Modeling extreme values with a GEV mixture probability distributions

Pascal Alain Dkengne Sielenou

September 28th, 2023

```
# library(xfun)
```

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

```
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"))
```

```
n <- 100000
```

```
loc <- 1
scale <- 0.5
shape <- -0.2
set.seed(1133)
x <- 10^(4)*generate_gev_sample(n = n, loc = loc, scale = scale, shape = shape)
```

```
nlargest <- 1000
```

```
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)
```

```
## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
```

```
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

## Warning in fgev.norm(x = x, start = start, ..., nsloc = nsloc, std.err =
## std.err, : optimization may not have succeeded

##   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] 10 11 12 13 14 15 16 17 18 19 20
```

```
gev_mixture_model$normalized_gev_parameters_object
```

```
##            loc_star       scale_star         shape_star
## 10 24431.2109423931 2619.91961470548 -0.293852465827186
## 11 24607.2086570341 2547.40266127139 -0.294242636372796
## 12 23633.2097801084 3357.87927912420 -0.362522572218454
## 13 23937.9071023575 3007.67296289125 -0.330132110224495
## 14 24861.8964765840 2322.70573943746 -0.273510732981684
## 15 24853.0310723524 2403.77586593845 -0.287128265495750
## 16 24238.0746133935 2903.17285148242 -0.327879231359023
## 17 25087.2114061197 2142.78668932798 -0.255847036752442
## 18 23618.4618421558 3402.00584992816 -0.369379365656601
## 19 23385.7057572796 3429.48815883843 -0.362354749809908
## 20 24634.5254630909 2599.99987534712 -0.307344598718113
```

```
gev_mixture_model$weighted_normalized_gev_parameters_object
```

```
##                            loc_star       scale_star         shape_star
## identic_weights     24298.9493738972 2794.25541348112 -0.314926705946950
## pessimistic_weights              NaN              NaN -0.313585746353088
## automatic_weights   24201.9104893711 2795.18209987531 -0.311626033400236
```

```
gev_mixture_model$automatic_weights_mw_statistics
```

```
## $function_value
## [1] 0.00102227669489621
##
## $gradient_value
## [1] 9.88397320132728e-06
##
## $function_reduction
## [1] 0.00353034755954195
##
## $number_iterations
## [1] 2041
##
## $convergence
## [1] 0
##
## $message
## [1] "Successful convergence"
```

```
gev_mixture_model$automatic_weights_pw_statistics
```

```
## $function_value
## [1] 0.00102712264977113
##
## $gradient_value
## [1] 6.55834678704864e-05
##
```

```
## $function_reduction
## [1] 0.00521586800857199
##
## $number_iterations
## [1] 2062
##
## $convergence
## [1] 0
##
## $message
## [1] "Successful convergence"
```

```
gev_mixture_model$automatic_weights_mw
```

```
##                 10                 11                 12                 13
## 0.2705419693502017 0.0000000000000000 0.0731542807315388 0.0894242292522226
##                 14                 15                 16                 17
## 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.2154127848055063
##                 18                 19                 20
## 0.0000000000000000 0.3514667358605306 0.0000000000000000
```

```
gev_mixture_model$pessimistic_weights_pw_shape
```

```
##                 10                 11                 12                 13
## 0.0927829772202200 0.0927467830967744 0.0866254005495207 0.0894771730868824
##                 14                 15                 16                 17
## 0.0946896707339967 0.0934089708409687 0.0896789815587366 0.0963770995551535
##                 18                 19                 20
## 0.0860334597983880 0.0866399394528250 0.0915395441065340
```

