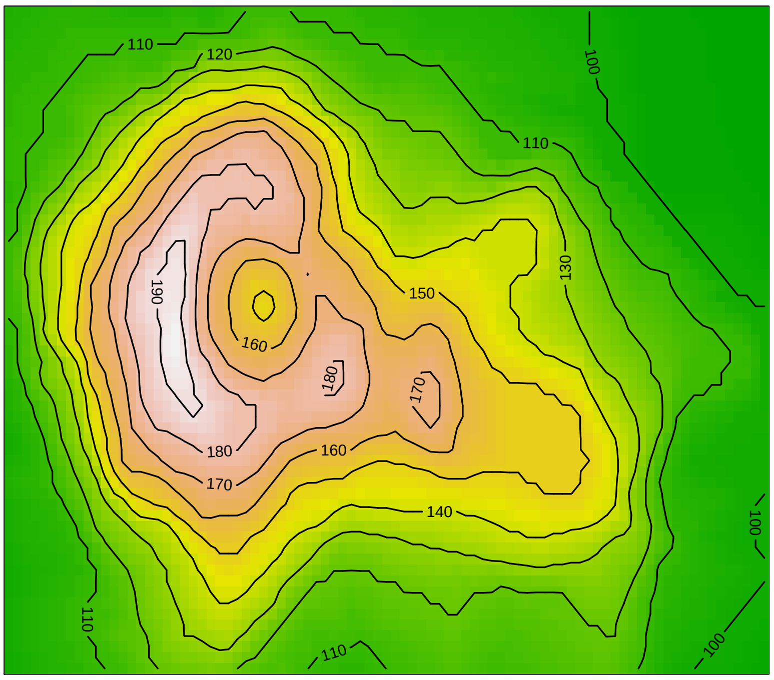
A User's Guide to the SpatialExtremes Package

Mathieu Ribatet[†] & Simone Padoan[*]

[†]Chair of Statistics
École Polytechnique Fédérale de Lausanne
Switzerland
[*]Laboratory of Environmental Fluid Mechanics and Hydrology
École Polytechnique Fédérale de Lausanne
Switzerland



Contents

List of Figures

Introduction

What is the SpatialExtremes package?

The **SpatialExtremes** package is an add-on package for the R [?] statistical computing system. It provides functions for the analysis of spatial extremes.

All comments, criticisms and queries on the package or associated documentation are gratefully received.

Obtaining the package/guide

The package can be downloaded from CRAN (The Comprehensive R Archive Network) at `http://cran.r-project.org/`. This guide (in pdf) will be in the directory `SpatialExtremes/doc/` underneath wherever the package is installed. You can get it by invoking

```
> vignette("SpatialExtremesGuide")
```

Contents

This guide contains a few elements of theory on the modelling of spatial extremes as well as examples on the use of the **SpatialExtremes** package. Section 1 gives an (light) introduction to max-stable processes and defines two different characterisations of such processes. Section 2 presents the methodology used in the package to fit max-stable processes to data while section 3 describes useful functions for prediction and visualizing fitted models. Details for the computation of the pairwise density and gradient are reported into the Annex A.

Caveat

I have checked these functions as best I can but, as ever, they may contain bugs. If you find a bug or suspected bug in the code or the documentation please report it to me at mathieu.ribatet@epfl.ch. Please include an appropriate subject line.

Legalese

Acknowledgements

This work has been supported by the Competence Center Environment and Sustainability within the EXTREMES project.

Chapter 1

An Introduction to Max-Stable Processes

A max-stable process $Z(\cdot)$ is the limit process of maxima of independent identically distributed random fields $Y_i(x)$, $x \in \mathbb{R}^d$. Namely, for suitable $a_n(x) > 0$ and $b_n(x) \in \mathbb{R}$,

$$Z(x) = \lim_{n \to +\infty} \frac{\max_{i=1}^n Y_i(x) - b_n(x)}{a_n(x)}, \qquad x \in \mathbb{R}^d \tag{1.1}$$

Note that (1.1) does not ensure that the limit exists. However, provided it does and from (1.1), we can see that max-stable processes might be appropriate models for modelling annual maxima of spatial data.

Currently, there are two different characterisations of a max-stable process. The first one, often referred to the *rainfall-storm* model, as first been introduced by **?**. More recently, **?** introduced a new characterisation of a max-stable process allowing for a random shape.

It is out of the scope of this document to describe fully the main differences between the two canonical constructions. We will restrict our attention to particular cases of these characterisations.

Unfortunately, closed forms for the density of these two models are only known for two different points in $\mathbb{R}^d$. Consequently, fitting max-stable processes to data is not straightforward and the SpatialExtremes package provides convenient tools for it.

1.1 The Smith's Model

The Smith's model[1] is given by:

$$\Pr[Z_1 \leq z_1, Z_2 \leq z_2] = \exp\left[-\frac{1}{z_1}\Phi\left(\frac{a}{2} + \frac{1}{a}\log\frac{z_2}{z_1}\right) - \frac{1}{z_2}\Phi\left(\frac{a}{2} + \frac{1}{a}\log\frac{z_1}{z_2}\right)\right] \tag{1.2}$$

where Φ is the standard normal cumulative distribution function and, for two given locations #1 and #2

$$a^2 = \Delta x^T \Sigma^{-1} \Delta x \quad \text{and} \quad \Sigma = \begin{bmatrix} cov_{11} & cov_{12} \\ cov_{12} & cov_{22} \end{bmatrix} \quad \text{or} \quad \Sigma = \begin{bmatrix} cov_{11} & cov_{12} & cov_{13} \\ cov_{12} & cov_{22} & cov_{23} \\ cov_{13} & cov_{23} & cov_{33} \end{bmatrix} \quad \text{and so forth}$$

where Δx is the distance vector between location #1 and location #2.

The derivation of the density is reported in section A.1.2. Currenlty, the package only handle 2 by 2 or 3 by 3 covariance matrix Σ.

[1]There's another form of the Smith's model that uses a Student distribution instead of the Normal one. However, it is not currently implemented.

1.2 The Schlather's Model

The Schlather's model is given by:

$$\Pr[Z_1 \leq z_1, Z_2 \leq z_2] = \exp\left[-\frac{1}{2}\left(\frac{1}{z_1}+\frac{1}{z_2}\right)\left(1+\sqrt{1-2(\rho(h)+1)\frac{z_1 z_2}{(z_1+z_2)^2}}\right)\right] \tag{1.3}$$

where h is the distance between location #1 and location #2 and $\rho(h)$ is a valid correlation function such as $-1 \leq \rho(h) \leq 1$.

Currently, there is three types of covariance functions implemented:

Whittle-Matérn	$\rho(h) = sill \frac{2^{1-smooth}}{\Gamma(smooth)} \left(\frac{h}{range}\right)^{smooth} K_{smooth}\left(\frac{h}{range}\right)$
Cauchy	$\rho(h) = sill \left[1+\left(\frac{h}{range}\right)^2\right]^{-smooth}$
Powered Exponential	$\rho(h) = sill \exp\left[-\left(\frac{h}{range}\right)^{smooth}\right]$

where h is the distance between locations #1 and #2, *sill*, *range* and *smooth* are the sill, the range and the smooth parameters of the covariance function, Γ is the gamma function and K_{smooth} is the modified Bessel function of the third kind with order *smooth*.

The derivation of the density is reported in section A.2.1.

Chapter 2

Fitting a Max-Stable Process to Data

As stated in the previous chapter, the densities of the two max-stable characterisations are only known for two different locations. The strategy used in the package is to use pairwise-likelihood instead of the "full" likelihood. Namely, the log pairwise-likelihood is given by:

$$\ell_p(\mathbf{y};\psi) = \sum_{i<j} \sum_{k=1}^{n_{i,j}} \log f(y_k^{(i)}, y_k^{(j)}) \tag{2.1}$$

where $\mathbf{y}$ is the data available on the whole region, $n_{i,j}$ is the number of common observations between sites i and j, $y_k^{(i)}$ is the k-th observation of the i-th site and $f(\cdot,\cdot)$ is the bivariate distribution of the max-stable process - see Annex A for the analytical forms.

Consequently, the max-stable process is fitted to data by maximizing the log pairwise-likelihood[1]. Properties of the maximum composite likelihood estimator[2] are well known [**??**]. In particular, because each pairwise score equation is unbiased, the sum of these score equations is unbiased too and leads to consistent estimations.

If one is interested only in fitting the covariance matrix Σ or the covariance function ρ to data, another fitting procedure is available. The strategy is estimate the extremal coefficient and to fit either the Smith's or the Schlather's models using a least square criterion.

2.1 Assuming Unit Fréchet Margins

To start working with the SpatialExtremes package, let consider a simple case study for which each location is unit Fréchet distributed. M. Schlather developed a R package called *RandomFields* to simulate spatial random fields from a max-stable process. Consequently, all we need is to define the coordinate of each location as well as the parameters of the covariance function in equation (A.29).

```
> library(RandomFields)
> n.site <- 40
> locations <- matrix(runif(2 * n.site, 0, 10), ncol = 2)
> colnames(locations) <- c("lon", "lat")
> ms0 <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = "wh", param = c(0, 1, 0.2, 3, 1.2), maxstable = "extr",
+     n = 80)
> ms0 <- t(ms0)
```

[1]This is why the fitting procedure may be time consuming with large region.

[2]In our case, the maximum pairwise likelihood estimator

For this application, the covariance function is taken to be the Whittle-Matérn covariance function with sill, range and smooth parameters equal to 0.8 (1 - 0.2), 3 and 1.2 respectively. The locations are distributed uniformly on the square $[0, 10]^2$.

To fit a max-stable process using pairwise likelihood and assuming unit Fréchet margins, all we have to do is to invoke:

```
> fitmaxstab(ms0, locations, cov.mod = "whitmat", fit.marge = FALSE)

        Estimator: MPLE
            Model: Schlather
   Pair. Deviance: 578749.5
              TIC: NA
Covariance Family: Whittle-Matern

Estimates
  Marginal Parameters:
  Assuming unit Frechet.
  Dependence Parameters:
   sill    range   smooth
 0.7646   0.2484  49.8503

Optimization Information
  Convergence: successful
  Function Evaluations: 588
```

From this output, we can see that we indeed use the Schlather's representation with a Whittle-Matérn covariance function. The pairwise deviance is given and the Takeuchi's information criterion (TIC) is not available (NA) - see section 2.4 for more details. The convergence was successful and the estimates of the covariance function are accessible. Note that large deviations from the theoretical values are not fatal as the parameters of the Whittle-Matérn covariance function are far from orthogonal. Thus, the range and smooth estimates may be totally different while leading (approximately) to the same covariance function.

