Homework1

Zhang YunMengGe_3170105497

2020/7/7

Question 1

The Iowa data set `iowa.csv` is a toy example that summarises the yield of wheat (bushels per acre) for the state of Iowa between 1930-1962. In addition to yield, year, rainfall and temperature were recorded as the main predictors of yield.

a.First, we need to load the data set into R using the command `read.csv()`. Use the help function to learn what arguments this function takes. Once you have the necessary input, load the data set into R and make it a data frame called `iowa.df`.

```
#Use the help function
?read.csv()
```

```
## starting httpd help server ... done
```

```
#load the data set
iowadata<-read.csv("data/Iowa.csv",header = TRUE, sep = ";")
iowa.df<-as.data.frame(iowadata)
head(iowa.df)
```

```
##   Year Rain0 Temp1 Rain1 Temp2 Rain2 Temp3 Rain3 Temp4 Yield
## 1 1930 17.75  60.2  5.83  69.0  1.49  77.9  2.42  74.4  34.0
## 2 1931 14.76  57.5  3.83  75.0  2.72  77.2  3.30  72.6  32.9
## 3 1932 27.99  62.3  5.17  72.0  3.12  75.8  7.10  72.2  43.0
## 4 1933 16.76  60.5  1.64  77.8  3.45  76.4  3.01  70.5  40.0
## 5 1934 11.36  69.5  3.49  77.2  3.85  79.7  2.84  73.4  23.0
## 6 1935 22.71  55.0  7.00  65.9  3.35  79.4  2.42  73.6  38.4
```

b. How many rows and columns does `iowa.df` have?

```
#get the number of row and col separately
nrow(iowa.df)
```

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

```r
ncol(iowa.df)
```

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

```r
#also wo can get them directly
dim(iowa.df)
```

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

c. What are the names of the columns of `iowa.df`?

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

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

d. What is the value of row 5, column 7 of `iowa.df`?

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

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

e. Display the second row of `iowa.df` in its entirety.

```r
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
```

Question 2: Syntax and class-typing.

a. For each of the following commands, either explain why they should be errors, or explain the non-erroneous result.

```
vector1 <- c("5", "12", "7", "32")
max(vector1)
sort(vector1)
sum(vector1)

#[1] "7"
#[1] "12" "32" "5"  "7"
#Error in sum(vector1) : 'type'(character)
```

The type of the elements is 'character'.

The way to find the max and to sort is according to ASCII code. Of course two characters can not be summed up.

b. For the next series of commands, either explain their results, or why they should produce errors.

```
vector2 <- c("5",7,12)
vector2[2] + vector2[3]
#Error in vector2[2] + vector2[3] :
```

The type of elements in the vector should be consistent, otherwise 7 and 12 will be automatically converted into the same character vector as "5", which cannot be summed.

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

```
## [1] 19
```

Different types of variables can be stored in the data frame. The second and third elements in the first row are numeric vectors, which can be added together.

```
list4 <- list(z1="6", z2=42, z3="49", z4=126)
list4[[2]]+list4[[4]]
list4[2]+list4[4]

#[1] 168
#Error in list4[2] + list4[4] :
```

The elements of the first formula are in numerical form and therefore can be added; the elements of the second formula are in the form of a linked list and cannot be added.

Question 3: Working with functions and operators.

a. The colon operator will create a sequence of integers in order. It is a special case of the function `seq()` which you saw earlier in this assignment. Using the help command `?seq` to learn about the function, design an expression that will give you the sequence of numbers from 1 to 10000 in increments of 372. Design another that will give you a sequence between 1 and 10000 that is exactly 50 numbers in length.

```
?seq()
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
```

```
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
```

b. The function `rep()` repeats a vector some number of times. Explain the difference between ‘rep(1:3, times=3) and rep(1:3, each=3).

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

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

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

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

The first three-digit number "1,2,3" is generated and repeated three times as a whole.

