Bedload Season Analyst

Release latest

sschwindt, beatriznegreiros

Aug 30, 2023

CONTENTS

This is the code documentation for algorithms applied for processing data in a paper entitled Meta-analysis of a large bedload transport rate dataset by Sebastian Schwindt, Beatriz Negreiros, Bridget Ochuko Mudiaga-Ojemu, and Marwan A. Hassan in the journal Geomorphology in May 2023. Do not hesitate to contact us for inquiries regarding this paper.

How to cite this code and data repository

If our study and codes helped you to accomplish your work, we won't ask you for a coffee, but to cite and spread the utility of our code - Thank you!

```
@software{bedload_seasons_2023,
          author       = {Sebastian Schwindt and
                          Beatriz Negreiros},
          title        = {Bedload Meta-analysis - Codes and Data},
          year         = 2023,
          publisher    = {GitHub \& Center for Open Science (OSF)},
          version      = {v1},
          doi          = {10.17605/OSF.IO/3ZMKN},
          url          = {https://doi.org/10.17605/OSF.IO/3ZMKN}
        }
```

Note: This documentation is also as available as style-adapted PDF.

CHAPTER

ONE

REQUIREMENTS

Time requirement: 5-10 min.

To get the code running, the following software is needed and their installation instructions are provided below:

```
fitter>=1.5.2
matplotlib>=3.1.2
numpy>=1.17.4
openpyxl>=3.0.9
pandas>=1.3.5
scikit_learn>=1.2.0
scikit_posthocs>=0.7.0
scipy>=1.7.3
seaborn>=0.12.2
statannotations>=0.5.0
statsmodels>=0.13.5
```

Tip: New to Python?

Start with downloading and installing the latest version of Anaconda Python. Alternatively, downloading and installing a pure Python interpreter will also work. Detailed information about installing Python is available in the Anaconda Docs and at hydro-informatics.com/python-basics.

To install the requirements in a new conda environment, download our specific environment.yml. Then open Anaconda Prompt (e.g., click on the Windows icon, tap `anaconda prompt`, and hit enter``). In Anaconda Prompt, enter the following command sequence to install the libraries in the **base** environment. The installation may take a while depending on your internet speed.

```
conda env create -f environment.yml
```

This will have created a new conda environment named `bed-data-env`. After the installation, activate the environment:

```
conda activate bed-data-env
```

If you are struggling with the dark window and blinking cursor of Anaconda Prompt, worry not. You can also pip-install the requirements following the instructions on hydro-informatics.com for virtual environments.

CHAPTER

TWO

GET CODE AND DATA

Open any git-able Terminal (get git at https://git-scm.com), and enter:

```
https://github.com/sschwindt/bedload-seasons.git
```

CHAPTER

THREE

USAGE

3.1 Configure

The `pyBedLoad/config.py` script imports all relevant packages an can be used to:

- modify constants, such as grain or water density
- switch between using all samplers in `data/bedload-data.xlsx` or comparable samplers only (default) in `data/bedload-data-valid-samplers.xlsx`

In addition, modifications to plot properties can be made in `pyBedLoad/mor_fun.py` (for *morphology fun* or *more fun* - choose your favorite), which provides the core functions for boxplots including Kruskal-Wallis H (KWH) tests, dataset completeness (application not explained in the docs), and global Spearman rank correlation matrix.

3.2 Global Correlations

To create a plot of the Spearman ranked correlation matrix shown in the manuscript, run the following command in an active Python environment (make sure to be in the repository `HOME/`):

```
python plot_correlations.py
```

After a successful run, this script will have saved a plot of the correlation matrix in `HOME/figures/` with the name `dataset-corr-spearman.png`. The code also creates a workbook correlation with the correlation matrix (`HOME/corr-spearman.xlsx`). Note that we hard-coded the Spearman rank correlation method in `pyBedLoad/mor_fun.py` (line 323 defining `corr_mat = df.corr(method='spearman')` – sorry for the laziness), which can also be changed to `pearson` or `kendall`.

3.3 Boxplots and (KWH) tests

To create the boxplots with with Kruskal-Wallis H (KWH) tests shown in the manuscript and supplemental material, run the following command in an active Python environment (make sure to be in the repository `HOME/`):

```
python bedload_base_stats.py
```

Running this script can take a couple of minutes on a slow computer, and boxplot-figures in the directory `HOME/figures/`.

3.4 Histograms of Variable Frequency

To re-make the histogram-like frequency plots showing the number of measurements per category in the supplemental material, run the following command in an active Python environment (make sure to be in the repository `HOME/`):

```
python plot_histograms.py
```

This script will have created multiple figures showing the histograms in the directory `HOME/figures/histograms/`.

CHAPTER

FOUR

CODE DOCUMENTATION

The following sections provide details of functions, their arguments, and outputs to help tweaking the code for individual purposes.

