Metatstatistical Extreme Value Analysis in Python - Version 1.01

Enrico Zorzetto

December 15, 2017

1 Introduction

This manuscript provides an overview of a Python Package For Extreme Value Analysis (in the following mevpy, Metatstatistical Extreme Value Analysis in Python). This is a set of functions that implement the Metastatistical Extreme Value (MEV) Distribution as introduced in [1] and related functions. The package was developed for application to rainfall daily data, but the method is general and can be applied to different fields. Some example of applications to rainfall data are included.

2 Theory

See e.g., [2] for GEV and [1, 3] for MEV.

3 List of functions

3.1 The gev package

1. `gev_fit`
2. `gev_fit_lmom`
3. `sample_lmom`
4. `gev_quant`
5. `gev_cdf`
6. `gev_random_quant`
7. `bootstrap`
8. `hess`

9. `gev_fit_ml`
10. `pot_fit`
11. `pot_boot`
12. `pot_quant`
13. `gpd_fit`
14. `gpd_fit_ml`
15. `gpd_fit_pwm`
16. `gpd_quant`
17. `gpd_cdf`
18. `gpd_random_quant`

3.2 The mev package

1. `wei_fit`
2. `wei_boot`
3. `wei_fit_pwm`
4. `wei_quant`
5. `wei_pdf`
6. `wei_mean_variance`
7. `wei_cdf`
8. `wei_surv`
9. `wei_random_quant`
10. `wei_fit_ls`
11. `wei_fit_mlpy`
12. `wei_fit_ml`
13. `wei_negloglike`
14. `mev_fun`
15. `mev_quant`
16. `mev_cdf`

17. `mev_fit`
18. `mev_CI`
19. `mev_boot_yearly`
20. `remove_missing_years`
21. `tab_rain_max`
22. `table_rainfall_maxima`
23. `fit_EV_models`
24. `shuffle_mat`
25. `cross_validation`
26. `slideover`

4 The gev package

In this package we implement some basic functions which allow to calibrate and validate traditional extreme value techniques based on the Generalized Extreme Value distribution [2] and on the Peak Over Threshold (POT) [4] method.

```
gev_fit(sample, how = 'lmom', std = False,
                      std_how = 'hess', std_num = 1000)
```

Returns:

1. `parhat, parstd, varcov` (if `std = True`)
2. `parhat` (if `std = False`)

where:

- `parhat`: tuple with the three estimated GEV parameters $(\hat{\xi}, \hat{\psi}, \hat{\mu})$
- `parstd`: tuple with the three estimated GEV parameter standard deviations $(\sigma^2_{\hat{\xi}}, \sigma^2_{\hat{\psi}}, \sigma^2_{\hat{\mu}})$
- `varcov`: 3x3 covariance matrix of the estimated parameters, in the same order $(\hat{\xi}, \hat{\psi}, \hat{\mu})$.

Arguments:

- `sample`: sample (e.g., of annual maxima) to which the fit of GEV is performed (numpy array)

Optional arguments:

- `how`: Fitting method for GEV. Possible options are `'lmom'` for fitting with L-moments method [5] or `'ml'` for maximum likelihood [2]. Default is `'lmom'`.
- `std`: If `'True'`, covariance matrix of the parameters is calculated. Default is 'False'.
- `std_how`: Method for computing covariance matrix. Possible choices are `'hess'` for numeric Hessian or `'boot'` for nonparametric bootstrap. Default is `'hess'`. This is needed only if `std = True`.
- `std_num`: Number or resampling for the bootstrap. Needed only if `std = True` and `std_now = 'boot'`. Default is `std_num = 1000` generations.

```
gev_fit_lmom(sample)
```

Returns: `parhat = (csi, psi, mu)`, tuple with GEV parameters estimated by means of L-moments method [5].
Arguments: `sample`, array of observations (e.g., annual maxima)

```
sample_lmom(sample)
```

Returns: `L1, L2, t3`, the first two L-moments and the ratio `t3 = L3/L2`.
Arguments: `sample`, array of observations.

```
gev_quant(Fi, csi, psi, mu, ci = False, varcov = [])
```

Returns:

1. `q, qu, ql` (if `ci = True`) Respectively the GEV quantile, upper and lower bounds corresponding to 95% confidence intervals.
2. `q` (if `ci = False`) GEV quantile

Arguments:

- `Fi`: Non exceedance probability for which we want to estimate the GEV quantile
- `csi, psi, mu`: GEV parameters ($csi > 0$ for Frechet heavy tailed distribution)

Optional arguments:

- `ci`: If `True`, confidence intervals are computed. Default is `False`.
- `varcov`: covariance matrix of GEV parameters, in the order `(csi, psi, mu)`. Needed only if `ci = True`, by default is empty.

```
gev_cdf(q, csi, psi, mu)
```

Returns: `Fhi`, GEV non exceedance probability
Arguments: `q, csi, psi, mu`: value q for which non exceedence probability must be computed and set of GEV parameters (shape, scale, location).

```
gev_random_quant(length, csi, psi, mu)
```

Returns: `xi`, sample generated from a GEV distribution
Arguments: `length, csi, psi, mu`: desired dimension of the sample and set of GEV parameters (shape, scale, location).

```
def bootstrap(sample, fitfun, npar = 3, ntimes = 1000)
```

Returns: `parstd, varcov`, array with the standard deviations for the parameters and their covariance matrix for some distribution obtained by non-parametric bootstrap.
Arguments:

- `sample`: sample of observations.
- `fitfun`: function for fitting the desired distribution to the bootstrap-generated samples.

