Trabalho de Estatística Computacional II

Aluno: Hevans Vinicius Pereira

Introdução

Neste trabalho irei fazer simulações de Monte Carlo para verificar se há viés e avaliar a qualidade dos estimadores da distribuição Reflected Weibull apresentada no artigo intitulado "Modifications of the Weibull distribution: A review" dos autores Almalki, Saad J. e Nadarajah, Saralees, que foi publicado no "Reliability Engineering and System Safety" em 05 de dezembro de 2013 (http://dx.doi.org/10.1016/j.ress.2013.11.010).

Para muitas aplicações práticas é importante modelar o tempo de vida para um conjunto de dados, e muitas distribuições (Weibull, Pareto, Gompertz, etc) são usadas para este fim. Entretanto, para algumas aplicações as distribuições clássicas não são tão adequadas e, por isso, surge a necessidade de se generalizar tais distribuições a fim de encontrar ajustes melhores. Muitas generalizações foram propostas ao longo dos anos e o artigo em questão apresenta uma nova forma.

A distribuição Reflected Weibull (RefW)

Considerando uma variável aleatória X que segue uma distribuição de Weibull com parâmetros a e b temos que $Y = -X$ tem distribuição reflected Weibull.

Com isto, a função de densidade de probabilidade de Y é $f(y) = \alpha\theta(-y)^{\theta-1}e^{-\alpha(-y)^{\theta}}$ para $-\infty < y < 0$ e $\alpha, \theta > 0$. A função de probabilidade acumulada é dada por $F(y) = e^{-\alpha(-y)^{\theta}}$ para $-\infty < y < 0$ e $\alpha, \theta > 0$.

Vamos implementar a função densidade de probabilidade conforme expressão acima.

```
drefw <- function(x, alpha, theta){
    # y deve ser negativo
    if (any(alpha <= 0))
        return(NA)
    if (any(theta <= 0))
        return(NA)
    if (any(x >= 0))
        stop(paste("x deve ser negativo", "\n", ""))
    if (!length(x)) {
        return(numeric(0))
    }

    pdf <- alpha * theta * ((-x) ** (theta-1)) * exp(-alpha * ((-x) ** theta))

    return(pdf)
}
```

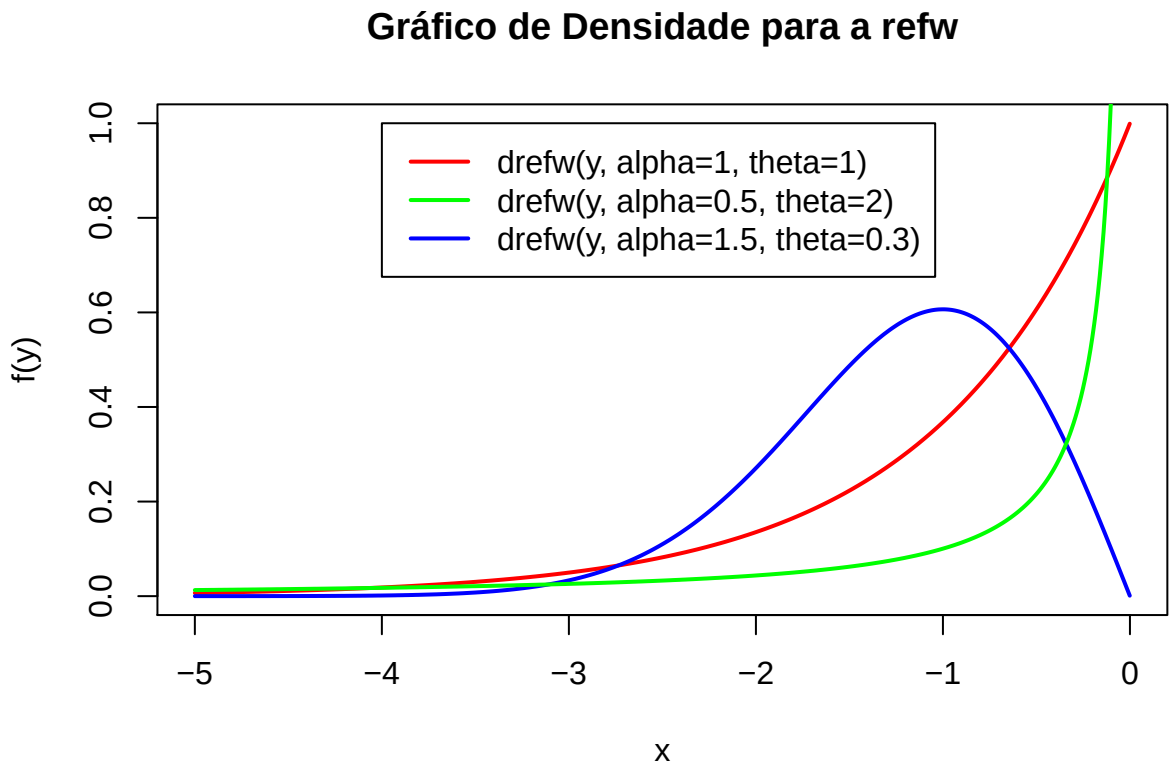
Vamos plotar a função densidade de probabilidade para alguns valores de α e θ.

```r
x <- seq(-5, -0.001, length.out = 500)

plot(x, drefw(x, 1, 1), type='l', col='red', lwd=2,
     xlim=c(-5,0), ylim=c(0,1), ylab='f(y)', main='Gráfico de Densidade para a refw')

lines(x, drefw(x, 0.5, 2), type='l', col='blue', lwd=2)
lines(x, drefw(x, 1.5, 0.3), type='l', col='green', lwd=2)

legend(-4, 1, legend=c("drefw(y, alpha=1, theta=1)",
                       "drefw(y, alpha=0.5, theta=2)",
                       "drefw(y, alpha=1.5, theta=0.3)"),
       col=c("red", "green", "blue"), lty=1, lwd=2)
```



Obtemos assim a figura 12 (pg 39) do artigo citado.

Vamos implementar a função probabilidade acumulada conforme expressão já apresentada.

```r
prefw <- function(q, alpha, theta){
    if (any(alpha <= 0))
        stop(paste("alpha deve ser positivo", "\n", ""))
    if (any(theta <= 0))
        stop(paste("theta deve ser positivo", "\n", ""))
    if (any(q >= 0))
        stop(paste("q deve ser negativo", "\n", ""))

    cdf <- exp(-alpha * ((-q) ** theta))
```

```
    return(cdf)
}
```