When using the Whittle-Matérn covariance function, it is sometimes preferable to fix the smooth parameter using prior knowledge on the process smoothness [?]. This can be done by:

```
> fitmaxstab(ms0, locations, cov.mod = "whitmat", smooth = 1.2,
+     fit.marge = FALSE)

        Estimator: MPLE
            Model: Schlather
   Pair. Deviance: 578853.6
              TIC: 579234.9
Covariance Family: Whittle-Matern

Estimates
  Marginal Parameters:
  Assuming unit Frechet.
  Dependence Parameters:
 sill  range
0.845  1.662

Standard Error Type: score

Standard Errors
   sill    range
0.02385  0.43865
```

```
Asymptotic Variance Covariance
       sill        range
sill    0.0005688  -0.0083251
range  -0.0083251   0.1924115

Optimization Information
  Convergence: successful
  Function Evaluations: 69
```

Now we can see that only the range parameter was estimated and standard errors as well as the TIC are available.

Despite the Whittle-Matérn is a flexible covariance function, one may want to consider other types of covariance functions. This is achieved by invoking:

```
> fitmaxstab(ms0, locations, cov.mod = "cauchy", fit.marge = FALSE)
> fitmaxstab(ms0, locations, cov.mod = "powexp", fit.marge = FALSE)
```

One may also consider the Smith's characterisation instead of the Schlather's one:

```
> sigma = matrix(c(100, 25, 25, 220), ncol = 2)
> sigma.inv = solve(sigma)
> sqrtCinv = t(chol(sigma.inv))
> model = list(list(model = "gauss", var = 1, aniso = sqrtCinv/2))
> msSmith <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = model, maxstable = "Bool", n = 50)
> msSmith <- t(msSmith)
> fitmaxstab(msSmith, locations, cov.mod = "gauss", fit.marge = FALSE)
```

```
        Estimator: MPLE
            Model: Smith
   Pair. Deviance: 217784.5
              TIC: 219046.9
Covariance Family: Gaussian

Estimates
  Marginal Parameters:
  Not estimated.
  Dependence Parameters:
 cov11   cov12   cov22
121.92   24.93  215.82

Standard Error Type: score

Standard Errors
cov11  cov12  cov22
21.69  25.70  40.67

Asymptotic Variance Covariance
       cov11    cov12    cov22
cov11   470.24   -22.49   110.82
cov12   -22.49   660.36   304.24
cov22   110.82   304.24  1653.76

Optimization Information
  Convergence: successful
  Function Evaluations: 74
```

Obvisously, for each covariance model, one can fix any parameter; so that all the two following codes are valid:

```
> fitmaxstab(ms0, locations, cov.mod = "gauss", cov12 = 0, fit.marge = FALSE)
> fitmaxstab(ms0, locations, cov.mod = "cauchy", range = 3, fit.marge = FALSE)
```

Note that passing `fit.marge = TRUE` in all the previous codes will result in fitting the GEV parameters for each location. However, be careful as it will be really CPU demanding as there will be `3 n.site + p` parameters to estimate - where `p` is the number of parameters for the covariance part.

It is also possible to used different optimization routines to fit the model to data. This is achieved by passing the `method` argument. For instance, if one want to use the `BFGS` method:

```
> fitmaxstab(ms0, locations, cov.mod = "gauss", cov12 = 0, fit.marge = FALSE,
+     method = "BFGS")
```

Instead of using the `optim` function, one may want to use the `nlm` or `nlminb` functions. This is done as before using the `method = "nlm"` or `method = "nlminb"` option.

If now, we want to fit the model using the least square criterion:

```
> fitcovmat(ms0, locations)

        Estimator: Least Square
            Model: Smith
  Objective Value: 0.01752087
Covariance Family: Gaussian

Estimates
  Marginal Parameters:
  Not estimated.
  Dependence Parameters:
  cov11    cov12    cov22
 6.2971  -0.7996   6.2065

Optimization Information
  Convergence: successful
  Function Evaluations: 100

> fitcovariance(ms0, locations, "whitmat")

        Estimator: Least Square
            Model: Schlather
  Objective Value: 0.004237805
Covariance Family: Whittle-Matern

Estimates
  Marginal Parameters:
  Assuming unit Frechet.
  Dependence Parameters:
   sill    range   smooth
 0.7242   0.2302  49.9986

Optimization Information
  Convergence: successful
  Function Evaluations: 686
```

2.2 With Unknown GEV Margins

In practice, the observations will never be drawn from a unit Fréchet distribution so that the previous section won't help much with concrete applications. One way to avoid this problem is to fit a GEV to each location and then transform all data to unit Fréchet. This is done using the `gev2frech` function and the following code:

```
> x <- c(2.2975896, 1.6448808, 1.3323833, -0.4464904, 2.2737603,
+     -0.2581876, 9.5184398, -0.5899699, 0.4974283, -0.8152157)
> gev2frech(x, 1, 2, 0.2)

 [1]  1.8404710  1.3667970  1.1776129  0.4578484  1.8211427  0.5105137
 [7] 21.7781994  0.4207148  0.7727342  0.3673129
```

The drawback of this approach is that standard errors are definitively lost as the margins are first fitted and then the covariance structure. Consequently, the standard errors related to the covariance function are underestimated as we suppose that data are originally unit Fréchet.

Fortunately, the SpatialExtremes solves this problem by fitting in *one step* both GEV and covariance parameters. This could be done in two ways. First, one can pass the option `fit.marge = TRUE`. However, as said in the previous section this will be really time consuming. Another drawback is that prediction at ungauged location won't be possible.

Another way may be to fit a *response surface* for the GEV parameters. Currently, the response surfaces allowed by the package are polynomial surfaces or a penalized smoothing spline[3].

The SpatialExtremes package defines response surfaces using the *R formula* approach e.g.

```
> y ~ lat + I(lon^2)
```

for a polynomial surface and

```
> n.knots <- 5
> knots <- quantile(locations[, 2], prob = 1:n.knots/(n.knots +
+     1))
> y ~ rb(lat, knots = knots, degree = 3, penalty = 0.5)
```

penalty coefficient (also known as the smoothing parameter) equals to 0.5.

Let start with a simple polynomial surface. For this purpose, we need to simulate a max-stable process and then transform the observations to the desired GEV scale. This could be done by the following lines:

```
> n.site <- 20
> locations <- matrix(runif(2 * n.site, 0, 10), ncol = 2)
> colnames(locations) <- c("lon", "lat")
> sigma = matrix(c(100, 25, 25, 220), ncol = 2)
> sigma.inv = solve(sigma)
> sqrtCinv = t(chol(sigma.inv))
> model = list(list(model = "gauss", var = 1, aniso = sqrtCinv/2))
> ms0 <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = model, maxstable = "Bool", n = 50)
> ms1 <- t(ms0)
> param.loc <- -10 + 2 * locations[, 2]
> param.scale <- 5 + 2 * locations[, 1] + locations[, 2]^2
> param.shape <- rep(0.2, n.site)
> for (i in 1:n.site) ms1[, i] <- param.scale[i] * (ms1[, i]^param.shape[i] -
+     1)/param.shape[i] + param.loc[i]
```

[3]this is an exclusive "or" (or *xor*) i.e. a response surface with a polynomial and a spline is not possible

Once the data are appropriately generated, we need to define the response surface for the fitted max-stable model. This is done invoking:

```
> loc.form <- y ~ lat
> scale.form <- y ~ lon + I(lat^2)
> shape.form <- y ~ 1
```

Lastly, one can easily fit the model to data using:

```
> fitmaxstab(ms1, locations, "gauss", loc.form = loc.form, scale.form = scale.form,
+     shape.form = shape.form)

Computing appropriate starting values
Starting values are defined
Starting values are:
      cov11       cov12       cov22   locCoeff1   locCoeff2 scaleCoeff1
 59.4871222   2.6420244 249.4890970 -13.3512737   1.5369631   4.8090426
scaleCoeff2 scaleCoeff3  shapeCoeff
  1.7217574   1.0909128   0.2496088
        Estimator: MPLE
            Model: Smith
   Pair. Deviance: 187937.5
              TIC: 198333.8
Covariance Family: Gaussian

Estimates
  Marginal Parameters:
    Location Parameters:
locCoeff1  locCoeff2
  -11.358      1.201
      Scale Parameters:
scaleCoeff1  scaleCoeff2  scaleCoeff3
      4.552        2.161        1.098
      Shape Parameters:
shapeCoeff
    0.3242
  Dependence Parameters:
   cov11     cov12     cov22
 99.9525   -0.8411  287.9212

Standard Error Type: score

Standard Errors
      cov11        cov12        cov22    locCoeff1    locCoeff2  scaleCoeff1
    98.3208      30.6267     379.3208       0.8793       0.5428       5.9724
scaleCoeff2  scaleCoeff3   shapeCoeff
     0.9533       0.4536       0.3431

Asymptotic Variance Covariance
             cov11      cov12      cov22      locCoeff1  locCoeff2  scaleCoeff1
cov11        9.667e+03  1.640e+03  3.612e+04  8.159e+01  5.036e+01  5.051e+02
cov12        1.640e+03  9.380e+02  6.833e+03  1.493e+01  9.994e+00  1.003e+02
cov22        3.612e+04  6.833e+03  1.439e+05  3.185e+02  1.963e+02  1.932e+03
locCoeff1    8.159e+01  1.493e+01  3.185e+02  7.732e-01  4.304e-01  4.393e+00
locCoeff2    5.036e+01  9.994e+00  1.963e+02  4.304e-01  2.946e-01  2.876e+00
scaleCoeff1  5.051e+02  1.003e+02  1.932e+03  4.393e+00  2.876e+00  3.567e+01
```

```
scaleCoeff2  5.477e+01  9.124e+00  2.189e+02  4.744e-01  2.611e-01  8.290e-01
scaleCoeff3  4.372e+01  8.103e+00  1.701e+02  3.796e-01  2.347e-01  2.263e+00
shapeCoeff   3.154e+01  6.276e+00  1.267e+02  2.805e-01  1.695e-01  1.605e+00
             scaleCoeff2  scaleCoeff3  shapeCoeff
cov11        5.477e+01    4.372e+01    3.154e+01
cov12        9.124e+00    8.103e+00    6.276e+00
cov22        2.189e+02    1.701e+02    1.267e+02
locCoeff1    4.744e-01    3.796e-01    2.805e-01
locCoeff2    2.611e-01    2.347e-01    1.695e-01
scaleCoeff1  8.290e-01    2.263e+00    1.605e+00
scaleCoeff2  9.089e-01    2.857e-01    2.155e-01
scaleCoeff3  2.857e-01    2.057e-01    1.511e-01
shapeCoeff   2.155e-01    1.511e-01    1.177e-01

Optimization Information
  Convergence: successful
  Function Evaluations: 2622
```

If we want to fit a spline for the location GEV parameter while preserving the polynomials for the scale and shape parameters, this will require a little more steps as the knots and the penalty coefficient must be defined[4].

```
> n.knots <- 5
> knots <- quantile(locations[, 2], 1:n.knots/(n.knots + 1))
> loc.form <- y ~ rb(lat, knots = knots, degree = 3, penalty = 0.5)
> fitmaxstab(ms1, locations, "gauss", loc.form = loc.form, scale.form = scale.form,
+     shape.form = shape.form)
```

Obviously, all these steps still remain valid when fitting the Schlather's model. Note that you can fix any parameter as before for example, if one want to suppose that `scaleCoeff3 = 1`, this is done by invoking the following line:

```
> fitmaxstab(ms1, locations, "powexp", loc.form = loc.form, scale.form = scale.form,
+     shape.form = shape.form, scaleCoeff3 = 1)
```

Please note that when using 3 smoothing splines for the GEV parameters, you might have to tweak the `ndeps` option in the `optim` function if you use the `BFGS` optimization procedure:

```
> fitmaxstab(ms1, locations, "powexp", loc.form = loc.form, scale.form = scale.form,
+     shape.form = shape.form, control = list(ndeps = rep(10^-6,
+         n.par)), method = "BFGS")
```

where `n.par` is the number of parameters to be estimated.

