

Universidade Estadual de Maringá
Estatística computacional II

Distribuição Gumbel

Wesley Oliveira Furriel RA:61493

Dr. Josmar Mazucheli

Maringá
2016

Conteúdo

1 Distribuição Gumbel	2
1.1 Implementando a Gumbel no R	3
1.2 Implementando a Gumbel no SAS	7
2 Método de Máxima Verossimilhança	10
2.1 Implementando no R	10
2.2 Implementando no SAS	11
3 Vetor Score e Matriz Hessiana	13
3.1 Implementando no R	14
4 Método dos Momentos	14
4.1 Implementando no R	15
5 Simulação	17
5.1 Implementando no SAS	17
5.2 Implementando no R	18
6 Bibliografia	20

1 Distribuição Gumbel

A distribuição Gumbel ou também conhecida como valor extremo, surge quando consideramos o logaritmo natural de uma variável aleatória com distribuição de Weibull. Ou seja, se a variável x tem uma distribuição Weibull com parâmetros α (escala) e β (forma), então a variável $Y = \log(x)$ detém uma distribuição Gumbel.

Em teoria da probabilidade e estatística, a distribuição Gumbel é usada para modelar a distribuição do máximo ou mínimo de um número de amostras de várias distribuições. Ela pode ser usada por exemplo, para representar a distribuição do nível máximo de um rio em uma ano particular se existir a lista dos valores máximo obtidos nos últimos 10 anos. Também pode ser usada para prever a chance de um terremoto com valores extremos em sua escala.

Esta distribuição foi nomeada com base nos estudos de Emil Julius Gumbel (1891-1966) e sua função densidade de probabilidade é dada por:

$$f(x | \sigma, \mu) = \frac{1}{\sigma} e^{-\frac{x-\mu}{\sigma}} e^{-e^{-\frac{x-\mu}{\sigma}}}, \quad x \in (-\infty; \infty) \quad (1)$$

Em que o parâmetro de escala é $\sigma > 0$ e o parâmetro de forma é $\mu \in \mathbb{R}$. Além disso, $\sigma = \frac{1}{\delta}$ e $\mu = \log(\alpha)$

A partir de (1) podemos obter a função de distribuição acumulada de Gumbel sendo ela:

$$F(x | \sigma, \mu) = e^{-e^{-\frac{x-\mu}{\sigma}}} \quad (2)$$

No que diz respeito a construção dos quantis da Gumbel, verificamos que se $q \in (0, 1)$ nós resolvemos a equação:

$$F(x) = e^{-e^{-\frac{x-\mu}{\sigma}}} = q \quad (3)$$

em que a solução é:

$$x_q = \mu - \ln(-\ln(q))\sigma \quad (4)$$

Como é possível simular variáveis aleatórias uniformemente distribuídas em $[0, 1]$, é possível estender este resultado para a simulação de outros modelos probabilísticos. Neste, caso faremos isso para a distribuição Gumbel à partir do método da transformada inversa. Para simular amostras aleatórias de uma variável aleatória X com função de distribuição Gumbel, é preciso gerar uma variável aleatória X segundo a propriedade $F_X(x) = \mathbb{P}(X \leq x)$ para todo x . Desse modo, geramos uma v.a. U uniformemente distribuída em $[0, 1]$, isto é, $U \sim U[0, 1]$ e então, pelo método da transforma inversa estabelecemos que a variável aleatória $X = F_X^{-1}(U)$.

$$X = F^{-1}(U) = \mu - \ln(-\ln(U))\sigma \quad (5)$$

1.1 Implementando a Gumbel no R

No *software R* é possível construir a família de funções **d**, **p**, **q**, **r** da distribuição Gumbel, o código foi implementado como pode ser observado abaixo, além disso utilizamos a biblioteca *reliar* que já tem a Gumbel implementada para comparar os resultados.

