Modeling extreme values with a GEV mixture probability distributions

Pascal Alain Dkengne Sielenou

September 28th, 2023

```r
# library(xfun)
```

```r
path <- ".."
```

```r
xfun::in_dir(dir = path, expr = source("./src/generate_gev_sample.R"))
xfun::in_dir(dir = path, expr = source("./src/calculate_gev_inverse_cdf.R"))
xfun::in_dir(dir = path, expr = source("./src/estimate_gev_mixture_model_parameters.R"))
xfun::in_dir(dir = path, expr = source("./src/plot_gev_mixture_model_pdf.R"))
xfun::in_dir(dir = path, expr = source("./src/plot_several_standardized_block_maxima_mean.R"))
xfun::in_dir(dir = path, expr = source("./src/estimate_gev_mixture_model_quantile.R"))
```

```r
n <- 100000
```

```r
loc <- 1
scale <- 0.5
shape <- -0.2
set.seed(1122)
x <- generate_gev_sample(n = n, loc = loc, scale = scale, shape = shape)
```

```r
nlargest <- 1000
```

```r
gev_mixture_model <- estimate_gev_mixture_model_parameters(x,
                                                           nsloc = NULL,
                                                           std.err = FALSE,
                                                           block_sizes = NULL,
                                                           minimum_nblocks = 50,
                                                           threshold = NULL,
                                                           nlargest = nlargest,
                                                           confidence_level = 0.95,
                                                           log_mv = TRUE,
                                                           log_pw = TRUE,
                                                           trace = FALSE)
```

```
##  Successful convergence.
##  Successful convergence.
```

```r
names(gev_mixture_model)
```

```