2.3 Assessing Uncertainties

As stated in section 2, because the model is fitted by maximizing the pairwise likelihood instead of the "full" likelihood, the model is *misspecified.* Consequently, the maximum pairwise likelihood estimator is still asymptotically normally distributed but with a different asymptotic covariance matrix. Namely, the maximum pairwise likelihood estimator $\hat{\psi}_p$ satisfies the following relation:

$$\hat{\psi}_p \sim \mathcal{N}\left(\psi, H(\psi)^{-1}J(\psi)H(\psi)^{-1}\right), \qquad n \to +\infty \tag{2.2}$$

where $H(\psi) = \mathbb{E}[\nabla^2 \ell_p(\psi; \mathbf{Y})]$ (the Hessian matrix) and $J(\psi) = \mathrm{Var}[\nabla \ell_p(\psi; \mathbf{Y})]$, where the expectations are with respect to the "full" density.

[4]Currently, an automatic criteria for defining the "best" penalty coefficient does not exist.

In practice, to get the standard errors we need to get efficient estimates of $H(\psi)$ and $J(\psi)$. The estimation of the former is straightforward and is given by $\hat{H}(\hat{\psi}_p) = \nabla^2 \ell_p(\hat{\psi}_p; \mathbf{y})$; that is the Hessian matrix evaluated at $\hat{\psi}_p$.

The estimation of $J(\psi)$ can be done in two different ways. First, it can be estimated using the "naive" estimator $\hat{J}(\hat{\psi}_p) = \nabla \ell_p(\hat{\psi}_p; \mathbf{y}) \ell_p(\hat{\psi}_p; \mathbf{y})^T$. In the SpatialExtremes package, this estimator is tagged `grad` as it uses the gradient of the log pairwise likelihood. Another estimator is given by noticing that $J(\psi)$ corresponds to the variance of the pairwise score equations $\ell_p(\psi; \mathbf{Y}) = 0$. Consequently, a second estimator, tagged `score`, is given by the sample variance of each contribution to the pairwise score function. Note that the second estimator is only accessible if independent replications of $\mathbf{Y}$ are available[5].

The standard errors are available by invoking the following two lines:

```
> fitmaxstab(ms0, locations, cov.mod = "gauss", fit.marge = FALSE,
+     std.err.type = "score")

        Estimator: MPLE
            Model: Smith
   Pair. Deviance: 52509.48
              TIC: 52780.95
Covariance Family: Gaussian

Estimates
  Marginal Parameters:
  Not estimated.
  Dependence Parameters:
 cov11   cov12   cov22
137.95   69.81  269.16

Standard Error Type: score

Standard Errors
cov11  cov12  cov22
22.19  30.21  55.58

Asymptotic Variance Covariance
       cov11   cov12   cov22
cov11   492.5   308.5   278.7
cov12   308.5   912.8  1043.9
cov22   278.7  1043.9  3088.6

Optimization Information
  Convergence: successful
  Function Evaluations: 86

> fitmaxstab(ms0, locations, cov.mod = "gauss", fit.marge = FALSE,
+     std.err.type = "grad")
```

2.4 Model Selection

It is sometimes useful to fit several models to data and then compare the models together. To this aim, the well-known Akaike information criterion (AIC) is often used. Because we work under miss-specification, AIC is not appropriate anymore. Instead, we will use a generalization of the Takeuchi's

[5]which will mostly be the case for spatial extremes.

information criterion. **?** show that, under miss-specification, an appropriate selection statistic is given by:

$$\text{TIC} = -\ell_p(\psi_p) - \text{tr}\left\{JH^{-1}\right\} \tag{2.3}$$

In accordance with the AIC, the best model will corresponds to the one that minimizes equation (2.3).

In practice, one can have a look at the output of the `fitmaxstab` function or use the `TIC` function.

```
> n.site <- 40
> locations <- matrix(runif(2 * n.site, 0, 10), ncol = 2)
> colnames(locations) <- c("lon", "lat")
> ms0 <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = "stable", param = c(0, 1, 0, 3, 1.2), maxstable = "extr",
+     n = 80)
> ms0 <- t(ms0)
> model1 <- fitmaxstab(ms0, locations, cov.mod = "powexp", fit.marge = FALSE,
+     sill = 1)
> model2 <- fitmaxstab(ms0, locations, cov.mod = "cauchy", fit.marge = FALSE,
+     sill = 1)
> TIC(model1, model2)

  model1   model2
472642.4 472988.6
```

Fortunately, with this simulated application, we clearly see that our `model1` is the best model.

TIC is useful when comparing different models. However, it may lack of power if the two models are nested. When dealing with nested models, one may prefer using the so called likelihood ratio statistics.

Because we are working with a misspecified model, the usual asymptotic χ_p^2 distribution, where p is the number of parameter to be estimated, doesn't hold anymore. There's two way to solve this issue: (a) adjusting the χ^2 distribution or (b) adjusting the composite likelihood so that the usual χ_p^2 holds - see **?**.

```
> require(RandomFields)
> n.site <- 30
> locations <- matrix(rnorm(2 * n.site, sd = sqrt(0.2)), ncol = 2)
> colnames(locations) <- c("lon", "lat")
> sigma <- matrix(c(100, 25, 25, 220), ncol = 2)
> sigma.inv <- solve(sigma)
> sqrtCinv <- t(chol(sigma.inv))
> model <- list(list(model = "gauss", var = 1, aniso = sqrtCinv/2))
> ms0 <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = model, maxstable = "Bool", n = 50)
> ms0 <- t(ms0)
> M0 <- fitmaxstab(ms0, locations, "gauss", fit.marge = FALSE)
> M1 <- fitmaxstab(ms0, locations, "gauss", fit.marge = FALSE,
+     cov11 = 100)
> anova(M0, M1)

Eigenvalues:  219.48

Analysis of Variance Table
   MDf Deviance Df  Chisq Pr(> sum lambda Chisq)
M1   2    56298
M0   3    56280  1 18.369                  0.773
```

From this output, we can see that M1 seems to be a better model than M0.

Chapter 3

Manipulating and Visualising Fitted Models

3.1 Prediction of the GEV parameters

Once the model is fitted, one may want to get the estimates of the GEV parameters at any locations. This is achieved using the *predict* function:

```
> fitted <- fitmaxstab(ms1, locations, "gauss", loc.form = loc.form,
+     scale.form = scale.form, shape.form = shape.form)

Computing appropriate starting values
Starting values are defined
Starting values are:
      cov11       cov12       cov22   locCoeff1   locCoeff2 scaleCoeff1
124.1334275  22.9688931 116.3156682 -10.0251222   1.6125755  -4.3727102
scaleCoeff2 scaleCoeff3  shapeCoeff
  3.7924393   0.9382657   0.1257981

> predict(fitted)

        lon       lat          loc      scale    shape
1  3.652536 1.7129927  -7.43776121  14.116336 0.168334
2  3.678214 6.8344681   2.13535735  56.128913 0.168334
3  3.314883 9.6581939   7.41349685  99.852457 0.168334
4  2.149816 0.9784233  -8.81082649   8.484307 0.168334
5  8.377888 6.4525365   1.42144656  62.954811 0.168334
6  5.260848 7.0541740   2.54603407  62.989248 0.168334
7  5.465879 5.1134111  -1.08166153  40.870299 0.168334
8  2.719089 0.3102123 -10.05985379   9.075036 0.168334
9  1.788405 8.1709779   4.63357627  70.646624 0.168334
10 7.372002 3.1726860  -4.70928627  30.201209 0.168334
11 9.030623 8.0214510   4.35407896  86.339018 0.168334
12 9.495964 3.0971002  -4.85057219  35.029713 0.168334
13 3.221411 0.7536396  -9.23099463  10.776443 0.168334
14 9.119104 8.9523454   6.09411704 101.700479 0.168334
15 5.250240 5.7351913   0.08057688  46.797908 0.168334
16 6.500299 7.3751564   3.14601784  70.510231 0.168334
17 5.983943 6.0288907   0.62956313  51.933655 0.168334
18 9.348134 3.0904060  -4.86308501  34.622345 0.168334
```

```
19 3.773971 5.7458114   0.10042819  43.243033 0.168334
20 6.982055 1.7543958  -7.36037002  22.534950 0.168334
```

If one want to get estimates of the GEV parameters at an ungauged locations, this is done by adding a matrix giving the new coordinates. Be careful, if new coordinates are supplied, the column names of the new coordinates should match with the ones of the original coordinates. For our application, this could be done as follows:

```
> new.coord <- cbind(3:6, 7:10)
> colnames(new.coord) <- c("lon", "lat")
> predict(fitted, new.coord)
```

```
  lon lat      loc     scale    shape
1   3   7 2.444771  56.63656 0.168334
2   4   8 4.313982  73.49772 0.168334
3   5   9 6.183194  92.27542 0.168334
4   6  10 8.052405 112.96964 0.168334
```

3.2 Visualising the Extremal Coefficient

The extremal coefficient is a useful quantity to assess the dependence between two locations x_1 and $x_2 \in \mathbb{R}^d$. Assuming that the data could be modeled by a max-stable process with unit Fréchet margin, the extremal coefficient $\theta(x_1 - x_2)$ satisfies:

$$\Pr\left[Z(x_1) \leq z, Z(x_2) \leq z\right] = \exp\left(-\frac{\theta(x_1 - x_2)}{z}\right) \tag{3.1}$$

where $1 \leq \theta(x_1 - x_2) \leq 2$ with the lower and upper bounds corresponding to complete dependence and independence between locations x_1 and x_2.

Consequently, the extremal coefficient function $\theta(\cdot)$ is a natural way to know how evolves the dependence between extremes in space.

