Return Curves Estimation

Lídia André, Callum Murphy-Barltrop, Jennifer Wadsworth

2024-06-14

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 . . . data *need to find better data yet*

```
set.seed(321)
data <- cbind(rnorm(1000), rnorm(1000))
```

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 :

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

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 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` as inputs.

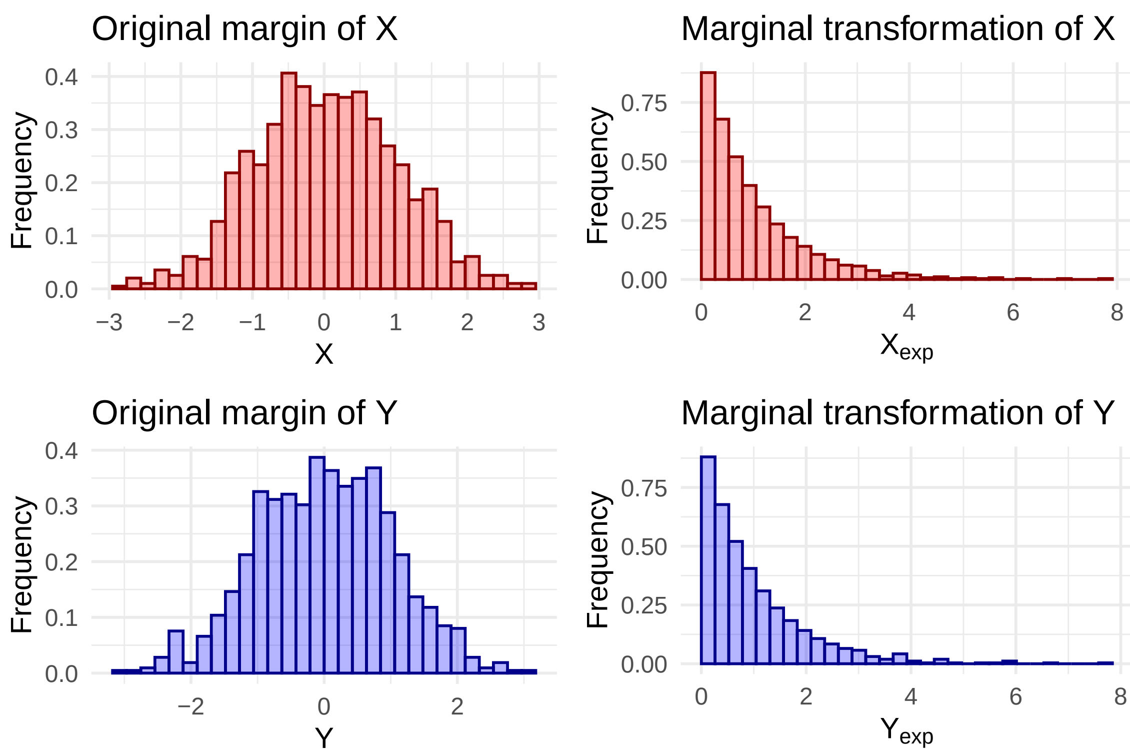
Function `margtransf` returns an object of S4 class `margtrasnf.class` with six attributes:

- `data`: matrix with the data on the original margins
- `qmarg`: vector of marginal quantiles used to fit the GPD
- `constrainedshape`: whether $\xi > -1$ or $\xi \in \mathbb{R}$
- `parameters`: matrix containing parameters (σ, ξ)
- `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 = data, 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 [1:1000, 1:2] 1.705 -0.712 -0.278 -0.12 -0.124 ...
#>   ..@ qmarg           : num [1:2] 0.95 0.95
#>   ..@ constrainedshape: logi TRUE
#>   ..@ parameters      : num [1:2, 1:2] 0.505 -0.303 0.398 -0.104
#>   ..@ thresh          : num [1:2] 1.65 1.69
#>   ..@ dataexp         : num [1:1000, 1:2] 3.101 0.266 0.489 0.602 0.599 ...

