skextremes Documentation

Kiko Correoso

Apr 10, 2022

CONTENTS

scikit-extremes is a python library to perform univariate extreme value calculations.

There are two main classical approaches to calculate extreme values:

- Gumbel/Generalised Extreme Value distribution (GEV) + Block Maxima.
- Generalised Pareto Distribution (GPD) + Peak-Over-Threshold (POT).

CHAPTER

ONE

DEPENDENCIES

To work with scikit-extremes you will need the following libraries:

- Numpy
- Scipy
- Matplotlib
- Numdifftools

CHAPTER

TWO

INSTALLATION

At this moment there isn't an official release. To install the package you can follow the next steps:

```
# lmoments has not been updated in a while so we use the master
# (see https://github.com/OpenHydrology/lmoments3/issues/8)
pip install git+https://github.com/OpenHydrology/lmoments3.git

git clone https://github.com/kikocorreoso/scikit-extremes.git

cd scikit-extremes

pip install -e .
```

CHAPTER

THREE

SUPPORT

If you find a bug, something wrong or want a new feature, please, open a new issue on Github.

If you want to ask about the usage of scikit-extremes or something related with extreme value theory/analysis with Python you can post a question at stackoverflow tagged with `scikit-extremes` or `skextremes`.

CHAPTER

FOUR

LICENSE

This software is licensed under the MIT license except:

- `skextremes.utils.bootstrap_ci` function that is based on the scikits-bootstrap package licensed under the Modified BSD License.

CHAPTER

FIVE

CONTENTS:

5.1 Quick and incomplete Extreme Value Theory introduction

Extreme Value Theory (EVT) is unique as a statistical discipline in that it develops techniques and models for describing the unusual rather than the usual, e.g., it is focused in the tail of the distribution.

By definition, extreme values are scarce, meaning that estimates are often required for levels of a process that are much greater than have already been observed. This implies an extrapolation from an small set of observed levels to unobserved levels. Extreme Value Theory provides models to enable such extrapolation.

5.1.1 Fields of interest

Applications of extreme value theory include predicting the probability distribution of:

- Several engineering design processes:
- Hydraulics engineering (extreme floods,. . .)
- Structural engineering (earthquakes, wind speed,. . .)
- Meteorology
- Extreme temperatures, rainfall, Hurricanes. . .
- Ocean engineering
- The size of freak waves (wave height)
- Environmental sciences
- Large wildfires
- Environmental loads on structures
- Insurance industry
- The amounts of large insurance losses, portfolio adjustment,. . .
- Financial industry
- Equity risks
- Day to day market risk
- Stock market crashes
- Material sciences
- Corrosion analysis

- Strenght of materials
- Telecommunications
- Traffic prediction
- Biology
- Mutational events during evolution
- Memory cell failure
- ...

5.1.2 A brief history of Extreme Value Theory

One of the earliest books on the statistics of extreme values is E.J. Gumbel (1958). Research into extreme values as a subject in it's own right began between 1920 and 1940 when work by E.L. Dodd, M. Frêchet, E.J. Gumbel, R. von Mises and L.H.C. Tippett investigated the asymptotic distribution of the largest order statistic. This led to the main theoretical result: the Extremal Types Theorem (also known as the Fisher–Tippett–Gnedenko theorem, the Fisher–Tippett theorem or the extreme value theorem) which was developed in stages by Fisher, Tippett and von Mises, and eventually proved in general by B. Gnedenko in 1943.

Until 1950, development was largely theoretical. In 1958, Gumbel started applying theory to problems in engineering. In the 1970s, L. de Haan, Balkema and J. Pickands generalised the theoretical results (the second theorem in extreme value theory), giving a better basis for statistical models.

Since the 1980s, methods for the application of Extreme Value Theory have become much more widespread.

5.1.3 General approaches to estimate extreme values

There are two primary approaches to analyzing extremes of a dataset:

- The first and more classical approach reduces the data considerably by taking maxima of long blocks of data, e.g., annual maxima. The generalized extreme value (GEV) distribution function has theoretical justification for fitting to block maxima of data.
- The second approach is to analyze excesses over a high threshold. For this second approach the generalized Pareto (GP) distribution function has similar justification for fitting to excesses over a high threshold.

Block-Maxima + Generalised Extreme Value (GEV) and Gumbel distribution

The generalized extreme value (GEV) family of distribution functions has theoretical support for fitting to block maximum data whereby the blocks are sufficiently large, and is given by:

$$G(z;\mu,\sigma,\xi) = exp\{-[1+\xi\frac{z-\mu}{\sigma}]^{-1/\xi}\}$$

The parameters μ ($-\infty < \mu < \infty$), σ ($\sigma > 0$) and ξ ($\infty < \xi < \infty$) are location, scale and shape parameters, respectively. The value of the shape parameter ξ differentiates between the three types of extreme value distribution in Extremal Types Theorem (also known as the Fisher–Tippett–Gnedenko theorem, the Fisher–Tippett theorem or the extreme value theorem).

