[Re] Local alignment statistics - Estimating mu & lambda with maximum likelihood

Nathan Brouwer

11/18/2019

Estimating mu and lambda - explanation

Statistical distributions are described by equations. For example, the normal distribution is described by two parameters, the mean and standard deviation. These equations can be complex-looking, but luckily R has all the math already set up within functions.

The Gumbel EVD is described by an equation with two parameters: mu and lambda. Mu is approximately the mode, but we want a precise value. There's no easy way to get an approximate value for lambda. To get these parameters we'll feed the results of our simulation to a sophisticated statistical algorithm the will figure out which values of lambda and mu best fit the data.

There are several ways to do the math underlying estimation of mu and lambda; Altschul and Gish use the **method of moments**; in R its easiest to use **maximum likelihood estimation** which is called **MLE** for short.

Estimation of mu and lambda is done with the function fevd(). The fevd() function will doing some complex math for us very quickly. Basically what its doing is figuring which parameters, if plugged in to a simulation, would result in data that is most similar to our observed data.

We can simulate random evd data like this (don't worry about specific.) "location" refers to mu, so we'll put our best estimate of mu, the mode. scale is equal to 1/lambda. I know lambda is often around 0.27, so I"ll put in values on either side of that

I'll simulate 3 sets of values, one set for each value of lambda. I'll simulate 1000 replications each.

First, the scale parameters I'll use

```
scale1 <- round(1/0.135, 4)
scale2 <- round(1/0.27,  4)
scale3 <- round(1/0.405, 4)
```

The location parameter I'll use is the mode from my previous basic analysis

```
mode.i <- 27
```

Now I'll run my simulations

```
# number of simulated values
ns <- 1000
library(extRemes)
```

```
## Loading required package: Lmoments
```

```
## Loading required package: distillery
```

```
##
## Attaching package: 'extRemes'
```

```
## The following objects are masked from 'package:stats':
##
##     qqnorm, qqplot
```

```
# simulate data with scale1
revd.scale1 <-  revd(n = ns, loc = mode.i, scale = scale1, type = "GEV")

# simulate data with scale2
revd.scale2 <-  revd(n = ns, loc = mode.i, scale = scale2, type = "GEV")

# simulate data with scale3
revd.scale3 <-  revd(n = ns, loc = mode.i, scale = scale3, type = "GEV")
```

Now I'll plot all 3 of these and compare to the original data. I'll make a 4 x 4 grid, with the original data in the upper left. This code is a bit dense, don't worry about details

First I'll set some stuff up for the plot. Don't worry about these details

```
all.data <- c(revd.scale1,revd.scale2,revd.scale3)
xlims <- c(min(all.data)-1, max(all.data)+1)
breakz <- seq(xlims[1],xlims[2], length.out = xlims[2]-xlims[1])
```

For reference, I'm going to plot the minimum and maximum of the original data on each plot.

First, load the data

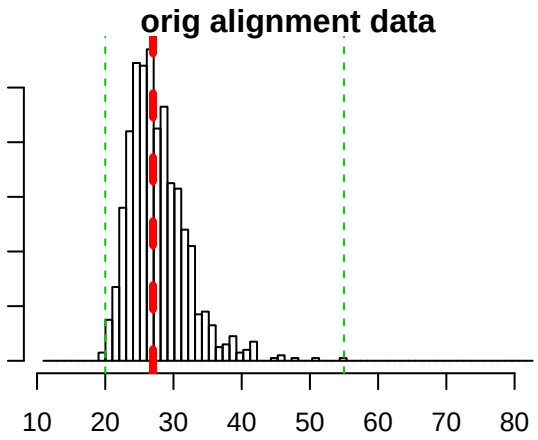
```
random.scores.191 <- read.csv(file = "random_scores_191.csv")
```

Then get the mins and maxes

```
max.i <- max(random.scores.191$score.i)
min.i <- min(random.scores.191$score.i)
```