4.1 mor_fun.py

Script provides functions for application at all levels, for instance, to plot data. more_fun is an acronym for 'morphoanalyst functions' or 'more fun', depending on your preference

bedPyLoad.mor_fun.**annotated_plot**(*df*, *target_var*, *num_var*, *x_label=None*, *y_label=None*, *plot_type='boxplot'*, *fig_format='png'*, *fig_path=None*, *color_palette=None*, *dpi=300*, *bbox='tight'*, *test='Kruskal'*, *text_format='simple'*, *text_offset=7*, *y_scale=None*)

Make an annotated plot with statannotations.stats. Read more about statannoation usage at https://github.com/trevismd/statannotations/tree/master/usage

Parameters

- **df** (*pd.DataFrame*) – DataFrame containing categorical and numerical data to be boxplotted. Categories will occur on the x-axis according to target_var. Numerical data according tu num_var
- **target_var** (*str*) – name of a target variable that must be contained in the column names of df
- **num_var** (*str*) – name of a numerical variable on the Y-axis that must be contained in the column names of df
- **x_label** (*str*) – if provided, this string replaces the column name of target_var (categorical)
- **y_label** (*str*) – if provided, this string replaces the column name of the numeric y variable
- **plot_type** (*str*) – default is 'boxplot', options are 'violinplot', 'swarmplot'
- **fig_format** (*str*) – file ending of image file; default is 'png'
- **fig_path** (*str*) – name and directory of image (figure) to save WITHOUT FILE FORMAT ending, MUST end on '/'
- **color_palette** (*str*,*list*,*dict*) – colors to be used with the *hue* variable
- **dpi** (*int*) – dots per inch for figure (default is 300)
- **bbox** (*str*) – default 'tight' applies narrow figure margins
- **test** (*str*) – type of statistical test for calculating p-values. Default is 'Kruskal'. Options are defined in statannotations.stats.StatTest.STATTEST_LIBRARY (line 88ff)

- **text_format** (*str*) – formatting of p-value annotations. Default is 'simple'. Options are 'star' and 'full'
- **text_offset** (*int*) – number of pixels for offset of p-value annotations. Default is 5.
- **y_scale** (*str*) – default is None but can be set to 'log' for logarithmic y axis.

Return int
0 = success, -1 = error occurred

bedPyLoad.mor_fun.**get_color_list**(*n*, *name='hsv'*)

Returns a list of n RGB colors

Parameters

- **n** – size of colormap list
- **name** (*str*) – type of color map - must be a standard matplotlib colormap name

Return list
colormap of size n

bedPyLoad.mor_fun.**plot_df_completeness**(*df*, *figure_base_name='base'*, *replace_col_names=None*)

Uses missingno package to create a plot of dataframe completeness

Parameters

- **df** (*pandas.DataFrame*) – Dataframe to be plotted
- **figure_base_name** (*str*) – syllable to be used with figure names
- **replace_col_names** (*dict*) – optional argument to overwrite column names

Returns
write plot

bedPyLoad.mor_fun.**plot_df_correlations**(*df*, *figure_base_name='base'*, *fontsize=16*, *replace_col_names=None*)

Creates a heatmap plot of correlations

Parameters

- **df** (*pandas.DataFrame*) – Dataframe to be plotted
- **figure_base_name** (*str*) – syllable to be used with figure names
- **fontsize** (*int*) – font size
- **replace_col_names** (*dict*) – optional argument to overwrite column names

Returns
write plot

bedPyLoad.mor_fun.**stats_test**(*dataframe: pandas.DataFrame*, *numeric_var_name: str*, *target_columns: list*, *numeric_var_as_categories_name: str = None*, *stats_results_xlsx: str = 'stats-results.xlsx'*, *figure_path: str = 'fitting-results/figures/'*)

Runs Dunn posthoc test on categories with reference to a non-normally distributed variable defined as *numeric_var_name*. This function is tweaked for this package and requires the global variables defined in config.py.

Parameters

- **dataframe** – A pandas dataframe containing all numeric and categorical data
- **numeric_var_name** – Name of a numerical variable (typical response variable) to be tested. MUST be a column name of *dataframe*

- **target_columns** – List of column names to be tested for differences with the numeric variable
- **numeric_var_as_categories_name** – For the Dunn test, the numerical variable should also be categorized (e.g. in categories 'low', 'average', 'high'). This argument is the name of a column in *dataframe* that contains the numerical variable as categories.
- **stats_results_xlsx** – Name of an xlsx file to store Dunn test results (default name applies if not provided)
- **figure_path** – directory or subdirectory where figures will be stored; MUST end on '/'

Return int success
successful execution when 0, otherwise -1

4.2 workbook_fun.py

4.3 bedload_base_stats.py

4.4 plot_correlations.py

Plot and save dataset completeness and Spearman correlatins

4.5 plot_histograms.py

Plot and save histograms of measurement frequency per variable category

plot_histograms.**plot_category_histograms**(*directory*)

plot histograms of all dataframe columns (as per the config.py) :param str directory: tell where plots should be saved; either absolute or relative path ending on '/'

Important: relative paths should not start with '/' or ''

Returns
0 if successful

CHAPTER

FIVE

DISCLAIMER AND LICENSE

5.1 Disclaimer (general)

No warranty is expressed or implied regarding the usefulness or completeness of the information provided for *bedload-seasons* and its documentation. References to commercial products do not imply endorsement by the Authors of *bedload-seasons*. The concepts, materials, and methods used in the codes and described in the docs are for informational purposes only. The Authors have made substantial effort to ensure the accuracy of the code and the docs and the Authors shall not be held liable, nor their employers or funding sponsors, for calculations and/or decisions made on the basis of application of *bedload-seasons*. The information is provided "as is" and anyone who chooses to use the information is responsible for her or his own choices as to what to do with the code, docs, and data and the individual is responsible for the results that follow from their decisions.

5.2 BSD 3-Clause License

PYTHON MODULE INDEX

INDEX