Vignette for the tailDepFun package

Anna Kiriliouk
Université de Namur
Faculté des sciences économiques, sociales et de gestion
Rempart de la vierge 8, B-5000 Namur, Belgium
E-mail: anna.kiriliouk@unamur.be

1 Introduction

This package provides functions for the estimation of tail dependence parameters for a variety of models. The estimators that are implemented are

- The pairwise M-estimator described in Einmahl et al. (2016), which will be referred to as Mestimator in the R code.
- The weighted least squares estimator described in Einmahl et al. (2018), which will be called WLS in the R code.

This package contains three main functions:

- EstimationGumbel: estimation of the parameter of a Gumbel model, also called a logistic model.
- EstimationMaxLinear: estimation of the parameters of a max-linear model, possibly defined on a directed acyclic graph (DAG).
- EstimationBR: estimation of the parameters of an (anisotropic) Brown–Resnick process.

All the above models are defined by means of their stable tail dependence function, denoted by ℓ. For this reason, functions that compute the (bias-corrected) empirical stable tail dependence function are available, as well as the function SelectGrid that aids to define a regular grid of indices in which the stable tail dependence function should be evaluated.

2 Choosing a grid

In the definition of the weighted least squares estimator (Einmahl et al., 2018, Section 2.2), we evaluate ℓ in the points $c_1, \ldots, c_q \in [0, \infty)^d$, with $c_m = (c_{m1}, \ldots, c_{md})$ for $m = 1, \ldots, q$ for $q \geq p$, where p denotes the dimension of the parameter vector of ℓ. The function selectGrid aids in selecting these points. For instance, the grids used in the simulation study in Section 3.1 of Einmahl et al. (2018) for dimension $d = 5$ and method WLS can be generated as follows

```
> selectGrid(cst = c(0,1), d = 5)
> selectGrid(cst = c(0,1), d = 5, nonzero = 3)
```

for pairs and triples respectively. For the simulation study on the Brown–Resnick process in Section 3.2 of Einmahl et al. (2018), we first need to define the locations of the stations on a 3×4 unit distance grid

```
> loc <- cbind(rep(1:3, 4), rep(1:4, each = 3))
> selectGrid(cst = c(0,1), d = 12, locations = loc, maxDistance = sqrt(2))
```

Finally, the grid used in the simulation study in Section 3.3 of Einmahl et al. (2018) for method WLS can be generated as follows

```
> selectGrid(cst = c(0,0.5,1), d = 4, nonzero = c(2:4))
```

Note that we always set `cst = c(0,1)` when we want to use an estimator based on extremal coefficients only. This function can also be used to select the pairs for the pairwise M-estimator. Then, setting `cst = c(0,1)` and `nonzero = 2`, we get q rows, where the two ones in each row correspond to the pair that is selected.

3 Gumbel model

The Gumbel model, also known as the logistic model, has stable tail dependence function

$$\ell(x_1, \ldots, x_d; \theta) = \left(x_1^{1/\theta} + \cdots + x_d^{1/\theta} \right)^{\theta}, \qquad \theta \in (0, 1].$$

We can generate observations from it using the copula package (Hofert et al., 2015), here in three dimensions with parameter value $\theta = 0.5$, and transform them to unit Pareto margins using ranks.

```
> library(copula)
> set.seed(1)
> n <- 1000
> data <- rCopula(n = n,copula = gumbelCopula(param = 1/0.5, dim = 3))
> x <- apply(data, 2, function(col) n/(n + 0.5 - rank(col)))
```

Then, we can estimate θ using either the pairwise M-estimator

```
> indices <- selectGrid(c(0,1), d = 3)
> EstimationGumbel(x, indices, k = 50, method = "Mestimator")$theta
[1] 0.4123743
```

or using the weighted least squares estimator, based on the points c_m on the grid $\{0, 0.5, 1\}^3$ having at least two positive coordinates

```
> indices <- selectGrid(c(0,0.5,1), d = 3, nonzero = c(2,3))
> EstimationGumbel(x, indices, k = 50, method = "WLS")$theta
[1] 0.4177336
```