Arguments:

- `npar`: number of parameters of the distribution fitted to the samples. Default is 3 for GEV, but must be modified if necessary (e.g., for GPD).

- `ntimes`: Number of resampling with replacement from the original sample. Default is 1000. The longer the better, if you like waiting.

```
hess(fun, y, data)
```

Returns: `hessian`, numeric hessian matrix (from EVIM package)
Arguments:

- `fun`: neg-log-likelihood function of which I want to evaluate the hessian matrix
- `y`: point in the parameter space at which I want to evaluate the hessian.
- `data`: sample for evaluating the likelihood function.

```
gev_fit_ml(data, std = False)
```

Returns:

1. `parhat, parstd, varcov` (if `std = True`) Respectively GEV ML-estimated parameters (csi, psi, mu), their standard deviations and covariance matrix (obtained evaluating numerically the hessian matrix).
2. `parhat` (if `std = False`) GEV ML-estimated parameters (csi, psi, mu)

Arguments:

- `data`: sample for GEV fit (e.g., of annual maxima of some process)
- `std`: if `True`, the covariance matrix is computed as well. Default is `False`.

```
pot_fit(data, datatype = 'df', way = 'ea', ea = 3, sp = 0.1, thresh = 10,
                how = 'ml', std = False, std_how = 'hess', std_num = 1000)
```

Returns:

1. `parhat, parpot, parstd, varcov` (if `std = True`)
2. `parhat` (if `std = False`)

where:

- `parhat` GEV estimated parameters (csi, psi, mu)
- `parpot` POT parametes (csi, beta, lambda), respectively GPD shape and scale parameter and Poisson arrival rate
- `parstd` Standard deviation of the POT parameters
- `varcov` Covariance matrix of the POT parameters

Arguments:

- `data`: daily rainfall data. Can be in a dataframe df or a 2d array

Optional arguments:

- `datatype`: Format of input data. If 'df' (default), input data are stored in a dataframe (with required fields YEAR, PRCP) If 'mat', they are stored in a 2d array with size nyears*ndays (num.of blocks) * (block size)
- `way`: Way for selecting the threshold. If 'ea' a threshold is selected such that a fixed average number of events/year exceedes it (default). If 'thresh' the threshold is specified directly. If 'sp'the threshold is determined from a given survival probability of being above threshold.
- `ea`: average number of threshold exceedances/block (only for way = 'ea'). Default is 3.
- `thresh`: Given value of the threshold (only for way = 'thresh'). Default is 10.
- `sp`: Probability of being above threshold (only for way = 'sp'). Default is 0.1.
- `how`: Method for fitting Generalized Pareto Distribution (GPD) to threshold exceedances. Default is 'ml' for maximum likelihood. Alternatively, 'pwm' for probability weighted moments.
- `std`: If True, compute parameters covariance matrix. Default is False.
- `std_how`: Method for computing parameter covariance matrix. Only required if std = True. Default is 'hess' for using the numeric hessian matrix. Alternatively, 'boot' for non-parametric bootstrap.
- `std_num`: Number of bootstrap resampling for computing parameter variances and covariances. Default is 1000. Only required for `std_how = 'boot'`.

```
pot_boot(sample, var_lambda, GPD_how = 'ml', ntimes = 1000)
```

Non parametric bootstrap procedure for Peak Over Threshold.
Returns:

- `parstd`: Array with standard deviations of the POT parameters $\left(\sigma^2_{\hat{\lambda}}, \sigma^2_{\hat{\xi}}, \sigma^2_{\hat{\beta}}\right)$, respectively Poisson rate λ, Generalized Pareto shape ξ, and scale β.
- `varcov`: covariance matrix of the same three parameters in the same order. Covariances between λ and the GPD parameters are assumed to be zero.

Arguments:

- `sample`: sample of excesses over threshold for fitting GPD
- `var_lambda`: Variance of the Poisson arrival rate λ assumed to be drawn from a Binomial distribution.

Additional Arguments:

- `GPD_how`: Method for fitting GPD. By default is equal to `'ml'` for maximum likelihood estimator. Alternatively, Probability Weighted Moments method (`'pwm'`) is available.
- `ntimes`: Number of samples generated in the bootstrap procedure. Default is 1000.

```
pot_quant(Fi, csi, psi, mu, ci = False, parpot = [], varcov = [])
```

Returns:

1. `q, qu, ql` (if `ci = True`) Respectively the POT-GEV quantile, upper and lower bounds corresponding to 95% confidence intervals.
2. `q` (if `ci = False`) POT-GEV quantile

Arguments:

- `Fi`: Non exceedance probability for which we want to estimate the POT-GEV quantile
- `csi, psi, mu`: GEV parameters ($csi > 0$ for Frechet heavy tailed distribution) estimated by means of the POT approach.

