Statistical Modelling of Environmental Extremes
Lab sheet Practical Session 2

Daniela Castro-Camilo · Institute Henri Poincaré, Paris – March 2022

To get started, open R/RStudio and install and load the `ismev` library.

```
# install.packages('ismev')
library(ismev)
```

Trends in GEV models - Ozone data

Task 1

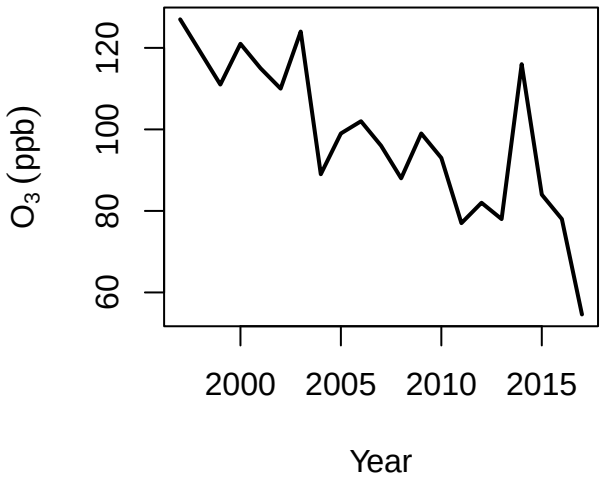
In this task we want to see what happens when we try to fit a GEV distribution to the `ozone.txt` dataset without accounting for the non-stationarity observed in Figure 1 (PS2[1]).

Load the `ozone.txt` dataset:

```
ozone = read.table('ozone.txt', header = T)
x = ozone$year
y = ozone$o3
n = length(y)
```

It is always a good idea to start by plotting the data

```
plot(x, y, type = 'l', lwd = 2, ylab = expression(O[3] ~ (ppb)), xlab = 'Year')
```



We now fit a stationary GEV model using the `ismev::gev.fit` function:

```
fit0 = gev.fit(y)
```

```
## $conv
## [1] 0
##
## $nllh
```

[1] PS2 refers to Practical Session 2 slides.

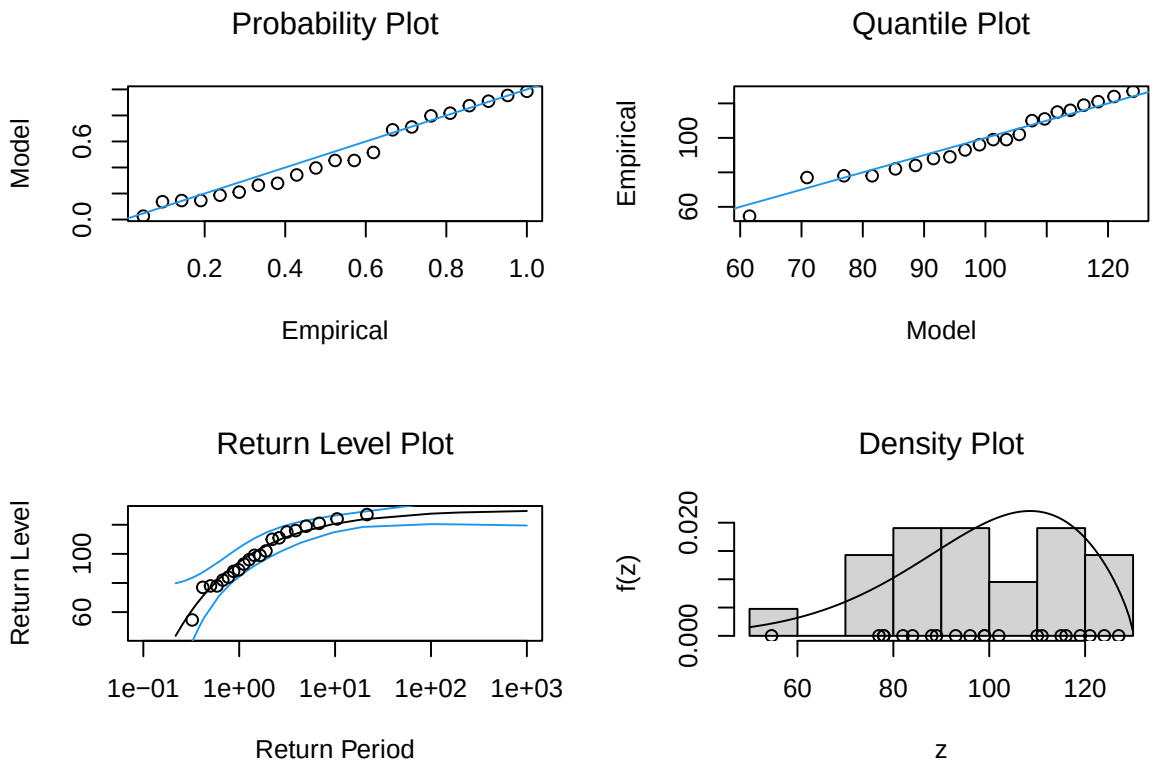
```
## [1] 90.01654
##
## $mle
## [1] 94.4438787 20.6665234 -0.5794478
##
## $se
## [1] 5.0577674 4.3064078 0.2068288
```

