Appendix

Final Seminary Work Statistics of extreme events and geostatistics

Gauthier Heroufosse, Valentino Mascherini

2023-01-15

Contents


```
rm(list=ls())

# T-100 event libraries
library(knitr)
library(ggplot2)
library(readr)
library(kableExtra)
library(dplyr)
library(lmom)
library(ismev)
library(extRemes)
```

Table 1: Descriptive statistics of the dataset grouped for every station. The statistics are mean, maximum and standard deviation, as minimum is always 0.

name	mean	max	sd	name	mean	max	sd
bad_gleichenberg	2.28	112.6	6.12	loibl	5.93	233.2	15.05
bad_ischl	4.70	151.4	9.14	mariazell	3.09	101.0	6.64
bregenz	3.72	134.0	8.05	mayrhofen	2.88	110.5	6.60
bruck_mur	2.12	86.7	5.79	mondsee	4.22	140.1	8.46
eisenstadt	1.74	131.7	5.10	murzzuschlag	2.67	102.9	6.38
feldkirch	3.21	99.4	6.79	obergurgl_vent	2.24	90.5	5.60
galtur	2.60	111.7	5.75	pabneukirchen	2.42	121.0	5.52
gleisdorf	2.19	89.3	5.87	patscherkofel	2.46	73.1	5.42
graz_flughafen	2.27	102.9	6.31	preitenegg	2.41	111.3	6.36
graz_universitat	2.23	84.2	6.19	reichenau_rax	2.39	121.6	6.18
grossraming	3.69	123.2	7.67	reichersberg	2.23	98.0	5.21
hieflau	4.48	121.0	8.99	reisach	3.93	169.5	11.15
holzgau	3.71	150.5	7.71	retz	1.19	91.4	3.77
innsbruck_flughafen	2.47	106.3	5.83	reutte	3.77	212.5	7.76
innsbruck_universitat	2.40	92.2	5.64	salzburg_flughafen	3.03	124.9	6.59
irdning_gumpenstein	2.80	91.8	6.49	schockl	2.70	113.6	7.07
jenbach	3.18	93.2	6.84	schoppernau	5.17	197.4	9.68
kanzelhohe	3.08	84.7	7.53	schrocken	6.12	183.5	10.98
klagenfurt_flughafen	2.46	94.1	6.67	seckau	2.24	101.2	6.06
kolbnitz	3.08	136.7	8.95	st_jakob_im_def	2.65	117.3	6.55
kremsmunster	2.72	110.0	6.00	st_michael_ob_bleiburg	2.82	95.5	7.42
kufstein	3.56	103.5	7.16	stift_zwettl	1.81	95.0	4.80
laa_an_der_thaya	1.32	83.4	4.12	stolzalpe	2.49	124.5	6.36
landeck	2.01	78.2	4.77	villacher_alpe	3.79	167.4	9.28
lienz	2.46	138.7	7.33	worterberg	1.93	88.5	5.59
linz_horsching	2.09	102.7	4.92	zell_am_see	3.01	94.2	6.46
lobming	2.48	125.6	6.88	zeltweg	2.15	89.7	5.96

```
library(lubridate)

# Kriging libraries
library(data.table)
library(tidyverse)
library(MASS)
library(geoR)
library(sf)
library(car)
library(AID)
library(raster)

setwd("C://Users//valen//Documents//Freiburg//BOKU//WT23//ExEv//Extreme_Events//Seminary_final")
```

1 Dataset

Table 1 shows the descriptive statistics for every single station.