Script - R 1.1: Script da distribuição Gumbel no R

```
dir<-"D:/Estatística/ESTATÍSTICA COMPUTACIONAL_II/MLE/Gumbel_Latex/"
setwd(dir)
library(MASS)
library(fitdistrplus)
#install.packages("reliar", dependencies = T)
#Biblioteca da Gumbel
library(reliar)
rm(list = ls())
# -----
# Função densidade de probabilidade (d) fdp
gumbeld<-function(x,mu,sigma){
  aux<-((x - mu)/sigma)
  exp(-aux - exp(-aux)) / sigma
}
gumbeld(x=2, mu=1.5, sigma=2)
dgumbel(x=2, mu=1.5, sigma=2)
# Função distribuição de prob. (p) dist. acumulada
gumbelp<-function(q,mu,sigma){
  exp(-exp((-q + mu) / sigma))
}
gumbelp(q=2, mu=1.5, sigma=2)
pgumbel(q=2, mu=1.5, sigma=2, lower.tail=TRUE, log.p=FALSE)
# Função quantil (q)
gumbelq<-function(p,mu,sigma){
  mu - sigma*log(-log(p))
}
gumbelq(p=0.5, mu=1.5, sigma=2)
qgumbel(p=0.5, mu=1.5, sigma=2, lower.tail = TRUE, log.p = FALSE)
#Função para valores aleatorios
gumbelr<-function(n,mu,sigma){
  U<-runif(n)
  gumbelq(U,mu,sigma)
}
gumbelr(n=1, mu=2, sigma=3)
rgumbel(n=1, mu=2, sigma=3)
# Verificando o ajuste
# -----
va<-gumbelr(n=1000, mu=2, sigma=1.5)
jpeg("graph0.jpeg", height = 6, width = 8, units = 'in', res=800)
hist(va, probability = T)
```

```

curve(gumbeld(x, mu=2, sigma=1.5), add=TRUE, col="blue", lwd=2)
dev.off()
# -----
# Grafico da densidade
x <- seq(-4, 20, length=100)
hx<-gumbeld(x=x, mu=0, sigma=1)
hx1<-gumbeld(x=x, mu=0.5, sigma=2)
hx2<-gumbeld(x=x, mu=1.5, sigma=3)
hx3<-gumbeld(x=x, mu=3, sigma=4)
jpeg("graph1.jpeg", height = 6, width = 8, units = 'in', res=800)
plot(x, hx, type="l", lty=2, lwd=2)
colors <- c("black", "blue", "darkgreen", "red")
labels <- c(expression(mu ~ "=0"~~~sigma~"=1"),
            expression(mu ~ "=0.5"~sigma~"=2"),
            expression(mu ~ "=1.5"~sigma~"=3"),
            expression(mu ~ "=3"~~~sigma~"=4"))
lines(x,hx1,col=colors[2],lwd=2)
lines(x,hx2,col=colors[3],lwd=2)
lines(x,hx3,col=colors[4],lwd=2)
legend("topright", inset=.05,
       labels, lwd=2, lty=c(2, 1, 1, 1), col=colors)
dev.off()

# -----
# Grafico da acumulada
hx<-gumbelp(q=x, mu=0, sigma=1)
hx1<-gumbelp(q=x, mu=0.5, sigma=2)
hx2<-gumbelp(q=x, mu=1.5, sigma=3)
hx3<-gumbelp(q=x, mu=3, sigma=4)
jpeg("graph2.jpeg", height = 6, width = 8, units = 'in', res=800)
plot(x, hx, type="l", lty=2, lwd=2)
lines(x,hx1,col=colors[2],lwd=2)
lines(x,hx2,col=colors[3],lwd=2)
lines(x,hx3,col=colors[4],lwd=2)
legend("bottomright", inset=.05,
       labels, lwd=2, lty=c(2, 1, 1, 1), col=colors)
dev.off()

# -----
y<- gumbelr(n=200, mu=2, sigma=3)
fit<-fitdist(y, "gumbel", start=list(mu=2, sigma=3))
jpeg("graph3.jpeg", height = 6, width = 8, units = 'in', res=800)
plot(fit)
dev.off()