Optional arguments:

- `ci`: If `True`, confidence intervals are computed. Default is `False`.
- `parpot`: array of POT parameters (GPD shape and scale, and Poisson rate), in the order `(csi, beta, lambda)`. Needed only if `ci = True`, by default is empty.
- `varcov`: covariance matrix of POT parameters in the same order (GPD shape and scale, and Poisson rate) `(csi, beta, lambda)`. Needed only if `ci = True`, by default is empty.

```
gpd_fit(sample, how = 'ml', std = False, std_how = 'hess', std_num = 1000)
```

Returns:

1. `parhat, parstd, varcov` (if `std = True`)
2. `parhat` (if `std = False`)

where:

- `parhat`: tuple with the two estimated GPD parameters $(\hat{\xi}, \hat{\beta})$
- `parstd`: tuple with the two estimated GPD parameter standard deviations $(\sigma^2_{\hat{\xi}}, \sigma^2_{\hat{\beta}})$
- `varcov`: 2x2 covariance matrix of the GPD estimated parameters, in the same order $(\hat{\xi}, \hat{\beta})$.

Arguments:

- `sample`: sample (e.g., of excesses over a threshold) to which the fit of GEV is performed (numpy array)

Optional arguments:

- `how`: Fitting method for GEV. Possible options are `'pwm'` for fitting with Probability Weighted Moments method, or `'ml'` for maximum likelihood [2]. Default is `'ml'`.
- `std`: If `'True'`, covariance matrix of the parameters is calculated. Default is 'False'.
- `std_how`: Method for computing covariance matrix. Possible choices are `'hess'` for numeric Hessian or `'boot'` for nonparametric bootstrap. Default is `'hess'`. This is needed only if `std = True`.

- `std_num`: Number or resampling for the bootstrap. Needed only if `std = True` and `std_now = 'boot'`. Default is `std_num = 1000` generations.

```
gpd_fit_ml(data, std = False)
```

Returns:

1. `parhat, parstd, varcov` (if `std = True`) Respectively GPD ML-estimated parameters (csi, beta), their standard deviations and covariance matrix (obtained evaluating numerically the hessian matrix).
2. `parhat` (if `std = False`) GPD ML-estimated parameters (csi, beta)

Arguments:

- `data`: sample for GPD fit.
- `std`: if `True`, the covariance matrix is computed as well. Default is `False`.

```
gpd_fit_pwm(sample, threshold = 0)
```

Returns: `csi, beta`: GPD estimated shape and scale parameters. See Hosking and Wallis, 1987. Arguments:

- `sample`: sample for GPD fit.
- `threshold`: Optional threshold to ignore data below a certain level. It does not renormalize the distribution. Default is zero.

```
gpd_quant(Fi, csi, beta, mu = 0, ci = False, varcov = [])
```

Returns:

1. `q, qu, ql` (if `ci = True`) Respectively the GPD quantile, upper and lower bounds corresponding to 95% confidence intervals.
2. `q` (if `ci = False`) GPD quantile

Arguments:

- `Fi`: Non exceedance probability for which we want to estimate the GPD quantile
- `csi, beta, mu`: GPD parameters: shape ($csi > 0$ for Heavy tailed distribution), scale (beta) and location (mu). By default location is set to zero (optional).

Optional arguments:

- `ci`: If `True`, confidence intervals are computed. Default is `False`.
- `varcov`: 2x2 covariance matrix of GPD parameters, in the order `(csi, beta)`. Needed only if `ci = True`, by default is empty. Does not include location.

```
gpd_cdf(q, csi, beta, mu = 0)
```

Returns: `Fhi`, GD non exceedance probability
Arguments: `q, csi, beta, mu`: value q for which GPD non exceedence probability must be computed and set of GPD parameters (shape, scale, location). By default location is zero (optional).

```
gpd_random_quant(length, csi, beta, mu = 0)
```

Returns: `xi`, sample generated from a GPD distribution
Arguments: `length, csi, beta, mu`: desired dimension of the sample and set of GPD parameters (shape, scale, location). By default location is zero (optional).

5 The mev package

Here we implement the basic functions for the calibration and validation of the MEV distribution [1, 3]

```
wei_fit(sample, how = 'pwm', threshold = 0, std = False,
                        std_how = 'boot', std_num = 1000)
```

Returns:

1. `parhat, parstd, varcov` (if `std = True`)

2. `parhat` (if `std = False`)

where:

- `parhat`: tuple with the 2 estimated WEI parameters $(N, \hat{C}, \hat{w})$: number of positive values in the sample (e.g., wet days), shape and scale WEI parameters.
- `parstd`: tuple with the 2 estimated WEI parameter standard deviations $(\sigma^2_{\hat{C}}, \sigma^2_{\hat{w}})$
- `varcov`: 2x2 covariance matrix of the estimated parameters, in the same order $(\hat{C}, \hat{w})$.

