scikit-cycling Documentation

Release 0.1.dev0

Guillaume Lemaitre

March 31, 2016

CONTENTS

Contents:

CHAPTER

ONE

SKCYCLING

1.1 skcycling package

1.1.1 Subpackages

skcycling.metrics package

Subpackages

skcycling.metrics.tests package

Submodules

skcycling.metrics.tests.test_ride module Testing the metrics developed to asses performance of a ride

skcycling.metrics.tests.test_ride.**test_ftp2pma**()
Testing the function converting the FTP to PMA

skcycling.metrics.tests.test_ride.**test_intensity_factor_ftp_score**()
Testing the function computing IF with FTP

skcycling.metrics.tests.test_ride.**test_intensity_factor_pma_score**()
Testing the function computing IF with PMA

skcycling.metrics.tests.test_ride.**test_normalized_power_score**()
Testing the function computing the NP

skcycling.metrics.tests.test_ride.**test_pma2ftp**()
Testing the function converting the PMA to FTP

skcycling.metrics.tests.test_ride.**test_training_stress_ftp_grappe_score**()
Testing the function to compute the stress based on ESIE and FTP

skcycling.metrics.tests.test_ride.**test_training_stress_ftp_score**()
Testing the function to compute the TSS from FTP

skcycling.metrics.tests.test_ride.**test_training_stress_pma_grappe_score**()
Testing the function to compute the stress based on ESIE and PMA

skcycling.metrics.tests.test_ride.**test_training_stress_pma_score**()
Testing the function to compute the TSS from PMA

Module contents

Submodules

skcycling.metrics.ride module

Metrics to asses the performance of a cycling ride

Functions named as `*_score` return a scalar value to maximize: the higher the better

Function named as `*_error` or `*_loss` return a scalar value to minimize: the lower the better

`skcycling.metrics.ride.`**`ftp2pma`**(*ftp*)

Convert the PMA to FTP

ftp [float] Functioning Threhold Power.

pma [float] Maximum Anaerobic Power.

`skcycling.metrics.ride.`**`intensity_factor_ftp_score`**(*X*, *ftp*)

Compute the intensity factor using the FTP

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

ftp [float] Functional Threshold Power.

score: float Return the intensity factor.

`skcycling.metrics.ride.`**`intensity_factor_pma_score`**(*X*, *pma*)

Compute the intensity factor using the PMAB

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maximum Anaerobic Power.

score: float Return the intensity factor.

`skcycling.metrics.ride.`**`normalized_power_score`**(*X*, *pma*)

Compute the normalized power for a given ride

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maxixum Anaerobic Power.

score [float] Return the normalized power.

`skcycling.metrics.ride.`**`pma2ftp`**(*pma*)

Convert the PMA to FTP

pma [float] Maximum Anaerobic Power.

ftp [float] Functioning Threhold Power.

`skcycling.metrics.ride.`**`training_stress_ftp_grappe_score`**(*X*, *ftp*)

Compute the training stress score using the FTP, considering Grappe et al. approach

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

ftp [float] Functional Threshold Power.

score: float Return the training stress score.

skcycling.metrics.ride.**training_stress_ftp_score**(*X*, *ftp*)

Compute the training stress score using the FTP

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

ftp [float] Functional Threshold Power.

score: float Return the training stress score.

skcycling.metrics.ride.**training_stress_pma_grappe_score**(*X*, *pma*)

Compute the training stress score using the PMA, considering Grappe et al. approach

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maximum Anaerobic Power.

tss_score: float Return the training stress score.

skcycling.metrics.ride.**training_stress_pma_score**(*X*, *pma*)

Compute the training stress score

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maximum Anaerobic Power.

score: float Return the training stress score.

Module contents

The *skcycling.metrics* module include score functions.

skcycling.metrics.**normalized_power_score**(*X*, *pma*)

Compute the normalized power for a given ride

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maxixum Anaerobic Power.

score [float] Return the normalized power.

skcycling.metrics.**intensity_factor_ftp_score**(*X*, *ftp*)

Compute the intensity factor using the FTP

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

ftp [float] Functional Threshold Power.

score: float Return the intensity factor.

skcycling.metrics.**intensity_factor_pma_score**(*X*, *pma*)

Compute the intensity factor using the PMAB

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maximum Anaerobic Power.

score: float Return the intensity factor.

skcycling.metrics.**training_stress_ftp_score**(*X*, *ftp*)

Compute the training stress score using the FTP

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

ftp [float] Functional Threshold Power.

