MAST20005/MAST90058: Week 3 Lab

Goals: (i) Explore data numerically and graphically; (ii) Getting familiar with probability, density calculations and random number generation; (iii) Parameter estimation and basic numerical optimization in R.

Data: In this lab we will analyse the Tasmanian Rainfall data (`tasmania.csv`), obtained from the Australian Bureau of Meteorology. Observations represent the maximum daily rainfall from 1995 to 2014 recorded by 54 weather stations in Tasmania.

1 Exploratory data analysis

Load the dataset in R by using the command below. **You will need to copy the file to your current working directory, or otherwise set your working directory to the folder with the file (see the previous week's lab notes).**

```
tasmania <- read.csv("tasmania.csv") # load data
dim(tasmania) # check data dimension

## [1] 20 55

names(tasmania[, 1:5]) # names of the first 5 columns

## [1] "Year"                             "Burnie..Round.Hill."
## [3] "Cape.Grim..Woolnorth."            "Wynyard.Airport"
## [5] "Smithton..Upper.Havelock.Street."

year <- tasmania[, 1]   # create a vector for year
s1   <- tasmania[, 2]   # create a vector for station 1 (Burnie)
s2   <- tasmania[, 3]   # create a vector for station 2 (Cape Grim)
```

Explore the distribution of extreme rainfall in Burnie and Cape Grim by computing some basic summary statistics.

```
summary(s1) # 5-number summary plus sample mean

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.00   37.60   47.40   48.84   54.20   87.20

sd(s1)  # sample standard deviation

## [1] 16.46196

IQR(s1) # interquartile range

## [1] 16.6

quantile(s1, type = 6) # sample percentiles using Type 6 approximations

##   0%  25%  50%  75% 100%
## 27.0 36.0 47.4 55.4 87.2
```

```r
quantile(s1, type = 7) # sample percentiles using Type 7 approximations
```

```
##   0%  25%  50%  75% 100% 
## 27.0 37.6 47.4 54.2 87.2
```

```r
summary(s2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    9.60   29.50   36.80   35.86   44.50   62.10       1
```

```r
sd(s2, na.rm = TRUE)
```

```
## [1] 13.24057
```

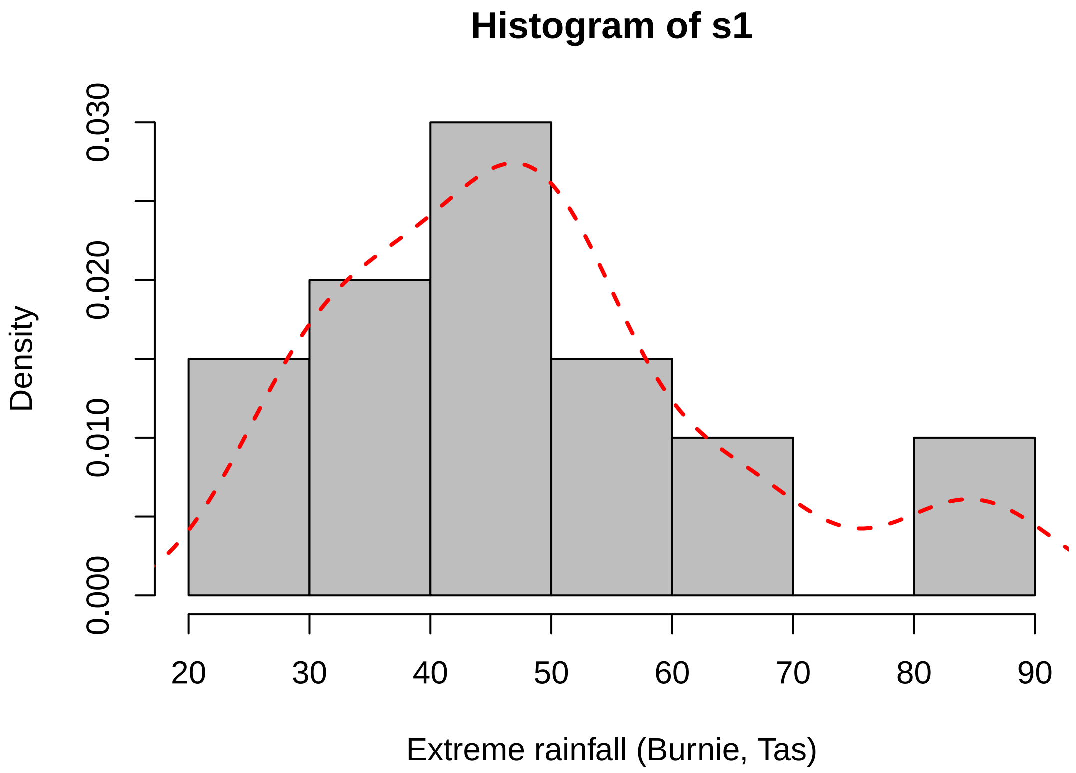
```r
IQR(s2, na.rm = TRUE)
```

```
## [1] 15
```