```

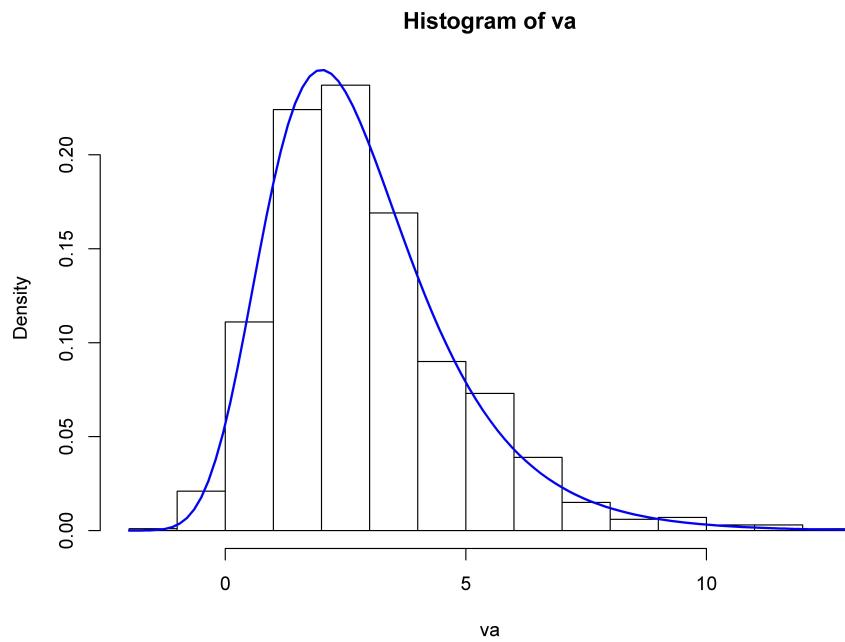


Figura 1: Ajuste da distribuição Gumbel

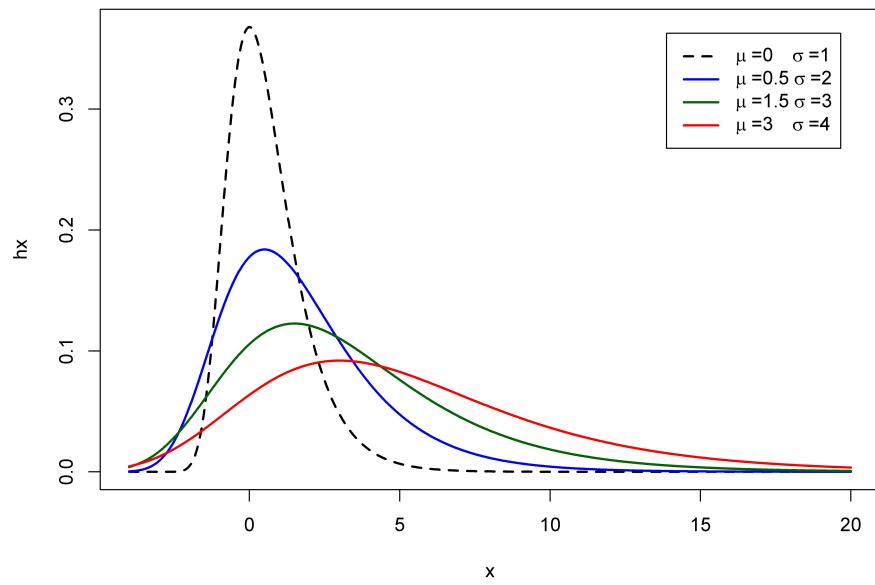


Figura 2: Função densidade da distribuição Gumbel

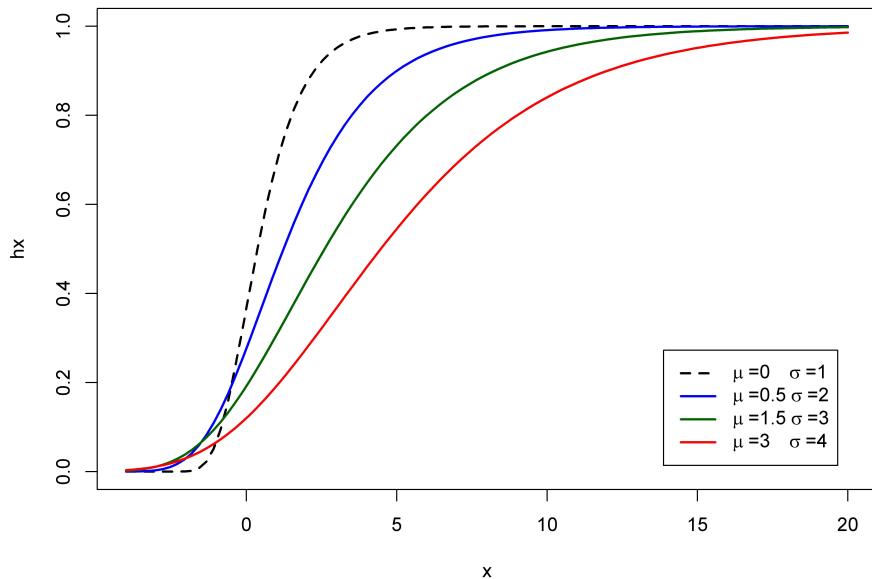


Figura 3: Função de distribuição acumulada Gumbel

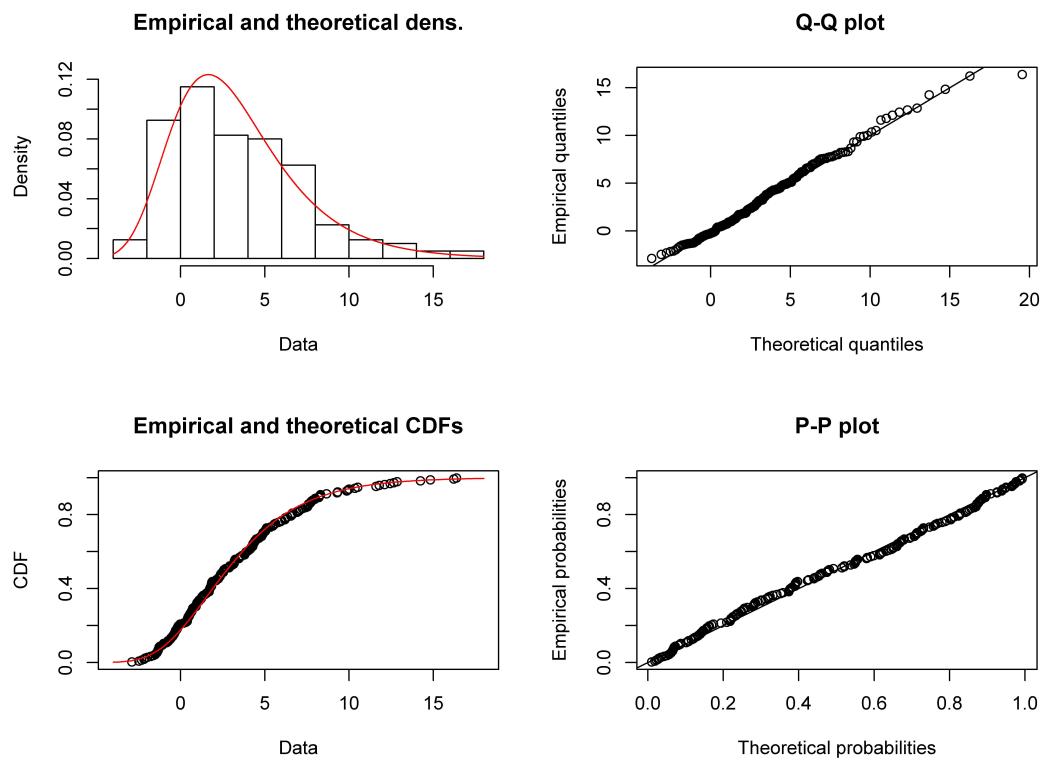


Figura 4: Densidade empírica e teórica

1.2 Implementando a Gumbel no SAS

Nesta seção implementamos as funções de densidade (d), a de distribuição (p), a quantil (q) e variate(random) (r) para valores aleatórios pelo SAS Data step e também, a partir do PROC IML, com o intuito de comparar as diferenças, dificuldades e facilidades de cada uma dessas formas de implementação. Os scripts com os detalhes dessas implementações, bem como as verificações para validação de suas funcionalidades seguem abaixo.

