Return Curves Estimation

Lídia André, Callum Murphy-Barltrop, Jennifer Wadsworth

2024-06-29

Contents

1 Introduction

This vignette provides complementary information to the R Documentation for the `ReturnCurves` package. It summarises the key methodologies implemented in the package and is heavily based on the works of Murphy-Barltrop et al. [2023] and Murphy-Barltrop et al. [2024]; for full details we refer the user to these articles.

The `ReturnCurves` package aims at estimating the p-probability return curve [Murphy-Barltrop et al., 2023], for small $p > 0$, while implementing pointwise and smooth approaches to estimate the so called angular dependence function first introduced by Wadsworth and Tawn [2013].

```
library(ReturnCurves)
```

To illustrate the functionality of the package, we use the data set `airdata` which contained air pollution data collected from Marylebone, London (UK). The data set contains 1427 daily measurements of air pollutant concentrations of NOx and PM10.

```
data(airdata)
```

2 Marginal transformation

The estimation of the angular dependence function and/or of the return curve is implemented for a bivariate vector (X_E, Y_E) marginally distributed as standard exponential, i.e, X_E, $Y_E \sim \text{Exp}(1)$. Thus, the original data (X, Y) needs to be marginally transformed, which is achieved via the probability integral transform. We follow the procedure of Coles and Tawn [1991] where the empirical cumulative distribution function $\tilde{F}$ is fitted below a threshold u, and a generalised Pareto distribution (GPD) is fitted above, giving the following

estimate of the marginal cumulative distribution function (cdf) of X or Y :

$$\hat{F}(z) = \begin{cases} 1 - \left(1 - \tilde{F}(u)\right) \left[1 + \hat{\xi}\frac{z-u}{\hat{\sigma}}\right]_+^{-1/\hat{\xi}}, & \text{if } z > u, \\ \tilde{F}(z), & \text{if } z \leq u, \end{cases} \tag{1}$$

where $\hat{\sigma}$ and $\hat{\xi}$ are the estimated scale and shape parameters of the GPD. Exponential margins are obtained by applying $-\log(1 - \hat{F}(\cdot))$ to each margin, where $\hat{F}(\cdot)$ is estimated separately for each margin.

This is done with the function `margtransf` which takes as inputs a matrix containing the original data, a vector of the marginal quantiles used to fit the GPD and a boolean value `constrainedshape` which decides whether $\xi > -1$ if set to `TRUE` (Default), or $\xi \in \mathbb{R}$ if set to `FALSE`.

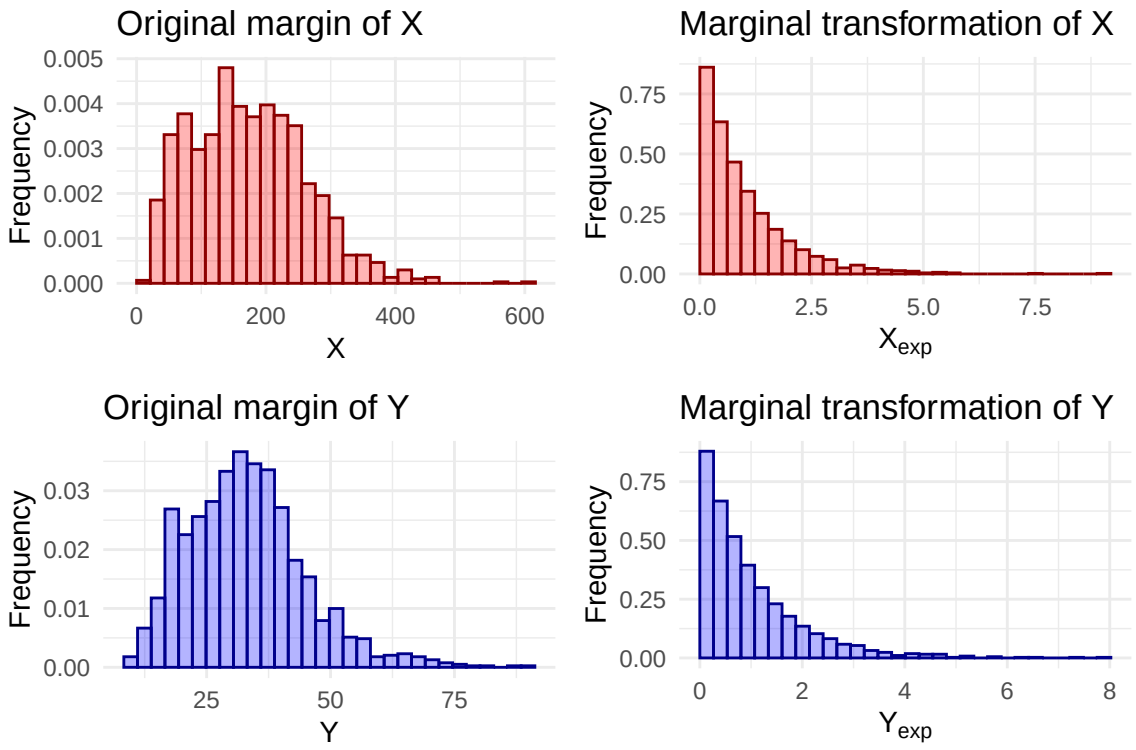
Function `margtransf` returns an object of S4 class `margtrasnf.class` with 6 attributes:

- `data`: matrix with the data on the original margins
- `qmarg`: vector of marginal quantiles used to fit the GPD
- `constrainedshape`: whether $\xi > -1$ (`TRUE`) or $\xi \in \mathbb{R}$ (`FALSE`)
- `parameters`: matrix containing estimates of parameters $(\hat{\sigma}, \hat{\xi})$
- `thresh`: vector containing threshold u above which the GPD is fitted
- `dataexp`: matrix with the data on standard exponential margins

```
# qmarg and constrainedshape set to the default values
expdata <- margtransf(data = airdata, qmarg = rep(0.95, 2), constrainedshape = T)

# attributes of the S4 object
str(expdata)
#> Formal class 'margtransf.class' [package "ReturnCurves"] with 6 slots
#>   ..@ data            : num [1:1427, 1:2] 154 132 120 105 175 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   .. .. ..$ : NULL
#>   .. .. ..$ : chr [1:2] "nox" "pm10"
#>   ..@ qmarg           : num [1:2] 0.95 0.95
#>   ..@ constrainedshape: logi TRUE
#>   ..@ parameters      : num [1:2, 1:2] 58.3333 -0.0521 9.984 -0.1428
#>   ..@ thresh          : num [1:2] 320.5 54.4
#>   ..@ dataexp         : num [1:1427, 1:2] 0.594 0.415 0.369 0.29 0.747 ...

# head of the data on standard exponential margins
head(expdata@dataexp)
#>           [,1]      [,2]
#> [1,] 0.5941359 0.1035535
#> [2,] 0.4150041 0.4307285
#> [3,] 0.3692105 0.1596506
#> [4,] 0.2900289 0.1827463
#> [5,] 0.7465895 0.2808641
#> [6,] 1.1461374 0.9017124
```

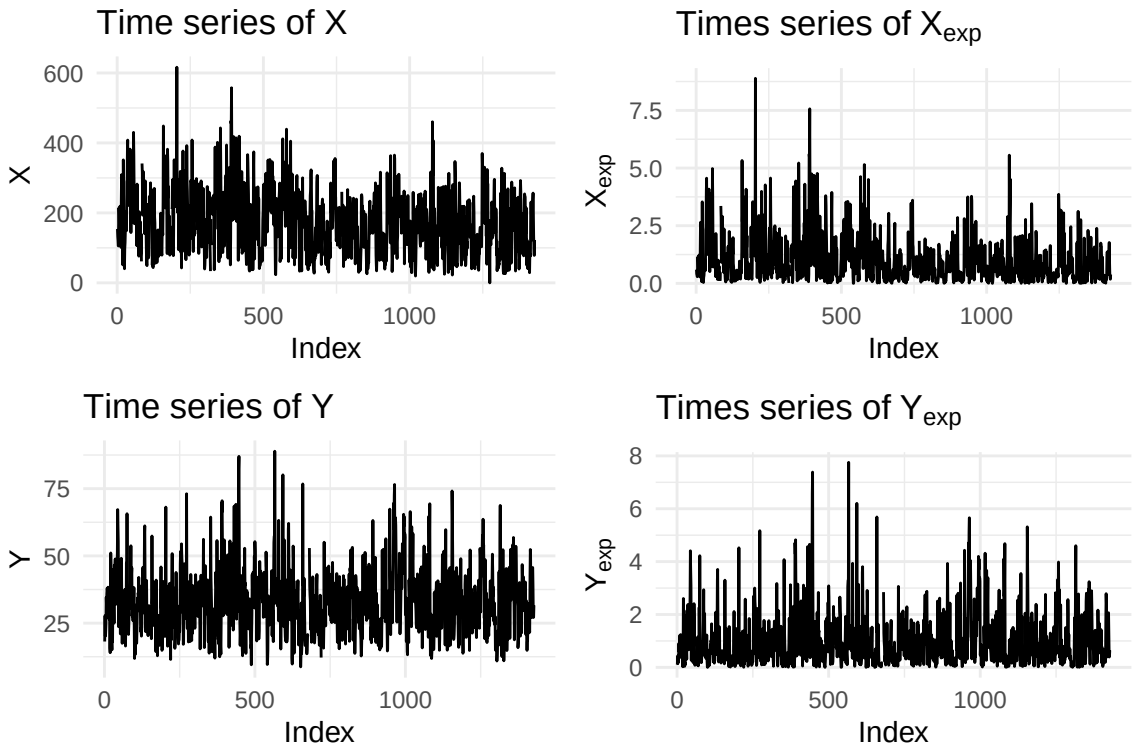
It is possible to plot an S4 object of `margtrasnf.class` with `plot`. By setting argument `which = "hist"`, histograms of each variable on original and standard exponential margins can be seen:

```
plot(expdata, which = "hist")
```