The sample mean and median suggest that the location for the distribution s1 is larger than that of s2. One observation is missing for s2 (corresponding to year 2014). Note that we need to use the option na.rm = TRUE to remove missing values (shown as NA in R). Measures of spread are obtained by computing the sample interquartile range (IQR) and sample standard deviation. The variability in s2 is slightly smaller than that of s1.

Next, explore the data by graphic summaries. The following gives an annotated density histogram for s1 and a smooth density estimate:

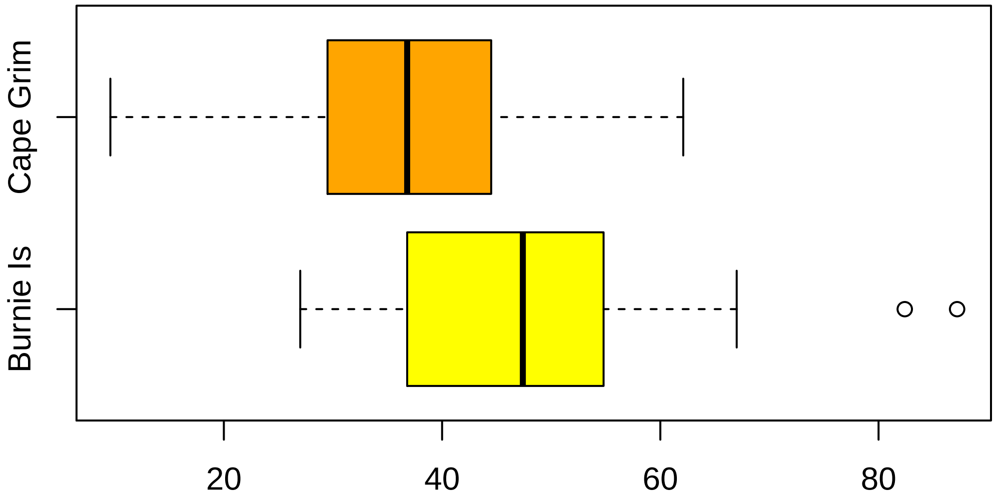
```r
hist(s1, freq = FALSE, xlab = "Extreme rainfall (Burnie, Tas)", col = 8)
smooth.density = density(s1)  # fits a smooth curve
lines(smooth.density, lty = 2, lwd = 2, col = 2)  # draws the curve
```



By adding the argument `nclass` in `hist` you can control the number of bins for the histogram (e.g. try to add the argument `nclass = 15` or `breaks = 15`). Note that this is treated as a suggestion only; R will try to choose a similar number of bins while ensuring that the breakpoints are at ‘pretty’ numbers. From the above plot it is clear that the sample distribution for `s1` is not quite symmetric.