Notice the output:

- `$conv` gives a value of zero (in row [1] of the output), which indicates successful convergence, i.e. no errors in fitting.
- `$nllh` shows the negative (maximised) log–likelihood.
- `$mle` shows the maximum likelihood estimates for μ, σ and ξ respectively.
- `$se` gives the associated standard errors for these parameters.

We can investigate the model performance using the in–built diagnostics:

```
par(font.main = 1)
gev.diag(fit0)
```



The function `gev.diag` shows a probability plot, quantile plot, return level (for different return periods) and fitted density plot (with histogram of the observations).

Task 2

Let's fit the non-stationary models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$:

```
# M_1: trend in location but not in scale
ydat = cbind(1:n)
fit1 = gev.fit(y, ydat = ydat, mul = 1, show = F)

# M_2: trend in scale but not in location
fit2 = gev.fit(y, ydat = ydat, sigl = 1, siglink = exp, show = F)
```

```
## Warning in sqrt(diag(z$cov)): NaNs produced
```

The argument `ydat` is a matrix of covariates (by column), which in this case corresponds to the time points $1, \ldots, 21$ (corresponding to years $1997, \ldots, 2017$), The argument `mul` tells `R` which column in that matrix to use for μ and `sigl` does the same for σ. The argument `siglink` indicates the inverse link function for σ (`siglink = exp` because we use the log-link). The `show = F` argument avoids printing details of the fit. We can see that (at least for me!), `fit2` is having issues trying to compute the standard errors of the estimates. Sometimes this can be solved by standardising the covariate:

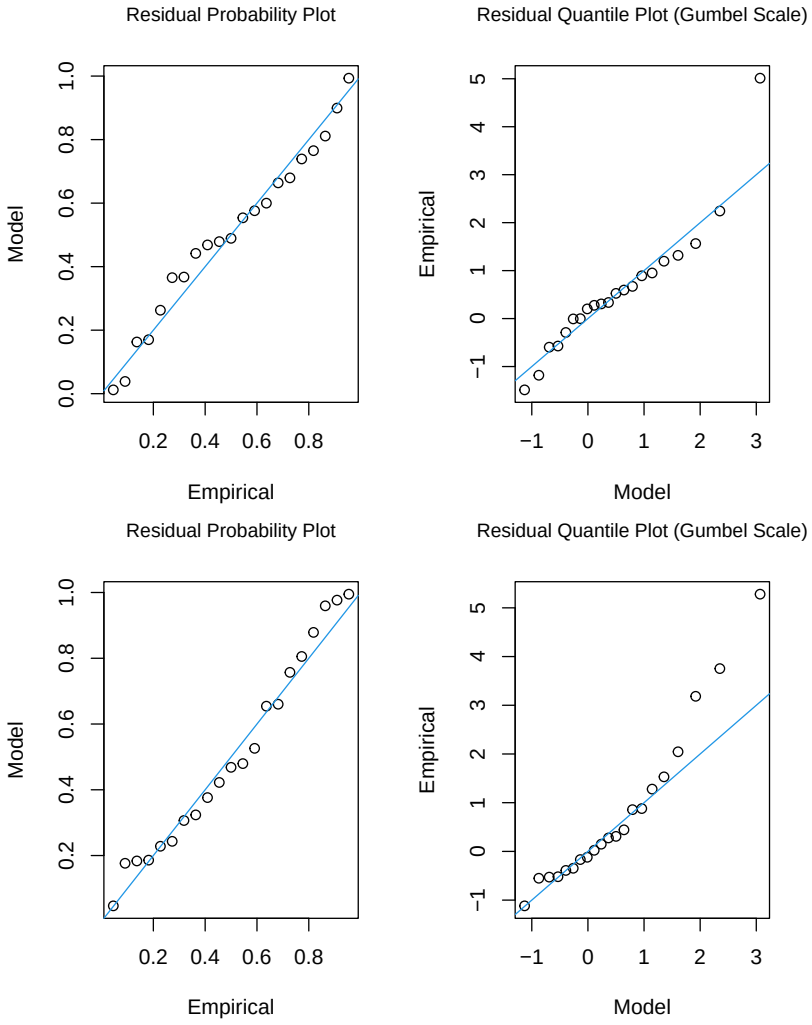
```
# M_2: trend in scale but not in location (standardised covariate)
ydat = scale(ydat)
fit2 = gev.fit(y, ydat = ydat, sigl = 1, siglink = exp, show = F)
```