# head of the data on standard exponential margins
head(expdata@dataexp)
#>           [,1]       [,2]
#> [1,] 3.1008831 0.06500483
#> [2,] 0.2662680 0.09971547
#> [3,] 0.4887599 2.43141796
#> [4,] 0.6024795 0.69414668
#> [5,] 0.5988365 3.55115450
#> [6,] 0.8974876 0.13911280
```

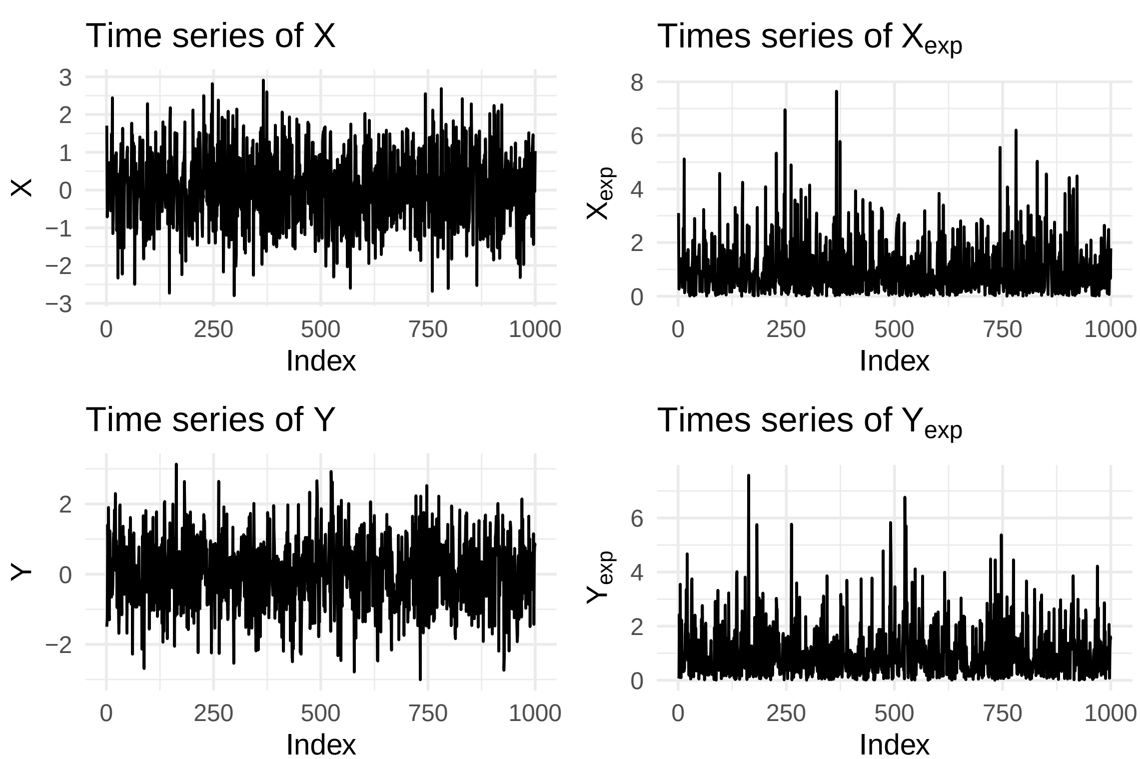
It is possible to plot an S4 object of `margtrasnf.class` with `plot`. By setting argument `which = "hist"`, histograms of each variable on original and standard exponential margins can be seen:

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



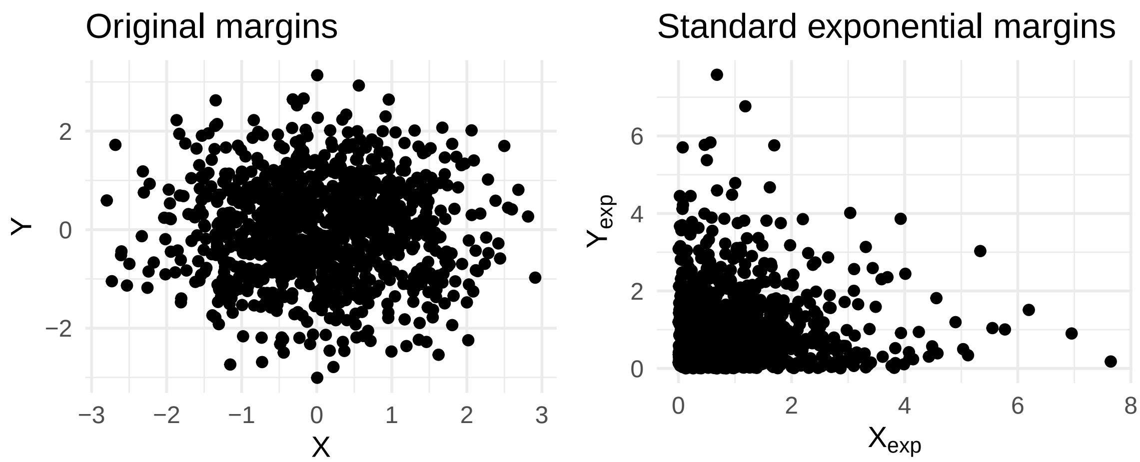
To visualise the time series of each variable on original and standard exponential margins, we need to set `which = "ts"`:

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



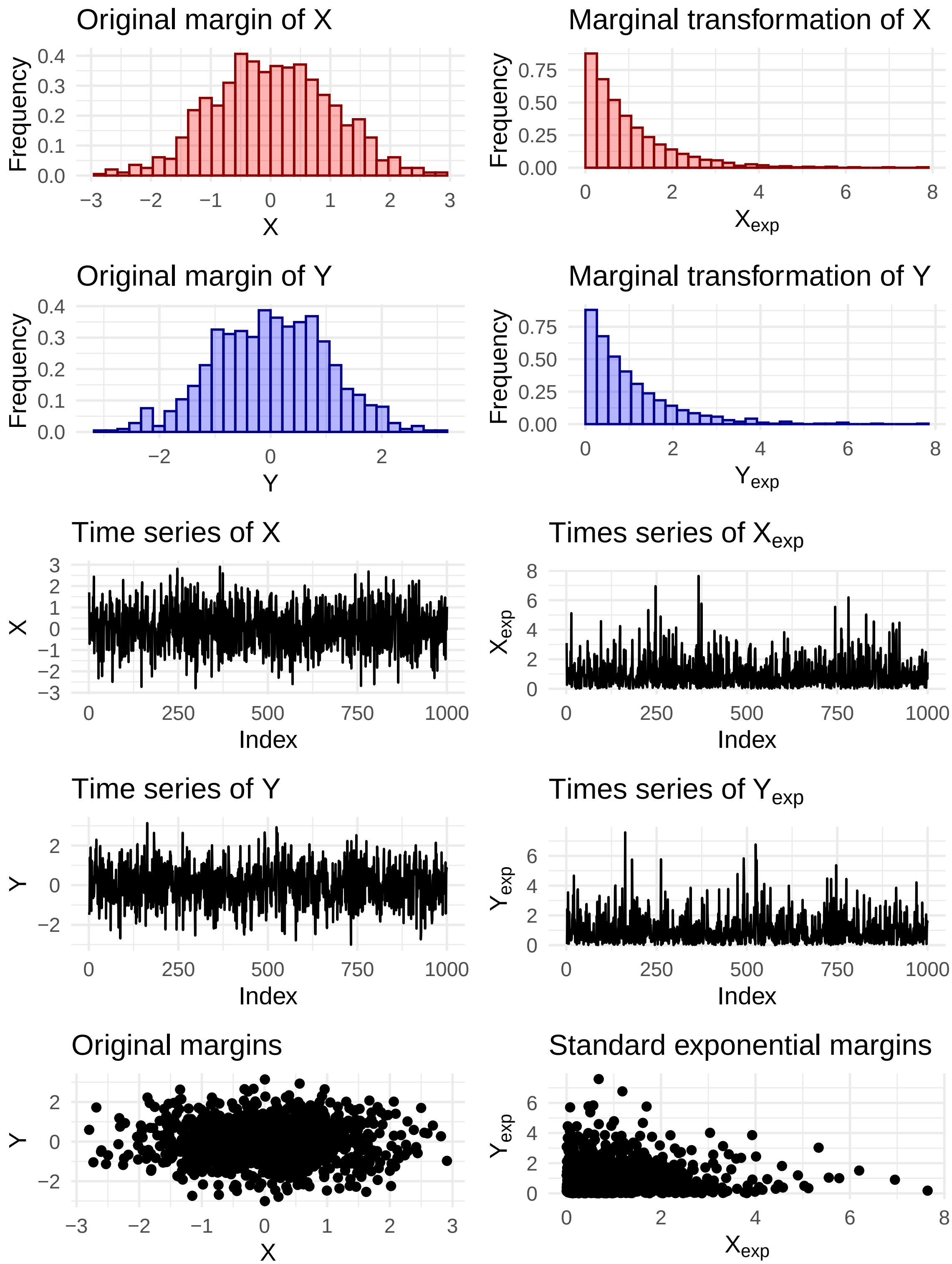
The joint distribution on original and standard exponential margins can be access with `which = "joint"`:

```
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)
```



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. A few methods, such as the one introduced by Wadsworth and Tawn [2013], aim at characterising the joint tail behaviour in both scenarios are available in the literature. Given standard exponentially distributed variables X_E and Y_E and a slowly varying function $\mathcal{L}(\cdot;\omega)$ at infinity,s 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, e.g., 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:

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

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 $\mid \boldsymbol{t}_\omega^1 \mid$ 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:

$$\mathcal{B}_k^1 = \left\{ (1-\hat{\alpha}_{x|y}^1)\left(1-\frac{v-\hat{\alpha}_{x|y}^1}{\hat{\alpha}_{y|x}^1-\hat{\alpha}_{x|y}^1}\right)^k + \sum_{i=1}^{k-1}\beta_i\binom{k}{i}\left(\frac{v-\hat{\alpha}_{x|y}^1}{\hat{\alpha}_{y|x}^1-\hat{\alpha}_{x|y}^1}\right)^i\left(1-\frac{v-\hat{\alpha}_{x|y}^1}{\hat{\alpha}_{y|x}^1-\hat{\alpha}_{x|y}^1}\right)^{k-i} + \right.$$

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

$\lambda(\omega) = \lambda(\omega; \boldsymbol{\beta})$ is assumed to be represented by an element of $\mathcal{B}_k^1$ on $\left[\hat{\alpha}_{x|y}^1, \hat{\alpha}_{y|x}^1\right]$.

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, marginal quantiles to fit the conditional extremes method if `constrained=T`, the polynomial degree k, the convergence tolerance and the initial values for $\boldsymbol{\beta}$ for the composite maximum likelihood procedure. Finally, due to its pointwise nature, a finer grid for ω when estimating the ADF using the Hill estimator is recommended.

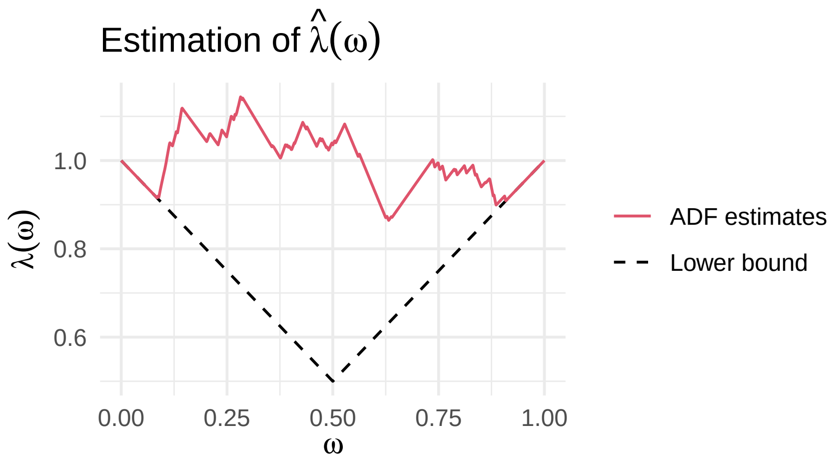
Function `adf_est` returns an object of S4 class `adf_est.class` with ten attributes, where the first nine are the inputs of the function and the last is a vector `adf` containing the estimates of $\lambda(\omega)$.