##  [1] "data"
##  [2] "data_largest"
##  [3] "block_sizes"
##  [4] "equivalent_block_sizes"
##  [5] "rejected_block_sizes"
##  [6] "block_maxima_indexes_object"
##  [7] "gev_models_object"
##  [8] "extremal_indexes"
##  [9] "normalized_gev_parameters_object"
## [10] "weighted_normalized_gev_parameters_object"
## [11] "identic_weights_mw"
## [12] "pessimistic_weights_mw"
## [13] "pessimistic_weights_pw_shape"
## [14] "pessimistic_weights_pw_scale"
## [15] "pessimistic_weights_pw_loc"
## [16] "automatic_weights_mw"
## [17] "automatic_weights_mw_statistics"
## [18] "automatic_weights_pw_shape"
## [19] "automatic_weights_pw_scale"
## [20] "automatic_weights_pw_loc"
## [21] "automatic_weights_pw_statistics"
```

```
gev_mixture_model$block_sizes
```

```
##  [1]  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
gev_mixture_model$normalized_gev_parameters_object
```

```
##            loc_star        scale_star         shape_star
## 8  2.60374780842982 0.149370455451573 -0.169075253127190
## 9  2.48361024746670 0.248807657227620 -0.316659597243573
## 10 2.52980641075431 0.203707823276824 -0.252602400506974
## 11 2.52601714904580 0.215731832793612 -0.277602063409150
## 12 2.57831642630518 0.175929782757252 -0.223932114175994
## 13 2.55837465048588 0.185658249983369 -0.232614907814503
## 14 2.48628362959065 0.241838400646719 -0.301948165624440
## 15 2.48393877303052 0.242134456859766 -0.301379229961072
## 16 2.50934752752142 0.224624875005753 -0.283568027452624
## 17 2.50111133482957 0.237272688240628 -0.303693028055625
## 18 2.42122884563705 0.303064536649907 -0.366587580178727
## 19 2.44874043529037 0.262147576758607 -0.316663660267862
## 20 2.39635941053167 0.315292990018172 -0.369693305299051
```

```
gev_mixture_model$weighted_normalized_gev_parameters_object
```

```