- $\xi = 0$, leading to, corresponds to the Gumbel distribution (type I). This special case can be formulated as

$$G(z;\mu,\sigma) = exp\{-exp(\frac{z-\mu}{\sigma})\}$$

- $\xi > 0$ correspond to the Frêchet (type II) and

- $\xi < 0$ correspond to the Weibull (type III)

distributions respectively. In practice, when we estimate the shape parameter ξ, the standard error for ξ accounts for our uncertainty in choosing between the three models.

Peak-Over-Threshold (POT) + Generalised Pareto (GP) distribution

TODO

TODO

TODO

TODO

5.1.4 References used to prepare this section

- S. Coles (2001): *An introduction to statistical modelling of extreme values*. Springer.
- E. Gilleland, , M. Ribatet and A. G. Stephenson (2013): *A software review for extreme value analysis*. Extremes, 16 (1), 103 - 119.
- L. Fawcett (2013): *Teaching materials of MAS8391 at Newcastle University (UK)*.

5.2 User guide

First of all you should import the package:

```
import skextremes as ske
import matplotlib.pyplot as plt
%matplotlib inline
```

Some datasets are included in the package. For example, we will use sea level data from Port Pirie, in Australia.

```
data = ske.datasets.portpirie()
```

```
print(data.description)
```

```
Annual Maximum Sea Levels at Port Pirie, South Australia
--------------------------------------------------------

Fields:
 year: numpy.array defining the year for the row data.
 sea_level: numpy.array defining annual maximum sea level recorded at
    Port Pirie, South Australia.

Source:
 -Coles, S. G. (2001). An Introduction to Statistical Modelling of
    Extreme Values. London: Springer.
```

As can be seen from the description of the dataset we have two fields, `sea_level`, with the annual maximum sea level records, and `year`, indicating the year of the record.

To get the dataset we can use the `.asarray` method to obtain all the fields as a `numpy.array` or just select a field to get a 1D `numpy.array` with the records for the field.

```
data_array = data.asarray()
sea_levels = data.fields.sea_level
```

```
print(sea_levels)
```

```
[ 4.03  3.83  3.65  3.88  4.01  4.08  4.18  3.8   4.36  3.96  3.98  4.69
  3.85  3.96  3.85  3.93  3.75  3.63  3.57  4.25  3.97  4.05  4.24  4.22
  3.73  4.37  4.06  3.71  3.96  4.06  4.55  3.79  3.89  4.11  3.85  3.86
  3.86  4.21  4.01  4.11  4.24  3.96  4.21  3.74  3.85  3.88  3.66  4.11
  3.71  4.18  3.9   3.78  3.91  3.72  4.    3.66  3.62  4.33  4.55  3.75
  4.08  3.9   3.88  3.94  4.33]
```

We have several options, located in the `skextremes.models` subpackage, to calculate extreme values. Depending the chosen model some methods and attributes will be available.

Models are divided in several packages:

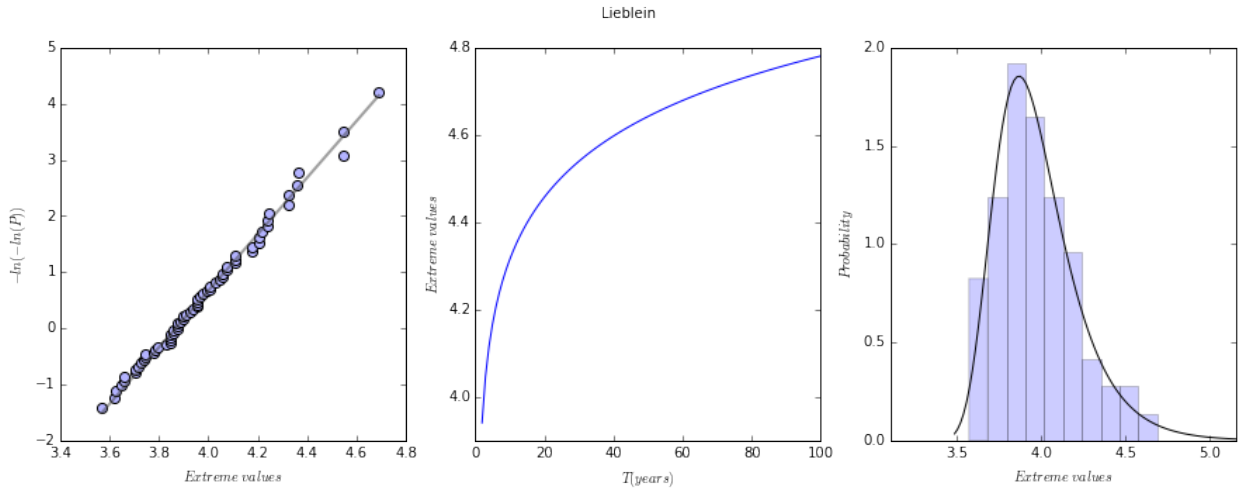
- `wind`: Some very basic functions to calculate wind extreme values obtained from the Wind Energy industry. These are very basic approximations based on parameters of wind data and should only be used to calculate the expected wind speed value for a 50 years return period.
- `engineering`: Basic models found in the literature based, mainly, on the Gumbel distribution and the Block Maxima approach.
- `classic`: A more classical approach from the theory to obtain extreme values using fitting accepted distributions generally used in the extreme value theory.

```
model = ske.models.engineering.Lieblein(sea_levels)
```