2 Automatic Annual Maxima

```
a = 0 # parameter for model

rlevel <- data_frame(name = sn$name, mle = rep(NA, length(sn$name)), lmom = rep(NA, length(sn$name)))

rmse <- data.frame(name = sn$name, mle = rep(NA, length(sn$name)), lmom = rep(NA, length(sn$name)))

mae <- data.frame(name = sn$name, mle = rep(NA, length(sn$name)), lmom = rep(NA, length(sn$name)))

indicators <- data.frame(name = sn$name, nt = rep(NA, length(sn$name)))

j <- 1 # counter

for (i in sn$name) {

  #---- get AM data ----

  command <- paste0("df <- ",i)

  eval(parse(text= command))

  # parse data

  df[, 1] <- as.POSIXct(df$datum)

  df <- data.frame(date=df$datum,
                   year = year(df$datum),
                   rr =  as.numeric(df$rr))

  df <- na.omit(df)

  AM <- aggregate(df, by = list(df$year), max)

  AM <- AM$rr

  #----- fit models and calculate RMSE and MAE -----

  mod_fit_mle <- fevd(AM, type = "GEV", method = "MLE", units = "mm")

  mod_fit_lmom <- fevd(AM, type = "GEV", method = "Lmoments", units = "mm")

  xp <- ppoints(n = AM, a = a) # get probabilities

  rp <- -1/log(xp) # get return period for the probs
  rp <- rp[which(rp > 5)] # only the ones > 5 for threshold


  x_ord <- sort(AM)

  x_ord <- x_ord[(length(x_ord) - length(rp) +1):length(x_ord)]

  # get return level for the model based on the observations' return period
```

```
  nt <- length(rp)

  rlevel_mle <- return.level(mod_fit_mle, return.period = rp)

  rlevel_lmom <- return.level(mod_fit_lmom, return.period = rp)

  # MLE

  rmse_mle <- (sqrt(sum(x_ord - as.numeric(rlevel_mle))^2 * 1/nt))

  mae_mle <- sum(abs((x_ord - as.numeric(rlevel_mle)))) * 1/nt

  # lmom

  rmse_lmom <- (sqrt(sum(x_ord - as.numeric(rlevel_lmom))^2 * 1/nt))

  mae_lmom <- sum(abs((x_ord - as.numeric(rlevel_lmom)))) * 1/nt

  # return level T100

  t_100_mle <- as.numeric(return.level(mod_fit_mle, 100)[1])

  t_100_lmom <- as.numeric(return.level(mod_fit_lmom, 100)[1])

  # insert in df

  rmse$mle[j] <- rmse_mle

  rmse$lmom[j] <- rmse_lmom

  mae$mle[j] <- mae_mle

  mae$lmom[j] <- mae_lmom

  # insert t-100 return level

  rlevel$mle[j] <- t_100_mle

  rlevel$lmom[j] <- t_100_lmom

  j <- j + 1
}
```

The code automates the obtaining of the annual maxima series and then performs the fitting of a GEV model with both MLE and L-moments. The T-100 return level is extracted and the CRMSE and CMAE are calculated. The results are stored in a data set that has been saved. The last part is the same for every method so it won't be included in the next ones. The return period calculation is the one that the *extRemes* package (Gilleland and Katz 2016) uses to produce the plots with the *plot.fevd* function. Then, those return periods are used to obtain return levels via the *return.level* function. The CRMSE and CMAE make it so we apply a threshold over the return levels, as described in the section 3.3 of the main report.

3 Automatic Peak Over Threshold

It is not to be confused the step of threshold selection (MLE or Lmom) which are shown below, with the one of model parameter estimation, which uses the same mathematical approaches.

3.1 Threshold selection via MLE

```
source(".....//JointMLEFunctions.r")

indicators <- data_frame(name = sn$name, u = rep(NA, length(sn$name)), umax = rep(NA, length(sn$name)),

j <- 1 # counter

for (i in (sn$name)) {

  #---- get data ----

  command <- paste0("df <- ",i)

  eval(parse(text= command))

  df[, 1] <- as.POSIXct(df$datum)

  df <- data.frame(date=df$datum,
                   year = year(df$datum),
                   rr =  as.numeric(df$rr))

  df$rr <- as.numeric(df$rr)

  df$rr <- ifelse(df$rr > 0, df$rr, 0)

  df <- na.omit(df)

  x1 <- df$rr

  #---- inner loop: thresholds loop ----

  fin_qtl <- 1 - 65/length(x1)

  steps <- seq(.75, fin_qtl, length.out = 20)

  #steps <- seq(.95, fin_qtl, length.out = 20)

  result_wads <- NHPP.diag(x1, k = 20, q1 = .9, q2 = fin_qtl)

  ustar <- as.numeric(result_wads$thresh)

  indicators$u[j] <- ustar

  indicators$umax[j] <- quantile(x1,fin_qtl)

  #indicators$umax[j] <- max(thresh)
```

```
  # fit GP with fevd

  #----- fit models and calculate RMSE and MAE -----

  mod_fit_mle <- fevd(x1, type = "GP", method = "MLE", threshold = ustar, units = "mm")

  mod_fit_lmom <- fevd(x1, type = "GP", method = "Lmoments", threshold = ustar, units = "mm")

  x2 <- x1[x1 > ustar]

  xp <- ppoints(n = x2, a = a) # get probabilities

  #xp <- xp [x1 > ustar]

  npy <- length(x2)/length(unique(df$year)) # number of points per year

  rp <- -1/log(xp)/npy # get return period for the probs
  rp <- rp[which(rp > 5)] # only the ones > 1


  x_ord <- sort(x2)

  x_ord <- x_ord[(length(x_ord) - length(rp) + 1):length(x_ord)]

  indicators$obs[j] <- length(x_ord)

  # get return level for the model based on the obs rp

  rlevel_mle <- return.level(mod_fit_mle, return.period = rp)

  rlevel_lmom <- return.level(mod_fit_lmom, return.period = rp)

  indicators$NAs_in_rl[j] <- length(rlevel_mle) -  length(na.omit(rlevel_mle)) +
                             length(rlevel_lmom) - length(na.omit(rlevel_lmom))

  ...

}
```