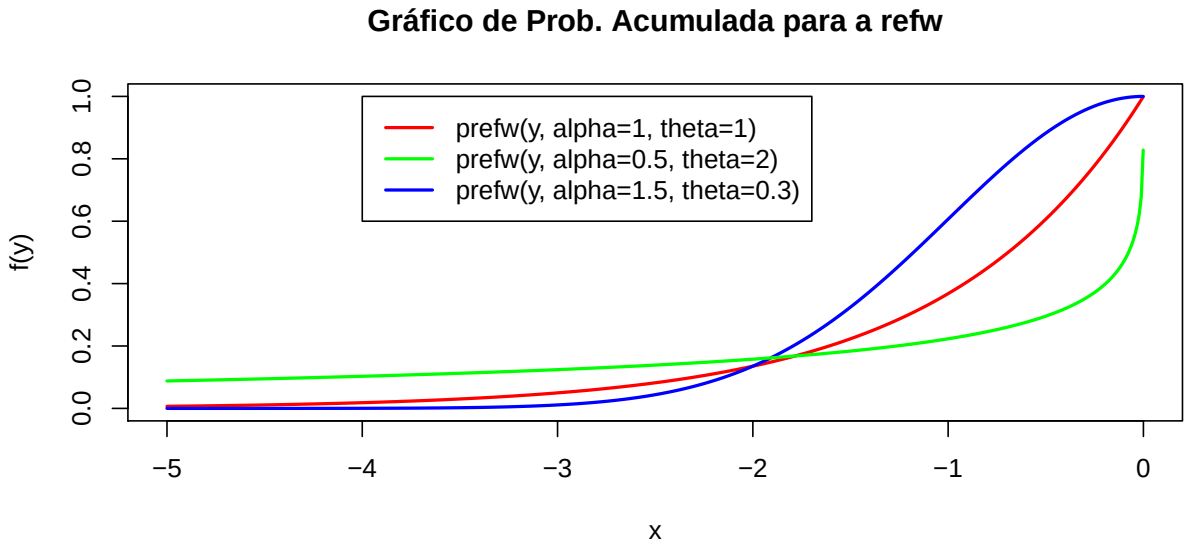
Podemos plotar os gráficos das distribuições de probabilidade acumuladas para alguns valores de α e θ.

```
x <- seq(-5, -0.001, length.out = 500)

plot(x, prefw(x, 1, 1), type='l', col='red', lwd=2,
     main='Gráfico de Prob. Acumulada para a refw',
     xlim=c(-5,0), ylim=c(0,1), ylab='f(y)')

lines(x, prefw(x, 0.5, 2), type='l', col='blue', lwd=2)
lines(x, prefw(x, 1.5, 0.3), type='l', col='green', lwd=2)

legend(-4, 1, legend=c("prefw(y, alpha=1, theta=1)",
                       "prefw(y, alpha=0.5, theta=2)",
                       "prefw(y, alpha=1.5, theta=0.3)"),
       col=c("red", "green", "blue"), lty=1, lwd=2)
```



Agora, vamos implementar a função quantil.

Pela definição de quantil para uma variável contínua temos $\int_{-\infty}^{q_p} f(y)dy = p$, em que q_p é o valor do quantil e depende do p escolhido. No nosso caso, temos $\int_{-\infty}^{q_p} \alpha\theta(-y)^{\theta-1}e^{-\alpha(-y)^\theta}dy = p$. Fazendo a mudança de variável $x = -\alpha(-y)^\theta$ temos $dx = \alpha\theta(-y)^{\theta-1}dy$.

Assim, $\int_{\infty}^{-\alpha(-q_p)^\theta} e^x dx = p$, ou seja, $e^{-\alpha(-q_p)^\theta} = p$. Aplicando o logaritmo natural de ambos os lados temos $-\alpha(-q_p)^\theta = ln(p)$ e isolando q, temos $q = -\left(-\frac{1}{\alpha}ln(p)\right)^{\frac{1}{\theta}}$.

Agora, podemos implementar a função quantil.

```
qrefw <- function(p, alpha, theta, lower.tail=TRUE){
    if (any(theta <= 0))
        stop(paste("theta deve ser positivo", "\n", ""))
```

```
    if (any(alpha <= 0))
        stop(paste("alpha deve ser positivo", "\n", ""))
    if (any(p < 0) | any(p > 1))
        stop(paste("p deve estar entre 0 e 1", "\n", ""))


    if (lower.tail){p <- p}
    else{p <- 1 - p}

    q <- -( ( (-1 / alpha) * log(p) ) ^ (1/theta) )
    return(q)
}
```

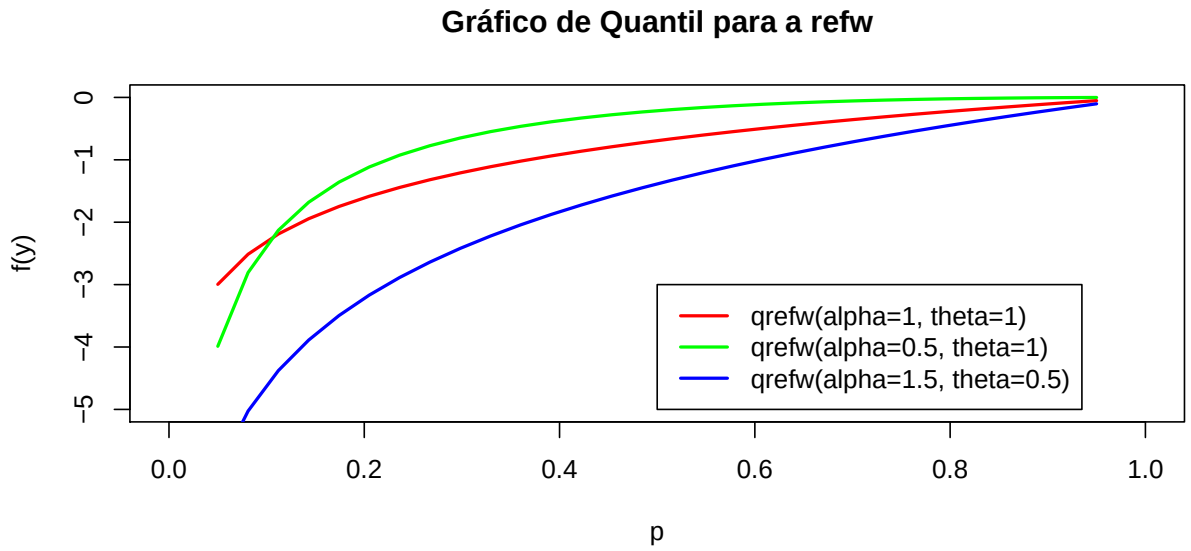
Podemos plotar os gráficos das funções quantílicas para alguns valores de α e θ.

```
p <- seq(0.05, 0.95, length = 30)

plot(p, qrefw(p, 1, 1), type='l', col='red', lwd=2,
     xlim=c(0, 1), ylim=c(-5, 0), ylab='f(y)', main='Gráfico de Quantil para a refw')

lines(p, qrefw(p, 0.5, 1), type='l', col='blue', lwd=2)
lines(p, qrefw(p, 1.5, 0.5), type='l', col='green', lwd=2)

legend(0.5, -3, legend=c("qrefw(alpha=1, theta=1)",
                         "qrefw(alpha=0.5, theta=1)",
                         "qrefw(alpha=1.5, theta=0.5)"),
       col=c("red", "green", "blue"), lty=1, lwd=2)
```



Por fim, vamos implementar uma função geradora de valores aleatórios conforme a distribuição que estamos considerando.