Once the model is fitted you can see the results:

```
model.plot_summary()
```

```
(<matplotlib.figure.Figure at 0xf449d70>,
 <matplotlib.axes._subplots.AxesSubplot at 0x4632f70>,
 <matplotlib.axes._subplots.AxesSubplot at 0x466a410>,
 <matplotlib.axes._subplots.AxesSubplot at 0x4693410>)
```



To see, for instance, the parameters obtained you can use:

```
print(model.c, model.loc, model.scale)
```

```
0 3.86774852633 0.198420194778
```

Another approach would be to use a more classic model:

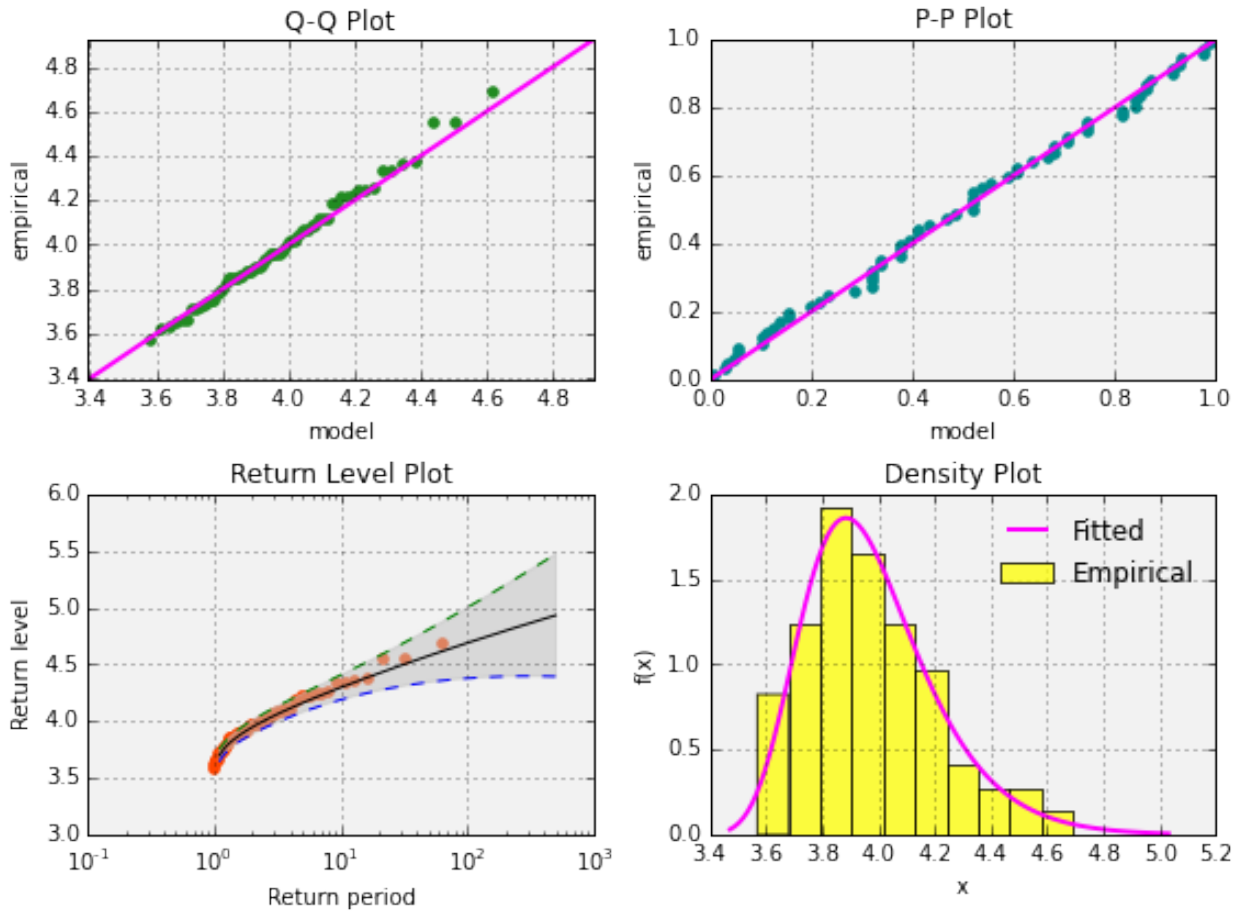
```
model = ske.models.classic.GEV(sea_levels, fit_method = 'mle', ci = 0.05,
                               ci_method = 'delta')
```

```
d:usersX003621AppDataLocalContinuumMiniconda3libsite-packagesnumdifftoolscore.py:753:␣
↪UserWarning: The stepsize (3.16814) is possibly too large!
  warnings.warn('The stepsize (%g) is possibly too large!' % h1[i])
d:usersX003621AppDataLocalContinuumMiniconda3libsite-packagesnumdifftoolscore.py:753:␣
↪UserWarning: The stepsize (0.18069) is possibly too large!
  warnings.warn('The stepsize (%g) is possibly too large!' % h1[i])
d:usersX003621AppDataLocalContinuumMiniconda3libsite-packagesnumdifftoolscore.py:753:␣
↪UserWarning: The stepsize (0.278302) is possibly too large!
  warnings.warn('The stepsize (%g) is possibly too large!' % h1[i])
d:usersX003621AppDataLocalContinuumMiniconda3libsite-packagesnumdifftoolscore.py:753:␣
↪UserWarning: The stepsize (0.0664633) is possibly too large!
  warnings.warn('The stepsize (%g) is possibly too large!' % h1[i])
```

```
model.params
```

```
OrderedDict([('shape', 0.050109518363545352),
             ('location', 3.8747498425529501),
             ('scale', 0.19804394476624812)])
```

```
model.plot_summary()
plt.show()
```



