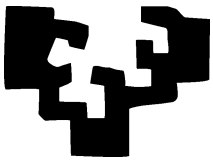
eman ta zabal zazu



Universidad del País Vasco

Euskal Herriko Unibertsitatea

Activity 2

1 Synopsis.

What this activity is about. In the previous seminar you saw how to assess estimators using the Monte Carlo method. You now have to use this technique again in conjunction with the theory, to examine the properties of two alternative estimators of the parameters of a Gumbel distribution.

What you need. You need to be fully acquainted with the content of previous seminars and practice assignment. You will also need access to a computer equiped with R.

2 Background

The Gumbel distribution is commonly used to model the distribution of extreme events: largest rainfall, strength of hurricanes, etc. in a period. The distribution function, dependent on two parameters μ (location) and β (scale, or dispersion) is given by:

$$F_X(x;\mu,\beta) = e^{-e^{-(x-\mu)/\beta}} \tag{1}$$

and the density by:

$$f_X(x;\mu,\beta) = \frac{1}{\beta} e^{-e^{-(x-\mu)/\beta}} e^{-(x-\mu)/\beta} \tag{2}$$

It can be shown that the mean and variance are given by

$$m = \mu + \gamma\beta \tag{3}$$

$$\sigma^2 = \beta^2\pi^2/6 \tag{4}$$

where γ is the so-called Euler-Mascheroni constant,

$$\gamma \approx 0.5772156649\ldots;$$

in order not to type it, in R can be computed as:

```
> - digamma(1)

[1] 0.5772157
```

There is no function in base R to compute density, distribution, etc. for a Gumbel distribution, but they are easy enough to to write. As for the generation of random numbers with a Gumbel distribution, you may use the fact that if U is distributed as $U(0,1)$, then X given by

$$X = \mu - \beta \ln[-\ln(U)] \tag{5}$$

is distributed as Gumbel with parameters μ and β.

Armed with all these facts, you are requested to compare the moment and maximum likelihood estimators for μ and β. The required work is similar to what you did in Activity 1; further insight will have been gained in Seminar 2, presenting a similar problem.

3 Problems

1. Write a function which computes the density of the Gumbel distribution for given x, μ and β. It will be used in the next question and to compute the likelihood.

2. Plot the density of the Gumbel distribution with $\mu = 2$ and $\beta = 1.5$ between 0 and 10.

3. Check that the log likelihood associated with a sample $(x_1, \ldots, x_n)$ is:

$$\log \ell(x_1, \ldots, x_n; \mu, \beta) = -n \log(\beta) - \sum_{i=1}^{n} e^{-(x_i - \mu)/\beta} - \sum_{i=1}^{n} (x_i - \mu)/\beta \tag{6}$$

4. Write a function that computes such log likelihood for a given sample stored in vector `sample`.

5. For a sample size $n = 100$ do the following:

   (a) Generate $N = 1000$ samples from the Gumbel distribution with $\mu = 2$ and $\beta = 1.5$, using (5).

   (b) For each sample, compute the moment and maximum likelihood estimators of the μ and β.

   (c) Compare both estimators looking in particular to:

      i. The estimated bias

      ii. The variance

      iii. The mean square error (MSE).

6. Is it worth here the extra trouble of using the MLE for the sample size considered?

7. *(optional)* Prove the following,

Theorem. If $F_X(x)$ is an strictly increasing distribution function, $U \sim U(0,1)$ and $X = F_X^{-1}(U)$, then $X \sim F_X(x)$.

It only takes one line! This is the general result which in the particular case of the Gumbel distribution justifies (5) above.

Respuesta:

$$P(X \leq x) = P(F_X^{-1}(U) \leq x) = P(U \leq F_X(x)) = F_X(x)$$

4 Hints, comments, complements

1. Concerning question 7: you have to show that

$$P(X \leq x) = F_X(x)$$

where $F_X(x)$ is the distribution function of the Gumbel. If you replace X by $F_X^{-1}(U)$, the conclusion will follow at once.

2. Looking at what we did in Seminar 2 will set you on track with the simulation.

3. To compute the log likelihood, remember that

$$\log \ell(x_1, \ldots, x_n; \mu, \beta) = \log \prod_{i=1}^{n} f_X(x_i; \mu, \beta) = \sum_{i=1}^{n} \log f_X(x_i; \mu, \beta) \tag{7}$$

When, as is the case here, you have a density with two levels of exponents, it is probably less error prone to take the logs first and add them up (rather than computing the product and taking the log).

4. You are requested to use the formulae given and write the code to compute the likelihood, etc. However, *for checking purposes*, you may turn to package `evd`, which offers the usual set of functions `{d,p,q,r}gumbel` to assist with the Gumbel distribution.

