

S-PLUS functions for extreme value modeling:

An accompaniment to the book *An Introduction to Statistical Modeling of Extreme Values*

Stuart Coles

1 Introduction

This document describes the S-PLUS functions written to support the computations carried out in *An Introduction to Statistical Modeling of Extreme Values*. The functions are intended for **illustrative purposes only** and no responsibility is accepted for their functionality or for their accuracy in any particular application. The datasets used in the book are also available as a separate file.

The files have been prepared using the `dump` command in S-PLUS. This means you should:

1. download the files and store under a filename of your choice;
2. open an S-PLUS session;
3. use the `source()` function to load both the functions and datasets.

For example, if you had stored the two files as `extreme.fns` and `extreme.dat`, you would type the commands `source("extreme.fns")` and `source("extreme.dat")` to copy the functions and datasets respectively into your current S-PLUS working directory.

2 Contents

2.1 Datasets

`dowjones.data` 1304×2 matrix containing daily closing prices of the Dow Jones Index. First column contains the calendar data; second column gives the index value.

`engine.data` 32×2 matrix giving engine failure time data. First column contains the corrosion level; second column gives engine failure time.

`euroex.data` vector of length 975 giving daily exchange rate values of the euro relative to the pound sterling.

`exchange.data` 975×2 matrix containing exchange rates of US dollar (first column) and Canadian dollar (second column), each relative to the pound sterling. Rates given are daily, and calendar date is contained in matrix row labels.

`fremantle.data` 86×3 matrix giving information about sea-levels at Fremantle, South Australia. First column gives year; second column gives annual maximum sea-level in meters; third column gives annual average of the Southern Oscillation Index.

`glass.data` vector of length 63 giving breaking strengths of glass fibers.

`portpirie.data` 65×2 matrix giving information about sea-levels at Port Pirie, southern Australia. First column gives year; second column gives annual maximum sea-level in meters.

`rain.data` vector of length 17531 containing daily rainfall accumulations in mm at a location in southwest England.

`venice.data` 51×11 matrix concerning sea-levels at Venice. First column gives year; remaining columns contain 10 largest observed sea-levels per year.

`wavesurge.data` 2894×2 matrix giving simultaneous measurements of oceanographic variables at a location off the southwest coast of England. First column gives wave height, second column gives surge height, each measured in meters and taken as typical over a 15 hour time window.

`wind.data` 40×3 matrix containing annual maximum wind speeds at Hartford and Albany in the U.S. First column gives year; second and third columns give annual maximum wind speed in mph at Hartford and Albany respectively.

2.2 Functions

The primary functions are as follows:

`gev.fit` Function for finding maximum likelihood estimators of the GEV model. Covariate models for each of the GEV parameters are allowed, together with link functions on any of the parameters.

`gev.diag` Takes as input the output of the function `gev.fit` and produces diagnostic plots to check the quality of model fit.

`gev.profxi` Plots the profile likelihood for ξ in the GEV model.

`gev.prof` Plots the profile likelihood for specified return level in the GEV model.

`gum.fit` Function for finding maximum likelihood estimators of the Gumbel distribution, corresponding to the special case of the GEV distribution with $\xi = 0$. Covariate models and link functions for either the location or scale parameter are enabled.

`gum.diag` Takes as input the output of the function `gum.fit` and produces diagnostic plots to check the quality of model fit.

`rlarg.fit` Function for finding maximum likelihood estimators of the r largest order statistic model. User can specify required number of order statistics to model. Covariate models and link functions are enabled.

`rlarg.diag` Takes as input the output from `rlarg.fit` and produces probability and quantile plots for each of the r largest order statistics.

`mrl.plot` Function for plotting mean residual life plots.

`gpd.fit` Function for finding maximum likelihood estimators of the generalized Pareto distribution at specified threshold. Covariate models and link functions are enabled.

`gpd.diag` Takes as input the output of the function `gpd.fit` and produces diagnostic plots to check quality of model fit.

`gpd.profxi` Plots the profile likelihood for ξ in the generalized Pareto model.

`gpd.prof` Plots the profile likelihood for specified return level in the generalized Pareto model.

`gpd.fitrange` Calculates and plots maximum likelihood estimates and confidence intervals for the generalized Pareto model over a range of thresholds.

`pp.fit` Finds maximum likelihood estimates and standard errors for the point process model. Covariate models and link functions are enabled for each of the model parameters, together with variable thresholds.

`pp.diag` Takes as input the output of the function `pp.fit` and produces diagnostic plots to check quality of model fit.

`pp.fitrange` Calculates and plots maximum likelihood estimates and confidence intervals for the point process model over a range of thresholds.

A detailed description of each of the various model types is given in the book.

3 Examples

We illustrate, by example, the use of each of the various model types.

3.1 Modeling block maxima

The data in `fremantle.data` consist of annual maximum sea-levels for a particular location, together with covariate information comprising year and the corresponding Southern Oscillation Index (SOI) value. Standard extreme value arguments suggest the GEV distribution is a suitable model for block maxima, but non-stationarity in the process suggests the distribution may vary from year-to-year. This is enabled by allowing the GEV parameters to be functions of the available covariates.