Script - SAS 1.1: Script da distribuição Gumbel via SAS Data Step

```
proc delete data=_all_;
run;
goptions reset=all;

%macro graph(dist=, var=, dado=, nome=);
ods html style=harvest;
options nodate nonumber;
options orientation=portrait;
ods pdf file="D:\Estatistica\ESTATISTICA_COMPUTACIONAL_II\
MLE\Gumbel_Latex\&nome..pdf" startpage= yes;
legend1 position=(top left inside) mode=protect
            value=("mu=0.5 sigma=2") label=none;
axis1 minor=none order=(-5 to 20 by 2);
symbol1 i=j v=none l=1 c=blue w=2;
proc gplot data=&dado;
plot &dist*&var/overlay haxis=axis1noframe autovref legend=legend1;
run;
ODS pdf close;
ods close;
%mend graph;

* Função densidade de probabilidade (d) fdp;
data dist1;
do x = -5 to 20 by 0.1;
    mu=0.5;
    sigma=2;
    dgumbel= exp((-x + mu) / sigma - exp((-x + mu) /
        sigma))/sigma;
    output;
end;
run;
*Função distribuição de prob. (p) dist. acumulada;
data dist2;
do q = -5 to 20 by 0.1;
    mu=0.5;
    sigma=2;
    pgumbel= exp(-exp((-q + mu) / sigma));
    output;
end;
run;
```

```

end;
run;
%graph(dist=dgumbel, var=x, dado=dist1,nome=densidade);
%graph(dist=pgumbel, var=q, dado=dist2,nome=acumulada);
*Funcao quantil (q);
data dist3;
do p = 0 to 1 by 0.01;
    mu=0.5;
    sigma=2;
    qgumbel= mu - sigma*log(-log(p));
    output;
end;
run;
*Funcao para valores aleatorios;
data dist4(drop=mu sigma u i);
do i= 1 to 10000;
call streaminit(666);/*Semente*/
    mu=0.5;
    sigma=2;
    u=rand("uniform");
    rgumbel=-log(-log(u))*sigma+mu;
    output;
end;
run;
proc nlp data=dist4 tech=nra;
    max l;
    parms mu=0.5 ,sigma=2;
    x=rgumbel;
    aux=(-x + mu) / sigma;
    l = l-log(sigma)+(log(exp(aux-exp(aux))));
run;

```

Script - SAS 1.2: Script da distribuição Gumbel via PROC IML

```

proc delete data=_all_;
run;
goptions reset=all;

proc iml;
/*Funcao densidade de probabilidade (d) fdp */
start dgumbel(x, mu, sigma);
    fdp= exp((-x + mu) / sigma - exp((-x + mu) / sigma))/sigma;
    return(fdp);
finish;
x = dgumbel(2,1.5,2);
print(x);
quit;

```

```

/*Funcao distribuicao de prob. (p) dist. acumulada*/
proc iml;
start pgumbel(q,mu,sigma);
    cdf= exp(-exp((-q + mu) / sigma));;
    return(cdf);
finish;
q = pgumbel(2,1.5,2);
print(q);
quit;
/*Funcao quantil (q)*/
proc iml;
start qgumbel(p,mu,sigma);
    qf=mu - sigma*log(-log(p));
    return(qf);
finish;
p = qgumbel(0.5,1.5,2);
print(p);
quit;
/*Funcao para valores aleatorios*/
proc iml;
start rgumbel(n,mu,sigma);
    aux      = j(n,1);
    call randgen(aux, "uniform");
    ran     = -log(-log(aux))*sigma+mu;
    return(ran);
finish;
call randseed(666);
random_gumbel = rgumbel(10,1.5,2);
print(random_gumbel);
quit;

```

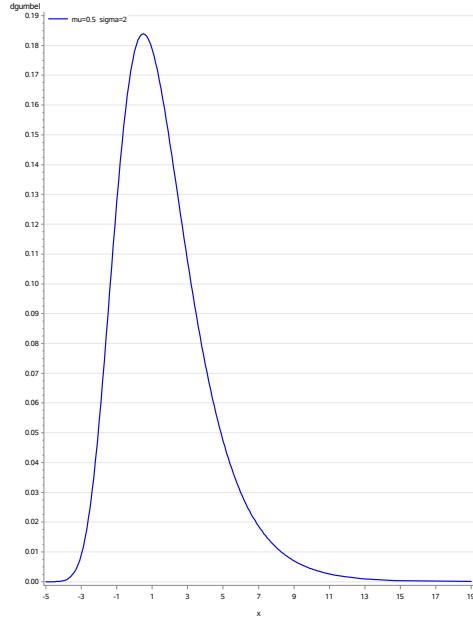


Figura 5: Função densidade de probabilidade Gumbel no SAS

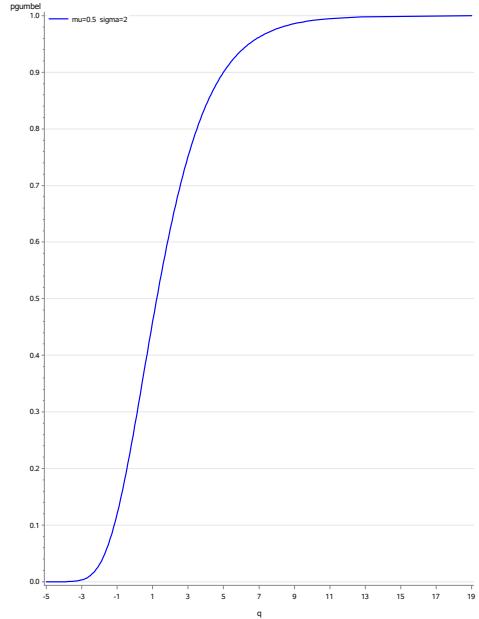


Figura 6: Função de distribuição acumulada Gumbel no SAS

2 Método de Máxima Verossimilhança

Aqui mostraremos o método de estimação para os parâmetros da Função Gumbel com o método da Máxima Verossimilhança (MLE). Sejam x_1, x_2, \dots, x_n um conjunto de observações oriundos de (1) a função de verossimilhança de (μ, σ) pode ser escrita na forma:

$$L(\mu, \sigma | \underline{x}) = \prod_{i=1}^n \frac{1}{\sigma} e^{-\frac{x_i+\mu}{\sigma}} - e^{-\frac{x_i+\mu}{\sigma}} \quad (6)$$