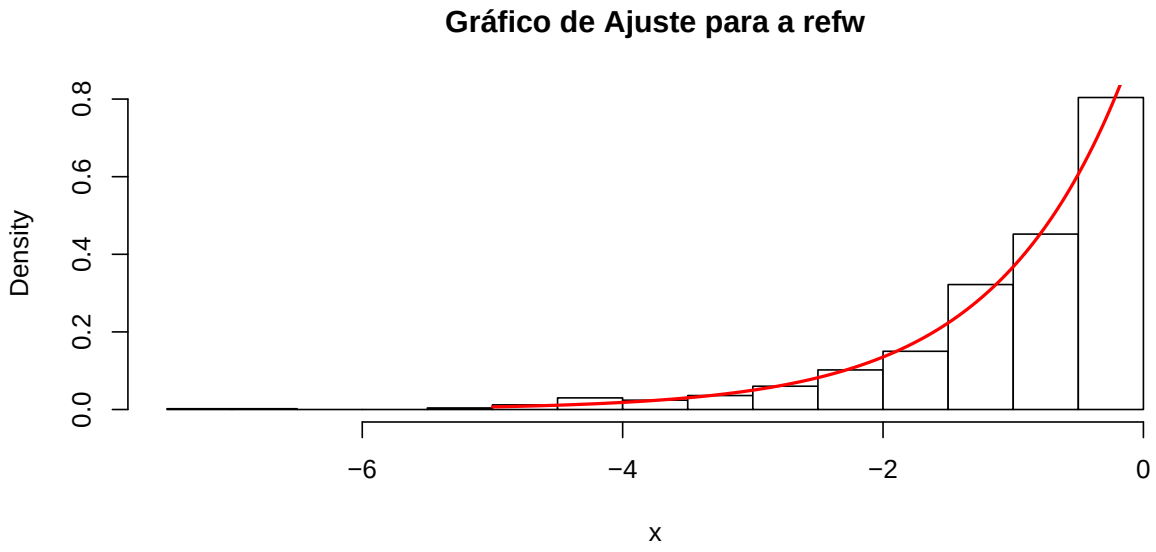
```r
rrefw <- function(n, alpha, theta){
  if (any(alpha <= 0))
    stop(paste("alpha deve ser positivo", "\n", ""))
  if (any(theta <= 0))
    stop(paste("theta deve ser positivo", "\n", ""))

  qrefw(runif(n), alpha, theta)
}
```

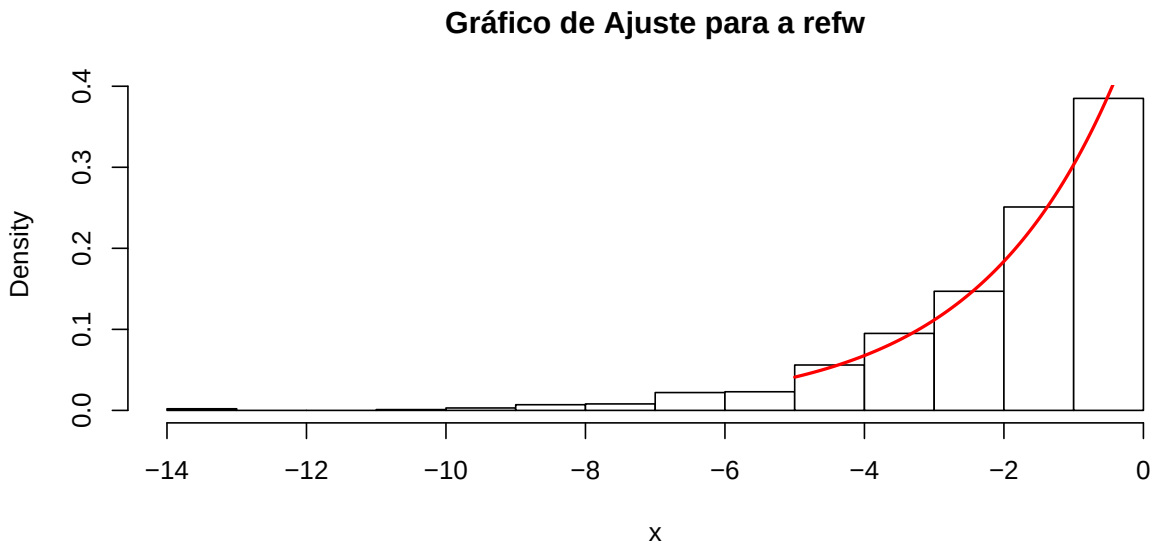
Podemos plotar alguns gráficos para verificar se a função `rrefw` está de acordo com o esperado.

```r
x <- seq(-5, -0.1, length.out = 1000)

vec <- rrefw(1000, 1, 1)
hist(vec, prob=TRUE, xlab='x', col='white', main='Gráfico de Ajuste para a refw')
lines(x, drefw(x, 1, 1), lwd = 2, col = "red")
```




```r
vec2 <- rrefw(1000, 0.5, 1)
hist(vec2, prob=TRUE, xlab='x', col='white', main='Gráfico de Ajuste para a refw')
lines(x, drefw(x, 0.5, 1), lwd=2, col='red')
```




```
vec3 <- rrefw(1000, 1.5, 0.5)
hist(vec3, prob=TRUE, xlab='x', col='white', main='Gráfico de Ajuste para a refw')
lines(x, drefw(x, 1.5, 0.5), lwd=2, col='red')
```

Agora que temos as funções relacionadas à distribuição podemos fazer a simulação de Monte Carlo.

Simulação Monte Carlo

Para fazer essa simulação usaremos o pacote `fitdistrplus`. Vamos criar o estimador de máxima verossimilhança da forma como feito em aula.

```r
library(fitdistrplus)

# Estimador de Máxima Verossimilhança
emv.reflected.weibull <- function(x, par)
{
    fit <- try(fitdist(x,
                       distr='refw',
                       start = list(alpha=par[[1]], theta=par[[2]]))$estimate,
               silent = TRUE)
    if( !is.numeric(fit) ){fit <- NA}
    return(fit)
}
```

Agora vamos definir uma semente e criar os objetos que receberam os valores gerados na simulação.

```r
set.seed(666) # Definir a semente para resultados reprodutíveis

# Definir os valores paramétricos para o parâmetro alpha
alphas <- c(0.5, 1, 1.5, 2)

# Definir os valores paramétricos para o parâmetro theta
thetas <- c(0.5, 1, 1.5)

# Cenários
param  <- expand.grid(alphas, thetas)


B    <- 1000 # Número de simulações
nmax <- 100 # Tamanho de amostra máximo
enes <- seq(10, nmax, 10) # Tamanhos de amostras


# Viés do estimador do parâmetro alpha
vies.alpha <- matrix(nrow = length(enes), ncol = nrow(param))

# Viés do estimador do parâmetro theta
vies.theta <- vies.alpha

# Erro-quadrático-médio do estimador do parâmetro alpha
eqm.alpha <- matrix(nrow = length(enes), ncol = nrow(param))

# Erro-quadrático-médio do estimador do parâmetro theta
eqm.theta <- eqm.alpha
```