The SpatialExtremes package proposes two way to plot the evolution of the extremal coefficient as the distance increases. The first one is to use an empirical estimation of the extremal coefficient; while the second uses a parametric estimation using the Smith's and Schlather's models.

? introduced a methodology to estimate non parametrically the extremal coefficient. The `fitextcoeff` function uses this methodology to get estimates for each pair of stations within the region and plots the evolution of these estimates as the distance increases. In addition, by default, a *lowess* curve can also be plotted to help detecting trends. This is done using the following lines and Figure 3.1 plots the resulting output.

```
> n.site <- 30
> locations <- matrix(runif(2 * n.site, 0, 10), ncol = 2)
> colnames(locations) <- c("lon", "lat")
> ms0 <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = "wh", param = c(0, 1, 0, 30, 0.5), maxstable = "extr",
+     n = 40)
> ms0 <- t(ms0)
> exco <- fitextcoeff(ms0, locations)
```

The closed form of the extremal coefficient function is known for both Smith's and Schlather's models. Namely, the function is given by:

Smith $\theta(x_1 - x_2) = 2\Phi\left(\frac{\sqrt{(x_1-x_2)^T\Sigma^{-1}(x_1-x_2)}}{2}\right)$

Schlather $\theta(||x_1 - x_2||) = 1 + \sqrt{\frac{1-\rho(||x_1-x_2||)}{2}}$

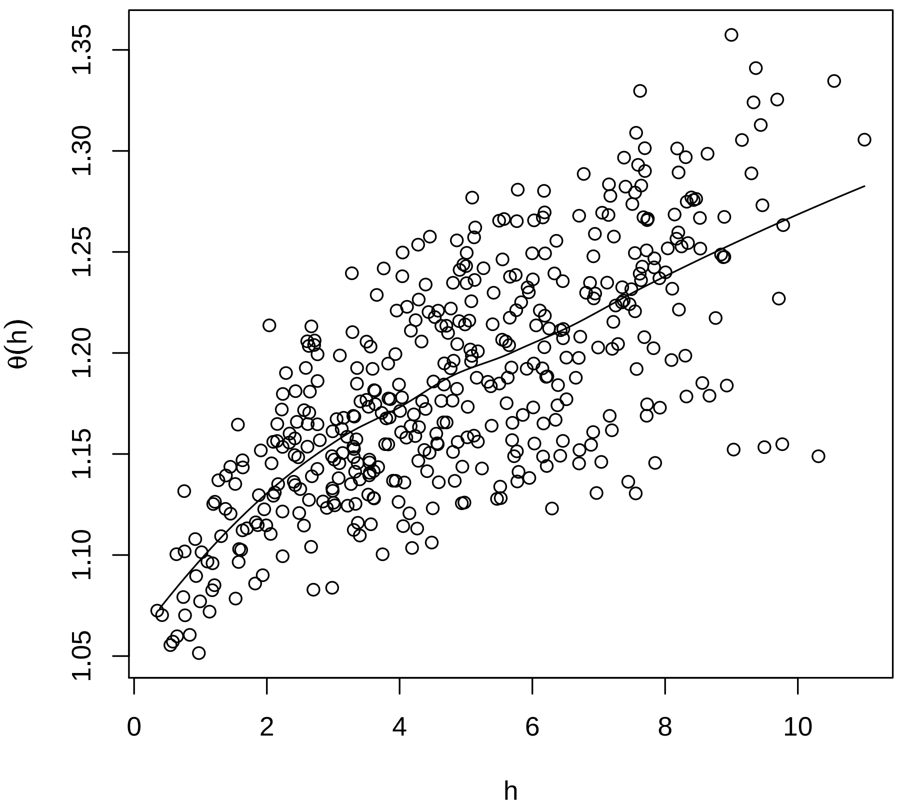


Figure 3.1: Evolution of the (non-parametric) estimates of the extremal coefficient as the distance increases.

The SpatialExtremes package allows to plot the evolution of the extremal coefficient function using the `extcoeff` function - see Fig. 3.2.

```
> fitted <- fitmaxstab(ms0, locations, "whitmat", fit.marge = FALSE)
> extcoeff(fitted)
```

3.3 Visualising the Covariance Function

Another way to assess how evolves the dependence between extremes as the distance increases is to plot the covariance function. This is done using the `covariance` function. Note that this function is (currently) only available for the Schlather's model.

Basically, there are two ways to call the `covariance` function. We can call it once we have fitted a max-stable process or by specifying directly the covariance parameters.

For illustration purpose, Fig. 3.3 compares the fitted covariance function to the theoretical one on a new artificial example.

```
> n.site <- 30
> locations <- matrix(runif(2 * n.site, 0, 10), ncol = 2)
> colnames(locations) <- c("lon", "lat")
```

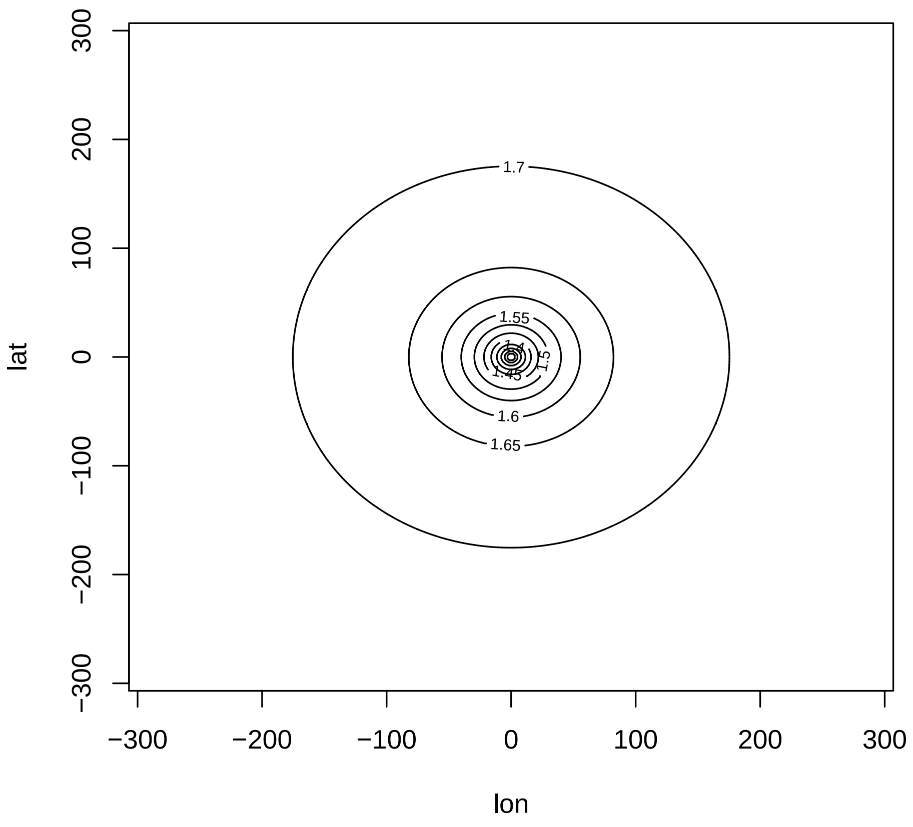


Figure 3.2: Evolution of the extremal coefficient function in $\mathbb{R}^2$.

```
> ms0 <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = "cauchy", param = c(0, 1, 0, 3, 1.2), maxstable = "extr",
+     n = 60)
> ms1 <- t(ms0)
> param.loc <- -10 + 2 * locations[, 2]
> param.scale <- 5 + 2 * locations[, 1] + locations[, 2]^2
> param.shape <- rep(0.2, n.site)
> for (i in 1:n.site) ms1[, i] <- param.scale[i] * (ms1[, i]^param.shape[i] -
+     1)/param.shape[i] + param.loc[i]
> loc.form <- loc ~ lat
> scale.form <- scale ~ lon + I(lat^2)
> shape.form <- shape ~ 1
> fitted <- fitmaxstab(ms1, locations, "cauchy", loc.form, scale.form,
+     shape.form)

Computing appropriate starting values
Starting values are defined
Starting values are:
        sill        range       smooth    locCoeff1    locCoeff2  scaleCoeff1
   0.9969235  221.3374802 9566.1896336  -12.3584913   -0.6289914   -0.2744743
 scaleCoeff2  scaleCoeff3   shapeCoeff
   2.2581711    1.1707514    0.3007569
```

```
> covariance(fitted, ylim = c(0, 1))
> covariance(sill = 1, range = 3, smooth = 1.2, cov.mod = "cauchy",
+     col = 3, add = TRUE)
> legend("topright", c("Fitted", "Theo"), lty = 1, col = c(1, 3),
+     inset = 0.05)
```

Note that one can also compute the covariance at a given distance by invoking:

```
> rbind(fitted = covariance(fitted, dist = seq(0, 10, 3))$cov.val,
+     theo = covariance(sill = 1, range = 3, smooth = 1.2, cov.mod = "cauchy",
+         dist = seq(0, 10, 3))$cov.val)

       [,1]      [,2]       [,3]         [,4]
fitted    1 0.3592970 0.02014778 0.0002809219
theo      1 0.4352753 0.14495593 0.0630957344
```

3.4 Producing a map of the GEV parameters and return levels

Most often, practitioners will like to have a map of the GEV parameters or a map of return levels with a given return period. This is done using the `map` function.