Aplicando log obtemos a função de log-verossimilhança:

$$l(\mu, \sigma | \underline{x}) = -n \ln (\sigma) + \sum_{i=1}^n \ln \left(e^{-\frac{x_i+\mu}{\sigma}} - e^{-\frac{x_i+\mu}{\sigma}} \right) \quad (7)$$

2.1 Implementando no R

Script - R 2.1: Implementação do MLE no R

```
# Estimacao por MLE
mu <- 1.5
sigma <- 2
theta <- c(mu, sigma)
set.seed(666)
x <- gumbelr(n=10000, mu=theta[1], sigma=theta[2])
n <- length(x)
```

```

logLEgumbel <- function(theta){
  mu <- theta[1]
  sigma <- theta[2]
  aux<-(-x+mu)/sigma
  mle<--n * log(sigma) + sum(log(exp(aux - exp(aux))))
  return(mle)
}
library(maxLik)
maxLik(logLik = logLEgumbel, start = c(1.3,1.9))

```

Script 1: R output

```

Maximum Likelihood estimation
Newton-Raphson maximisation, 4 iterations
Return code 1: gradient close to zero
Log-Likelihood: -2254.967 (2 free parameter(s))
Estimate(s): 1.508984 1.973307

```

2.2 Implementando no SAS

Script - SAS 2.1: Implementação do MLE no SAS IML

```

*MLE no SAS;
proc iml;
start MLEgumbel(param) global(x);
  mu = param[1];
  sigma = param[2];
  n      = nrow(x);
  mle = -n#log(sigma) + sum(log(exp((-x+mu)/sigma -
    exp((-x+mu)/sigma))));
  return (mle);
finish;
/*Gerar valores aleatorios*/
start rgumbel(n,mu,sigma);
  aux      = j(n,1);
  call randgen(aux,"uniform");
  ran      = -log(-log(aux))*sigma+mu;
  return(ran);
finish;
call randseed(666);
x = rgumbel(10000,1.5,2);
print(x);
sup = {. 0, /* Li -inf < mu; 0 < sigma */
       . .}; /* Ls mu < inf; sigma < inf */
iniciais = {1.3 2}; /*Valores iniciais*/
opt = {1,2};

```

```

call nlpnra(it, result, "MLEgumbel", inicioais, opt, sup);
print it result inicioais/*Mostrar Resultados*/;
quit;

```

Script - SAS 2.2: Estimação no SAS

```

*Funcao para valores aleatorios;
data dist4(drop=mu sigma u i);
do i= 1 to 5000;
call streaminit(666); /*Semente*/
    mu=0.5;
    sigma=2;
    u=rand("uniform");
    rgumbel=-log(-log(u))*sigma+mu;
output;
end;
run;
* Estima dos Parametros;
proc nlp data=dist4 tech=nra;
    max l;
    parms mu=0.5 ,sigma=2;
    x=rgumbel;
    aux=(-x + mu) / sigma;
    l = -log(sigma)+(log(exp(aux-exp(aux))));
run;

```

Optimization Start						
Parameter Estimates						
N	Parameter	Estimate	Gradient Objective Function	Lower Bound Constraint	Upper Bound Constraint	
1	X1	1.300000	385.247113	.	.	.
2	X2	2.000000	-62.591471	0	.	.

Value of Objective Function = -22843.21959

Iteration	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	0	-22837	6.1442	346.6	0.106	-64.681
2	*	0	3	0	-22824	13.0863	243.6	0.295
3	*	0	4	0	-22811	12.6570	13.7354	1.000
4	*	0	5	0	-22811	0.0212	0.0543	1.000
5	0	6	0	-22811	3.193E-7	0.000693	1.000	-664E-9

Optimization Results						
Parameter Estimates						
N	Parameter	Estimate	Gradient Objective Function			
1	X1	1.467677	-0.000693			
2	X2	2.017283	0.000647			

Value of Objective Function = -22811.31088

it	result		iniciais	
6	1.4676775	2.0172835	1.3	2

3 Vtor Score e Matriz Hessiana

Sabemos que as estimativas de máxima verossimilhança são os parâmetros μ, σ que maximizam a função verossimilhança da amostra. Estabelecendo que os dados observados sejam o mais provável possível. Dessa forma os valores de $\hat{\mu}$ e $\hat{\sigma}$ são as raízes do vetor gradiente (vetor das primeiras derivadas da função log-verossimilhança), sendo conhecido também como vetor score:

$$U(\mu, \sigma, \underline{x}) = \begin{cases} \frac{1}{\sigma^2} \left(-n\mu - n\sigma + \sum_{i=1}^n e^{\frac{-x_i+\mu}{\sigma}} \mu - e^{\frac{-x_i+\mu}{\sigma}} x_i + x_i \right) \\ -\frac{1}{\sigma} \left(-n + \sum_{i=1}^n e^{\frac{-x_i+\mu}{\sigma}} \right) \end{cases} \quad (8)$$