The second produces three digits "1,2,3", where each digit is repeated three times.

MB.Ch1.2.

The orings data frame gives data on the damage that had occurred in US space shuttle launches prior to the disastrous Challenger launch of 28 January 1986. The observations in rows 1, 2, 4, 11, 13, and 18 were included in the pre-launch charts used in deciding whether to proceed with the launch, while remaining rows were omitted. Create a new data frame by extracting these rows from orings, and plot total incidents against temperature for this new data frame. Obtain a similar plot for the full data set.

```r
#install.packages("DAAG")
library(DAAG)
```

```
## Warning: package 'DAAG' was built under R version 4.0.2
```

```
## Loading required package: lattice
```

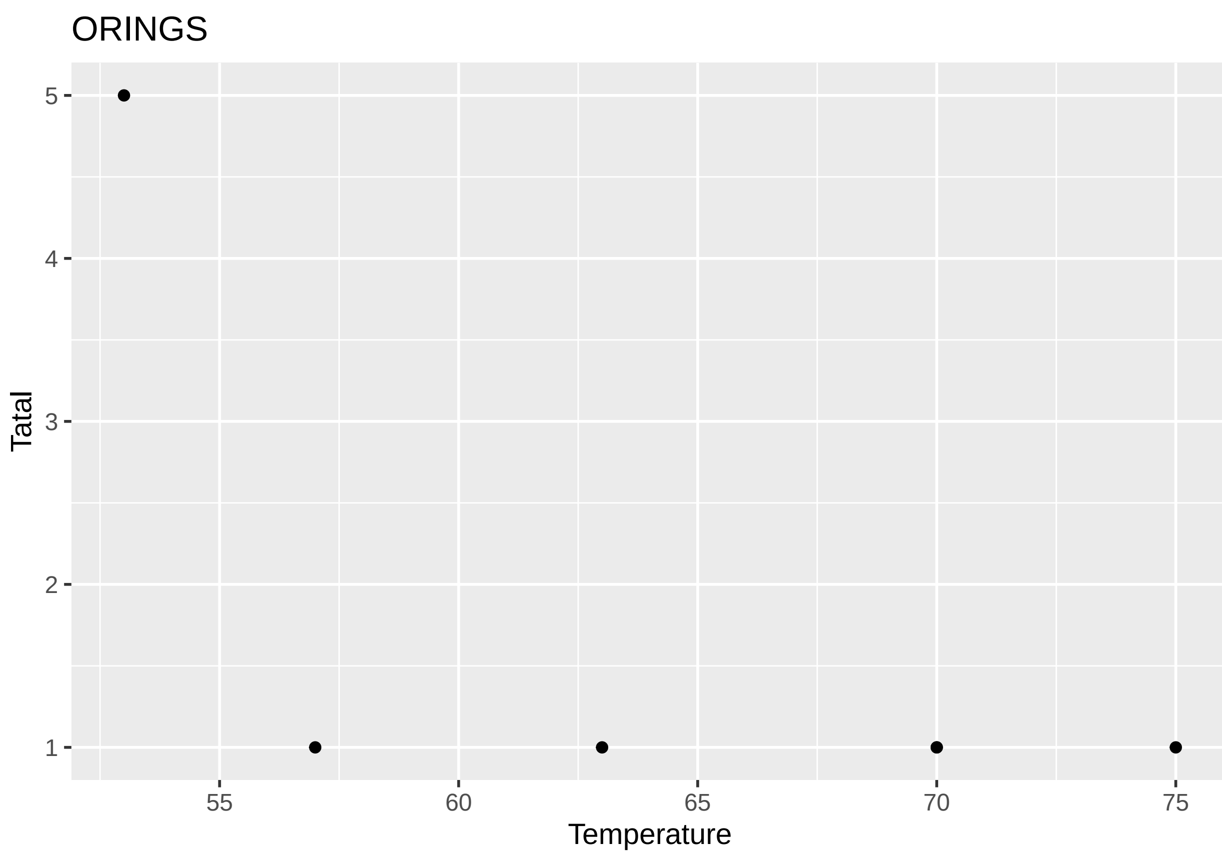
```r
oringsdata<-data.frame(orings)
head(oringsdata)
```

```
##   Temperature Erosion Blowby Total
## 1          53       3      2     5
## 2          57       1      0     1
## 3          58       1      0     1
## 4          63       1      0     1
## 5          66       0      0     0
## 6          67       0      0     0
```

```r
neworings<-oringsdata[c(1,2,4,11,13,18),]
head(neworings)
```

```
##    Temperature Erosion Blowby Total
## 1           53       3      2     5
## 2           57       1      0     1
## 4           63       1      0     1
## 11          70       1      0     1
## 13          70       1      0     1
## 18          75       0      2     1
```

```r
ggplot(data = neworings) +
geom_point(aes(x = Temperature, y = Total)) +
labs(x = "Temperature",
y = "Tatal",
title = "ORINGS")
```



