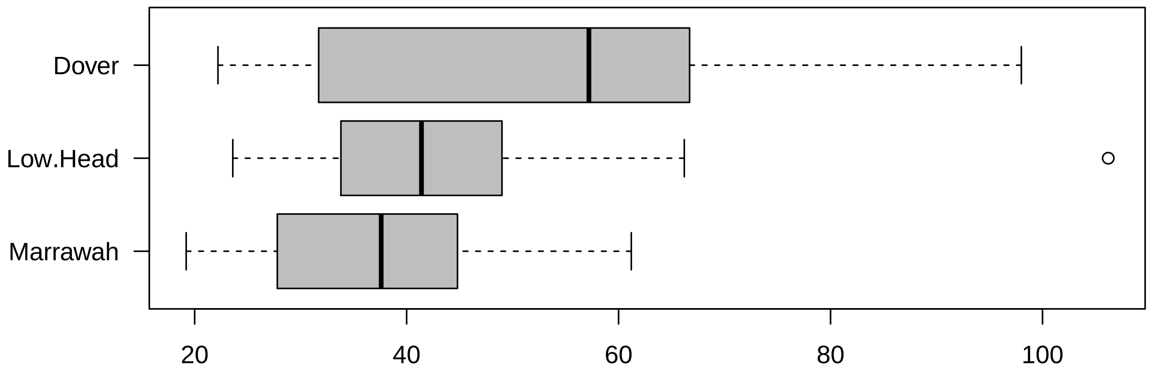
MAST20005/MAST90058: Week 3 Lab Solutions

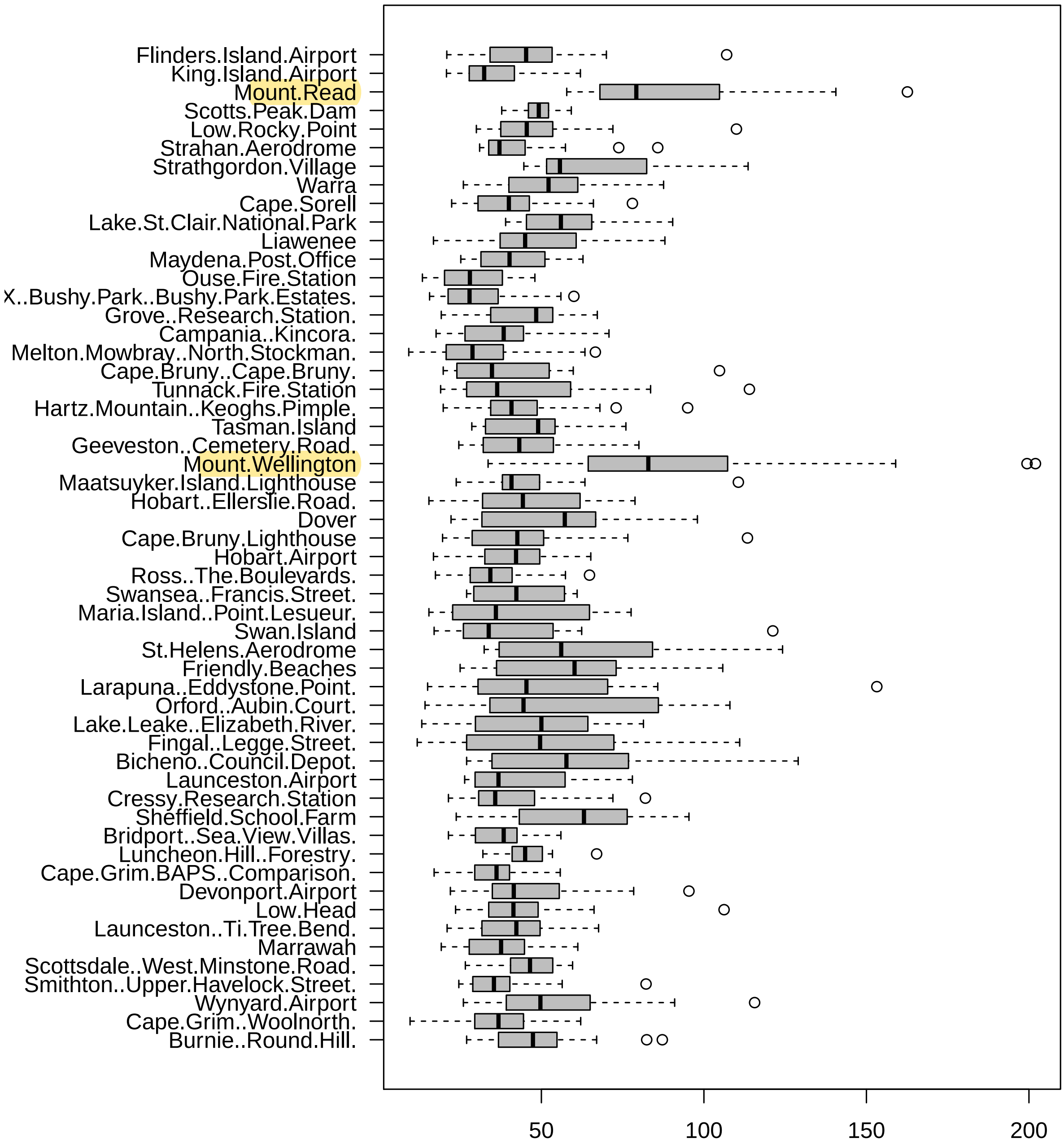
```r
library(MASS)
library(evd)
tasmania <- read.csv("tasmania.csv")
```