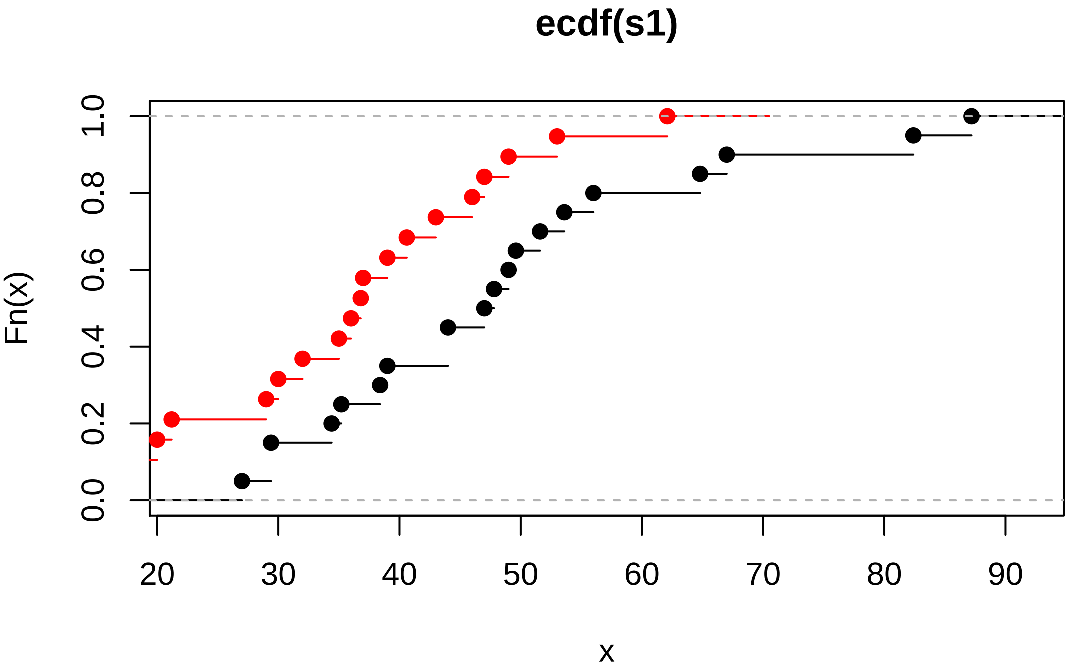
Compare the main features of two or more distributions we can draw multiple boxplots on the same plotting window.

```
boxplot(s1, s2, horizontal = TRUE, names = c("Burnie Is", "Cape Grim"),
        col = c("yellow", "orange"))
```



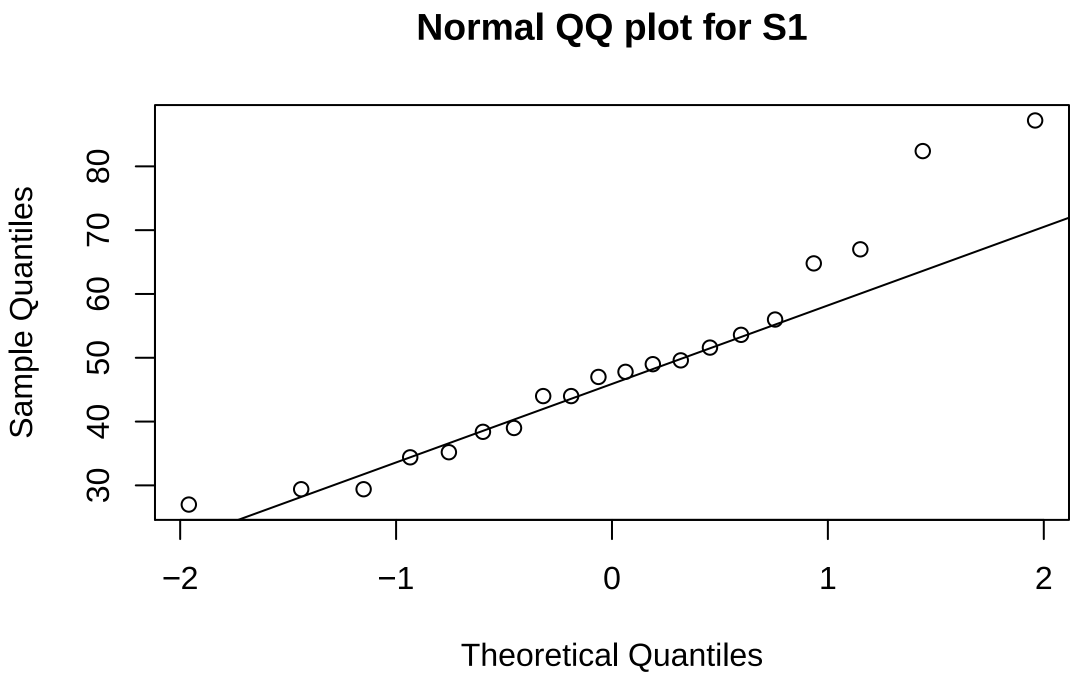
The following computes and plots empirical cdfs:

```r
ecdf1 <- ecdf(s1)
ecdf2 <- ecdf(s2)
plot(ecdf1)
plot(ecdf2, col = 2, add = TRUE)
```