The primary function for estimating the GEV distribution is `gev.fit`. This takes the form

```
gev.fit <-  
  function(xdat, ydat = NULL, mul = NULL, sigl = NULL, shl = NULL,  
           mulink = identity, siglink = identity, shlink = identity)
```

To fit a time-homogeneous model it is necessary only to give the argument corresponding to the vector containing the block maximum data. For example, the Fremantle annual maximum sea level series is in the second column of `fremantle.data`. So, with the command

```
gev.fit(fremantle.data[,2])
```

we get the output

```
$conv:  
[1] T  
  
$nllh:  
[1] -43.56663  
  
$mle:  
[1] 1.4823418 0.1412723 -0.2174282  
  
$se:  
[1] 0.01672520 0.01149847 0.06378006
```

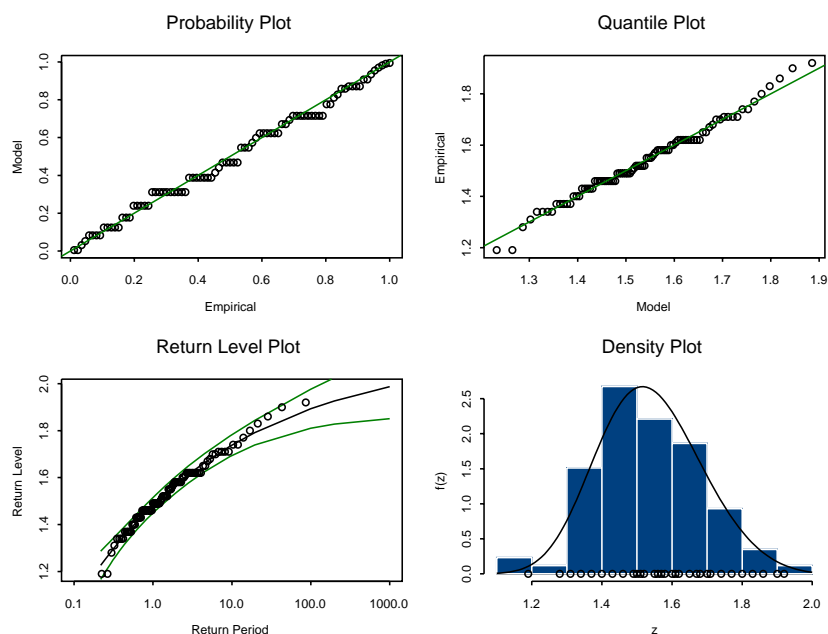


Figure 1: Diagnostic plots for GEV model fitted to Fremantle annual maximum sea-levels.

Respectively: `$conv$` is a true/false indicator — T implies the log-likelihood has been successfully maximized; F would imply failure of the numerical algorithm and that subsequent results are not trustworthy; `$nllh` is the minimized value of the negative log-likelihood; `$mle` contains the maximum likelihood estimates of μ , σ and ξ respectively; `$se` gives the standard errors of the maximum likelihood estimates, again in the order μ , σ and ξ .

To assess the quality of the fitted model the function `gev.diag` provides a suite of four graphical diagnostics. Using, for example,

```
fm.gev <- gev.fit(fremantle.data[,2])
```

stores the fitted object from the application of `gev.fit` in the object `fm.gev`. Then

```
gev.diag(fm.gev)
```

produces the suite of graphs shown in Figure 1.

Profile log-likelihoods for either the shape parameter ξ , or for a specified return level, can be produced using the functions `gev.profxi` and `gev.prof` respectively. For example, to produce the profile likelihood for the shape parameter ξ over the range -0.5 to 0.1 :

```
gev.profxi(fm.gev, -0.5, 0.1)
```

produces the output in Figure 2. Similarly, the profile likelihood for the 100-year return level of the interval 1.8 to 2.2 is obtained with

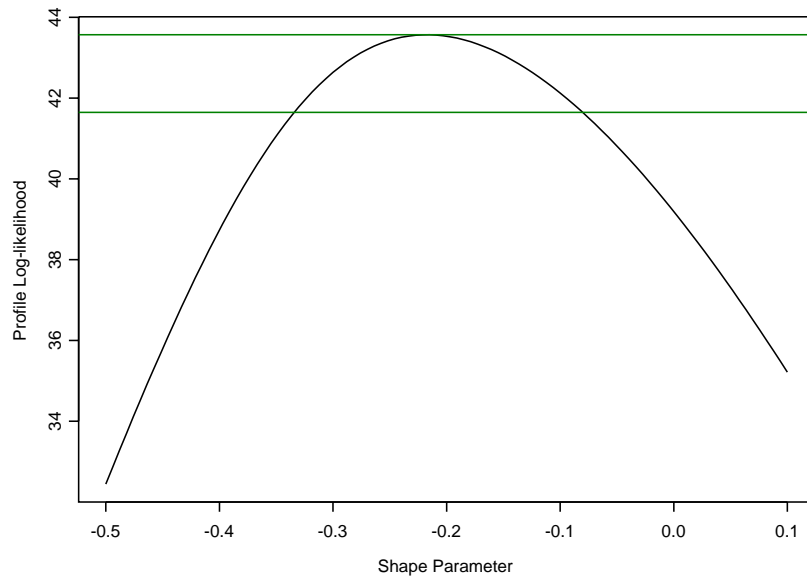


Figure 2: Profile log-likelihood of ξ in GEV model fitted to Fremantle annual maximum sea-levels.