##                              loc_star        scale_star         shape_star
## identic_weights     2.50206789607069 0.231198563513062 -0.285847641008983
## pessimistic_weights 2.50530584834054 0.233253648858017 -0.282912345050145
## automatic_weights   2.54084889988313 0.181329949333266 -0.196765266601923
```

```
gev_mixture_model$automatic_weights_mw_statistics
```

```
## $function_value
## [1] 0.00145526780276121
##
## $gradient_value
## [1] 2.06513884906667e-06
##
## $function_reduction
## [1] 0.0155243661588763
##
## $number_iterations
## [1] 1563
##
## $convergence
## [1] 0
##
## $message
## [1] "Successful convergence"
```

```r
gev_mixture_model$automatic_weights_pw_statistics
```

```
## $function_value
## [1] 0.000759289762150079
##
## $gradient_value
## [1] 2.32786305421895e-05
##
## $function_reduction
## [1] 0.0230777494767138
##
## $number_iterations
## [1] 3144
##
## $convergence
## [1] 0
##
## $message
## [1] "Successful convergence"
```

```r
gev_mixture_model$automatic_weights_mw
```

```
##                 8                 9                10                11
## 0.175310446545628 0.000000000000000 0.000000000000000 0.000000000000000
##                12                13                14                15
## 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000
##                16                17                18                19
## 0.000000000000000 0.000000000000000 0.000000000000000 0.824689553454373
##                20
## 0.000000000000000
```

```r
gev_mixture_model$pessimistic_weights_pw_shape
```

```
##                  8                  9                 10                 11
## 0.0863247107594706 0.0744800681928340 0.0794071762610771 0.0774466322006962
##                 12                 13                 14                 15
## 0.0817167526483910 0.0810102944022902 0.0755838760119930 0.0756268906097408
##                 16                 17                 18                 19
## 0.0769859639092991 0.0754521075386969 0.0708527348261090 0.0744797655791226
##                 20
## 0.0706330270602794
```