5. In theory, the function computing the density of the Gumbel distribution should have three arguments, the values of x, μ and β. However, in order to use the function `curve` for plotting it, we need a function which only takes argument x. You can use the trick of setting both μ and β outside, like this:

```
> mu <- 2 ; beta <- 1.5
>    gumbel <- function(x) {
       # ... your code for computing the density
       #     as a function of x, mu and beta     ...
     }
```

and then use

```
> curve(gumbel,from=0,to=10)
```

to do the plot.

6. The moment estimators of μ and β can be obtained by equating the sample and population moments: the sample mean to (3) and the sample variance to (4). This gives two equations from which $\hat{\mu}_{MLE}$ and $\hat{\beta}_{MLE}$ can be obtained.

7. For the MLE, you might write the likelihood or log likelihood, set the two derivatives equal to zero and solve for μ and β; but the two equations are non-linear, not easy two solve. It is just as easy (and more general) to write a function which computes the log likelihood, receiving the values of the parameters as arguments in one vector, `par`, and pass it to a function which does the maximization. Something like,

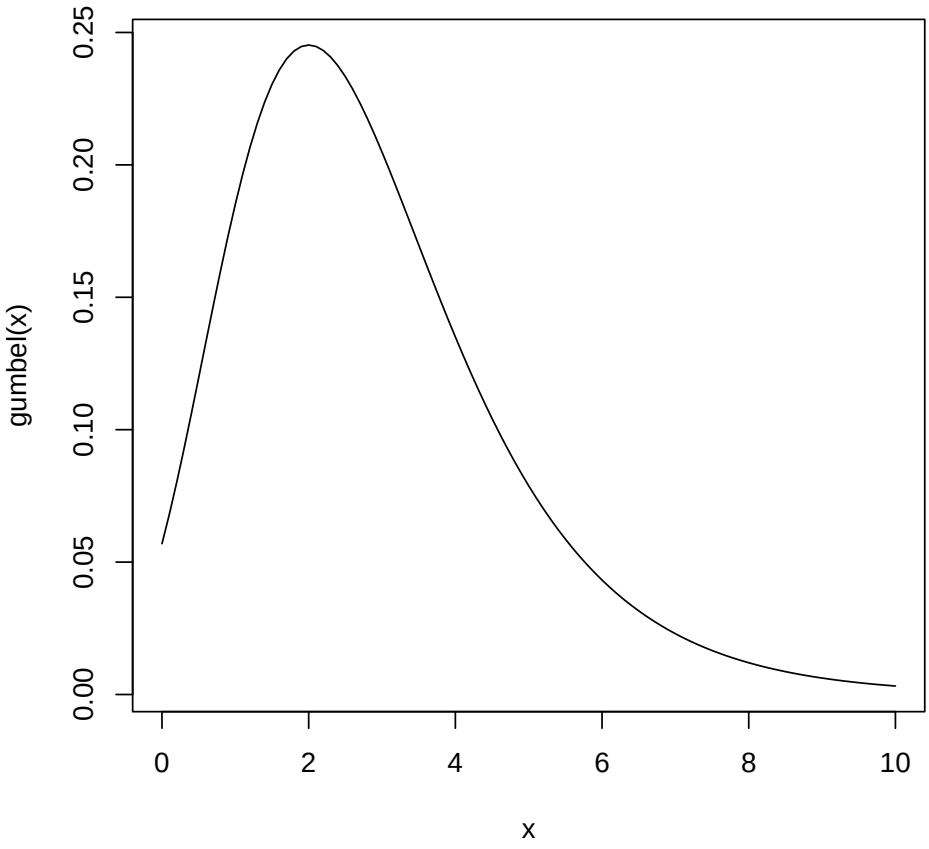
```
> gumbel.lik <- function(par) {
    # ...  Your code here, accessing 'sample' defined beforehand,
    #      and returning the loglikelihood in 'll'                ...
    return(ll)
  }
```

and then invoke function `optim` as demonstrated in Seminar 2.

