Modeling extreme values with a GEV mixture probability distributions

Pascal Alain Dkengne Sielenou

September 08th, 2023

```
# library(xfun)
```

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

```
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/generate_gev_sample.R"))
xfun::in_dir(dir = path, expr = source("./src/plot_normalized_gev_mixture_model_pdf.R"))
xfun::in_dir(dir = path, expr = source("./src/calculate_gev_inverse_cdf.R"))
xfun::in_dir(dir = path, expr = source("./src/calculate_gev_mixture_model_inverse_cdf.R"))
xfun::in_dir(dir = path, expr = source("./src/calculate_gev_mixture_model_cdf.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,
                                                           log_mv = TRUE,
                                                           log_pw = TRUE,
                                                           trace = FALSE)
```

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

```
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  1.41370371485639 0.433419418648051 -0.05204184146067875
## 10 1.57622645005483 0.335838579775539  0.01469818354793086
## 11 1.62898723619989 0.289057692100018  0.06280119249957608
## 12 1.48489346538707 0.365923646277552 -0.00774059477708921
## 13 1.43817913615470 0.386293737001565 -0.01792973264839613
## 14 1.40670954806691 0.420024122980837 -0.03688007766472778
## 15 1.62987966523946 0.262750392610992  0.08830975241117328
## 16 1.33676882105268 0.432838677330390 -0.03518325217318527
## 17 1.67944509403983 0.222698896762505  0.14164100942262223
## 18 1.40193760191317 0.424839955981841 -0.03976984795315776
## 19 1.64788992139209 0.227082091184602  0.13794250850065726
## 20 1.52169987658957 0.366591625428071 -0.00852079847170092
```

```
gev_mixture_model$weighted_normalized_gev_parameters_object
```

```
##                              loc_star        scale_star         shape_star
## identic_weights     1.51386004424555 0.347279903006830 0.0206105417694187
## pessimistic_weights 1.52618431032864 0.352921818152725 0.0251609479667012
## automatic_weights   1.67243783031621 0.225268530655557 0.1407030404745860
```

```
gev_mixture_model$automatic_weights_mw
```

```
##                      9                     10                     11
##  0.00000000000000e+00  2.16840434497101e-19  0.00000000000000e+00
##                     12                     13                     14
##  2.16840434497101e-19  0.00000000000000e+00  0.00000000000000e+00
##                     15                     16                     17
##  5.42101086242752e-20 -1.08420217248550e-19  9.32000458348474e-01
##                     18                     19                     20
##  0.00000000000000e+00  6.79995416515258e-02  8.67361737988404e-19
```

```
gev_mixture_model$automatic_weights_mw_statistics
```

```
## $function_value
## [1] 0.00452025026194538
##
```

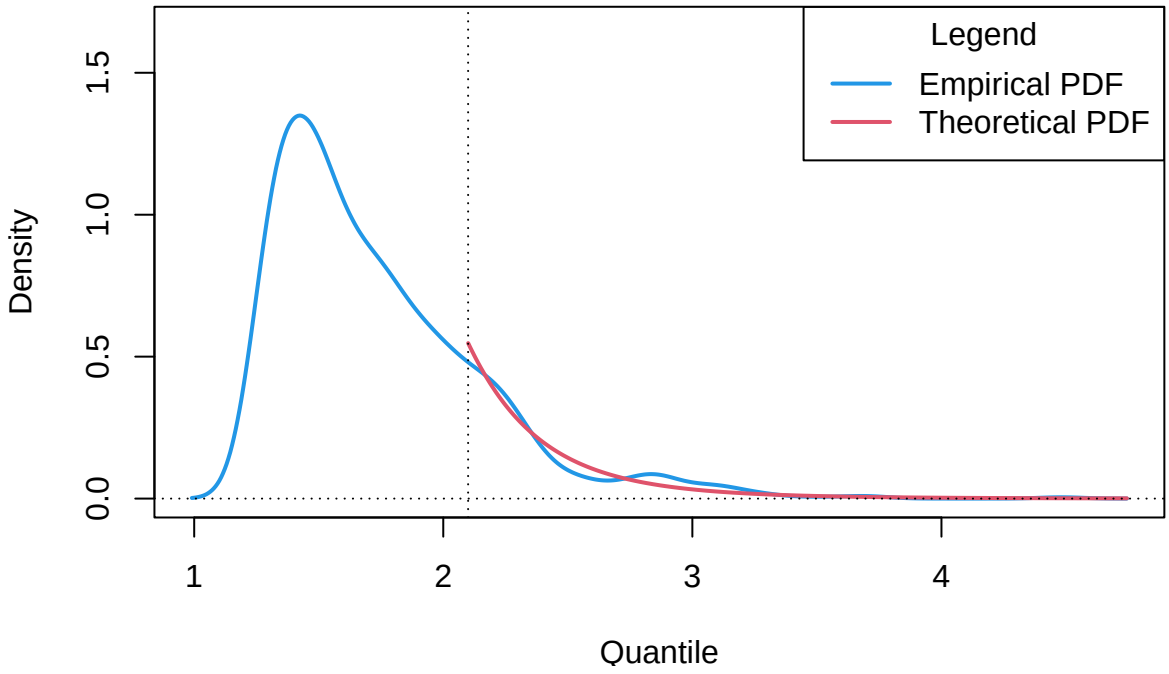
```
## $gradient_value
## [1] 9.88674862845151e-06
##
## $function_reduction
## [1] 0.0124256427056289
##
## $number_iterations
## [1] 1776
##
## $convergence
## [1] 0
##
## $message
## [1] "Successful convergence"
```

```
gev_mixture_model$automatic_weights_pw_statistics
```