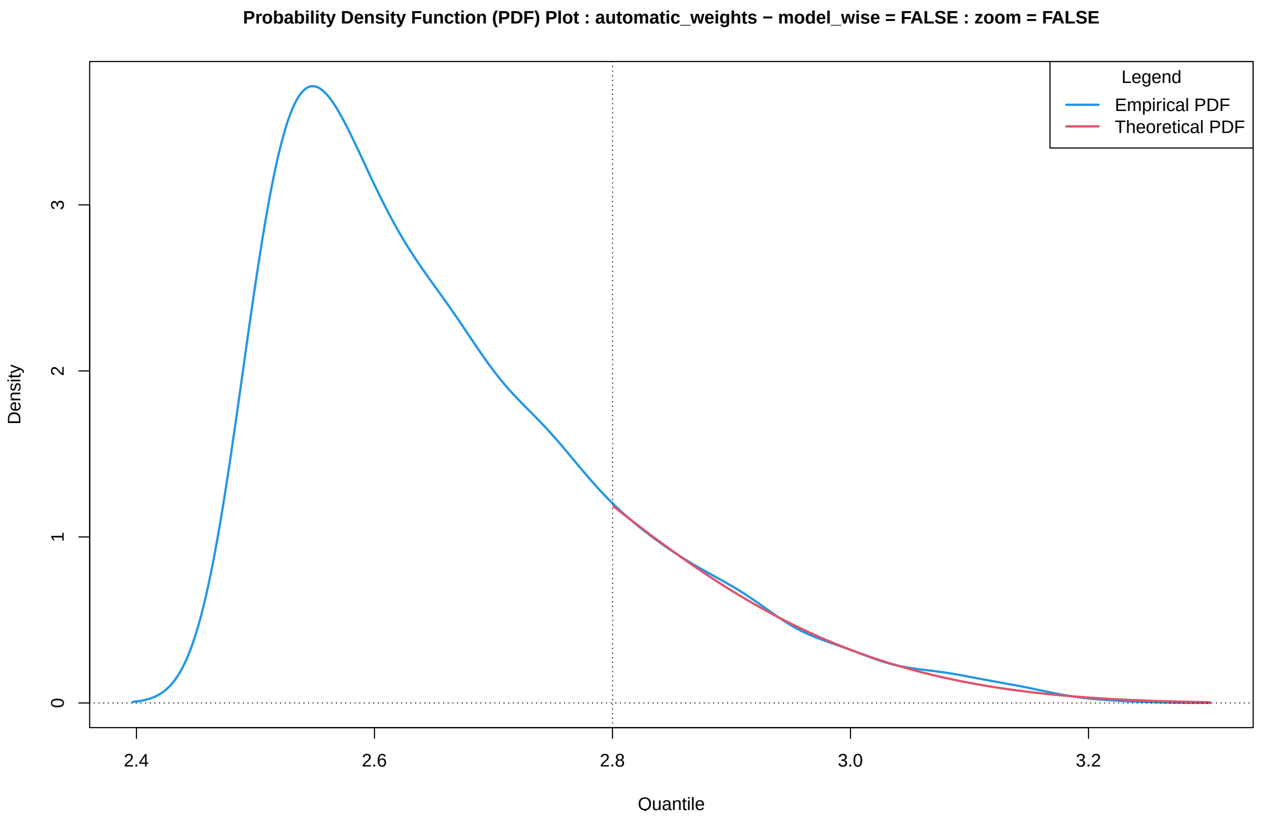
```r
gev_mixture_model$pessimistic_weights_pw_scale
```

```
##                  8                  9                 10                 11
## 0.0708065321447594 0.0782092916973113 0.0747604219177226 0.0756647679795717
##                 12                 13                 14                 15
## 0.0727123019637210 0.0734231332535008 0.0776661260054841 0.0776891229486572
##                 16                 17                 18                 19
## 0.0763406588638038 0.0773123330834036 0.0825699109189437 0.0792595872086683
##                 20
## 0.0835858120144524
```

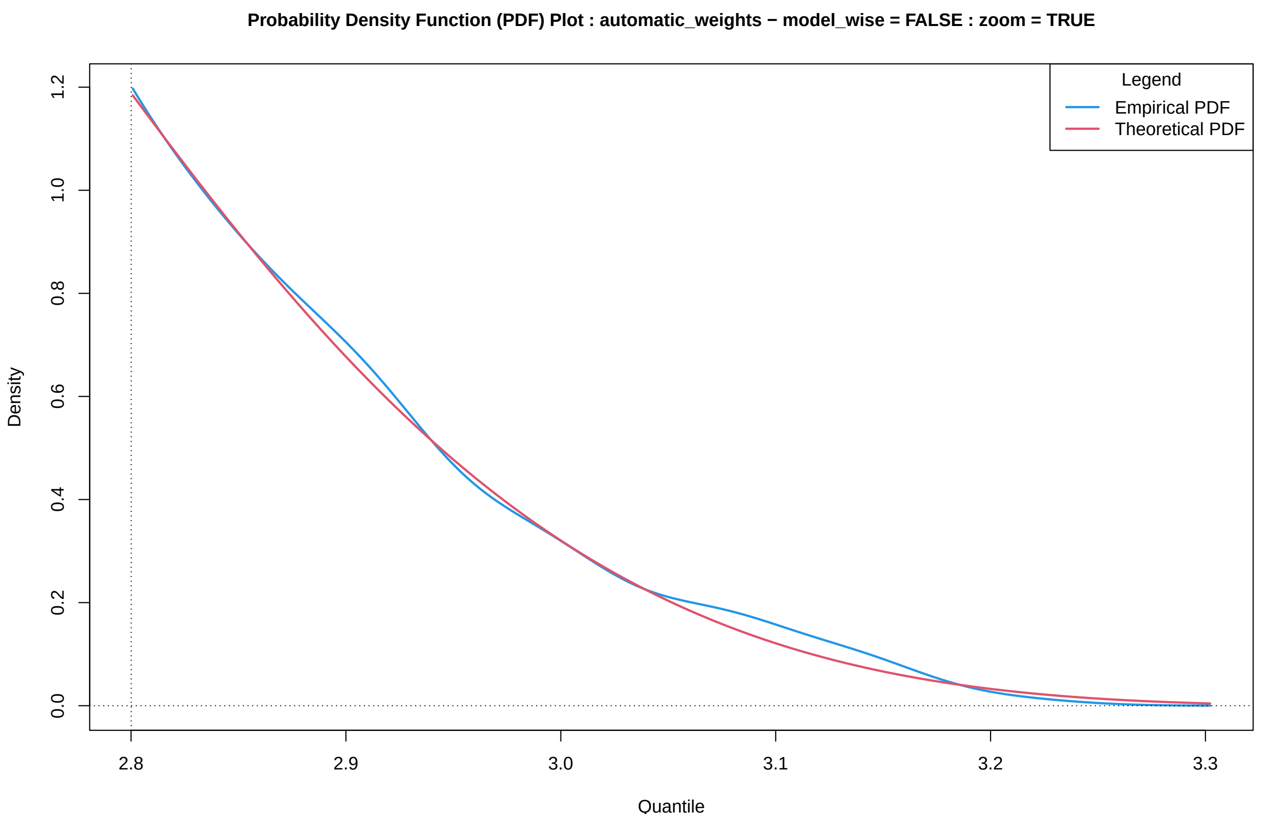
```r
gev_mixture_model$pessimistic_weights_pw_loc
```

```
##                  8                  9                 10                 11
## 0.0850182136246157 0.0753940191695924 0.0789586354781310 0.0786600066927906
##                 12                 13                 14                 15
## 0.0828833443448373 0.0812468745610292 0.0755958458524118 0.0754187921012966
##                 16                 17                 18                 19
## 0.0773596425527028 0.0767251102820384 0.0708345268130596 0.0728103516678096
##                 20
## 0.0690946368596850
```

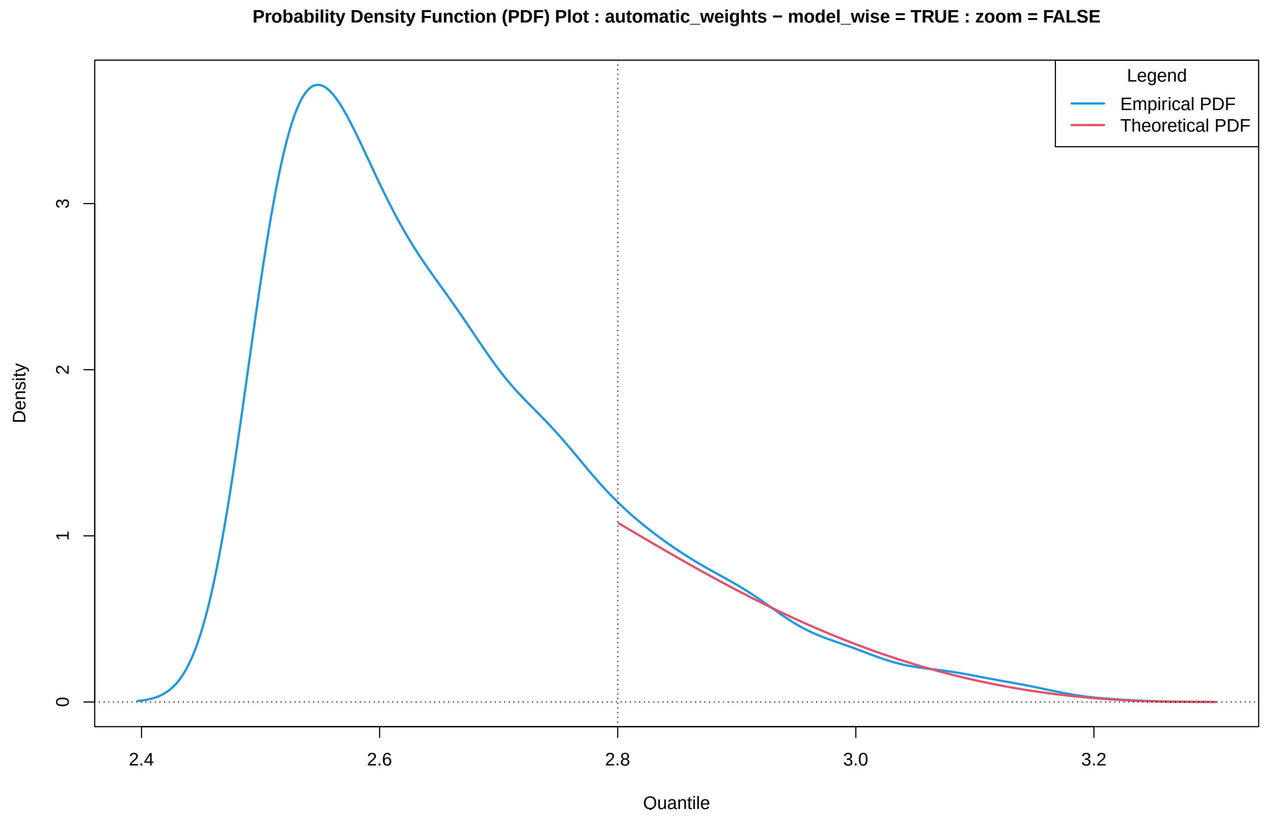
```r
plot_gev_mixture_model_pdf(gev_mixture_model,
                           type = "automatic_weights",
                           model_wise = FALSE,
                           zoom = FALSE,
                           xlab = "Quantile",
                           ylab = "Density",