In Einmahl et al. (2018, Section 3.1), we simulate from a process in the max-domain of attraction of the Gumbel model, also known as the outer power Clayton copula (Hofert

et al., 2015). The outer power Clayton copula is the Archimedean copula with generator $\psi(t) = \psi_\beta(t^\theta)$, where $\theta \in (0, 1]$ and

$$\psi_\beta(t) = \frac{1}{(1+\beta t)^{1/\beta}}, \qquad \beta > 0.$$

We can fix $\beta = 1$ and hence focus on the generator $\psi(t) = (1 + t^\theta)^{-1}$. A sample from this copula for $\theta = 0.7$ and $d = 3$ can be obtained using the copula package as follows.

```
> opc <- opower(copClayton, thetabase = 1)
> copClayton <- onacopulaL(opc, list(theta = 1/0.7, comp = c(1:3)))
> data <- rnacopula(n = n, copula = copClayton)
```

4 Brown–Resnick process

A full definition of the process and its stable tail dependence function can be found in Einmahl et al. (2016, Section 4.1). Let $\theta = (\alpha, \rho, \beta, c)$ denote the parameters of the anisotropic Brown–Resnick process, where $\alpha \in (0, 2]$, $\rho > 0$, $\beta \in [0, \pi/2)$ and $c > 0$.

We first illustrate the estimation of an isotropic Brown–Resnick process using the pairwise M-estimator. We define coordinates of four locations and we select all pairs of locations.

```
> locations <- rbind(c(1,1),c(2,1),c(1,2),c(2,2))
> indices <- selectGrid(cst = c(0,1), d = 4, locations = locations)
```

Then we generate data from the Brown–Resnick process using the SpatialExtremes package (Ribatet, 2015).

```
> set.seed(2)
> library(SpatialExtremes)
> x <- rmaxstab(n = 5000, coord = locations, cov.mod = "brown",
+               range = 3, smooth = 1)
```

We calculate the estimator for $k = 300$. This could take a couple of minutes.

```
> EstimationBR(x, locations, indices, k = 300, method = "Mestimator",
+              iterate = TRUE, isotropic = TRUE, Tol = 1e-04)
$theta
[1] 1.235120 2.689146

$theta_pilot
[1] 1.235124 2.689140

$covMatrix
             [,1]        [,2]
[1,]  0.009990198 -0.02362328
[2,] -0.023623279  0.11335579

$value
[1] 0.01474498
```

Standard errors are simply the square roots of the diagonal elements of `covMatrix`. Setting `iterate = TRUE` means that we use the two-step optimal weighting procedure as described in (Einmahl et al., 2016, Corollary 3.3): `theta` returns the parameter estimates obtained using the optimal weight matrix, which is defined as the inverse of the matrix Σ evaluated in `theta_pilot`.

Next we estimate the parameters using the weighted least squares estimator. We use the bias-corrected stable tail dependence function (Beirlant et al., 2016). For `method = "WLS"` the option `iterate = TRUE` means that we use the continuous updating procedure as described in Einmahl et al. (2018, Corollary 2.3). .

```
> EstimationBR(x, locations, indices, 300, method = "WLS", isotropic = TRUE,
+              biascorr = TRUE, iterate = TRUE)
$theta
[1] 1.113500 3.087141

$theta_pilot
[1] 1.115788 3.092344

$covMatrix
             [,1]        [,2]
[1,]  0.02644506 -0.07761775
[2,] -0.07761775  0.31759008

$value
[1] 0.006486834
```

Note that since we set `iterate = TRUE`, we can use Einmahl et al. (2018, Corollary 2.5) to test the goodness-of-fit of the model. The test statistic is $300 \times 0.006486834 = 1.94605$, which we should compare to a high quantile of the χ^2_4 distribution.

If we want to mimic the two-step weighting procedure from Einmahl et al. (2016) instead of continuous updated weighting, we could do so as follows

```
> result <- EstimationBR(x, locations, indices, 300, method = "WLS",
+                        isotropic = TRUE, biascorr = TRUE, covMat = FALSE)
> Sigma <- AsymVarBR(locations, indices, par = result$theta_pilot,
+                      method = "WLS")
> EstimationBR(x, locations, indices, 300, method= "WLS", isotropic = TRUE,
+              biascorr = TRUE, Omega = solve(Sigma))$theta
[1] 1.115812 3.092290
```