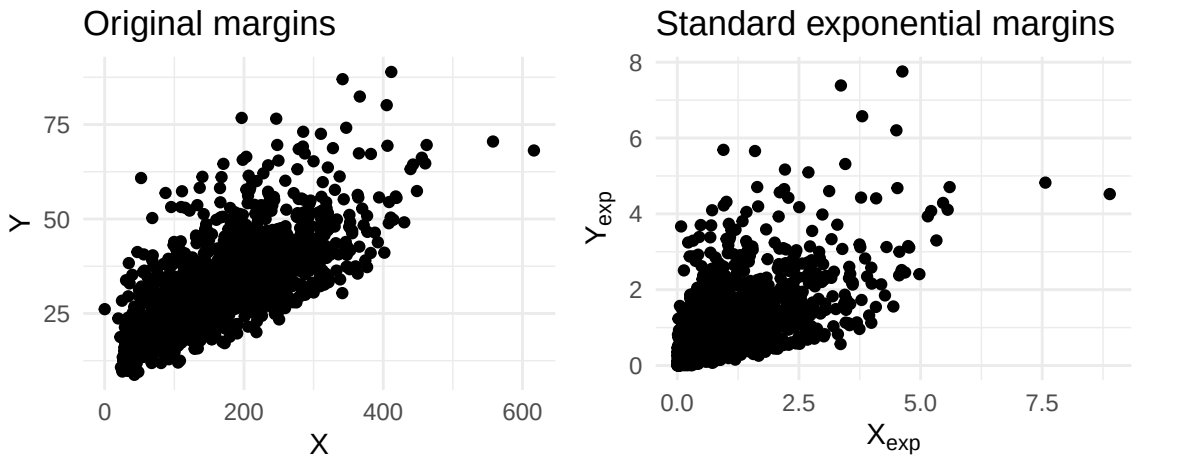
To visualise the time series of each variable on original and standard exponential margins, we need to set `which = "ts"`:

```
plot(expdata, which = "ts")
```



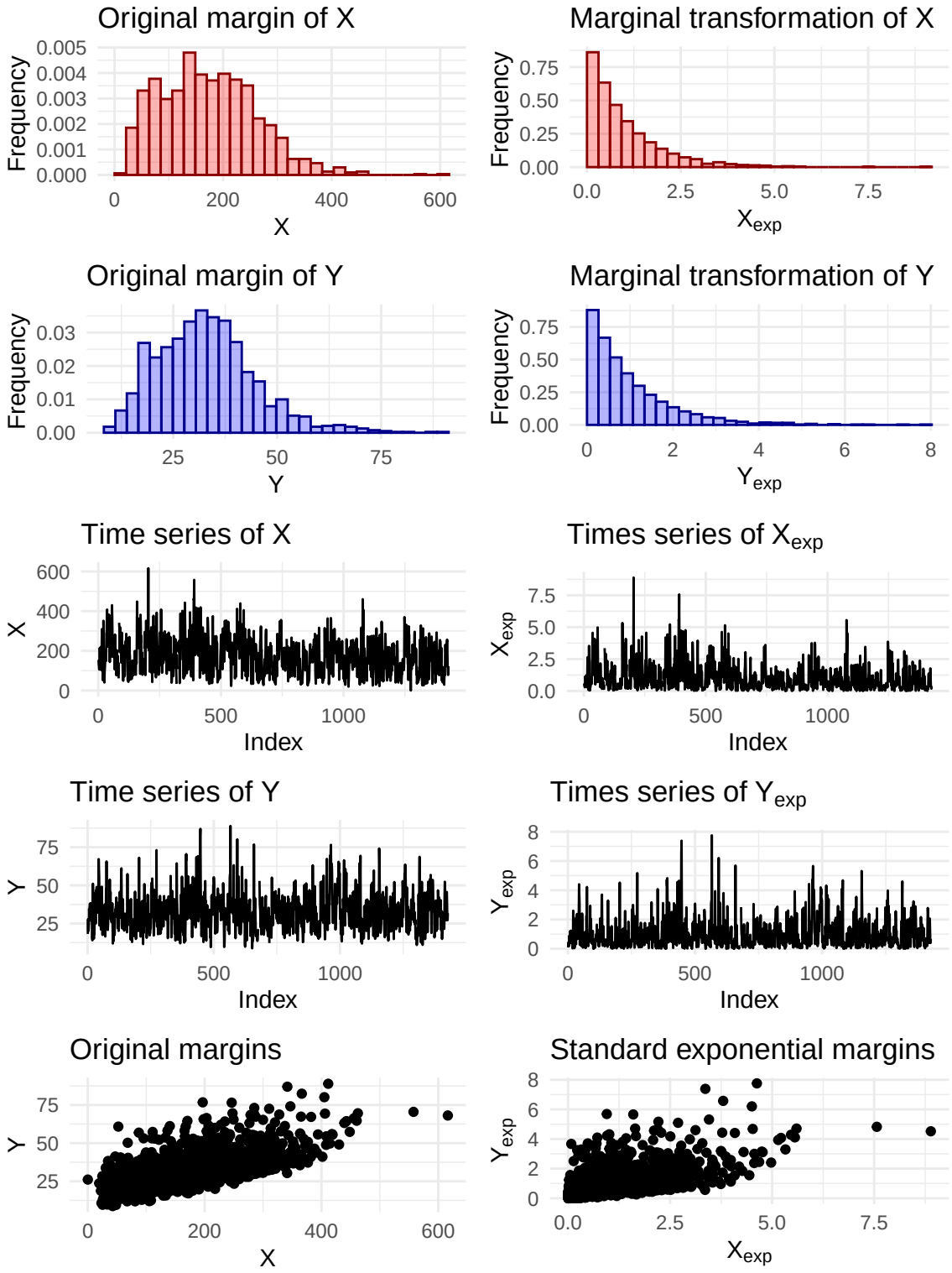
The joint distribution on original and standard exponential margins can be accessed with `which = "joint"`:

```
plot(expdata, which = "joint")
```



Finally, it is possible to plot all these together by setting `which = "all"`, which is the default for this argument.

```
plot(expdata, which = "all") # or just plot(expdata)
```



2.1 Assessing the marginal tail fits

When transforming the data onto standard exponential variables as in equation (1), it is assumed that above a threshold u data follows a GPD. It is possible to assess if this is a reasonable assumption through checking if there is an agreement between model and empirical GPD quantiles. This is done via QQ plots in the

`ReturnCurves` package by plotting points $\left(F_{GPD}^{-1}\left(\frac{i}{n_{exc}+1}\right)+u, X_{(i)}^{GPD}+u\right)$, where $X_{(i)}^{GPD}$ denotes the i-th ordered increasing statistic ($i=1,\ldots,n$) of the exceedances, i.e., $X^{GPD}=(X-u \mid X>u)$, n_{exc} denotes the sample size of these exceedances, and F_{GPD}^{-1} denotes the inverse of the cumulative distribution function of a GPD. Finally, the uncertainty on the empirical quantiles is quantified using a bootstrap approach. If temporal dependence is present in the data, then a block bootstrap approach is required, i.e., `blocksize > 1`.