                           main = "Probability Density Function (PDF) Plot")
```



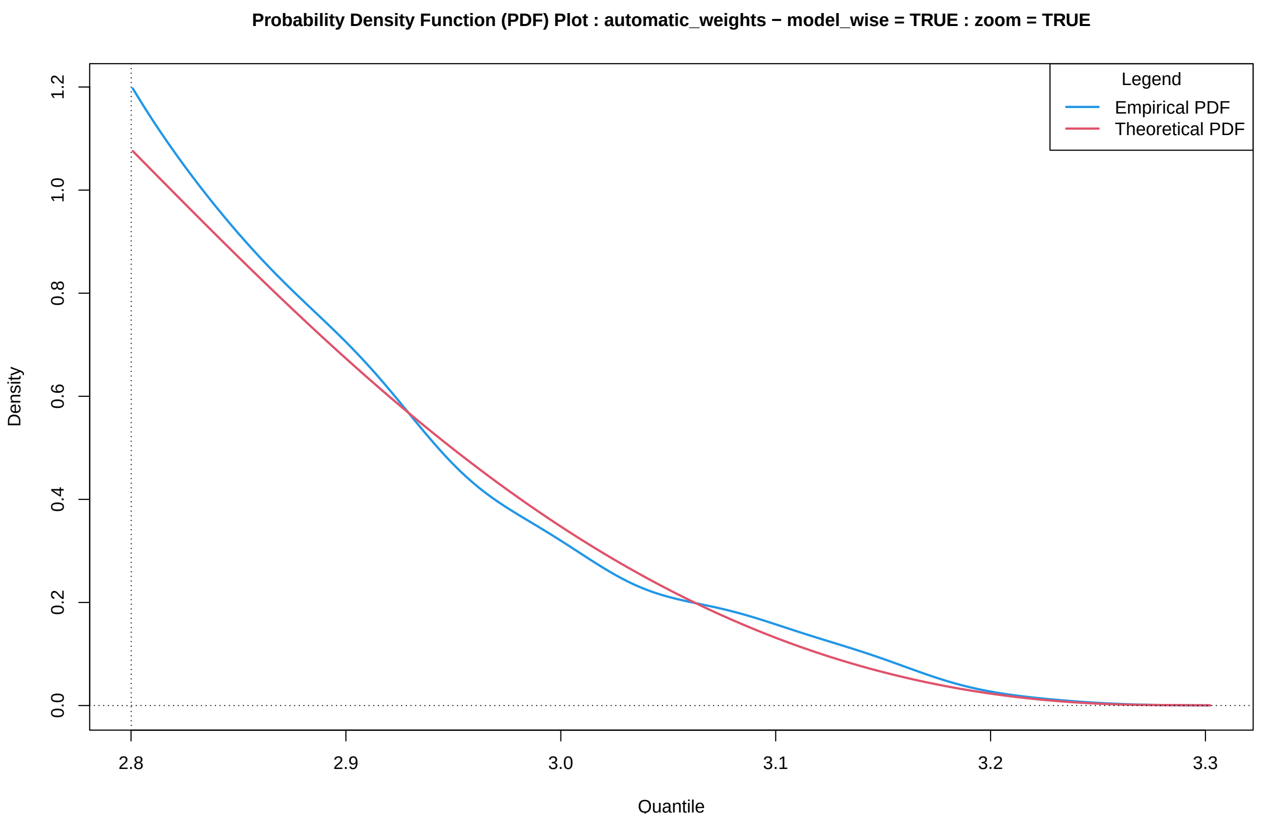
```r
plot_gev_mixture_model_pdf(gev_mixture_model,
                           type = "automatic_weights",
                           model_wise = FALSE,
                           zoom = TRUE,
                           xlab = "Quantile",
                           ylab = "Density",
                           main = "Probability Density Function (PDF) Plot")
```



```r
plot_gev_mixture_model_pdf(gev_mixture_model,
                           type = "automatic_weights",
                           model_wise = TRUE,
                           zoom = FALSE,
                           xlab = "Quantile",
                           ylab = "Density",
                           main = "Probability Density Function (PDF) Plot")
```



```
plot_gev_mixture_model_pdf(gev_mixture_model,
                           type = "automatic_weights",
                           model_wise = TRUE,
                           zoom = TRUE,
                           xlab = "Quantile",
                           ylab = "Density",
                           main = "Probability Density Function (PDF) Plot")
```



```
estimator_types <- c("automatic_weights_mw",
                     "pessimistic_weights_mw",
                     "identic_weights_mw",
                     "automatic_weights_pw",
                     "pessimistic_weights_pw",
                     "identic_weights_pw",
                     "empirical",
                     "confidence_interval_mw",
                     "confidence_interval_pw")
```

```
alpha <- 10^(-14)
```

```
results_mw <- estimate_gev_mixture_model_quantile(gev_mixture_model,
                                                  alpha = alpha,