If we want to estimate an isotropic Brown–Resnick process, we need to transform the coordinates of our locations, since we can only simulate isotropic Brown–Resnick processes. Hence, we multiply the coordinates of our locations with $V^{-1}(\beta, c)$; see Einmahl et al. (2016, Section 4.1). Here we take $\beta = 0.25$ and $c = 1$.

```
> Vmat <- matrix(c(cos(0.25),1.5*sin(0.25),-sin(0.25),1.5*cos(0.25)),nrow=2)
> locationsAniso <- locations %*% t(solve(Vmat))
> EstimationBR(x, locationsAniso, indices, 300, method = "WLS",
+              biascorr = TRUE, iterate = TRUE)$theta
[1] 1.1275481 3.1058154 0.4099169 1.5394794
```

Finally, some tips for the use of this function:

- If the number of locations d is small ($d < 8$ say), a sufficiently large sample size (eg $n > 2000$) is needed to obtain a satisfying result. However, if d is large, a sample size of $n = 500$ should suffice.
- The tolerance parameter is used when calculating the three- and four-dimensional integrals in the asymptotic covariance matrix; see the supplementary material in Einmahl et al. (2016). A tolerance of 10^{-4} often suffices, although the default tolerance is a safer choice.
- For an anisotropic process, it is advised to try a couple of starting values if d is small, preferably a starting value with $c < 1$ and one with $c > 1$.
- Setting `iterate = TRUE` has a more significant effect when d is large.
- If the number of pairs q is large, then `method = "Mestimator"` will be rather slow. This is due to the calculation of the weight matrix Ω and the covariance matrix. Setting `iterate = FALSE` and `covMat = FALSE` will make estimation fast even for several hundreds of pairs of locations.
- `method = "WLS"`, it is not advised to change the values of `k1` or `tau`; the default values are chosen as advised in Beirlant et al. (2016).

Note that an extension two triples and more general grids (i.e., where `cst` is not necessarily `c(0,1)`) might be available in the future.

5 Max-linear model

The max-linear model is described in detail in Einmahl et al. (2018, Section 3.3). Its parameter matrix is a $d \times r$ matrix $B := (b_{jt})_{j,t}$, where r denotes the number of factors and d the dimension. The factor loadings b_{jt} are non-negative constants such that $\sum_{t=1}^{r} b_{jt} = 1$ for every $j \in \{1, \ldots, d\}$ and all column sums of B are positive. Note that B has $p = d \times (r-1)$ free elements. The parameter vector $\theta \in \mathbb{R}^p$ is defined by stacking the columns of B in decreasing order of their sums, leaving out the column with the lowest sum.

To illustrate estimation of a 2-factor model in dimension $d = 3$, we simulate data with parameter vector $\theta = c(b_{11}, b_{12}, b_{13}) = c(0.3, 0.5, 0.9)$.

```
> set.seed(1)
> n <- 1000
> fr <- matrix(-1/log(runif(2*n)), nrow = n, ncol = 2)
> data <- cbind(pmax(0.3*fr[,1], 0.7*fr[,2]),pmax(0.5*fr[,1], 0.5*fr[,2]),
+               pmax(0.9*fr[,1], 0.1*fr[,2]))
```

Then we transform to unit Pareto, select a grid and estimate the parameters using the weighted least squares estimator. Note that the choice `cst = c(0,1)` will usually not lead to a valid estimator; see (Einmahl et al., 2018, Section 3.3).

```
> x <- apply(data, 2, function(i) n/(n + 0.5 - rank(i)))
> indices <- selectGrid(cst = c(0,0.5,1), d = 3)
```

```
> EstimationMaxLinear(x, indices, k = 100, method = "WLS",
+                     iterate = TRUE, GoFtest = TRUE,
+                     startingValue = c(0.3,0.5,0.9))
$theta
[1] 0.3284527 0.5007583 0.9115711

$theta_pilot
[1] 0.3152544 0.4893125 0.8980283

$covMatrix
             [,1]         [,2]         [,3]
[1,] 0.0022120835 0.0021556005 0.0006626992
[2,] 0.0021556005 0.0026694102 0.0009185456
[3,] 0.0006626992 0.0009185456 0.0011039448

$value
[1] 0.2069704

$GoFresult
$GoFresult$value
[1] 1.961863

$GoFresult$s
[1] 2
```