```
# 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 [1:1000, 1:2] 3.101 0.266 0.489 0.602 0.599 ...
#>   ..@ 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 $T^1_\omega = (T_\omega - u_\omega \mid T_\omega > u_\omega) \sim \text{Exp}(\lambda(\omega)) \Leftrightarrow \lambda(\omega)T^1_\omega \sim \text{Exp}(1)$ as $u_\omega \to \infty$, 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 $\left(F_E^{-1}(i/(n+1), T^1_{(i)}\right)$, where F_E^{-1} denotes the inverse of the cumulative distribution of a standard exponential distribution and $T^{-1}_{(i)}$ is the i-th ordered increasing statistic, $i = 1, \dots, n$. 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 (`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`, 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.

```r
# Goodness of fit of the adf for three rays w
rays <- c(0.25, 0.5, 0.75)
## blocksize, nboot and alpha are set to the default values
gofh <- sapply(rays, adf_gof, adf = lambdah, blocksize = 1, 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 [1:1000, 1:2] 3.101 0.266 0.489 0.602 0.599 ...
#>   .. .. ..@ 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 1
#>   ..@ nboot    : num 250
#>   ..@ alpha    : num 0.05
#>   ..@ gof      :List of 4
#>   .. ..$ model    : num [1:50] 0.0198 0.04 0.0606 0.0817 0.1032 ...
#>   .. ..$ empirical: num [1:50] 0.026 0.054 0.112 0.126 0.141 ...
#>   .. ..$ lower    : num [1:50] 0.026 0.026 0.026 0.026 0.054 ...
#>   .. ..$ upper    : num [1:50] 0.112 0.141 0.235 0.283 0.302 ...

# head of the list elements of slot gof
head(gofh[[1]]@gof$model)
#> [1] 0.01980263 0.04000533 0.06062462 0.08167803 0.10318424 0.12516314
head(gofh[[1]]@gof$empirical)
#> [1] 0.02601951 0.05395085 0.11152974 0.12629937 0.14122529 0.21832026
head(gofh[[1]]@gof$lower)
#> [1] 0.02601951 0.02601951 0.02601951 0.02601951 0.05395085 0.11152974
head(gofh[[1]]@gof$upper)
#> [1] 0.1115297 0.1412253 0.2351376 0.2831817 0.3022290 0.3205085
```