This is done using the function `marggpd` function which takes as inputs an S4 object of class `margtransf.class`, the size of blocks of the bootstrap procedure and the corresponding number of samples, and the significance level α for the tolerance intervals. It then returns an S4 object of class `marggpd.class` with an extra attribute `marggpd` containing a list with:

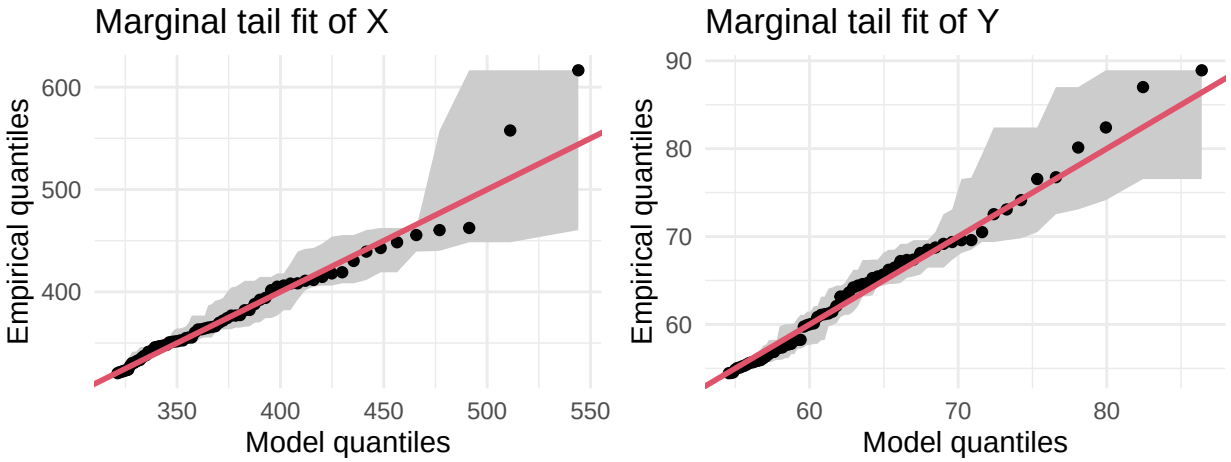
- `model`: a list containing the model quantiles for each variable,
- `empirical`: a list containing the empirical quantiles for each variable,
- `lower`: a list containing the lower bounds of the tolerance intervals for each variable,
- `upper`: a list containing the upper bounds of the tolerance intervals for each variable.

```
# nboot and alpha are set to the default values
# blocksize is set to 10 to account for temporal dependence
uncgpd <- marggpd(margdata = expdata, blocksize = 10, nboot = 250, alpha = 0.05)

# attributes of the S4 object
str(uncgpd)
#> Formal class 'marggpd.class' [package "ReturnCurves"] with 5 slots
#>   ..@ margdata :Formal class 'margtransf.class' [package "ReturnCurves"] with 6 slots
#>   .. .. ..@ data            : num [1:1427, 1:2] 154 132 120 105 175 ...
#>   .. .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. .. ..$ : NULL
#>   .. .. .. .. ..$ : chr [1:2] "nox" "pm10"
#>   .. .. ..@ qmarg           : num [1:2] 0.95 0.95
#>   .. .. ..@ constrainedshape: logi TRUE
#>   .. .. ..@ parameters      : num [1:2, 1:2] 58.3333 -0.0521 9.984 -0.1428
#>   .. .. ..@ thresh          : num [1:2] 320.5 54.4
#>   .. .. ..@ dataexp         : num [1:1427, 1:2] 0.594 0.415 0.369 0.29 0.747 ...
#>   ..@ blocksize: num 10
#>   ..@ nboot    : num 250
#>   ..@ alpha    : num 0.05
#>   ..@ marggpd  :List of 4
#>   .. ..$ model    :List of 2
#>   .. .. ..$ : num [1:71] 321 322 323 324 325 ...
#>   .. .. ..$ : num [1:71] 54.6 54.7 54.9 55 55.1 ...
#>   .. ..$ empirical:List of 2
#>   .. .. ..$ : num [1:71] 321 321 322 322 323 ...
#>   .. .. ..$ : num [1:71] 54.5 54.5 54.5 54.9 55 ...
#>   .. ..$ lower    :List of 2
#>   .. .. ..$ : num [1:71] 321 321 321 321 322 ...
#>   .. .. ..$ : num [1:71] 54.5 54.5 54.5 54.5 54.5 ...
#>   .. ..$ upper    :List of 2
#>   .. .. ..$ : num [1:71] 322 323 324 329 330 ...
#>   .. .. ..$ : num [1:71] 54.5 54.9 55.1 55.1 55.2 ...

# head of the list elements of slot marggpd for variable X
head(uncgpd@marggpd$model[[1]])
#> [1] 321.2739 322.1004 322.9382 323.7876 324.6489 325.5225
head(uncgpd@marggpd$empirical[[1]])
```

```
#> [1] 320.5833 321.4167 321.8333 322.4167 323.0417 323.5000
head(uncgpd@marggpd$lower[[1]])
#> [1] 320.5833 320.5833 321.4167 321.4167 321.8333 322.4167
head(uncgpd@marggpd$upper[[1]])
#> [1] 321.8333 323.0417 323.5000 328.7635 330.2917 332.8750
```

It is possible to plot an S4 object of `marggpd.class` with `plot`, where the QQ plots with the model and empirical quantiles for each variable are shown. The points should lie close to the line $y = x$; for a good fit and agreement between these quantiles, the line $y = x$ should mainly lie within the $(1 - \alpha)\%$ tolerance intervals.

```
plot(uncgpd)
```



3 Estimation of the angular dependence function

In bivariate extremes, interest may lie in studying regions where both variables are extreme or where only one is extreme. For this, methods that aim at characterising the joint tail behaviour in both scenarios, such as the one introduced by Wadsworth and Tawn [2013], are required. Given standard exponentially distributed variables X_E and Y_E and a slowly varying function $\mathcal{L}(\cdot;\omega)$ at infinity, the joint tail behaviour of (X_E, Y_E) is captured through $\lambda(\omega)$ via the assumption

$$\Pr(X_E > \omega u,\, Y_E > (1-\omega)u) = \mathcal{L}(e^u;\omega)e^{-\lambda(\omega)u} \quad \text{as } u \to \infty,$$

which can be rewritten as

$$\Pr\left(\min\left\{\frac{X_E}{\omega}, \frac{Y_E}{1-\omega}\right\}\right) = \mathcal{L}(e^u;\omega)e^{-\lambda(\omega)u} \quad \text{as } u \to \infty, \tag{2}$$

where $\omega \in [0,1]$ and $\lambda(\omega) \geq \max\{\omega, 1-\omega\}$ is called the angular dependence function (ADF). In the case of asymptotic dependence (see for instance, Coles et al. [1999]), $\lambda(\omega) = \max\{\omega, 1-\omega\}$, for all $\omega \in [0,1]$.

Lastly, defining a min-projection variable at ω, $T_\omega = \min\left\{\frac{X_E}{\omega}, \frac{Y_E}{1-\omega}\right\}$, equation (2) implies that

$$\Pr(T_\omega > u + t \mid T_\omega > u) = \frac{\mathcal{L}(e^{u+t};\omega)}{\mathcal{L}(e^u;\omega)}e^{-\lambda(\omega)t} \to e^{-\lambda(\omega)t} \quad \text{as } u \to \infty, \tag{3}$$

for any $\omega \in [0,1]$ and $t > 0$. In other words, for all $\omega \in [0,1]$ and, as $u_\omega \to \infty$, $T_\omega^1 := (T_\omega - u_\omega \mid T_\omega > u_\omega) \sim \text{Exp}(\lambda(\omega))$. Estimation of the ADF can be done in different ways; Murphy-Barltrop et al. [2024] present a few.

For the `ReturnCurves` package, two approaches are implemented: a pointwise estimator using the Hill estimator [Hill, 1975], $\hat{\lambda}_H$, and a smoother estimator based on Bernstein-Bézier polynomials estimated via

composite likelihood methods, $\hat{\lambda}_{CL}$. For the latter, Murphy-Barltrop et al. [2024] propose using a family of Bernstein-Bézier polynomials to improve the estimation of the ADF. Given $k \in \mathbb{N}$, it is assumed that $\lambda(\omega) = \lambda(\omega; \boldsymbol{\beta})$ can be represented by the following family of functions:

$$\mathcal{B}_k^* = \Bigg\{ (1-\omega)^k + \sum_{i=1}^{k-1} \beta_i \binom{k}{i} \omega^i (1-\omega)^{k-i} + \omega^k =: f(\omega) \mid \omega \in [0,1], \\ \boldsymbol{\beta} \in [0,\infty)^{k-1} \text{ such that } f(\omega) \geq \max\{\omega, 1-\omega\} \Bigg\}. \tag{4}$$

As T_ω^1 is exponentially distributed when $u_\omega \to \infty$, the parameter vector $\boldsymbol{\beta}$ can be estimated using a composite likelihood function defined as

$$\mathcal{L}_C(\boldsymbol{\beta}) = \left[\prod_{\omega \in \Omega} \lambda(\omega; \boldsymbol{\beta})^{|\boldsymbol{t}_\omega^1|} \right] \exp\left\{ - \sum_{\omega \in \Omega} \sum_{t_\omega^1 \in \boldsymbol{t}_\omega^1} \lambda(\omega; \boldsymbol{\beta}) t_\omega \right\}, \tag{5}$$

where $| \boldsymbol{t}_\omega^1 |$ represents the cardinality of set $\boldsymbol{t}_\omega^1 := \{t_\omega - u_\omega \mid t_\omega \in \boldsymbol{t}_\omega, t_\omega > u_\omega\}$ for some large values u_ω, and Ω is a finite subset spanning the interval $[0,1]$. The estimator of the ADF through composite likelihood methods is given by $\lambda(\cdot; \hat{\boldsymbol{\beta}}_{CL})$ where $\hat{\boldsymbol{\beta}}_{CL}$ maximises equation (5).