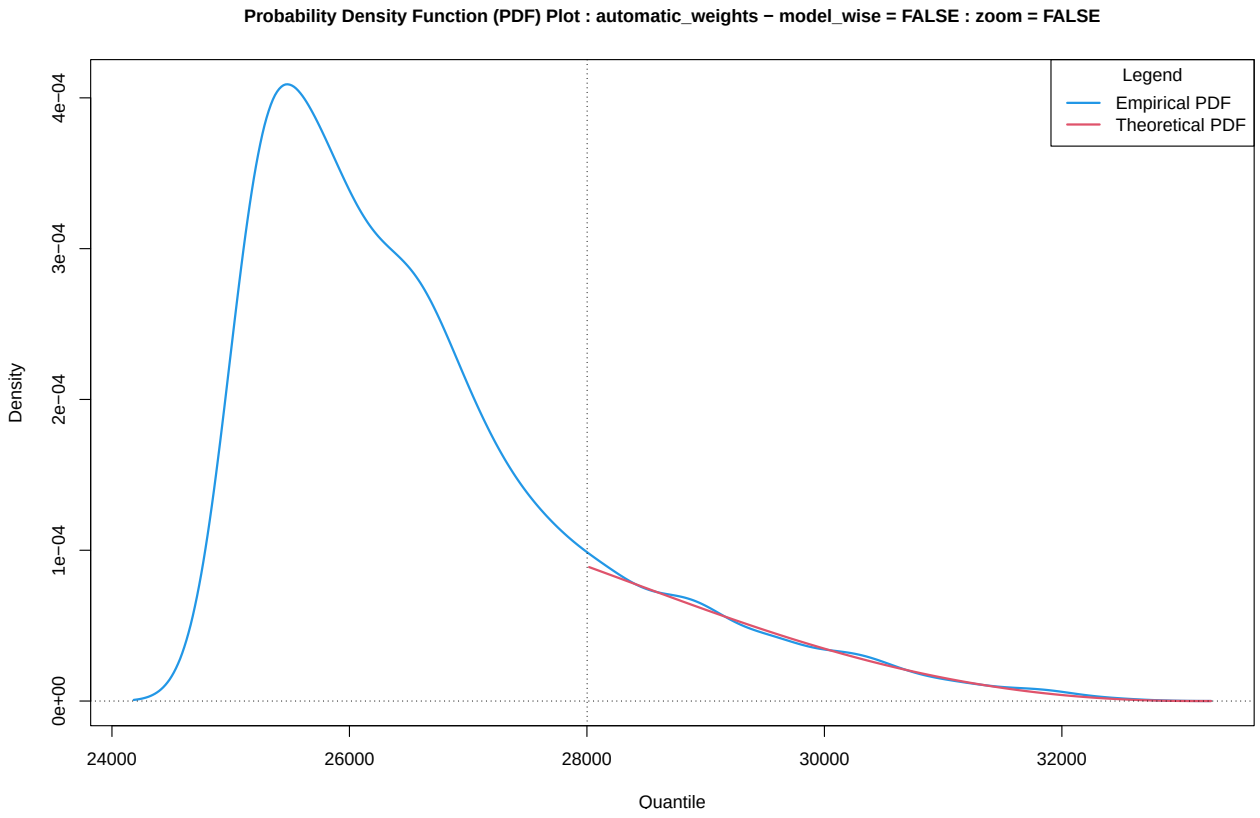
```
gev_mixture_model$pessimistic_weights_pw_scale
```

```
##  10  11  12  13  14  15  16  17  18  19  20
## NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
```