score: float Return the training stress score.

skcycling.metrics.**training_stress_pma_score**(*X*, *pma*)

Compute the training stress score

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maximum Anaerobic Power.

score: float Return the training stress score.

skcycling.metrics.**pma2ftp**(*pma*)

Convert the PMA to FTP

pma [float] Maximum Anaerobic Power.

ftp [float] Functioning Threhold Power.

skcycling.metrics.**ftp2pma**(*ftp*)

Convert the PMA to FTP

ftp [float] Functioning Threhold Power.

pma [float] Maximum Anaerobic Power.

skcycling.metrics.**training_stress_pma_grappe_score**(*X*, *pma*)

Compute the training stress score using the PMA, considering Grappe et al. approach

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

pma [float] Maximum Anaerobic Power.

tss_score: float Return the training stress score.

skcycling.metrics.**training_stress_ftp_grappe_score**(*X*, *ftp*)

Compute the training stress score using the FTP, considering Grappe et al. approach

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

ftp [float] Functional Threshold Power.

score: float Return the training stress score.

skcycling.power_profile package

Subpackages

skcycling.power_profile.tests package

Submodules

skcycling.power_profile.tests.test_power_profile module Test the power profile class.

`skcycling.power_profile.tests.test_power_profile.`**`test_2`**`()`
Test if the rpp is computed properly.

`skcycling.power_profile.tests.test_power_profile.`**`test_rpp_load_no_weight`**`()`
Test the routine to read the Rpp with no weight.

Module contents

Submodules

skcycling.power_profile.rpp module

Record power-profile

This module contains class and methods related to the record power-profile.

class `skcycling.power_profile.rpp.`**`Rpp`**`(`*max_duration_rpp*, *cyclist_weight=None*`)`
Bases: `object`

Record power-profile

Can perform online updates via *partial_fit* method.

max_duration_rpp [int] Integer representing the maximum duration in minutes to build the record power-profile model.

cyclist_weight [float, default None] Float in order to normalise the record power-profile depending of its weight. By default this is None in order to avoid using the data from normalized rpp without this data.

rpp_ [array-like, shape (60 * max_duration_rpp,)] Array in which the record power-profile is stored. The units used is the second.

rpp_norm_ [array-like, shape (60 * max_duration_rpp,)] Array in which the weight-normalized record power-profile is stored. The units used is the seconds.

max_duration_rpp_ [int] The maximum duration of the record power-profile.

cyclist_weight_ [float] Cyclist weight.

`aerobic_meta_model``(`*ts=None*, *starting_time=4*, *normalized=False*, *method='lsq'*`)`

Compute the aerobic metabolism model from the record power-profile

ts [array-like, shape (n_samples,)] Array containing the sample to take into account. None if we want to pick up all the data.

start_time [int, default 4] Starting time to consider when fitting the linear model.

normalized [bool, default False] Return a weight-normalized rpp if True.

method [string, default 'lsq'] Which type of tehcnic to use to make the fitting ('lsq', 'lm').

slope [float] slope of the regression line.

intercept [float] intercept of the regression line.

stderr [float] Standard error of the estimate.

coeff_det [float] Coefficient of determination.

[1] Pinot et al., "Determination of Maximal Aerobic Power on the Field in Cylcing" (2014)

`fit` (*X*, *in_parallel=True*)

Fit the data to the RPP

X : array-like, shape (n_samples,)

in_parallel [boolean] If True, the rpp will be computed on all the available cores.

self [object] Returns self.

classmethod `load_from_npy` (*filename*, *cyclist_weight=None*)

Load the record power-profile from an npy file

filename [str] String containing the path to the NPY file containing the array representing the record power-profile.

cyclist_weight [float or None, default None] Float in order to normalise the record power-profile depending of its weight. By default this is None in order to avoid using the data from normalized rpp without this data.

self [object] Returns self

`partial_fit` (*X*, *refit=False*, *in_parallel=True*)

Incremental fit of the RPPB

X : array-like, shape (n_samples,)

in_parallel [boolean] If True, the rpp will be computed on all the available cores.

self [object] Returns self.

`resampling_rpp` (*ts*, *method_interp='linear'*, *normalized=False*)

Resampling the record power-profile

ts [array-like, shape (n_sample,)] An array containaining the time landmark to sample.

method_interp [string, default 'linear'] Name of the method to interpolate the data. Specifies the kind of interpolation as a string ('linear', 'nearest', 'zero', 'slinear', 'quadratic, 'cubic' where 'slinear', 'quadratic' and 'cubic' refer to a spline interpolation of first, second or third order) or as an integer specifying the order of the spline interpolator to use.

normalized [bool, default False] Return a weight-normalized rpp if True.

rpp [array-like, shape (n_samples,)] Return a resampled record power-profile.