The method used was developed and described in a paper by Wadsworth (2016), and it has been slightly modified to be applied for our purposes. It uses the function *NHPP.diag()* which is sourced from a script from the supplemental material of the aforementioned paper. To put it very simply, through the joint asymptotic distribution of MLE of non-homogeneous Poisson processes (NHPP), it is obtained a threshold that defines the point from where a GPD would perform best if applied. The shape parameter ξ of the distribution is the one it is considered after establishing the asymptotic distribution, and from estimator of such parameters, a white noise process is derived. Due to characteristics inherent to the extreme value theory, it is assumed that white noise is found more frequently in the tail of the observations rather than in the main body. A distribution using the paramters obtained after the changepoint is tested against a null distribution, and a p-value is obtained. If that is significant, the threshold is selected. Due to the complex mathematics of the approach, it is suggested to directly consult the paper of Wadsworth (2016) to understand the details of the matter.

3.2 Threshold selection via L-moments

```
indicators <- data_frame(name = sn$name, u = rep(NA, length(sn$name)), umax = rep(NA, length(sn$name)),
                         nt = rep(NA, length(sn$name)))

j <- 1 # counter

for (i in (sn$name)) {

  #---- get data ----

  command <- paste0("df <- ",i)

  eval(parse(text= command))

  df[, 1] <- as.POSIXct(df$datum)

  df <- data.frame(date=df$datum,
                   year = year(df$datum),
                   rr =  as.numeric(df$rr))

  df$rr <- as.numeric(df$rr)

  df$rr <- ifelse(df$rr > 0, df$rr, 0)

  df <- na.omit(df)

  x1 <- df$rr

  #---- inner loop: thresholds loop ----

  fin_qtl <- 1 - 65/length(x1) # upper limit

  # thresholds to be evaluated

  steps <- seq(.75, fin_qtl, length.out = 20)

  thresh <- quantile(x1, steps)

  result <- data.frame(thresh = unique(thresh), du = rep(NA, length(unique(thresh))))

  k <- 1

  # inner loop: threshold selection using l-moments

  for (l in unique(thresh)) {

    x2 <- x1 [x1 > l]

    # get parameters of the fitted lmom distribution

    lmoms <- samlmu(x2)
```

```
  pars <- pelgpa(lmoms)

  eps <-  -as.numeric(pars[3]) # lmom package has epsilon as opposite

  # get biased moments using the pars

  tau3 <- (1 + eps)/(3 - eps)

  tau4 <- tau3 * (1 + 5 * tau3) / (5 + tau3)

  # get unbiased lmoments using the lmomco package

  unbiased <- lmom.ub(x2)

  t3 <- unbiased$TAU3

  t4 <- unbiased$TAU4

  du <- sqrt((t3 - tau3)^2 + (t4 - tau4)^2)

  result$du[k] <- du

  k <- k + 1
}


indx <- which(result$du == min(result$du))

ustar <- result$thresh[indx] # obtained threshold

indicators$u[j] <- ustar

indicators$umax[j] <- max(thresh)

# fit GP with fevd

#----- fit models and calculate RMSE and MAE -----

mod_fit_mle <- fevd(x1, type = "GP", method = "MLE", threshold = ustar, units = "mm")

mod_fit_lmom <- fevd(x1, type = "GP", method = "Lmoments", threshold = ustar, units = "mm")

x2 <- x1[x1 > ustar]

xp <- ppoints(n = x2, a = a) # get probabilities

npy <- length(x2)/length(unique(df$year)) # number of points per year

rp <- -1/log(xp)/npy # get return period for the probs
rp <- rp[which(rp > 5)] # only the ones > 5
```

```
  x_ord <- sort(x2)

  x_ord <- x_ord[(length(x_ord) - length(rp) + 1):length(x_ord)]

  indicators$obs[j] <- length(x_ord)

  # get return level for the model based on the obs rp

  rlevel_mle <- return.level(mod_fit_mle, return.period = rp)

  rlevel_lmom <- return.level(mod_fit_lmom, return.period = rp)

  indicators$NAs_in_rl[j] <- length(rlevel_mle) -  length(na.omit(rlevel_mle)) +
                             length(rlevel_lmom) - length(na.omit(rlevel_lmom))

  ...

}
```