Agora que definimos todos os objetos que receberão os dados da simulação podemos realizar a simulação em si. Conforme feito em aula, para cada cenário vamos gerar valores baseado nos valores de α e θ para cada tamanho de amostra, para isso usaremos dois laços de repetição e a família apply.

```r
set.seed(666)
colunas <- c()

for(i in 1:nrow(param))
{
```

```
    k <- 1
    X <- rrefw(nmax * B, alpha = param[i,1], theta = param[i,2])
    X <- matrix(X, ncol = B, nrow = nmax)


    for(n in enes)
    {
        x   <- data.frame(X[1:n,])
        fit <- sapply(x, emv.reflected.weibull, par = param[i,])

        vies.alpha[k,i] <- mean(fit[][1,] - param[i,1], na.rm = TRUE)
        vies.theta[k,i] <- mean(fit[][2,] - param[i,2], na.rm = TRUE)

        eqm.alpha[k,i] <- mean((fit[][1,] - param[i,1])^2, na.rm = TRUE)
        eqm.theta[k,i] <- mean((fit[][2,] - param[i,2])^2, na.rm = TRUE)

        k <- k + 1

    }
    colunas <- append(colunas, paste('a=',param[i,1],' t=',param[i,2], sep=""))
}
```

Podemos obter a matriz com o resultado dos valores de viés para cada valor de α e θ em cada simulação. A seguir podemos ver os números nessa matriz.

```
rownames(vies.alpha) <- paste("amostra=", enes, sep="")
colnames(vies.alpha) <- colunas
vies.alpha
```

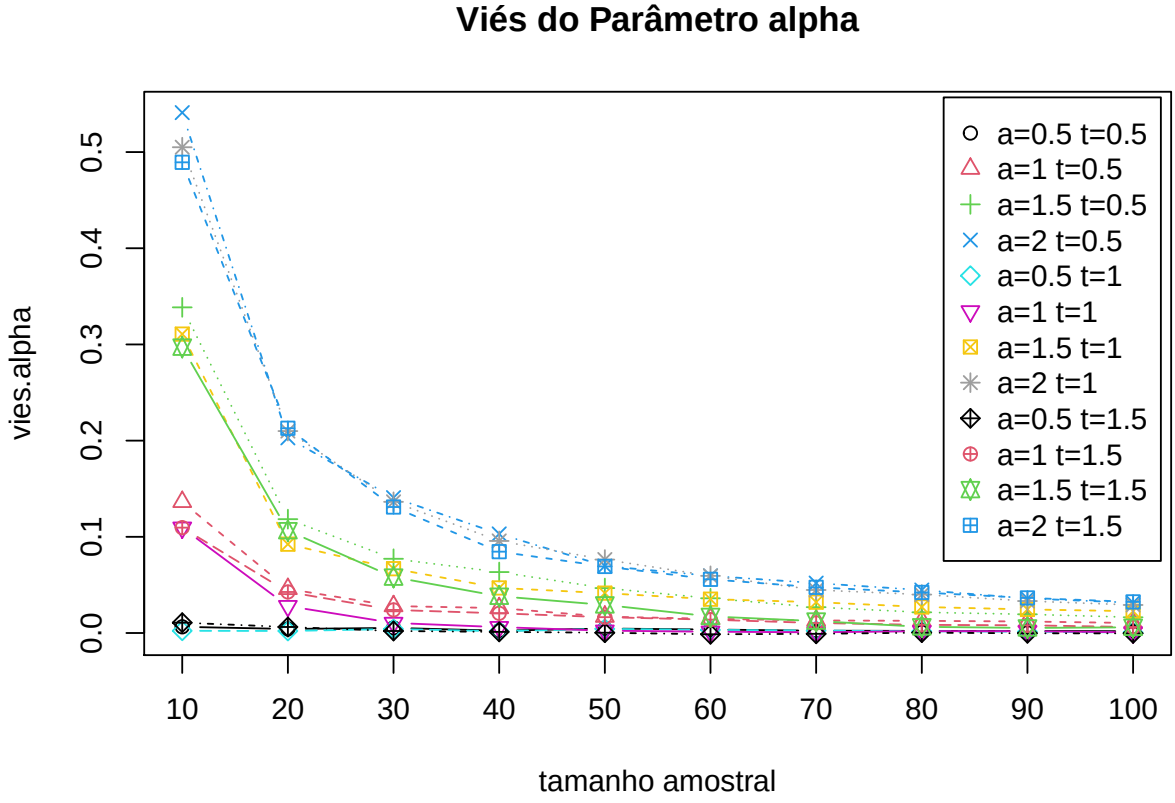
```
##             a=0.5 t=0.5  a=1 t=0.5 a=1.5 t=0.5  a=2 t=0.5    a=0.5 t=1
## amostra=10  0.006440390 0.13634404  0.33857008 0.54102026 0.0022892210
## amostra=20  0.004192266 0.04585127  0.11834257 0.20285146 0.0020922742
## amostra=30  0.005290555 0.02798600  0.07715491 0.14070592 0.0041427698
## amostra=40  0.002394635 0.02591884  0.06336648 0.10303475 0.0018079970
## amostra=50  0.004642947 0.01698415  0.04671049 0.06970107 0.0043005926
## amostra=60  0.003605686 0.01479337  0.03608794 0.05939805 0.0032228883
## amostra=70  0.002392111 0.01304471  0.02684960 0.05186717 0.0025825455
## amostra=80  0.002185899 0.01264669  0.02152669 0.04466082 0.0017303809
## amostra=90  0.002059039 0.01194747  0.01945214 0.03589930 0.0013448239
## amostra=100 0.001883443 0.01036278  0.01633580 0.03151880 0.0005623674
##                  a=1 t=1  a=1.5 t=1    a=2 t=1   a=0.5 t=1.5   a=1 t=1.5
## amostra=10  0.1094849972 0.31075194 0.50502569  0.0107436675 0.109547888
## amostra=20  0.0279695918 0.09234310 0.20995297  0.0062523955 0.042662741
## amostra=30  0.0104510604 0.06686679 0.13637675  0.0020840270 0.023985990
## amostra=40  0.0061242542 0.04686508 0.09571842  0.0011499100 0.020462182
## amostra=50  0.0024123663 0.04145977 0.07635548  0.0002779487 0.016478887
## amostra=60  0.0013298378 0.03516161 0.05942295 -0.0013224312 0.013875628
## amostra=70  0.0007096614 0.03206247 0.04471208 -0.0008522574 0.010420352
## amostra=80  0.0018682624 0.02706743 0.03998366  0.0003718691 0.008458457
## amostra=90  0.0018547122 0.02452253 0.03324611 -0.0003053167 0.008007588
## amostra=100 0.0006658815 0.02263828 0.02914939 -0.0003917278 0.006190870
##             a=1.5 t=1.5  a=2 t=1.5
## amostra=10  0.297194652 0.48933083
```

```
## amostra=20  0.106359103 0.21287882
## amostra=30  0.058098970 0.13097270
## amostra=40  0.038217113 0.08456386
## amostra=50  0.029206532 0.06919311
## amostra=60  0.017239574 0.05580643
## amostra=70  0.012400376 0.04749452
## amostra=80  0.006410777 0.04215811
## amostra=90  0.005262748 0.03648486
## amostra=100 0.005933306 0.03237207
```

Vamos plotar os gráficos do viés de cada simulação para as variáveis α e θ, bem como os gráficos dos erros quadráticos médios para cada parâmetro.