`skcycling.power_profile.rpp.`**`compute_ride_rpp`** (*X*, *max_duration_rpp*, *in_parallel=True*)

Compute the record power-profile

X : array-like, shape (n_samples,)

in_parallel [boolean] If True, the rpp will be computed on all the available cores.

rpp [array-like, shape (n_samples,)] Array containing the record power-profile of the current ride.

Module contents

class `skcycling.power_profile.`**`Rpp`**(*max_duration_rpp*, *cyclist_weight=None*)

Bases: `object`

Record power-profile

Can perform online updates via *partial_fit* method.

max_duration_rpp [int] Integer representing the maximum duration in minutes to build the record power-profile model.

cyclist_weight [float, default None] Float in order to normalise the record power-profile depending of its weight. By default this is None in order to avoid using the data from normalized rpp without this data.

rpp_ [array-like, shape (60 * max_duration_rpp,)] Array in which the record power-profile is stored. The units used is the second.

rpp_norm_ [array-like, shape (60 * max_duration_rpp,)] Array in which the weight-normalized record power-profile is stored. The units used is the seconds.

max_duration_rpp_ [int] The maximum duration of the record power-profile.

cyclist_weight_ [float] Cyclist weight.

`aerobic_meta_model`(*ts=None*, *starting_time=4*, *normalized=False*, *method='lsq'*)

Compute the aerobic metabolism model from the record power-profile

ts [array-like, shape (n_samples,)] Array containing the sample to take into account. None if we want to pick up all the data.

start_time [int, default 4] Starting time to consider when fitting the linear model.

normalized [bool, default False] Return a weight-normalized rpp if True.

method [string, default 'lsq'] Which type of tehcnic to use to make the fitting ('lsq', 'lm').

slope [float] slope of the regression line.

intercept [float] intercept of the regression line.

stderr [float] Standard error of the estimate.

coeff_det [float] Coefficient of determination.

[1] Pinot et al., "Determination of Maximal Aerobic Power on the Field in Cylcing" (2014)

`fit`(*X*, *in_parallel=True*)

Fit the data to the RPP

X : array-like, shape (n_samples,)

in_parallel [boolean] If True, the rpp will be computed on all the available cores.

self [object] Returns self.

classmethod **`load_from_npy`**(*filename*, *cyclist_weight=None*)

Load the record power-profile from an npy file

filename [str] String containing the path to the NPY file containing the array representing the record power-profile.

cyclist_weight [float or None, default None] Float in order to normalise the record power-profile depending of its weight. By default this is None in order to avoid using the data from normalized rpp without this data.

self [object] Returns self

`partial_fit`(*X*, *refit=False*, *in_parallel=True*)

Incremental fit of the RPPB

X : array-like, shape (n_samples,)

in_parallel [boolean] If True, the rpp will be computed on all the available cores.

self [object] Returns self.

`resampling_rpp`(*ts*, *method_interp='linear'*, *normalized=False*)

Resampling the record power-profile

ts [array-like, shape (n_sample,)] An array containaining the time landmark to sample.

method_interp [string, default 'linear'] Name of the method to interpolate the data. Specifies the kind of interpolation as a string ('linear', 'nearest', 'zero', 'slinear', 'quadratic, 'cubic' where 'slinear', 'quadratic' and 'cubic' refer to a spline interpolation of first, second or third order) or as an integer specifying the order of the spline interpolator to use.

normalized [bool, default False] Return a weight-normalized rpp if True.

rpp [array-like, shape (n_samples,)] Return a resampled record power-profile.

`skcycling.power_profile.`**`compute_ride_rpp`**(*X*, *max_duration_rpp*, *in_parallel=True*)

Compute the record power-profile

X : array-like, shape (n_samples,)

in_parallel [boolean] If True, the rpp will be computed on all the available cores.

rpp [array-like, shape (n_samples,)] Array containing the record power-profile of the current ride.

skcycling.restoration package

Subpackages

skcycling.restoration.tests package

Submodules

skcycling.restoration.tests.test_denoise module

`skcycling.restoration.tests.test_denoise.`**`test_moving_average`**()

Test the moving average

`skcycling.restoration.tests.test_denoise.`**`test_outliers_thres_rejection`**()

Test the outlier rejection method based on thresholding

`skcycling.restoration.tests.test_denoise.`**`test_outliers_unknown_method`**()

Test to check if an error is risen in case the method is unknown

Module contents

Submodules

skcycling.restoration.denoise module

Methods to denoise the power signal provided from a ride

`skcycling.restoration.denoise.`**`moving_average`**(*X*, *win=30*)

Apply an average filter to the data

X [array-like, shape (n_samples,)] Array containing the ride or a selection of a ride.

win [interger] Size of the sliding window.

avg [array-like (float)] Return the denoised data mean-filter.