Silva Lomba and Fraga Alves (2020) have recently developed this approach based on the L-moments in order to automate threshold selection. The theory behind uses L-moments ratios τ_3 and τ_4, that is L-skewness and L-kurtosis. Then we consider the unbiased estimators of the probability weighted moments α_r t_3 and t_4, which are sample L-skewness and L-kurtosis. Those are calculated for a sample (n = 20) of thresholds selected from the 75 % quantile to the maximum quantile. In the original paper, they use 20 sample thresholds from the 25 % with steps of 3.5 % until 96.5 %, but since the 25 % quantile is too low for the precipitation dataset, we stepped it up to 75%. The threshold which has the least RMSE between biased and unbiased estimators is selected as the best one, and a fitting of a GP model is performed. The *lmomco* package (Asquith 2022) has the function *lmom.ub()* which obtains the unbiased estimators from a dataset.

4 T-100 year event

Here is the distribution of the stations with their T-100 rain return event of the POT-WAD (Figure 1), alongside the results as a table 2.

5 Kriging

5.1 Import and preprocessing of the data

```
load("T_100_all.RData")

stations <- read_csv(file = "data/station_names_loc.csv")

# Join the t-100 event dataset and the location dataset
data_tot <- inner_join(rl_tot, stations, by='name')


# Checking if we truely have no NA
sum(is.na(data_tot))
# No missing value
```

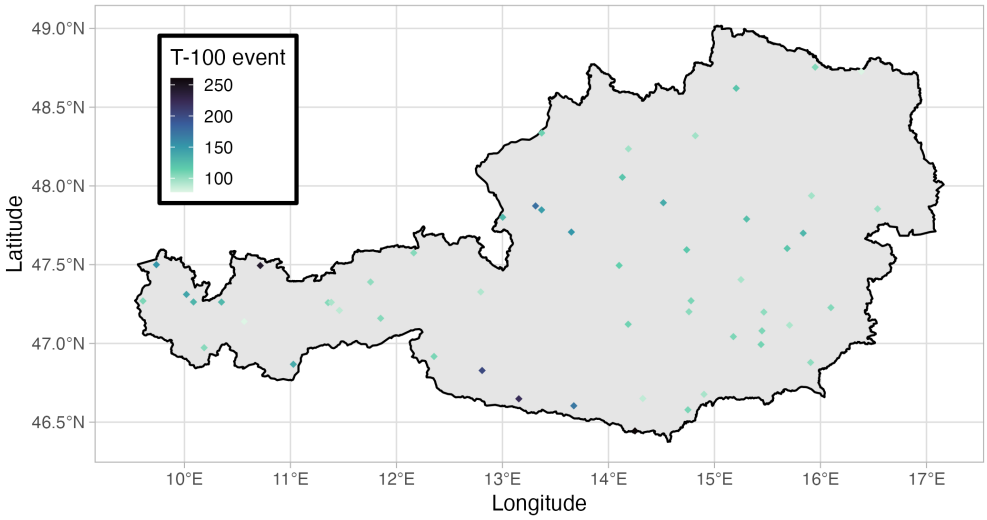

Figure 1: Stations with their T-100 resulting rain event

Table 2: Results of the various methods regarding the T-100 return level values, for all the stations.

