2014

van Hees V, Sabia S, Anderson K, Denton S, Oliver J, Catt M, Abell J, Kivimaki M, Trenell M, Singh-Maoux A (2015). 'A Novel, Open Access Method to Assess Sleep Duration Using a Wrist-Worn Accelerometer.' PloS One, 10(11). doi: 10.1371/journal.pone.0142533, http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0142533.

The detection of sleep and wake states uses a heuristic model based on the algorithm described in:

Cole, R.J., Kripke, D.F., Gruen, W.'., Mullaney, D.J., & Gillin, J.C. (1992). Automatic sleep/wake identification from wrist activity. Sleep, 15 5, 461-9.

The activity index feature is based on the index described in:

Bai J, Di C, Xiao L, Evenson KR, LaCroix AZ, Crainiceanu CM, et al. (2016) An Activity Index for Raw Accelerometry Data and Its Comparison with Other Activity Metrics. PLoS ONE 11(8): e0160644. https://doi.org/10.1371/journal.pone.0160644

Methods

calculate_endpoints(self)	Calculates the clinical sleep endpoints for each 24 hour day.
calculate_major_rest_periods(self)	Gets the major rest periods for each 24 hour day.
calculate_sleep_predictions(self)	Gets the sleep/wake predictions for each 24 hour day.
export(self)	Exports a results table to csv.
load(self)	Loads full data into 24 hour days.
run(self)	Runs the full package end to end on the input data.
visualize(self)	Builds all visual reports and exports them to pdf.

calculate_endpoints (*self*)
Calculates the clinical sleep endpoints for each 24 hour day.

calculate_major_rest_periods (*self*)
Gets the major rest periods for each 24 hour day.

calculate_sleep_predictions (*self*)
Gets the sleep/wake predictions for each 24 hour day.

export (*self*)
Exports a results table to csv.

load (*self*)
Loads full data into 24 hour days.

run (*self*)
Runs the full package end to end on the input data.

visualize (*self*)
Builds all visual reports and exports them to pdf.

1.5 sleeppy.utils module

sleeppy.utils.**activity_index** (*windowed_array*)
Compute activity index of windowed tri-axis accelerometer signal.

Activity index defined here as: sqrt(mean(vx, vy, vz)) where vx, vy, vz are the variances in x, y, z

Activity index accounting for device specific systematic variance is defined as: sqrt(max(mean(vx-vsx, vy-vsy, vz-vsz), 0)) where vx, vy, vz are defined as above and vsx, vsy, vsz are the axis specific systematic variances

Parameters

windowed_array [array-like] Full X,Y,Z tri-axis accelerometer signal with dimensions [number of windows, size of window, 3]

Returns

activity_indices [array-like] Array of activity indices for input signal.

sleeppy.utils.**bin2df**(*full_path*)
Reads geneactiv .bin files into a pandas dataframe.

Parameters

full_path [str] Full path to the geneactiv .bin file.

Returns

decode [dataframe] Dataframe of decoded geneactiv data.

sleeppy.utils.**downsample**(*df*, *factor*)
Downsamples a pandas data frame to a desired frequency after using an antialiasing filter.

Parameters

df [data frame] Data to be downsampled.

factor [int] Factor by which to downsample.

Returns

df [data frame] Downsampled data frame

sleeppy.utils.**downsample_by_mean**(*df*, *fs='50L'*)
Downsamples a pandas data frame to a desired frequency by mean based aggregation.

Parameters

df [data frame] Data to be downsampled. Data frame must have pandas date time index.

fs [str] New sampling rate in string format.

Returns

downsampled data frame

sleeppy.utils.**high_pass_filter**(*windowed_array*, *sampling_rate*, *hp_cutoff*, *order*)
High-pass filter a given windowed sensor signal.

Parameters

windowed_array [array-like] Sensor signal windowed with shape [number_of_windows, samples_per_window, number_of_data_channels].

sampling_rate [float]

hp_cutoff [float] High pass filter cutoff.

order [int] Order of the filter.

Returns

filtered [array-like] High pass filtered data.

sleeppy.utils.**load_geneactiv_csv**(*full_path*)
Loads geneactiv csv data into a pandas dataframe.

Parameters

full_path [string] Full path to a geneactiv accelerometer csv file.

Returns

df [data frame] Pandas data frame of geneactiv accelerometer data including temperature and light.

sleeppy.utils.**make_dummy_data**(*freq='50L'*)

Makes dummy sleep data. Default dummy data is 24 hours at 20hz. Sleep window from 10pm to 8am. Temperature set at 27 degrees celsius.

Parameters

freq [string] Frequency expressed as string. 20hz Default expressed as "50L". 100hz as "10L".

Returns

input_df, start_sleep, end_sleep [3-tuple of data frame, date-time object, date-time object] Dataframe of dummy sleep data, start of sleep window, end of sleep window.

sleeppy.utils.**non_overlapping_windows**(*df*, *window_size*)

Slices input data frame into windows of specified size.

Parameters

df [data frame] Data frame to be windowed along the vertical axis.

window_size [int] Window size in number of samples.

Returns

windowed_data [numpy array] Data windowed to the specified size.

sleeppy.utils.**number_of_wake_bouts**(*predictions*)

Calculates the number of wake bouts present in an array of sleep/wake predictions in one minute epochs. Number of wake bouts is exclusive of first wake before sleep, and last wake after sleep.

Parameters

predictions [array-like] Binary sleep/wake predictions. Awake encoded as 1 and sleep as 0.

Returns

nwb [int] Total number of wake bouts based on sleep/wake predictions provided.

sleeppy.utils.**percent_time_asleep**(*predictions*)

Calculates the percent of time spent asleep on an array of sleep/wake predictions in one minute epochs.

Parameters

predictions [array-like] Binary sleep/wake predictions. Awake encoded as 1 and sleep as 0.

Returns

pta [float] Percentage of time spent asleep based on sleep/wake predictions provided.

sleeppy.utils.**sleep_onset_latency**(*predictions*)

Calculates sleep onset latency on an array of sleep/wake predictions in one minute epochs. This corresponds to the total number of minutes awake before the first sleep period.

Parameters

predictions [array-like] Binary sleep/wake predictions. Awake encoded as 1 and sleep as 0.

Returns

sol [int] Total number of minutes spent awake before the first sleep period.

sleeppy.utils.**total_sleep_time**(*predictions*)

Calculates total sleep time on an array of sleep/wake predictions in one minute epochs.

Parameters

predictions [array-like] Binary sleep/wake predictions. Awake encoded as 1 and sleep as 0.

Returns

tst [int] Total time spent asleep based on sleep/wake predictions provided.

sleeppy.utils.**wake_after_sleep_onset**(*predictions*)

Calculates number of minutes awake after first sleep period on an array of sleep/wake predictions in one minute epochs. Value is exclusive of the last wake period after sleep.

Parameters

predictions [array-like] Binary sleep/wake predictions. Awake encoded as 1 and sleep as 0.

Returns

waso [int] Total number of minutes spent awake after the first sleep period.

PYTHON MODULE INDEX

S

INDEX