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 are shown. The points should lie close to the line $y = x$; for a good fit and agreement between these quantile, line $y = x$ should mainly lie within the $(1 - \alpha)\%$ tolerance intervals.

```r
grid.arrange(plot(gofh[[1]]), plot(gofh[[2]]), plot(gofh[[3]]), ncol = 3)
```

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. In addition, given any point $(x, y) \in \mathrm{RC}(p)$, event $\{X > x, Y > y\}$ is expected to happen once each return period $1/p$, on average. This is equivalent to observing np points in the region $(x, \infty) \times (y, \infty)$ in a sample size of n from (X, Y).

Since 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 $\mathrm{RC}(p)$ 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 $\mathrm{RC}(p)$ 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 $\mathrm{RC}(p)$, back transform them onto the original margins. Estimates of $\mathrm{RC}(p)$ 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 $\hat{\mathrm{RC}}(p)$ 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 marginal quantile for the min-projection variable T^1, and, when applicable, smaller than $1 - q_\alpha$, where q_α are the marginal quantiles used in the conditional extremes method.

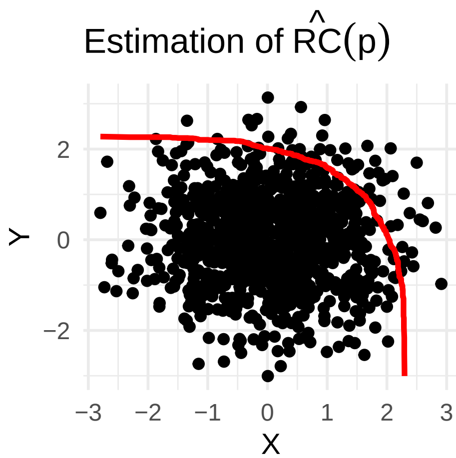
Function `rc_est` returns an S4 object of class `rc_est.class` with twelve attributes, where the last slot `rc` contains a matrix with the estimates of the return curve on the original margins.

```
n <- dim(data)[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",
```

```r
              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 13 slots
#>   ..@ data       : num [1:1000, 1:2] 1.705 -0.712 -0.278 -0.12 -0.124 ...
#>   ..@ qmarg      : num [1:2] 0.95 0.95
#>   ..@ 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 0.01
#>   ..@ 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] -2.8 -2.56 -2.3 -2.08 -1.97 ...

# head of the vector with adf estimates for the first estimator
head(rch@rc)
#>           [,1]     [,2]
#> [1,] -2.797440 2.275493
#> [2,] -2.560148 2.269500
#> [3,] -2.299140 2.262939
#> [4,] -2.083273 2.262939
#> [5,] -1.967764 2.262939
#> [6,] -1.873685 2.262939
```