name	mle_AM	lmom_AM	mle	lmom	mle_wad	lmom_wad
grossraming	135.74933	131.61547	144.67459	145.85704	137.46681	134.54641
reutte	263.07469	228.03006	122.31976	113.37407	243.53630	234.48618
schrocken	152.47195	160.73702	132.70975	129.64291	130.57360	123.64363
jenbach	99.95830	99.92922	101.66513	100.02312	101.91972	100.24751
loibl	223.65782	219.54906	245.78817	248.76191	260.80859	269.34283
gleisdorf	87.28427	87.08835	91.04844	91.89402	93.47800	93.70448
obergurgl_vent	100.17732	101.78672	135.26549	134.15279	137.99769	137.44852
mariazell	105.41256	106.90912	125.95006	124.57312	123.91560	119.90782
st_michael_ob_bleiburg	91.93190	88.34541	92.52514	90.35443	108.21659	111.76812
preitenegg	106.09356	111.79561	100.59851	104.28865	96.59743	92.55079
lienz	132.80593	132.37760	162.52604	162.06558	201.93221	200.89236
mondsee	134.53296	140.81587	141.73682	139.54555	146.32123	146.43813
irdning_gumpenstein	98.09650	107.60174	117.72176	117.22381	114.04243	111.42417
lobming	140.69940	143.32727	111.87890	107.54772	107.30009	102.23682
kolbnitz	159.93050	160.34202	164.66426	165.50461	174.83158	182.29607
klagenfurt_flughafen	88.73920	88.41334	87.85930	90.46848	87.44616	89.42796
innsbruck_flughafen	128.25170	113.99510	103.72059	103.46214	104.71723	105.03094
laa_an_der_thaya	78.19501	77.56300	98.17277	99.43149	78.53093	80.65772
pabneukirchen	102.27646	112.55566	101.76903	101.82177	97.06760	92.91806
stolzalpe	116.29738	111.27284	111.04648	107.16636	110.59305	110.95214
hieflau	117.29597	119.45920	146.63289	148.27695	118.09162	114.09838
graz_flughafen	103.09181	104.99130	104.92780	105.07452	109.69162	112.21100
reisach	180.18036	186.21651	200.01213	199.97009	222.01464	230.92468
reichersberg	92.19782	94.75838	108.41833	108.93018	111.43004	111.42463
linz_horsching	91.20856	90.58020	94.50148	95.28538	97.16486	98.20793
patscherkofel	72.50824	76.98081	87.01969	87.03184	88.71661	88.68780
bregenz	138.12648	140.78668	148.55651	146.86916	151.74769	148.49195
salzburg_flughafen	138.09368	139.71166	124.70376	123.78706	129.05852	124.48173
st_jakob_im_def	102.56070	104.08848	108.77729	111.18907	107.94290	108.55420
innsbruck_universitat	106.03982	98.20515	87.88578	86.78461	94.31658	97.70291
landeck	73.86536	75.37471	80.06326	80.10650	77.92811	76.23968
bruck_mur	93.45916	92.37071	93.06868	94.73219	94.54702	93.77587
stift_zwettl	87.89523	89.10195	93.99447	92.08324	120.46586	122.41880
worterberg	85.38809	85.92776	99.30360	98.82866	105.55151	106.69571
kufstein	113.45986	107.77676	115.72031	119.23853	104.74126	99.48559
kremsmunster	112.47689	114.05084	117.10725	115.20192	120.48298	120.11125
seckau	102.94085	106.60236	107.39667	106.03191	107.91665	107.32152
feldkirch	124.60258	108.79575	109.84874	108.64594	107.12104	102.48857
bad_ischl	135.48764	136.64180	153.89305	152.57036	150.85345	147.84768
murzzuschlag	103.90835	103.06330	112.36742	115.43323	119.87806	122.05193
kanzelhohe	90.10729	90.86747	92.76524	95.11177	94.98638	98.55267
galtur	89.64610	86.92947	106.19412	106.59298	104.28422	107.66943
eisenstadt	108.61414	117.64338	99.59860	96.88972	99.11922	95.52171
graz_universitat	95.67160	95.87321	101.09765	104.50345	107.29322	110.36976
zeltweg	84.21237	83.50828	98.16097	101.92102	103.26045	108.20455
reichenau_rax	136.49828	122.22378	119.31680	116.08295	130.49539	130.36919
schoppernau	182.88784	201.07901	201.56216	199.77007	141.23810	126.95932
mayrhofen	134.69551	116.11337	105.70759	105.14343	104.85689	103.48741
villacher_alpe	183.46244	180.35080	173.58717	174.59200	170.24827	169.47691
bad_gleichenberg	96.24781	95.07934	104.78424	104.79236	102.15529	99.79220
retz	79.77519	81.93080	83.48828	84.95915	106.20634	106.25304
holzgau	142.39380	134.64780	134.89025	135.96176	128.66152	123.66140
zell_am_see	104.82779	105.09002	91.13250	87.32755	92.26751	89.68931
schockl	98.89625	99.13156	102.67055	103.31695	101.85788	102.55486

```r
# We use the pot_mle_wad data  => column 6
data_tot_geo <- as.geodata(data_tot, coords.col = 8:9, data.col = 6)
```