MB.Ch1.4. For the data frame ais (DAAG package)

(a) Use the function str() to get information on each of the columns. Determine whether any of the columns hold missing values.

```
library(DAAG)

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 ...
```

```
#is.na(ais)
which(is.na(ais))
```

```
## integer(0)
```

(b) Make a table that shows the numbers of males and females for each different sport. In which sports is there a large imbalance (e.g., by a factor of more than 2:1) in the numbers of the two sexes?

```
ais<-data.frame(ais)
selectais=table(ais$sex,ais$sport)
selectais
```

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

```
ais2<-t(selectais)
proportion_sport<-ais2[,1]/ais2[,2]
proportion_sport
```

```
##    B_Ball     Field       Gym   Netball       Row      Swim     T_400m   T_Sprnt
## 1.0833333 0.5833333       Inf       Inf 1.4666667 0.6923077 0.6111111 0.3636364
##    Tennis    W_Polo
## 1.7500000 0.0000000
```

```r
ifimbanlance<-((ais2[,1]/ais2[,2]>2)|(ais2[,1]/ais2[,2]<0.5))
ifimbanlance
```

```
##   B_Ball    Field      Gym Netball      Row     Swim   T_400m T_Sprnt   Tennis  W_Polo
##    FALSE    FALSE     TRUE    TRUE    FALSE    FALSE    FALSE    TRUE    FALSE    TRUE
```

From the table `proportion_sport` above, we can find out the porportion of female to male.

The table `ifimbanlance` tells that if the proportion is imbalance. If yes, it turns out to be `True`, if no it turns out to be `False`.

MB.Ch1.6.