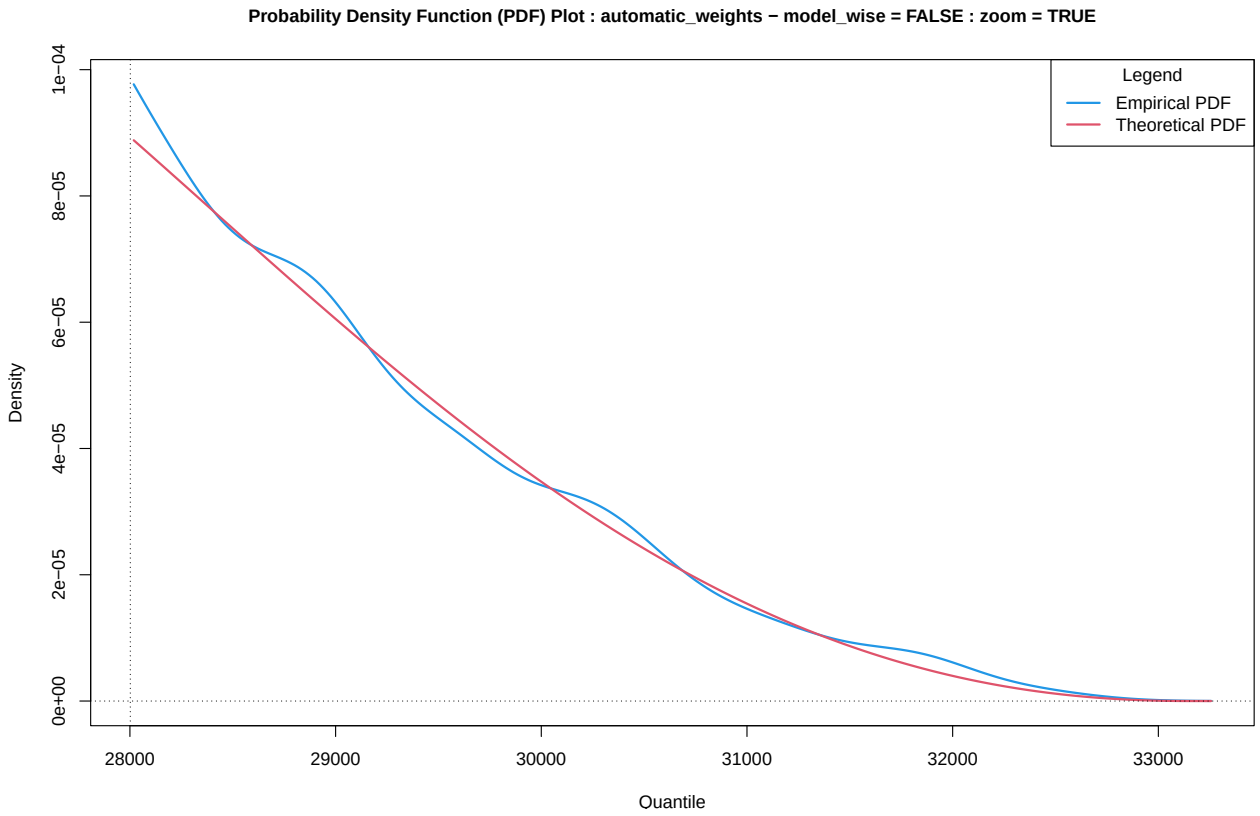
```
gev_mixture_model$pessimistic_weights_pw_loc
```

```
##  10  11  12  13  14  15  16  17  18  19  20
## NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
```