Finally, Murphy-Barltrop et al. [2024] showed that incorporating knowledge of the conditional extremes [Heffernan and Tawn, 2004] parameters $\alpha_{y|x}$ and $\alpha_{x|y}$ improves the estimation of the ADF. In particular, the authors show that, in order to satisfy theoretical properties of $\lambda(\omega)$, $\lambda(\omega) = \max\{\omega, 1-\omega\}$ for all $\omega \in [0, \alpha_{x|y}^1] \cup [\alpha_{y|x}^1, 1]$ with $\alpha_{x|y}^1 = \alpha_{x|y}/(1+\alpha_{x|y})$ and $\alpha_{y|x}^1 = 1/(1+\alpha_{y|x})$. Thus, after estimating the conditional extremes parameters $\alpha_{y|x}$ and $\alpha_{x|y}$ through maximum likelihood estimation, we can set $\lambda(\omega) = \max\{\omega, 1-\omega\}$ for $\omega \in [0, \hat{\alpha}_{x|y}^1) \cup (\hat{\alpha}_{y|x}^1, 1]$. Then, for the Hill estimator, $\lambda(\omega) = \hat{\lambda}_H$ for $\omega \in \left[\hat{\alpha}_{x|y}^1, \hat{\alpha}_{y|x}^1\right]$. For the composite likelihood estimator, a rescaling of equation (4) is needed to ensure continuity at $\hat{\alpha}_{x|y}^1$ and $\hat{\alpha}_{y|x}^1$, as defined below:

$$\mathcal{B}_k^1 = \Bigg\{ (1-\hat{\alpha}_{x|y}^1) \left(1 - \frac{v - \hat{\alpha}_{x|y}^1}{\hat{\alpha}_{y|x}^1 - \hat{\alpha}_{x|y}^1}\right)^k + \sum_{i=1}^{k-1} \beta_i \binom{k}{i} \left(\frac{v - \hat{\alpha}_{x|y}^1}{\hat{\alpha}_{y|x}^1 - \hat{\alpha}_{x|y}^1}\right)^i \left(1 - \frac{v - \hat{\alpha}_{x|y}^1}{\hat{\alpha}_{y|x}^1 - \hat{\alpha}_{x|y}^1}\right)^{k-i} + \\ \hat{\alpha}_{y|x}^1 \left(\frac{v - \hat{\alpha}_{x|y}^1}{\hat{\alpha}_{y|x}^1 - \hat{\alpha}_{x|y}^1}\right)^k =: f(v) \mid v \in \left[\hat{\alpha}_{x|y}^1, \hat{\alpha}_{y|x}^1\right], \boldsymbol{\beta} \in [0,\infty)^{k-1} \text{ such that } f(v) \geq \max\{v, 1-v\} \Bigg\}.$$

$\lambda(\omega) = \lambda(\omega; \boldsymbol{\beta})$ is assumed to be represented by an element of $\mathcal{B}_k^1$ on $\left[\hat{\alpha}_{x|y}^1, \hat{\alpha}_{y|x}^1\right]$. Finally, the estimators used for estimation are processed in order to satisfy theoretical conditions on λ as identified in Murphy-Barltrop et al. [2024].

Estimation of the ADF can be done using the function `adf_est` which takes as inputs:

- an S4 object of class `margtransf.class` representing the marginal transformation of the data,
- a sequence of rays `w` in $[0,1]$,
- a string `method` indicating which estimator to get, λ_H or λ_{CL},
- and a boolean value `constrained` which decides whether to incorporate conditional extremes parameters $\alpha_{y|x}$ and $\alpha_{x|y}$ in the estimation.

Additional arguments can be defined outside of the default values; these include marginal quantiles for the min-projection variable T^1 at ray ω, marginal quantiles to fit the conditional extremes method if `constrained=TRUE`, and, if `method= "cl"`, the polynomial degree k, the initial values for $\boldsymbol{\beta}$ for the composite maximum likelihood procedure, and the convergence tolerance. Convergence is declared when the difference of log-likelihood values between iterations does not exceed the value of `tol`. This repeated optimisation helps to avoid convergence to local maxima, although does not guarantee finding the global maximum.

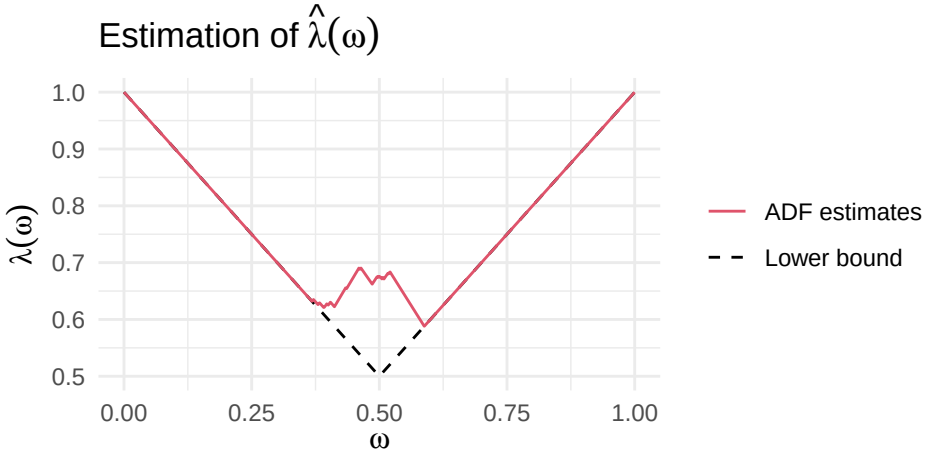
Function `adf_est` returns an object of S4 class `adf_est.class` with 11 attributes, where the first 9 are the inputs of the function and the last 2 are vectors:

- `interval`: contains the maximum likelihood estimates from the conditional extremes model $\hat{\alpha}^1_{x|y}$ and $\hat{\alpha}^1_{y|x}$ if `constrained = TRUE`. Otherwise, it returns the values 0 and 1; this has no meaningful interpretation as the estimation is performed in an unconstrained interval.
- `adf`: contains the estimates of $\lambda(\omega)$.

```
# Estimation using Hill estimator without conditional extremes parameters
whill <- seq(0, 1, by = 0.001)
## q and constrained are set to the default values here
lambdah <- adf_est(margdata = expdata, w = whill, method = "hill",
                   q = 0.95, constrained = F)

# Estimation using Hill estimator with conditional extremes parameters
## q and qalphas are set to the default values
lambdah2 <- adf_est(margdata = expdata, w = whill, method = "hill", q = 0.95,
                    qalphas = rep(0.95, 2), constrained = T)

# Estimation using CL method without conditional extremes parameters
## w, q and constrained are set to the default values here
lambdacl <- adf_est(margdata = expdata, w = seq(0, 1, by = 0.01), method = "cl",
                    q = 0.95, constrained = F)

# Estimation using CL method with conditional extremes parameters
## w, q and qalphas are set to the default values
lambdacl2 <- adf_est(margdata = expdata, w = seq(0, 1, by = 0.01), method = "cl",
                     q = 0.95, qalphas = rep(0.95, 2), constrained = T)

# attributes of the S4 object
str(lambdah)
#> Formal class 'adf_est.class' [package "ReturnCurves"] with 11 slots
#>   ..@ dataexp    : num [1:1408, 1:2] 0.594 0.415 0.369 0.29 0.747 ...
#>   ..@ w          : num [1:1001] 0 0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 ...
#>   ..@ method     : chr "hill"
#>   ..@ q          : num 0.95
#>   ..@ qalphas    : num [1:2] 0.95 0.95
#>   ..@ k          : num 7
#>   ..@ constrained: logi FALSE
#>   ..@ tol        : num 1e-04
#>   ..@ par_init   : num [1:6] 0 0 0 0 0 0
#>   ..@ interval   : num [1:2] 0 1
#>   ..@ adf        : num [1:1001] 1 0.999 0.998 0.997 0.996 0.995 0.994 0.993 0.992 0.991 ...

# head of the vector with adf estimates for the first estimator
head(lambdah@adf)
#> [1] 1.000 0.999 0.998 0.997 0.996 0.995
```