1.
```r
summary(tasmania[, c(7, 9, 30)])

##     Marrawah         Low.Head           Dover
##  Min.   :19.20   Min.   : 23.60   Min.   :22.20
##  1st Qu.:28.50   1st Qu.: 33.80   1st Qu.:33.55
##  Median :37.60   Median : 41.40   Median :57.20
##  Mean   :37.85   Mean   : 45.31   Mean   :52.73
##  3rd Qu.:42.10   3rd Qu.: 49.00   3rd Qu.:66.65
##  Max.   :61.20   Max.   :106.20   Max.   :98.00
##                  NA's   :3

par(mar = c(4, 6, 1, 1))  # adjust margins to fit axis labels
boxplot(tasmania[, c(7, 9, 30)], col = 8, horizontal = TRUE, las = 1)
```



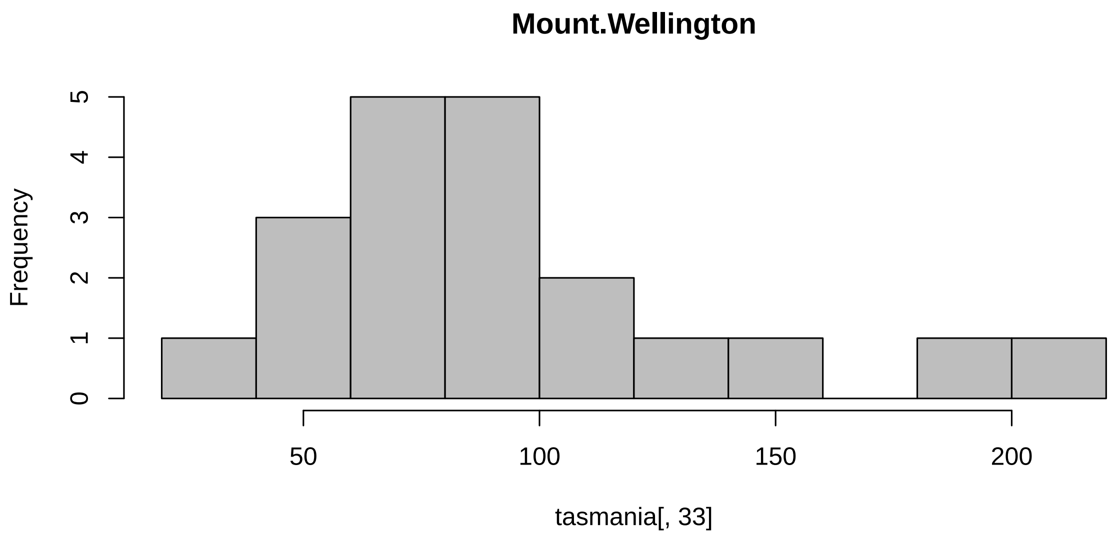
2. The data frame `tasmania` is also a list, which means we can pass it directly to `boxplot()`. However, we should omit the first column, since these are dates rather than rainfall measurements.

```r
par(mar = c(4, 14, 1, 1))
boxplot(tasmania[, -1], col = 8, horizontal = TRUE, las = 1)
```

