Modeling extreme values with a GEV mixture probability distributions

Pascal Alain Dkengne Sielenou

September 08th, 2023

```r
source("./src/estimate_gev_mixture_model_parameters.R")
source("./src/plot_gev_mixture_model_pdf.R")
source("./src/generate_gev_sample.R")
source("./src/plot_normalized_gev_mixture_model_pdf.R")
source("./src/calculate_gev_inverse_cdf.R")
source("./src/calculate_gev_mixture_model_inverse_cdf.R")
source("./src/calculate_gev_mixture_model_cdf.R")
```

```r
n <- 10000

nlargest <- 1000

x <- generate_gev_sample(n = n, loc = 1, scale = 0.5, shape = 0.1)
#x <- rnorm(n = n)

gev_mixture_model <- estimate_gev_mixture_model_parameters(x,
                                                           nsloc = NULL,
                                                           std.err = FALSE,
                                                           block_sizes = NULL,
                                                           minimum_nblocks = 50,
                                                           nlargest = nlargest,
                                                           confidence_level = 0.95,
                                                           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]  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
## 9  2.21148202968153 0.701283021401025 0.0651780564331574
## 10 2.41770222111227 0.547845258462672 0.1458617049465941
## 11 2.26543297654134 0.640478060086311 0.0962442278086965
## 12 2.52214062662878 0.506952597447726 0.1524211123443312
## 13 2.43202269755868 0.608518223863495 0.0974497043831831
## 14 2.34212007226810 0.623476936412195 0.1014877364889105
## 15 2.46456699413990 0.556643248727518 0.1254108378075822
## 16 2.26808174749938 0.631759617894213 0.0978039898123760
## 17 2.04986455294450 0.763682798248232 0.0476291605240021
## 18 1.96940702067090 0.793487569494478 0.0434595597776205
## 19 2.41776912700514 0.592451099425241 0.1033663332399102
## 20 2.43441858267285 0.532391250781759 0.1495249898221849
```

```
gev_mixture_model$weighted_normalized_gev_parameters_object
```

```
##                            loc_star        scale_star        shape_star
## identic_weights     2.31625072072695 0.624914140187072 0.102153117782379
## pessimistic_weights 2.34104415834567 0.632451810208506 0.103406376243221
## automatic_weights   2.52214062662878 0.522252938572417 0.148764835520519
```

```
gev_mixture_model$automatic_weights_mw
```

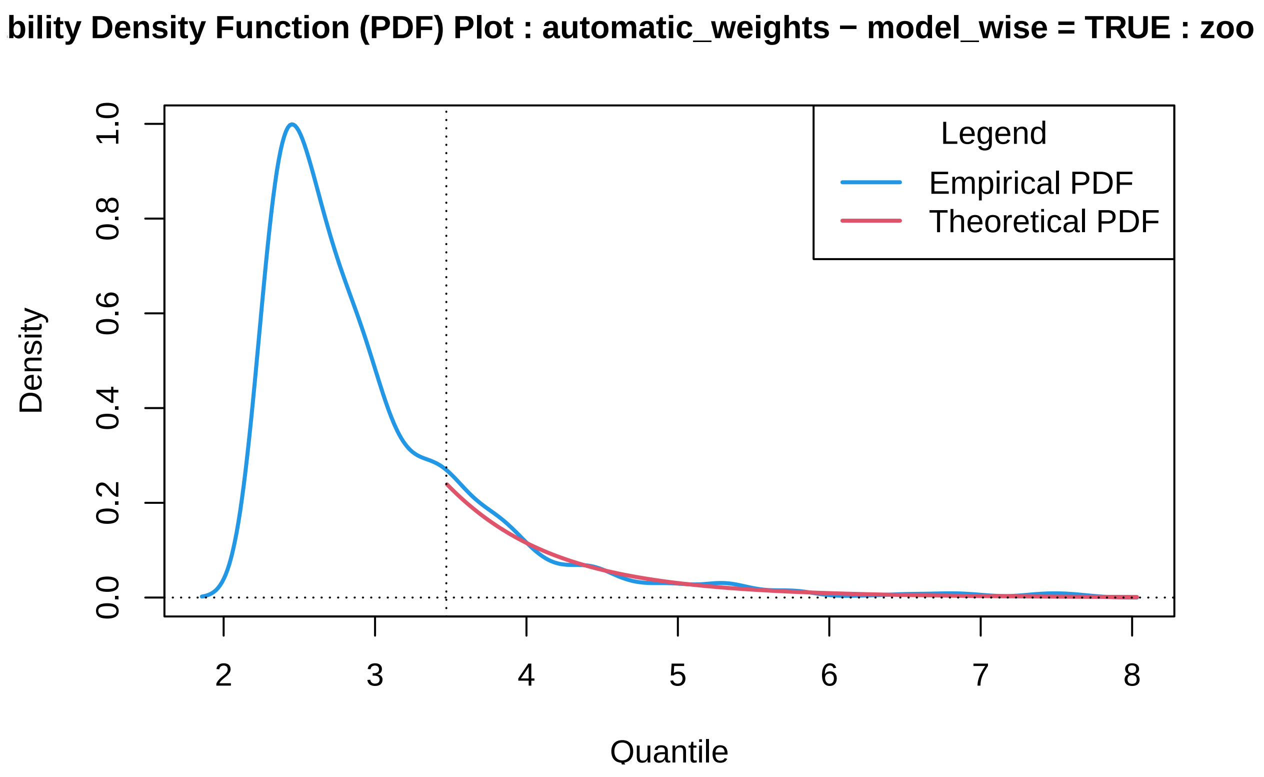
```