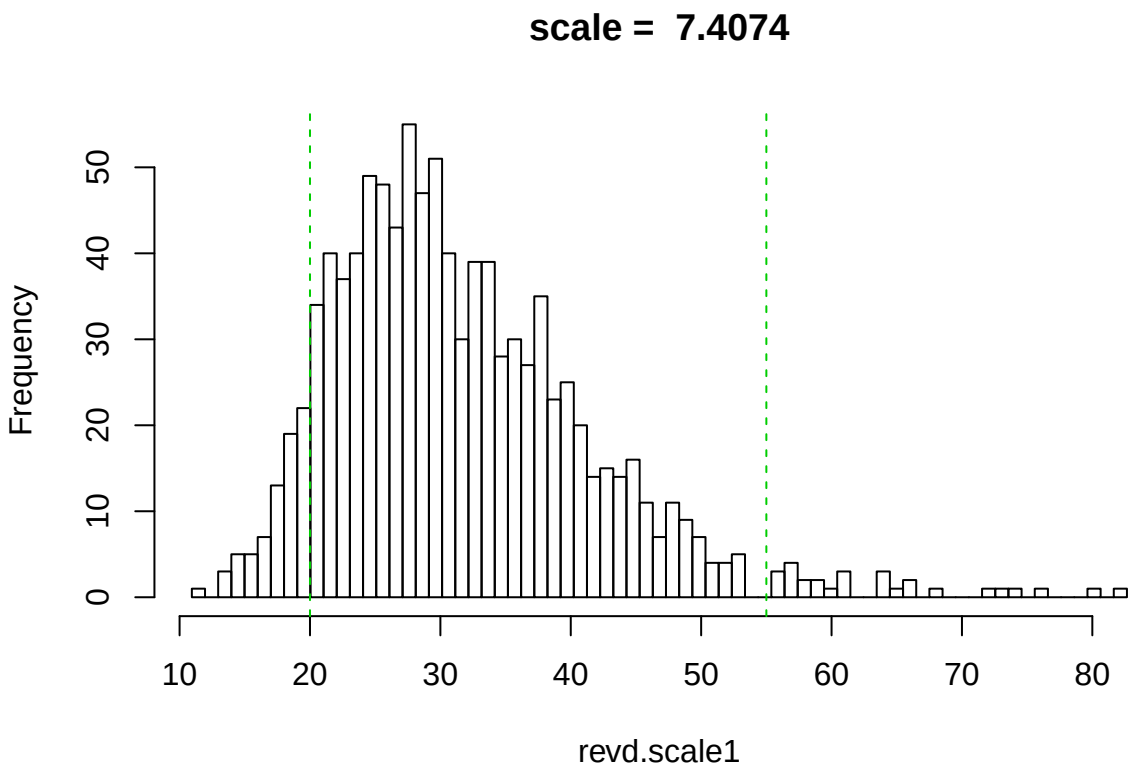
Now I'm going to plot the data from my simulation. This is the data from my for() loop in a previous script, which I reloaded The mode is in red, and min and max are in green. Again, this is data from actual alignment.s

```
par(mfrow = c(2,2), mar = c(2,1,1,2))
hist(random.scores.191$score.i,
     breaks = breakz,
     main = "orig alignment data",
     xlim = xlims)
abline(v = mode.i, col = 2, lwd = 4, lty = 2)
abline(v = max.i, col = 3, lwd = 1, lty = 2)
abline(v = min.i, col = 3, lwd = 1, lty = 2)
```



