Training Exercises

Laura Lázaro Soraluce

April 1, 2025

1 Exercise 1

1.1 Import the data from R and print its structure

```
hatco<-read.table("hatco.txt", header=TRUE)
str(hatco)
```

```
## 'data.frame': 100 obs. of  10 variables:
##  $ client: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ x1    : num  4.1 1.8 3.4 2.7 6 1.9 4.6 1.3 5.5 4 ...
##  $ x2    : num  0.6 3 5.2 1 0.9 3.3 2.4 4.2 1.6 3.5 ...
##  $ x3    : num  6.9 6.3 5.7 7.1 9.6 7.9 9.5 6.2 9.4 6.5 ...
##  $ x4    : num  4.7 6.6 6 5.9 7.8 4.8 6.6 5.1 4.7 6 ...
##  $ x5    : num  2.4 2.5 4.3 1.8 3.4 2.6 3.5 2.8 3.5 3.7 ...
##  $ x6    : num  2.3 4 2.7 2.3 4.6 1.9 4.5 2.2 3 3.2 ...
##  $ x7    : num  5.2 8.4 8.2 7.8 4.5 9.7 7.6 6.9 7.6 8.7 ...
##  $ y     : int  32 43 48 32 58 45 46 44 63 54 ...
##  $ x8    : int  0 1 1 1 0 1 0 1 0 1 ...
```

1.2 Using the function within do the following:

1.2.1 (i) convert the last column of the data frame to a factor with two levels: 0 (small) and 1 (large);

```
hatco<-within(hatco, {x8<-factor(x8, level=c(0,1), labels=c("small", "large"))})
```

1.2.2 and (ii) convert the first column of the data frame to a character variable.

```
hatco <- within(hatco, {client <- as.character(client)})
str(hatco)

## 'data.frame': 100 obs. of  10 variables:
##  $ client: chr  "1" "2" "3" "4" ...
##  $ x1    : num  4.1 1.8 3.4 2.7 6 1.9 4.6 1.3 5.5 4 ...
##  $ x2    : num  0.6 3 5.2 1 0.9 3.3 2.4 4.2 1.6 3.5 ...
##  $ x3    : num  6.9 6.3 5.7 7.1 9.6 7.9 9.5 6.2 9.4 6.5 ...
##  $ x4    : num  4.7 6.6 6 5.9 7.8 4.8 6.6 5.1 4.7 6 ...
##  $ x5    : num  2.4 2.5 4.3 1.8 3.4 2.6 3.5 2.8 3.5 3.7 ...
##  $ x6    : num  2.3 4 2.7 2.3 4.6 1.9 4.5 2.2 3 3.2 ...
##  $ x7    : num  5.2 8.4 8.2 7.8 4.5 9.7 7.6 6.9 7.6 8.7 ...
##  $ y     : int  32 43 48 32 58 45 46 44 63 54 ...
##  $ x8    : Factor w/ 2 levels "small","large": 1 2 2 2 1 2 1 2 1 2 ...
```

1.3 Compute the number of small and large companies in the data.

```
table(hatco$x8)

##
## small large
##    60    40
```

1.4 The variable y is an indicator of the client loyalty to the supplier. Compute a summary of this variable consisting only of the mean and the median. Do it first for all clients, and then separate for small and large clients.

```
# Columns 3 & 4 show the Median & Mean
summary(hatco$y)[3:4]

## Median   Mean
##   46.5   46.1

summary(hatco$y[hatco$x8=="small"])[3:4]

##   Median     Mean
## 49.00000 48.76667

summary(hatco$y[hatco$x8=="large"])[3:4]

## Median   Mean
##   42.5   42.1
```

1.5 Create two new data frames separating small and large clients.

```
hatco_small<-subset(hatco, x8=="small")
hatco_large<-subset(hatco, x8=="large")
```

1.6 Print the data for one client of each type randomly selected (use the function sample).

```
hatco_small[sample(nrow(hatco_small),1),]

##    client  x1  x2  x3  x4  x5  x6  x7  y    x8
## 17     17 3.2 4.1 5.7 5.1 3.6 2.9 6.2 38 small

hatco_large[sample(nrow(hatco_large),1),]

##    client  x1  x2  x3  x4 x5  x6  x7  y    x8
## 70     70 2.3 3.7 8.3 5.2   3 2.3 9.1 49 large
```

2 Exercise 2

2.1 Create a function with name p.arith

It computes the n first terms of an arithmetic progression of the type: $a_{n+1} = a_1 + d \cdot n$. The function should have three arguments: n, a_1 and d and the returned value should be a list with the following components:

- The vector v with the n terms.
- The sum of the elements in v.
- The product of the elements in v.

```
p.arith<-function(n, a1, d){
  a<-numeric(n)
  a[1]<-a1
  i=2
  while (i<=n){
    a[i]=a[i-1]+d*i
    i<-i+1
  }
  sum_els<-sum(a)
  prod_els<-prod(a)
  return (list(a, sum(a), prod(a)))
}
```

2.2 Create a second version of the function above with name p.arith.explict, including the logical argument explicit that is FALSE by default. When this argument is TRUE then the computation of the sum and the product of the n elements will be performed using the following explicit expressions:

$$\sum_{i=1}^{n} a_i = n\frac{a_1 + a_n}{2} \prod_{i=1}^{n} a_i = d^n \frac{\Gamma(a_1/d + n)}{\Gamma(a_1/d)}$$

```
sum_els<-function(a){

}
p.arith.explict<-function(n, a1, d, explicit=FALSE){
  if (!explicit){
    return(p.arith(n,a1,d))
  }
  else {
    a<-numeric(n)
    a[1]<-a1
    i=2
    while (i<=n){
      a[i]=a[i-1]+d*i
      i<-i+1
    }
    sum_els<-n*(a[1]+a[n])/2
    prod_els<-d^n * gamma(a[1]/d +n)/gamma(a1/d)
    return (list(a, sum(a), prod(a)))
  }
}
```

2.3 Use the function system.time to measure the computing time of the function p.arith.explicit and find out whether the explicit expression are quicker or not. Evaluate the function for $n = 2e + 6$.

```
time1<-system.time(p.arith(2e6,1,1))
time2<-system.time(p.arith.explict(2e6,1,1,TRUE))
time3<-system.time(p.arith.explict(2e6,1,1,FALSE))

time1

##    user  system elapsed
##   0.543   0.008   0.550

time2

##    user  system elapsed
##   2.495   0.008   2.503

time3

##    user  system elapsed
##   0.536   0.000   0.536
```