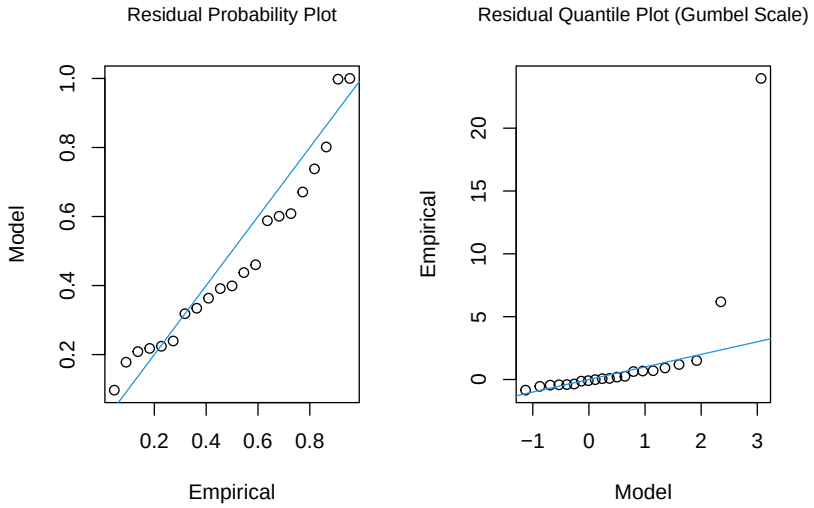
Finally, we fit $\mathcal{M}_3$:

```
# M_3: trend in location and scale
fit3 = gev.fit(y, ydat = ydat, mul = 1, sigl = 1, siglink = exp, show = F)
```

As before, we can investigate the models performance using `gev.diag`:

```
par(cex.main = .9, font.main = 1)
gev.diag(fit1); gev.diag(fit2); gev.diag(fit3)
```





We can see that only probability plots and qq plots are returned. This is because the `ismev` package does not have built-in functions to compute return levels for non-stationary models. We will see how to compute, plot and interpret them in Task 3. For now, let's compare all four models:

```
## Compare using Deviance statistics (chi-squared)
lik0 = -fit0$nllh
lik1 = -fit1$nllh
lik2 = -fit2$nllh
lik3 = -fit3$nllh

delta.01 = 2*(lik1 - lik0)
delta.02 = 2*(lik2 - lik0)
delta.13 = 2*(lik3 - lik1)
delta.23 = 2*(lik3 - lik2)
# Number of parameters in each model ("dimensionality")
dims = c(3,4,4,5)
delta.01>qchisq(0.95, dims[2]-dims[1]) # M_0 rejected, M_1 preferred
```

```
## [1] TRUE
```

```
delta.02>qchisq(0.95, dims[3]-dims[1]) # M_0 not rejected, M_2 excluded
```

```
## [1] FALSE
```

```
delta.13>qchisq(0.95, dims[4]-dims[2]) # M_1 not rejected, M_3 excluded
```

```
## [1] FALSE
```

```
delta.23>qchisq(0.95, dims[4]-dims[3]) # M_2 rejected, M_3 preferred
```

```
## [1] TRUE
```

```
# model 1 seems to be the "best"

## Compare using AIC and BIC
aic = rep(NA, 4)
aic[1] = -2*lik0 + 2*dims[1]
aic[2] = -2*lik1 + 2*dims[2]
aic[3] = -2*lik2 + 2*dims[3]
aic[4] = -2*lik2 + 2*dims[4]

bic = rep(NA, 4)
bic[1] = -2*lik0 + log(n)*dims[1]
bic[2] = -2*lik1 + log(n)*dims[2]
```

```
bic[3] = -2*lik2 + log(n)*dims[3]
bic[4] = -2*lik2 + log(n)*dims[4]

data.frame(aic, bic)
```

```
##        aic      bic
## 1 186.0331 189.1666
## 2 164.2925 168.4706
## 3 185.5633 189.7414
## 4 187.5633 192.7859
```

```
which.min(aic); which.min(bic)
```

```
## [1] 2
```

```
## [1] 2
```

```
# model 1 seems to be the "best"
```