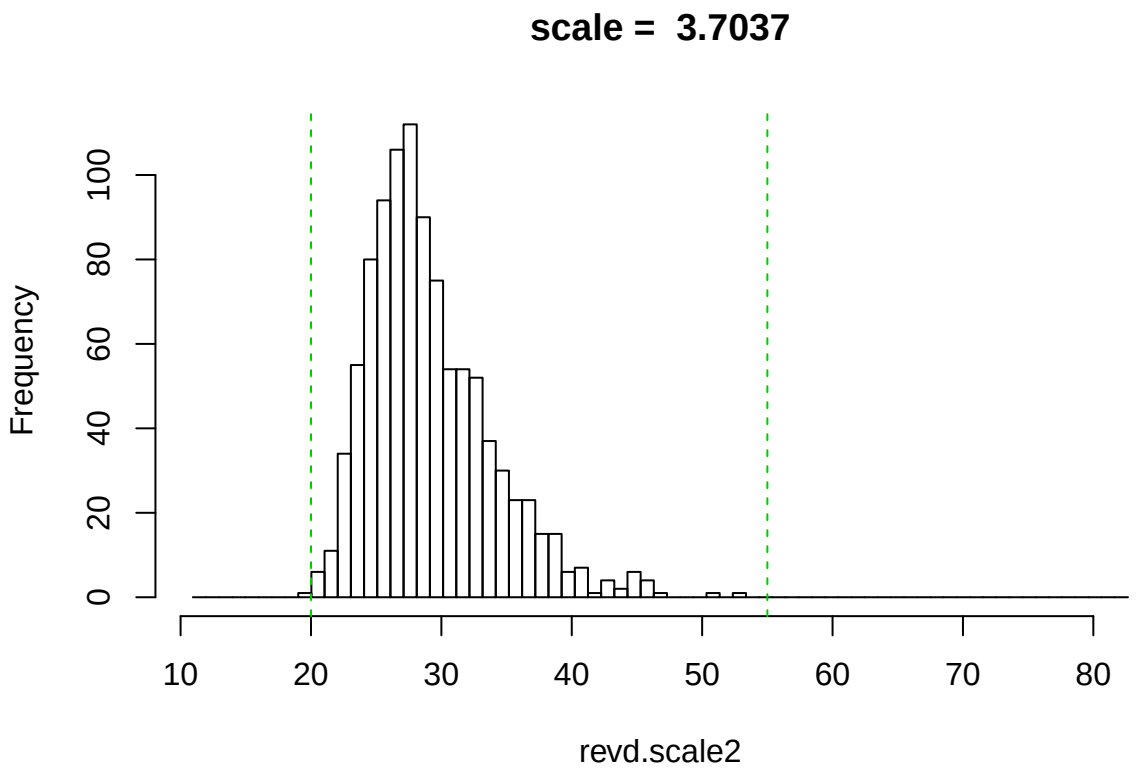
Now I'm going to plot the data for the first set of data simulated from the evd. No alignments were involved in this - this is just using the equation for an EVD via the revd() function ("random extreme value distribution") to simulate data for a given value of mu and lambda. Recall that for mu I used the mode of the original alignment data. The green lines are the min and max of the original alignment data. Note that the data simulated from an EVD goes much higher and much lower than the min and max of the alignment data.

```
hist(revd.scale1,
     breaks = breakz,
     main = paste("scale = ",scale1), xlim = xlims)
abline(v = max.i, col = 3, lwd = 1, lty = 2)
abline(v = min.i, col = 3, lwd = 1, lty = 2)
```