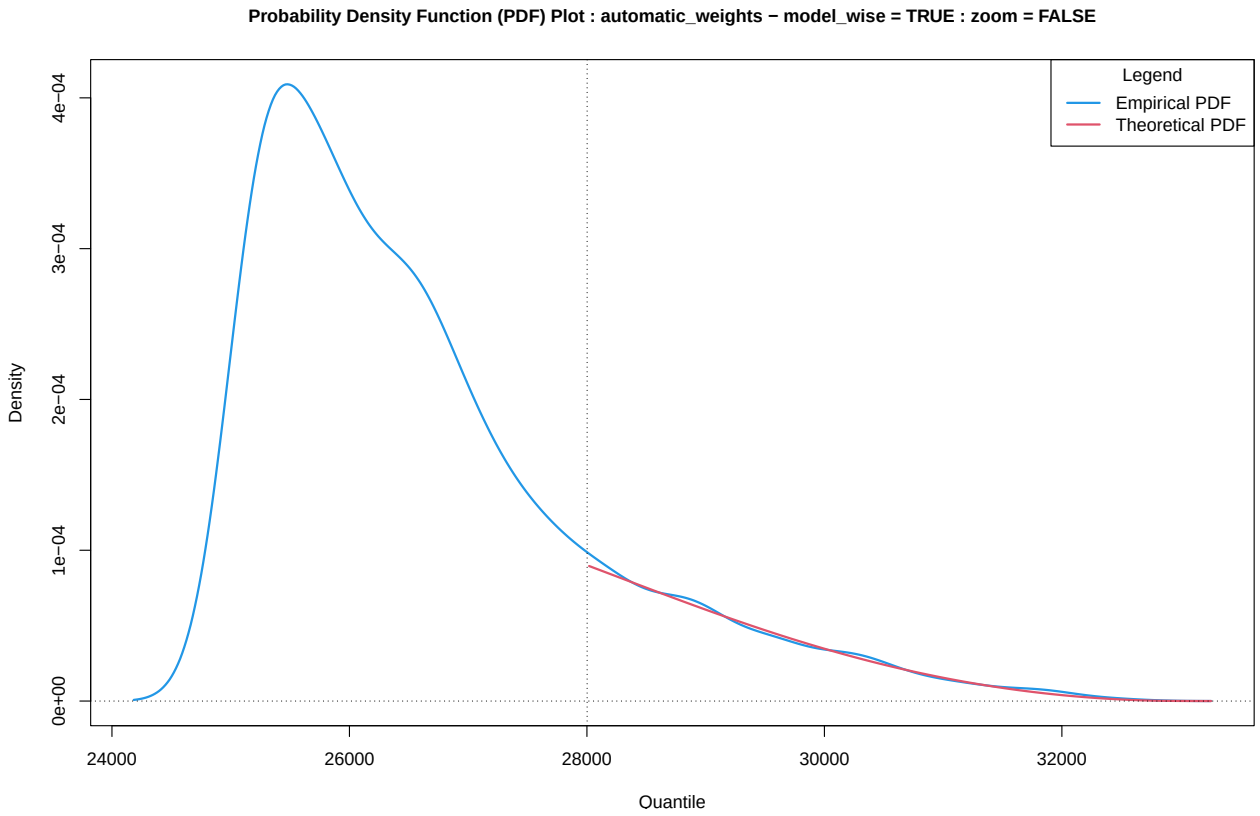
```
plot_gev_mixture_model_pdf(gev_mixture_model,
                           type = "automatic_weights",
                           model_wise = FALSE,
                           zoom = FALSE,
                           xlab = "Quantile",
                           ylab = "Density",
                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



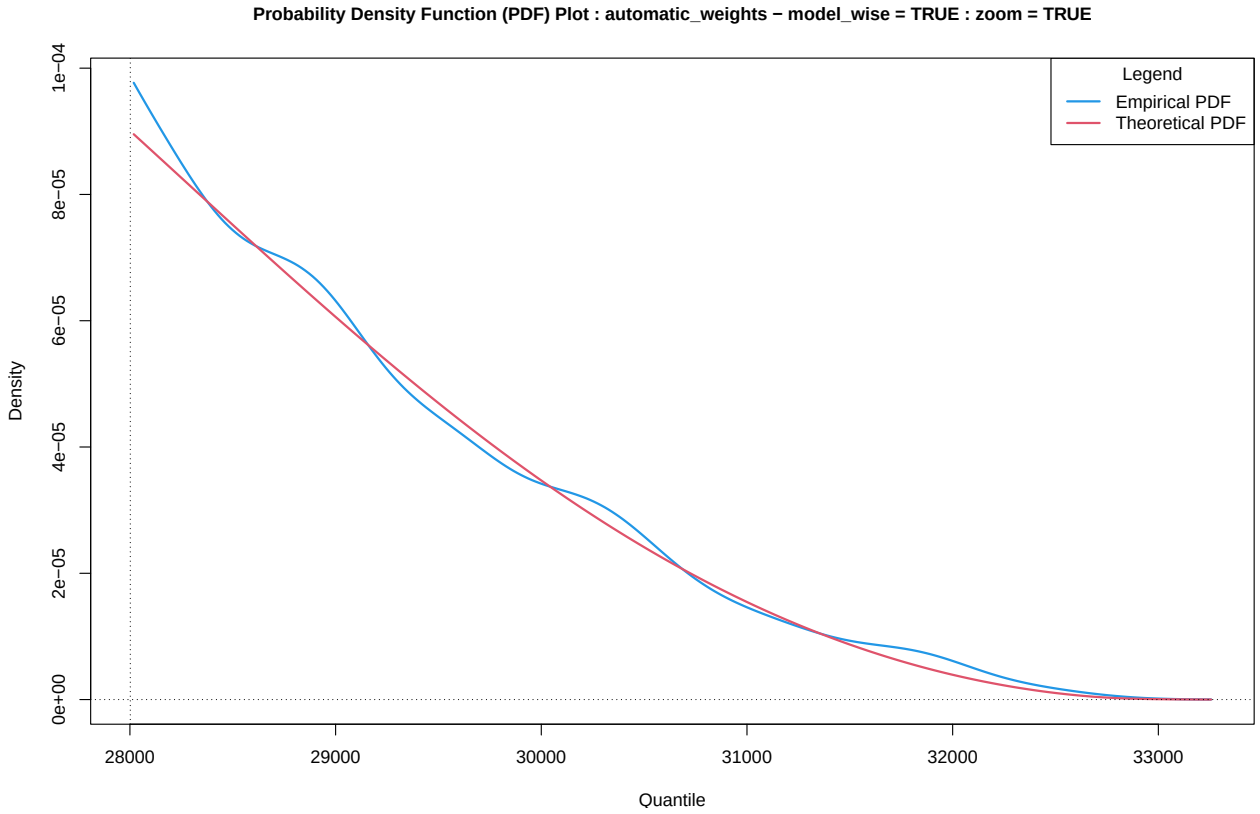