5.2 Initial plot of the dataset

```r
at <- raster::getData('GADM', country='AUT', level=0) %>% st_as_sf()

at_border <- st_boundary(at)
bnd2 <- at_border[[3]][[1]][[2]]

pred.grid <- expand.grid(seq(9, 18, by=0.02), seq(46, 49, by=0.02))

ggplot() +
  coord_equal() +
  geom_sf(data=at) +
  geom_sf(data = at_border, size = 4, shape = 23) +
  geom_point(data=data_tot,
             aes(x=long, y=lat, color=mle_wad),
             shape=18,
             size=1.4)  +
  xlab("Longitude") + ylab("Latitude") +
  ggtitle("Estimated T-100 rain event at the measurement points") +
  theme_light()
```

5.3 Statistical distribution and transformations

Figure 2 are the statistical distributions of the data and the relative transformations.

5.3.1 Boxcox fitting

The boxcot transformation is useful to find the lambda that brings the data normal. Then, we can easily choose the transformation that is the closest from the lambda value.

```r
par(pty="m" )
boxcox(data_tot_geo)

boxcot <- boxcoxfit(data_tot_geo$data)

# perform boxcot
databx <- boxcoxnc(data_tot$mle_wad, verbose = FALSE) # Find best alpha

data_tot <- as.data.table(data_tot)
data_tot[, databx := databx$tf.data]
# reverse boxcot here
lambda <- databx$lambda.hat
```

We obtain a lambda parameter of -1.85 for the boxcot fit.

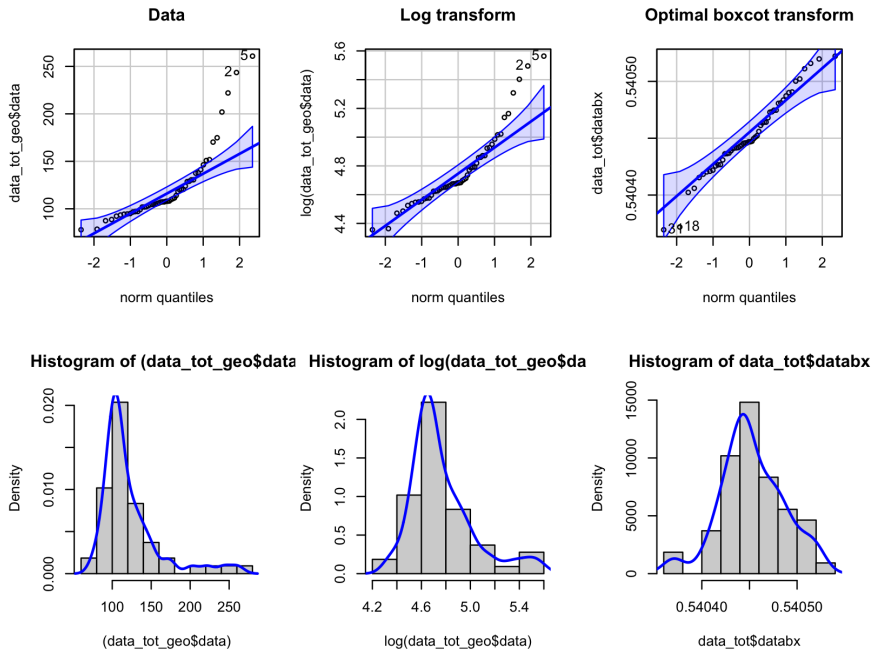

Figure 2: Statistical distribution of the dataset

```
par(mfrow = c(2,3))
qqPlot(data_tot_geo$data)
title("Data")

qqPlot(log(data_tot_geo$data))
title("Log transform")

qqPlot(data_tot$databx)
title("Optimal boxcot transform")


par(pty="s" )
hist((data_tot_geo$data), prob = T)
lines(density((data_tot_geo$data)), col='blue', lwd=2)

hist(log(data_tot_geo$data), prob = T)
lines( density(log(data_tot_geo$data)),col='blue', lwd=2)

hist(data_tot$databx, prob = T)
lines(density(data_tot$databx), col='blue', lwd=2)
```

The original data is particularly not normal on the extreme. After a log transformation, the data is already closer from a normal distribution. The boxcox distribution offers the most normal distribution of the three. Both distribution will be used as the `eyefit` function from the `geoR` package doesn't let the user adjust the parameters when the lambda differs from 0.

