Assessment

Laura Lázaro Soraluce

April 5, 2025

1 Exercise 1

The file Olympics100m.csv (available in PRADO) consists of a sample of n = 50 athletes with the following variables: YEAR, NAME, TIME, Country and Gender. Download the file from PRADO and solve the following:

- Import the data into a data frame with name olympics. When doing it, take into account that Country and Gender should be registered as factors. Print the structure of the data frame that you have created.

```
olympics<-read.csv('Olympics100m.csv')
# olympics <-read.csv('Olympics100m.csv', stringAsFactors=TRUE)
# Putting stringAsFactors=TRUE would also put the field name as a factor.

olympics$Country <- as.factor(olympics$Country)
olympics$Gender <- as.factor(olympics$Gender)


str(olympics)

## 'data.frame': 50 obs. of  5 variables:
##  $ YEAR   : int  1896 1900 1904 1906 1908 1912 1920 1924 1928 1932 ...
##  $ NAME   : chr  "Tom Burke" "Frank Jarvis" "Archie Hahn" "Archie Hahn" ...
##  $ TIME   : num  12 11 11 11.2 10.8 ...
##  $ Country: Factor w/ 15 levels "AUS","BLR","CAN",..: 15 15 15 15 12 15 15 5 3 15 ..
##  $ Gender : Factor w/ 2 levels "female","male": 2 2 2 2 2 2 2 2 2 2 ...
```

- Writing just one sentence, find out the number of athletes from each country.

```
table(olympics$Country)

##
## AUS BLR CAN FRG GBR GDR GER GRE JAM NED POL SAF TRI URS USA
##   2   1   2   1   3   1   1   1   6   1   1   1   1   2  26
```

- Writing just one sentence, compute the 0.05 quantile of the variable TIME separately for each level of the factor Gender.

```
tapply(olympics$TIME, olympics$Gender, quantile, 0.05)

## female   male
## 10.710  9.738
```

- Add a new column in the data frame with the normalised TIME values, scaled to a range between 0 and 1.

```
normalize<-function(x){
  return ((x-min(x))/(max(x)-min(x)))
}
olympics$norm_time<-normalize(olympics$TIME)
```

- Create a new data frame with name USA consisting of the data for the USA athletes.

```
USA<-subset(olympics, Country=='USA')
```

2 Exercise 2

The Beta distribution, with shape parameters $a > 0$ and $b > 0$, is a continuous distribution with probability density function:

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, \quad 0 < x < 1$$

where $\Gamma(\cdot)$ is the Gamma function.
The maximum likelihood estimators for the parameters a and b of the Beta distribution do not have closed-form expressions. Instead, they must be computed numerically. Using the sample data below, derive the maximum likelihood estimators of the Beta distribution, using appropriate tools in R including, if possible, those described in the worksheet 4. To learn more about the Beta distribution, you can refer to the documentation of the dbeta function and the references cited there.

```
values<-scan(text= "0.43 0.29 0.47 0.61 0.37 0.25 0.37 0.27 0.22 0.45
0.21 0.05 0.21 0.47 0.55 0.25 0.28 0.38 0.09 0.24
0.46 0.44 0.26 0.09 0.13 0.30 0.22 0.45 0.12 0.76
0.09 0.42 0.17 0.35 0.07 0.51 0.52 0.61 0.66 0.18
0.22 0.07 0.20 0.31 0.49 0.40 0.37 0.10 0.25 0.55")
```

```r
data<-data.frame(values)

# We use the exact same functions as in Worksheet 4.
logl<-function(theta)
{
  a<-theta[1]
  b<-theta[2]
  if(a<=0|b<=0) return(Inf)
  # We change the gamma function to beta
  l<-sum(log(dbeta(values,shape1=a,shape2=b)))
  return(-l)
}

# Method 1: Optimization using `optim`
res_optim <- optim(par = c(1, 1), fn = logl)
a_optim <- res_optim$par[1]
b_optim <- res_optim$par[2]

library(Rsolnp)
# Method 2: Optimization using `solnp`
res_solnp <- solnp(pars = c(1, 1), fun = logl, LB = c(0, 0))

##
## Iter: 1 fn: -20.2705  Pars:  2.12383 4.43369
## Iter: 2 fn: -20.2705  Pars:  2.12383 4.43369
## solnp--> Completed in 2 iterations

a_solnp <- res_solnp$pars[1]
b_solnp <- res_solnp$pars[2]

# Show results
cat("Optim: a =", a_optim, ", b =", b_optim, "\n")

## Optim: a = 2.123827 , b = 4.433687

cat("Solnp: a =", a_solnp, ", b =", b_solnp, "\n")

## Solnp: a = 2.123826 , b = 4.433686
```