To illustrate this feature, let start with a brand new application.

```
> n.site <- 30
> locations <- matrix(runif(2 * n.site, 0, 10), ncol = 2)
> colnames(locations) <- c("lon", "lat")
> ms0 <- MaxStableRF(locations[, 1], locations[, 2], grid = FALSE,
+     model = "wh", param = c(0, 1, 0, 3, 0.5), maxstable = "extr",
+     n = 40)
> ms0 <- t(ms0)
> ms1 <- ms0
> fun.loc <- function(x) 4.26 * exp(-x) * (1 - exp(-x)) * (1 -
+     3 * exp(-x))
> fun.scale <- function(x) 2 * sin(pi * x/4) + 10
> fun.shape <- function(x) (fun.scale(x) - 7)/15
> param.loc <- fun.loc(locations[, 1])
> param.scale <- fun.scale(locations[, 2])
> param.shape <- fun.shape(locations[, 1])
> for (i in 1:n.site) ms1[, i] <- param.scale[i] * (ms1[, i]^param.shape[i] -
+     1)/param.shape[i] + param.loc[i]
> n.knots <- 7
> knots.lon <- quantile(locations[, 1], 1:n.knots/(n.knots + 1))
> knots.lat <- quantile(locations[, 2], 1:n.knots/(n.knots + 1))
> loc.form <- loc ~ rb(lon, degree = 3, knots = knots.lon, penalty = 0.2)
> scale.form <- scale ~ rb(lat, degree = 3, knots = knots.lat,
+     penalty = 0.5)
> shape.form <- shape ~ rb(lon, degree = 3, knots = knots.lon,
+     penalty = 0.4)
> schlather <- fitmaxstab(ms1, locations, "whitmat", loc.form,
+     scale.form, shape.form)

Computing appropriate starting values
Starting values are defined
Starting values are:
         sill         range        smooth     locCoeff1     locCoeff2
  0.999999997   2.684835073   0.116761750  35.431031342  -8.749898109
```

```
   locCoeff3     locCoeff4     locCoeff5     locCoeff6     locCoeff7
  0.689433270  -1.537202205   2.851620132  -4.526704890   2.332209348
   locCoeff8     locCoeff9   scaleCoeff1   scaleCoeff2   scaleCoeff3
  0.788525720  -0.644652129 -11.169942075  11.174257781  -2.125200812
  scaleCoeff4   scaleCoeff5   scaleCoeff6   scaleCoeff7   scaleCoeff8
  3.190714507  -0.821042797   0.143392104  -1.224905435   3.021195151
  scaleCoeff9   shapeCoeff1   shapeCoeff2   shapeCoeff3   shapeCoeff4
 -1.854435201   0.220418173   0.044478981  -0.022222097   0.064497760
  shapeCoeff5   shapeCoeff6   shapeCoeff7   shapeCoeff8   shapeCoeff9
 -0.149138663   0.301535139  -0.240735358   0.043091130   0.001205809
```

One can have a contour plot (Fig. 3.4) for the evolution of the GEV parameters in $\mathbb{R}^d$ by invoking the following code:

```
> par(mfrow = c(1, 3))
> map(schlather, "loc", col = rainbow(80))
> title("Location")
> map(schlather, "scale", col = heat.colors(80))
> title("Scale")
> map(schlather, "shape", col = topo.colors(100))
> title("Shape")
```

Note that tuning the option `col` will allow users to choose an appropriate color palette.

Fig. 3.5 plots a map of the 50-year return level while focusing on a specific part of the region under study:

```
> new.ranges <- cbind(c(3, 9), c(2, 10))
> colnames(new.ranges) <- c("lon", "lat")
> map(schlather, "quant", ret.per = 50, ranges = new.ranges)
```

```
Computing appropriate starting values
Starting values are defined
Starting values are:
        sill        range       smooth    locCoeff1    locCoeff2  scaleCoeff1
   0.9330906  129.4646528 2974.2901518  -15.5315189   -1.7852388    7.0995311
 scaleCoeff2  scaleCoeff3   shapeCoeff
   0.7649932    0.8527710    0.3259783
```

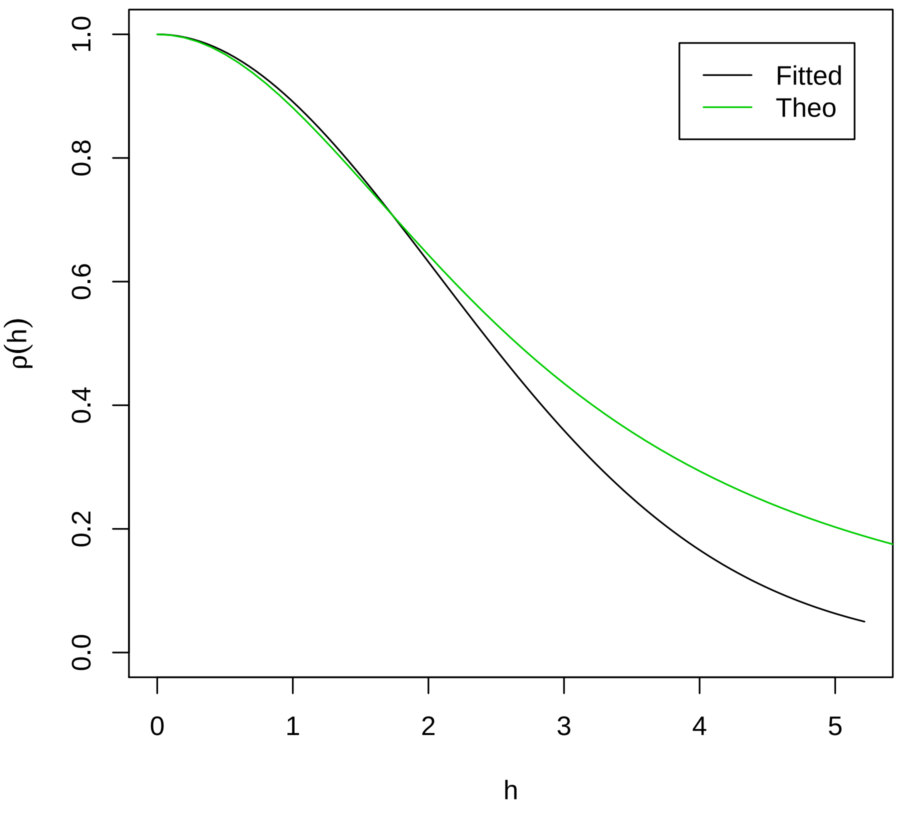


Figure 3.3: Comparison between the fitted covariance function and the theoretical one.

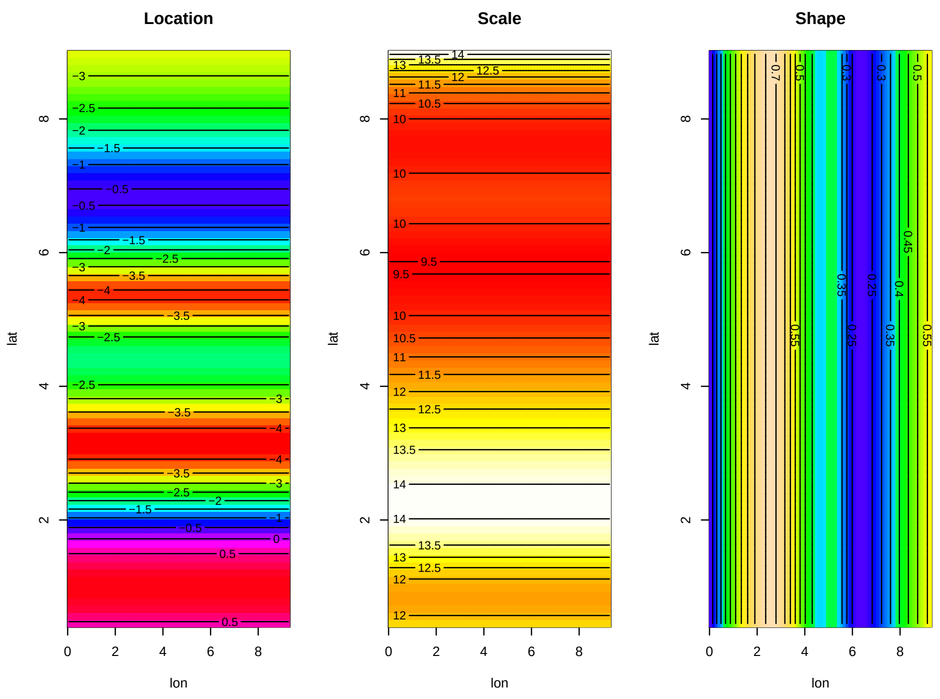


Figure 3.4: Contour plots of the GEV parameters.

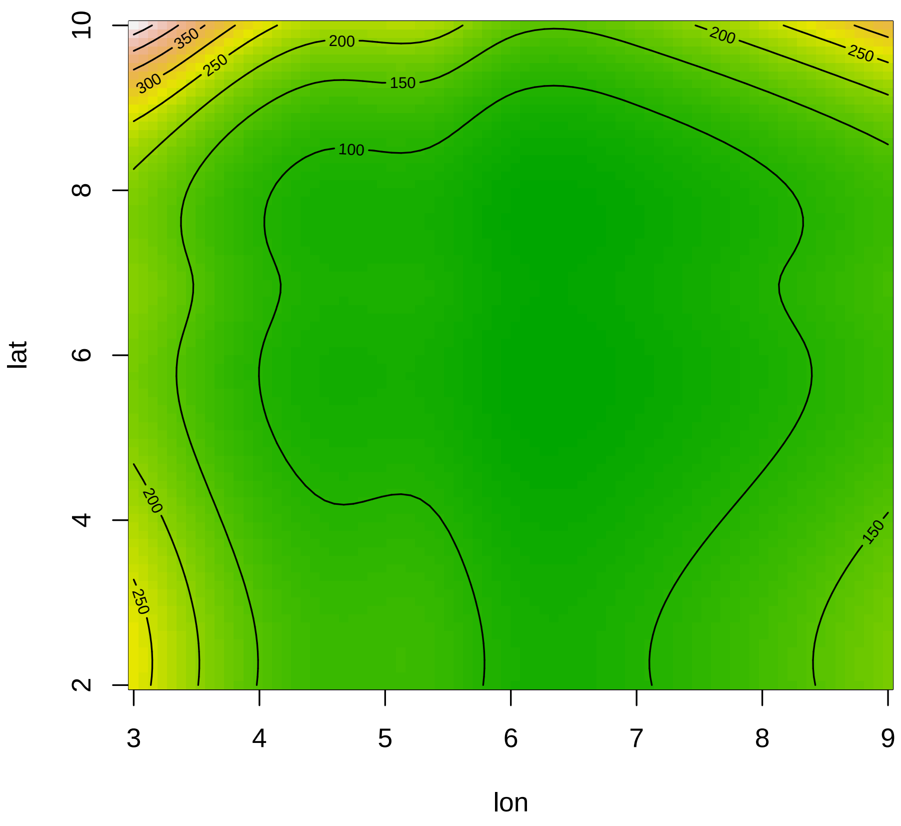


Figure 3.5: Contour plot of the 50-year return level.

Appendix A

Density and Gradient Computations

A.1 The Smith's Model

Let us recall that the Smith's model is given by:

$$\Pr[Z_1 \leq z_1, Z_2 \leq z_2] = \exp\left[-\frac{1}{z_1}\Phi\left(\frac{a}{2}+\frac{1}{a}\log\frac{z_2}{z_1}\right) - \frac{1}{z_2}\Phi\left(\frac{a}{2}+\frac{1}{a}\log\frac{z_1}{z_2}\right)\right] \tag{A.1}$$

where Φ is the standard normal cumulative distribution function and, for two locations #1 and #2

$$a^2 = \Delta x^T \Sigma^{-1} \Delta x \quad \text{and} \quad \Sigma = \begin{bmatrix} cov_{11} & cov_{12} \\ cov_{12} & cov_{22} \end{bmatrix} \quad \text{or} \quad \Sigma = \begin{bmatrix} cov_{11} & cov_{12} & cov_{13} \\ cov_{12} & cov_{22} & cov_{23} \\ cov_{13} & cov_{23} & cov_{33} \end{bmatrix}$$

where Δx is the distance vector between location #1 and location #2.

