A practical introduction to modelling extreme values with R

Harry southworth

Data Clarity Consulting Ltd

harry@dataclarityconsulting.co.uk

2016-02-04

Getting the data

http://www.metoffice.gov.uk/public/weather/climate-historic

To get the data

- Click on a (red) station then on "Historic station data" in the pop-up
- Ctrl+A to highlight all, then Ctrl+C to copy
- Then Ctrl+V in your favourite text editor

Manually delete any text from the top of the file, reduce the column names to a single row.

Scroll to the bottom and delete all instances of "Provisional" (either the word or the whole row for our purposes).

(In real life, I'd use a Python script to automate the process.)

(Comment on getting the data)

Some values are estimated and have a '*' next to them. These get turned into missing values when we coerce to numeric. Do a global search and replace to remove them.

In real life, we might want to do this analysis repeatedly:

- Estimated and provisional values might get updated
- More data is added every month

If we wanted to update the analysis monthly, manually getting the data would be slow, error-prone and annoying. Using a Python (or other) script to do it would:

- Be more reliable (less error-prone)
- Be faster
- Quickly pay back on the time invested.

(Hint: you need to learn more than one computer language. Sorry, but it wasn't me that changed the world.)

Why R?

R is open source, 'free', very widely used by statisticians

- Very good graphics (especially ggplot2 package)
- Good data manipulation (dplyr and tidyr packages)
- Many packages available
- Good evm software (ahem, texmex package)

Alternatives (but I don't know what evm functionality they have):

- Julia, Python NumPy
- Matlab, SAS

Why texmex?

Because I'm a major author of it and

- Enables MLE, maximum penalized likelihood, MCMC, parametric bootstrap
- Large test suite gives confidence in results
- Easy modelling with covariates

Alternatives (which might not include all that functionality)

- VGAM
- ismev, evd, extRemes

Reading the data into R and plotting

```
dataPath <- "your/path/to/the/data/"
dataPath <- "data/"
d <- read.table(file.path(dataPath, "newtonRigg.txt"), header=TRUE)

# Coerce all fields to numeric
d <- apply(d, 2, as.numeric)

# Convert to data.frame
d <- as.data.frame(d)

# Remove rows with missing rainfall
d <- d[!is.na(d$rain), ]

# Create dates (just for nice plotting and exploring)
d$month <- formatC(d$mm, width=2, flag="0")
d$date <- as.Date(paste(d$yyyy, d$month, "15", sep="-"))

# Plot the data
plot(d$date, d$rain, xlab="Date", ylab="Monthly rainfall (mm)", col="blue")
```

Mean residual life and parameter stability plots

```
library(texmex)

?gpdRangeFit
?mrl

par(mfrow=c(1, 2))
rainStab <- gpdRangeFit(d$rain)
plot(rainStab)

rainMRL <- mrl(d$rain)
plot(rainMRL)
```

Simple GP model

```
?evm

g0 <- evm(rain, data=d, th=50)
g0
summary(g0)

par(mfrow=c(2, 2))
plot(g0)

# If happy, refit using MCMC
b0 <- evm(rain, data=d, th=50, method="sim", iter=100000, thin=10)

par(mfrow=c(3, 2))
plot(b0)
```

Return levels

```
class(b0)
?predict.evmSim

p <- predict(b0, M=12*c(1:1000), ci.fit=TRUE)
plot(p)
abline(h=max(d$rain))
```

Comments

The `summary` function gives us t-ratios and the results of 1000 runs from the fitted model: check that most observations are within the simualted envelope

Notice the fitted value of ξ, its standard error and its posterior distribution: values $\xi < 0$ imply a finite upper endpoint for rainfall

We're assuming stationarity, which might not be correct

No account taken of seasonality

Suggest removing the last observation (December 2015) and refitting. How influential is the last observation?

Including a covariate

First, reduce to 'winter' months (chosen for our data-driven purposes)

```
w <- d[d$month %in% c(11:12, 1:2), ]
w0 <- evm(rain, data=w, th=50)
w1 <- evm(rain, data=w, th=50, phi = ~yyyy)
w2 <- evm(rain, data=w, th=50, xi = ~yyyy)

# w2 has fitted values of xi < -0.5 so abandon it

AIC(w0); AIC(w1)

summary(w1)

2*(logLik(w1) - logLik(w0))
```

Comments

Remember to plot the fitted model and refit if anything looks bad (use a higher threshold, for example)

The assumed relationship is linear. To include non-linear relationship use

```
library(splines)
evm(rain, data=w, th=50, xi = ~ns(yyyy, df=4)
```

... or use the gpd function in the VGAM package

Goes beyond what we can reasonably cover today

Return levels

```
bw2 <- evm(rain, data=w, th=50, phi=~yyyy, method="sim")

# Remember to plot it!

nd <- data.frame(yyyy=2016, mm=12)
predict(bw2, newdata=nd, M=100, ci=TRUE)
```

Comments

As discussed in the previous session, this is a bit simplistic. A climate is a complicated thing

The liver data

Background

- A clinical trial with approximately 160 patients in each of 4 dose groups
- Dose A $<$ Dose B $<$ Dose C $<$ Dose D
- Doses are linear on log scale, so replace by 1, 2, 3, 4
- Large values of ALT, AST suggest liver injury
- (Interpretation of TBL and ALP is a little contovertial)

The liver data are included in the texmex package:

```
head(liver)
str(liver)
summary(liver)
```

Variables ending with 'M' are on-treatment maxima. Those ending with 'B' are baseline values

Plot the data

```
liver$d.alt <- liver$ALT.M / liver$ALT.B
liver$ndose <- as.numeric(liver$dose)

boxplot(d.alt ~ ndose, data=liver, log="y")

# Remove medians then plot again
meds <- tapply(liver$d.alt, liver$dose, median)
i <- match(liver$dose, names(meds))
liver$a.alt <- c(log(liver$d.alt / meds[i]))

boxplot(a.alt ~ ndose, data=liver)
```

Threshold selection

```
rf <- gpdRangeFit(liver$a.alt)
m <- mrl(liver$a.alt)

par(mfrow=c(2, 2))
plot(rf)
plot(m)
```

Fit some models

```
m0 <- evm(a.alt, data=liver, th=0.1)
plot(m0)

m1 <- evm(a.alt, data=liver, th=0.1, phi=~ndose)
m2 <- evm(a.alt, data=liver, th=0.1, xi=~ndose)
m3 <- evm(a.alt, data=liver, th=0.1, phi=~ndose, xi=~ndose)

AIC(m0); AIC(m1); AIC(m2); AIC(m3)

plot(m3)

b3 <- evm(a.alt, data=liver, th=0.1, phi=~ndose, xi=~ndose,
          method="sim")
par(mfrow=c(3, 4))
plot(b3)
```

Return levels

```
# Create new data.frame and get the predicted return levels
nd <- data.frame(ndose = 1:4)
rl <- predict(b3, newdata=nd, M=c(100, 500, 1000), ci=TRUE,
              alpha=.1)

# Exponentiate, restructure, then reapply medians
rl <- lapply(rl, function(x) x[, 1:4] <- exp(x[, 1:4]))
rl <- as.data.frame(do.call("rbind", as.list(rl)))
rownames(rl) <- 1:nrow(rl)

rl$dose <- rep(LETTERS[1:4], length.out=nrow(rl))
rl[, 1:4] <- rl[, 1:4] + meds[match(rl[, 5], LETTERS[1:4])]

names(rl) <- c("mean", "median", "lo", "hi", "dose")
rl$M <- rep(c(100, 500, 1000), each=4)

rl
```