5 Example solution

Compute and plot the Gumbel density:

```
> mu <- 2 ; beta <- 1.5
> gumbel <- function(x) {
    (1/beta)*exp(-(x-mu)/beta)*exp(-exp(-(x-mu)/beta))
  }
> curve(gumbel,from=0,to=10)
```



Set the constants controlling the simulation. In `results` we will store the estimates computed in each iteration.

```
> N <- 1000
> n <- 100
> eulerm = 0.57721566490153
> # else you could use: eulerm <- - digamma(1)
> sample  <- rep(0,n)
> results <- matrix(0,N,4)
> colnames(results) <- c("mu.M", "beta.M", "mu.MLE", "beta.MLE")
```

A function computing the log likelihood associated with the data in `sample`:

```
> gumbel.lik <- function(par) {
    n <- length(sample)
    mu <- par[1] ; beta <- par[2]
    ll <- - sum((sample-mu))/beta -
      sum(exp(-(sample-mu)/beta)) -
      n*log(beta)
    return(ll)
  }
```

One could also make use of the `evd` package:

```
> require(evd)
> gumbel.lik <- function(par) {
    n <- length(sample)
    mu <- par[1] ; beta <- par[2]
    ll <- sum(dgumbel(sample,loc=mu,scale=beta, log=TRUE))
    return(ll)
  }
```

Now the main loop of the simulation. We set the seed to have reproducible results:

```
> set.seed(1234)
> for (i in 1:N) {
    sample <- mu - beta*log((-log(runif(n))))          # sample generation
    par    <- c(1,1)                                   # initial values
    sol    <- optim(par, gumbel.lik,                   # maximize likelihood
                    control=list(fnscale=-1))
    par <- sol$par                                     # MLEs of parameters
    results[i,3] <- par[1]                             # save in their places
    results[i,4] <- par[2]
    results[i,2] <- betahat <- sqrt(6*var(sample)/pi^2)   # moment estimators of
    results[i,1] <- mean(sample) - eulerm * betahat       # 'beta' and 'nu'
  }
>
```

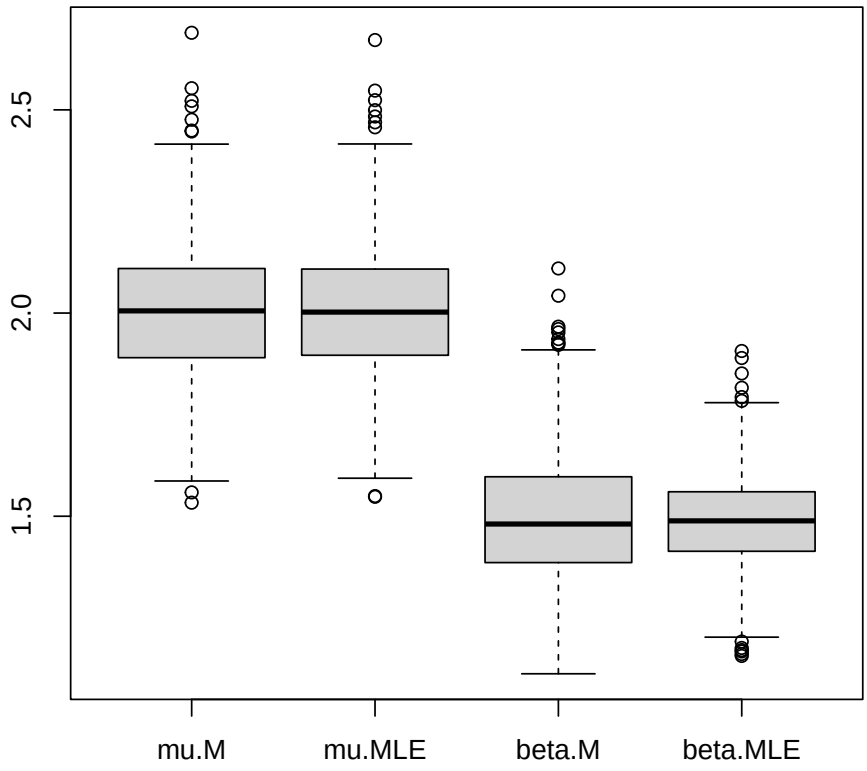
Once done with the simulation, analyze results. Both estimators have no or small bias, which seems to be slightly larger with the MLE:

```
> colMeans(results)

    mu.M   beta.M   mu.MLE beta.MLE
2.006826 1.494185 2.007985 1.489558
```

Concerning the dispersion, it appears that the MLE has smaller variance:

```
> boxplot(results[,c(1,3,2,4)])
>
```



This is in fact confirmed when we compute the sample variance of the estimators. The reduction is hardly worth the trouble for μ, but quite significant for β:

```
> apply(results,2,var)

      mu.M     beta.M     mu.MLE   beta.MLE
0.02714589 0.02421099 0.02634043 0.01283794
```

In terms of MSE, this is how both estimators fare:

```
> apply(results,2,var) + ( colMeans(results) -
                           c(2, 1.5, 2, 1.5) )^2

      mu.M     beta.M     mu.MLE   beta.MLE
0.02719248 0.02424481 0.02640419 0.01294697
```