                                                  confidence_level = 0.95,
                                                  do.ci = TRUE,
                                                  estimator_type = estimator_types[1])

results_mw
```

```
##   lower         estimate upper
## 1    NA 3.47610800056145    NA
```

```
results_pw <- estimate_gev_mixture_model_quantile(gev_mixture_model,
                                                  alpha = alpha,
                                                  confidence_level = 0.95,
                                                  do.ci = TRUE,
                                                  estimator_type = estimator_types[4])
```

```
results_pw
```

```
##   lower         estimate upper
## 1    NA 3.45839179494427    NA
```

```
quantile(x = x, probs = 1 - alpha)
```

```
##             100%
## 3.20626360125057
```

```
true_rl <- calculate_gev_inverse_cdf(p = 1 - alpha, loc = loc, scale = scale, shape = shape)
true_rl
```

```
## [1] 3.49603840060645
```

```
est_rl_pw <- estimate_gev_mixture_model_quantile(gev_mixture_model,
                                                 alpha = alpha,
                                                 confidence_level = 0.95,
                                                 do.ci = TRUE,
                                                 estimator_type = estimator_types[9])

est_rl_pw
```

```
##               lower         estimate            upper
## 8   3.0256095536058 3.47893190542071 3.93225425723562
## 9  3.16868326507378 3.26925085422013 3.36981844336648
## 10 3.10426270387997 3.33556713918782 3.56687157449567
## 11 3.12380273276933 3.30284865902194 3.48189458527455
## 12 3.05106862582189 3.36245361440674 3.67383860299158
## 13 3.04354071311319   3.355314006308 3.66708729950281
## 14 3.11405898337059 3.28710754203475 3.46015610069891