Começando pelo gráfico do viés para o parâmetro α.

```
matplot(vies.alpha, type='b', col=c(1:12), pch=c(1:12), xaxt = "n",
        main='Viés do Parâmetro alpha', xlab='tamanho amostral')
axis(1, at=c(1:length(enes)), labels=enes)
legend("topright",
       inset=0.01,
       legend=colunas,
       col=c(1:12),
       pch=1:12,
       bg= ("white"),
       horiz=F
)
```



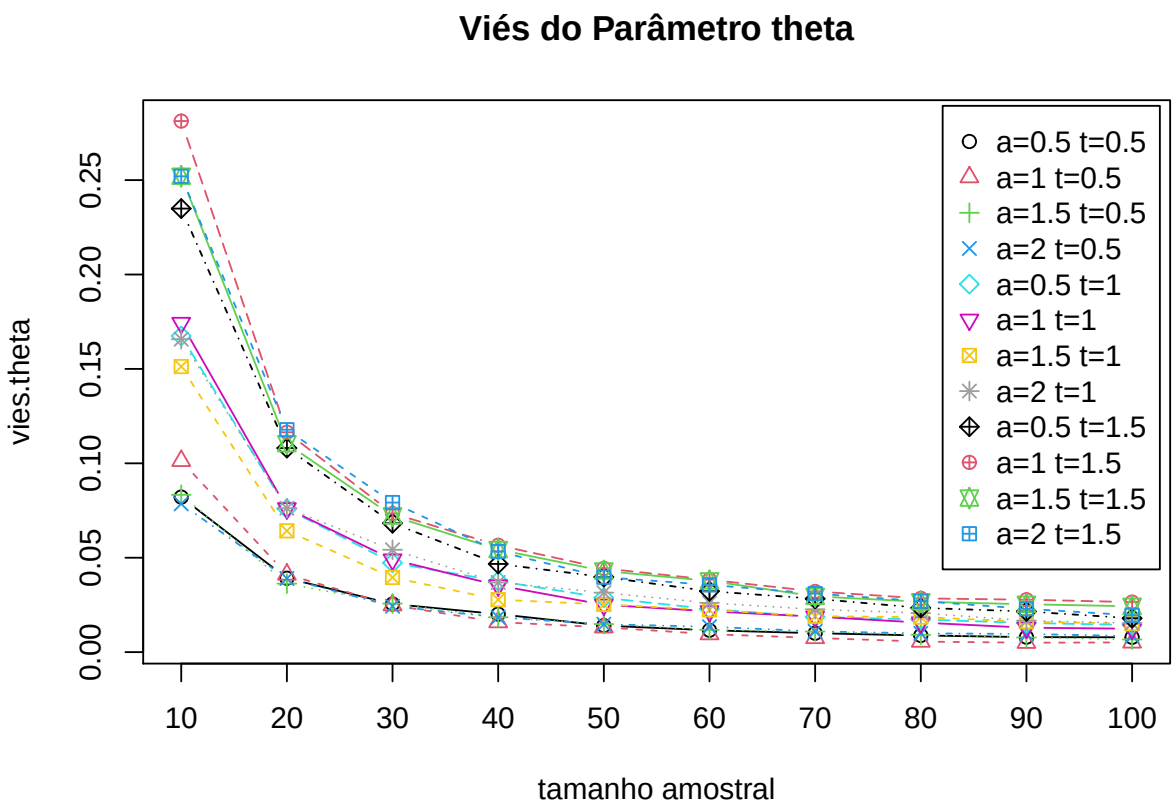
Agora vamos obter a matriz com o resultado dos valores de viés de θ para cada valor de α e θ em cada simulação. A seguir podemos ver os números nessa matriz.

```
rownames(vies.theta) <- paste("amostra=", enes, sep="")
colnames(vies.theta) <- colunas
vies.theta
```

```
##             a=0.5 t=0.5   a=1 t=0.5 a=1.5 t=0.5   a=2 t=0.5  a=0.5 t=1
## amostra=10  0.082112755 0.101304064 0.083316007 0.078534187 0.16742849
## amostra=20  0.039135444 0.041103974 0.036277940 0.039008123 0.07596260
## amostra=30  0.025487789 0.024860273 0.025706982 0.024307930 0.04729335
## amostra=40  0.020131173 0.015891785 0.018449893 0.018455623 0.03764418
## amostra=50  0.014048435 0.013146908 0.014868521 0.014771355 0.02877789
## amostra=60  0.011583308 0.009530731 0.011792913 0.013465347 0.02282244
## amostra=70  0.010008033 0.007508468 0.010917667 0.010998247 0.01864145
## amostra=80  0.008831833 0.005594926 0.009183702 0.009946885 0.01727206
## amostra=90  0.007943965 0.004982258 0.007752574 0.009604757 0.01547429
## amostra=100 0.007859227 0.005139470 0.006673308 0.008515288 0.01435301
##                a=1 t=1  a=1.5 t=1    a=2 t=1 a=0.5 t=1.5  a=1 t=1.5 a=1.5 t=1.5
## amostra=10  0.17432406 0.15125987 0.16573425  0.23492109 0.28127740  0.25206160
## amostra=20  0.07615328 0.06423750 0.07622903  0.10807311 0.11645808  0.11042629
## amostra=30  0.04901186 0.03954955 0.05420227  0.06825961 0.07347948  0.07214299
## amostra=40  0.03532560 0.02777458 0.03687690  0.04671250 0.05659085  0.05432023
## amostra=50  0.02508260 0.02527923 0.03148623  0.03977347 0.04435395  0.04298434
## amostra=60  0.02163500 0.02240642 0.02608441  0.03227501 0.03836287  0.03775014
## amostra=70  0.01879332 0.01900006 0.02265638  0.02826924 0.03213446  0.02955731
## amostra=80  0.01564843 0.01854623 0.02070325  0.02344139 0.02846964  0.02667646
## amostra=90  0.01295306 0.01612941 0.01666062  0.02154929 0.02776619  0.02534741
## amostra=100 0.01239917 0.01509515 0.01541005  0.01793022 0.02650892  0.02426869
##              a=2 t=1.5
## amostra=10  0.25201397
## amostra=20  0.11780081
## amostra=30  0.07922077
## amostra=40  0.05329234
## amostra=50  0.03967222
## amostra=60  0.03564448
## amostra=70  0.03097805
## amostra=80  0.02700295
## amostra=90  0.02295082
## amostra=100 0.01951474
```

Agora vamos plotar o viés para o parâmetro θ.

```
matplot(vies.theta, type = 'b', col=c(1:12), pch=c(1:12), xaxt = "n",
        main='Viés do Parâmetro theta', xlab='tamanho amostral')
axis(1, at=c(1:length(enes)), labels=enes)
legend("topright",
       inset=0.01,
       legend=colunas,
       col=c(1:12),
       pch=1:12,
       bg= ("white"),
       horiz=F
)
```



Podemos observar, do estudo de simulação, que o viés para os parâmetros α e θ é relativamente alto para tamanho amostral menor que 30, o que nos impõe muita cautela quando pretendemos afirmar, em dados reais, que uma distribuição segue o modelo Reflected Weibull se houver poucas observações. Também podemos ver que o viés diminui, tendendo para zero, com o aumento do tamanho amostral e acaba estabilizando próximo de zero.