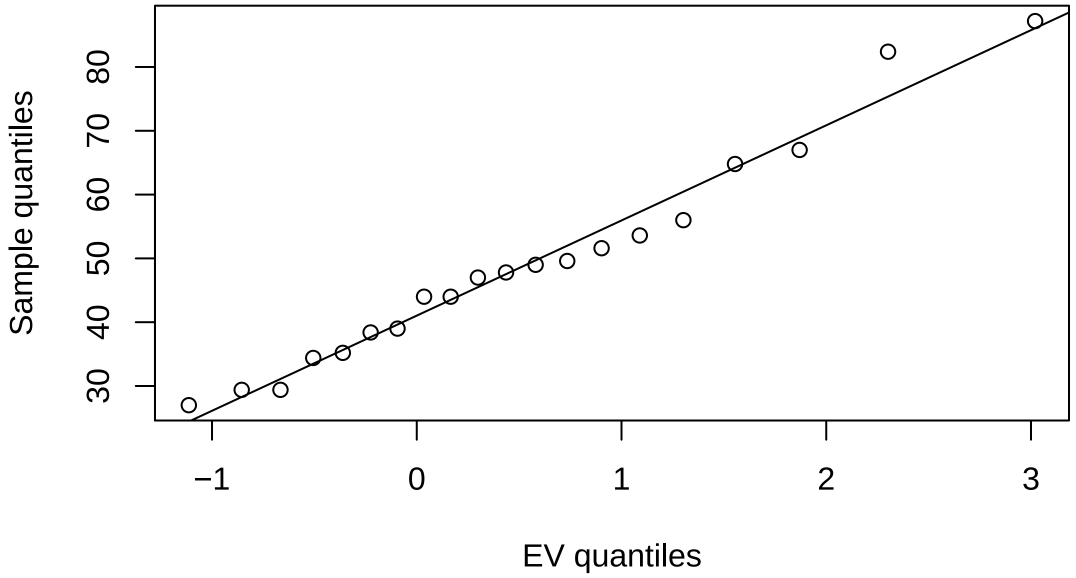
The QQ plot compares the empirical distribution of the data against some theoretical distribution. The following compares the sample distribution with the theoretical normal distribution:

```r
qqnorm(s1, main = "Normal QQ plot for S1")
qqline(s1)
```



Although the central part of the data distribution is compatible with the normality assumption, note that the right tail deviates from the straight line. Probability theory suggests that extremes such as rainfall maxima follow the so-called Extreme Value distribution with inverse cdf $\mu + \sigma F^{-1}(p)$, where $F^{-1}(p) = -\log(-\log(p))$ is the inverse cdf of the standard EV distribution.

```r
Finv <- function(p) {-log(-log(p))}  # quantile function
p <- (1:20) / 21
y <- sort(s1) # order statistics
x <- Finv(p)  # theoretical quantiles
plot(x, y, ylab = "Sample quantiles", xlab = "EV quantiles")
fit <- lm(y ~ x) # this computes the "line of best fit"
                 # (more details later in the semester)
abline(fit)  # this plots the "line of best fit"
```



From the last QQ plot the EV model seems to be more appropriate than the normal model, since the points in EV QQ plot are a little closer to the straight line compared to the previous normal QQ plot. As an exercise, repeat the same analysis for other weather stations.

2 Distributions and random numbers

We look at some of the basic operations associated with probability distributions. There are a large number of probability distributions available, but we only look at a few. If you would like to know what distributions are available you can do a search using the command `help.search("distribution")`.

Here we give details about the commands associated with the normal distribution and briefly mention the commands for other distributions. The functions for different distributions are very similar where the differences are noted below. To get a full list of the distributions available in R you can use the following command:

```
> help(Distributions)
```

For every distribution there are four commands. The commands for each distribution are determined by a prefix indicating the functionality: `d-` returns the value of the probability density function (or probability mass function for discrete data), `p-` returns the cumulative density function, `q-` returns the inverse cumulative density function (quantiles), and `r-` returns randomly generated numbers.

For the commands below, we use the normal distribution, but similar calculation can be applied to other distributions.