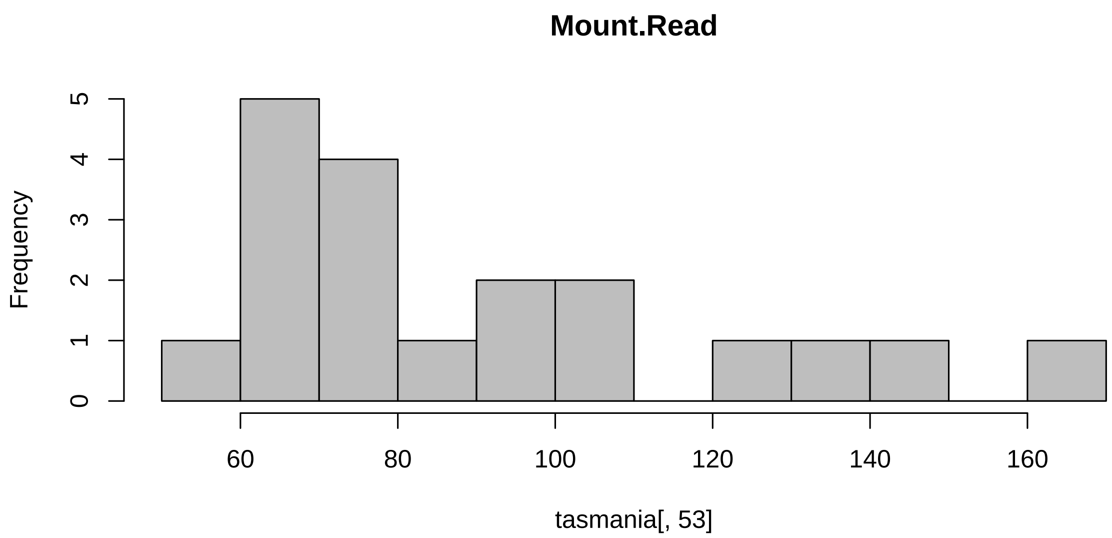


3. Mount Wellington and Mount Read have much higher maximum rainfall than the other stations. Perhaps these are the only two mountainous stations in this dataset?

```
hist(tasmania[, 33], 10, col = 8, main = names(tasmania)[33])
```




```
hist(tasmania[, 53], 10, col = 8, main = names(tasmania)[53])
```



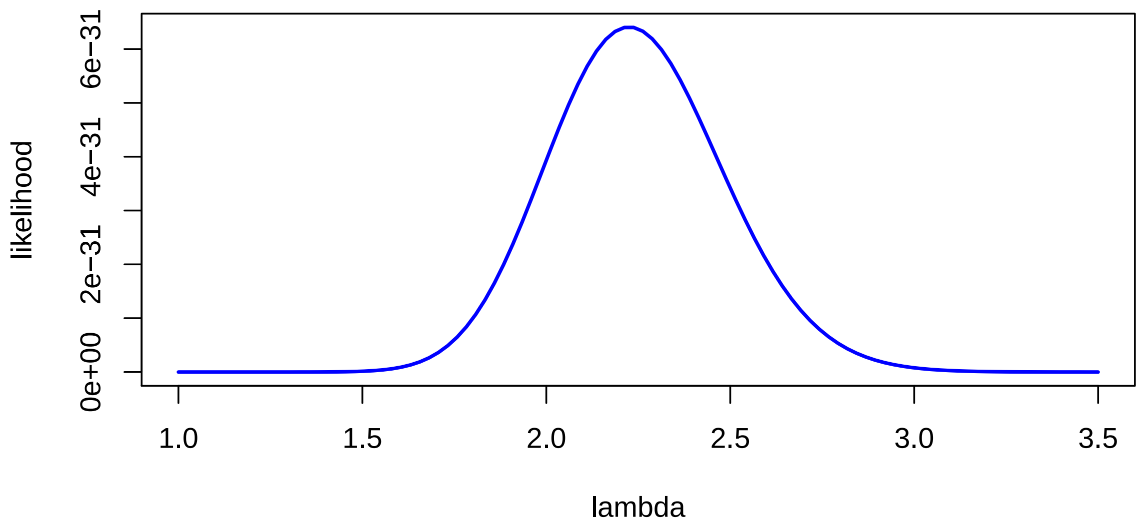

```
dgumbel <- function(x, mu, sigma)
    exp((mu - x) / sigma - exp((mu - x) / sigma)) / sigma
start1 <- list(mu = 50, sigma = 10)
fitdistr(tasmania[, 33], densfun = dgumbel, start = start1)

##        mu          sigma
##   74.389990   32.104613
##  ( 7.519231) ( 5.825075)

fitdistr(tasmania[, 53], densfun = dgumbel, start = start1)

## Error in fitdistr(tasmania[, 53], densfun = dgumbel, start = start1):  'x'
contains missing or infinite values
```