Based on graphical and numerical diagnostics, we choose model $\mathcal{M}_1$.

Task 3

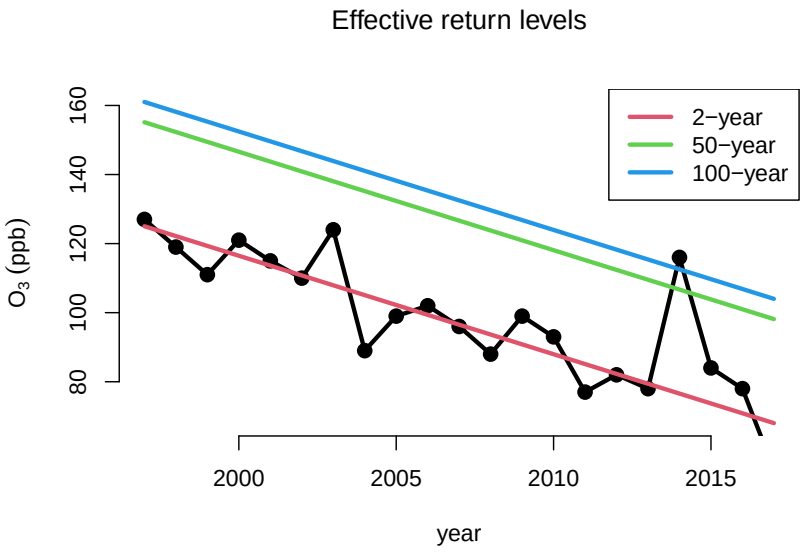
Let's compute effective return levels (ERLs) and reproduce Figure 4 (PS2) based on $\mathcal{M}_1$. First we create a function to compute ERLs for a given time t and return period r following the formula in slide 8 (PS2):

```
# "Manual" computation
zr.gev = function(t, r){
  mu    = fit1$mle[1] + fit1$mle[2]*t
  sigma = fit1$mle[3]
  xi    = fit1$mle[4]
  mu + sigma/xi*((-log(1-1/r))^(-xi)-1)
}
```

Then, we apply our function `zr.gev` to the 21 years of data for differrent return periods ($r = 2, 50, 100$).

```
rl2   = sapply(1:21, zr.gev, r = 2)
rl50  = sapply(1:21, zr.gev, r = 50)
rl100 = sapply(1:21, zr.gev, r = 100)

par(mar = c(5,7,5,1), font.main = 1)
plot(x, y, type = 'l', pch = 16, lwd = 3, ylab = expression(O[3] ~ (ppb)),
     xlab = 'year', axes = F, ylim = range(rl2, rl50, rl100),
     main = 'Effective return levels')
points(x, y, pch = 16, cex = 1.5)
lines(x, rl2, col = 2, lwd = 3)
lines(x, rl50, col = 3, lwd = 3)
lines(x, rl100, col = 4, lwd = 3)
axis(1); axis(2)
legend('topright', c('2-year', '50-year', '100-year'), col = 2:4, lwd = rep(3,3))
```