5.4 Variogram

The fitting of the variogram has been done with the `eyefit` function from the `geoR` package.

```
# x11()
#
# v.eye1 <- eyefit(variog(data_tot_geo, max.dist = 1.6, uvec = seq(0, 1.6, l = 10), lambda = 0))
# #
# save(v.eye1, file = "variogram_fit.RData")

load("variogram_fit.RData")

vario.fit <- data.frame(Model = "exponential",
                        Sill = 0.062,
                        Phi = 0.61,
                        Nugget = 0.0077,
                        PracticalRange = 1.83)

bin1 <- variog(data_tot_geo, uvec = seq(0, 1.6, l = 10), lambda = 0)
plot(bin1)
lines(v.eye1)
abline(h=var(log(data_tot$mle_wad)), col='red', lt=2)
```

5.5 Kriging

As the `eyefit` function doesn't let the user adjust the parameters when the lambda differs from 0, two distinct kriging will be completed. The first with a lambda = 0 and fit of the variogram and the second one with the optimal lambda but without the correctly fitted variogram.

5.5.1 Kriging with lambda = 0

```
kc <- krige.conv(data_tot_geo, loc = pred.grid, krige = krige.control(obj.m = v.eye1))

summary(kc$predict)

# Create df with pred.grid values and kc$predict

df <- data.frame(
longitude=pred.grid$Var1,
latitude=pred.grid$Var2,
value=kc$predict
)

spg <- df
coordinates(spg) <- ~ longitude + latitude
# coerce to SpatialPixelsDataFrame
gridded(spg) <- TRUE
# coerce to raster
rasterDF <- raster(spg)
# then I crop
rasterDF_crop <- crop(rasterDF, extent(at))
```

```
rasterDF_masked <- mask(rasterDF_crop, at)

df_masked <- raster::as.data.frame(rasterDF_masked,xy=TRUE)
colnames(df_masked) <- colnames(df)


ggplot() +
  geom_tile(data=df_masked, aes(x=longitude, y=latitude, fill=value)) +
  scale_fill_viridis_c(option = "mako", name = "T-100 rain event", na.value = 'white', direction = -1) +
  coord_equal() +
  xlab("Longitude") + ylab("Latitude") +
  geom_sf(data = at, size = 4, shape = 23, inherit.aes = FALSE, fill = NA) +
  geom_point(data=data_tot,
             aes(x=long, y=lat, color="Measurement points"),
             shape=18,
             size=1.4) + scale_color_manual("", values="black") + theme_light() +
  guides(fill = guide_colorbar(order = 1),
              color = guide_legend(order = 2))
```

5.5.2 Kriging with the optimal lambda from boxcot

```
load('variogram_fit_boxcot.RData')

kc <- krige.conv(data_tot_geo, loc = pred.grid, krige = krige.control(obj.m = v.eye1))

summary(kc$predict)

df <- data.frame(
longitude=pred.grid$Var1,
latitude=pred.grid$Var2,
value=kc$predict
)

spg <- df
coordinates(spg) <- ~ longitude + latitude
# coerce to SpatialPixelsDataFrame
gridded(spg) <- TRUE
# coerce to raster
rasterDF <- raster(spg)
# then I crop
rasterDF_crop <- crop(rasterDF, extent(at))
rasterDF_masked <- mask(rasterDF_crop, at)

df_masked <- raster::as.data.frame(rasterDF_masked,xy=TRUE)
colnames(df_masked) <- colnames(df)


ggplot() +
  geom_tile(data=df_masked, aes(x=longitude, y=latitude, fill=value)) +
  scale_fill_viridis_c(option = "mako", name = "T-100 rain event", na.value = 'white', direction=-1) +
  coord_equal() +
```

```
  xlab("Longitude") + ylab("Latitude") +
  geom_sf(data = at, size = 4, shape = 23, inherit.aes = FALSE, fill = NA) +
  geom_point(data=data_tot,
             aes(x=long, y=lat, color="Measurement points"),
             shape=18,
             size=1.4) + scale_color_manual("", values="black") + theme_light() +
  guides(fill = guide_colorbar(order = 1), color = guide_legend(order = 2))
```