Podemos obter a matriz com o resultado dos valores de erro quadrático médio para o parâmetro α para cada valor de α e θ em cada simulação. A seguir podemos ver os números nessa matriz.

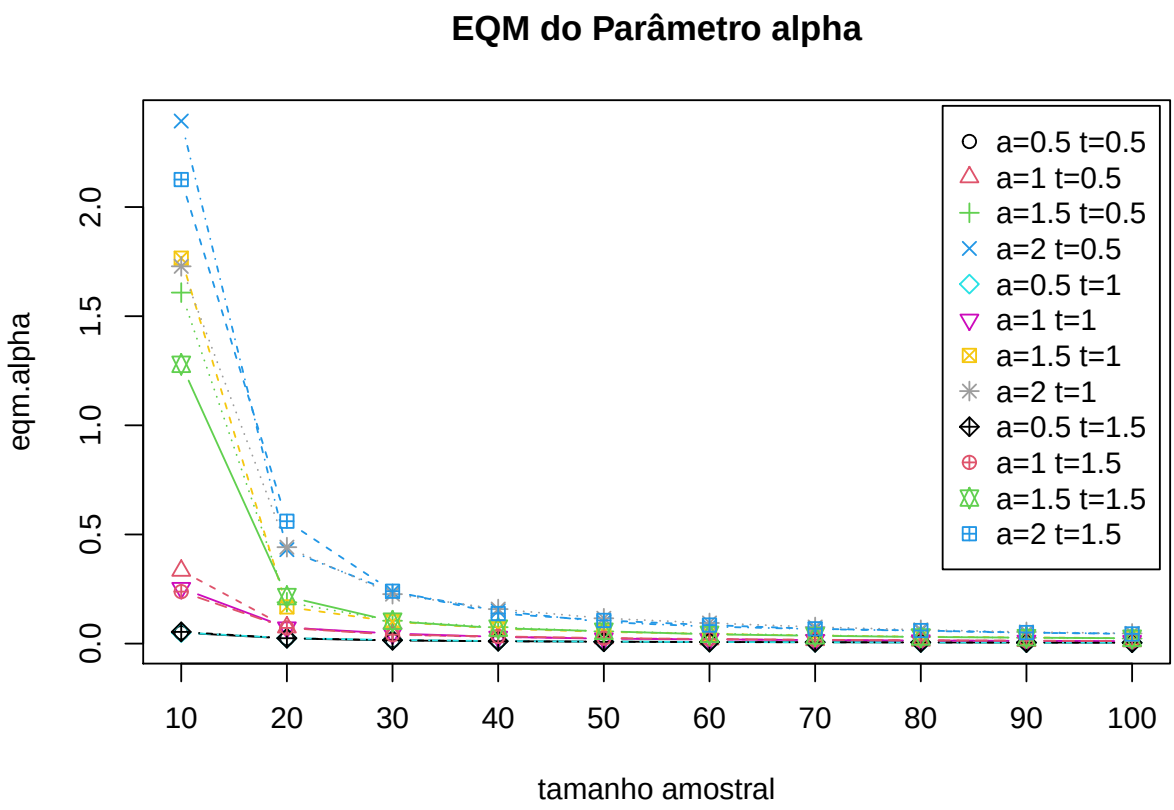
```
rownames(eqm.alpha) <- paste("amostra=", enes, sep="")
colnames(eqm.alpha) <- colunas
eqm.alpha
```

```
##             a=0.5 t=0.5  a=1 t=0.5 a=1.5 t=0.5  a=2 t=0.5   a=0.5 t=1
## amostra=10  0.049931020 0.33442367  1.60805100 2.39394575 0.050706486
## amostra=20  0.023831798 0.07464995  0.18931909 0.43057827 0.025056624
## amostra=30  0.014484010 0.04085591  0.10476641 0.23782846 0.016728736
## amostra=40  0.011011996 0.03024543  0.07529278 0.15014270 0.011663946
## amostra=50  0.009075435 0.02483575  0.05593547 0.09887503 0.009162846
## amostra=60  0.007621721 0.01999222  0.04590707 0.07799003 0.007416679
## amostra=70  0.006337257 0.01658749  0.03861183 0.06649178 0.006231237
## amostra=80  0.005434867 0.01423622  0.03088278 0.05894823 0.005135947
## amostra=90  0.004913957 0.01238161  0.02680848 0.04922036 0.004810861
## amostra=100 0.004449805 0.01099388  0.02325629 0.04643684 0.004256257
##                a=1 t=1  a=1.5 t=1    a=2 t=1 a=0.5 t=1.5  a=1 t=1.5 a=1.5 t=1.5
## amostra=10  0.25426979 1.76567988 1.72843553 0.053700677 0.23806420  1.27953099
```

```
## amostra=20  0.07339807 0.16818241 0.44174990 0.024671450 0.07189478  0.21532939
## amostra=30  0.04634263 0.10054287 0.22655815 0.015495101 0.04085787  0.10272413
## amostra=40  0.03085458 0.06823090 0.15811886 0.010811964 0.03296405  0.07071823
## amostra=50  0.02292021 0.05384244 0.11603243 0.008332386 0.02516723  0.05650542
## amostra=60  0.01920821 0.04406896 0.09439642 0.006992863 0.02098095  0.04275931
## amostra=70  0.01624945 0.03689474 0.07598087 0.006125847 0.01750085  0.03587489
## amostra=80  0.01419681 0.03160681 0.06318998 0.005484251 0.01508390  0.03047147
## amostra=90  0.01262616 0.02816994 0.05334002 0.004673056 0.01340732  0.02715467
## amostra=100 0.01134604 0.02474951 0.04651053 0.004091322 0.01124812  0.02452012
##              a=2 t=1.5
## amostra=10  2.12592271
## amostra=20  0.56072945
## amostra=30  0.24003613
## amostra=40  0.13788323
## amostra=50  0.10628255
## amostra=60  0.08500898
## amostra=70  0.06775077
## amostra=80  0.05987723
## amostra=90  0.05064505
## amostra=100 0.04433300
```

Podemos também plotar o erro quadrático médio para o parâmetro α.

```
matplot(eqm.alpha, type = 'b', col=c(1:12), pch=c(1:12), xaxt = "n",
        main='EQM do Parâmetro alpha', xlab='tamanho amostral')
axis(1, at=c(1:length(enes)), labels=enes)
legend("topright",
       inset=0.01,
       legend=colunas,
       col=c(1:12),
       pch=1:12,
       bg= ("white"),
       horiz=F
)
```



Podemos obter a matriz com o resultado dos valores de erro quadrático médio para o parâmetro θ para cada valor de α e θ em cada simulação. A seguir podemos ver os números nessa matriz.