It is possible to plot an S4 object of `adf_est.class` with `plot`, where a comparison of the estimated ADF and its lower bound, $\max\{\omega, 1-\omega\}$, is shown.

```
# plot of the ADF estimation based on the unconstrained Hill estimator
plot(lambdah)
```



3.1 Goodness-of-fit of the angular dependence function

After estimation of the ADF, it is important to assess its goodness-of-fit. Noting that $T_\omega^1 = (T_\omega - u_\omega \mid T_\omega > u_\omega) \sim \text{Exp}(\lambda(\omega)) \Leftrightarrow \lambda(\omega) T_\omega^1 \sim \text{Exp}(1)$, we can investigate whether there is agreement between model and empirical exponential quantiles, or not. This is done in the `ReturnCurves` package through QQ plots by plotting points $\left(F_E^{-1}(i/(n+1)), T_{(i)}^1\right)$, where F_E^{-1} denotes the inverse of the cumulative distribution function of a standard exponential distribution and $T_{(i)}^{-1}$ is the i-th ordered increasing statistic, $i = 1, \ldots, n$. The uncertainty of the empirical quantiles is quantified using a bootstrap approach. If temporal dependence is present in the data, a block bootstrap approach should be used, i.e. `blocksize` > 1.

The assessment of the goodness-of-fit of $\lambda(\omega)$ can be done using the function `adf_gof` which takes an S4 object of class `adf_est.class`, a ray ω to be considered, the size of the blocks for the bootstrap procedure and the corresponding number of samples, and the significance level α for the tolerance intervals as inputs. In turn, it returns an S4 object of class `adf_gof.class` with an extra attribute `gof` containing a list with the model and empirical quantiles, and the lower and upper bounds of the tolerance interval.

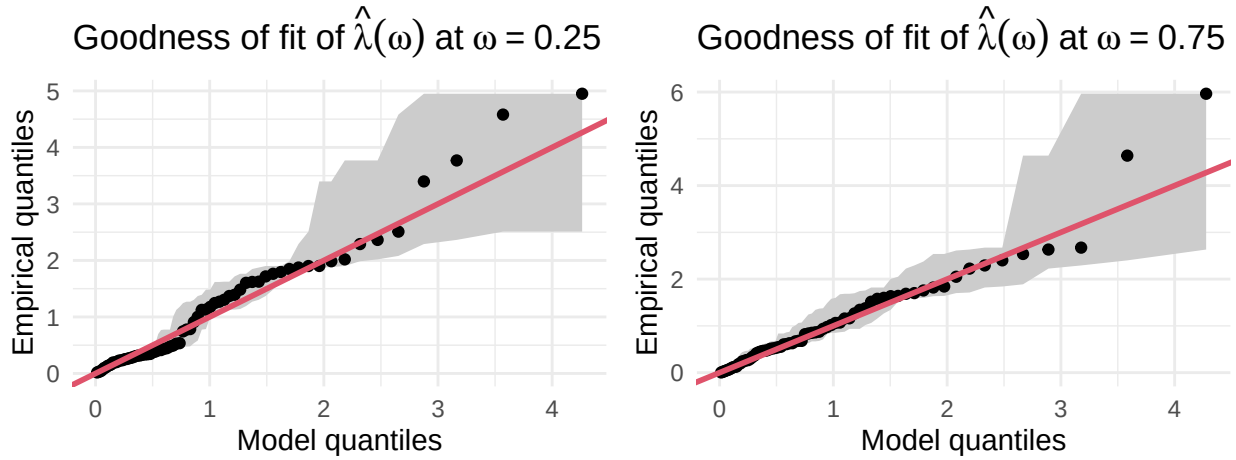
We note that this function is implemented to evaluate the fit at a single ray ω; therefore, we recommend repeating the procedure for a few rays to have a better representation. In addition, if the ray provided by the user was not used for the estimation of the ADF, then the closest ω in the grid is used instead.

```
# Goodness of fit of the adf for twp rays w
rays <- c(0.25, 0.75)
## nboot and alpha are set to the default values
## blocksize is set to 10 to account for temporal dependence
gofh <- sapply(rays, adf_gof, adf = lambdah, blocksize = 10, nboot = 250, alpha = 0.05)

# attributes of the S4 object
str(gofh[[1]])
#> Formal class 'adf_gof.class' [package "ReturnCurves"] with 6 slots
#>   ..@ adf      :Formal class 'adf_est.class' [package "ReturnCurves"] with 11 slots
#>   .. .. ..@ dataexp    : num [1:1408, 1:2] 0.594 0.415 0.369 0.29 0.747 ...
#>   .. .. ..@ w          : num [1:1001] 0 0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 ...
#>   .. .. ..@ method     : chr "hill"
#>   .. .. ..@ q          : num 0.95
#>   .. .. ..@ qalphas    : num [1:2] 0.95 0.95
#>   .. .. ..@ k          : num 7
#>   .. .. ..@ constrained: logi FALSE
#>   .. .. ..@ tol        : num 1e-04
#>   .. .. ..@ par_init   : num [1:6] 0 0 0 0 0 0
```

```
#>   .. .. ..@ interval   : num [1:2] 0 1
#>   .. .. ..@ adf        : num [1:1001] 1 0.999 0.998 0.997 0.996 0.995 0.994 0.993 0.992 0.991 ...
#>   ..@ ray      : num 0.25
#>   ..@ blocksize: num 10
#>   ..@ nboot    : num 250
#>   ..@ alpha    : num 0.05
#>   ..@ gof      :List of 4
#>   .. ..$ model    : num [1:70] 0.0142 0.0286 0.0432 0.058 0.073 ...
#>   .. ..$ empirical: num [1:70] 0.0177 0.0297 0.0393 0.0612 0.0918 ...
#>   .. ..$ lower    : num [1:70] 0.0177 0.0177 0.0177 0.0297 0.0297 ...
#>   .. ..$ upper    : num [1:70] 0.0393 0.1018 0.1929 0.1929 0.2016 ...

# head of the list elements of slot gof
head(gofh[[1]]@gof$model)
#> [1] 0.01418463 0.02857337 0.04317217 0.05798726 0.07302514 0.08829261
head(gofh[[1]]@gof$empirical)
#> [1] 0.01769958 0.02967577 0.03930040 0.06115842 0.09180755 0.10471095
head(gofh[[1]]@gof$lower)
#> [1] 0.01769958 0.01769958 0.01769958 0.02967577 0.02967577 0.03930040
head(gofh[[1]]@gof$upper)
#> [1] 0.0393004 0.1018077 0.1928653 0.1928653 0.2015994 0.2015994
```

As before, it is possible to plot an S4 object of `adf_gof.class` with `plot`, where the QQ plot with the model and empirical quantiles is shown. The points should lie close to the line $y = x$; for a good fit and agreement between these quantiles, the line $y = x$ should mainly lie within the $(1 - \alpha)\%$ tolerance intervals.

```
library(gridExtra)
grid.arrange(plot(gofh[[1]]), plot(gofh[[2]]), ncol = 2)
```



4 Estimation of the return curve

Given a probability p and the joint survivor function $\Pr(X > x, Y > y)$ of the bivariate vector (X, Y), the p-probability return curve is defined as

$$\mathrm{RC}(p) := \{(x, y) \in \mathbb{R}^2 : \Pr(X > x, Y > y) = p\}. \tag{6}$$

The interest lies in values of p close to 0 as these are the ones characterising rare joint exceedances events. Given any point $(x, y) \in \mathrm{RC}(p)$, the event $\{X > x, Y > y\}$ is expected to happen once each return period $1/p$, on average. This is equivalent to having an expected value of np points in the region $(x, \infty) \times (y, \infty)$ in a sample size of n from (X, Y).

Since the probability p is close to 0, methods that can accurately capture the behaviour of the joint tail are necessary in order to realistically extrapolate and estimate $\text{RC}(p)$ for values of p outside of the observation period. Murphy-Barltrop et al. [2023] consider a couple of methods to achieve this, one of which uses the ADF $\lambda(\omega)$ given in equation (2) to characterise the joint tail behaviour.