##                 9                10                11                12
## 0.000000000000000 0.000000000000000 0.000000000000000 0.539881036411926
##                13                14                15                16
## 0.460118963588079 0.000000000000000 0.000000000000000 0.000000000000000
##                17                18                19                20
## 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000
```

```
gev_mixture_model$automatic_weights_mw_statistics
```

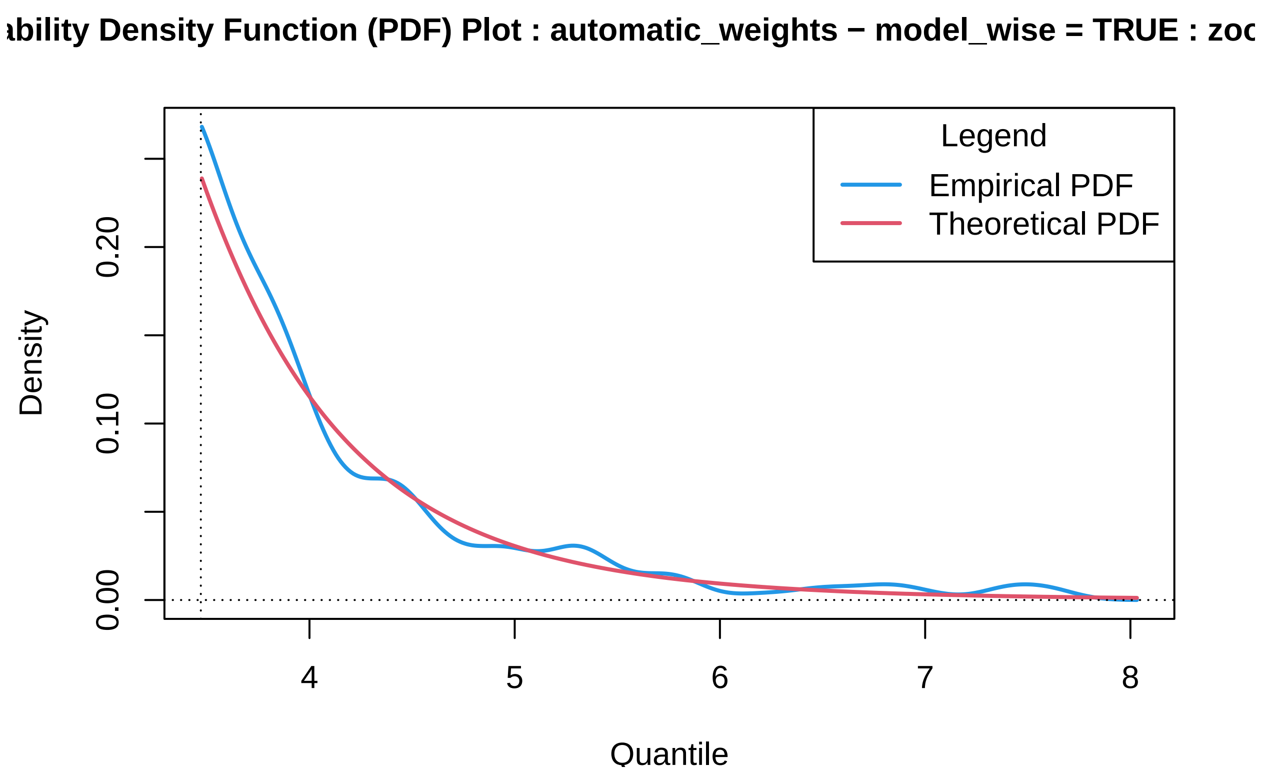
```
## $function_value
## [1] 0.00159638197375357
##
## $gradient_value
## [1] 1.28567998360918e-05
##
## $function_reduction
## [1] 0.00633723686278284
##
## $number_iterations
## [1] 1686
```

```
##
## $convergence
## [1] 0
##
## $message
## [1] "Successful convergence"
```

```
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")
```



```
gev_mixture_model_parameters <- gev_mixture_model$normalized_gev_parameters_object

shapes <- gev_mixture_model_parameters$shape_star
scales <- gev_mixture_model_parameters$scale_star
locations <- gev_mixture_model_parameters$loc_star

weights <- gev_mixture_model$automatic_weights_mw


#
p <- 0.95

q_initial_guesses <- sapply(1:length(weights), function(j) calculate_gev_inverse_cdf(p = p,
                                                                                      loc = locations[j]
                                                                                      scale = scales[j],
                                                                                      shape = shapes[j])
q_initial_guesses
```

```
##  [1] 4.50971560453362 4.45432428033032 4.46755613389509 4.42655669219545
##  [5] 4.52818861430837 4.50336965664996 4.46791337369446 4.44550802452341
##  [9] 4.48643775222620 4.48509458280305 4.47749781992695 4.42517601945482
```

```
range(q_initial_guesses)
```

```
## [1] 4.42517601945482 4.52818861430837
```

```
block_size <- max(gev_mixture_model$block_sizes)
y <- gev_mixture_model$data_largest
threshold <- find_threshold_associated_with_given_block_size(x = y, block_size = block_size)
```

```r
library(evd)

data <- y[y > threshold]

M3 <- fgev(data, prob = 0.95)

M3
```

```
##
## Call: fgev(x = data, prob = 0.95)
## Deviance: 314.675315649226
##
## Estimates
##       quantile           scale           shape
## 3.523819342571  0.345490276736  0.627898399701
##
## Standard Errors
##        quantile             scale             shape
## 0.0154618298845  0.0321413773338  0.1006542853766
##
## Optimization Information
##   Convergence: successful
##   Function Evaluations: 58
##   Gradient Evaluations: 14
```

```r
M4 <- fgev(data)

M4
```

```