A.1.1 Useful quantities

Computation of the density as well as the gradient of the density is not difficult but "heavy" though. For computation facilities and to help readers, we define:

$$c_1 = \frac{a}{2} + \frac{1}{a}\log\frac{z_2}{z_1} \quad \text{and} \quad c_2 = \frac{a}{2} + \frac{1}{a}\log\frac{z_1}{z_2} \tag{A.2}$$

From these definitions, we note that $c_1 + c_2 = a$.

A.1.2 Density computation

From (A.1), we note the standard normal distribution appears. Consequently, we need to compute its derivatives at c_1 and c_2 with respect to z_1 and z_2.

$$\frac{\partial c_1}{\partial z_1} = \frac{1}{a}\left(-\frac{z_2}{z_1^2}\frac{z_1}{z_2}\right) = -\frac{1}{az_1} \qquad \frac{\partial c_1}{\partial z_2} = \frac{1}{a}\frac{1}{z_1}\frac{z_1}{z_2} = \frac{1}{az_2} \tag{A.3}$$

$$\frac{\partial c_2}{\partial z_1} = -\frac{\partial c_1}{\partial z_1} = \frac{1}{az_1} \qquad \frac{\partial c_2}{\partial z_2} = -\frac{\partial c_1}{\partial z_2} = -\frac{1}{az_2} \tag{A.4}$$

As the normal distribution appears in the Smith's characterisation, the following quantities will be useful:

$$\frac{\partial\Phi(c_1)}{\partial z_1} = \frac{\partial\Phi(c_1)}{\partial c_1}\frac{\partial c_1}{\partial z_1} = -\frac{\varphi(c_1)}{az_1} \qquad \frac{\partial\Phi(c_1)}{\partial z_2} = \frac{\partial\Phi(c_1)}{\partial c_1}\frac{\partial c_1}{\partial z_2} = \frac{\varphi(c_1)}{az_2} \tag{A.5}$$

$$\frac{\partial\Phi(c_2)}{\partial z_1} = \frac{\partial\Phi(c_2)}{\partial c_2}\frac{\partial c_2}{\partial z_1} = \frac{\varphi(c_2)}{az_1} \qquad \frac{\partial\Phi(c_2)}{\partial z_2} = \frac{\partial\Phi(c_2)}{\partial c_2}\frac{\partial c_2}{\partial z_2} = -\frac{\varphi(c_2)}{az_2} \tag{A.6}$$

$$\frac{\partial\varphi(c_1)}{\partial z_1} = \frac{\partial\varphi(c_1)}{\partial c_1}\frac{\partial c_1}{\partial z_1} = \frac{c_1\varphi(c_1)}{az_1} \qquad \frac{\partial\varphi(c_1)}{z_2} = \frac{\partial\varphi(c_1)}{\partial c_1}\frac{\partial c_1}{\partial z_2} = -\frac{c_1\varphi(c_1)}{az_2} \tag{A.7}$$

$$\frac{\partial\varphi(c_2)}{\partial z_1} = \frac{\partial\varphi(c_2)}{\partial c_2}\frac{\partial c_2}{\partial z_1} = -\frac{c_2\varphi(c_2)}{az_1} \qquad \frac{\partial\varphi(c_2)}{\partial z_2} = \frac{\partial\varphi(c_2)}{\partial c_2}\frac{\partial c_2}{\partial z_2} = \frac{c_2\varphi(c_2)}{az_2} \tag{A.8}$$

Define

$$A = \frac{1}{z_1}\Phi(c_1) \quad \text{and} \quad B = \frac{1}{z_2}\Phi(c_2) \tag{A.9}$$

Consequently, $F(z_1, z_2) = exp(-A - B)$ and

$$\frac{\partial F}{\partial z_1}(z_1, z_2) = -\left(\frac{\partial A}{\partial z_1} + \frac{\partial B}{\partial z_1}\right) F(z_1, z_2) \qquad \frac{\partial F}{\partial z_2}(z_1, z_2) = -\left(\frac{\partial A}{\partial z_2} + \frac{\partial B}{\partial z_2}\right) F(z_1, z_2) \tag{A.10}$$

By noting that

$$\frac{\partial A}{\partial z_1} = -\frac{\Phi(c_1)}{z_1^2} + \frac{1}{z_1}\left(-\frac{\varphi(c_1)}{az_1}\right) = -\frac{\Phi(c_1)}{z_1^2} - \frac{\varphi(c_1)}{az_1^2} \tag{A.11}$$

$$\frac{\partial B}{\partial z_1} = \frac{1}{z_2}\frac{\varphi(c_2)}{az_1} = \frac{\varphi(c_2)}{az_1z_2} \tag{A.12}$$

$$\frac{\partial A}{\partial z_2} = \frac{1}{z_1}\frac{\varphi(c_1)}{az_2} = \frac{\varphi(c_1)}{az_1z_2} \tag{A.13}$$

$$\frac{\partial B}{\partial z_2} = -\frac{\Phi(c_2)}{z_2^2} + \frac{1}{z_2}\left(-\frac{\varphi(c_2)}{az_2}\right) = -\frac{\Phi(c_2)}{z_2^2} - \frac{\varphi(c_2)}{az_2^2} \tag{A.14}$$

and

$$\frac{\partial^2 A}{\partial z_2 \partial z_1} = \frac{\partial}{\partial z_2}\left(-\frac{\Phi(c_1)}{z_1^2} - \frac{\varphi(c_1)}{az_1^2}\right) = -\frac{\varphi(c_1)}{az_1^2z_2} + \frac{c_1\varphi(c_1)}{a^2z_1^2z_2} = -\frac{c_2\varphi(c_1)}{a^2z_1^2z_2} \tag{A.15}$$

$$\frac{\partial^2 B}{\partial z_2 \partial z_1} = \frac{\partial}{\partial z_2}\frac{\varphi(c_2)}{az_1z_2} = -\frac{c_1\varphi(c_2)}{a^2z_1z_2^2} \tag{A.16}$$

So that,

$$\frac{\partial F}{\partial z_1}(z_1, z_2) = \left(\frac{\Phi(c_1)}{z_1^2} + \frac{\varphi(c_1)}{az_1^2} - \frac{\varphi(c_2)}{az_1z_2}\right) F(z_1, z_2) \tag{A.17}$$

$$\frac{\partial F}{\partial z_2}(z_1, z_2) = \left(\frac{\Phi(c_2)}{z_2^2} + \frac{\varphi(c_2)}{az_2^2} - \frac{\varphi(c_1)}{az_1z_2}\right) F(z_1, z_2) \tag{A.18}$$

Finally,

$$\frac{\partial^2 F}{\partial z_2 \partial z_1}(z_1, z_2) = -\left(\frac{\partial^2 A}{\partial z_2 \partial z_1} + \frac{\partial^2 B}{\partial z_2 \partial z_1}\right) F(z_1, z_2) - \left(\frac{\partial A}{\partial z_1} + \frac{\partial B}{\partial z_1}\right)\frac{\partial F}{\partial z_2}(z_1, z_2) \tag{A.19}$$

Thus, it leads to the following relation:

$$f(z_1, z_2) = \left[\frac{c_2\varphi(c_1)}{a^2z_1^2z_2} + \frac{c_1\varphi(c_2)}{a^2z_1z_2^2} + \left(\frac{\Phi(c_1)}{z_1^2} + \frac{\varphi(c_1)}{az_1^2} - \frac{\varphi(c_2)}{az_1z_2}\right)\left(\frac{\Phi(c_2)}{z_2^2} + \frac{\varphi(c_2)}{az_2^2} - \frac{\varphi(c_1)}{az_1z_2}\right)\right] F(z_1, z_2) \tag{A.20}$$

A.1.3 Gradient computation

As said in section 2.3, the maximum pairwise likelihood estimator ψ_p satisfies:

$$\psi_p \sim \mathcal{N}\left(\psi, H^{-1} J H^{-1}\right)$$

where H is the Fisher information matrix and J as defined in Section 2.3. This section aims to derive analytical form for J.

Let us recall that the log pairwise likelihood is defined by:

$$\ell_p(\mathbf{z}, \Psi) = \sum_{k=1}^{n_{obs}} \sum_{i=1}^{n_{site}-1} \sum_{j=i+1}^{n_{site}} \log f(z_k^{(i)}, z_k^{(j)})$$

where n_{obs} is the number of observations, $\mathbf{z}_k = (z_k^{(1)}, \dots, z_k^{(n_{site})})$ is the k-th observation vector, n_{site} is the number of site within the region and f is the bivariate density.

Consequently, the gradient of the log pairwise density is given by:

$$\nabla \ell_p(\Psi) = \sum_{i=1}^{n_{site}-1} \sum_{j=i+1}^{n_{site}} \nabla \log f(z_k^{(i)}, z_k^{(j)})$$

Define:

$$\begin{aligned}
A &= -\frac{\Phi(c1)}{z_1} - \frac{\Phi(c2)}{z_2} \\
B &= \frac{\Phi(c_2)}{z_2^2} + \frac{\varphi(c_2)}{a z_2^2} - \frac{\varphi(c_1)}{a z_1 z_2} \\
C &= \frac{\Phi(c_1)}{z_1^2} + \frac{\varphi(c_1)}{a z_1^2} - \frac{\varphi(c_2)}{a z_1 z_2} \\
D &= \frac{c_2 \varphi(c_1)}{a^2 z_1^2 z_2} + \frac{c_1 \varphi(c_2)}{a^2 z_1 z_2^2}
\end{aligned}$$

so that,

$$\log f(z_k^{(i)}, z_k^{(j)}) = A + log(BC + D)$$

With Unit Fréchet Margins

For clarity purposes, let start our computations assuming that the observations have unit Fréchet margins. For this special case, the logarithm of the bivariate density f is only a function of the Mahalanobis distance a, the gradient w.r.t. the covariance matrix elements cov_{11}, cov_{12} and cov_{22} is given through the following relation[1]:

$$\nabla_\Sigma \log f(z_k^{(i)}, z_k^{(j)}) = \frac{\partial}{\partial a} \log f(z_k^{(i)}, z_k^{(j)}) \nabla_\Sigma a^T$$

where $\nabla_\Sigma a$ is the gradient of the Mahalanobis distance w.r.t. the covariance matrix element i.e. $(\frac{\partial a}{\partial cov_{11}}, \frac{\partial a}{\partial cov_{12}}, \frac{\partial a}{\partial cov_{22}})$.