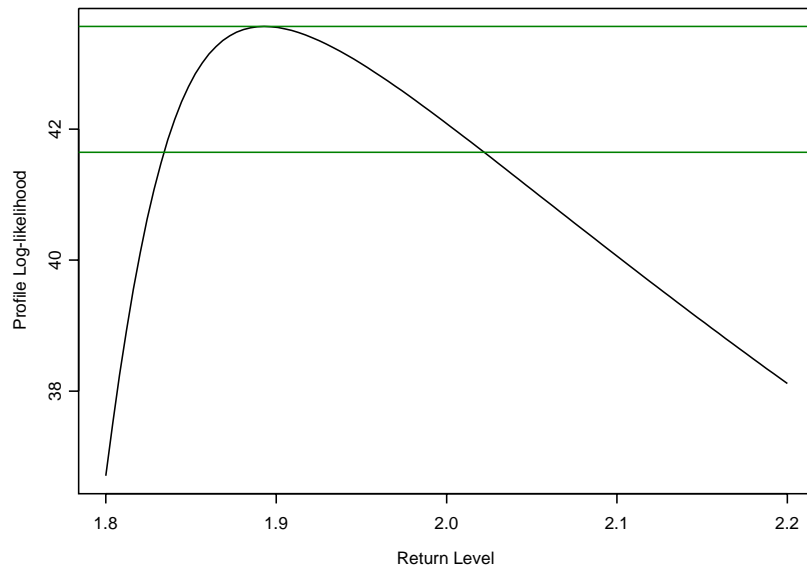


Figure 3: Profile log-likelihood of 100-year return level in GEV model fitted to Fremantle annual maximum sea-levels.

```
gev.profxi(fm.gev,100,1.8,2.2)
```

where the second argument is 100 corresponding to the required return period in years. This output is shown in Figure 3. Some care is needed when using the routines for plotting profile log-likelihoods as the robustness of the functions depends on making sensible choices for plotting intervals. If the output does not look ‘sensible’ it is likely that an inappropriate interval has been selected. A good strategy is to use intervals that are centered on the maximum likelihood estimates (which are obtained from `gev.fit` for ξ and `gev.diag` for return levels) and then use increasingly wider intervals until a satisfactory plot is obtained.

The full functionality of `gev.fit` enables the GEV parameters to be modeled in generalized linear form as functions of specified covariates. For example, the Fremantle annual maximum sea-levels are potentially varying through time, and also with the value of SOI. The years in the Fremantle data vary from 1897 to 1989. For numerical purposes, it is often preferable to re-scale the year variable so that it is centered and over a simple range. In this case, the transformation

$$\text{year}^* = (\text{year} - 1943)/46$$

creates an indexing of year on the scale $[-1, 1]$. The other possible covariate SOI varies from -1.78 to 2.12 and is therefore unlikely to benefit from any re-scaling. In S-PLUS we can therefore create a potential design matrix containing the two columns of available covariates:

```
cov <- cbind((fremantle.data[,1]-1943)/46,fremantle.data[,3])
```

Now, any or all of the GEV parameters can be modeled as (generalized) linear functions of any (or all) of the specified covariates. For example, to allow a linear time trend in the parameter μ , set `ydat=cov`, which specifies the potential covariate matrix as `cov`, and `mul=1`, which says that μ is to be modeled as a linear function of column 1 of `cov`. So,

```
gev.fit(fremantle.data[,2],ydat=cov,mul=1)
```

leads to

```
$model:
$model[[1]]:
[1] 1

$model[[2]]:
NULL

$model[[3]]:
NULL

$link:
[1] "c(identity, identity, identity)"

$conv:
[1] T
```

```

$nullh:
[1] -49.91281

$mle:
[1] 1.47570267 0.09348008 0.12432583 -0.12530836

$se:
[1] 0.01501726 0.02381444 0.01044765 0.06973575

```

The variables `$model` and `$link` contain a summary of the chosen model; the other output variables have similar format to the previous output. The sequence of terms in both `$mle` and `$se` is such that all of the parameters for μ are expressed first, then all of those for σ , then all of those for ξ . So, the model for μ is estimated as

$$\mu = 1.476 + 0.0935 \times \text{year}^*$$

while the estimates of σ and ξ are 0.124 and -0.125 respectively. The value of `$nullh` for this model enables a test of the validity of this model relative to the time-homogeneous model using standard properties of the likelihood function.

Models for the other parameters can be expressed similarly. For example, to model also σ as a function of year, it is normal to express the relationship using a link function such as

$$\sigma = \exp(\beta_0 + \beta_1 \times \text{year}^*),$$

which ensures that σ respects positivity for all choices of parameter values. This model is fitted to the Fremantle data as follows:

```
gev.fit(fremantle.data[,2], ydat=cov, mul=1, sigl=1, siglink=exp)
```

leading to

```

$model:
$model[[1]]:
[1] 1

$model[[2]]:
[1] 1

$model[[3]]:
NULL

$link:
[1] "c(identity, exp, identity)"

$conv:
[1] T

>nullh:
[1] -50.75242

$mle:

```

```
[1] 1.4772329 0.0853897 -2.0835669 -0.1635199 -0.1362345
```

```
$se:
```

```
[1] 0.01526941 0.02395112 0.08514507 0.12607053 0.07496995
```

In this case, the additional term `sig1=1` has been used to express the requirement for σ to be modeled linearly in terms of column 1 of `cov`, while the exponential inverse link function for this parameter has been expressed by `siglink=exp`. The output format is similar, so, for example, the estimated relationship for σ is

$$\sigma = \exp(-2.08 - 0.16 \times \text{year}^*)$$

although by comparison of log-likelihoods this model offers no significant improvement on the previous model.