##
## Call: fgev(x = data)
## Deviance: 314.67531576556
##
## Estimates
##            loc           scale           shape
## 3.797775590982  0.345499756111  0.627903834336
##
## Standard Errors
##             loc             scale             shape
## 0.0313419585286  0.0321410777596  0.1006637380539
##
## Optimization Information
##   Convergence: successful
##   Function Evaluations: 58
##   Gradient Evaluations: 13
```

```r
Fn <- ecdf(y)
```

```r
p <- seq(from = Fn(threshold), to = 0.999, length.out = 20)
p
```

```
##  [1] 0.817000000000000 0.826578947368421 0.836157894736842 0.845736842105263
##  [5] 0.855315789473684 0.864894736842105 0.874473684210526 0.884052631578947
##  [9] 0.893631578947368 0.903210526315789 0.912789473684210 0.922368421052632
## [13] 0.931947368421053 0.941526315789474 0.951105263157895 0.960684210526316
## [17] 0.970263157894737 0.979842105263158 0.989421052631579 0.999000000000000
```

```
quantiles <- calculate_gev_mixture_model_inverse_cdf(p = p, locations, scales, shapes, weights, iterati

quantiles
```

```
##  [1] 3.46002362720123 3.50037403721237 3.54310128254791 3.58851347891750
##  [5] 3.63698054074955 3.68895172384621 3.74497981548163 3.80575522529425
##  [9] 3.87215524485294 3.94531731745674 4.02675179184284 4.11852261338232
## [13] 4.22355147932363 4.34616202207139 4.49313231782663 4.67595233541690
## [17] 4.91642203510300 5.26399272589798 5.87626758614756 8.58485515252426
```

```
probaility <- calculate_gev_mixture_model_cdf(q = quantiles, locations, scales, shapes, weights)

probaility
```

```
##  [1] 0.817000000000000 0.826578947368421 0.836157894736842 0.845736842105263
##  [5] 0.855315789473684 0.864894736842105 0.874473684210526 0.884052631578947
##  [9] 0.893631578947368 0.903210526315789 0.912789473684210 0.922368421052631
## [13] 0.931947368421053 0.941526315789474 0.951105263157895 0.960684210526316
## [17] 0.970263157894737 0.979842105263158 0.989421052631579 0.999000000000000
```

```
qnorm(p = p)
```

```
##  [1] 0.903991327564401 0.940732215808152 0.978789071673981 1.018319139079168
##  [5] 1.059508146669658 1.102577866825039 1.147796381232619 1.195492315470876
##  [9] 1.246075054188577 1.300064255422559 1.358134364813947 1.421184402498285
## [13] 1.490452629208801 1.567716246440469 1.655666013005180 1.758681749578263
## [17] 1.884675336150669 2.050498710750263 2.305154988620419 3.090232306167813
```

```
calculate_gev_inverse_cdf(p = p, loc = 1, scale = 0.5, shape = 0.1)
```

```
##  [1] 1.86691633888349 1.90187030320189 1.93881492756769 1.97800578793964
##  [5] 2.01974928995287 2.06441700722850 2.11246542385426 2.16446371347282
##  [9] 2.22113381047765 2.28340990302904 2.35252980147078 2.43018103339487
## [13] 2.51874615311531 2.62174040574472 2.74465546629867 2.89676214014992
## [17] 3.09555949298756 3.38047039184760 3.87580619642557 5.97581256378162
```