## 15 3.11205921233154 3.28725426856778 3.46244932480402
## 16  3.0777710980143 3.30124890535201 3.52472671268971
## 17 3.10483641927938 3.28229722930349 3.45975803932761
## 18 3.15385980495856 3.24796982366611 3.34207984237365
## 19 3.10912388577637 3.27652078805839 3.44391769034041
## 20 3.14765550062584  3.2492375475842 3.35081959454256
```

```
est_rl_pw_range <- range(as.matrix(est_rl_pw))
est_rl_pw_range
```

```
## [1] 3.02560955360580 3.93225425723562
```

```
est_rl_mw <- estimate_gev_mixture_model_quantile(gev_mixture_model,
                                                 alpha = alpha,
                                                 confidence_level = 0.95,
                                                 do.ci = TRUE,
                                                 estimator_type = estimator_types[8])

est_rl_mw
```

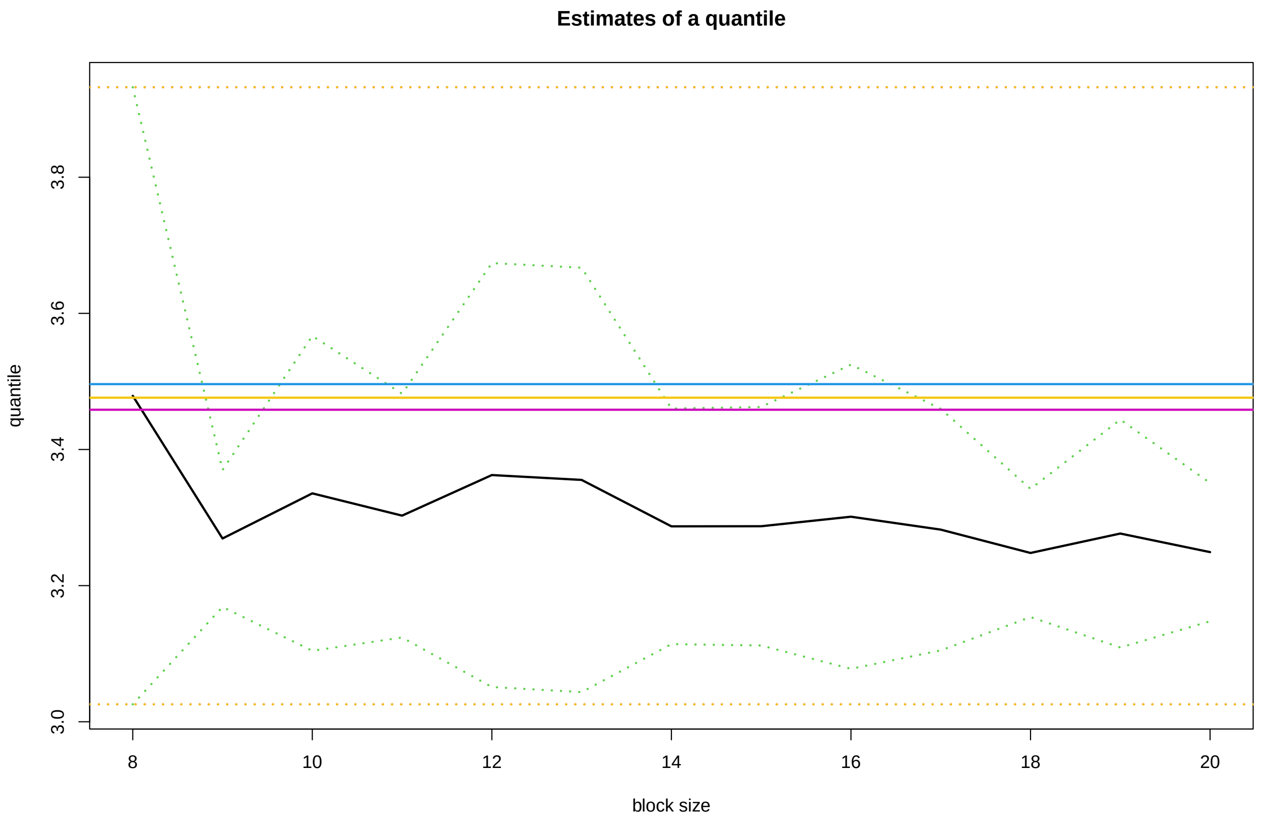
```
##               lower         estimate            upper
## 8   3.0256095536058 3.47893190542071 3.93225425723562
## 19 3.10912388577637 3.27652078805839 3.44391769034041
```

```
est_rl_mw_range <- range(as.matrix(est_rl_mw))
est_rl_mw_range
```

```
## [1] 3.02560955360580 3.93225425723562
```

```
matplot(x = rownames(est_rl_pw),
        y = est_rl_pw,
        xlab = "block size",
        ylab = "quantile",
        main = "Estimates of a quantile",
        cex = 1,
        cex.lab = 1,
        cex.axis = 1,
        type = "l",
        lty = c("dotted", "solid", "dotted"),
        lwd = c(2,2,2),
        col = c(3, 1, 3))

abline(h = true_rl, col = 4, lwd = 2)
abline(h = results_mw[2], col = 7, lwd = 2)
abline(h = results_pw[2], col = 6, lwd = 2)
abline(h = est_rl_pw_range, col = 6, lty = "dotted", lwd = 2)
abline(h = est_rl_mw_range, col = 7, lty = "dotted", lwd = 2)
```


Estimates of a quantile
quantile
block size