To model, say, μ as a linear function of both year and SOI, we simply need to expand `mul` to include both columns of `cov`:

```
gev.fit(fremantle.data[,2], ydat=cov, mul=c(1,2))
```

leading to

```
$model:
```

```
$model[[1]]:
```

```
[1] 1 2
```

```
$model[[2]]:
```

```
NULL
```

```
$model[[3]]:
```

```
NULL
```

```
$link:
```

```
[1] "c(identity, identity, identity)"
```

```
$conv:
```

```
[1] T
```

```
$nllh:
```

```
[1] -53.89875
```

```
$mle:
```

```
[1] 1.48157143 0.09724321 0.05451785 0.12073275 -0.14998895
```

```
$se:
```

```
[1] 0.01460816 0.02387103 0.01963315 0.01012813 0.06666154
```

Hence, by this model,

$$\mu = 1.482 + 0.0972 \times \text{year}^* + 0.0545 \times \text{SOI}$$

Finally, storing this fitted object as

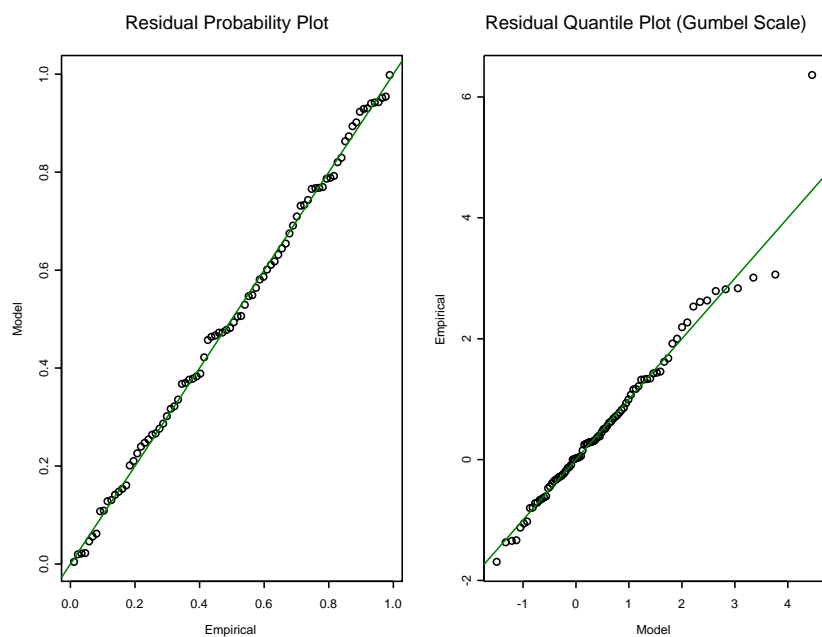


Figure 4: Diagnostic plots for GEV model with covariates fitted to Fremantle annual maximum sea-levels.

```
fm.gev2 <- gev.fit(fremantle.data[,2], ydat=cov, mul=c(1,2))
```

the quality of the fitted model can be assessed via

```
gev.diag(fm.gev2)
```

which produces the output in Figure 4. Since the model is non-stationary, only a reduced range of model diagnostics are produced, and these are presented on the scale of a Gumbel variable.

3.2 Modeling r -largest order statistics

The block maximum modeling approach extends to give an asymptotic model for the joint distribution of the largest r order statistics within a block, for fixed values of r . In particular, the model can be fitted to a series of largest annual order statistics. In S-PLUS this is achieved with the function `rlarg.fit`, which has the syntax

```
rlarg.fit <-
  function(xdat, r = dim(xdat)[2], ydat = NULL, mul = NULL,
           sigl = NULL, shl = NULL, mulink = identity,
           siglink = identity, shlink = identity)
\ batim}
```

The data for this model should be in matrix form, with rows corresponding to blocks and columns to the order statistic. So, for example, the first row consists of a vector comprising the first, second, third and so on order

statistic within the first block. By way of example, the matrix `venice.data` is a 51×11 matrix in which each row contains the 10 largest sea-levels recorded per year in Venice. The first column of the matrix gives the years of measurement. The function `rlarg.fit` is used in much the same way as `gev.fit`, with the extra requirement to specify as `r` the number of order statistics per block (year) to include in the analysis. For example, with the Venice data `r` can be chosen in the range $1, 2, \dots, 10$. If left unspecified, the function will adopt the number of columns of the data matrix `xdat` for `r`. If any of the blocks have fewer order statistics available, they should be entered in the data matrix as NA; `rlarg.fit` will handle the missing information appropriately. The matrix `venice.data` has only 6 order statistics available for the year 1935, for example.

The model is fitted as follows:

```
\begin{verbatim}
  rlarg.fit(venice.data[,2:11])
```

leading to

```
$conv:
[1] T

$nllh:
[1] 1149.268

$mle:
[1] 120.3961096 12.6929941 -0.1148273

$se:
[1] 1.34704941 0.52516209 0.01901014
```

Notice that `venice.data[,2:11]` is used as the data matrix since the first column of `venice.data` contains the years and not the data. The syntax of the output is identical to that of `gev.fit`, so, for example, the maximum likelihood estimates for μ , σ and ξ are 120.4, 12.7 and -0.115 respectively.

Diagnostics for this model are obtained by storing the fitted object,

```
venice.rl <- rlarg.fit(venice.data[,2:11])
```

for example, and then

```
rlarg.diag(venice.rl)
```

leading to two figures. The first is shown in Figure 5 and corresponds to the same suite of diagnostics for the annual maxima, but based on the analysis of the r largest order statistics. The second, shown in Figure 6, is a suite of probability and quantile plots for each individual order statistic.