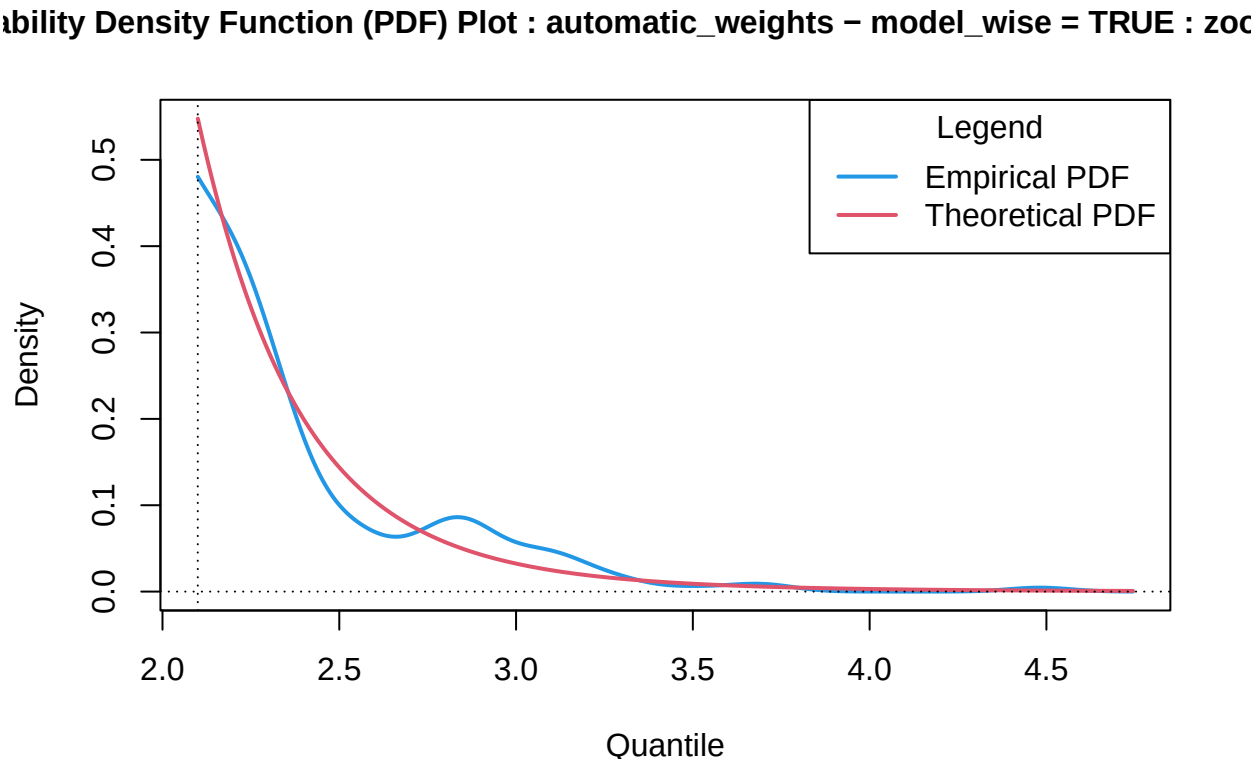
```
## $function_value
## [1] 0.00451952376277734
##
## $gradient_value
## [1] 2.5801123390623e-05
##
## $function_reduction
## [1] 0.0200102268005128
##
## $number_iterations
## [1] 3361
##
## $convergence
## [1] 0
##
## $message
## [1] "Successful convergence"
```

```
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")
```