De (6), a matrix $H(\theta)$, matrix hessian de $l(\mu, \sigma | \underline{x})$, $H(\mu, \sigma | \underline{x})$, é dada por:

$$H(\mu, \sigma | \underline{x}) = \begin{bmatrix} -\frac{-2n\mu\sigma - n\sigma^2 + \sum_{i=1}^n e^{W_i}\mu^2 + 2e^{W_i}\mu\sigma - 2e^{W_i}x_i\mu - 2e^{W_i}x_i\sigma + e^{W_i}x_i^2 + 2x_i\sigma}{\sigma^4} & \frac{-n\sigma + \sum_{i=1}^n e^{W_i}(\mu + \sigma - x_i)}{\sigma^3} \\ \frac{-n\sigma + \sum_{i=1}^n e^{W_i}(\mu + \sigma - x_i)}{\sigma^3} & -\frac{\sum_{i=1}^n e^{W_i}}{\sigma^2} \end{bmatrix} \quad (9)$$

onde $W_i = \frac{-x_i + \mu}{\sigma}$

3.1 Implementando no R

Script - R 3.1: Score e Hessiana no R

```
# -----
# Score
Uscore <- function(n, x, theta){
  mu <- theta[1]
  sigma <- theta[2]
  W <- -(x - mu)/sigma
  Es <- matrix(nrow=2)
  Es[1] <- n / sigma - sum(1 / sigma * exp(W))
  Es[2] <- (sum((mu-x) * exp(Z) + x) - (mu + sigma) * n) / sigma ^ 2
  return(Es)
}
# -----
# Matriz Hessiana
Hessian <- function(n, x, theta){
  mu <- theta[1]
  sigma <- theta[2]
  W <- -(x - mu)/sigma
  Hs <- matrix(ncol=2, nrow=2)
  Hs[1,1] <- -1 / sigma ^ 2 * sum(exp(W))
  Hs[1,2] <- (-n * sigma + sum(exp(W) * (sigma - x + mu))) / sigma ^ 3
  Hs[2,1] <- Hs[1,2]
  Hs[2,2] <- (2 * n * mu * sigma + n * sigma ^ 2
    - sum((-x+mu)*(mu+2*sigma-x)*exp(W) + 2 * x * sigma)) / sigma ^ 4
  return(Hs)
}
```

4 Método dos Momentos

Outra maneira de encontrar estimadores dos parâmetros, é através do método dos momentos. Ele é baseado nos momentos teóricos e amostrais das variáveis aleatórias envolvidas. O estimador de momentos do parâmetro de forma μ e da escala σ de uma distribuição

Gumbel, podem ser encontrados pela equação dos momentos amostrais correspondentes aos momentos teóricos. Eles são soluções das seguintes igualdades.

$$\bar{X} = \mu + \gamma\sigma \quad (10)$$

onde $\gamma \approx 0.5772$ uma constante Euler–Mascheroni

$$S^2 = \frac{\pi^2}{6}\sigma^2 \quad (11)$$

os ME de μ e σ são obtidos como respectivamente

$$\bar{\mu}_{ME} = \bar{X} - \gamma\bar{\sigma}_{ME} \quad (12)$$

$$\bar{\sigma}_{ME} = \frac{\sqrt{6}}{\pi} S \quad (13)$$

4.1 Implementando no R

Script - R 4.1: Momentos no R

```
dir<-"D:/Estatística/ESTATÍSTICA COMPUTACIONAL_II/MLE/Gumbel_Latex/"
setwd(dir)
rm(list = ls())
# -----
# Método dos momentos
library(reliaR)
x<-rgumbel(n=1000, mu=1.5, sigma=2)
MMgumbel <- function(theta, x) {
  mu <- theta[1]
  sigma <- theta[2]
  xbar <- mu + 0.5772*sigma
  s <- (pi^2/6)*sigma^2
  m1 <- xbar - x
  m2 <- s - (x - xbar)^2
  f <- cbind(m1, m2)
  return(f)
}
library(gmm)
mm<-gmm(MMgumbel, x, c(mu=1.5, sigma=2)); mm
vcov(mm)
muest<-mm$coefficients[1]
sigmaest<-mm$coefficients[2]
# -----
library(fastR)
qgumbel.plot <- function(x){qgumbel(x, muest, sigmaest)}
jpeg("momento_plot.jpeg", height = 6, width = 8, units = 'in', res=800)
```

```

xqqmath(x, distribution = qgumbel.plot,
        xlab = "Quantil gumbel",
        ylab = "Quantil amostral")
dev.off()

```

Script 2: R output

```

Method
twoStep

Objective function value:  4.681648e-09

      mu      sigma
1.4485  1.9904

Convergence code =  0

      mu      sigma
mu    0.0083286115 0.0005179957
sigma 0.0005179957 0.0053320877

