Reference manual for the cox-fatigue-life module

Managed by
Kubat Narynbek Ulu
Ecole Centrale de Nantes, LRCCP
Version 1

This manual describes the functionality of the `cox-fatigue-life` *Python* module developed for statistical analysis of of fatigue life results with the use of the Cox regression (proportional hazards model).

Contents

1 Installation

1.1 Prerequisites

For ease of installation and avoidance of conflicts between packages, it is recommended to use *Anaconda* distribution of *Python*, it is open source and can be downloaded for free from `https://www.anaconda.com/`. The present code has been tested in version 5.2.0. Otherwise, please install the following:

Python — tested version 3.6.4; `http://www.python.org/`; pre-installed in *Anaconda.*

R — tested version 3.5.1; `https://www.r-project.org/`; pre-installed in *Anaconda.*

NumPy — tested version 1.14.0; `https://www.scipy.org/`; pre-installed in *Anaconda.*

Matplotlib — tested version 2.1.2; `https://www.scipy.org/`; pre-installed in *Anaconda.*

rpy2 — tested version 2.9.1; `https://pypi.org/project/rpy2/`; installation required in *Anaconda.*

1.2 Installation

The installation is straightforward. Copy the `cox_fatigue_analysis.py` and `r_scipt_cox.R` into your working directory or by creating a folder named “CoxFatigueAnalysis” in the python directory `/lib/site-packages/`.

2 Module functions

2.1 Constructor functions

CoxFatigueAnalysis(...) — Creates the base python object that is used in subsequent analysis.

Input variables:

1. *list_of_covariate_names* - a python list of names of covariates (strings), should be lower-case, not start with number, not contain spaces and not be *'fatigue_life'* or *'fatigue_survival'*
2. *list_of_covariate_data_types* - a python list of data type of each respective covariate; these can be either python or numpy data types (e.g. float, int, str, etc.)

Returns: *None*

Example: creating an object for analysis of 3 covariates displacement (continuous - decimal number), material type (categorical - name), and material batch number (integer value).

```
CoxFatigueAnalysis([’displacement’, ’material_type’,
’batch_number’], [float, str, int])
```

2.2 Data import functions

import_from_csv(...) — Inputs fatigue life data from a CSV file.

Input variables:

1. *filepath* - path and name of the file; if the file is in the working directory, only the name needs to be indicated, otherwise the full path; the details of data arrangement is described in the article; in summary: column 1 is fatigue life, column 2 is survival status of a sample (0 for survived, 1 for failed), column 3 is the applied load, the remaining columns are other covariates; refer to `example.csv` for an example.

- *separator* - separator used in creation of the CSV file; default is ’,’; e.g. French systems create CSV files with ’;’

Returns: *None*

Example: `import_from_csv(’example.csv’, ’;’)`

2.3 Analysis functions

cox_regression()

Runs the cox regression within the R-environment by calling R-function `coxph`. Executing *cox_regression*() is required before executing any other function of the present module (except for the constructor).

Input variables: *None*

Returns: *None*

cox_zph()

Runs the proportionality test within the R-environment by calling R-function `coxzph`. Executing *cox_regression*() is required before calling *cox_zph*()

Input variables: *None*

Returns: *None*

get_cox_survfit_1var(...)

Computes the predicted survivor function for a Cox proportional hazards model for a single covariate.

Input variables:

1. *input_covariate* - the name of the covariate (string), indicated in the constructor, that is to be analyzed.

2. *input_covariate_data* - a python list of corresponding values of the covariate to be analyzed.

3. *constant_covariates* - a python list of covariate names (strings) that are to be kept constant.

4. *constant_covariates_values* - a python list of corresponding values of the covariates to be kept constant; must equal in length to *constant_covariates*.

Returns:

1. A NumPy array of survival times.

2. A NumPy array of survival probabilities.

3. A NumPy array of survival statuses.

Example: computing a survivor function for covariate *displacement*, while keeping the other two covariates constant: only considering ’material a’ and batch number 21.

```
get_cox_survfit_1var(’displacement’, [1, 2, 3, 4],
[’material_type’, ’batch_number’], [’material_a’, 21])
```

2.4 Plotting functions

plot_cox_survival_1var(...)

Plots a survival function estimate for a single covariate.

Input variables:

1. *axes* - a Matplotlib axes object, where the survival function will be plotted.
2. *input_covariate* - the name of the covariate (string), indicated in the constructor, that is to be analyzed.
3. *input_covariate_data* - a python list of corresponding values of the covariate to be analyzed.
4. *constant_covariates* - a python list of covariate names (strings) that are to be kept constant.
5. *constant_covariates_values* - a python list of corresponding values of the covariates to be kept constant; must equal in length to *constant_covariates*.

- $axes_format$ - default value is `True`; overrides the formatting options of the passed Matplotlib axes object; sets x-axis label to 'Time', y-axis label to 'Survival Probability Estimate', y-axis limits from 0 to 1, and puts the legend in the lower left corner.