Estimation of $\text{RC}(p)$ is done with standard exponentially distributed variables; therefore, the first step is to transform the original data onto standard exponential margins using equation (1), and then, after estimation of $\text{RC}(p)$, back transform them onto the original margins. Estimates of $\text{RC}(p)$ are obtained through estimates of t and u from equation (3), and rays ω. In particular, the value of $t > 0$ can be obtained by first estimating u as the $(1-p^*)$-th quantile of T_ω, where $p^* > p$, is a small probability, and then ensuring that $\Pr(T_\omega > t+u) = p$. Since u is estimated as the $(1-p^*)$-th quantile of T_ω, we have that $\Pr(T_\omega > u) = p^*$; thus,

$$p = \Pr(T_\omega > t+u) = \Pr(T_\omega > u)\Pr(T_\omega > t+u \mid T_\omega > u) = p^* e^{-\hat{\lambda}(\omega)t},$$

which leads to $t = -\log(p/p^*)/\hat{\lambda}(\omega)$. Finally, the estimates of the return curve $\hat{\text{RC}}(p)$ can be obtained by setting $(x, y) := (\omega(t+u), (1-\omega)(t+u))$.

In the `ReturnCurves` package, estimation of the return curve is done through function `rc_est` which shares the same inputs as function `adf_est` with an additional argument `p` representing the curve survival probability. This probability value should be smaller than $1-q$, where q is the quantile for the min-projection variables T^1_ω, and, when applicable, smaller than $1-q_\alpha$, where q_α are the quantiles used in the conditional extremes method.

Function `rc_est` returns an S4 object of class `rc_est.class` with 14 attributes, with a list and a matrix in the last 2 slots:

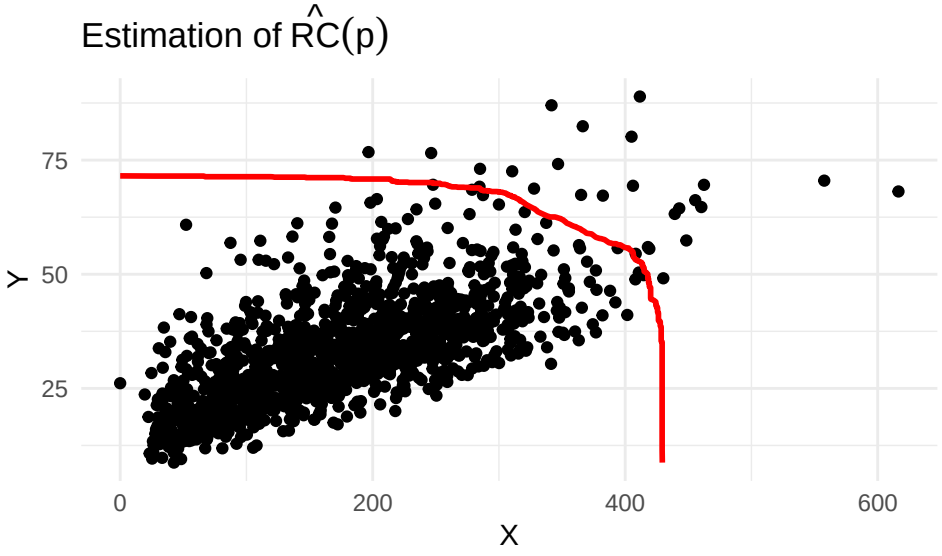
- `interval`: vector with the maximum likelihood estimates from the conditional extremes model $\hat{\alpha}^1_{x|y}$ and $\hat{\alpha}^1_{y|x}$ if `constrained = TRUE`. Otherwise, it returns the values 0 and 1; this has no meaningful interpretation as the estimation is performed in an unconstrained interval.
- `rc`: matrix with the estimates of the return curve on the original margins.

```
n <- dim(airdata)[1]
prob <- 10/n
# Estimation using Hill estimator without conditional extremes parameters
whill <- seq(0, 1, by = 0.001)
## q and constrained are set to the default values here
rch <- rc_est(margdata = expdata, w = whill, p = prob, method = "hill",
              q = 0.95, constrained = F)

# Estimation using Hill estimator with conditional extremes parameters
## q and qalphas are set to the default values
rch2 <- rc_est(margdata = expdata, w = whill, p = prob, method = "hill", q = 0.95,
               qalphas = rep(0.95, 2), constrained = T)

# Estimation using CL method without conditional extremes parameters
## w, q and constrained are set to the default values here
rccl <- rc_est(margdata = expdata, w = seq(0, 1, by = 0.01), p = prob, method = "cl",
               q = 0.95, constrained = F)

# Estimation using CL method with conditional extremes parameters
## w, q and qalphas are set to the default values
rccl2 <- rc_est(margdata = expdata, w = seq(0, 1, by = 0.01), p = prob, method = "cl",
                q = 0.95, qalphas = rep(0.95, 2), constrained = T)

# attributes of the S4 object
str(rch)
#> Formal class 'rc_est.class' [package "ReturnCurves"] with 14 slots
```

```
#>   ..@ data            : num [1:1408, 1:2] 154 132 120 105 175 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   .. .. ..$ : NULL
#>   .. .. ..$ : chr [1:2] "nox" "pm10"
#>   ..@ qmarg           : num [1:2] 0.95 0.95
#>   ..@ constrainedshape: logi TRUE
#>   ..@ w               : num [1:1001] 0 0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 ...
#>   ..@ p               : num 0.00701
#>   ..@ method          : chr "hill"
#>   ..@ q               : num 0.95
#>   ..@ qalphas         : num [1:2] 0.95 0.95
#>   ..@ k               : num 7
#>   ..@ constrained     : logi FALSE
#>   ..@ tol             : num 0.001
#>   ..@ par_init        : num [1:6] 0 0 0 0 0 0
#>   ..@ interval        : num [1:2] 0 1
#>   ..@ rc              : num [1:1001, 1:2] 0 26.2 29.6 33.1 34.2 ...

# head of the vector with adf estimates for the first estimator
head(rch@rc)
#>          [,1]     [,2]
#> [1,]  0.00000 71.54081
#> [2,] 26.19885 71.50107
#> [3,] 29.61038 71.50107
#> [4,] 33.08634 71.50107
#> [5,] 34.20411 71.50107
#> [6,] 36.17626 71.50107
```

It is possible to plot an S4 object of `rc_est.class` with `plot`, where the original data is plotted with the estimated line for the return curve $\hat{\mathrm{RC}}(p)$.

```
# plot of the ADF estimation based on the unconstrained Hill estimator
plot(rch)
```



4.1 Uncertainty of the return curve estimates

Murphy-Barltrop et al. [2023] propose a procedure to assess the uncertainty of the return curve estimates.

For large positive $m \in \mathbb{N}$, let

$$\boldsymbol{\Theta} := \left\{ \frac{\pi(m+1-j)}{2(m+1)} \mid 1 \leq j \leq m \right\}, \tag{7}$$

define a set of angles. For each $\theta \in \boldsymbol{\Theta}$, the line $L_\theta := \{(x, y) \in \mathbb{R}^2_+ \mid \tan(\theta) > 0\}$ intersects the estimated $\hat{\mathrm{RC}}(p)$ exactly once, i.e. $\{(\hat{x}_\theta, \hat{y}_\theta)\} := \hat{\mathrm{RC}}(p) \cap L_\theta$ where $(\hat{x}_\theta, \hat{y}_\theta) \in \hat{\mathrm{RC}}(p)$. Moreover, let $\hat{d}_\theta := \left(\hat{x}^2_\theta + \hat{y}^2_\theta\right)^{1/2}$ denote the L_2-norm of the point estimate. Uncertainty in the return curve estimates is quantified using the distribution of $\hat{d}_\theta$ at each angle $\theta \in \boldsymbol{\Theta}$ as follows: for $k = 1, \ldots,$ `nboot`:

1. Bootstrap the original data set; when temporal dependence is present, a block bootstrap should be used.
2. For each $\theta \in \boldsymbol{\Theta}$, obtain $\hat{d}_{\theta,k}$ for the corresponding return curve estimate.

Finally, given $\theta \in \boldsymbol{\Theta}$, empirical estimates of the mean, median and $(1-\alpha)\%$ confidence intervals for $\hat{d}_\theta$ can be obtained using the sample of $\hat{d}_{\theta,k}$. These are available through function `rc_unc`, which takes as inputs:

- `retcurve`: an S4 object of class `rc_est.class` containing the return curve estimates,
- `blocksize`: size of blocks for the block bootstrap procedure; if no temporal dependence is present, then set `blocksize = 1` (default),
- `nboot`: number of bootstrap samples to be taken,
- `nangles`: number of angles m,
- `alpha`: significance level to compute the $(1-\alpha)\%$ confidence intervals.

Function `rc_unc` returns an S4 object of class `rc_unc.class` with 6 attributes, where the last slot `unc` contains a list with:

- `median`: a vector containing the empirical estimates of the median return curve
- `mean`: a vector containing the empirical estimates of the mean return curve
- `lower`: a vector containing the lower bound of the confidence interval
- `upper`: a vector containing the upper bound of the confidence interval

For simplicity, just the uncertainty of the return curve obtained using the unconstrained Hill estimator is computed here.