Arguments:

- `sample`: sample to which the fit of WEI is performed (numpy array)

Optional arguments:

- `how`: Fitting method for WEI. Possible options are `'pwm'` for fitting with Probability Weighted Moment method [6] or `'ml'` for maximum likelihood, or `'ls'` for least squares. Default is `'pwm'`.
- `threshold`: Optional threshold to censor small values, without accounting for their mass of probability. Default is zero.
- `std`: If `True`, covariance matrix of the parameters is calculated. Default is `False`.
- `std_how`: Method for computing covariance matrix. Possible choices are `'hess'` for numeric Hessian or `'boot'` for nonparametric bootstrap. Default is `'boot'`. This is needed only if `std = True`.
- `std_num`: Number or samples generated for the bootstrap. Needed only if `std = True` and `std_how = 'boot'`. Default is `std_num = 1000` generations.

```
wei_boot(sample, fitfun, npar = 2, ntimes = 1000)
```

Returns: `parstd, varcov`, array with the standard deviations for the parameters of the WEI (C,w) and their covariance matrix for some distribution (WEI here) obtained by non-parametric bootstrap.
Arguments:

- `sample`: sample of observations.

- `fitfun`: any function for fitting the desired distribution to the bootstrap-generated samples.

Arguments:

- `npar`: number of parameters of the distribution fitted to the samples. Default is 2 for WEI, but can be modified if necessary for others distributions.
- `ntimes`: Number of resampling with replacement from the original sample. Default is 1000.

```
wei_fit_pwm(sample, threshold = 0)
```

Returns: `n,C,w`: Number of events > 0 in the fitting sample and PWM-estimated parameters scale and shape (C,w) of the WEI distribution. Arguments:

- `sample`: sample to which the fit of WEI is performed (numpy array)

Optional arguments:

- `threshold`: Optional threshold to censor small values, without accounting for their mass of probability. Default is zero.

```
wei_quant(Fi, C, w, ci = False, varcov = [])
```

Returns:

1. `q, qu, ql` (if `ci = True`) Respectively the WEI quantile, and upper and lower bounds corresponding to 95% confidence intervals.
2. `q` (if `ci = False`) WEI quantile

Arguments:

- `Fi`: Non exceedance probability for which we want to estimate the WEI quantile
- `C,w`: WEI parameters (scale and shape)

Optional arguments:

- `ci`: If `True`, confidence intervals are computed. Default is `False`.

- `varcov`: covariance matrix of WEI parameters, in the order `(C,w)`. Needed only if `ci = True`, by default is empty.

`wei_pdf(x,C,W)`

Returns: `pdf`, WEI probability density function (scalar or array depending on the input `x`)
Arguments:

- `x`: Point at which the WEI pdf is evaluated. Can be a scalar or a list/array, and returns an output of the same type.
- `C,w`: Scale and shape parameters of the WEI distribution

`wei_mean_variance(C,w)`

Returns: `mu, var`: Theoretical mean and average of the WEI distribution
Arguments: `C,w`: Scale and shape parameters of the WEI distribution

---

`wei_cdf(q, C, w)`

Returns: `Fhi`, WEI non exceedance probability (scalar or array depending on the input `q`)
Arguments:

- `q`: Point at which the WEI non exceedance probability is evaluated. Can be a scalar or a list/array, and returns an output of the same type.
- `C,w`: Scale and shape parameters of the WEI distribution

`wei_surv(q, C, w)`

Returns: `Shi`, WEI exceedance (or survival) probability (scalar or array depending on the input `q`)
Arguments:

- `q`: Point at which the WEI exceedance probability is evaluated. Can be a scalar or a list/array, and returns an output of the same type.

- `C,w`: Scale and shape parameters of the WEI distribution

`wei_random_quant(length,C,w)`

Returns: `xi`, sample generated from a WEI distribution. It returns a scalar if `length = 1`, an array if `length > 1`
Arguments: `length, C, w`: desired dimension of the sample and set of WEI parameters (scale C and shape w).

`wei_fit_ls(sample)`

Returns: `n,C,w`: Number of events > 0 in the fitting sample and Least Squares-estimated parameters scale and shape (C,w) of the WEI distribution. Arguments: `sample`: sample to which the fit of WEI is performed (numpy array)

`wei_fit_mlpy(sample)`

Returns: `n,C,w`: Number of events > 0 in the fitting sample and Maximum Likelihood-estimated parameters scale and shape (C,w) of the WEI distribution. This function uses the built-in function from scipy.
Arguments: `sample`: sample to which the fit of WEI is performed (numpy array)

`wei_fit_ml(sample, std = False)`

Returns:

- `n,C,w`
  (if `std` = `False`)
- `n,C,w,parhat, varcov`
  (if `std` = `True`)

where:

- `n,C,w`: Number of events > 0 in the fitting sample and Maximum Likelihood-estimated parameters scale and shape (C, w) of the WEI distribution.