Fitting the model to fewer of the available order statistics is achieved by specifying the argument `r`: for example,

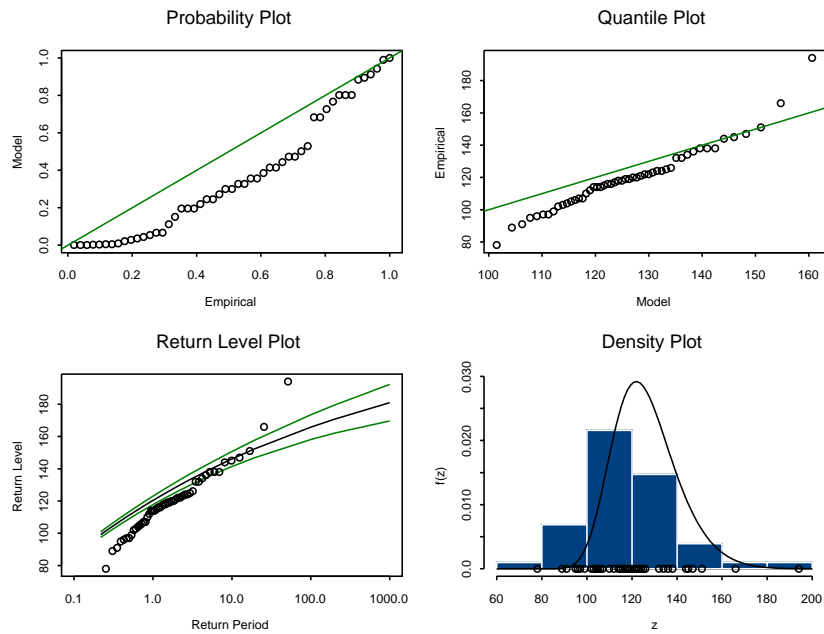


Figure 5: Diagnostic plots for annual maxima based on r largest order statistic model fitted to Venice sea-levels.

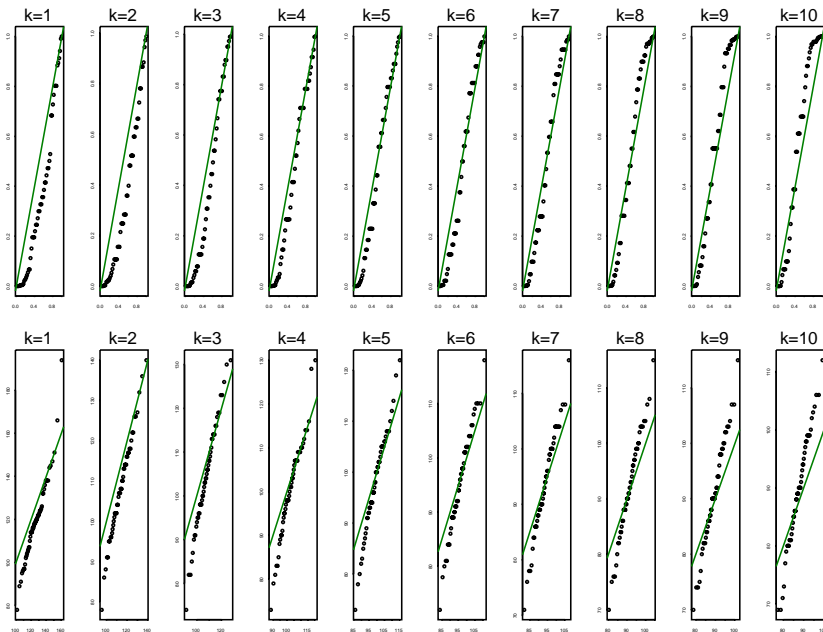


Figure 6: Probability plots (top row) and quantile plots (bottom row) for each order statistic in the r largest order statistic model fitted to Venice sea-levels.

```
rlarg.fit(venice.data[,2:11],r=5)
```

would fit the same model but using only the largest 5 order statistics for each year. Non-stationary models are fitted and interpreted in exactly the same way as for `gev.fit`. For example, with a scaled year covariate

```
cov <- matrix((venice.data[,1]-1956)/51,ncol=1)
```

the r largest order statistic model with a time-trend in the location parameter μ is fitted as

```
rlarg.fit(venice.data[,2:11],ydat=cov,mul=1)
```

generating the output

```
$model:
$model[[1]]:
[1] 1

$model[[2]]:
NULL

$model[[3]]:
NULL

$link:
[1] "c(identity, identity, identity)"

$conv:
[1] T

$nlh:
[1] 1093.888

$mle:
[1] 116.85914160 24.56817634 11.60507588 -0.06937427

$se:
[1] 1.25360770 2.08451745 0.60516413 0.02622634
```

The interpretation of the output, as well as the extendibility to other parameter forms, is the same as for the `gev.fit` function.

3.3 Modeling threshold excesses: the generalized Pareto model

The standard model for threshold excesses is the generalized Pareto model. We illustrate the procedure on the dataset `rain.data`, which comprises a vector of 17531 consecutive daily rainfall measurements in mm. The first step is threshold selection. A mean residual life plot is produced by

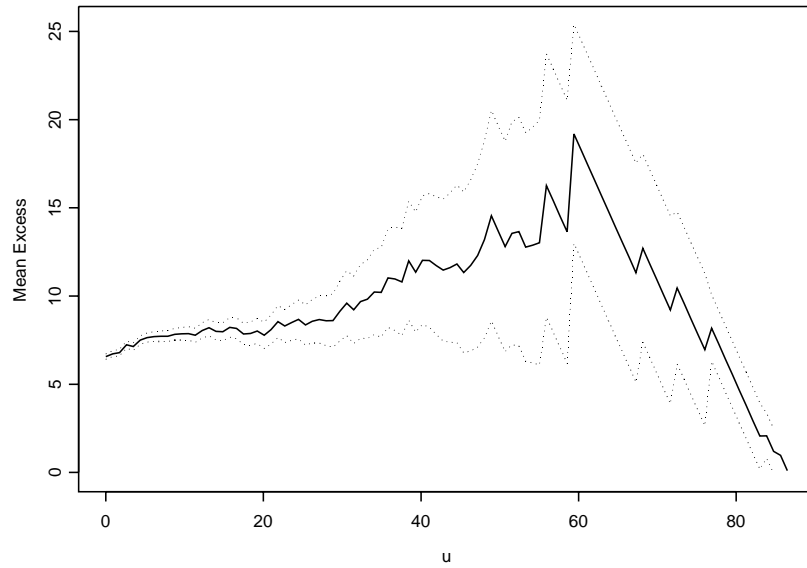


Figure 7: Mean residual life plot for daily rainfall data.