Given a set of values `dnorm` returns the value of the pdf at each point. If you only give it the points, it assumes you want to use a mean of zero and standard deviation of one. There are options to use different values for the mean and standard deviation. Try the following:

```r
dnorm(0)

## [1] 0.3989423

dnorm(0) * sqrt(2 * pi)

## [1] 1

dnorm(0, mean = 4, sd = 2)

## [1] 0.02699548

dnorm(c(-1, 0, 1))

## [1] 0.2419707 0.3989423 0.2419707

x <- seq(-20, 20, by = 0.1)
y <- dnorm(x)
plot(x, y)
y <- dnorm(x, mean = 2.5, sd = 0.5)
plot(x, y)
```

The second function we examine is pnorm(). Given a number or a list it computes the probability that a normally distributed random number will be less than that number (i.e. it returns the normal cdf). It accepts the same options as dnorm():

```r
pnorm(0)  # lower tail probability (cdf)

## [1] 0.5

pnorm(1)

## [1] 0.8413447

pnorm(1, lower.tail = FALSE)  # upper tail probability

## [1] 0.1586553

pnorm(0, mean = 2, sd = 3)

## [1] 0.2524925

x <- seq(-20, 20, by = 0.1)
y <- pnorm(x)
plot(x, y)
y <- pnorm(x, mean = 3, sd = 4)
plot(x, y)
```

The next function we look at is qnorm() which is the inverse of pnorm(). The idea behind qnorm() is that you give it a probability, and it returns the number whose cumulative distribution matches the probability. For example, try:

```
qnorm(c(0.25, 0.5, 0.75), mean = 1, sd = 2) # quartiles for N(1,2)

## [1] -0.3489795  1.0000000  2.3489795

x <- seq(0, 1, by = 0.05)
y <- qnorm(x)
plot(x, y)
y <- qnorm(x, mean = 3, sd = 2)
plot(x, y)
y <- qnorm(x, mean = 3, sd = 0.1)
plot(x, y)
```

The last function we examine is the `rnorm()` function which can generate random numbers whose distribution is normal. The argument that you give it is the number of random numbers that you want, and it has optional arguments to specify the mean and standard deviation:

```
rnorm(4)

## [1] -0.08676448  0.62984032 -1.36913780  1.06512790

rnorm(4, mean = 3, sd = 3)

## [1] 2.205825 4.538623 4.304493 5.650095

rnorm(4, mean = 3, sd = 3)

## [1] -2.317260  4.099793 -2.630690  2.899554

y <- rnorm(200)
hist(y)
y <- rnorm(200, mean = -2)
hist(y)
qqnorm(y)
qqline(y)
```

3 Maximum likelihood estimation

In this task, we will look at maximum likelihood estimation. In simple cases the MLE is available in closed form. For example, if observations were iid from $\mathrm{N}(\mu, \sigma^2)$, then we already know that the MLE for μ and σ^2 are $\hat{\mu} = \bar{X}$ and $\hat{\sigma}^2 = (n-1)S^2/n$. Thus, the ML estimates for the extreme rainfall data can be quickly computed as:

```
mu.hat = mean(s1)
mu.hat

## [1] 48.84

n = length(s1)
sigma.hat = sqrt((n - 1) * var(s1) / n)
sigma.hat

## [1] 16.04514
```

Probability theory says that a better model for maxima is the extreme value (EV) distribution, or Gumbel distribution, with pdf

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma} \exp\left\{-\frac{x-\mu}{\sigma}\right\} \exp\left[-\exp\left\{-\frac{x-\mu}{\sigma}\right\}\right], \quad x \in \mathbb{R},\, \mu \in \mathbb{R},\, \sigma > 0.$$

The log-likelihood function for this model is,

$$\ell(\mu, \sigma) = -n \log \sigma - \sum_i \frac{x_i - \mu}{\sigma} - \sum_i \exp\left\{-\frac{x_i-\mu}{\sigma}\right\}.$$

Unfortunately, as it is often the case in practice, there is no closed form solution for the MLE and maximisation has to be carried out numerically.