- `n,C,w`: array with the standard deviations of WEI scale and shape parameters.
- `varcov`: covariance matrix 2x2 of the WEI parameters (in the same order scale and shape)

Arguments:

- `sample`: sample to which the fit of WEI is performed (numpy array)
- `std`: (optional) If `True`, computes the standard deviations and covariances of the WEI parameters using the numeric hessian. Default is `False`.

`wei_negloglike(parhat, data)`

Returns: `nll`: WEI Negloglikelihood function evaluated at the point (C, w) in the WEI parameter space for the given sample.
Arguments:

- `parhat`: Array with the 2 parameters of the WEI distribution, in the order (C, w) scale and shape respectively.
- `data`: sample used to evaluate the likelihood

`mev_fun(y,pr,N,C,W)`

Returns: `mev0f`: Difference from true non exceedance probability and MEV estimate, to minimize numerically for computing quantiles.
Arguments:

- `y`: quantile
- `pr`: non exceedance probability
- `N,C,w`: Arrays of yearly- (or more generally block-) values of wet events N and Scale C and shape w parameters of the WEI distribution

---

`mev_quant(Fi,x0,N,C,W)`

Returns: `quant`: MEV quantile obtained by numerical minimization of the function `mev_fun(y,pr,N,C,W)` for a given value of non exceedance probability `pr`.
Arguments:

- `Fi`: non exceedance probability
- `x0`: initial guess for the solution of the nonlinear equation. Must be a valid one. ADD MULTIPLE OPTIONS SHOULD THE MINIMIZATION NOT CONVERGE SOMETIMES
- `N,C,w`: Arrays of yearly- (or more generally block-) values of wet events N and Scale C and shape w parameters of the WEI distribution

---

`mev_cdf(quant,N,C,W)`

Returns: `mev_cdf`: MEV non exceedance probability computed for a given quantile `quant`.
Arguments:

- `quant`: quantile for which the non exceedance probability is computed.
- `N,C,w`: Arrays of yearly- (or more generally block-) values of wet events N and Scale C and shape w parameters of the WEI distribution

---

`mev_fit(df, ws = 1, how = 'pwm', threshold = 0)`

Returns: `N,C,w`: Arrays of yearly- (or more generally block-) values of wet events N and Scale C and shape w parameters of the WEI distribution
Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)

Optional arguments:

- `ws`: Dimension of the blocks in which the WEI distributions are fitted. Default is 1 year.
- `how`: Fitting method for the WEI. Default is `'pwm'` for Probability Weighted Moments. Maximum Likelihood (`'ml'`) and Least-Squares (`'ls'`) are also available.
- `threshold`: Threshold for fitting WEI censoring the values below threshold, without accounting for the relative mass of probability. Default is zero.

```
mev_CI(df, Fi_val, x0, ws = 1, ntimes = 1000,
     MEV_how = 'pwm', MEV_thresh = 0.0, std_how = 'boot')
```

Confidence intervals for MEV estimated quantiles. CIs are obtained by means of a bootstrap procedure in which each year in the record is resampled separately, so that the observed inter-annual variability is preserved. The procedure resamples both zeros and non zeros values in the sample, so that both their intensity distribution and number of wet events changes in general.
Returns: `Q_est, Q_up, Q_low`: MEV Estimated quantile, upper and lower bounds of the 95% confidence interval. They can be scalar or arrays depending on the dimension of `Fi_val`
Arguments:

- `Fi_val`: Values of non exceedance probability for which quantiles and confidence intervals are evaluated. Can be scalar or array, and outputs have the same type.
- `x0`: initial guess for the numerical minimization procedure for finding MEV quantiles.

Optional arguments:

- `ws`: Dimension of the blocks in which the WEI distributions are fitted. Default is 1 year.
- `ntimes`: Number of resampling with replacement from the original sample. Default is 1000.
- `MEV_how`: Fitting method for the WEI. Default is `'pwm'` for Probability Weighted Moments. Maximum Likelihood (`'ml'`) and Least-Squares (`'ls'`) are also available.
- `MEV_thresh`: Threshold for fitting WEI censoring the values below threshold, without accounting for the relative mass of probability. Default is zero.
- `std_how`: Type of confidence intervals. If equal to `'boot'` the bootstrap procedure is repeated `ntimes`, MEV quantiles are computed from each generation and then their standard deviation is computed. Then, 95% CIs are obtained under the hypothesis of normal distribution of MEV quantiles around the estimated value $\pm 1.95\sigma$ (default). If `std_how` is equal to `'boot_cdf'`, the assumption of normal distribution is not made and the observed distribution of quantiles is used to evaluate the 5% and 95% limits. The band obtained in this way need not be centered at the MEV estimated value.