5.6 Comparison with the real rain data event - 13-years return period

5.6.1 Real max rain data event

```
rain_data <- read_csv("data/rr_max_eobs.csv")

df_rain <- data.frame(
longitude=rain_data$lon,
latitude=rain_data$lat,
value=rain_data$rr_max
)

spg <- df_rain
coordinates(spg) <- ~ longitude + latitude
# coerce to SpatialPixelsDataFrame
gridded(spg) <- TRUE
# coerce to raster
rasterDF <- raster(spg)
# then I crop
rasterDF_crop <- crop(rasterDF, extent(at))
rasterDF_masked <- mask(rasterDF_crop, at)

df_rain_masked <- raster::as.data.frame(rasterDF_masked,xy=TRUE)
colnames(df_rain_masked) <- colnames(df_rain)


ggplot() +
  geom_tile(data=df_rain_masked, aes(x=longitude, y=latitude, fill=value)) +
  scale_fill_viridis_c(option = "mako", name = "Max rr event", na.value = 'white', direction=-1) +
  coord_equal() +
  xlab("Longitude") + ylab("Latitude") +
  geom_sf(data = at, size = 4, shape = 23, inherit.aes = FALSE, fill = NA) + theme_light() +
  guides(fill = guide_colorbar(order = 1), color = guide_legend(order = 2))

summary(df_rain_masked$value)
```

5.6.2 Data from the HistAlp dataset

```
data_13 <- read_csv("data/T_13_WAD.csv")

data_13 <- inner_join(data_13, stations, by='name')
```

```
data_13_geo <- as.geodata(data_13, coords.col = 5:6, data.col = 3)

# perform boxcot
data13bx <- boxcoxnc(data_13$mle, verbose = FALSE) # Find best alpha

data_13 <- as.data.table(data_13)
data_13[, data13bx := data13bx$tf.data]
# reverse boxcot here
lambda <- data13bx$lambda.hat


# x11()
#
# v.eye1 <- eyefit(variog(data_13_geo, max.dist = 1.6, uvec = seq(0, 1.6, l = 10), lambda = 0))
#
# save(v.eye1, file = "variogram_fit_13.RData")

load("variogram_fit_13.RData")

bin1 <- variog(data_13_geo, uvec = seq(0, 1.6, l = 10), lambda = 0)
plot(bin1)
lines(v.eye1)
abline(h=var(log(data_13$mle)), col='red', lt=2)
```

```
kc <- krige.conv(data_13_geo, loc = pred.grid, krige = krige.control(obj.m = v.eye1))

summary(kc$predict)

# Create df with pred.grid values and kc$predict

df <- data.frame(
longitude=pred.grid$Var1,
latitude=pred.grid$Var2,
value=kc$predict
)

spg <- df
coordinates(spg) <- ~ longitude + latitude
# coerce to SpatialPixelsDataFrame
gridded(spg) <- TRUE
# coerce to raster
rasterDF <- raster(spg)
# then I crop
rasterDF_crop <- crop(rasterDF, extent(at))
rasterDF_masked <- mask(rasterDF_crop, at)

df_masked <- raster::as.data.frame(rasterDF_masked,xy=TRUE)
colnames(df_masked) <- colnames(df)


ggplot() +
  geom_tile(data=df_masked, aes(x=longitude, y=latitude, fill=value)) +
  scale_fill_viridis_c(option = "mako", name = "T-13 rain event", na.value = 'white', direction = -1) +
```

```
  coord_equal() +
  xlab("Longitude") + ylab("Latitude") +
  geom_sf(data = at, size = 4, shape = 23, inherit.aes = FALSE, fill = NA) +
  geom_point(data=data_13,
             aes(x=long, y=lat, color="Measurement points"),
             shape=18,
             size=1.4) + scale_color_manual("", values="black") + theme_light() +
  guides(fill = guide_colorbar(order = 1),
              color = guide_legend(order = 2))
```

References

Asquith, W. H. 2022. *Lmomco—l-Moments, Censored l-Moments, Trimmed l-Moments, l-Comoments, and Many Distributions*.

Gilleland, Eric, and Richard W. Katz. 2016. "extRemes 2.0: An Extreme Value Analysis Package in R." *Journal of Statistical Software* 72 (8): 1–39. https://doi.org/10.18637/jss.v072.i08.

Silva Lomba, Jessica, and Maria Isabel Fraga Alves. 2020. "L-Moments for Automatic Threshold Selection in Extreme Value Analysis." *Stochastic Environmental Research and Risk Assessment* 34 (3): 465–91.

Wadsworth, Jennifer L. 2016. "Exploiting Structure of Maximum Likelihood Estimators for Extreme Value Threshold Selection." *Technometrics* 58 (1): 116–26.