5.3 skextremes.utils

This module provides utility functions that are used within scikit-extremes that are also useful for external consumption.

`skextremes.utils.`**`bootstrap_ci`**(*data*, *statfunction=<function average>*, *alpha=0.05*, *n_samples=100*)

Given a set of data `data`, and a statistics function `statfunction` that applies to that data, computes the bootstrap confidence interval for `statfunction` on that data. Data points are assumed to be delineated by axis 0.

This function has been derived and simplified from scikits-bootstrap package created by cgevans (https://github.com/cgevans/scikits-bootstrap). All the credits shall go to him.

Parameters

data [array_like, shape (N, ...) OR tuple of array_like all with shape (N, ...)] Input data. Data points are assumed to be delineated by axis 0. Beyond this, the shape doesn't matter, so long as `statfunction` can be applied to the array. If a tuple of array_likes is passed, then samples from each array (along axis 0) are passed in order as separate parameters to the statfunction. The type of data (single array or tuple of arrays) can be explicitly specified by the multi parameter.

statfunction [function (data, weights = (weights, optional)) -> value] This function should accept samples of data from `data`. It is applied to these samples individually.

alpha [float, optional] The percentiles to use for the confidence interval (default=0.05). The returned values are (alpha/2, 1-alpha/2) percentile confidence intervals.

n_samples [int or float, optional] The number of bootstrap samples to use (default=100)

Returns

confidences [tuple of floats] The confidence percentiles specified by alpha

Calculation Methods

'pi' [Percentile Interval (Efron 13.3)] The percentile interval method simply returns the 100*alphath bootstrap sample's values for the statistic. This is an extremely simple method of confidence interval calculation. However, it has several disadvantages compared to the bias-corrected accelerated method.

If you want to use more complex calculation methods, please, see scikits-bootstrap package.

References

Efron (1993): 'An Introduction to the Bootstrap', Chapman & Hall.

`skextremes.utils.`**`gev_momfit`**`(`*`data`*`)`

Estimate parameters of Generalised Extreme Value distribution using the method of moments. The methodology has been extracted from appendix A.4 on EVA (see references below).

Parameters

data [array_like] Sample extreme data

Returns

tuple tuple with the shape, location and scale parameters. In this, case, the shape parameter is always 0.

References

DHI, (2003): 'EVA(Extreme Value Analysis - Reference manual)', DHI.

`skextremes.utils.`**`gum_momfit`**`(`*`data`*`)`

Estimate parameters of Gumbel distribution using the method of moments. The methodology has been extracted from Wilks (see references below).

Parameters

data [array_like] Sample extreme data

Returns

tuple tuple with the shape, location and scale parameters. In this, case, the shape parameter is always 0.

References

Wilks,D.S. (2006): 'Statistical Methods in the Atmospheric Sciences, second edition', Academic Press.

5.4 skextremes.models.wind

This module contains algorithms found in the literature and used extensively in wind energy engineering as standard methods.

For more information visit:

https://www.ecn.nl/publications/ECN-C–98-096

https://webstore.iec.ch/preview/info_iec61400-1%7Bed3.0%7Den.pdf

skextremes.models.wind.**wind_EWTSII_Davenport**(*vave*, *k*, *T=50*, *n=23037*)

Algorithm appeared in the European Wind Turbine Standards II (EWTS II). Davenport variation.

It uses 10-minute wind speeds to obtain the return period extreme wind speed for the T return period defined.

Parameters

vave [float or int] Long term mean wind speed

k [float or int] Weibull k parameter as defined in the wind industry. To obtain the k parameter using scipy have a look here. The *c* parameter in scipy is the k equivalent in the wind industry.

T [float or int] Return period in years. Default value is 50 (years).

n [floar or int] the number of independent events per year. Default value is 23037 for 10-min time steps and 1-yr extrema.

Returns

vref [float] Expected extreme wind speed at the return period defined.

References

Dekker JWM, Pierik JTG (1998): 'European Wind Turbine Standards II', ECN-C-99-073, ECN Solar & Wind Energy, Netherlands.

skextremes.models.wind.**wind_EWTSII_Exact**(*vave*, *k*, *T=50*, *n=23037*)

Algorithm appeared in the European Wind Turbine Standards II (EWTS II). Exact variation.

It uses 10-minute wind speeds to obtain the return period extreme wind speed for the T return period defined.

Parameters

vave [float or int] Long term mean wind speed

k [float or int] Weibull k parameter as defined in the wind industry. To obtain the k parameter using scipy have a look here. The *c* parameter in scipy is the k equivalent in the wind industry.