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 $\hat{\mathrm{RC}}(p)$.

```r
# 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 $\theta \in \boldsymbol{\Theta}$, 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`,
- `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 six 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.

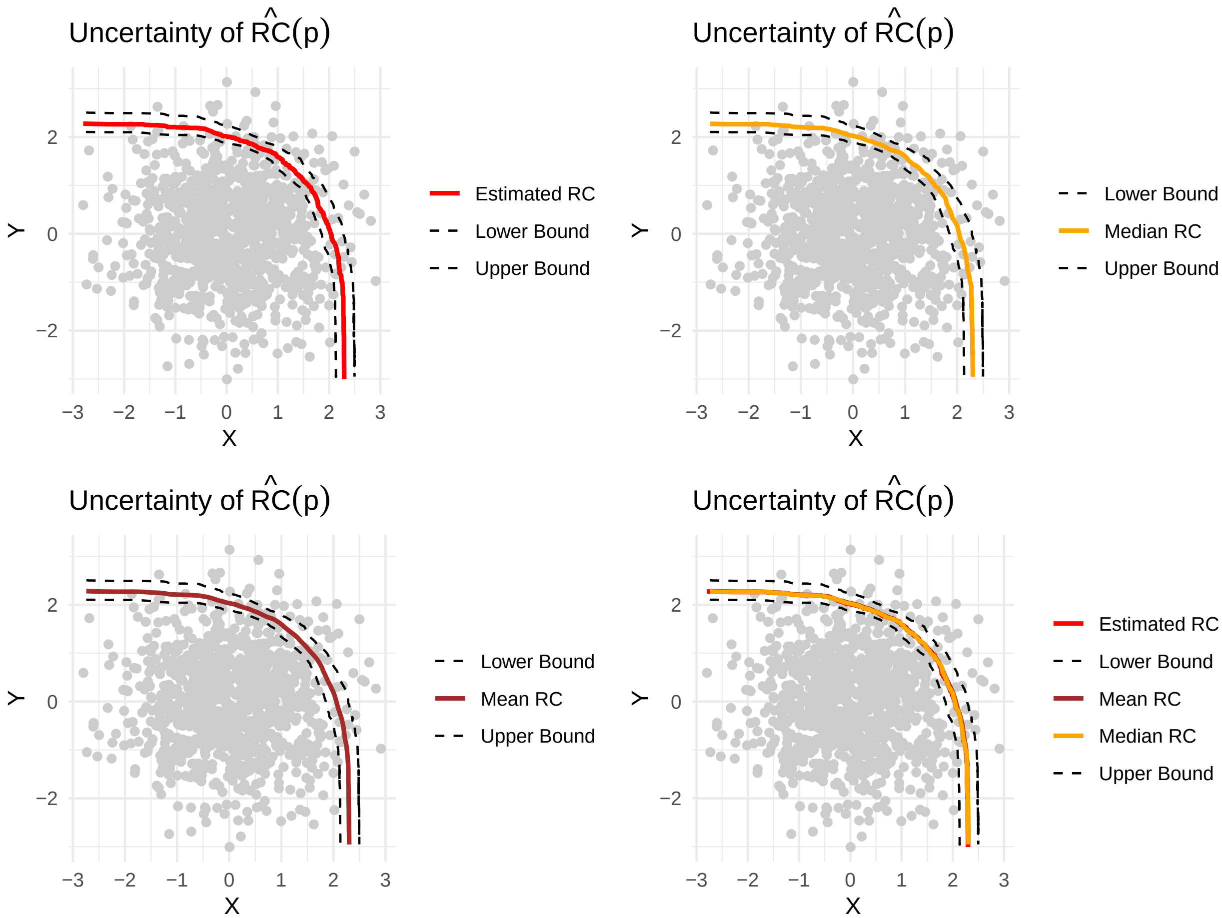
```
# blocksize, nangles and alpha set to default
# nboot set to 100 for computational time < 5s
```