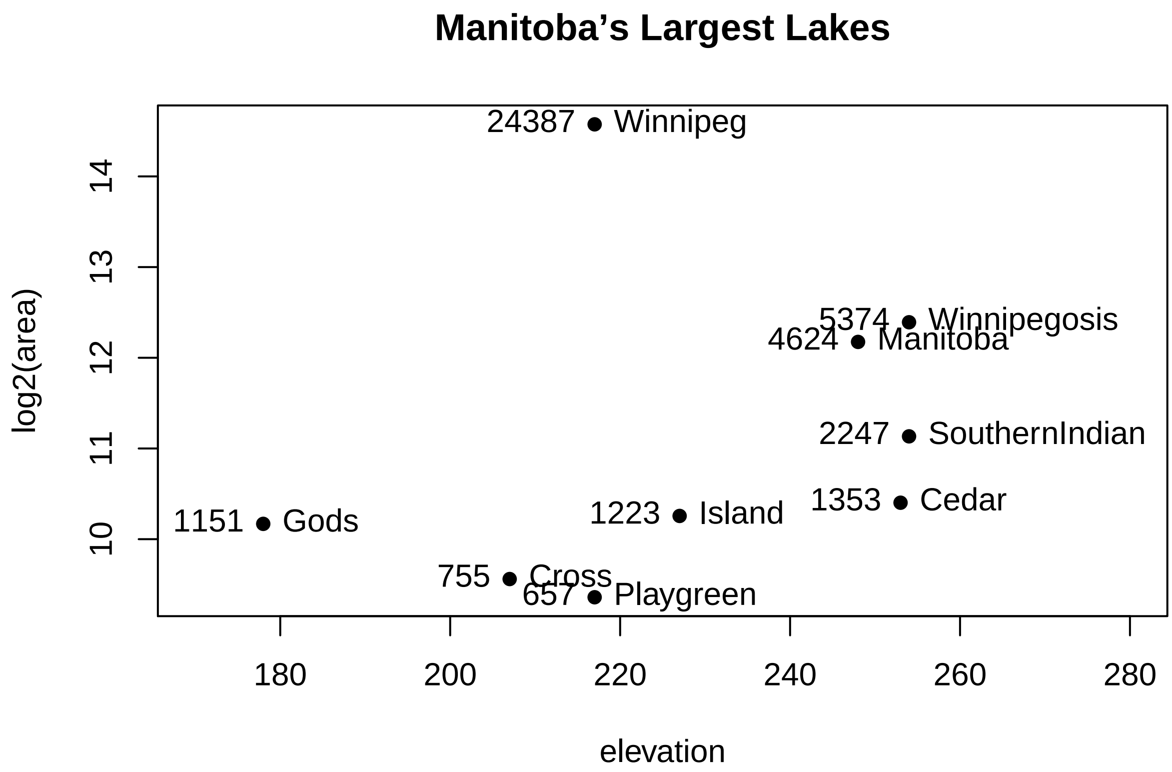
Create a data frame called Manitoba.lakes that contains the lake's elevation (in meters above sea level) and area (in square kilometers) as listed below. Assign the names of the lakes using the row.names() function. elevation area Winnipeg 217 24387 Winnipegosis 254 5374 Manitoba 248 4624 SouthernIndian 254 2247 Cedar 253 1353 Island 227 1223 Gods 178 1151 Cross 207 755 Playgreen 217 657

(a) Use the following code to plot log2(area) versus elevation, adding labeling information (there is an extreme value of area that makes a logarithmic scale pretty much essential):

```r
Manitoba.lakes<-data.frame(elevation=c(217,254,248,254,253,227,178,207,217),area=c(24387,5374,4624,2247
row.names(Manitoba.lakes)<-c("Winnipeg","Winnipegosis","Manitoba","SouthernIndian","Cedar","Island","Go
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
```