Returns: *None*

Example: plotting survival function for covariate *displacement* at values of 1 mm, 2 mm, 3 mm, 4 mm, while keeping the other two covariates constant: only considering 'material a' and batch number 21; overriding axes formatting is disabled.

```
plot_cox_survival_1var(axes, 'displacement', [1, 2, 3, 4],
['material_type', 'batch_number'], ['material_a', 21], False)
```

plot_wohler_curve(...)

Plots a probabilistic *S-N* curve based on Cox regression. Contour lines are plotted for the mean, the 95% confidence intervals, and the minimum and maximum threshold probabilities.

Input variables:

1. *axes* - a Matplotlib axes object, where the survival function will be plotted.
2. *load_covariate_name* - name of the covariate (string) corresponding to the loading variable (S in the S-N curve).
3. *load_range* - a python list of minimum and maximum load values `[min, max]`.
4. *load_resolution* - resolution of the load-axis on the wohler curve; i.e. the interval at which the survival function is estimated; e.g. with *load_range* `min`=1 and `max`=10 and resolution of 1, the survival will be estimated at values of 1, 2, 3 ... 9, 10.
5. *constant_covariates* - a python list of covariate names (strings) that are to be kept constant.
6. *constant_covariates_values* - a python list of corresponding values of the covariates to be kept constant; must equal in length to *constant_covariates*.

- *axes_format* - overrides the formatting options of the passed Matplotlib axes object; sets x-axis label to 'Time', y-axis label to 'Load'; default value is `True`.

- *contour_regions_interval* - controls the probability interval at which color changes; decimal format; default value is 0.05 (5% percent).

- *contour_min*$'_t$*hreshold* - the minimum survival probability threshold for color change; default value is 0.001 (e.g. almost complete failure at survival probability of 0.1%).

- *contour_max_threshold* - maximum survival probability threshold for color changes; default value is 0.999 (e.g. almost complete survival of 99.9%).

- *colormap* - selection of a Matplotlib colormap; more info at `https://matplotlib.org/examples/color/colormaps_reference.html`; default colormap is viridis.

- *linecolors* - color of contour lines; default color is black.

- *linestyles* - a python list of Matplotlib styles of contour lines corresponding to the min threshold, min 95% confidence interval, mean, max 95% confidence interval, and max threshold; default is `['solid', 'dotted', 'solid', 'dotted', 'solid']`.

Returns: a python list of:

1. raw values of survival estimates (times, probabilities, loading values).

2. a line contour object of Matplotlib.

3. the min/max probability thresholds inputted during plotting.

Example: plotting a probabilistic *S*-*N* curve for displacement loading range from 1 to 10 mm (resolution at each 0.1 mm) of 'material a' from batch 21.

```
plot_wohler_curve(axes, 'displacement', [1, 10], 0.1,
['material_type', 'batch_number'], ['material_a', 21])
```

compare_curves(...) Plots a comparison of two probabilistic *S*-*N* curves. The user can choose whether to compare the failure, survival, min or max 95% confidence intervals, or the mean contour lines of each *S*-*N* curve.

Input variables:

1. *axes* - a Matplotlib axes object, where the survival function will be plotted.

2. *curve*1 - the return of the first *S*-*N* curve plotted by `plot_wohler_curve(...)`.

3. *curve*2 - the return of the second *S*-*N* curve plotted by `plot_wohler_curve(...)`.

4. *curve_to_plot* - choice of which contour line to plot; probability values: `'failure'` corresponds to `contour_min_threshold` (default is 0.001), `'survival'` corresponds to `contour_max_threshold` (default is 0.999), `'min95'` and `'max95'` correspond to min and max 95% confidence intervals, `'mean'` corresponds to the mean (0.5).

- *labels* - a python list of names (strings) of length 2 for the labels of each curve; default is `['curve 1', 'curve 2']`.

- *curve_colors* - a python list of Matplotlib colors of length 2 for each curve; default is `['black', 'grey']`.

Returns: *None*

Example: Plotting two probabilistic *S-N* curves of two materials (a and b, all other conditions are the same) and then plotting the comparison of their mean values.

```
curve 1 = plot_wohler_curve(axes, 'displacement', [1, 10], 0.1,
['material_type', 'batch_number'], ['material_a', 21])
curve 2 = plot_wohler_curve(axes, 'displacement', [1, 10], 0.1,
['material_type', 'batch_number'], ['material_b', 21])
```

```
compare_curves(axes2, curve1, curve2, 'mean')
```

2.5 Printing functions

print() Prints the results of the cox regression for each covariate: β, standard error, hazard ratio and its 95% confidence intervals, p-value.

Input variables: *None*

Returns: *None*

Example: `CoxFatigueAnalysis.print()`

print_cox_r_output() Prints the output of the R-environment function `summary` of the cox analysis object.

Input variables: *None*

Returns: *None*

Example: `CoxFatigueAnalysis.print_cox_r_output()`

Index