The results of `GoFtest` permit us to do the test described in Einmahl et al. (2018, Corollary 2.6). We need to compare `GoFresult$value` to the 95% quantile of a χ^2 distribution with $s = 2$ degrees of freedom, given by 5.99. For the two-step weighting procedure from Einmahl et al. (2016) we would do

```
> result <- EstimationMaxLinear(x, indices, k = 100, method = "WLS",
+                               startingValue = c(0.3,0.5,0.9))
> Sigma <- AsymVarML(indices, par = result$theta_pilot)
> while(rcond(Sigma) < 1e-05){Sigma <- Sigma + (1e-05)*diag(nrow(indices))}
> EstimationMaxLinear(x, indices, 100, method = "WLS", Omega = solve(Sigma),
+                     startingValue = result$theta_pilot)$theta
[1] 0.3277286 0.4988443 0.9105570
```

The correction on `Sigma` is done because it is not invertible otherwise; see Einmahl et al. (2018, Remark 3.1).

In the above estimation, the function `EstimationMaxLinear` assumed a 2-factor model because we did not provide `Bmatrix` and `Ldot`. If we want to fit a max-linear model based on a directed acyclic graph, for instance the one in Gissibl and Klüppelberg (2015, Example 2.1) or Einmahl et al. (2018, Section 3.3), we need to define a `Bmatrix`, corresponding to the matrix of coefficients B, and a `Ldot`, corresponding to the total derivative of $L(\theta) = (\ell(c_m; \theta))_{m=1,\dots,q}$. For instance, B is given by

```
> Bmatrix <- function(theta){
```

```
+  temp <- max(theta[1]*theta[3],theta[2]*theta[4])
+  B<- cbind(c(1, theta[1], theta[2], temp),
+            c(0, 1 - theta[1], 0, (1 - theta[1])*theta[3]),
+            c(0, 0, 1 - theta[2], (1 - theta[2])*theta[4]),
+            c(0, 0, 0, 1 - temp - (1 - theta[1])*theta[3] -
+                                  (1 - theta[2])*theta[4]))
+  return(B)
+ }
```

and then we generate data from the DAG as follows

```
> d <- r <- 4
> n <- 1000
> theta <- c(0.3, 0.8, 0.4, 0.55)
> B <- Bmatrix(theta)
> set.seed(1)
> fr <- matrix(-1/log(runif(r*n)), nrow = n, ncol = r)
> data <- cbind(B[1,1]*fr[,1], pmax(B[2,1]*fr[,1], B[2,2]*fr[,2]),
+          pmax(B[3,1]*fr[,1], B[3,3]*fr[,3]),
+          pmax(B[4,1]*fr[,1], B[4,2]*fr[,2], B[4,3]*fr[,3], B[4,4]*fr[,4]))
> x <- apply(data, 2, function(i) n/(n + 0.5 - rank(i)))
```

We then estimate using a grid as in Einmahl et al. (2016, Section 3.3)

```
> indices <- selectGrid(cst = c(0,0.5,1), d = 4, nonzero = c(2:4))
> EstimationMaxLinear(x, indices, k = 100, method = "WLS", Bmatrix = Bmatrix,
+              startingValue = c(0.3,0.8,0.4,0.55), covMat = FALSE)$theta
[1] 0.3502117 0.8692499 0.4124804 0.5909954
```

Note that in order to calculate `covMat`, we would also need to provide `Ldot`.

References

Beirlant, J., Escobar-Bach, M., Goegebeur, Y., and Guillou, A. (2016). Bias-corrected estimation of stable tail dependence function. *Journal of Multivariate Analysis*, 143:453–466.

Einmahl, J. H., Kiriliouk, A., Krajina, A., and Segers, J. (2016). An M-estimator of spatial tail dependence. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(1):275–298.

Einmahl, J. H., Kiriliouk, A., and Segers, J. (2018). A continuous updating weighted least squares estimator of tail dependence in high dimensions. *Extremes*, 21(2):205–233.

Gissibl, N. and Klüppelberg, C. (2015). Max-linear models on directed acyclic graphs. Available at `http://arxiv.org/abs/1512.07522`.

Hofert, M., Kojadinovic, I., Maechler, M., and Yan, J. (2015). *copula: Multivariate Dependence with Copulas*. R package version 0.999-14.

Ribatet, M. (2015). *SpatialExtremes: Modelling Spatial Extremes*. R package version 2.0-2.