```
plot_gev_mixture_model_pdf(gev_mixture_model,
                           type = "automatic_weights",
                           model_wise = FALSE,
                           zoom = TRUE,
                           xlab = "Quantile",
                           ylab = "Density",
                 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




```
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 33451.9221800636    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 33169.944584519    NA
```

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

```
##             100%
## 32398.5999344147
```

```
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
## 10 31785.6331182716 33193.8400286708 34602.0469390699
## 11 31838.9990617533 33131.4558793662 34423.9126969791
## 12 31977.0574408066 32890.9673163774 33804.8771919482
## 13 31924.6208420283 32941.8486396079 33959.0764371874
## 14 31854.0527001457 33005.8957357253 34157.7387713048
## 15 31911.0964484679 32937.4226714265 33963.7488943851
## 16 32008.6108155339 32797.2486031696 33585.8863908053
## 17 31917.3922902304 32856.0849222868 33794.7775543432
## 18 31983.2721593442 32847.3946297078 33711.5171000714
## 19  32032.262671914  32751.425340214  33470.588008514
## 20 31739.7340612639  32990.599646043 34241.4652308221
```

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

```
## [1] 31739.7340612639 34602.0469390699
```

```
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
## 10 31785.6331182716 33193.8400286708 34602.0469390699
## 12 31977.0574408066 32890.9673163774 33804.8771919482
## 13 31924.6208420283 32941.8486396079 33959.0764371874
## 17 31917.3922902304 32856.0849222868 33794.7775543432
## 19  32032.262671914  32751.425340214  33470.588008514
```

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

```
## [1] 31785.6331182716 34602.0469390699
```

```r
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)
```