3 R session: Weather extremes

To get started, you will need to be seated at a computer with R installed, and initiate R, which is usually done through menus selected from the Start menu, or an icon. In addition the libraries ismev and evd should be installed. We will connect these, and install some of our own supplementary routines, using the commands

```
> library(ismev)
> library(evd)
> source('Rstufflee.r')
```

Provided these all go through without a hitch, we are ready to go!

1. In this question, we will do a simple analysis of annual maximum wind speeds recorded at Boston, Massachusetts, for 50 years from 1936 to 1985.

(a) Provided you have the file boston.txt in your working directory, this can be loaded into Rusing the command:

```
> boston<-scan('boston.txt')
```

We have now created an R object called boston which is a single column containing consecutive years with annual maximum wind speeds in mph. We can have a look at this by simply typing:

```
> boston
```

(b) We now wish to separate out the year and maximum components into separate vectors. This can be done using the commands:

```
bosyear<-as.numeric(boston[seq(1,length(boston),2)])
bosmax<-as.numeric(boston[seq(2,length(boston),2)])
```

which has the effect of creating vectors bosyear and bosmax containing the years and maxima respectively. [Note that when entering consecutive similar commands in R, it is convenient to use the up arrow to bring up the previous command and then edit it!] We can check the vectors by simply typing:

```
> bosyear
> bosmax
```

(c) Now we can have a look at the annual maxima over time using the command:

```
> plot(bosyear,bosmax)
```

If you like you can give your plot some nice labels:

```
> plot(bosyear,bosmax,xlab='Year',ylab='Wind speed (mph)',
    main='Annual maximum wind speeds at Boston MA')
```

(d) We are now ready to carry out an extreme value analysis on the annual maxima. Since these are regarded as observations on i.i.d. random variables, we can forget about the vector `bosyear`. We fit the GEV to the data in `bosmax`:

```
>gev.fit(bosmax)
```

Notice the output:

- `$conv` gives a value of zero (in row [1] of the output), which indicates successful convergence, i.e. no errors in fitting;
- `$nllh` shows the negative (maximised) log–likelihood;
- `$mle` shows the maximum likelihood estimates for μ, σ and ξ respectively;
- `$se` gives the associated standard errors for these parameters.

(e) We can investigate the model performance using the in–built diagnostics. First we must store the relevant information from the fit in an object we name ourselves, e.g.

```
> fit1<-gev.fit(bosmax)> gev.diag(fit1)
```

creates the 'fit' object `fit1` and then runs the diagnostic routines on the stored object. Make sure you interpret the four plots in the context of Section 1.2.6.

(f) We can obtain inference on return levels using the additional command which we have supplied in `Rstufflee.R`, which is called `gev.ret(data,period)`. This command refits the GEV model, and then provides us with the inference on the specified return level. E.g. for the 100–year level q_{100}, we would type:

```
> gev.ret(bosmax,100)
```

In addition to the information we obtained earlier, we get the 100–year return level estimate with associated standard error. Notice how this matches up with the return level plot in the diagnostic plots.

(g) If we want to construct a confidence interval for q_{100}, we are better off using the method of profile–likelihood as described in Section 1.2.8. We can use the function `gev.prof(fit,period,lower-bound, upper-bound)`. This command is slightly unstable, and relies on an appropriate choice of the bounds for the profile–likelihood. For the Boston annual maxima, the following works well for the 100–year level:

```
> gev.prof(fit1,100,75,130)
```

Note that this enables us to read off the 95% confidence interval (the default) for q_{100}. Suppose we wanted a 99% interval we would use:

```
> gev.prof(fit1,100,<lower>,<upper>,conf=0.99)
```

for appropriate choices of `<lower>` and `<upper>`. You may like to experiment. Note how asymmetrical these intervals are, and how misleading it would be to base the confidence intervals on $\pm 1.96(\text{s.e.})$!

2. In this question, we will analyse annual maximum sea levels (in cm) observed at Venice, Italy, between the years 1931 and 1981 (inclusive).

(a) Load the data into R by typing:

```
> data(venice)
```

Now look at the data by typing

```
> venice
```

You should see a matrix with 51 rows (one for each of the years 1931–1981) and 11 columns. The values in each column correspond to the year, and the *ten* largest sea levels observed in each of these years (in descending order) For example, in 1979, the ten largest sea levels were: 166, 140, 131, 130, 122, 118, 116, 115, 115, 112, the largest being 166cm.

(b) We intend to fit the Generalised Extreme Value distribution to the set of annual maxima – i.e. the largest sea levels only (166cm in 1979, for example). Extract the set of annual maxima in the following way:

(i) Create a new vector to store the set of annual sea level maxima by typing:

```
> maxima<-vector('numeric', length=51)
```

(ii) Now type:

```
> maxima<-venice[,2]
```

which will store the observations from column 2 in `venice` – i.e. the largest sea levels from each year – in the vector `maxima`.

We can fit the Generalised Extreme Value distribution to the set of annual maxima using the function `gev.fit`. Type

```
> gev.fit(maxima)
```

Write down the maximum likelihood estimates of μ, σ and ξ, along with their estimated standard errors. Also make a note of the value of the maximised log–likelihood.

(c) Now produce a time series plot of the set of annual maxima by typing

```
> plot(maxima~venice[,1],type='l',xlab='Year',ylab='Sea level
(cm)')
```

which will plot the annual maxima against the first column in `venice`, which corresponds to the year. This will also provide convenient labels for both the x and y axes in the plot. Does the time series plot of annual maxima look stationary?