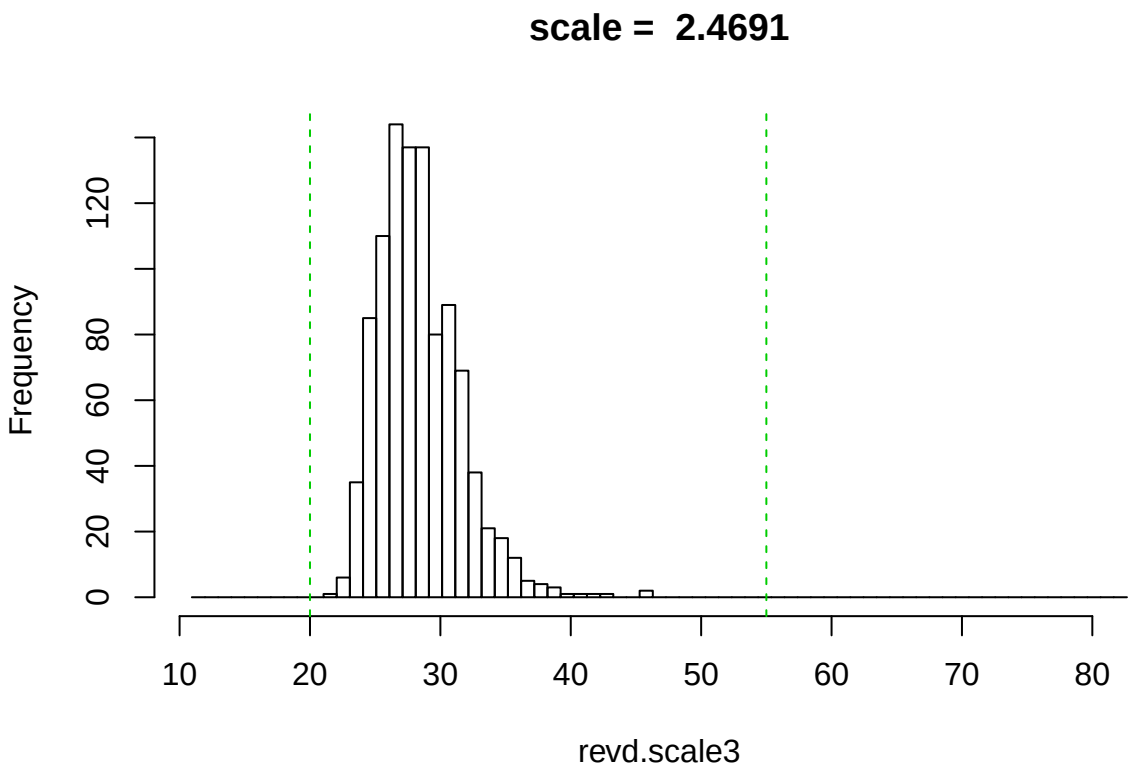
Now the second set of data simulated from the EVD. This is actually pretty close, with values going just a bit above and bit below the min and max.

```
hist(revd.scale2,
     breaks = breakz,
     main = paste("scale = ",scale2), xlim = xlims)
abline(v = max.i, col = 3, lwd = 1, lty = 2)
abline(v = min.i, col = 3, lwd = 1, lty = 2)
```



Finally the third set of simulated data. This isn't too bad for the minimum, but the value fall pretty short of the maximum.

```
hist(revd.scale3,
     breaks = breakz,
     main = paste("scale = ",scale3), xlim = xlims)
abline(v = max.i, col = 3, lwd = 1, lty = 2)
abline(v = min.i, col = 3, lwd = 1, lty = 2)
```



Estimating mu and lambda in R

Let's have R estimate mu and lambda for real for us. This is done easily like this:

```
fit.gumbel <- fevd(random.scores.191$score.i,
                   type = "Gumbel",
                   method = "MLE")
```

(For the brave, a good outline of the math is here:https://stats.stackexchange.com/questions/71197/usable-estimators-for-parameters-in-gumbel-distribution; this requires some familiarity with the concept of likelihoods).

The summary() function gets important results for the model

```
summary(fit.gumbel)
```

```
##
## fevd(x = random.scores.191$score.i, type = "Gumbel", method = "MLE")
##
## [1] "Estimation Method used: MLE"
##
##
##  Negative Log-Likelihood Value:  2771.199
##
##
##  Estimated parameters:
##  location     scale
```

```
## 26.188561  3.296625
##
##  Standard Error Estimates:
##   location      scale
## 0.10973480 0.08125105
##
##  Estimated parameter covariance matrix.
##             location       scale
## location 0.012041726 0.002783948
## scale    0.002783948 0.006601733
##
##  AIC = 5546.399
##
##  BIC = 5556.214
```

What we're looking for is under "Estimated parameters". mu = Location, and lambda = 1/scale If scale = 3.48, 1/3.48 = 0.2873563

We can save the summary output to an object

```
summary.gumbel <- summary(fit.gumbel)
```