For clarity purposes, we first compute the following quantities:

$$\begin{aligned}
\frac{\partial c_1}{\partial a} &= \frac{1}{2} - \frac{1}{a^2} \log \frac{z_2}{z_1} = \frac{c_2}{a} & \frac{\partial c_2}{\partial a} &= \frac{c_1}{a} \\
\frac{\partial \Phi(c_1)}{\partial a} &= \frac{\partial \Phi(c_1)}{\partial c_1} \frac{\partial c_1}{\partial a} = \frac{c_2 \varphi(c_1)}{a} & \frac{\partial \Phi(c_2)}{\partial a} &= \frac{c_1 \varphi(c_2)}{a} \\
\frac{\partial \varphi(c_1)}{\partial a} &= \frac{\partial \varphi(c_1)}{\partial c_1} \frac{\partial c_1}{\partial a} = -\frac{c_1 c_2 \varphi(c_1)}{a} & \frac{\partial \varphi(c_2)}{\partial a} &= -\frac{c_1 c_2 \varphi(c_2)}{a} \\
\frac{\partial c_2 \varphi(c_1)}{\partial a} &= \frac{c_1(1 - c_2^2)\varphi(c_1)}{a} & \frac{\partial c_1 \varphi(c_2)}{\partial a} &= \frac{(1 - c_1^2) c_2 \varphi(c_2)}{a}
\end{aligned}$$

[1]algebra operators are defined component-wise.

Consequently, we have:

$$
\begin{aligned}
dA_a &= \frac{\partial A}{\partial a} = -\frac{1}{z_1}\frac{c_2\varphi(c_1)}{a} - \frac{1}{z_2}\frac{c_1\varphi(c_2)}{a} = -\frac{c_2\varphi(c_1)}{az_1} - \frac{c_1\varphi(c_2)}{az_2} \\
dC_a &= \frac{\partial C}{\partial a} = \frac{1}{z_1^2}\frac{c_2\varphi(c_1)}{a} + \frac{1}{z_1^2}\frac{-\frac{c_1c_2\varphi(c_1)}{a}a - \varphi(c_1)}{a^2} - \frac{1}{z_1z_2}\frac{-\frac{c_1c_2\varphi(c_2)}{a}a - \varphi(c_2)}{a^2} \\
&= \frac{c_2\varphi(c_1)}{az_1^2} - \frac{(1+c_1c_2)\varphi(c_1)}{a^2z_1^2} + \frac{(1+c_1c_2)\varphi(c_2)}{a^2z_1z_2} \\
&= \frac{\left[c_2(a-c_1)-1\right]\varphi(c_1)}{a^2z_1^2} + \frac{(1+c_1c_2)\varphi(c_2)}{a^2z_1z_2} \\
&= \frac{(c_2^2-1)\varphi(c_1)}{a^2z_1^2} + \frac{(1+c_1c_2)\varphi(c_2)}{a^2z_1z_2} \\
dB_a &= \frac{\partial B}{\partial a} = \frac{(c_1^2-1)\varphi(c_2)}{a^2z_2^2} + \frac{(1+c_1c_2)\varphi(c_1)}{a^2z_1z_2} \\
dD_a &= \frac{\partial D}{\partial a} = \frac{1}{z_1^2z_2}\frac{\frac{c_1(1-c_2^2)\varphi(c_1)}{a}a^2 - 2ac_2\varphi(c_1)}{a^4} + \frac{1}{z_1z_2^2}\frac{\frac{(1-c1^2)c_2\varphi(c_2)}{a}a^2 - 2ac_1\varphi(c_2)}{a^4} \\
&= \frac{(c_1-2c_2-c_1c_2^2)\varphi(c_1)}{a^3z_1^2z_2} + \frac{(c_2-2c_1-c_1^2c_2)\varphi(c_2)}{a^3z_1z_2^2}
\end{aligned}
$$

Finally,

$$
\nabla_\Sigma \log f(x_k^{(i)}, x_k^{(j)}) = \left[dA_a + \frac{(CdB_a + BdC_a + dD_a)}{BC+D}\right]\cdot \nabla_\Sigma a^T
$$

With Ordinary GEV Margins

In the previous section, we derived the gradient assuming unit Fréchet margins. Now, we consider the more general case where margins are supposed to be ordinary GEV.

We have to be aware that the bivariate density changes when we do not suppose unit Fréchet margins anymore. For instance, the bivariate density evaluated at two observations y_1 and y_2 with ordinary GEV margins is now given by:

$$
f(y_1, y_2) = f(z_1, z_2)|J(y_1, y_2)| \tag{A.21}
$$

where z_1 (resp. z_2) is the transformation of y_1 (resp. y_2) to the unit Fréchet scale and $|J(y_1, y_2)|$ is the determinant of the Jacobian related to the transformation $(y_1, y_2) \mapsto (z_1, z_2)$.

For clarity purpose, we can write the logarithm of the bivariate density as follows:

$$
\log f(y_1, y_2) = A + \log\left(BC + D\right) + E
$$

where $E = \log|J(y_1, y_2)|$ and the quantities A, B, C and D are the same as in the previous section.

The transformation from y_i to z_i is given by:

$$
z_i = \left(1 + \xi_i\frac{y_i - \mu_i}{\sigma_i}\right)_+^{\frac{1}{\xi_i}} \tag{A.22}
$$

where μ_i, σ_i and ξ_i are the GEV location, scale and shape parameters and $x_+ = \min(0, x)$.

Consequently, we need a response surface (see section 2.2) to model the evolution of the GEV parameters in space. Let suppose that we have a polynomial response surface for each GEV parameter, one can write:

$$
\begin{aligned}
\mu &= X_\mu\beta_\mu & \text{(A.23)}\\
\sigma &= X_\sigma\beta_\sigma & \text{(A.24)}\\
\xi &= X_\xi\beta_\xi & \text{(A.25)}
\end{aligned}
$$

where $\mu = (\mu_1, \ldots, \mu_{n_{site}})$, $\sigma = (\sigma_1, \ldots, \sigma_{n_{site}})$ and $\xi = (\xi_1, \ldots, \xi_{n_{site}})$ are the vector for the location, scale and shape GEV parameters for all the sites within the region study, X_μ, X_σ and X_ξ are the design matrices for each GEV parameters and β_μ, β_σ and β_ξ are the regression coefficients to be estimated.

Consequently, from one ordinary GEV observation $\mathbf{y}$, one can transform it to unit Fréchet margins using the following transformation:

$$\mathbf{z}_i = \left\{ 1 + \frac{X_\xi^{(i)} \beta_\xi (\mathbf{y}_i - X_\mu^{(i)} \beta_\mu)}{X_\sigma^{(i)} \beta_\sigma} \right\}^{1/(X_\xi^{(i)} \beta_\xi)}, \qquad i = 1, \ldots, n_{site} \tag{A.26}$$

where $X^{(i)}$ stands for the i-th row of the design matrix X and $\mathbf{z}_i$ denotes the i-th element of the vector $\mathbf{z}$.

Consequently, $|J(y_1, y_2)|$ is given by:

$$|J(y_1, y_2)| = \frac{1}{X_\sigma^{(i)} \beta_\sigma X_\sigma^{(j)} \beta_\sigma} \left(1 + X_\xi^{(i)} \beta_\xi \frac{y_i - X_\mu^{(i)} \beta_\mu}{X_\sigma^{(i)} \beta_\sigma} \right)_+^{\frac{1}{X_\xi^{(i)} \beta_\xi} - 1} \left(1 + X_\xi^{(j)} \beta_\xi \frac{y_2 - X_\mu^{(j)} \beta_\mu}{X_\sigma^{(j)} \beta_\sigma} \right)_+^{\frac{1}{X_\xi^{(j)} \beta_\xi} - 1} \tag{A.27}$$

It is easy to see that:

$$\begin{aligned}
\frac{\partial \mathbf{z}_i}{\partial \beta_\mu} &= -\frac{\mathbf{z}_i^{1 - X_\xi^{(i)} \beta_\xi} X_\mu^{(i)}}{X_\sigma^{(i)} \beta_\sigma} \\
&= -\frac{\mathbf{z}_i^{1-\xi_i}}{\sigma_i} \cdot X_\mu^{(i)} \\
\frac{\partial \mathbf{z}_i}{\partial \beta_\sigma} &= -\frac{\mathbf{z}_i^{1 - X_\xi^{(i)} \beta_\xi} \left(\mathbf{y}_i - X_\mu^{(i)} \beta_\mu \right)}{X_\sigma^{(i)} \beta_\sigma^2} \\
&= -\frac{\mathbf{z}_i^{1-\xi_i} (\mathbf{y}_i - \mu_i)}{\sigma_i} \cdot \frac{1}{\beta_\sigma} \\
\frac{\partial \mathbf{z}_i}{\partial \beta_\xi} &= -\frac{\mathbf{z}_i \log \mathbf{z}_i}{\beta_\xi} + \frac{\mathbf{z}^{(i)} \left(\mathbf{y}_i - X_\mu^{(i)} \beta_\mu \right)}{\beta_\xi X_\sigma^{(i)} \beta_\sigma \mathbf{z}_i^{X_\xi^{(i)} \beta_\xi}} \\
&= \left[\mathbf{z}_i^{1-\xi_i} \frac{(\mathbf{y}_i - \mu_i)}{\sigma_i} - \mathbf{z}_i \log \mathbf{z}_i \right] \cdot \frac{1}{\beta_\xi}
\end{aligned}$$

where the operator $\cdot$ performs operations component-wise.

To obtain the gradient of the logarithm of the bivariate density, we need to compute the partial derivatives of A, B, C, D and E w.r.t. β_μ, β_σ and β_ξ.

For shortness, we do it in "one step" with the convention $\beta = (\beta_\mu, \beta_\sigma, \beta_\xi)$.