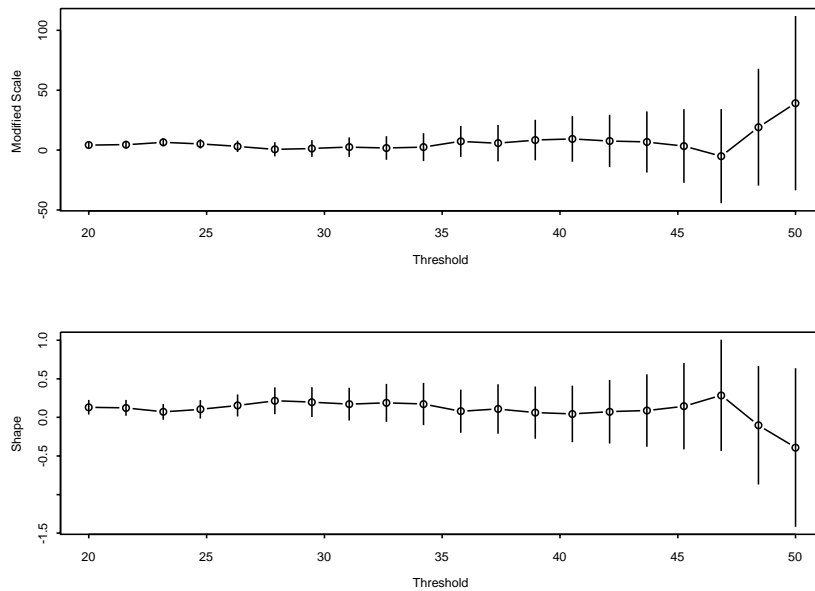


Figure 8: Maximum likelihood estimates and 95% confidence intervals for transformed parameters in generalized Pareto model applied to daily rainfall data.

```
mrl.plot(rain.data)
```

leading to Figure 7. A more detailed diagnostic is obtained by fitting the generalized Pareto model over a range of thresholds and looking for stability in appropriate parameters. For the rainfall data

```
gpd.fitrange(rain.data,20,50,nint=20)
```

produces a plot of such values with 20 even spaced thresholds (specified by `nint`) over the interval $[20, 50]$. This plot is shown in Figure 8.

Model fitting for a specified threshold uses `gpd.fit`, which again works like `gev.fit`. In its simplest form, to fit a time-constant model at a threshold of 30 mm to the daily rainfall data, the command

```
gpd.fit(rain.data,30)
```

leads to

```
$threshold:
[1] 30

$nexc:
[1] 152

$conv:
[1] T

$nllh:
[1] 485.0937

$mle:
[1] 7.4402657 0.1844994

$rate:
[1] 0.008670355

$se:
[1] 0.9585315 0.1012040
```

In this case the maximum likelihood estimates and their parameters correspond to σ and ξ respectively, the two parameters of the generalized Pareto distribution. Similarly for the standard errors. Additional output arguments for this model are `$threshold`, the specified threshold value; `$nexc`, the number of observations exceeding the threshold; and `$rate`, the proportion of the data that exceed the threshold. The assumption when fitting the generalized Pareto model is that data are *daily*. If this is not the case it is necessary to specify the argument `npy` which corresponds to the number of observations per year. This does not affect the parameter estimates, but does affect subsequent calculations such as return levels which are usually expressed on an annual scale. The default setting for `npy` is 365.

Diagnostics are produced with the function `gpd.diag`. So, for example,

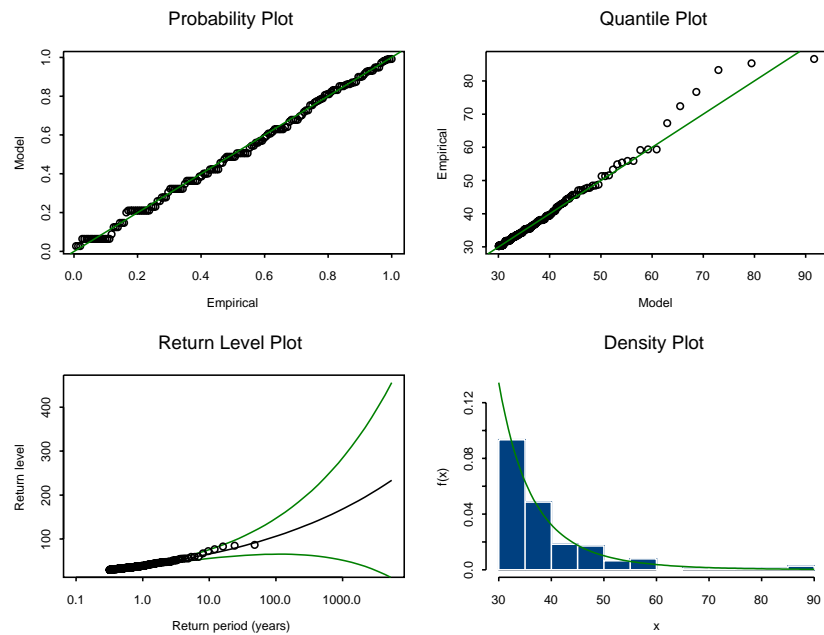


Figure 9: Diagnostics for generalized Pareto model fitted to daily rainfall data.

```
rain.gpd <- gpd.fit(rain.data,30)
```

and

```
gpd.diag(rain.gpd)
```

produces Figure 9. Profile log-likelihoods can also be obtained following the same syntax as for `gev.profxi` and `gev.prof`:

```
gpd.profxi(rain.gpd,-.1,.5)
```

and

```
gpd.prof(rain.gpd,100,70,400)
```

produce plots of the log-profile likelihood for ξ and the 100-year return level over the intervals $[-.1, .5]$ and $[70, 400]$ respectively. These plots are shown in Figures 10 and 11. Notice in particular that the profile likelihood plot for return levels is expressed on an annual scale, which is why it is essential to correctly specify the number of observations per year when fitting the model.