T [float or int] Return period in years. Default value is 50 (years).

n [floar or int] the number of independent events per year. Default value is 23037 for 10-min time steps and 1-yr extrema.

Returns

vref [float] Expected extreme wind speed at the return period defined.

References

Dekker JWM, Pierik JTG (1998): 'European Wind Turbine Standards II', ECN-C-99-073, ECN Solar & Wind Energy, Netherlands.

skextremes.models.wind.**wind_EWTSII_Gumbel**(*vave*, *k*, *T=50*, *n=23037*)

Algorithm appeared in the European Wind Turbine Standards II (EWTS II). Gumbel variation.

It uses 10-minute wind speeds to obtain the return period extreme wind speed for the T return period defined.

Parameters

vave [float or int] Long term mean wind speed

k [float or int] Weibull k parameter as defined in the wind industry. To obtain the k parameter using scipy have a look here. The *c* parameter in scipy is the k equivalent in the wind industry.

T [float or int] Return period in years. Default value is 50 (years).

n [floar or int] the number of independent events per year. Default value is 23037 for 10-min time steps and 1-yr extrema.

Returns

vref [float] Expected extreme wind speed at the return period defined.

References

Dekker JWM, Pierik JTG (1998): 'European Wind Turbine Standards II', ECN-C-99-073, ECN Solar & Wind Energy, Netherlands.

`skextremes.models.wind.`**`wind_vref_5vave`**(*vave*, *factor=5*)

It calculates the 50 year return expected maximum wind speed as 5 times the long term average wind speed.

It uses 10-minute wind speeds to obtain the 50-year return period extreme wind speed.

Parameters

vave [float or int] Long term mean wind speed

factor [float or int] Factor used to obtain vref. Default value is 5.

Returns

vref [float] vref wind speed, i.e., 50 years expected maximum wind speed in the same units used by the vave input parameter.

5.5 skextremes.models.engineering

This module contains algorithms found in the literature and used extensively in some fields.

The following paragraphs have been adapted from Makonnen, 2006

The return period of an event of a specific large magnitude is of fundamental interest. All evaluations of the risks of extreme events require methods to statistically estimate their return periods from the measured data. Such methods are widely used in building codes and regulations concerning the design of structures and community planning, as examples. Furthermore, it is crucial for the safety and economically optimized engineering of future communities to be able to estimate the changes in the frequency of various natural hazards with climatic change, and analyzing trends in the weather extremes.

The return period R (in years) of an event is related to the probability P of not exceeding this event in one year by

$$R = \frac{1}{1 - P}$$

A standard method to estimate R from measured data is the following. One first ranks the data, typically annual extremes or values over a threshold, in increasing order of magnitude from the smallest $m = 1$ to the largest $m = N$ and associates a cumulative probability P to each of the mth smallest values. Second, one fits a line to the ranked values by some fitting procedure. Third, one interpolates or extrapolates from the graph so that the return period of the extreme value of interest is estimated.

Basically, this extreme value analysis method, introduced by Hazen (1914), can be applied directly by using arithmetic paper. However, interpolation and extrapolation can be made more easily when the points fall on a straight line, which is rarely the case in an order-ranked plot of a physical variable on arithmetic paper. Therefore, almost invariably, the analysis is made by modifying the scale of the probability P, and sometimes also that of the random variable x, in such a way that the plot against x of the anticipated cumulative distribution function $P = F(x)$ of the variable appears as a straight line. Typically, the Gumbel probability paper (Gumbel 1958) is used because in many cases the distribution of the extremes, each selected from r events, asymptotically approaches the Gumbel distribution when r goes to infinity.

class skextremes.models.engineering.**Harris1996**(*data=None*, *ppp='Harris1996'*, ***kwargs*)

Calculate extreme values based on yearly maxima using Harris1996 plotting positions and a least square fit.

This methodology differ from others in the module in the location of the probability plotting position.

Parameters

data [array_like] Extreme values dataset.

preconditioning [int or float] You can choose to apply an exponent to the extreme data values before performing the Gumbel curve fit. Preconditioning can often improve the convergence of the curve fit and therefore improve the estimate T-year extreme wind speed. Default value is 1.

Attributes

results [dict] A dictionary containing different parameters of the fit.

c [float] Value of the 'shape' parameter. In the case of the Gumbel distribution this value is always 0.

loc [float] Value of the 'localization' parameter.

scale [float] Value os the 'scale' parameter.

distr [frozen `scipy.stats.gumbel_r` distribution] Frozen distribution of type `scipy.stats.gumbel_r` with `c`, `loc` and `scale` parameters equal to `self.c`, `self.loc` and `self.scale`, respectively.

Methods

Methods to calculate the fit:

_ppp_harris1996

Methods to plot results:

self.plot_summary()

_ppp_harris1996()

Review of the traditional Gumbel extreme value method for analysing yearly maximum windspeeds or similar data, with a view to improving the process. An improved set of plotting positions based on the mean values of the order statistics are derived, together with a means of obtaining the standard deviation of each position. This enables a fitting procedure using weighted least squares to be adopted, which gives results similar to the traditional Lieblein BLUE process, but with the advantages that it does not require tabulated coefficients, is available for any number of data up to at least 50, and provides a quantitative measure of goodness of fit.