3 Exercise 3

The median of a (numeric) vector of length n is the element that divides the ordered vector in two groups of the same size. Notice that if n is odd you can always find such an element in the vector, but not if n is even. In the case of even n, the usual convention is to define the median as the average of the two

central elements in the ordered vector.

From the definition above, your task is to define your own (original) function in R computing the median. The name of the function should be mymedian, and it should have a single argument, the x vector. It must return the computed median ignoring possible missing values in the vector. You can check your function with the following results:

```
mymedian<-function(x){
  if (length(x)==0 || !is.numeric(x)) return('NA')
  x<-na.omit(x) # If the vector has NA values, we ignore them
  x<-sort(x) # We sort it from smallest to largest

  # If it's an odd length, we take the exact middle number
  if ((length(x)%%2)!=0) {
    return(x[length(x)/2 +1])
  }
  # If it's even, we take the mean of the two middle numbers
  else {
    sum_els<-x[length(x)/2]+x[length(x)/2 + 1]
    return(sum_els/2)
  }
}
mymedian(c(1,1,5,3,6))

## [1] 3

mymedian(c(1,1,5,3,6,1))

## [1] 2

mymedian(cars$speed)

## [1] 15
```

4 Exercise 4

The five-number summary is a set of descriptive statistics that provides concise information about the distribution of a dataset. It consists of the five most important sample percentiles:

- the sample minimum,
- the lower (or first) quartile
- the median,

- the upper (or third) quartile, and
- the sample maximum.

```r
# How fivenum() calculates the quantiles:
# 1. Sorting the data from smallest to largest.
# 2. Finds the median.
# 3. The 1st quartile is the median of the data below the median.
# 4. The 3rd quartile is the median of the data above the median.

# How the function quantile() calculates the quantiles:
# 1. Sort the data from smallest to largest.
# 2. The min and max values are to the 0% and 100% quantiles.
# 3. The p% quantile position is j= 1+(n-1)*p/100.
# 4. If j is an integer, the quantile is the jth ordered value.
# 5. If j is not an integer, the quantile is the mean of the two
#    closest values.

# We calculate the 5% and 95% percentiles the same way quantile()
#   does but with: p*(n+1)/100.

sevennum <- function(x) {
  x <- sort(x)
  n <- length(x)

  med<-mymedian(x)

  q05<-5/100*(length(x)+1)
  # If the position is an integer, we take that exact value.
  if (q05%%1==0) {
    q05<-x[q05]
  }
  # If it isn't, we take the mean of the two closest values.
  else {
    q05<-(x[q05]+x[q05+1])/2
  }

  q95<-95/100*(length(x)+1)
  if (q95%%1==0) {
    q95<-x[q95]
  }
  else {
    q95<-(x[q95]+x[q95+1])/2
  }
```

```
  # We use fivenum() for the other values
  return(c(fivenum(x)[1], q05, fivenum(x)[2:4], q95, fivenum(x)[5]))
}

sevennum(1:100)

## [1]   1.0   5.5  25.5  50.5  75.5  95.5 100.0

sevennum(1:101)

## [1]   1.0   5.5  26.0  51.0  76.0  96.5 101.0

sevennum(cars$speed)

## [1]  4.0  5.5 12.0 15.0 19.0 24.0 25.0
```