The full syntax for the `gpd.fit` function is

```
gpd.fit <-
```

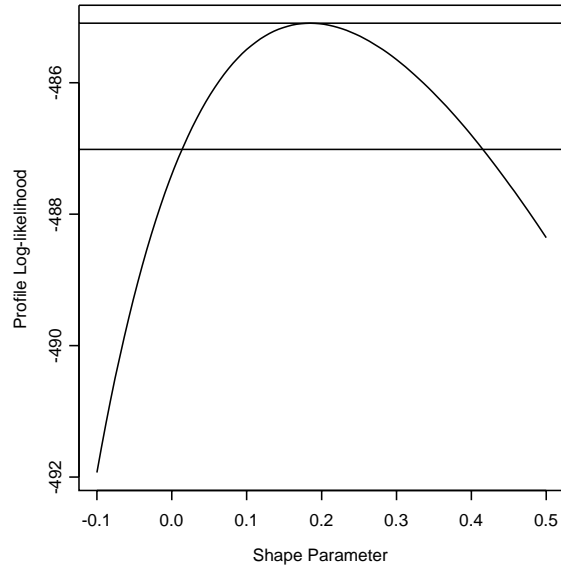


Figure 10: Profile likelihood for ξ in generalized Pareto model fitted to daily rainfall data.

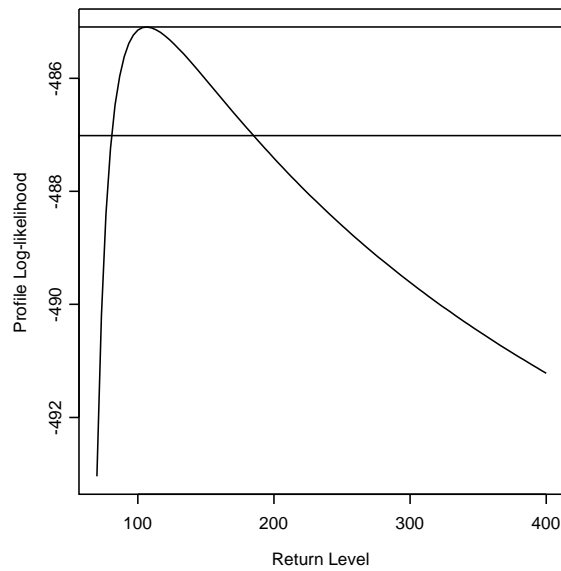


Figure 11: Profile likelihood for 100-year return level in generalized Pareto model fitted to daily rainfall data.

```
function(xdat, ufun, npy = 365, ydat = NULL, sigl
        = NULL, shl = NULL, siglink = identity,
        shlink = identity)
```

It is therefore possible to include covariate terms in a (generalized) linear form for either of the parameters σ or ξ in the same manner as for `gev.fit`. Moreover, it is possible to allow the threshold to be time-varying. This is achieved by specifying the argument `ufun` to be something other than a constant. There are two options:

1. Set `ufun` to be a function which operates on the vector `1:length(xdat)` to calculate a threshold value at each point of time;
2. Set `ufun` to be a vector of length `length(xdat)` so that the j th term in the vector corresponds to the required threshold for the j th observation of `xdat`.

Care must be exercised in using a time-varying threshold with the generalized Pareto model because of the dependence of the interpretation of the parameter σ with threshold. For that reason it is preferable to use the point process model and likelihood for dealing with problems of this type.

3.4 Modeling threshold excesses: the point process model

A more flexible model for threshold excesses is based on the point process representation for extremes. The point process model that corresponds to the generalized Pareto analysis of the daily rainfall data is estimated with

```
pp.fit(rain.data,30)
```

leading to

```
$threshold:
[1] 30

$npy:
[1] 365

$nexc:
[1] 152

$conv:
[1] T

$nllh:
[1] 461.9818

$mle:
[1] 39.5506470 9.2023509 0.1844992

$se:
[1] 1.2022478 0.9261280 0.1012041
```

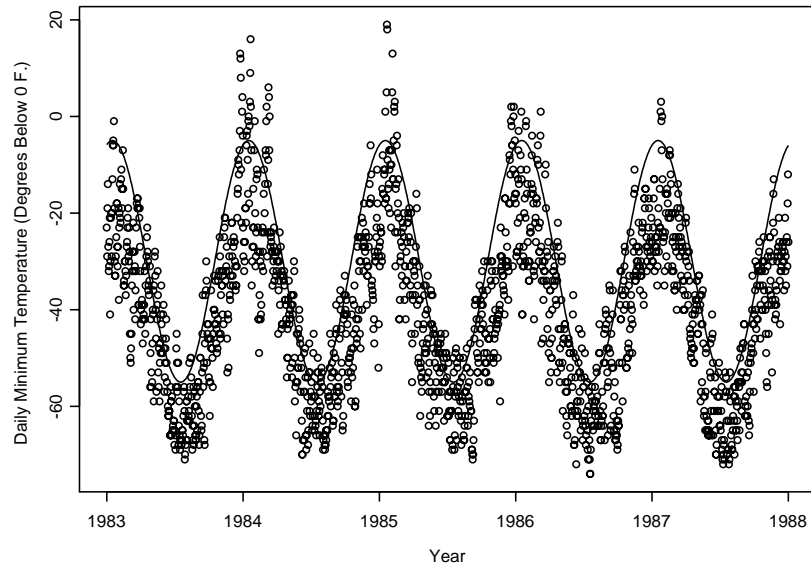


Figure 12: Negated daily minimum temperatures at Wooster, Ohio, with proposed time-varying threshold.