There are number of routines available to maximize the log-likelihood function. One simple approach is given by the function `fitdistr()` in the `MASS` package, which can be used to fit a number of models (see the help page for `fitdistr()` to see the available distributions). Since the Gumbel distribution is not in the default list of distributions, we must write its pdf before using `fitdistr()`.

```r
library(MASS)

# Prepares the Gumbel pdf:
dgumbel <- function(x, mu, sigma)
    exp((mu - x) / sigma - exp((mu - x) / sigma)) / sigma
# Fits the Gumbel distribution
gumbel.fit <- fitdistr(x = s1, densfun = dgumbel,
                       start = list(mu = 50, sigma = 10))
gumbel.fit

##        mu          sigma
##   41.542604   12.326253
##  ( 2.899086) ( 2.205542)
```

Note that `fitdistr()` requires 3 main arguments: data, a pdf (or pmf) and a starting point to use for the optimisation algorithm. Choosing suitable starting parameter values is crucial to ensure a good solution. If preliminary estimates are available you should consider them as starting points.

For the Normal, log-Normal, geometric, exponential and Poisson distributions, `fitdistr()` uses closed-form MLEs. For example:

```r
normal.fit <- fitdistr(x = s1, densfun = "normal")
normal.fit

##       mean          sd
##   48.840000   16.045136
##  ( 3.587802) ( 2.536959)
```

For all other distributions, `fitdistr()` maximises the likelihood function numerically using the optimisation routine `optim()`.

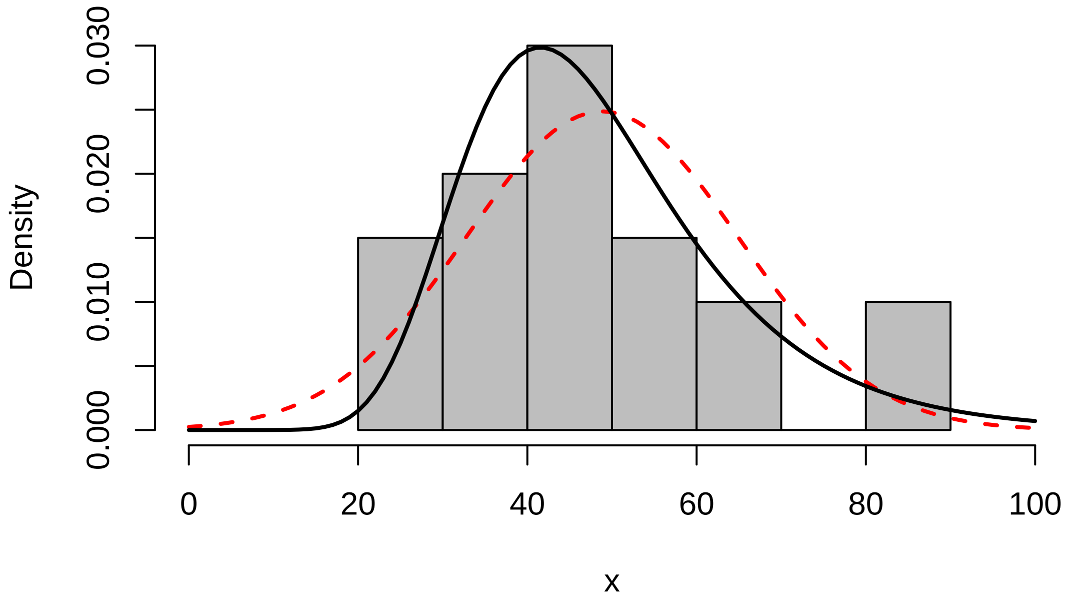
In practical applications, we frequently write the negative log-likelihood from scratch and directly use `optim()` or some other optimisation routine, depending on the likelihood at hand. Most numerical optimisation routines assume that you want to minimise a function. Note that maximising the log-likelihood ℓ is equivalent to minimising the negative log-likelihood function $-\ell$.

```
neg.llk <- function(theta) { # negative log-likelihood
    mu    <- theta[1]
    sigma <- theta[2]
    out   <- -sum(log(dgumbel(s1, mu, sigma)))
    return(out)
}
fit <- optim(c(50, 10), neg.llk) # fits MLEs
theta.hat <- fit$par # returns estimates
theta.hat

## [1] 41.53789 12.32450
```

The arguments in `optim()` are the vector of starting values and the function to be minimised. A number of other tuning parameters can be used depending on the optimisation method specified in `optim()` (thus the slightly different result compared to that from `fitdistr()` is due to a different convergence tolerance definition). The function returns a number of useful outputs, including final parameter estimates and value of the minimised negative log-likelihood, and information about convergence (to see them type `fit`). A detailed explanation can be found in `Value` in `help(optim)`. Various classic optimisation algorithms are available within this function (see `Details` and references in `help(optim)`). When the parameters are known to be in some interval, optimisation can be carried out using the method `method = "L-BFGS-B"` for `optim()`, or the function `nlminb()`, which allow us to specify upper and lower bound for each parameter.