```

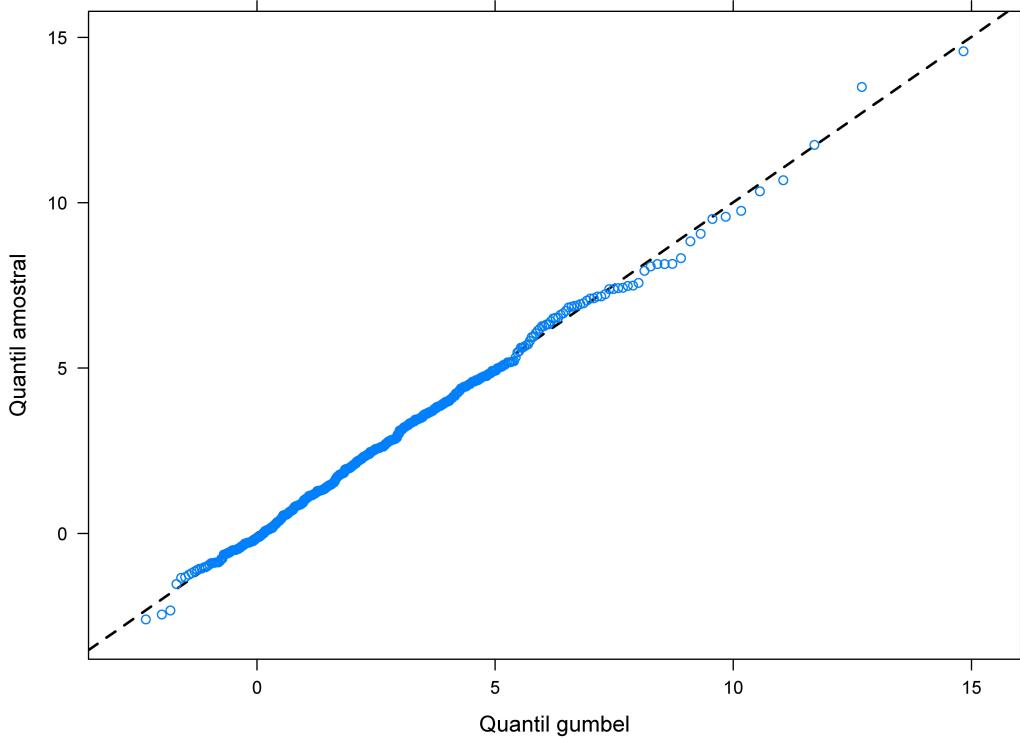


Figura 7: Q-Q plot Momento

5 Simulação

A aleatoriedade dos métodos de simulação baseia-se na geração de uma sequência de v.a independentes x_1, x_2, \dots, x_n uniformemente distribuídas no intervalo $[0, 1]$. As primeiras técnicas utilizadas para gerar valores aleatórios utilizavam-se de processos físicos, que eram ditos como aleatórios. Porém, para a grande maioria das tabelas de números aleatórios obtidos por processos físicos foram identificados vícios e dependências. Na estatística a simulação trata-se de um método efetuado via computador que envolve a utilização de números pseudo aleatórios, ou seja, valores determinísticos gerados em computador, que do ponto de vista estatístico podem ser encarados como valores aleatórios independentes de uma Uniforme $[0,1]$.

5.1 Implementando no SAS

Script - SAS 5.1: Simulação no SAS

```
proc delete data=_all_;
run;
/*SIMULA 0*/
%let mu=1;
%let sigma=1.5;
%macro simul(stop=,step=,nsimulacao=);
  %do m = 10 %to &stop %by &step;
data simul;
call streaminit(1212);
  do b = 1 to &nsimulacao;
    do i = 1 to &m;
      u=rand("uniform");
      x=&mu-log(-log(u))*&sigma;
      output;
    end;
  end;
run;
proc nlp data=simul tech=nra nopolish outest=estimacao(keep=_type_
  mu
  sigma where=(_type_="PARMS"));
  max ll;
  parms mu=1,sigma=1.5;
  bounds sigma > 0;
  aux=(-x+mu)/sigma;
  ll= -log(sigma)+(log(exp(aux-exp(aux)))); *EXPRESSAO SEM
  SOMATORIO;
  by b;
run;
data resultados;
set estimacao;
vicio_mu=mu-&mu;
vicio_sigma=sigma-&sigma;
```

```

run;
proc means data=resultados noprint;
var _numeric_;
output out = v&m mean=/autoname;
run;
%end;
%mend simul;
%simul(stop=150,step=10,nsimulacao=400);
/*Juntando os data-sets e apresentando os resultados*/
data resultados_simulacao;
set v10 v20 v30 v40 v50 v60 v70 v80 v90 v100 v110 v120 v130 v140
    v150;
n+10;
if last.mu_mean then n=10;
run;

```

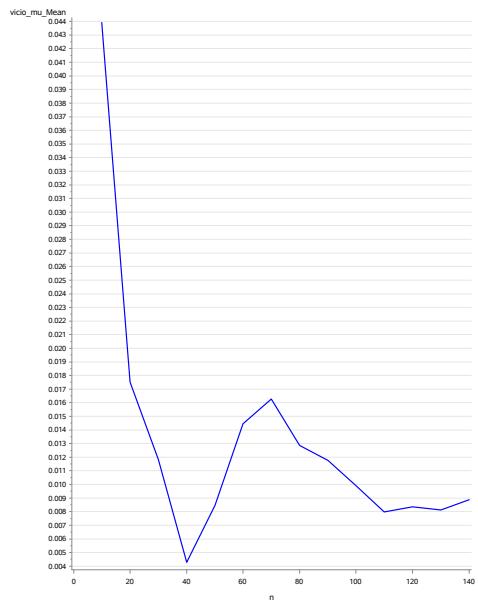


Figura 8: μ

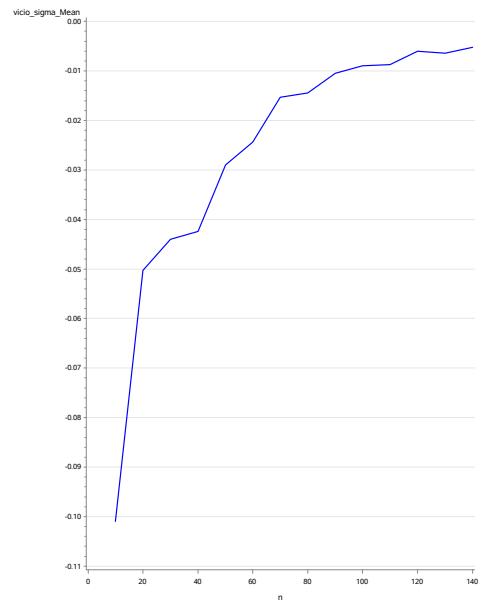


Figura 9: σ

5.2 Implementando no R

Script - R 5.1: Simulação no R

```

library(reliaR)
dir<- "D:/Estatistica/ESTATISTICA_COMPUTACIONAL_II/MLE/Gumbel_Latex/"
setwd(dir)
rm(list = ls())