bility Density Function (PDF) Plot : automatic_weights − model_wise = TRUE : zoo




```
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] 2.60648338981699 2.59582674978021 2.57283909285170 2.55935924896897
##  [5] 2.55553072519481 2.58836213949738 2.52221840733803 2.55749012357476
##  [9] 2.50179235381568 2.59211711890790 2.48151360719230 2.59688555829456
```

```
range(q_initial_guesses)
```

```
## [1] 2.48151360719230 2.60648338981699
```

```
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: 11.2238252008678
##
## Estimates
##       quantile           scale           shape
## 2.129822618113  0.150069644830  0.640421963399
##
## Standard Errors
##         quantile             scale             shape
## 0.00684670573505  0.01454366206080  0.09949356843329
##
## Optimization Information
##   Convergence: successful
##   Function Evaluations: 72
##   Gradient Evaluations: 15
```

```r
M4 <- fgev(data)

M4
```

```
##
## Call: fgev(x = data)
## Deviance: 11.2238263286345
##
## Estimates
##            loc           scale           shape
## 2.248104138590  0.150081979566  0.640485920448
##
## Standard Errors
##             loc            scale            shape
## 0.0139560758785  0.0145404231112  0.0995392717871
##
## Optimization Information
##   Convergence: successful
##   Function Evaluations: 98
##   Gradient Evaluations: 15
```

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

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

```
##  [1] 0.833000000000000 0.841736842105263 0.850473684210526 0.859210526315789
##  [5] 0.867947368421053 0.876684210526316 0.885421052631579 0.894157894736842
##  [9] 0.902894736842105 0.911631578947368 0.920368421052632 0.929105263157895
## [13] 0.937842105263158 0.946578947368421 0.955315789473684 0.964052631578947
## [17] 0.972789473684211 0.981526315789474 0.990263157894737 0.999000000000000
```

```r
quantiles <- calculate_gev_mixture_model_inverse_cdf(p = p*0.1, locations, scales, shapes, weights, ite

quantiles
```

```
##  [1] 1.48706783307372 1.48789253470738 1.48871263750304 1.48952823015028
##  [5] 1.49033939884935 1.49114622740596 1.49194879732159 1.49274718787943
##  [9] 1.49354147622642 1.49433173745142 1.49511804465984 1.49590046904485
## [13] 1.49667907995543 1.49745394496133 1.49822512991521 1.49899269901206
## [17] 1.49975671484605 1.50051723846486 1.50127432942184 1.50202804582585
```

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

probaility
```

```
##  [1] 0.0833000000000002 0.0841736842105264 0.0850473684210530 0.0859210526315786
##  [5] 0.0867947368421052 0.0876684210526315 0.0885421052631577 0.0894157894736840
##  [9] 0.0902894736842107 0.0911631578947366 0.0920368421052629 0.0929105263157893
## [13] 0.0937842105263157 0.0946578947368420 0.0955315789473685 0.0964052631578948
## [17] 0.0972789473684211 0.0981526315789476 0.0990263157894738 0.0999000000000002
```

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

```
##  [1] 0.966088297132373 1.001621742608749 1.038467127816570 1.076779138413667
##  [5] 1.116740576415721 1.158569814335721 1.202530917796036 1.248947680670282
##  [9] 1.298223554621685 1.350870741699792 1.407554063282192 1.469159716538730
## [13] 1.536908207017596 1.612550936048200 1.698738737882498 1.799784091686931
## [17] 1.923469983777603 2.086345163158907 2.336337007166036 3.090232306167813
```

```r
calculate_gev_inverse_cdf(p = p*0.1, loc = 2.52214, scale = 0.5222, shape = 0.1487)
```

```
##  [1] 2.07750635029760 2.07942570300296 2.08133440970566 2.08323267584489
##  [5] 2.08512070109648 2.08699867959228 2.08886680012908 2.09072524636772
##  [9] 2.09257419702287 2.09441382604401 2.09624430278819 2.09806579218487
## [13] 2.09987845489338 2.10168244745343 2.10347792242883 2.10526502854513
## [17] 2.10704391082109 2.10881471069467 2.11057756614355 2.11233261180067
```

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

```
##  [1] 1.92640062733718 1.96134887237630 1.99830772835014 2.03753446817997
##  [5] 2.07933753588073 2.12409097004787 2.17225425985083 2.22440027768804
##  [9] 2.28125556154545 2.34376010343445 2.41315913846605 2.49114984874789
## [13] 2.58012756342819 2.68362471556287 2.80715633094270 2.96002430998413
## [17] 3.15976273247287 3.44578307486668 3.94174422551237 5.97581256378162
```