Lab 1 exercise - R basic commands

CaiHuaiyu(Edward) - 2030026003

2023/2/25

Q1. Let x be defined as follow.
a. Add 3 to just the even-index elements.
b. Compute the square root of each element.

```
x <- c(11,10,5,9,14,12,18,2,17,9,1,7,1,15,16,7,16,12,13,3,8,3,11,1,5,7,7,10,6,10,17,13,1,2,17,1,15,8,12
```

```
# Write your answer here
#a)
y<-1:length(x)
y<-x[seq(2,length(x),2)]+3
y
```

```
##  [1] 13 12 15  5 12 10 18 10 15  6  6  4 10 13 13 16  5  4 11  6 18  6 15 20 15
## [26] 20 16 12  5 19 10 15  8 10 12 10 14 15 11  6 17 21 14  4 21 15 13  9 15  6
```

```
#b)
m<-1:length(x)
m<-sqrt(x)
m
```

```
##   [1] 3.316625 3.162278 2.236068 3.000000 3.741657 3.464102 4.242641 1.414214
##   [9] 4.123106 3.000000 1.000000 2.645751 1.000000 3.872983 4.000000 2.645751
##  [17] 4.000000 3.464102 3.605551 1.732051 2.828427 1.732051 3.316625 1.000000
##  [25] 2.236068 2.645751 2.645751 3.162278 2.449490 3.162278 4.123106 3.605551
##  [33] 1.000000 1.414214 4.123106 1.000000 3.872983 2.828427 3.464102 1.732051
##  [41] 2.449490 3.872983 3.162278 1.732051 3.162278 3.464102 4.358899 4.123106
##  [49] 3.872983 3.464102 4.242641 4.123106 1.000000 3.605551 2.236068 3.000000
##  [57] 4.242641 1.414214 4.472136 4.000000 3.872983 2.645751 3.000000 3.464102
##  [65] 3.464102 2.236068 2.236068 2.645751 2.645751 3.000000 4.123106 2.645751
##  [73] 3.872983 3.316625 3.162278 3.464102 4.358899 2.828427 3.605551 1.732051
##  [81] 2.645751 3.741657 3.605551 4.242641 1.000000 3.316625 4.242641 1.000000
##  [89] 3.464102 4.242641 2.236068 3.464102 2.236068 3.162278 1.732051 2.449490
##  [97] 3.316625 3.464102 1.000000 1.732051
```

Q2. Given x = [3 15 9 12 -1 -12 9 6 1], provide the command that will
a) set the values of x that are positive to zero
b) set values that are multiples of 3 to 3
c) multiply the values of x that are even by 5
d) extract the values of x that are greater than 10 into a vector called y
e) Find the index position of elements in x which are larger than 4

```r
# Write your answer here
x<-c(3,15,9,12,-1,-12,9,6,1)
#a)
y<-1:length(x)
y<-x
y[y>0]<-0
y
```

```
## [1]   0   0   0   0  -1 -12   0   0   0
```

```r
#b)
y<-1:length(x)
y<-x
y[y%%3==0]<-3
y
```

```
## [1]  3  3  3  3 -1  3  3  3  1
```

```r
#c)
y<-1:length(x)
y<-x
y[y%%2==0]<-5*y[y%%2==0]
y
```

```
## [1]   3  15   9  60  -1 -60   9  30   1
```

```r
#d)
y<-1:length(x)
y<-x[x>10]
y
```

```
## [1] 15 12
```

```r
#e)
y<-1:length(x)
y<-which(x>4)
y
```

```
## [1] 2 3 4 7 8
```