References

Harris RI, (1996), 'Gumbel re-visited – a new look at extreme value statistics applied to wind speeds', Journal of Wind Engineering and Industrial Aerodynamics, 59, 1-22.

plot_summary()

Summary plot including PP plot, QQ plot, empirical and fitted pdf and return values and periods.

Returns

4-panel plot including PP, QQ, pdf and return level plots

class skextremes.models.engineering.**Lieblein**(*data=None*, *ppp='Lieblein'*, ***kwargs*)

Calculate extreme values based on yearly maxima using Lieblein plotting positions and a least square fit.

This methodology differ from others in the module in the location of the probability plotting position.

Parameters

data [array_like] Extreme values dataset.

preconditioning [int or float] You can choose to apply an exponent to the extreme data values before performing the Gumbel curve fit. Preconditioning can often improve the convergence of the curve fit and therefore improve the estimate T-year extreme wind speed. Default value is 1.

Attributes

results [dict] A dictionary containing different parameters of the fit.

c [float] Value of the 'shape' parameter. In the case of the Gumbel distribution this value is always 0.

loc [float] Value of the 'localization' parameter.

scale [float] Value os the 'scale' parameter.

distr [frozen `scipy.stats.gumbel_r` distribution] Frozen distribution of type `scipy.stats.gumbel_r` with `c`, `loc` and `scale` parameters equal to `self.c`, `self.loc` and `self.scale`, respectively.

Methods

Methods to calculate the fit:

_ppp_lieblein

Methods to plot results:

self.plot_summary()

_ppp_lieblein()

Lieblein-BLUE (Best Linear Unbiased Estimator) to obtain extreme values using a Type I (Gumbel) extreme value distribution.

It approaches the calculation of extremes using a very classical methodology provided by Julius Lieblein. It exists just to check how several consultants made the calculation of wind speed extremes in the wind energy industry.

It calculates extremes using an adjustment of Gumbel distribution using least squares fit and considering several probability-plotting positions used in the wild.

References

Lieblein J, (1974), 'Efficient methods of Extreme-Value Methodology', NBSIR 74-602, National Bureau of Standards, U.S. Department of Commerce.

plot_summary()

Summary plot including PP plot, QQ plot, empirical and fitted pdf and return values and periods.

Returns

4-panel plot including PP, QQ, pdf and return level plots

class skextremes.models.engineering.**PPPLiterature**(*data=None*, *ppp='Weibull'*, ***kwargs*)

Calculate extreme values based on yearly maxima using several plotting positions and a least square fit.

This methodology differ from others in the module in the location of the probability plotting position.

Parameters

data [array_like] Extreme values dataset.

preconditioning [int or float] You can choose to apply an exponent to the extreme data values before performing the Gumbel curve fit. Preconditioning can often improve the convergence of the curve fit and therefore improve the estimate T-year extreme wind speed. Default value is 1.

Attributes

results [dict] A dictionary containing different parameters of the fit.

c [float] Value of the 'shape' parameter. In the case of the Gumbel distribution this value is always 0.

loc [float] Value of the 'localization' parameter.

scale [float] Value os the 'scale' parameter.

distr [frozen `scipy.stats.gumbel_r` distribution] Frozen distribution of type `scipy.stats.gumbel_r` with `c`, `loc` and `scale` parameters equal to `self.c`, `self.loc` and `self.scale`, respectively.

Methods

Methods to calculate the fit:

_ppp_adamowski

_ppp_beard

_ppp_blom

_ppp_gringorten

_ppp_hazen

_ppp_hirsch

_ppp_iec56

_ppp_landwehr

_ppp_laplace

_ppp_mm

_ppp_tukey

_ppp_weibull

Methods to plot results:

self.plot_summary()

_ppp_adamowski()

Perform the calculations using the Adamowski method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.25}{N + 0.5}$$

References

De, M., 2000. A new unbiased plotting position formula for gumbel distribution. Stochastic Envir. Res. Risk Asses., 14: 1-7.

_ppp_beard()

Perform the calculations using the Beard method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.31}{N + 0.38}$$

References

De, M., 2000. A new unbiased plotting position formula for gumbel distribution. Stochastic Envir. Res. Risk Asses., 14: 1-7.

_ppp_blom()

Perform the calculations using the Blom method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.375}{N + 0.25}$$

References

De, M., 2000. A new unbiased plotting position formula for gumbel distribution. Stochastic Envir. Res. Risk Asses., 14: 1-7.

_ppp_gringorten()

Perform the calculations using the Gringorten method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.44}{N + 0.12}$$

References

Adeboye, O.B. and M.O. Alatise, 2007. Performance of probability distributions and plotting positions in estimating the flood of River Osun at Apoje Sub-basin, Nigeria. Agric. Eng. Int.: CIGR J., Vol. 9.

_ppp_hazen()

Perform the calculations using the Hazen method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.5}{N}$$

References

Adeboye, O.B. and M.O. Alatise, 2007. Performance of probability distributions and plotting positions in estimating the flood of River Osun at Apoje Sub-basin, Nigeria. Agric. Eng. Int.: CIGR J., Vol. 9.