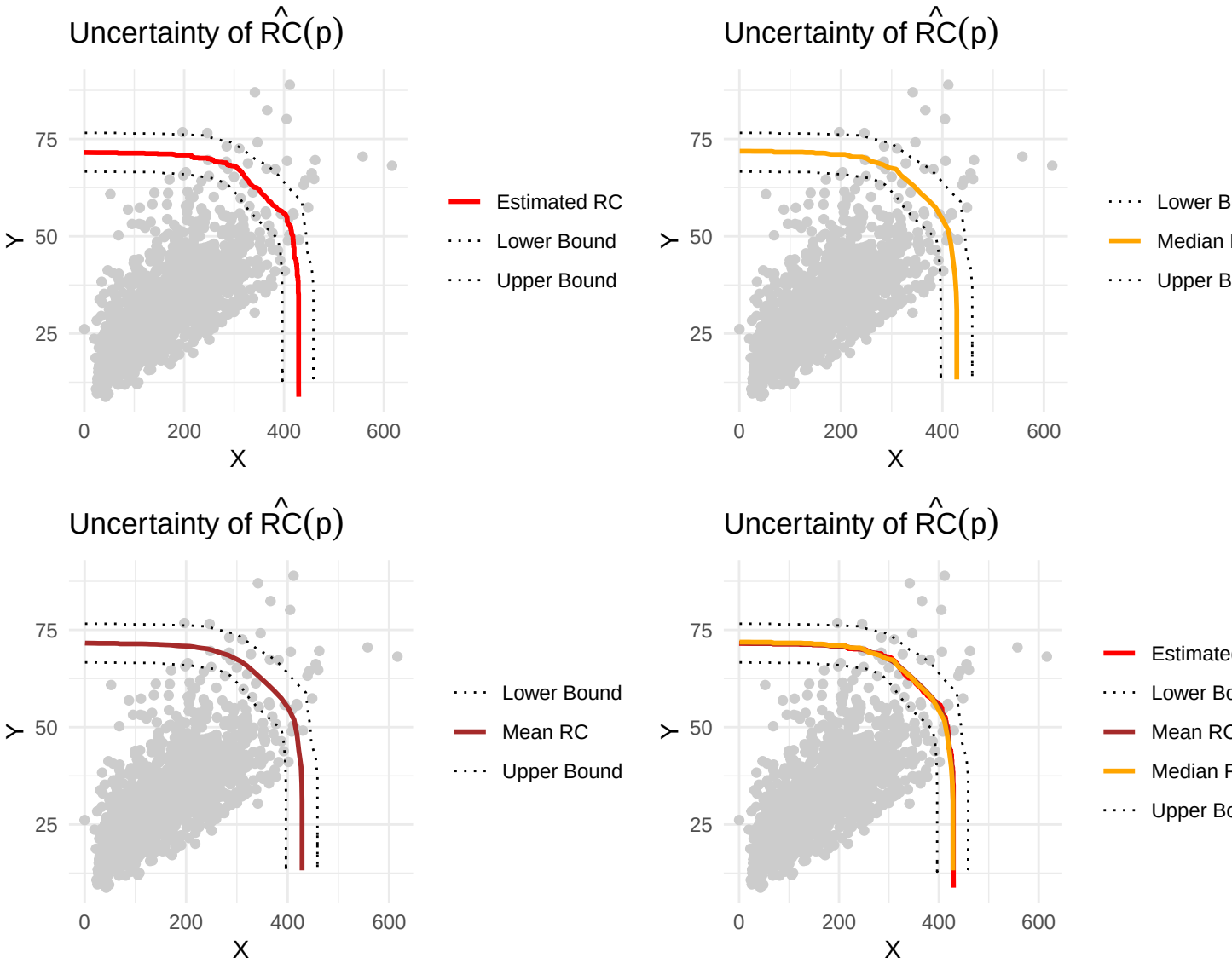
```
# nangles and alpha set to default
# nboot set to 50 for simplicity
# blocksize is set to 10 to account for temporal dependence
rch_unc <- rc_unc(rch, blocksize = 10, nboot = 50, nangles = 150, alpha = 0.05)

# attributes of the S4 object
str(rch_unc)
#> Formal class 'rc_unc.class' [package "ReturnCurves"] with 6 slots
#>   ..@ retcurve :Formal class 'rc_est.class' [package "ReturnCurves"] with 14 slots
#>   .. .. ..@ data            : num [1:1408, 1:2] 154 132 120 105 175 ...
#>   .. .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. .. ..$ : NULL
#>   .. .. .. .. ..$ : chr [1:2] "nox" "pm10"
#>   .. .. ..@ qmarg           : num [1:2] 0.95 0.95
#>   .. .. ..@ constrainedshape: logi TRUE
#>   .. .. ..@ w               : num [1:1001] 0 0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 .
#>   .. .. ..@ p               : num 0.00701
#>   .. .. ..@ method          : chr "hill"
#>   .. .. ..@ q               : num 0.95
#>   .. .. ..@ qalphas         : num [1:2] 0.95 0.95
#>   .. .. ..@ k               : num 7
#>   .. .. ..@ constrained     : logi FALSE
#>   .. .. ..@ tol             : num 0.001
```

```
#>   .. .. ..@ par_init        : num [1:6] 0 0 0 0 0 0
#>   .. .. ..@ interval        : num [1:2] 0 1
#>   .. .. ..@ rc              : num [1:1001, 1:2] 0 26.2 29.6 33.1 34.2 ...
#>   ..@ blocksize: num 10
#>   ..@ nboot    : num 50
#>   ..@ nangles  : num 150
#>   ..@ alpha    : num 0.05
#>   ..@ unc      :List of 4
#>   .. ..$ median: num [1:150, 1:2] 0.657 1.313 1.97 2.628 3.286 ...
#>   .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. ..$ : chr [1:150] "50%" "50%" "50%" "50%" ...
#>   .. .. .. ..$ : NULL
#>   .. ..$ mean  : num [1:150, 1:2] 0.654 1.308 1.962 2.617 3.272 ...
#>   .. ..$ lower : num [1:150, 1:2] 0.602 1.205 1.808 2.411 3.014 ...
#>   .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. ..$ : chr [1:150] "2.5%" "2.5%" "2.5%" "2.5%" ...
#>   .. .. .. ..$ : NULL
#>   .. ..$ upper : num [1:150, 1:2] 0.706 1.411 2.118 2.824 3.531 ...
#>   .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. ..$ : chr [1:150] "97.5%" "97.5%" "97.5%" "97.5%" ...
#>   .. .. .. ..$ : NULL

# head of the list elements of slot unc
head(rch_unc@unc$median)
#>          [,1]     [,2]
#> 50% 0.6565469 71.86131
#> 50% 1.3132359 71.86131
#> 50% 1.9702093 71.86131
#> 50% 2.6276096 71.86131
#> 50% 3.2855795 71.86131
#> 50% 3.9442626 71.86131
head(rch_unc@unc$mean)
#>           [,1]     [,2]
#> [1,] 0.6539659 71.61320
#> [2,] 1.3080491 71.61204
#> [3,] 1.9623911 71.61087
#> [4,] 2.6171339 71.60969
#> [5,] 3.2724197 71.60852
#> [6,] 3.9283912 71.60735
head(rch_unc@unc$lower)
#>           [,1]     [,2]
#> 2.5% 0.6023811 66.65456
#> 2.5% 1.2048819 66.65404
#> 2.5% 1.8076328 66.65353
#> 2.5% 2.4107646 66.65301
#> 2.5% 3.0144082 66.65249
#> 2.5% 3.6186953 66.65197
head(rch_unc@unc$upper)
#>            [,1]   [,2]
#> 97.5% 0.7056667 76.583
#> 97.5% 1.4114862 76.583
#> 97.5% 2.1176113 76.583
#> 97.5% 2.8241952 76.583
```

```
#> 97.5% 3.5313915 76.583
#> 97.5% 4.2393542 76.583
```

It is possible to plot an instance of the S4 class `rc_unc.class` with function `plot`; this takes the S4 object and an extra argument `which` as inputs. If `which = "rc"` (default), then the estimated return curve is plotted, setting `which = "median"` shows the empirical median estimates of the return curve, while setting `which = "mean"` shows the empirical mean estimates of the return curve. All plots show the uncertainty associated with the estimated return curve in dashed lines. Finally, by setting `which = "all"`, plots the estimated return curve, the empirical median and mean estimates and the associated uncertainty.

```
library(gridExtra)
grid.arrange(plot(rch_unc, which = "rc"), plot(rch_unc, which = "median"),
             plot(rch_unc, which = "mean"), plot(rch_unc, which = "all"), nrow = 2)
```



4.2 Goodness-of-fit of the return curve estimates

It is important to assess the goodness-of-fit of the return curve estimates, given that the true return curve is unknown in reality. This is implemented in the `ReturnCurves` package based on the approach proposed by Murphy-Barltrop et al. [2023].