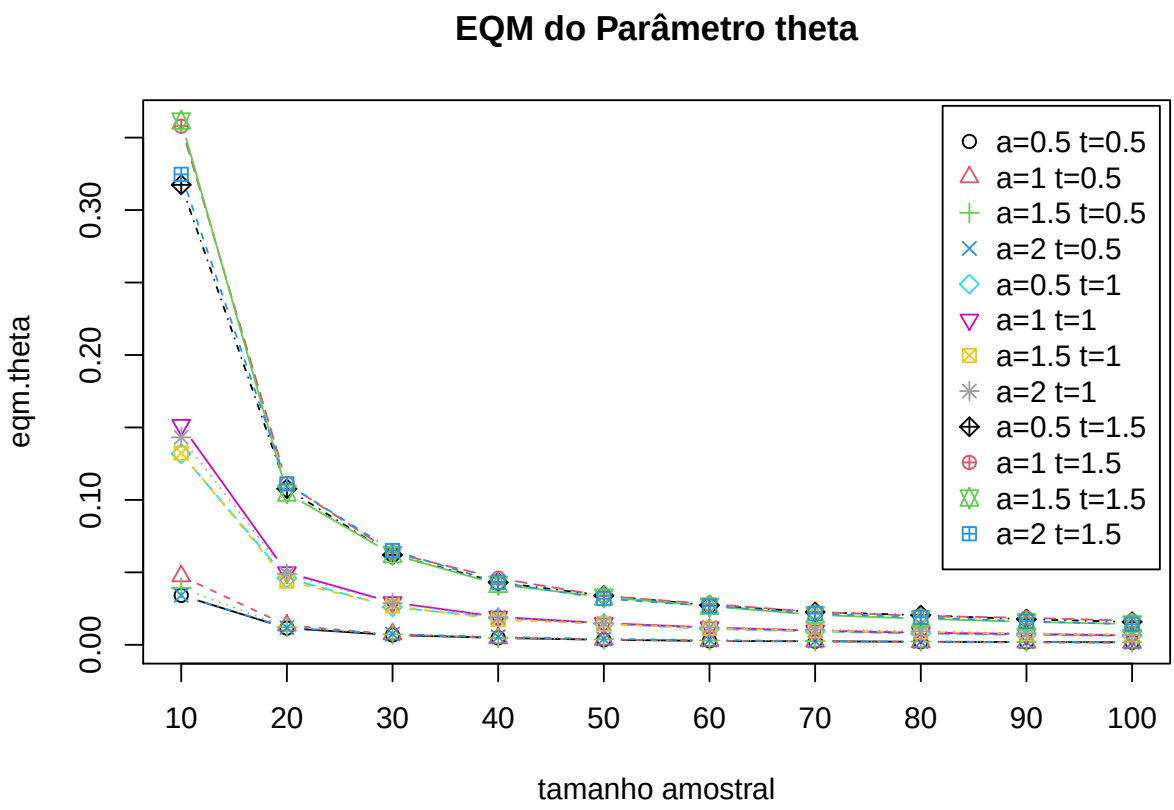
```
rownames(eqm.theta) <- paste("amostra=", enes, sep="")
colnames(eqm.theta) <- colunas
eqm.theta
```

```
##             a=0.5 t=0.5   a=1 t=0.5 a=1.5 t=0.5   a=2 t=0.5   a=0.5 t=1
## amostra=10  0.034121377 0.047388215 0.039213699 0.034200441 0.131825214
## amostra=20  0.011524445 0.013421892 0.012138866 0.011840667 0.045997879
## amostra=30  0.006769023 0.007324823 0.007231345 0.007370951 0.026058957
## amostra=40  0.004766521 0.004643426 0.004713514 0.005501058 0.018620754
## amostra=50  0.003476804 0.003647334 0.003820871 0.004074111 0.014555175
## amostra=60  0.002721607 0.002820345 0.002999974 0.003243840 0.010976541
## amostra=70  0.002429901 0.002278739 0.002598152 0.002622769 0.009169936
## amostra=80  0.002120972 0.001863155 0.002243676 0.002219394 0.007663335
## amostra=90  0.001925399 0.001627431 0.001898992 0.001980064 0.006788608
## amostra=100 0.001711269 0.001448518 0.001654334 0.001722050 0.006129833
##                 a=1 t=1   a=1.5 t=1     a=2 t=1 a=0.5 t=1.5  a=1 t=1.5
## amostra=10  0.151619253 0.132328723 0.143160309  0.31741506 0.35785208
## amostra=20  0.049954333 0.043954171 0.048829889  0.10764910 0.11118938
## amostra=30  0.029285212 0.026831985 0.028650189  0.06204603 0.06352437
## amostra=40  0.019258501 0.017401304 0.018977332  0.04312153 0.04592774
## amostra=50  0.014924704 0.014282257 0.015064211  0.03394178 0.03369533
## amostra=60  0.011979133 0.011503215 0.012073043  0.02725107 0.02816922
```

```
## amostra=70  0.009747924 0.009586115 0.010379690  0.02260322 0.02266926
## amostra=80  0.008357709 0.008867768 0.009349996  0.02044458 0.01997638
## amostra=90  0.007438530 0.007642014 0.008009726  0.01764157 0.01860694
## amostra=100 0.006523252 0.006933795 0.006931457  0.01582952 0.01664797
##             a=1.5 t=1.5  a=2 t=1.5
## amostra=10   0.36142157 0.32464529
## amostra=20   0.10473340 0.11107056
## amostra=30   0.06238095 0.06499814
## amostra=40   0.04173161 0.04315880
## amostra=50   0.03297346 0.03185099
## amostra=60   0.02668643 0.02675611
## amostra=70   0.02064687 0.02154493
## amostra=80   0.01801416 0.01897017
## amostra=90   0.01580998 0.01621317
## amostra=100  0.01438208 0.01421296
```

Agora vamos plotar o erro quadrático médio para o parâmetro θ.

```
matplot(eqm.theta, type = 'b', col=c(1:12), pch=c(1:12), xaxt = "n",
        main='EQM do Parâmetro theta', xlab='tamanho amostral')
axis(1, at=c(1:length(enes)), labels=enes)
legend("topright",
       inset=0.01,
       legend=colunas,
       col=c(1:12),
       pch=1:12,
       bg= ("white"),
       horiz=F
)
```



Podemos observar, do estudo de simulação, que o erro quadrático médio para os parâmetros α e θ é relativamente alto para tamanho amostral menor que 20, o que nos impõe muita cautela quando pretendemos afirmar, em dados reais, que uma distribuição segue o modelo Reflected Weibull se houver poucas observações. Também podemos ver o viés diminui com o aumento do tamanho amostral e acaba estabilizando, próximo de zero.