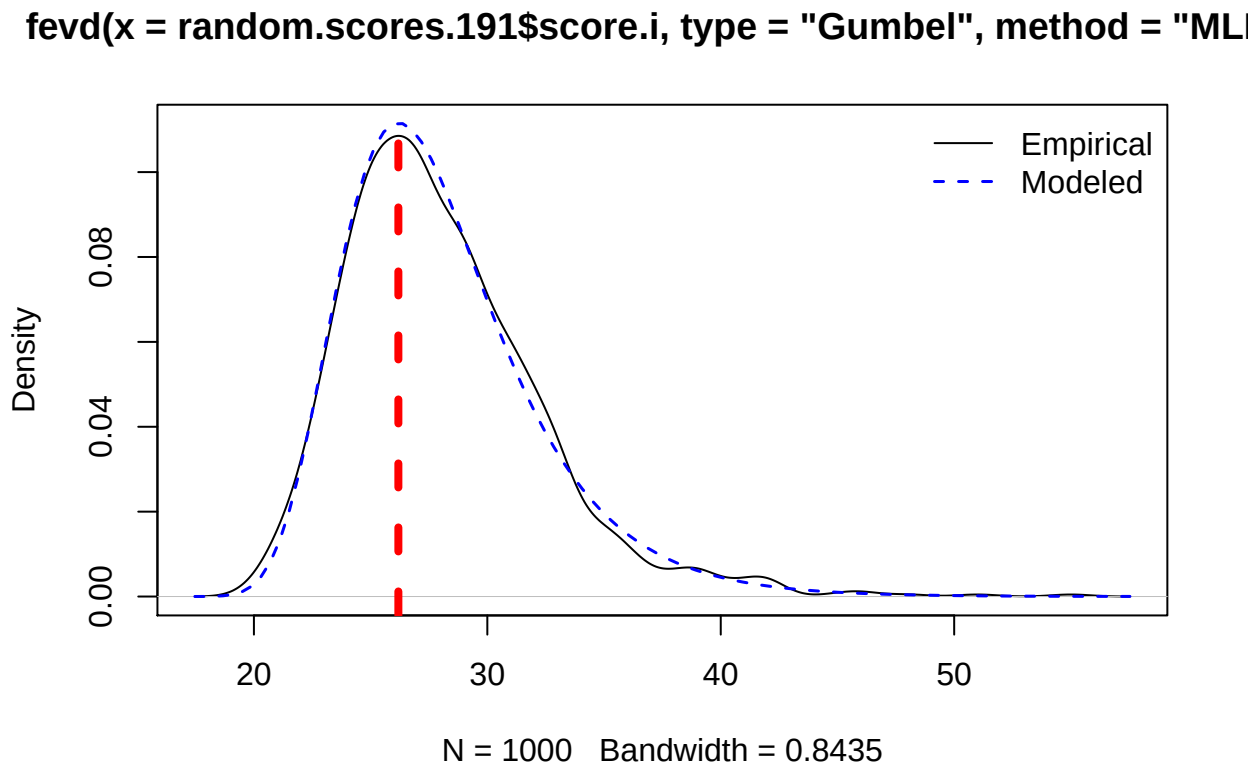
```
##
## fevd(x = random.scores.191$score.i, type = "Gumbel", method = "MLE")
##
## [1] "Estimation Method used: MLE"
##
##
##  Negative Log-Likelihood Value:  2771.199
##
##
##  Estimated parameters:
##  location     scale
## 26.188561  3.296625
##
##  Standard Error Estimates:
##   location      scale
## 0.10973480 0.08125105
##
##  Estimated parameter covariance matrix.
##             location       scale
## location 0.012041726 0.002783948
## scale    0.002783948 0.006601733
##
##  AIC = 5546.399
##
##  BIC = 5556.214
```

With some trial and error I figured out where location (mu) and scale (1/lambda) were stored and saved them to their own object.

```
mu.param  <- summary.gumbel$par[1]
scale.param <- summary.gumbel$par[2]
lambda.parm  <- 1/scale.param
```

I can plot the original data represented as a smooth curve (black) and compare it to the modeled data (blue dashed). The lines are similar to each other so I there is good agreement.

```
par(mfrow = c(1,1))
plot(fit.gumbel, type = "density")
abline(v = mu.param, col = 2, lwd = 4, lty = 2)
```