`skcycling.restoration.denoise.`**`outliers_rejection`**(*X*, *method='threshold'*, *thres=2500.0*)

Remove the outliers from the given ride

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

method [string, default 'threshold'] String to specified which outliers detection method to use.

thres [float, default 2500.] The maximum power to consider in case the method 'threshold' is considered.

X [array-like, shape (n_samples,)] Array containing the power intensities, outliers free.

Module contents

The *`skcycling.restoration`* module include denoising methods.

`skcycling.restoration.`**`outliers_rejection`**(*X*, *method='threshold'*, *thres=2500.0*)

Remove the outliers from the given ride

X [array-like, shape (n_samples,)] Array containing the power intensities for a ride.

method [string, default 'threshold'] String to specified which outliers detection method to use.

thres [float, default 2500.] The maximum power to consider in case the method 'threshold' is considered.

X [array-like, shape (n_samples,)] Array containing the power intensities, outliers free.

`skcycling.restoration.`**`moving_average`**(*X*, *win=30*)

Apply an average filter to the data

X [array-like, shape (n_samples,)] Array containing the ride or a selection of a ride.

win [interger] Size of the sliding window.

avg [array-like (float)] Return the denoised data mean-filter.

skcycling.utils package

Subpackages

skcycling.utils.tests package

Submodules

skcycling.utils.tests.test_checker module Testing the checker methods

`skcycling.utils.tests.test_checker.`**`test_check_float_convertion`**`()`

Test if an integer is converted to float

`skcycling.utils.tests.test_checker.`**`test_check_float_no_conversion`**`()`

Test if a float is not converted when a float is given

`skcycling.utils.tests.test_checker.`**`test_check_x_convert_float`**`()`

Test if array X is converted into float if the input is not.

`skcycling.utils.tests.test_checker.`**`test_check_x_not_vector`**`()`

Test if an error is risen if X is not a vector.

`skcycling.utils.tests.test_checker.`**`test_check_x_np_float64`**`()`

Test everything goes fine with numpy double.

skcycling.utils.tests.test_io_fit module Testing the input/output methods for FIT files

`skcycling.utils.tests.test_io_fit.`**`test_load_power_check_file_exist`**`()`

Test if an error is risen if the FIT file does not exist.

`skcycling.utils.tests.test_io_fit.`**`test_load_power_if_no_power`**`()`

Test if a warning if raise if there is no power data.

`skcycling.utils.tests.test_io_fit.`**`test_load_power_normal_file`**`()`

Test if a normal file can be loaded correctly.

`skcycling.utils.tests.test_io_fit.`**`test_load_power_not_fit`**`()`

Test if an error is risen in case that the file is not a FIT file.

Module contents

Submodules

skcycling.utils.checker module

Helper function to check data conformity

skcycling.utils.io_fit module

Methods to handle input/output files.

`skcycling.utils.io_fit.`**`load_power_from_fit`**`(`*filename*`)`

Method to open the power data from FIT file into a numpy array.

filename [str,] Path to the FIT file.

power_rec [ndarray, shape (n_samples)] Power records of the ride.

Module contents

`skcycling.utils.`**`load_power_from_fit`**(*filename*)

Method to open the power data from FIT file into a numpy array.

filename [str,] Path to the FIT file.

power_rec [ndarray, shape (n_samples)] Power records of the ride.

1.1.2 Submodules

1.1.3 skcycling.setup module

`skcycling.setup.`**`configuration`**(*parent_package=''*, *top_path=None*)

1.1.4 Module contents

Cycling Processing Toolbox (Toolbox for SciPy)

`scikit-cycling` (a.k.a `skcycling`) is a set of python methods to analyse file extracted from powermeters.

Subpackages

metrics Metrics to quantify cyclist ride.

power_profile Record power-profile of cyclist.

restoration Utility for denoising cyclist ride.

utils Utility to read and save cycling ride.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX