Homework 1

Yuhao Mao 3170102264

2020/7/6

Problem 1: Iowa data set

1. Load data with `read.csv()`.

```
iowa.df<-read.csv("data/iowa.csv",header=T,sep=';')
```

2. Learn number of rows and columns.

```
dim(iowa.df)
```

```
## [1] 33 10
```

Therefore, there are 33 rows and 10 columns.

3. Get names of columns.

```
colnames(iowa.df)
```

```
##  [1] "Year"  "Rain0" "Temp1" "Rain1" "Temp2" "Rain2" "Temp3" "Rain3" "Temp4"
## [10] "Yield"
```

4. value in row 5, column 7.

```
iowa.df[5,7]
```

```
## [1] 79.7
```

5. Second row of the data set.

```
iowa.df[2,]
```

```
##   Year Rain0 Temp1 Rain1 Temp2 Rain2 Temp3 Rain3 Temp4 Yield
## 2 1931 14.76  57.5  3.83    75  2.72  77.2   3.3  72.6  32.9
```

Problem 2: Syntax

1. Explain following commands.

```
vector1 <- c("5", "12", "7", "32") ## No print-out.
## This sentence assigns the
## combination/list to the variable vector1.

max(vector1) ## "7".
## Because they are all strings and
```

```
## the order of strings is based on characters at the beginning.
## Here "7" is bigger than "5", "1" and "3".

sort(vector1) ## "12" "32" "5"  "7".
## Same as before, the strings are sorted
## based on characters at the beginning.

sum(vector1) ## Error.
## strings cannot be added.
```

2. Explain following commands.

```
vector2 <- c("5",7,12)
vector2[2] + vector2[3] ## Error.
## vectors are the same type, so they are all strings.
## strings cannot be added together.

dataframe3 <- data.frame(z1="5",z2=7,z3=12)
dataframe3[1,2] + dataframe3[1,3] ## 19.
## 7+12=19.

list4 <- list(z1="6", z2=42, z3="49", z4=126)
list4[[2]]+list4[[4]] ## 168.
## This indexing returns values. 42+126=168.
list4[2]+list4[4] ## Error.
## This indexing returns lists. Lists cannot be added together.
```

3. Producing lists and replicates.

- Lists. From 1 to 10000 in increments of 372.

```
seq(1, 10000, by=372)
```

```
##  [1]    1  373  745 1117 1489 1861 2233 2605 2977 3349 3721 4093 4465 4837 5209
## [16] 5581 5953 6325 6697 7069 7441 7813 8185 8557 8929 9301 9673
```

From 1 to 10000 with exactly 50 numbers.

```
seq(1, 10000, length.out=50)
```

```
##  [1]     1.0000   205.0612   409.1224   613.1837   817.2449  1021.3061
##  [7]  1225.3673  1429.4286  1633.4898  1837.5510  2041.6122  2245.6735
## [13]  2449.7347  2653.7959  2857.8571  3061.9184  3265.9796  3470.0408
## [19]  3674.1020  3878.1633  4082.2245  4286.2857  4490.3469  4694.4082
## [25]  4898.4694  5102.5306  5306.5918  5510.6531  5714.7143  5918.7755
## [31]  6122.8367  6326.8980  6530.9592  6735.0204  6939.0816  7143.1429
## [37]  7347.2041  7551.2653  7755.3265  7959.3878  8163.4490  8367.5102
## [43]  8571.5714  8775.6327  8979.6939  9183.7551  9387.8163  9591.8776
## [49]  9795.9388 10000.0000
```

- Replicates.

```
rep(1:3, times=3)
```

```
## [1] 1 2 3 1 2 3 1 2 3
```

```
rep(1:3, each=3)
```

```
## [1] 1 1 1 2 2 2 3 3 3
```

It is clear that the first one repeats the whole list for three times but the second one repeats each element for three times.

Problem 3: Orings data set

Read in data set.

```
library(faraway)
data(orings)
```

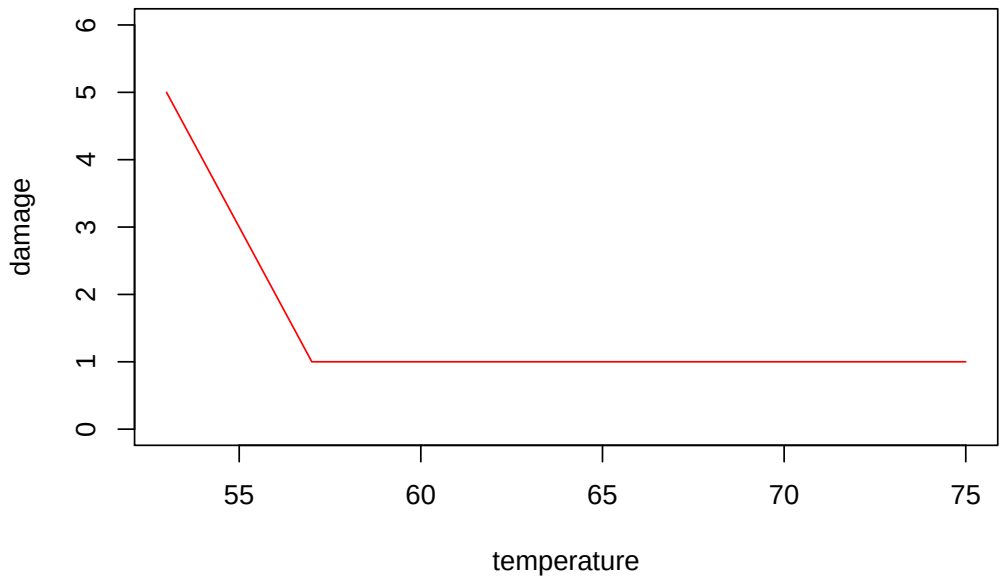
Extract used rows

```
Orings.used <- orings[c(1,2,4,11,13,18),]
Orings.used
```

```
##    temp damage
## 1    53      5
## 2    57      1
## 4    63      1
## 11   70      1
## 13   70      1
## 18   75      1
```

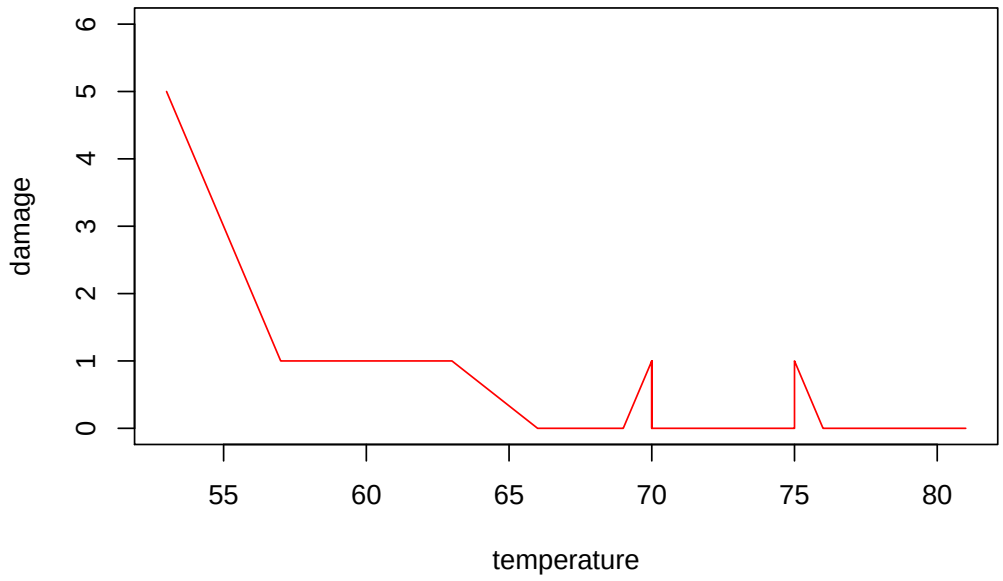
Plot extracted information.

```
plot(damage~temp, data=Orings.used,
       xlab="temperature",ylab="damage",col="red", ylim=c(0,6),type='l')
```



Plot the whole data set.

```
plot(damage~temp, data=orings,
       xlab="temperature",ylab="damage",col="red", ylim=c(0,6),type='l')
```



Problem 4: ais data set

1. Read in data set.

```
library(DAAG,warn.conflicts=F, quietly=T)
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:faraway':
##
##     melanoma
```

```
data("ais")
```

Use `str()` to extract information.

```
str(ais)
```

```
## 'data.frame':    202 obs. of  13 variables:
##  $ rcc   : num  3.96 4.41 4.14 4.11 4.45 4.1 4.31 4.42 4.3 4.51 ...
##  $ wcc   : num  7.5 8.3 5 5.3 6.8 4.4 5.3 5.7 8.9 4.4 ...
##  $ hc    : num  37.5 38.2 36.4 37.3 41.5 37.4 39.6 39.9 41.1 41.6 ...
##  $ hg    : num  12.3 12.7 11.6 12.6 14 12.5 12.8 13.2 13.5 12.7 ...
##  $ ferr  : num  60 68 21 69 29 42 73 44 41 44 ...
##  $ bmi   : num  20.6 20.7 21.9 21.9 19 ...
##  $ ssf   : num  109.1 102.8 104.6 126.4 80.3 ...
```

```
##  $ pcBfat: num  19.8 21.3 19.9 23.7 17.6 ...
##  $ lbm   : num  63.3 58.5 55.4 57.2 53.2 ...
##  $ ht    : num  196 190 178 185 185 ...
##  $ wt    : num  78.9 74.4 69.1 74.9 64.6 63.7 75.2 62.3 66.5 62.9 ...
##  $ sex   : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sport : Factor w/ 10 levels "B_Ball","Field",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Whether it contains missing values.

```
any(is.na(ais))
```

```
## [1] FALSE
```

The answer is NO.

2. Create table to count the number of males and females for each sport.

```
sport <- unique(ais$sport)
table.df <- read.table(text = "",col.names = sport)
for(i in 1:length(sport))
{
  s <-  sport[i]
  table.df[1,i] <-  sum(ais$sex=='f' & ais$sport==s)
  table.df[2,i] <-  sum(ais$sex=='m' & ais$sport==s)
}
row.names(table.df) = c('f', 'm')
table.df
```

```
##   B_Ball Row Netball Swim Field T_400m T_Sprnt Tennis Gym W_Polo
## f     13  22      23    9     7     11       4      7   4      0
## m     12  15       0   13    12     18      11      4   0     17
```

Large imbalance sports:

```
index <-  table.df[1,]/table.df[2,]>2 | table.df[1,]/table.df[2,]<0.5
colnames(table.df)[index]
```

```
## [1] "Netball" "T_Sprnt" "Gym"     "W_Polo"
```

Problem 5: Manitoba data set

Create data set:

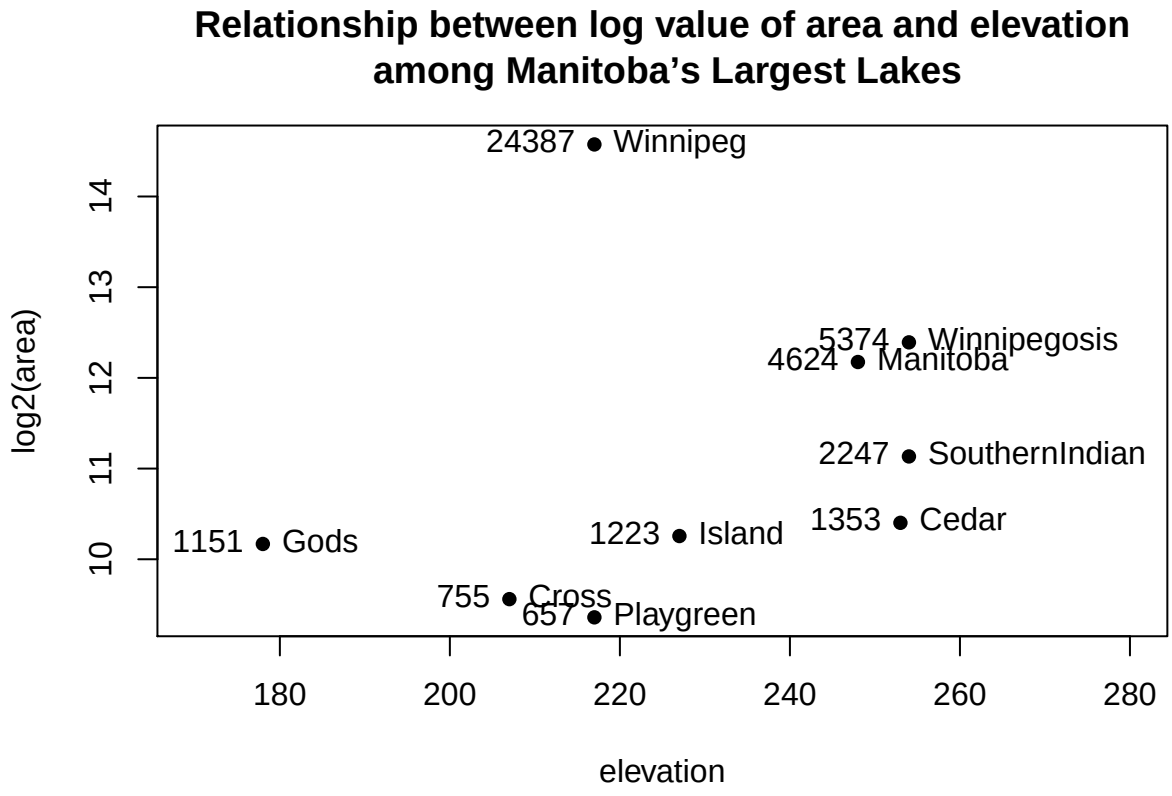
```
Manitoba.lakes <- setNames(
  data.frame(
    t(data.frame(c("Winnipeg", 217, 24387),
                 c("Winnipegosis", 254, 5374),
                 c("Manitoba", 248, 4624),
                 c("SouthernIndian", 254,2247),
                 c("Cedar", 253, 1353),
                 c("Island", 227, 1223),
                 c("Gods", 178, 1151),
                 c("Cross", 207, 755),
                 c("Playgreen", 217, 657)))
    ,row.names = NULL,stringsAsFactors = FALSE
  ),
```

```
  c("district","elevation","area")
)
row.names(Manitoba.lakes) <- Manitoba.lakes$district
Manitoba.lakes$district <- NULL
Manitoba.lakes$elevation <-  as.numeric(Manitoba.lakes$elevation)
Manitoba.lakes$area <-  as.numeric(Manitoba.lakes$area)
Manitoba.lakes
```

```
##                elevation  area
## Winnipeg             217 24387
## Winnipegosis         254  5374
## Manitoba             248  4624
## SouthernIndian       254  2247
## Cedar                253  1353
## Island               227  1223
## Gods                 178  1151
## Cross                207   755
## Playgreen            217   657
```

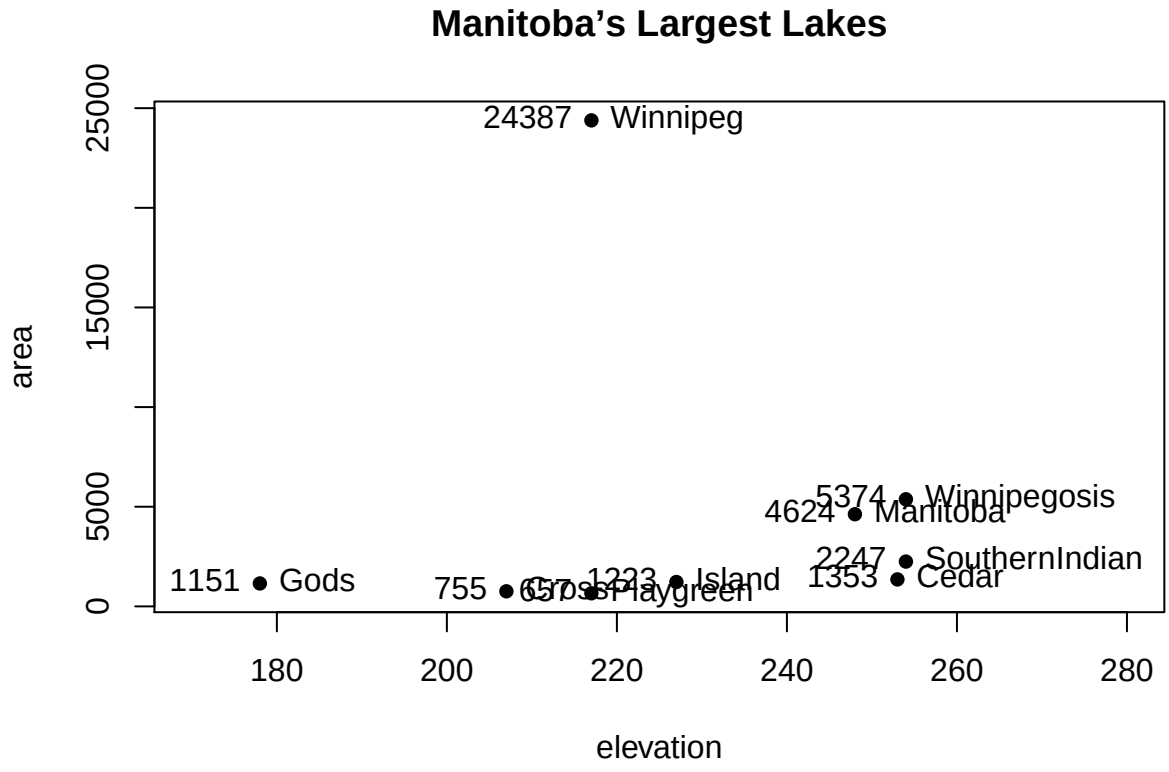
1. Plot log2(area) against elevation.

```
attach(Manitoba.lakes)
plot(log2(area) ~ elevation, pch=16, xlim=c(170,280))
# NB: Doubling the area increases log2(area) by 1.0
text(log2(area) ~ elevation, labels=row.names(Manitoba.lakes), pos=4)
text(log2(area) ~ elevation, labels=area, pos=2)
title("Relationship between log value of area and elevation\n among Manitoba's Largest Lakes")
```



2. Plot area against elevation.

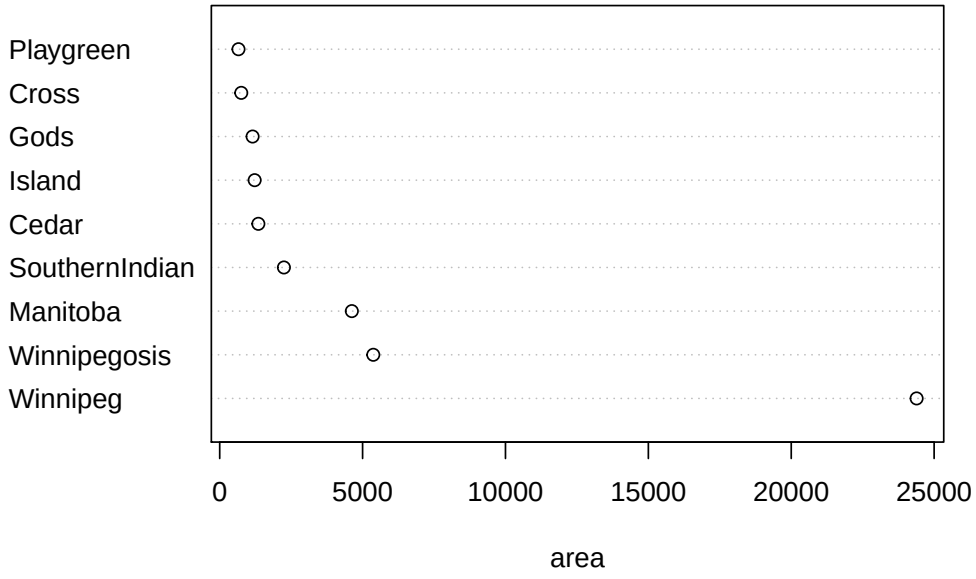
```
plot(area ~ elevation, pch=16, xlim=c(170,280), ylog=T)
text(area ~ elevation, labels=row.names(Manitoba.lakes), pos=4, ylog=T)
text(area ~ elevation, labels=area, pos=2, ylog=T)
title("Manitoba's Largest Lakes")
```



Problem 6: Dot chart for Manitoba's lakes.

1. Areas on a linear scale.

```
dotchart(area,labels=row.names(Manitoba.lakes),xlab="area")
```



2. Areas on a logarithmic scale.

```
dotchart(log2(area),labels=row.names(Manitoba.lakes),xlab="log2(area)")
```