_ppp_hirsch()

Perform the calculations using the Hirsch method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) + 0.5}{N + 1}$$

References

Jay, R.L., O. Kalman and M. Jenkins, 1998. Integrated planning and management for Urban water supplies considering multi uncertainties. Technical Report, Department of Civil and Environmental Engineering, Universities of California.

_ppp_iec56()

Perform the calculations using the IEC56 method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.5}{N + 0.25}$$

References

Forthegill, J.C., 1990. Estimating the cumulative probability of failure data points to be plotted on weibull and other probability paper. Electr. Insulation Transact., 25: 489-492.

`_ppp_landwehr()`

Perform the calculations using the Landwehr method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.35}{N}$$

References

Makkonen, L., 2008. Problem in the extreme value analysis. Structural Safety, 30: 405-419.

`_ppp_laplace()`

Perform the calculations using the Laplace method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) + 1}{N+2}$$

References

Jay, R.L., O. Kalman and M. Jenkins, 1998. Integrated planning and management for Urban water supplies considering multi uncertainties. Technical Report, Department of Civil and Environmental Engineering, Universities of California.

`_ppp_mm()`

Perform the calculations using the McClung and Mears method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 0.4}{N}$$

References

Makkonen, L., 2008. Problem in the extreme value analysis. Structural Safety, 30: 405-419.

`_ppp_tukey()`

Perform the calculations using the Tukey method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) - 1/3}{N + 1/3}$$

References

Makkonen, L., 2008. Problem in the extreme value analysis. Structural Safety, 30: 405-419.

`_ppp_weibull()`

Perform the calculations using the Weibull method available for the probability positions.

Probability positions are defined as:

$$P = \frac{(N+1) + 1}{N+1}$$

References

Hynman, R.J. and Y. Fan, 1996. Sample quantiles in statistical packages. Am. Stat., 50: 361-365.

`plot_summary()`

Summary plot including PP plot, QQ plot, empirical and fitted pdf and return values and periods.

Returns

4-panel plot including PP, QQ, pdf and return level plots

5.6 skextremes.models.classic

Module containing classical generalistic models

Gumbel: To be used applying the Block Maxima approach

Generalised extreme value distribution (GEV): To be used applying the Block Maxima approach

Generalised Pareto Distribution (GPD): To be used applying the Peak-Over-Threshold approach TODO

`class skextremes.models.classic.GEV(`*data*, *fit_method='mle'*, *ci=0*, *ci_method=None*, *return_periods=None*, *frec=1*`)`

Class to fit data to a Generalised extreme value (GEV) distribution.

Parameters

data [array_like] 1D array_like with the extreme values to be considered

fit_method [str] String indicating the method used to fit the distribution. Availalable values are 'mle' (default value), 'mom' and 'lmoments'.

ci [float (optional)] Float indicating the value to be used for the calculation of the confidence interval. The returned values are (ci/2, 1-ci/2) percentile confidence intervals. E.g., a value of 0.05 will return confidence intervals at 0.025 and 0.975 percentiles.

ci_method [str (optional)] String indicating the method to be used to calculate the confidence intervals. If `ci` is not supplied this parameter will be ignored. Possible values depend of the fit method chosen. If the fit method is 'mle' possible values for ci_method are 'delta' and 'bootstrap', if the fit method is 'mom' or 'lmoments' possible value for ci_method is 'bootstrap'.

'delta' is for delta method. 'bootstrap' is for parametric bootstrap.

return_period [array_like (optional)] 1D array_like of values for the *return period*. Values indicate **years**.

frec [int or float] Value indicating the frecuency of events per year. If frec is not provided the data will be treated as yearly data (1 value per year).

Attributes and Methods

params [OrderedDict] Ordered dictionary with the values of the *shape*, *location* and *scale* parameters of the distribution.

c [flt] Float value for the *shape* parameter of the distribution.

loc [flt] Float value for the *location* parameter of the distribution.

scale [flt] Float value for the *scale* parameter of the distribution.

distr [object] Frozen RV object with the same methods of a continuous scipy distribution but holding the given *shape*, *location*, and *scale* fixed. See http://docs.scipy.org/doc/scipy/reference/stats.html for more info.

data [array_like] Input data used for the fit

fit_method [str] String indicating the method used to fit the distribution, values can be 'mle', 'mom' or 'lmoments'.

`cdf`(*quantiles*)

Cumulative distribution function of the given frozen RV.

Parameters

x [array_like] quantiles

Returns

cdf [ndarray] Cumulative distribution function evaluated at *x*

`pdf`(*quantiles*)

Probability density function at x of the given frozen RV.

Parameters

x [array_like] quantiles

Returns

pdf [ndarray] Probability density function evaluated at x

`plot_density`()

Histogram of the empirical pdf data and the pdf plot of the fitted distribution. All parameters are predefined from the frozen fitted model and empirical data available.

Returns

Density plot.

`plot_pp`()

PP (probability) plot between empirical and fitted data. All parameters are predefined from the frozen fitted model and empirical data available.