`mev_boot(df)`

Non parametric bootstrap procedure for MEV (not used to date as it destroys interannual variability of observed samples).
Returns: `dfr`: New data frame obtained from the original one by means of a bootstrapping procedure in which i) the number of wet days for each year is sample (with replacement) from the ones observed the sample, and then ii) the same for the events in each year (are sampled with replacement from all the non-zero values in the record). Then for each generation the MEV quantile is computed and upper and lower bounds are obtained under the hypothesis of normal distribution.
Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)

`mev_boot_yearly(df)`

Non parametric bootstrap procedure for MEV (preferred).
Returns: `dfr`: New data frame obtained from the original one by means of a bootstrapping procedure in which the data (zeros and non zeros) in each year of record are separately resampled with resubstitution.
Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)

`remove_missing_years(df,nmin)`

Removes years in the dataset with too many missing data, may them, be not appearing in the record, negative (for rainfall, -9999) and infinite or nans.
Returns:

- `df`: New dataset obtained after the removal of years with too few data.
- `nyears2`: number of years in the record after the removal.

- `nyears1`: original number of years in the records.

Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)
- `nmin`: Minimum number of events missing from a year of record for which the year gets removed from the dataset. For example if nmin = 36, we remove all years with 10% or more of missing data.

```
tab_rain_max(df)
```

Returns:

- `Xi`: arrays with the annual maxima of the dataset in `df`, sorted in ascending order
- `Fi`: Non exceedance frequency for the values in `Xi`, obtained with the Weibull plotting position formula.
- `TR`: Observed Return Times corresponding to the annua maxima in `Xi` and to their non exceedance frequencies in `Fi`.

Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)

```
table_rainfall_maxima(df, how = 'pwm', thresh = 0 )
```

Returns:

- `Xi`: arrays with the annual maxima of the dataset in `df`, sorted in ascending order
- `Fi`: Non exceedance frequency for the values in `Xi`, obtained with the Weibull plotting position formula.
- `TR`: Observed Return Times corresponding to the annua maxima in `Xi` and to their non exceedance frequencies in `Fi`.

- `NCW`: Array of shape (nyears, 3) with the values of N, C, and W estimated for each of the years in the dataset. the three variables N, C, and W are stored respectively in the 0th, 1st and 2nd columns (outer dim.) of the array NCW. Each row (inner dimension) corresponds to a yearly value of N,C, and W.

Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)

Additional Arguments:

- `how`: Fitting method for the WEI. Default is `'pwm'` for Probability Weighted Moments. Maximum Likelihood (`'ml'`) and Least-Squares (`'ls'`) are also available.
- `thresh`: Threshold for fitting WEI censoring the values below threshold, without accounting for the relative mass of probability. Default is zero.

```
fit_EV_models(df, tr_min = 5, ws = 1, GEV_how = 'lmom', MEV_how = 'pwm',
              MEV_thresh = 0, POT_way = 'ea', POT_val = 3, POT_how = 'ml',
              ci = False, ntimes = 1000, std_how_MEV = 'hess',
              std_how_GEV = 'hess', std_how_POT = 'hess', rmy = 36)
```

Fit the Extreme Values Models GEV, POT, and GEV to a dataset stored in a dataframe df.
Returns:

- `TR_val, XI_val, Fi_val, QM, QG, QP, QuM, QuG, QuP, QlM, QlG, QlP, FhM, FhG, FhP` (if `ci = True`)

- `TR_val, XI_val, Fi_val, QM, QG, QP, FhM, FhG, FhP` (if `ci = False`)

where:

- `TR_val, XI_val, Fi_val`: Empirical return times, Annual Maxima, and non exceedance frequency (Weibull plotting position) corresponding to quantiles computed with MEV, GEV, and POT.

- `QM, QG, QP`: Arrays of quantiles estimated with MEV, GEV, and POT respectively (same size of `TR_val`)

- `QM, QG, QP`: Arrays of quantiles estimated with MEV, GEV, and POT respectively (same size of `TR_val`)

- `QuM, QuG, QuP`: Arrays of 95% CIs upper bounds for MEV, GEV, and POT respectively.

- `QlM, QlG, QlP`: Arrays of 95% CIs lower bounds for MEV, GEV, and POT respectively.

- `FhM, FhG, FhP`: Arrays of non exceedance probabilities for MEV, GEV, and POT respectively, computed for the observed annual maxima in `XI_val`.

Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)

Additional Arguments:

- `tr_min`: Minimum return time for which quantiles are computed. Default is 5.

- `ws`: Dimension of the blocks in which the WEI distributions are fitted. Default is 1 year.

- `GEV_how`: Fitting method for GEV. Possible options are `'lmom'` for fitting with L-moments method [5] or `'ml'` for maximum likelihood [2]. Default is `'lmom'`.

- `MEV_how`: Fitting method for the WEI. Default is `'pwm'` for Probability Weighted Moments. Maximum Likelihood (`'ml'`) and Least-Squares (`'ls'`) are also available.

- `MEV_thresh`: Threshold for fitting WEI, censoring the values below threshold, without accounting for the relative mass of probability. Default is zero.