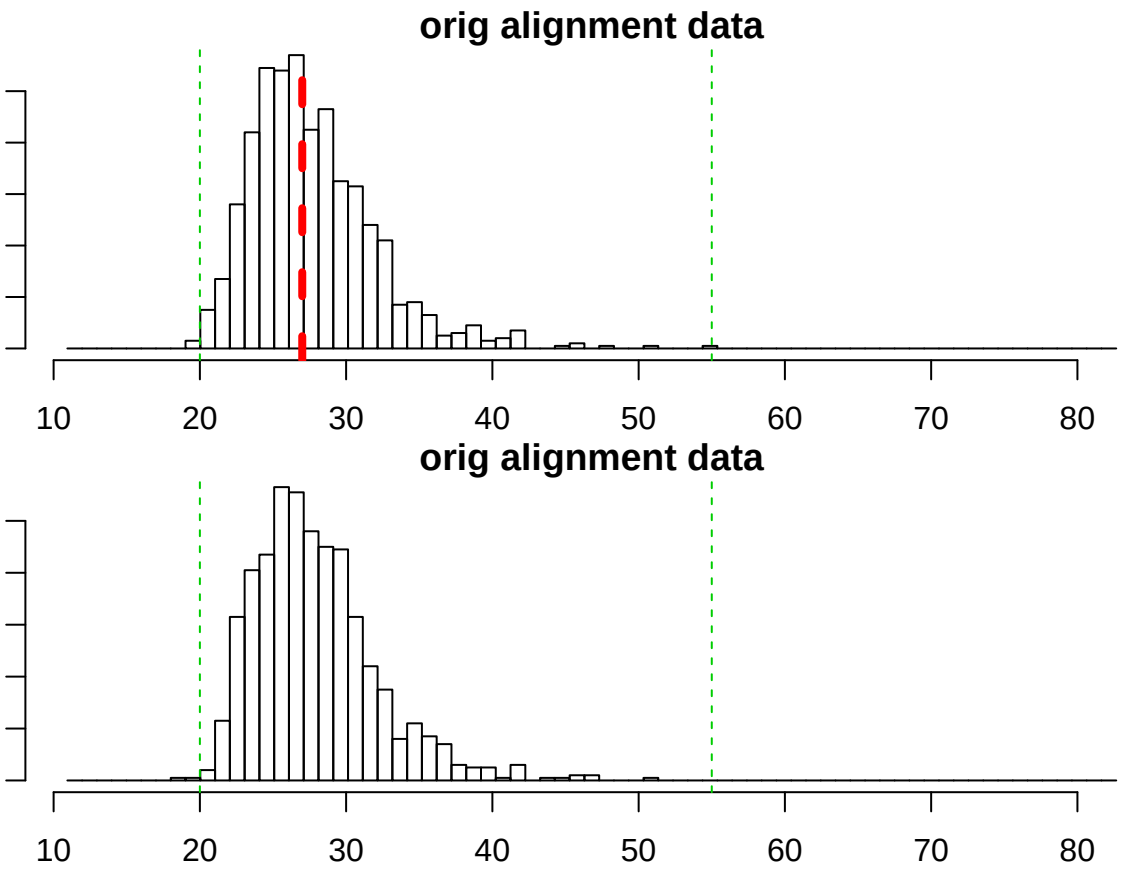
I can also simulate data based on the mu and scale (1/lambda) from our model

```
revd.from.model <-  revd(n = ns, loc = mu.param, scale = 1/lambda.parm, type = "GEV")
```

Now I'll compare this to the observed data

```
# observed data
par(mfrow = c(2,1), mar = c(2,1,1,2))
hist(random.scores.191$score.i,
     breaks = breakz,
     main = "orig alignment data",
     xlim = xlims)
abline(v = mode.i, col = 2, lwd = 4, lty = 2)
abline(v = max.i, col = 3, lwd = 1, lty = 2)
abline(v = min.i, col = 3, lwd = 1, lty = 2)

# simulated data from fitted model
hist(revd.from.model,
     breaks = breakz,
     main = "orig alignment data",
     xlim = xlims)
abline(v = max.i, col = 3, lwd = 1, lty = 2)
abline(v = min.i, col = 3, lwd = 1, lty = 2)
```


orig alignment data
10
20
30
40
50
60
70
80
orig alignment data
10
20
30
40
50
60
70
80