The Mount Read data is missing the first value. Let's just omit it and do the MLE calculation:

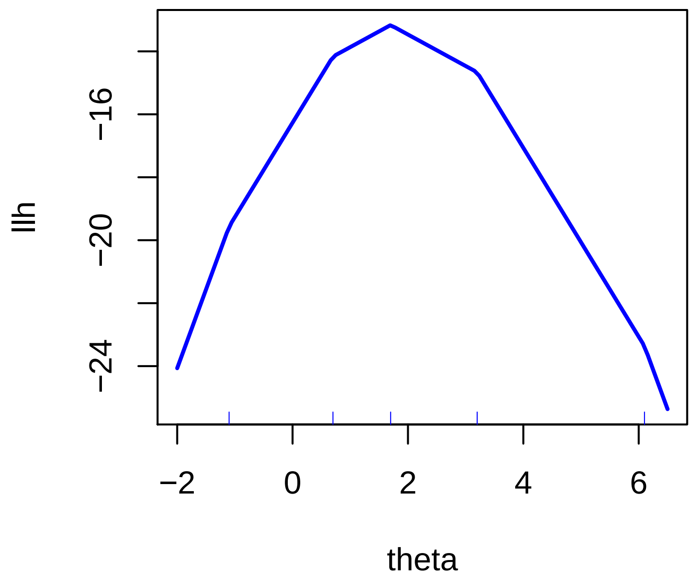
```
fitdistr(tasmania[-1, 53], densfun = dgumbel, start = start1)

## Warning in log(dens(parm, ...)):  NaNs produced

##        mu          sigma
##   78.659729   20.726543
##   ( 4.976391) ( 4.026766)
```

4.
```
a <- 0.577215
b <- 1.978
sigma.mm <- function(x)
    sqrt((mean(x^2) - mean(x)^2) / (b - a^2))
mu.mm <- function(x)
    mean(x) - a * sigma.mm(x)

B <- 10000  # number of simulated samples

# Initialise vectors to hold estimates from the simulations.
sigma.mms  <- numeric(B)
mu.mms     <- numeric(B)

# Run simulations.
for (i in 1:B) {
    x <- rgumbel(20, 50, 10)
    sigma.mms[i]  <- sigma.mm(x)
    mu.mms[i]     <- mu.mm(x)
}

# Evaluate simulation results.
mm.mean <- c(mean(mu.mms), mean(sigma.mms))
mm.var  <- c( var(mu.mms),  var(sigma.mms))
true <- c(50, 10)
bias.estimate <- mm.mean - true
mm.mean

## [1] 50.235427  9.477394

bias.estimate

## [1]  0.2354274 -0.5226064

mm.var

## [1] 5.699199 4.629722
```

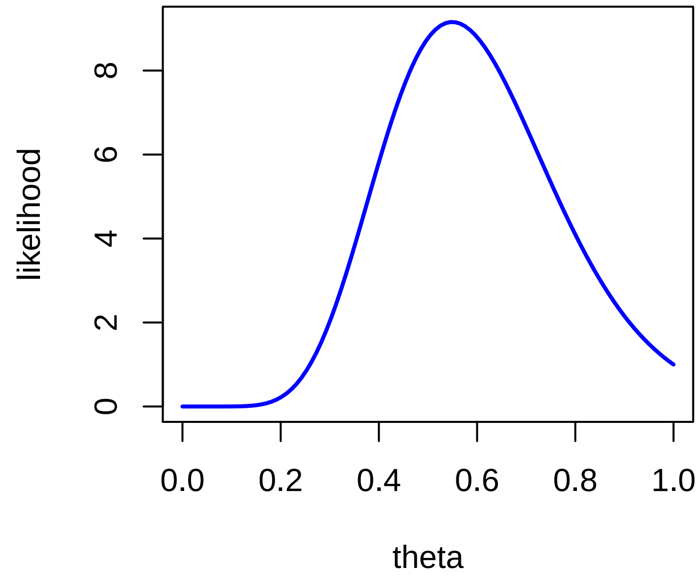
5.
```r
d <- rep(0:6, c(5, 7, 12, 9, 5, 1, 1))  # create vector with the data
d

##  [1] 0 0 0 0 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3
## [34] 4 4 4 4 4 5 6

table(d)  # table of counts

## d
##  0  1  2  3  4  5  6
##  5  7 12  9  5  1  1

mean(d)  # MLE

## [1] 2.225

lambda <- seq(1, 3.5, length.out = 100)
likelihood <- numeric(100)
for (i in 1:100)
    likelihood[i] <- prod(dpois(d, lambda[i]))
plot(lambda, likelihood, type = "l", lwd = 2, col = 4)
```