```r
rch_unc <- rc_unc(rch, blocksize = 1, nboot = 100, 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 13 slots
#>   .. .. ..@ data       : num [1:1000, 1:2] 1.705 -0.712 -0.278 -0.12 -0.124 ...
#>   .. .. ..@ qmarg      : num [1:2] 0.95 0.95
#>   .. .. ..@ 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 0.01
#>   .. .. ..@ 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] -2.8 -2.56 -2.3 -2.08 -1.97 ...
#>   ..@ blocksize: num 1
#>   ..@ nboot    : num 100
#>   ..@ nangles  : num 150
#>   ..@ alpha    : num 0.05
#>   ..@ unc      :List of 4
#>   .. ..$ median: num [1:150, 1:2] -2.74 -2.69 -2.63 -2.58 -2.52 ...
#>   .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. ..$ : chr [1:150] "50%" "50%" "50%" "50%" ...
#>   .. .. .. ..$ : NULL
#>   .. ..$ mean  : num [1:150, 1:2] -2.74 -2.69 -2.63 -2.58 -2.52 ...
#>   .. ..$ lower : num [1:150, 1:2] -2.74 -2.69 -2.64 -2.58 -2.53 ...
#>   .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. ..$ : chr [1:150] "2.5%" "2.5%" "2.5%" "2.5%" ...
#>   .. .. .. ..$ : NULL
#>   .. ..$ upper : num [1:150, 1:2] -2.74 -2.68 -2.63 -2.57 -2.51 ...
#>   .. .. ..- 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$median)
#>          [,1]     [,2]
#> 50% -2.742524 2.272356
#> 50% -2.687627 2.270863
#> 50% -2.632740 2.269283
#> 50% -2.577874 2.267126
#> 50% -2.522998 2.265119
#> 50% -2.467979 2.265119
head(rch_unc@unc$mean)
#>           [,1]     [,2]
#> [1,] -2.742432 2.281187
#> [2,] -2.687452 2.279277
#> [3,] -2.632504 2.276845
#> [4,] -2.577555 2.274800
```

```r
#> [5,] -2.522557 2.273586
#> [6,] -2.467499 2.272795
head(rch_unc@unc$lower)
#>           [,1]     [,2]
#> 2.5% -2.744281 2.103411
#> 2.5% -2.691119 2.103049
#> 2.5% -2.637941 2.102687
#> 2.5% -2.584766 2.101588
#> 2.5% -2.531622 2.099461
#> 2.5% -2.478347 2.099226
head(rch_unc@unc$upper)
#>            [,1]     [,2]
#> 97.5% -2.740098 2.505498
#> 97.5% -2.682821 2.501814
#> 97.5% -2.625497 2.501292
#> 97.5% -2.568185 2.499856
#> 97.5% -2.510861 2.498252
#> 97.5% -2.453450 2.497592
```

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.

```r
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 $\mathrm{RC}(p)$, 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 $\hat{\mathrm{RC}}(p)$, 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 are 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 six attributes with the last slot `gof` containing a list with