```

```

# Iniciando a simulacao
par1<-1.5 #mu
par2<-2 #sigma
nrep<-1000
m<-seq(10,100,10)
results<-matrix(nrow=length(m), ncol=3)
mle<-matrix(ncol=2, nrow = nrep)
k<-1
for (n in m){
  set.seed(666)
  for (i in 1:nrep){
    x<- rgumbel(n=n, mu=par1, sigma=par2)
    mle[i,]<-fitdist(x, "gumbel", start=list(mu=par1, sigma=par2))$estimate}
  results[k,1]<-n
  results[k,2]<-mean(mle[,1]) - par1
  results[k,3]<-mean(mle[,2]) - par2
  k<-k+1
  cat(k,i,n, "\n")}
jpeg("graph4.jpeg", height = 6, width = 8, units = 'in', res=800)
par(mfrow = c(1,2))
plot(results[,1], results[,2], type="l", col="red", lwd=2)
plot(results[,1], results[,3], type="l", col="blue", lwd=2)
dev.off()

```

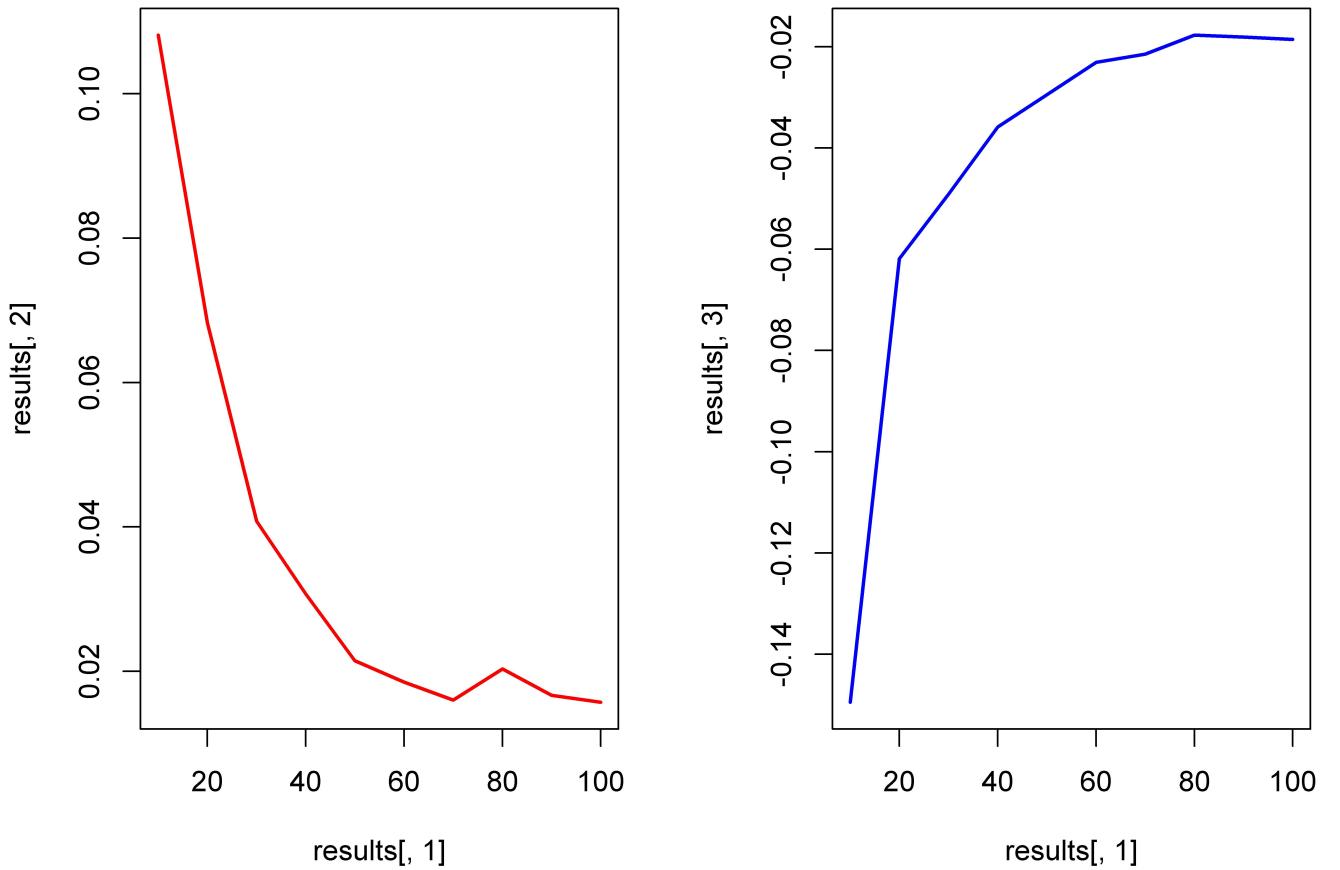


Figura 10: Resultados simulação

6 Bibliografia

Referências

- [1] Melo V. S., *Computer generation of statistical distributions*, DTIC Document ,2000.
- [2] Aydin, Demet and Şenoğlu, Birdal, *Monte Carlo Comparison of the Parameter Estimation Methods for the Two-Parameter Gumbel Distribution*, Journal of Modern Applied Statistical Methods, 2015.
- [3] Steenberger, M. R. (2006). Maximum likelihood programming in R. University of North Carolina, Chapel Hill.
- [4] Mazucheli, J., Ghitany, M. E., and Louzada, F. (2016). Comparisons of Ten Estimation Methods for the Parameters of Marshall-Olkin Extended Exponential Distribution. Communications in Statistics-Simulation and Computation, (just-accepted).