Seasonality in GEV models - Precipitation data

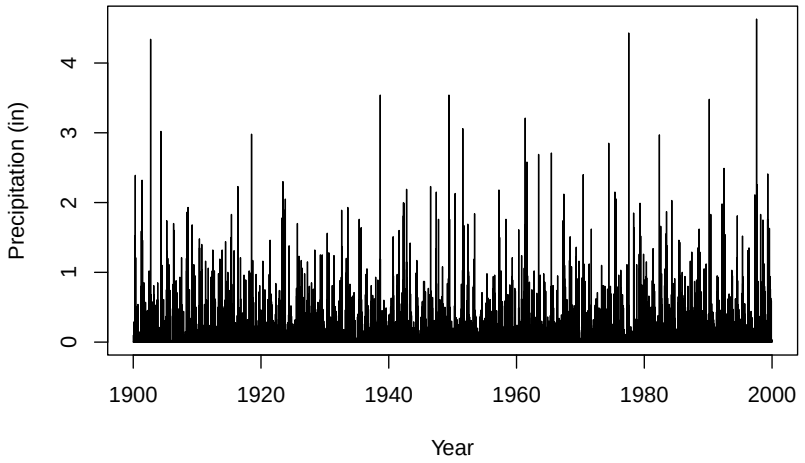
Task 4

In this task we want to fit the model in Equation 3 (PS2) that includes harmonic terms in the GEV parameters to reproduce Figure 5 (PS2). We start by loading and plotting the data:

```
fort = read.table('prcpFC.txt', header = T)
head(fort)
```

```
##   obs tobs month day year Prec
## 1   1    1     1   1 1900    0
## 2   2    2     1   2 1900    0
## 3   3    3     1   3 1900    0
## 4   4    4     1   4 1900    0
## 5   5    5     1   5 1900    0
## 6   6    6     1   6 1900    0
```

```
dates = as.Date(with(fort, paste(year, month, day,sep="-")), "%Y-%m-%d")
plot(dates, fort$Prec, type = 'l', ylab = 'Precipitation (in)', xlab = 'Year')
```



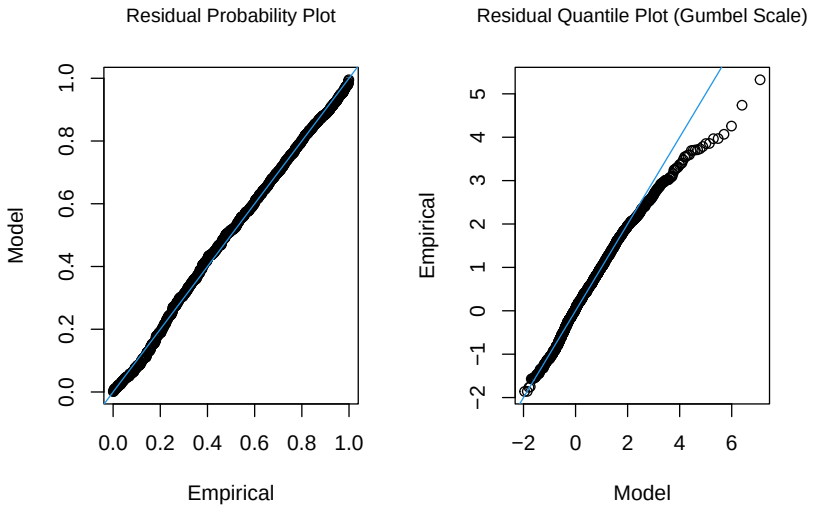
Note that we have the original daily data and therefore we need to compute monthly maxima. One way to do it is using the `Tidyverse` library (another way is using `for` loops):

```
library(tidyverse)
mmax  = fort %>% group_by(year, month) %>%
  summarize(mmax = max(Prec)) %>%
  as.data.frame
mmax$tobs = 1:nrow(mmax)
```

Note that we included the column `tobs` to denote time points $(1, \ldots, 1200)$. Next, we define the covariate matrix and fit the model

```
ydat = matrix(nrow = nrow(mmax), ncol = 2)
n = nrow(mmax)
T = 12
ydat[,1] = sin(2*pi*mmax$tobs/T)
ydat[,2] = cos(2*pi*mmax$tobs/T)

fit = gev.fit(mmax$mmax, ydat = ydat, mul = 1:2, sigl = 1:2, siglink = exp, show = F)
par(cex.main = .9, font.main = 1)
gev.diag(fit)
```