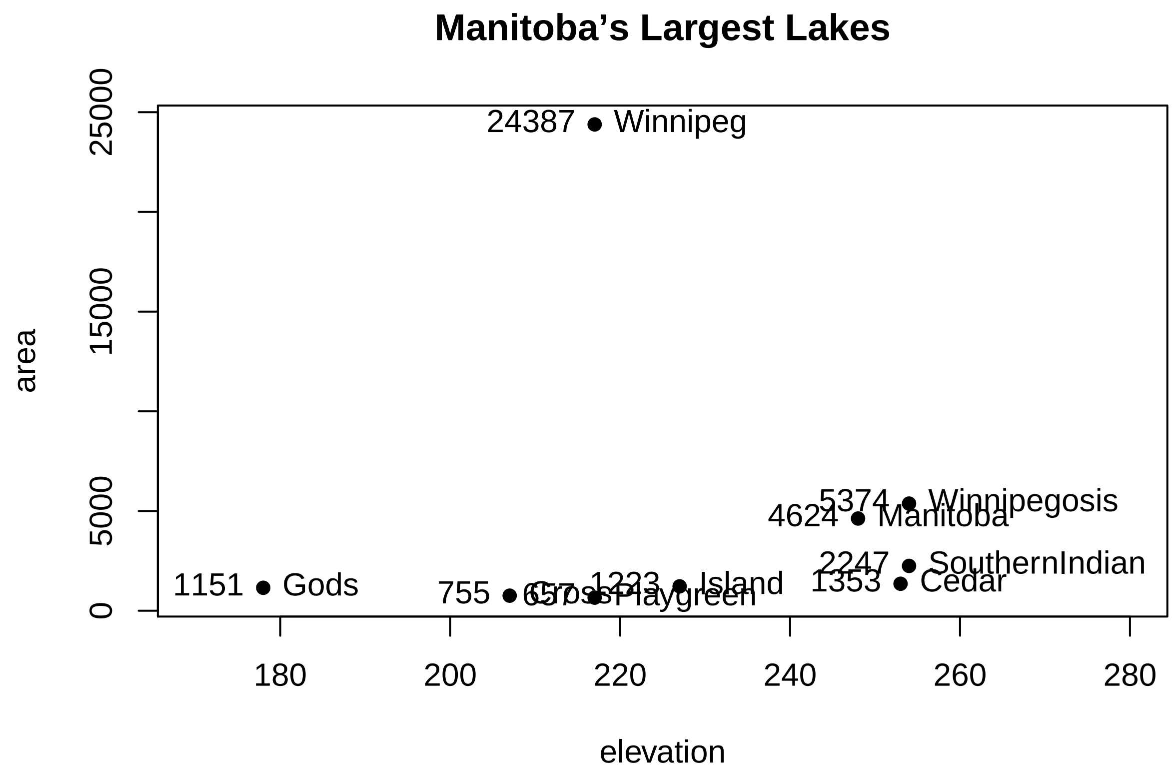
```r
attach(Manitoba.lakes)
plot(log2(area) ~ elevation, pch=16, xlim=c(170,280))
# NB: Doubling the area increases log2(area) by 1.0
text(log2(area) ~ elevation, labels=row.names(Manitoba.lakes), pos=4)
text(log2(area) ~ elevation, labels=area, pos=2)
title("Manitoba's Largest Lakes")
```



captions that explain the labeling on the points and on the y-axis. It will be necessary to explain how distances on the scale relate to changes in area.

(b) Repeat the plot and associated labeling, now plotting area versus elevation, but specifying log="y" in order to obtain a logarithmic y-scale.

```
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")
```