- `POT_way`: For POT, Way for selecting the threshold. If `'ea'` a threshold is selected such that a fixed average number of events/year exceedes it (default). If 'thresh' the threshold is specified directly. If `'sp'`the threshold is determined from a given survival probability of being above threshold. Default is `ea`.

- `POT_val`: Value for POT threshold selection. Depending on the value of `POT_way`, this quantity is: Value of the threshold if `POT_way='thresh'`, value of the exceedance probability above threshold if `POT_way='sp'`, mean number of excesses per block if `POT_way='ea'`. Default is the latter with value `POT_val=3` excesses/block.
- `POT_how`: Method for fitting Generalized Pareto Distribution (GPD) to threshold exceedances. Default is `'ml'` for maximum likelihood. Alternatively, `'pwm'` for probability weighted moments.
- `ci`: If `True`, confidence intervals are computed. Default is `False`.
- `ntimes`: Number or resampling for the bootstrap. Needed only if `ci = True` and `std_how = 'boot'`. Default is `ntimes = 1000` generations.
- `std_how_MEV`: Method for computing confidence intervals for MEV. Possible choices are `'boot'` for nonparametric bootstrap with normal distribution of quantiles, and `'boot_cdf'` for a bootstrap which does not employs the assumption of normal distribution but instead uses the observed cdf of quantiles. Not centered at the MEV estimate.
- `std_how_GEV`: Method for computing covariance matrix for GEV. Possible choices are `'hess'` for numeric Hessian or `'boot'` for nonparametric bootstrap. Default is `'hess'`, but this is only available for Maximum-Likelihood GEV estimation.
- `std_how_POT`: Method for computing covariance matrix for POT. Possible choices are `'hess'` for numeric Hessian or `'boot'` for nonparametric bootstrap. Default is `'hess'`, but this is only available for Maximum-Likelihood GEV estimation.
- `rmy`: Remove Missing Years: minimum number of missing values per year to keep that year in the analysis. Default is 36.

`shuffle_mat(datamat, nyears, ncal, nval)`

Randomly resample the years in a dataset and divide it in two different subsample for independent validation and calibration of Extreme Value Models. Note: here the events within each year are not shuffled, just the order with which years appear in the record.
Returns:

- `mat_cal`: Array with calibration data only, shape (ncal, ndays) with e.g., `ndays = 366` (=fixed block size)

- `max_cal`: Array with annual maxima from the calibration sample, of size ncal
- `max_val`: Array with annual maxima from the validation sample, of size nval
- `datamat_r`: Array with all the data, with shuffled years.

Arguments:

- `datamat`: Array with observations in the dataset. Must have shape (nyears, ndays) with e.g., `ndays = 366`.
- `nyears`: Number of years in the dataset
- `ncal`: Number of years for calibration
- `nval`: Number of years for validation

`shuffle_all(datamat, nyears, ncal, nval)`

Shuffle all the daily values, and the yearly numbers of wet days in a dataset and divide it in two different sub-sample for independent validation and calibration of Extreme Value Models. Note: here the events within all the time series are reshuffled (no resubstituion), and so is the number of wet days/years as in Zorzetto et al., 2016.
Returns:

- `mat_cal`: Array with calibration data only, shape (ncal, ndays) with e.g., `ndays = 366` (=fixed block size)
- `max_cal`: Array with annual maxima from the calibration sample, of size ncal
- `max_val`: Array with annual maxima from the validation sample, of size nval
- `datamat_r`: Array with all the data, with shuffled years.

Arguments:

- `datamat`: Array with observations in the dataset. Must have shape (nyears, ndays) with e.g., `ndays = 366`.
- `nyears`: Number of years in the dataset
- `ncal`: Number of years for calibration

- `nval`: Number of years for validation

```
cross_validation(df, ngen, ncal, nval, tr_min = 5, ws=[1],
                 GEV_how = 'lmom', MEV_how = 'pwm', MEV_thresh = 0,
         POT_way = 'ea', POT_val = 3, cross = True, ncal_auto = 100
         shuff = 'year')
```

Perform a cross validation for evaluating performance of EV models: GEV, POT (for different threshold values) and MEV (for different window sizes for WEI fitting) performing hgen random reshufflings of the years in the time series and evaluating model performances for a set of ntr return times.
Returns:

- `TR_val`: arrays of empirical return times for which quantiles are computed.
- `m_rmse`: array with the normalized Root Mean Square Error (RMSE) computed for MEV estimates. Array has shape (nwinsizes, ntr)
- `g_rmse`: array with the normalized Root Mean Square Error (RMSE) computed for GEV estimates. Array has shape (ntr)
- `p_rmse`: array with the normalized Root Mean Square Error (RMSE) computed for POT estimates. Array has shape (nthresh, ntr)
- `em`: MEV relative errors for nwinsizes different window sizes for fitting WEI. Array with shape (ngen, nwinsizes, ntr)
- `eg`: GEV relative errors. Array with shape (ngen, ntr)
- `ep`: POT relative errors for nthresh different values of the threshold. Array with shape (ngen, nthresh, ntr)

Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)
- `ngen`: Number of generations of reshuffled datasets for computing RMSEs.
- `ncal`: Number of years for calibration. (= sample size)
- `nval`: Number of years for valibration. (= maximum Return Time tested)

Additional Arguments:

- `tr_min`: Minimum return time for which quantiles are computed. Default is 5.