6.
```r
d <- c(6.1, -1.1, 3.2, 0.7, 1.7)  # data from the hint
theta <- seq(-2, 6.5, length.out = 100)
llh <- numeric(100)
for (i in 1:100)
    llh[i] <- -length(d) * log(2) - sum(abs(d - theta[i]))
plot(theta, llh, type = "l", lwd = 2, col = 4)
rug(d, col = 4)  # plot the location of data points, on the x-axis
```

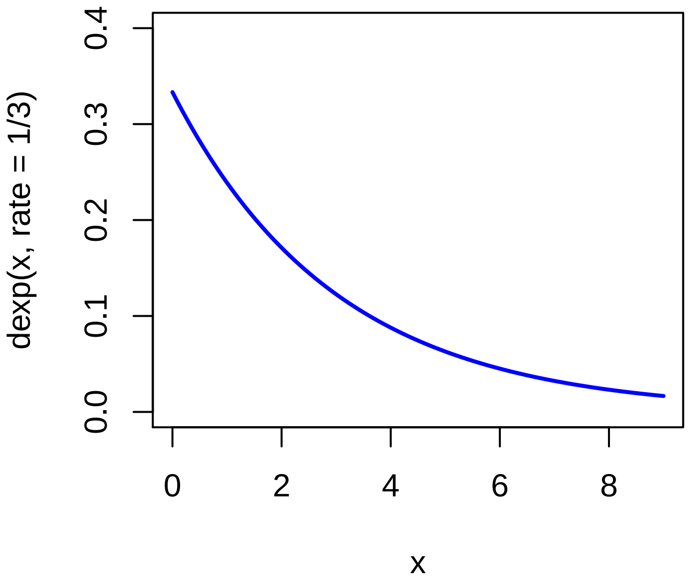


7.
```r
x <- c(0.0256, 0.3051, 0.0278, 0.8971, 0.0739,
       0.3191, 0.7379, 0.3671, 0.9763, 0.0102)
sum(log(x))

## [1] -18.2063

sum(x)

## [1] 3.7401

ddistr <- function(x, theta)
    theta * x^(theta - 1)
theta <- seq(0, 1, length.out = 100)
likelihood <- numeric(100)
for (i in 1:100)
    likelihood[i] <- prod(ddistr(x, theta[i]))
plot(theta, likelihood, type = "l", lwd = 2, col = 4)
```




```
-length(x) / sum(log(x))  # MLE using formula

## [1] 0.5492604

fitdistr(x, densfun = ddistr, start = list(theta = 0.5))

## Warning in stats::optim(x = c(0.0256, 0.3051, 0.0278, 0.8971, 0.0739, 0.3191,
:  one-dimensional optimization by Nelder-Mead is unreliable:
## use "Brent" or optimize() directly

##       theta
##   0.5492188
##  (0.1736776)

# Don't worry about the above warning.
# The optimisation is working fine in this case.
```

8.
```r
curve(dexp(x, rate = 1/3), 0, 9, ylim = c(0, 0.4), lwd = 2, col = 4)
```




```r
B <- 10000
xbars <- numeric(B)
for (i in 1:B) {
    x <- rexp(20, rate = 1/3)
    xbars[i] <- mean(x)
}
c(mean(xbars), var(xbars))  # estimates from simulations

## [1] 2.9987859 0.4577881

c(3, 3^2 / 20)  # true values

## [1] 3.00 0.45

fitdistr(c(3.5, 8.1, 0.9, 4.4, 0.5), "exponential")

##       rate
##   0.2873563
##  (0.1285097)

1 / 0.2873563  # MLE for theta rather than the rate parameter

## [1] 3.48
```