$$\begin{aligned}
\frac{\partial A}{\partial \beta} &= \frac{\partial A}{\partial z_1} \cdot \nabla_\beta z_1 + \frac{\partial A}{\partial z_2} \cdot \nabla_\beta z_2 \\
\frac{\partial B}{\partial \beta} &= \frac{\partial B}{\partial z_1} \cdot \nabla_\beta z_1 + \frac{\partial B}{\partial z_2} \cdot \nabla_\beta z_2 \\
\frac{\partial C}{\partial \beta} &= \frac{\partial C}{\partial z_1} \cdot \nabla_\beta z_1 + \frac{\partial C}{\partial z_2} \cdot \nabla_\beta z_2 \\
\frac{\partial D}{\partial \beta} &= \frac{\partial D}{\partial z_1} \cdot \nabla_\beta z_1 + \frac{\partial D}{\partial z_2} \cdot \nabla_\beta z_2
\end{aligned}$$

where $\nabla_\beta z_1$ (resp. $\nabla_\beta z_2$) is the gradient of z_1 (reps. z_2) w.r.t. β and the partial derivatives of A, B, C

and D w.r.t. z_1 are given by the following equations:

$$\begin{aligned}
dA_{z_1} &= \frac{\partial A}{\partial z_1} = \frac{\varphi(c_1) + a\Phi(c_1)}{az_1^2} - \frac{\varphi(c_2)}{az_1z_2} \\
dB_{z_1} &= \frac{\partial B}{\partial z_1} = \frac{c_1\varphi(c_2)}{a^2z_1z_2^2} + \frac{c_2\varphi(c_1)}{a^2z_1^2z_2} \\
dC_{z_1} &= \frac{\partial C}{\partial z_1} = \frac{(a+c_2)\varphi(c_2)}{a^2z_1^2z_2} - \frac{2\Phi(c_1)}{z_1^3} - \frac{(2a+c_2)\varphi(c_1)}{a^2z_1^3} \\
dD_{z_1} &= \frac{\partial D}{\partial z_1} = \frac{[1 - c_2(a+c_2)]\,\varphi(c_1)}{a^2z_1^3z_2} - \frac{[1 + c_1(a+c_2)]\,\varphi(c_2)}{a^3z_1^2z_2^2}
\end{aligned}$$

while the partial derivatives of A, B, C and D w.r.t. z_2 are given by:

$$\begin{aligned}
dA_{z_2} &= \frac{\partial A}{\partial z_2} = \frac{\varphi(c_2) + a\Phi(c_2)}{az_2^2} - \frac{\varphi(c_1)}{az_1z_2} \\
dB_{z_2} &= \frac{\partial B}{\partial z_2} = \frac{(a+c_1)\varphi(c_1)}{a^2z_1z_2^2} - \frac{2\Phi(c_2)}{z_2^3} - \frac{(2a+c_1)\varphi(c_2)}{a^2z_2^3} \\
dC_{z_2} &= \frac{\partial C}{\partial z_2} = \frac{c_1\varphi(c_2)}{a^2z_1z_2^2} + \frac{c_2\varphi(c_1)}{a^2z_1^2z_2} \\
dD_{z_2} &= \frac{\partial D}{\partial z_2} = \frac{[1 - c_1(a+c_1)]\,\varphi(c_2)}{a^2z_1z_2^3} - \frac{[1 + c_2(a+c_1)]\,\varphi(c_1)}{a^3z_1^2z_2^2}
\end{aligned}$$

For the Jacobian part E, we have:

$$\begin{aligned}
dE_\mu &= \frac{\partial E}{\partial \beta_\mu} = \frac{\xi_1 - 1}{\sigma_1 z_1^{\xi_1}} \cdot X_\mu^{(1)} + \frac{\xi_2 - 1}{\sigma_2 z_2^{\xi_2}} \cdot X_\mu^{(2)} \\
dE_\sigma &= \frac{\partial E}{\partial \beta_\sigma} = \left(\frac{(y_1 - \mu_1)(\xi_1 - 1)}{\sigma_1 z_1^{\xi_1}} + \frac{(y_2 - \mu_2)(\xi_2 - 1)}{\sigma_2 z_2^{\xi_2}} - 2 \right) \cdot \frac{1}{\beta_\sigma} \\
dE_\xi &= \frac{\partial E}{\partial \beta_\xi} = \frac{(1 - \xi_1)(y_1 - \mu_1)}{\sigma_1 \xi_1 z_1^{\xi_1}} \cdot X_\xi^{(1)} + \frac{(1 - \xi_2)(y_2 - \mu_2)}{\sigma_2 \xi_2 z_2^{\xi_2}} \cdot X_\xi^{(2)} - \log z_1 \cdot \frac{1}{\beta_\xi} - \log z_2 \frac{1}{\beta_\xi}
\end{aligned}$$

Finally, we have:

$$\nabla_\beta \log f(y_1, y_2) = \frac{\partial A}{\partial \beta} + \frac{C\frac{\partial B}{\partial \beta} + B\frac{\partial C}{\partial \beta}}{BC + D} + \frac{\partial E}{\partial \beta} \tag{A.28}$$

where

$$\frac{\partial E}{\partial \beta} = (dE_\mu, dE_\sigma, dE_\xi)^T$$

A.2 The Schlather's Model

The Schlather's model is given by:

$$\Pr[Z_1 \leq z_1, Z_2 \leq z_2] = \exp\left[-\frac{1}{2}\left(\frac{1}{z_1} + \frac{1}{z_2}\right)\left(1 + \sqrt{1 - 2(\rho(h) + 1)\frac{z_1z_2}{(z_1 + z_2)^2}}\right)\right] \tag{A.29}$$

where h is the distance between location #1 and location #2 and $\rho(h)$ is a valid correlation function such as $-1 \leq \rho(h) \leq 1$.

A.2.1 Density computation

Computation of the density as well as the gradient of the density is not difficult but "heavy" though.

By noting that,

$$\frac{\partial^2}{\partial z_1 \partial z_2} \exp(V(z_1, z_2)) = \left[\frac{\partial^2}{\partial z_1 \partial z_2} V(z_1, z_2) + \left(\frac{\partial}{\partial z_1} V(z_1, z_2)\right)\left(\frac{\partial}{\partial z_2} V(z_1, z_2)\right)\right] \exp(V(z_1, z_2))$$

where $V(z_1, z_2)$ is any function in $\mathcal{C}^2$.

Consequently, to compute the (bivariate) density, we only need to compute the partial derivatives and the mixed partial derivatives. For our case, it turns out to be:

$$V(z_1, z_2) = -\frac{1}{2}\left(\frac{1}{z_1} + \frac{1}{z_2}\right)\left(1 + \sqrt{1 - 2(\rho(h) + 1)\frac{z_1 z_2}{(z_1 + z_2)^2}}\right)$$

$$\frac{\partial}{\partial z_1} V(z_1, z_2) = -\frac{\rho(h) z_1 - c1 - z_2}{2 c_1 z_1^2} \quad \frac{\partial}{\partial z_2} V(z_1, z_2) = -\frac{\rho(h) z_2 - c1 - z_1}{2 c_1 z_2^2} \quad \frac{\partial^2}{\partial z_1 \partial z_2} V(z_1, z_2) = \frac{1 - \rho(h)^2}{2 c_1^3}$$

where

$$c_1 = \sqrt{z_1^2 + z_2^2 - 2 z_1 z_2 \rho(h)}$$

Lastly,

$$f(z_1, z_2) = \left[\frac{1 - \rho(h)^2}{2 c_1^3} + \left(-\frac{\rho(h) z_1 - c1 - z_2}{2 c_1 z_1^2}\right)\left(-\frac{\rho(h) z_2 - c1 - z_1}{2 c_1 z_2^2}\right)\right] \exp(V(z_1, z_2)) \qquad (A.30)$$

A.2.2 Gradient computation

With Unit Fréchet Margins

From equation (A.30), we have:

$$\log f(z_1, z_2) = A + \log(B + CD)$$

where

$$A = V(z_1, z_2) \quad B = \frac{1 - \rho(h)^2}{2 c_1^3} \quad C = -\frac{\rho(h) z_1 - c1 - z_2}{2 c_1 z_1^2} \quad D = -\frac{\rho(h) z_2 - c1 - z_1}{2 c_1 z_2^2}$$

As the bivariate density is only a function of the covariance function $\rho(h)$, we have:

$$\nabla \log f(z_1, z_2) = \frac{\partial}{\partial \rho(h)} \log f(z_1, z_2) \left(\nabla \rho(h)\right)^T$$

where $\nabla \rho(h)$ is the vector of the partial derivatives of the covariance function $\rho(h)$ with respect to its parameters.

$$\begin{aligned}
dA_\rho &= \frac{\partial A}{\partial \rho(h)} = \frac{1}{2 c_1} \\
dB_\rho &= \frac{\partial B}{\partial \rho(h)} = -\frac{\rho(h)}{c_1^3} + \frac{3(1 - \rho(h)) z_1 z_2}{c_1^5} \\
dC_\rho &= \frac{\partial C}{\partial \rho(h)} = -\frac{z_1 - z_2 \rho(h)}{2 c_1^3} \\
dD_\rho &= \frac{\partial D}{\partial \rho(h)} = -\frac{z_2 - z_1 \rho(h)}{2 c_1^3}
\end{aligned}$$

So that,

$$\nabla \log f(z_1, z_2) = \left[dA_\rho + \frac{(C dB_\rho + B dC_\rho + dD_\rho)}{BC + D}\right] \left(\nabla \rho(h)\right)^T$$

Note that when using the Whittle-Matérn covariance function, the standard errors are not available if the `smooth` parameter is hand fixed because the Bessel function is not derivable w.r.t. this parameter.

With Ordinary GEV Margins

For the derivation of the gradient with ordinary GEV margins, most of the computations have already been done in Section A.1.3. Especially, we only need to compute the partial derivatives of A, B, C and D w.r.t. z_1 and z_2.

$$\begin{aligned}
dA_{z_1} &= \frac{\partial A}{\partial z_1} = -\frac{\rho(h)z_1 - c1 - z_2}{2c_1 z_1^2} \\
dB_{z_1} &= \frac{\partial B}{\partial z_1} = \frac{3(\rho(h)^2-1)(z_1-\rho(h)z_2)}{2c_1^5} \\
dC_{z_1} &= \frac{\partial C}{\partial z_1} = \frac{2z_1^3\rho(h) + 6z_1z_2^2\rho(h)^2 - 3z_1^2z_2(1+\rho(h)^2) - 2c_1^3 - 2z_2^3}{2c_1^3z_1^3} \\
dD_{z_1} &= \frac{\partial C}{\partial z_2} = -\frac{(z_2\rho(h) - c_1 - z_1)(z_2\rho(h) + c1 - z_1)}{2c_1^3z_2^2}
\end{aligned}$$

and

$$\begin{aligned}
dA_{z_2} &= \frac{\partial A}{\partial z_2} = -\frac{\rho(h)z_2 - c1 - z_1}{2c_1 z_2^2} \\
dB_{z_2} &= \frac{\partial B}{\partial z_2} = \frac{3(\rho(h)^2-1)(z_2-\rho(h)z_1)}{2c_1^5} \\
dC_{z_2} &= \frac{\partial C}{\partial z_2} = -\frac{(z_1\rho(h) - c_1 - z_2)(z_1\rho(h) + c1 - z_2)}{2c_1^3z_1^2} \\
dD_{z_2} &= \frac{\partial D}{\partial z_2} = \frac{2z_2^3\rho(h) + 6z_1^2z_2\rho(h)^2 - 3z_1z_2^2(1+\rho(h)^2) - 2c_1^3 - 2z_1^3}{2c_1^3z_2^3}
\end{aligned}$$

Finally, we have:

$$\nabla_\beta \log f(y_1, y_2) = \frac{\partial A}{\partial \beta} + \frac{C\frac{\partial B}{\partial \beta} + B\frac{\partial C}{\partial \beta}}{BC + D} + \frac{\partial E}{\partial \beta} \tag{A.31}$$

where $\frac{\partial A}{\partial \beta}$, $\frac{\partial B}{\partial \beta}$, $\frac{\partial C}{\partial \beta}$, $\frac{\partial D}{\partial \beta}$ and $\frac{\partial E}{\partial \beta}$ have been already defined in Section A.1.3.