- `ws`: Array or list of window sizes (in years) for fitting the single Weibull for MEV. Default is one single window of size [1] year. We name the size of this array nwinsizes, or number of window sizes.

- `GEV_how`: Fitting method for GEV. Possible options are `'lmom'` for fitting with L-moments method [5] or `'ml'` for maximum likelihood [2]. Default is `'lmom'`.

- `MEV_how`: Fitting method for the WEI. Default is `'pwm'` for Probability Weighted Moments. Maximum Likelihood (`'ml'`) and Least-Squares (`'ls'`) are also available.

- `MEV_thresh`: Threshold for fitting WEI, censoring the values below threshold, without accounting for the relative mass of probability. Default is zero.

- `POT_way`: For POT, Way for selecting the threshold. If `'ea'` a threshold is selected such that a fixed average number of events/year exceedes it (default). If 'thresh' the threshold is specified directly. If `'sp'`the threshold is determined from a given survival probability of being above threshold. Default is `ea`.

- `POT_val`: Array or list of values for POT threshold selection. Depending on the value of `POT_way`, these quantities are: Values of the threshold, if `POT_way='thresh'`; Values of the exceedance probability above threshold if `POT_way='sp'`; Values of the mean number of excesses per block if `POT_way='ea'`. The default option is the latter with only one value `POT_val=3` excesses/block. We name the size of this array nthresh, or number of thresholds.

- `cross`: If True, analysis is performed in cross validation with independent samples for calibration and validation (note: in this case the combined length of cal. and val. samples must not exceed the number of years of the original dataset.) If False, the same sample is used to generate calibration and validation samples. Default is True.

- `ncal_auto`: This is the common length (in years) of the sample for calibration and validation when cross = False (only used in this case; default is 100 year - but make sure the original sample is at least 100 years long or select a lower value).

- `shuff`: Method for obtaining synthetic series based on the original one. If `shuff='year'` (default) only the years in the records are resampled with ersubstitution. If `shuff='all'`, all daily data in the time series and the number of wet days/ year are reshuffled (without resubstitution) to construct the synthetic series.

```
slideover(df, winsize = 30, Tr = 100, display = True,
                                        ci = True, ntimes = 100)
```

Perform extreme value analysis on a long rainfall record using sliding and overlapping window. **Returns**:

- `central_year, mq, mqu, mql, gq, gqu, gql, pq, pqu, pql, fig1, fig2` (if `ci = True`)
- `central_year, mq, gq, pq, fig1` (if `ci = False`)

where:

- `central_year`: array with central years of each window, for plotting results.
- `mq, gq, pq`: arrays with quantiles estimated with MEV, GEV and POT respectively.
- `mqu , gqu, pqu`: arrays of 95% CIs upper bounds for MEV, GEV, and POT respectively.
- `central_year`: arrays of 95% CIs lower bounds for MEV, GEV, and POT respectively.
- `fig1, fig2`: figures with results.

Arguments:

- `df`: dataframe with the rainfall daily data for distribution fitting. It must contain a field PRCP with the observations (e.g., daily rainfall amounts) and a field YEAR with the year of observation (integer with format yyyy)

Additional Arguments:

- `winsize`: Size of the window. Default is 30 years.
- `Tr`: Return time for which extreme quantiles are evaluated. Default is 30 years.

- `display`: If True, show the plots during execution.
- `ci`: If True, compute the 95% confidence intervals for the quantiles.
- `ntimes`: Number of bootstrap resampling. Default is 100. Keep it low enough because it may take a long time to run, especially for computing MEV CIs.

Acknowledegments

Enrico Zorzetto acknowledges support from the Division of Earth and Ocean Sciences, Duke University and the NASA Earth and Space Science Fellowship (NESSF 17-EARTH17F-0270).

References

[1] E. Zorzetto, G. Botter, and M. Marani, "On the emergence of rainfall extremes from ordinary events," *Geophysical Research Letters*, vol. 43, no. 15, pp. 8076–8082, 2016.

[2] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, *An introduction to statistical modeling of extreme values*, vol. 208. Springer, 2001.

[3] M. Marani and M. Ignaccolo, "A metastatistical approach to rainfall extremes," *Advances in Water Resources*, vol. 79, pp. 121–126, 2015.

[4] R. L. Smith, "Max-stable processes and spatial extremes," *Unpublished manuscript*, vol. 205, 1990.

[5] J. R. Hosking, "L-moments: analysis and estimation of distributions using linear combinations of order statistics," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 105–124, 1990.

[6] J. A. Greenwood, J. M. Landwehr, N. C. Matalas, and J. R. Wallis, "Probability weighted moments: definition and relation to parameters of several distributions expressable in inverse form," *Water Resources Research*, vol. 15, no. 5, pp. 1049–1054, 1979.