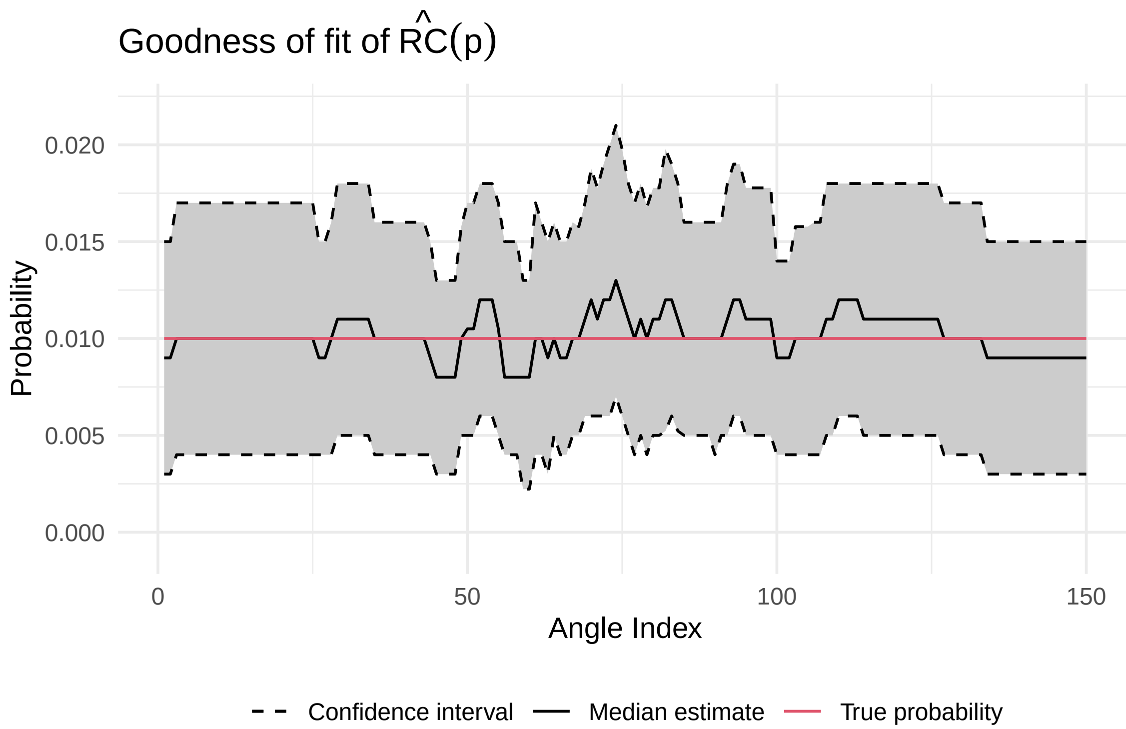
- `median`: a vector with the median of the empirical probabilities,
- `lower`: a vector with the lower bound of the confidence interval,
- `upper`: a vector with the upper bound of the confidence interval.

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

```r
# blocksize, nboot, nangles and alpha set to default
rch_gof <- rc_gof(rch, blocksize = 1, 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 13 slots
#>   .. .. ..@ data       : num [1:1000, 1:2] 1.705 -0.712 -0.278 -0.12 -0.124 ...
#>   .. .. ..@ qmarg      : num [1:2] 0.95 0.95
#>   .. .. ..@ 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 0.01
#>   .. .. ..@ 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] -2.8 -2.56 -2.3 -2.08 -1.97 ...
#>   ..@ blocksize: num 1
#>   ..@ nboot    : num 250
#>   ..@ alpha    : num 0.05
#>   ..@ gof      :List of 3
#>   .. ..$ median: Named num [1:150] 0.009 0.009 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ...
#>   .. .. ..- attr(*, "names")= chr [1:150] "50%" "50%" "50%" "50%" ...
#>   .. ..$ lower : Named num [1:150] 0.003 0.003 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 ...
#>   .. .. ..- attr(*, "names")= chr [1:150] "2.5%" "2.5%" "2.5%" "2.5%" ...
#>   .. ..$ upper : Named num [1:150] 0.015 0.015 0.017 0.017 0.017 0.017 0.017 0.017 0.017 0.017 ...
#>   .. .. ..- 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$median)
#>   50%   50%   50%   50%   50%   50% 
#> 0.009 0.009 0.010 0.010 0.010 0.010
head(rch_gof@gof$lower)
#>  2.5%  2.5%  2.5%  2.5%  2.5%  2.5% 
#> 0.003 0.003 0.004 0.004 0.004 0.004
head(rch_gof@gof$upper)
#> 97.5% 97.5% 97.5% 97.5% 97.5% 97.5% 
#> 0.015 0.015 0.017 0.017 0.017 0.017
```

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, value p should not be within the range of the data and not too extreme given the nature of empirical probabilities.

```r
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.