(d) We will now attempt to model variations through time in the sequence of annual sea level maxima by modelling a linear trend in the location parameter μ, i.e. $\mu(t) =$

$\beta_0 + \beta_1(t)$, where t represents the time–point (so $t = 1$ corresponds to 1931, etc.) Set up a time matrix by typing:

```
> time<-matrix(1:51,ncol=1)
```

Now type

```
> gev.fit(maxima, ydat=time, mul=1)
```

which tells R to use the matrix `ydat` as a matrix of covariates, and `mul=1` tells R which column in that matrix to use (as well as which parameter to use it for – μ!). Write down the maximum likelihood estimates for β_0, β_1, σ and ξ, along with their estimated standard errors, and make a note of the maximised log–likelihood.

(e) Use the maximised log–likelihood values from parts (b) and (d) to perform a likelihood ratio test to see if the model which allows for a trend provides a significant improvement over the stationary fit (Hint: $\chi^2_1(5\%) = 3.84$).

(f) Write down the simple linear regression equation for μ found from the fit in part (d), i.e. $\mu(t) = \beta_0 + \beta_1(t)$. We will now write an R function to calculate the fitted trend at each time point, and then superimpose this on the plot produced in part (c). Type

```
> trend.plot<-vector('numeric',51)
```

The vector `trend.plot` will take the fitted values of the trend for μ obtained from the equation. Now write

```
> for(i in 1:51)
+ {
+ trend.plot[i]<-beta0+beta1*time[i,1]
+ }
```

where `beta0` and `beta1` should be replaced with the estimated values found in the fit in part (d). Now type

```
> lines(trend.plot~venice[,1])
```

which should superimpose a plot of the trend line against the year on the original time series plot.

3. In this question we will investigate the use of "Peaks Over Threshold" to circumvent the problems of serial dependence when modelling threshold exceedances. We will do this by examining hourly gust maximum wind speeds observed at High Bradfield, a location in the Peak District in central northern England.

(a) These data were collected by the U.K. Meteorological Office, and are not included with any of the standard R packages. Thus, to load the data, type

```
> gusts<-scan('bradfield.txt')
```

which will store the data in a vector called `gusts`. Now produce a time series plot of these data, by typing

```
> plot(ts(gusts))
```

The data you see correspond to the hourly gust maximum wind speeds (in knots) collected over a ten–year period (1975–1984 inclusive) in the month of January; thus, the first observation is the maximum gust wind speed observed between midnight and 01:00 on the 1st January 1975, etc. We restrict our analysis to January because the U.K. has a seasonally varying wind climate, and the strongest wind speeds are usually observed in the month of January (i.e. in January we observe 'genuine' extremes of wind speed). Comment on the nature of this time series.

(b) We will now investigate the extent of temporal dependence in the series.

(i) Type

`> acf(gusts)` and
`> pacf(gusts)`

These commands will produce plots of the autocorrelation, and *partial* autocorrelation function.

(ii) Now type

```
> plot(gusts[1:7259]~gusts[2:7260])
```

This will produce a plot of the time series against the series at lag 1 (the length of this dataset is 7260).

Using your plots in (i) and (ii) above, comment on the degree of short–term temporal dependence present in the series.

(c) We now intend to fit the Generalised Pareto Distribution (GPD) to a set of threshold exceedances. Use the command

```
> mrl.plot(gusts)
```

to produce a mean residual life plot for the gust data, and use this to choose an appropriate threshold for identifying extremes.

(d) Now fit the GPD to the set of threshold exceedances, by using

```
> gpd.fit(gusts,threshold)
```

where `threshold` is your chosen threshold from the mean residual life plot in part (c). Make a note of the estimates for σ and ξ (as well as their estimated standard errors).

(e) Now *decluster* the series of gusts and employ a *Peaks Over Threshold* analysis. Type

```
> cluster.peaks<-cluster10(gusts,threshold)
```

again, where `threshold` is the threshold identified in part (c). The function `cluster10` uses a value of $\kappa = 10$ observations to identify clusters of extremes, i.e. a cluster of extremes is deemed to have terminated as soon as at least 10 observations fall below the threshold. Now fit the GPD to the set of cluster peak excesses, and make a note of the parameter estimates and estimated standard errors.*[Note: you can vary the declustering interval κ by using different functions, e.g.* `cluster20` *or* `cluster30`*]*

(f) We will now calculate the *threshold exceedance rate* for each of the approaches in parts (d) and (e). Typing

`> length(gusts[gusts>threshold])/length(gusts)` and
`> length(cluster.peaks)/length(gusts)`

where `threshold` is as before, will work out the threshold exceedance rate λ_u for *all* excesses, and *cluster peak* excesses, respectively. Write down these threshold exceedance rates.

(g) You should now compare estimates of the 1000–observation return level using (i) all threshold excesses and (ii) cluster peak excesses. Typing

```
> gpd.ret(data,threshold,1000)
```

but replacing `data` with `gusts` and then `cluster.peaks` (and the `threshold` is that identified in part (c)) will estimate this value for *all* excesses and *cluster peak* excesses, respectively. The output produced will be the same as before – i.e. you will get estimates of the GPD parameters and their standard errors, but now you will also get an estimate of the specified return level (and its standard error via the delta method).

(h) Comment on your estimates of the 1000–observation return level in part (g) and your GPD parameter estimates in parts (d) and (e). Which approach to inference do you trust most?