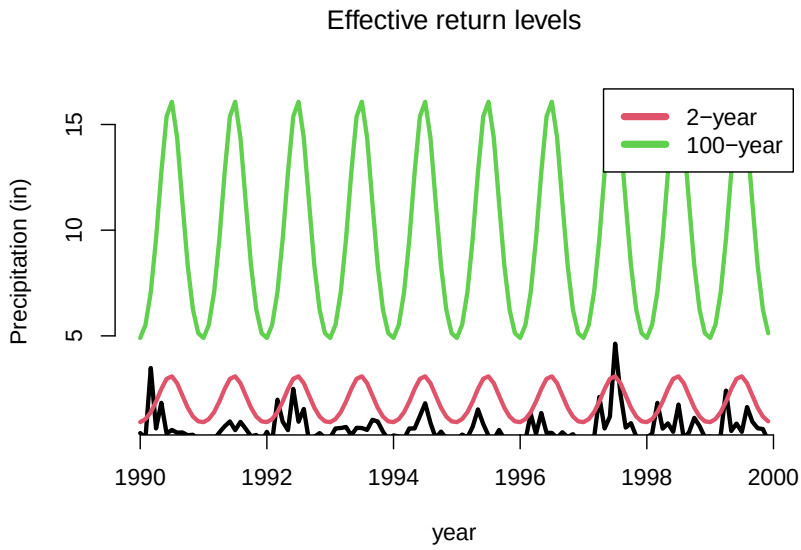


To compute the effective return levels, we need to redefine our function `zr.gev` (or create a new one) because it is model-specific:

```
zr.gev = function(t, r){
  mu    = fit$mle[1] + fit$mle[2]*sin(2*pi*t/T) + fit$mle[3]*cos(2*pi*t/T)
  sigma = exp(fit$mle[4] + fit$mle[5]*sin(2*pi*t/T) + fit$mle[6]*cos(2*pi*t/T))
  xi    = fit$mle[7]
  mu + sigma/xi*((-log(1-1/r))^(-xi)-1)
}
rl2   = sapply(1:n, zr.gev, r = 2*12)
rl100 = sapply(1:n, zr.gev, r = 100*12)
```

Now we are ready to reproduce Figure 5 (PS2):

```
par(mar = c(5,7,5,1), font.main = 1)
dates = zoo::as.yearmon(with(mmax, paste(year, month,sep="-")))
plot(dates[mmax$year>=1990], mmax$mmax[mmax$year>=1990],
     type = 'l', pch = 16, lwd = 3, ylab = 'Precipitation (in)',
     xlab = 'year', axes = F, ylim = range(rl2, rl100), main = 'Effective return levels')
lines(dates[mmax$year>=1990], rl2[mmax$year>=1990], col = 2, lwd = 3, lty = 1)
lines(dates[mmax$year>=1990], rl100[mmax$year>=1990], col = 3, lwd = 3, lty = 1)
axis(1); axis(2)
legend('topright', c('2-year', '100-year'), col = 2:3, lwd = rep(5,2), bg = 'white')
```



Additional material

The `extRemes` library is another powerful tool for EV analysis. It is more complex and complete than `ismev`, allowing us to, for example, compute return levels of non-stationary models, including confidence bands. Here we show how to use the `extRemes` package to fit $\mathcal{M}_1$, reproduce Figures 3 and 4 (PS2) and compare return levels for different years.

We start by loading the data and fitting $\mathcal{M}_1$ again:

```
library(extRemes)
ozone = read.table('ozone.txt', header = T)
x = ozone$year
y = ozone$o3
n = length(y)
ydat = cbind(1:n)
fit1 = gev.fit(y, ydat = ydat, mul = 1, show = F)
```

We fit $\mathcal{M}_1$ using `extRemes` and compare the results (point estimates and confidence intervals) with `fit1`:

```
df = data.frame(time = 1:n, o3 = ozone$o3)
fit1.alt = fevd(o3, data = df, location.fun = ~time, type = 'GEV')
# compare with fit1
fit1$mle; fit1.alt$results$par
```

```
## [1] 124.717046433  -2.850743576   8.644415682  -0.006788701
```

```
##           mu0           mu1         scale         shape
## 124.705992188  -2.849220704   8.644050147  -0.006089197
```

```
fit1$se; sqrt(diag(solve(fit1.alt$results$hessian))) # same
```

```
## [1] 3.8869932 0.3191698 1.4635238 0.1279989
```