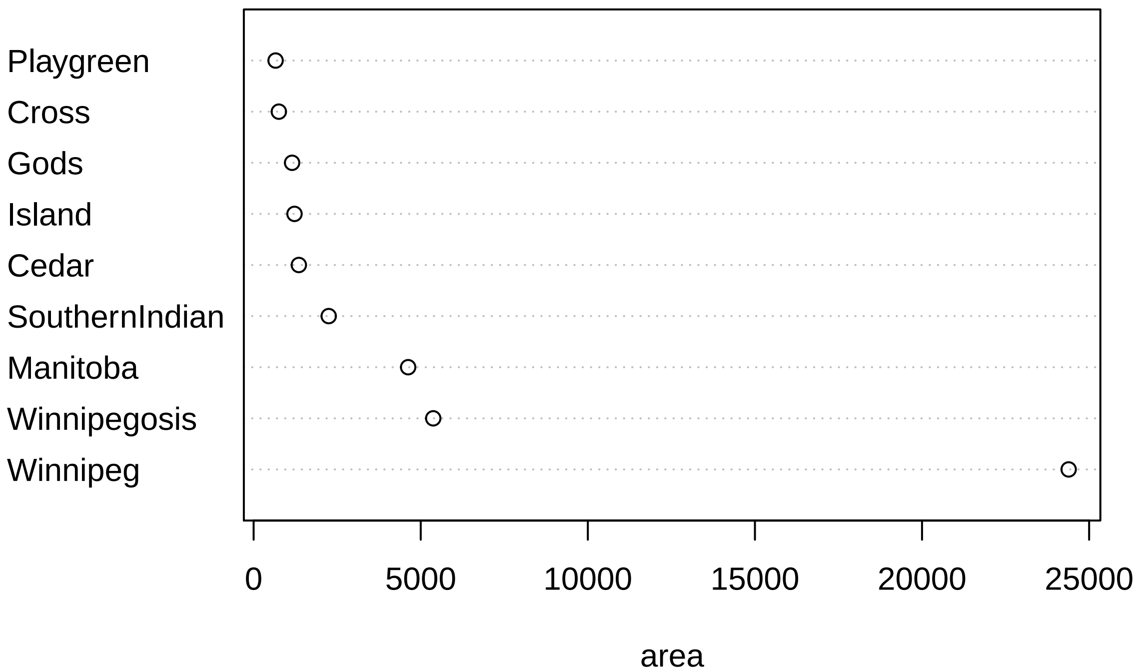


#MB.Ch1.7.

Look up the help page for the R function dotchart(). Use this function to display the areas of the Manitoba lakes (a) on a linear scale, and (b) on a logarithmic scale. Add, in each case, suitable labeling information.

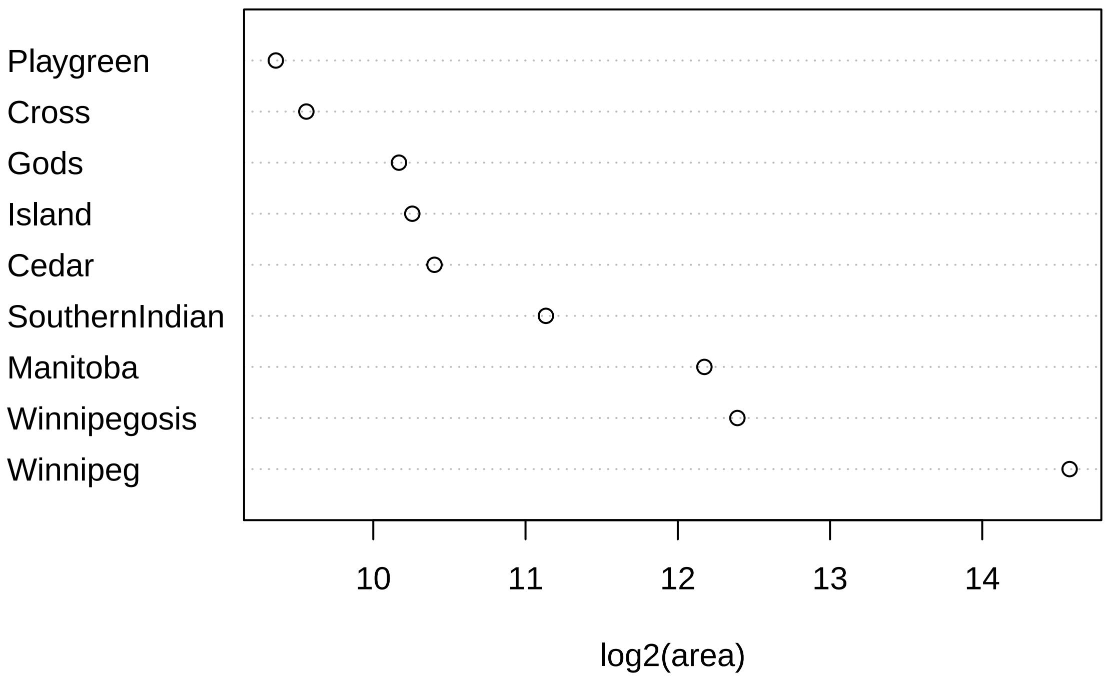
The areas of the Manitoba lakes (a) on a linear scale

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



The areas of the Manitoba lakes (a) on a logarithmic scale

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



MB.Ch1.8. Using the sum() function, obtain a lower bound for the area of Manitoba covered by water.

```
sum(area)
```

```
## [1] 41771
```