Embora tenham sido realizados apenas 1000 simulações, pode-se notar que os vieses e EQM baixos indicam um bom ajuste, não necessitando o aumento do número de simulações. Ainda assim, se aumentassemos o números de simulações obteríamos valores mais precisos de viés e EQM.

O comportamento do viés e do EQM nos mostral que temos um bom estimador da distribuição.

Aplicação

Vamos usar essa distribuição para ver o ajuste ao conjunto de dados `Leukocyte_Profiles.xlsx` disponibilizado.

```
library(readxl)
dados <- as.data.frame(read_excel("Leukocyte_Profiles.xlsx"))
```

Para estimar o melhores valores de α e θ que ajustam os dados da coluna `H/L ratio` vamos usar a função de máxima verossimilhança criada anteriormente. Neste caso todos os valores são positivos e a distribuição Reflected Weibull não tem como cobrir estes dados, por isso vamos multiplicá-los por (-1) para verificar o ajuste.

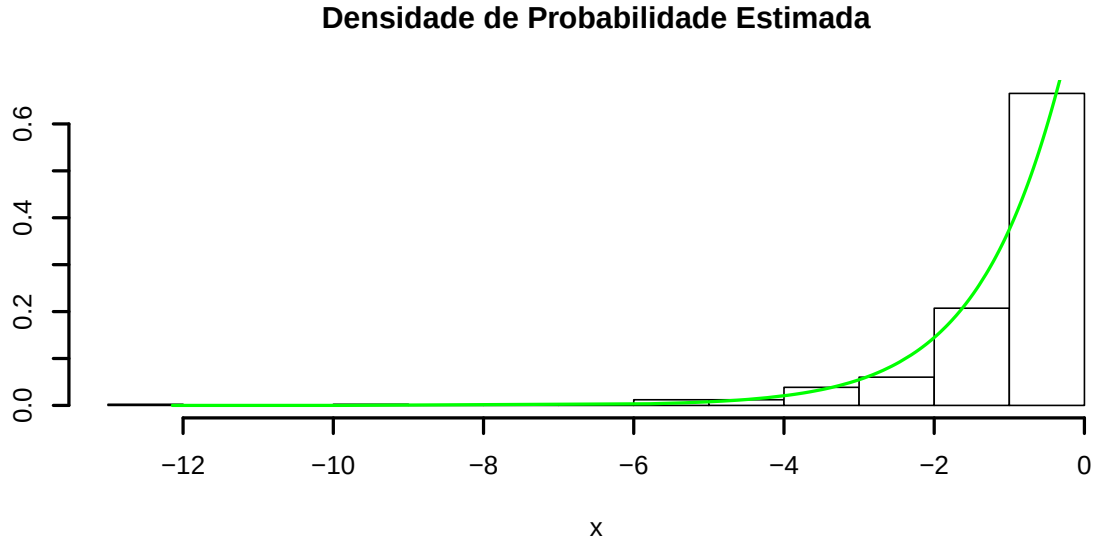
```r
x <- dados$`H/L ratio`
par.hat <- emv.reflected.weibull(-x, list(1,2))
par.hat
```

```
##     alpha     theta
## 0.9400173 1.0215260
```

Obtemos assim uma α de aproximadamente 0.94 e um θ de aproximadamente 1.02.

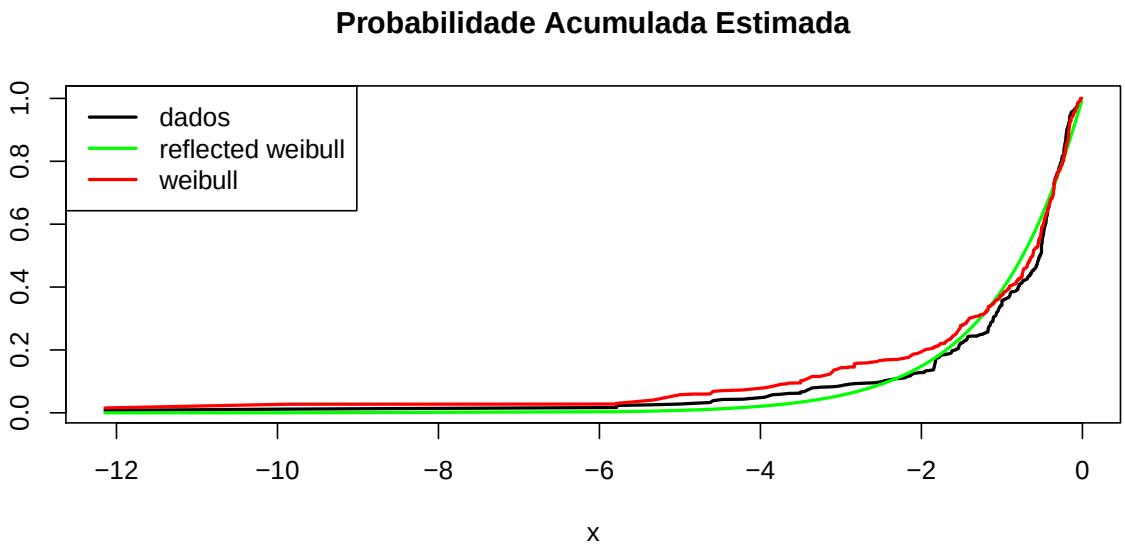
Também podemos plotar a estimativa das função de densidade de probabilidade para os dados.

```r
hist(-dados$`H/L ratio`, prob=TRUE, ylab='', xlab='x', lwd=2, col='white',
     main='Densidade de Probabilidade Estimada')
lines(sort(-dados$`H/L ratio`),
      drefw(sort(-dados$`H/L ratio`), alpha=par.hat[1], theta=par.hat[2]),
      type='l', col='green', lwd=2)
```



Bem como a função de probabilidade acumulada para os dados.

```r
soma <- cumsum(dados$`H/L ratio`) / sum(dados$`H/L ratio`)
plot(sort(-dados$`H/L ratio`), soma, type='l', lwd=2,
     main='Probabilidade Acumulada Estimada', ylab='', xlab='x')
lines(sort(-dados$`H/L ratio`), prefw(sort(-dados$`H/L ratio`), par.hat[1], par.hat[2]),
      type='l', col='green', lwd=2)
lines(sort(-dados$`H/L ratio`), pweibull(sort(dados$`H/L ratio`), par.hat[1], par.hat[2]),
      type='l', col='red', lwd=2)
legend("topleft", c("dados", "reflected weibull", "weibull"),
       col=c("black", "green", "red"), lwd=2)
```



Conclusão

Embora muitas funções de probabilidade conhecidas cumpram um bom papel para modelar adequadamente diversos conjuntos de dados, pode ser preciso obter uma nova distribuição a fim de obter um melhor ajuste.

Referências

- Almalki, Saad J. & Nadarajah, Saralees: Modifications of the Weibull distribution: A review. Reliability Engineering and System Safety. (2013)
- Aulas de Estatística Computacional II do prof. Ricardo Puziol de Oliveira