Q3. Given the matrix A as following

```r
a <- c(2,4,1,6,7,2,3,5,9)
A <- matrix(a, nrow=3, ncol=3, byrow=T)
A
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    1
## [2,]    6    7    2
## [3,]    3    5    9
```

provide the commands needed to
a) assign the last 2 rows of A to a matrix called y
b) compute the sum over the columns of A
c) compute the sum over the rows of A
d) compute the standard deviation of each column of A
e) Standardize the matrix A such that columns of A are centered to have mean 0 and scaled to have standard deviation 1.

```
# Write your answer here
#a)
y = matrix(c(A[2,],A[3,]), nrow = 2, ncol = 3, byrow = T)
y
```

```
##      [,1] [,2] [,3]
## [1,]    6    7    2
## [2,]    3    5    9
```

```
#b)
ncol(A)
```

```
## [1] 3
```

```
sum(A[,1])
```

```
## [1] 11
```

```
sum(A[,2])
```

```
## [1] 16
```

```
sum(A[,3])
```

```
## [1] 12
```

```
#c)
nrow(A)
```

```
## [1] 3
```

```
sum(A[1,])
```

```
## [1] 7
```

```
sum(A[2,])
```

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

```r
sum(A[3,])
```

```
## [1] 17
```

```r
#d)
sd(A[,1])
```

```
## [1] 2.081666
```

```r
sd(A[,2])
```

```
## [1] 1.527525
```

```r
sd(A[,3])
```

```
## [1] 4.358899
```

```r
#e)
scale(A,center=T,scale=T)
```

```
##            [,1]       [,2]       [,3]
## [1,] -0.8006408 -0.8728716 -0.6882472
## [2,]  1.1208971  1.0910895 -0.4588315
## [3,] -0.3202563 -0.2182179  1.1470787
## attr(,"scaled:center")
## [1] 3.666667 5.333333 4.000000
## attr(,"scaled:scale")
## [1] 2.081666 1.527525 4.358899
```

Q4. The data file "country.txt" in Ispace contains numerous population indicators for a sample of 115 countries. Read the data file into R and answer the following questions.
a. How many variables are included in the dataset?
b. What is the percentage of developing country (develop=1) in this data?
c. What is the standard deviation of the variable death rate?
d. What is the range of GDP for the countries in which death rate is greater than 8?
e. What is the mean of GDP for the developing countries which have death rate higher than 10?
f. What is the correlation coefficient between GDP and birthrate? Discuss the relationship between GDP and the birthrate.

```r
# Write your answer here
#setwd("C:\Users\54056\Desktop\UIC\Year 3\Semester2\Introduction to data visualization\R\Lab1")
rt <- read.table("country.txt",head=TRUE);
#rt
ncol(rt)
```

```
## [1] 11
```

```
#b)
sum(rt[[8]])/nrow(rt)
```

```
## [1] 0.7652174
```

```
#c)
sd(rt[['deathrate']])
```

```
## [1] 4.475512
```

```
#d)
result<-rt[which(rt$deathrate>8),]
range(result$gdp)
```

```
## [1]   120 22470
```

```
#e)
result<-rt[which(rt$deathrate>10),]
#result
mean(result$gdp)
```

```
## [1] 2576.283
```

```
#f)
cor.test(rt$gdp,rt$birthrate)
```

```
##
##  Pearson's product-moment correlation
##
## data:  rt$gdp and rt$birthrate
## t = -10.828, df = 113, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.7930835 -0.6102314
## sample estimates:
##       cor
## -0.713604
```

```
#they are negative relative
```

Q5. Fibonacci sequence is defined by F(1)=F(2)=1 and F(n)=F(n–1)+F(n–2). Write an R function called `Fibonacci` which displays the first n Fibonacci numbers and computes their average. For example, if the input n is 20, the output of Fibonacci (20) shall look like:
The first 20 Fibonacci numbers are:
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
The average is 885.5

```r
# write your answer here
Fibonacci <- function(n)
{
  f <- c()
  f[1] <- 1
  f[2] <- 1
  if(n == 1|n == 2)
  {
    f[n] <- f[n]
    return(f)
  }
  else
  {
    for(i in 3:n)
    {
      f[i] <- f[i-1] + f[i-2]
    }
    return(f)
  }
}
Fibonacci(20)
```

```
##  [1]    1    1    2    3    5    8   13   21   34   55   89  144  233  377  610
## [16]  987 1597 2584 4181 6765
```

```r
mean(Fibonacci(20))
```

```
## [1] 885.5
```