Next, carry out a visual check on the fitted models. The following plots the fitted normal and Gumbel models over the histogram of the data:

```
# Write fitted normal and Gumbel pdfs
pdf1 <- function(x) dnorm(x, mean = mu.hat, sd = sigma.hat)
pdf2 <- function(x) dgumbel(x, mu = theta.hat[1], sigma = theta.hat[2])

# Plot data and fitted models
hist(s1, freq = FALSE, col = "gray", main = NULL, xlab = "x", xlim = c(0,100))
curve(pdf1, from = 0, to = 100, col = 2, lty = 2, lwd = 2, add = TRUE)
curve(pdf2, from = 0, to = 100, col = 1, lty = 1, lwd = 2, add = TRUE)
```



From the plot, the Gumbel distribution appears to be a better fit for the data compared to the normal distribution.

4 Method of moments estimation

In this task we compute point estimates using the method of moments (MM). Assume that the data are generated by $X \sim \text{Gumbel}(\mu, \sigma)$ with pdf given in the previous section. The first two moments of this distribution are:

$$\mathbb{E}(X) = a\sigma + \mu$$
$$\mathbb{E}(X^2) = b\sigma^2 + 2a\sigma\mu + \mu^2$$

where $a = 0.577215$ and $b = 1.978$. Let $\bar{X} = \sum_i X_i/n$ and $M_2 = \sum_i X_i^2/n$ be the first two sample moments. The method of moment estimators for μ and σ are:

$$\widetilde{\sigma} = \sqrt{\frac{M_2 - \bar{X}^2}{b - a^2}}$$
$$\widetilde{\mu} = \bar{X} - a\widetilde{\sigma}$$

Compute estimates for the extreme rainfall data recorded at the Burnie station:

```
x.bar  <- mean(s1)
x2.bar <- mean(s1^2)
a <- 0.577215
b <- 1.978
sigma.tilde <- sqrt((x2.bar - x.bar^2) / (b - a^2))
sigma.tilde

## [1] 12.51076
```

```
mu.tilde <- x.bar - a * sigma.tilde
mu.tilde

## [1] 41.6186
```

Note that the MM estimates are quite close to the maximum likelihood estimates obtained previously (above), but the MM estimates are much quicker to compute. How could you compare the accuracy of the two estimators?

Exercises

1. Compare the maximum rainfall in Marrawah, Low Head and Dover. Do this both numerically and graphically.

2. Create a box plot the compares the maximum rainfall from all 54 weather stations. (Hints: you might need to consult the in-built documentation for the `boxplot()` function; remember that a data frame is also a list.)

3. Which two stations have very different rainfall measurements to the others? Can you explain why that might be? (What is special about these two stations?) Plot histograms for both of these stations. Fit the Gumbel distribution to the data for these two stations, using maximum likelihood.

4. Use a simulation to explore the bias and variance of the MM estimators for the Gumbel distribution. (Hint: the R package `evd` has a function `rgumbel()` which is useful here.)

5. Do question 1b(ii) from the tutorial problems. In addition to calculating the MLE, also plot the likelihood function.

6. See question 1c(iii) from the tutorial problems. Plot the log-likelihood function for the sample given in the hint. Now do it for a larger sample (generate your own data).

7. See question 3c from the tutorial problems. For the X sample, verify the given statistics and plot the likelihood function. Calculate the MLE using the formula derived in that question. Also calculate the MLE by using `fitdistr()`.

8. See question 4 from the tutorial problems. Plot the pdf for $\theta = 3$, using the already available function in R for the pdf of an exponential distribution (do not define your own). Do some simulations to verify that $\bar{X}$ is unbiased for θ and that its variance is θ^2/n (do this for $n = 20$ and $\theta = 3$). For the given sample, calculate the MLE by using `fitdistr()`.