Again, a default of 365 is assumed for the number of observations per year, which can be overridden by specifying the `npv` argument in the function call. The output has a similar format to that of previous fitting functions, but now the parameter estimates correspond to μ , σ and ξ , which are the parameters of the associated annual maximum GEV distribution. Observe that the shape parameter estimate is identical to that of the corresponding generalized Pareto analysis. The scale parameter of the generalized Pareto model is a deterministic function of μ and σ , as is the estimated threshold exceedance rate. Output from `pp.fit` can be taken directly into `pp.diag` to obtain diagnostic plots that are, in fact, identical to those obtained with `gpd.diag`. The function `pp.fitrange` also has a similar utility and syntax to `gpd.fitrange`.

The advantages of the point process representation are realized when modeling non-stationary threshold exceedances. The full syntax of `pp.fit` is

```
pp.fit <-
  function(xdat, ufun, npv = 365, ydat = NULL, mul = NULL,
          sigl = NULL, shl = NULL, mulink = identity,
          siglink = identity, shlink = identity)
```

Its full functionality is therefore similar to that of earlier fitting routines. By way of example, Figure 12 shows a time series of negated temperatures with a suggested time-varying threshold. The temperatures themselves are in the vector `wooster.data`. For reference, the threshold was chosen to be periodic with period one-year, with parameter values selected by trial-and-error to give a reasonably constant crossing rate over the entire observation period. Explicitly:

```
x <- 1:length(wooster.data)
usin <- function(x, a, b, d) <- a + b * sin(((x - d) * 2 * pi)/365.25)
wu <- usin(x,-30,25,-75)
```

leads to `wu` containing the vector of threshold values shown in Figure 12. Various models can be considered for these data; a particularly suitable model comprises periodic effects in both μ and σ , but a constant value of ξ . The model matrix in this case can be constructed as

```
x <- 1:length(wooster.data)
ydat <- cbind(sin(x*2*pi/365.25),cos(x*2*pi/365.25))
```

and then the model fitted as

```
wooster.pp <- pp.fit(-wooster.data,u=wu,ydat=ydat,mul=1:2,sigl=1:2,siglink=exp)
```

yielding

```
$model:
$model[[1]]:
[1] 1 2

$model[[2]]:
[1] 1 2

$model[[3]]:
NULL

$link:
[1] "c(identity, exp, identity)"

$npv:
[1] 365

$nexc:
[1] 276

$conv:
[1] T

$nullh:
[1] -143.5577

$mle:
[1] -15.1012405  9.4738025  29.2103623  0.4989012  0.1878006  0.3584862 -0.3457587

$se:
[1] 0.60603357 0.82720781 0.75411618 0.17161935 0.06926877 0.06035249 0.06099985
```

Notice that an exponential inverse link function on σ has been specified again to preserve positivity on that parameter. The sequence of the parameter outputs is in the order μ, σ and ξ . So, for example, the estimated model for μ is

$$\mu(t) = -15.1 + 9.47 \times \sin(2\pi t/365.25) + 29.21 \times \cos(2\pi t/365.25)$$

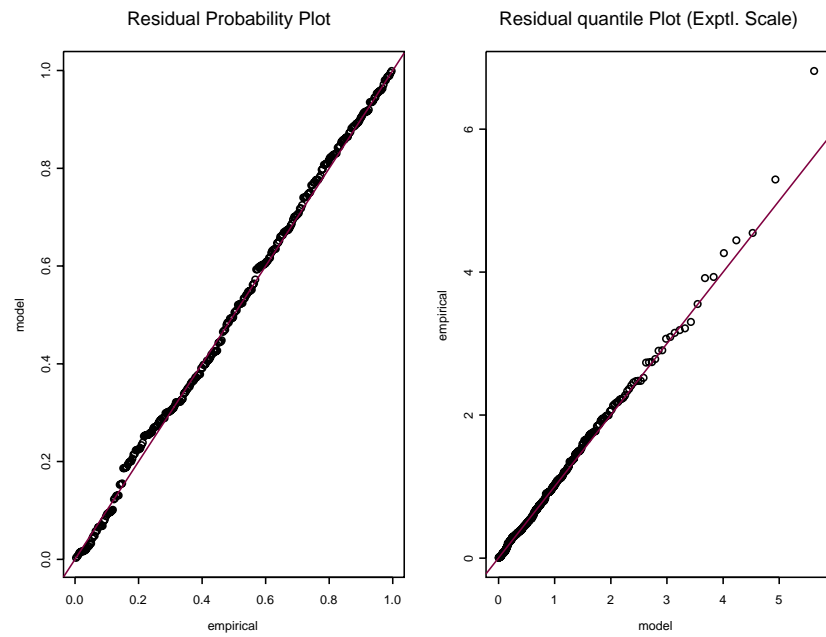


Figure 13: Diagnostics for periodic point process model fitted to negated Wooster daily minimum temperatures.

Finally, diagnostics for the fitted model can be obtained as

```
pp.diag(wooster.pp)
```

and shown in Figure 13. Again, because of the non-stationarity in the model, these are calculated with reference to a common scale — in this case, the exponential distribution.