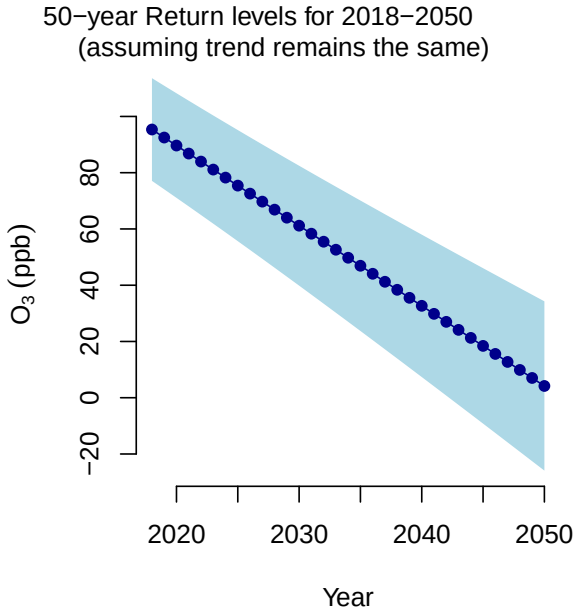
```
##       mu0       mu1     scale     shape
## 3.8852339 0.3189501 1.4627430 0.1284110
```

Compute the 50-year return levels for years 2018-2050 assuming trend remains the same (big assumption):

```
zdat = 22:54
v = make.qcov(fit1.alt, vals = list(mu1 = zdat))
rl50 = return.level(fit1.alt, return.period = 50, qcov = v)
```

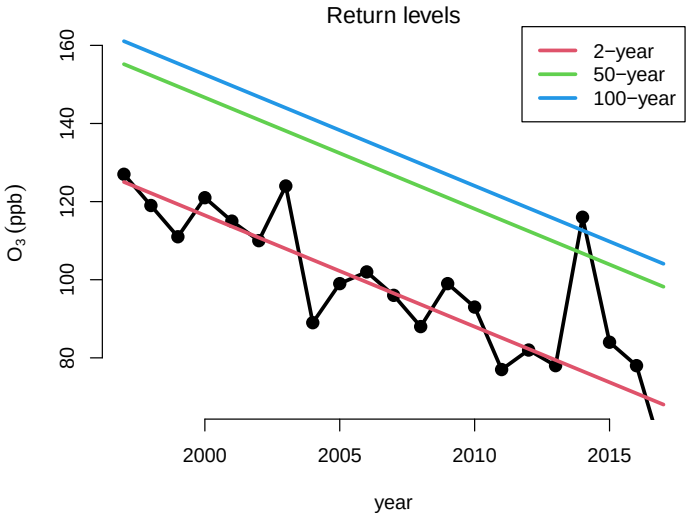
One of the nice thing about `extRemes::return.level` is that it give us confidence intervals based on the normal approximation of the MLEs. We can then reproduce Figure 3 (PS2):

```
rl50 = return.level(fit1.alt, return.period = 50, qcov = v, do.ci = T)
par(mar = c(4,7,5,1.5), pty = 's', font.main = 1)
all.years = ozone$year[1]:2050
plot(all.years[zdat], rl50[,2],
     type = 'l', ylab = expression(O[3] ~ (ppb)),
     xlab = 'Year', ylim = range(rl50[,1:3]),
     main = '', col = 'darkblue', axes = F)
axis(1); axis(2)
polygon(x = c(rev(all.years[zdat]), all.years[zdat]),
        y = c(rev(rl50[,1]), rl50[,3]),
        border = NA, col = 'lightblue')
lines(all.years[zdat], rl50[,2], col = 'darkblue')
points(all.years[zdat], rl50[,2], pch = 16, col = 'darkblue')
mtext('50-year Return levels for 2018-2050
      (assuming trend remains the same)', side = 3, line = -5, outer = TRUE)
```



Compute the effective 2-, 50- and 100-year return levels (Figure 4, PS2):

```
rl2.50.100 = return.level(fit1.alt, return.period = c(2,50,100))
par(mar = c(5,7,1,1), font.main = 1)
plot(x, y, type = 'l', pch = 16, lwd = 3, ylab = expression(O[3] ~ (ppb)),
     xlab = 'year', axes = F, ylim = range(rl2.50.100), main = 'Return levels')
points(x, y, pch = 16, cex = 1.5)
lines(x, rl2.50.100[,1], col = 2, lwd = 3)
lines(x, rl2.50.100[,2], col = 3, lwd = 3)
lines(x, rl2.50.100[,3], col = 4, lwd = 3)
axis(1); axis(2)
legend('topright', c('2-year', '50-year', '100-year'), col = 2:4, lwd = rep(3,3))
```