Returns

PP plot.

`plot_qq`()

QQ (Quantile-Quantile) plot between empirical and fitted data. All parameters are predefined from the frozen fitted model and empirical data available.

Returns

QQ plot.

`plot_return_values`()

Return values and return periods of data. If confidence interval information has been provided it will show the confidence interval values.

Returns

Return values and return periods plot.

`plot_summary`()

Summary plot including PP plot, QQ plot, empirical and fitted pdf and return values and periods.

Returns

4-panel plot including PP, QQ, pdf and return level plots

`ppf`(*q*)

Percent point function (inverse of cdf) at q of the given frozen RV.

Parameters

q [array_like] lower tail probability

Returns

x [array_like] quantile corresponding to the lower tail probability q.

`stats`(*moments*)

Some statistics of the given RV.

Parameters

moments [str, optional] composed of letters ['mvsk'] defining which moments to compute: 'm' = mean, 'v' = variance, 's' = (Fisher's) skew, 'k' = (Fisher's) kurtosis. (default='mv')

Returns

stats [sequence] of requested moments.

`class` skextremes.models.classic.**Gumbel**(*data*, *fit_method='mle'*, *ci=0*, *ci_method=None*, *return_periods=None*, *frec=1*)

Class to fit data to a Gumbel distribution. Note that this is a special case of the `GEV` class where the 'shape' is fixed to 0.

Parameters

data [array_like] 1D array_like with the extreme values to be considered

fit_method [str] String indicating the method used to fit the distribution. Availalable values are 'mle' (default value), 'mom' and 'lmoments'.

ci [float (optional)] Float indicating the value to be used for the calculation of the confidence interval. The returned values are (ci/2, 1-ci/2) percentile confidence intervals. E.g., a value of 0.05 will return confidence intervals at 0.025 and 0.975 percentiles.

ci_method [str (optional)] String indicating the method to be used to calculate the confidence intervals. If `ci` is not supplied this parameter will be ignored. Possible values depend of the fit method chosen. If the fit method is 'mle' possible values for ci_method are 'delta' and 'bootstrap', if the fit method is 'mom' or 'lmoments' possible value for ci_method is 'bootstrap'.

'delta' is for delta method. 'bootstrap' is for parametric bootstrap.

return_period [array_like (optional)] 1D array_like of values for the *return period*. Values indicate **years**.

frec [int or float] Value indicating the frecuency of events per year. If frec is not provided the data will be treated as yearly data (1 value per year).

Attributes and Methods

params [OrderedDict] Ordered dictionary with the values of the *shape*, *location* and *scale* parameters of the distribution.

c [flt] Float value for the *shape* parameter of the distribution.

loc [flt] Float value for the *location* parameter of the distribution.

scale [flt] Float value for the *scale* parameter of the distribution.

distr [object] Frozen RV object with the same methods of a continuous scipy distribution but holding the given *shape*, *location*, and *scale* fixed. See http://docs.scipy.org/doc/scipy/reference/stats.html for more info.

data [array_like] Input data used for the fit

fit_method [str] String indicating the method used to fit the distribution, values can be 'mle', 'mom' or 'lmoments'.

`cdf`(*quantiles*)

Cumulative distribution function of the given frozen RV.

Parameters

x [array_like] quantiles

Returns

cdf [ndarray] Cumulative distribution function evaluated at *x*

`pdf`(*quantiles*)

Probability density function at x of the given frozen RV.

Parameters

x [array_like] quantiles

Returns

pdf [ndarray] Probability density function evaluated at x

`plot_density`()

Histogram of the empirical pdf data and the pdf plot of the fitted distribution. All parameters are predefined from the frozen fitted model and empirical data available.

Returns

Density plot.

`plot_pp`()

PP (probability) plot between empirical and fitted data. All parameters are predefined from the frozen fitted model and empirical data available.

Returns

PP plot.

`plot_qq`()

QQ (Quantile-Quantile) plot between empirical and fitted data. All parameters are predefined from the frozen fitted model and empirical data available.

Returns

QQ plot.

`plot_return_values`()

Return values and return periods of data. If confidence interval information has been provided it will show the confidence interval values.

Returns

Return values and return periods plot.

`plot_summary`()

Summary plot including PP plot, QQ plot, empirical and fitted pdf and return values and periods.

Returns

4-panel plot including PP, QQ, pdf and return level plots

`ppf`(*q*)

Percent point function (inverse of cdf) at q of the given frozen RV.

Parameters

q [array_like] lower tail probability

Returns

x [array_like] quantile corresponding to the lower tail probability q.

`stats`(*moments*)

Some statistics of the given RV.

Parameters

moments [str, optional] composed of letters ['mvsk'] defining which moments to compute: 'm' = mean, 'v' = variance, 's' = (Fisher's) skew, 'k' = (Fisher's) kurtosis. (default='mv')

Returns

stats [sequence] of requested moments.

`class skextremes.models.classic.GPD`(*data*, *fit_method='mle'*, *ci=0*, *ci_method=None*, *return_periods=None*, *frec=1*)

PYTHON MODULE INDEX

S

INDEX

Symbols

B

C

G

H

L

M

P

S

W