Given the return curve $\text{RC}(p)$, the probability of lying in a survival region $(x, \infty) \times (y, \infty)$ is p. Given the same set of angles $\boldsymbol{\Theta}$ as in equation (7), for each $\theta_j \in \boldsymbol{\Theta}$, the empirical probability $\hat{p}_j$ of lying in $(\hat{x}_{\theta_j}, \infty) \times (\hat{y}_{\theta_j}, \infty)$, where $(\hat{x}_{\theta_j}, \hat{y}_{\theta_j})$ is the corresponding point in $\hat{\text{RC}}(p)$, is given by the proportion of points in that region. The goodness-of-fit of the estimated return curve is then assessed via a bootstrap procedure; for each angle $\theta_j \in \boldsymbol{\Theta}$, the original data set is bootstrapped and empirical probability estimates $\hat{p}_j$ are obtained. When temporal dependence is present in the data, a block bootstrap approach should be taken and the size of the blocks must be defined. We note that for each j, `nboot` empirical probabilities are estimated and, so the median and the $(1-\alpha)\%$ pointwise confidence intervals for the probabilities can be obtained by taking the 50%, $(\alpha/2)\%$ and $(1-\alpha/2)\%$ quantiles of the set of empirical probabilities for each j, respectively.

The goodness-of-fit for an estimated return curve is implemented through function `rc_gof`. This shares the same input arguments as the `rc_unc` function and returns an S4 object with 5 attributes with the last slot `gof` containing a list with:

- `median`: a vector with the median of the empirical probabilities,
- `lower`: a vector with the lower bound of the confidence interval,
- `upper`: a vector with the upper bound of the confidence interval.

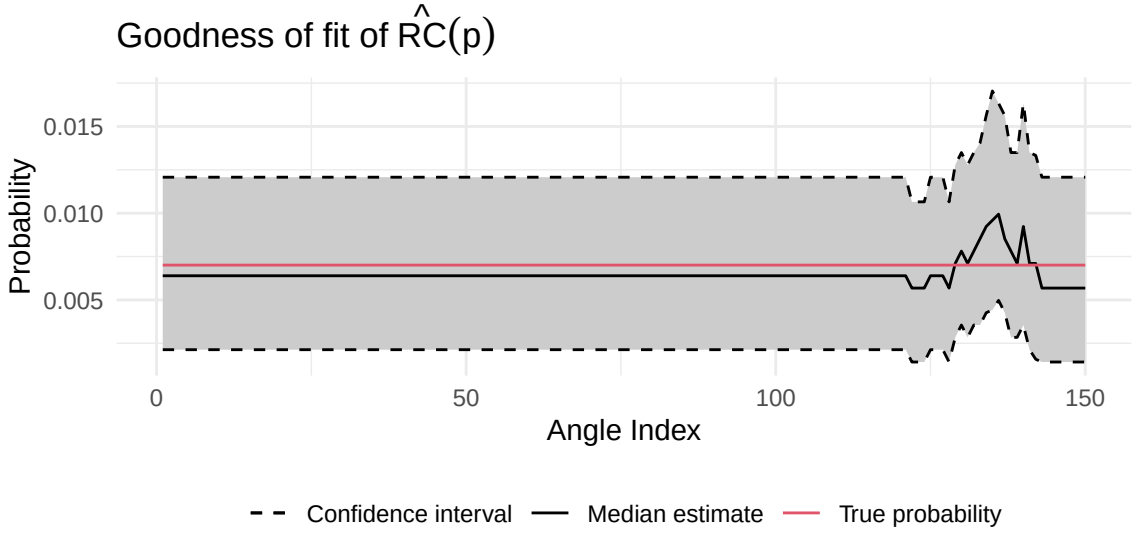
For simplicity, just the goodness-of-fit of the return curve obtained using the unconstrained Hill estimator is computed here.

```
# nboot, nangles and alpha set to default
# blocksize is set to 10 to account for temporal dependence
rch_gof <- rc_gof(rch, blocksize = 10, nboot = 250, nangles = 150, alpha = 0.05)

# attributes of the S4 object
str(rch_gof)
#> Formal class 'rc_gof.class' [package "ReturnCurves"] with 5 slots
#>   ..@ retcurve :Formal class 'rc_est.class' [package "ReturnCurves"] with 14 slots
#>   .. .. ..@ data            : num [1:1408, 1:2] 154 132 120 105 175 ...
#>   .. .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. .. ..$ : NULL
#>   .. .. .. .. ..$ : chr [1:2] "nox" "pm10"
#>   .. .. ..@ qmarg           : num [1:2] 0.95 0.95
#>   .. .. ..@ constrainedshape: logi TRUE
#>   .. .. ..@ w               : num [1:1001] 0 0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 .
#>   .. .. ..@ p               : num 0.00701
#>   .. .. ..@ method          : chr "hill"
#>   .. .. ..@ q               : num 0.95
#>   .. .. ..@ qalphas         : num [1:2] 0.95 0.95
#>   .. .. ..@ k               : num 7
#>   .. .. ..@ constrained     : logi FALSE
#>   .. .. ..@ tol             : num 0.001
#>   .. .. ..@ par_init        : num [1:6] 0 0 0 0 0 0
#>   .. .. ..@ interval        : num [1:2] 0 1
#>   .. .. ..@ rc              : num [1:1001, 1:2] 0 26.2 29.6 33.1 34.2 ...
#>   ..@ blocksize: num 10
#>   ..@ nboot    : num 250
#>   ..@ alpha    : num 0.05
#>   ..@ gof      :List of 3
#>   .. ..$ median: Named num [1:150] 0.00639 0.00639 0.00639 0.00639 0.00639 ...
```

```
#>   .. .. ..- attr(*, "names")= chr [1:150] "50%" "50%" "50%" "50%" ...
#>   .. ..$ lower : Named num [1:150] 0.00213 0.00213 0.00213 0.00213 0.00213 ...
#>   .. .. ..- attr(*, "names")= chr [1:150] "2.5%" "2.5%" "2.5%" "2.5%" ...
#>   .. ..$ upper : Named num [1:150] 0.0121 0.0121 0.0121 0.0121 0.0121 ...
#>   .. .. ..- attr(*, "names")= chr [1:150] "97.5%" "97.5%" "97.5%" "97.5%" ...

# head of the list elements of slot gof
head(rch_gof@gof$median)
#>         50%         50%         50%         50%         50%         50%
#> 0.006392045 0.006392045 0.006392045 0.006392045 0.006392045 0.006392045
head(rch_gof@gof$lower)
#>        2.5%        2.5%        2.5%        2.5%        2.5%        2.5%
#> 0.002130682 0.002130682 0.002130682 0.002130682 0.002130682 0.002130682
head(rch_gof@gof$upper)
#>      97.5%      97.5%      97.5%      97.5%      97.5%      97.5%
#> 0.01207386 0.01207386 0.01207386 0.01207386 0.01207386 0.01207386
```

It is possible to plot an instance of the S4 class `rc_gof.class` with function `plot`, where a comparison between the true probability p (in red) and the empirical median estimates (in black) is shown. Ideally, p should be contained in the confidence region, shaded in grey. Finally, in practice, the value of p should be within the range of the data and not too extreme, given the nature of empirical probabilities.

```
plot(rch_gof)
```



References

S. G. Coles and J. A. Tawn. Modelling extreme multivariate events. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(2):377–392, 1991. ISSN 00359246. doi: 10.1111/j.2517-6161.1991.tb01830.x.

Stuart Coles, Janet Heffernan, and Jonathan Tawn. Dependence measures for extreme value analyses. *Extremes*, 2:339–365, 1999.

J. E. Heffernan and J. A. Tawn. A conditional approach for multivariate extreme values (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):497–546, 2004. doi: https://doi.org/10.1111/j.1467-9868.2004.02050.x.

B. M. Hill. A Simple General Approach to Inference About the Tail of a Distribution. *The Annals of Statistics*, 3(5):1163 – 1174, 1975. doi: 10.1214/aos/1176343247. URL https://doi.org/10.1214/aos/1176343247.

C. J. R. Murphy-Barltrop, J. L. Wadsworth, and E. F. Eastoe. New estimation methods for extremal bivariate return curves. *Environmetrics*, 34, 8 2023. ISSN 1099095X. doi: 10.1002/env.2797.

C. J. R. Murphy-Barltrop, J. L. Wadsworth, and E. F. Eastoe. Improving estimation for asymptotically independent bivariate extremes via global estimators for the angular dependence function, 2024.

J. L. Wadsworth and J. A. Tawn. A new representation for multivariate tail probabilities. *Bernoulli*, 19: 2689–2714, 11 2013. ISSN 13507265. doi: 10.3150/12-BEJ471.