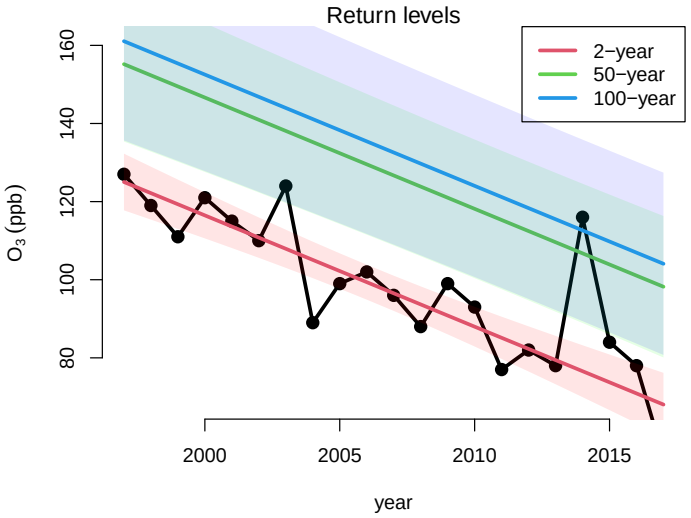


We can also reproduce Figure 4 including confidence bands with the small drawback that we need to compute each return level separately:

```
v     = make.qcov(fit1.alt, vals = list(mu1 = df$time))
rl2   = return.level(fit1.alt, return.period = 2, qcov = v, do.ci = T)
rl50  = return.level(fit1.alt, return.period = 50, qcov = v, do.ci = T)
rl100 = return.level(fit1.alt, return.period = 100, qcov = v, do.ci = T)

par(mar = c(5,7,1,1), font.main = 1)
plot(x, y, type = 'l', pch = 16, lwd = 3, ylab = expression(O[3] ~ (ppb)),
     xlab = 'year', axes = F, ylim = range(rl2.50.100), main = 'Return levels')
points(x, y, pch = 16, cex = 1.5)
# Plot rl2
polygon(x = c(rev(x), x),
        y = c(rev(rl2[,1]), rl2[,3]),
        border = NA, col = adjustcolor( "red", alpha.f = 0.1))
lines(x, rl2[,2], col = 2, lwd = 3)
# Plot rl50
polygon(x = c(rev(x), x),
        y = c(rev(rl50[,1]), rl50[,3]),
        border = NA, col = adjustcolor( "green", alpha.f = 0.1))
lines(x, rl50[,2], col = 3, lwd = 3)
# Plot rl100
polygon(x = c(rev(x), x),
        y = c(rev(rl100[,1]), rl100[,3]),
        border = NA, col = adjustcolor( "blue", alpha.f = 0.1))
lines(x, rl100[,2], col = 4, lwd = 3)

axis(1); axis(2)
legend('topright', c('2-year', '50-year', '100-year'), col = 2:4, lwd = rep(3,3))
```



We can also compare effective return levels. The example below compares the effective 50-year return levels for 1997 ($t = 1$) and 2017 ($t = 21$):

```
v1997 = make.qcov(fit1.alt, vals = list(mu1 = 1))
v2017 = make.qcov(fit1.alt, vals = list(mu1 = 21))
round(return.level(fit1.alt, return.period = 50, do.ci = TRUE,
                   qcov = v2017, qcov.base = v1997), 2)
```

```
## fevd(x = o3, data = df, location.fun = ~time, type = "GEV")
##
## [1] "Normal Approx."
##
##      95% lower CI Estimate 95% upper CI Standard Error
## [1,]       -69.49   -56.98       -44.48           6.38
```

We can see that the estimated difference in effective return levels is about -56.98 (-69.49, -44.48) ppb. Since 0 not contained, this result suggests that the 50-year return level of annual max O_3 has decreased, with statistical significance at the 5% level, on the order of 57 ppb between 1197 and 2017.