Homework 1

Zhang Zile,3230104237

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`.
   b. How many rows and columns does `iowa.df` have?
   c. What are the names of the columns of `iowa.df`?
   d. What is the value of row 5, column 7 of `iowa.df`?
   e. Display the second row of `iowa.df` in its entirety.

```r
# a
iowa.df<-read.csv("data/Iowa.csv", sep = ';', header=T)
# b
nrow(iowa.df)  #33
```

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

```r
ncol(iowa.df)  #10
```

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

```r
# c
colnames(iowa.df) #"Year"  "Rain0" "Temp1" "Rain1" "Temp2" "Rain2" "Temp3" "Rain3" "Temp4" "Yield"
```

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

```r
# d
iowa.df[5,7] #79.7
```

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

```r
# e
iowa.df[,2]
```

```
##  [1] 17.75 14.76 27.99 16.76 11.36 22.71 17.91 23.31 18.53 18.56 12.45 16.05
## [13] 27.10 19.05 20.79 21.88 20.02 23.17 19.15 18.28 18.45 22.00 19.05 15.67
## [25] 15.92 16.75 12.34 15.82 15.24 21.72 25.08 17.79 26.61
```

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")
```

- Correct,it means vector1 is assigned by a vector which contains 4 characters. The result is:

```
> vector1
[1] "5"  "12" "7"  "32"
```

```
max(vector1)
```

- Correct,it returns the maximum element in the vector. The result is:

```
> max(vector1)
[1] "7"
```

```
sort(vector1)
```

Correct,it sorts the vector in order.The result is:

```
> sort(vector1)
[1] "12" "32" "5"  "7"
```

```
sum(vector1)
```

- Error,because the type of elements are character and the arguments of sum should be numeric or complex or logical vectors.

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]

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

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

- Error.A vector can contain elements of only one type.When vector2 is assigned in this example,the entire vector gets converted to character.As a result,you can not do "vector2[2] + vector2[3]" because their type are characters.
- Correct.It creates a dataframe, the elements are of data types character, numeric, numeric separately and numeric variable can be added.The result is:

```
> dataframe3[1,2] + dataframe3[1,3]
[1] 19
```

- "list4[[2]]+list4[[4]]" is correct and the result is 168 because numeric variable can be added.But the type of "list4[2]" and "list4[4]" is still list so they can not be added.

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.
    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
# a
  seq(1,10000,372)
```

```
##  [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
```

```r
  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
```

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

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

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

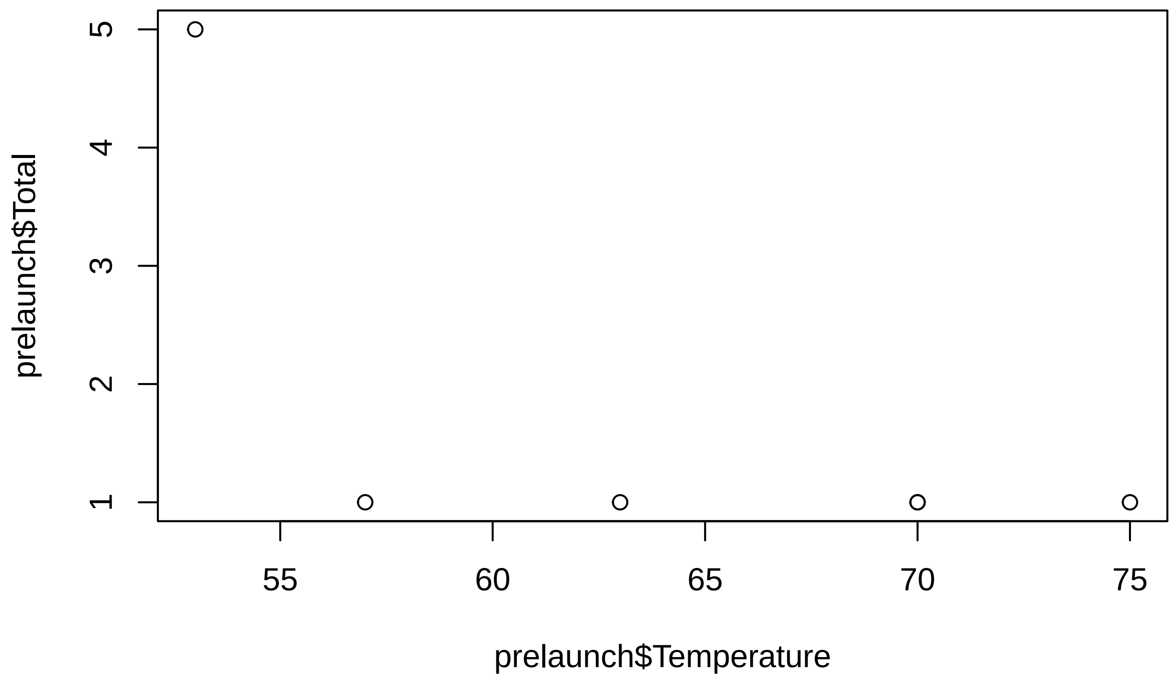
```
## [1] 1 1 1 2 2 2 3 3 3
```

- When comes to "rep(1:3,times=3)",the times argument specifies how many times the entire vector should be repeated.
- However,the each argument specifies how many times each individual element of the vector should be repeated.

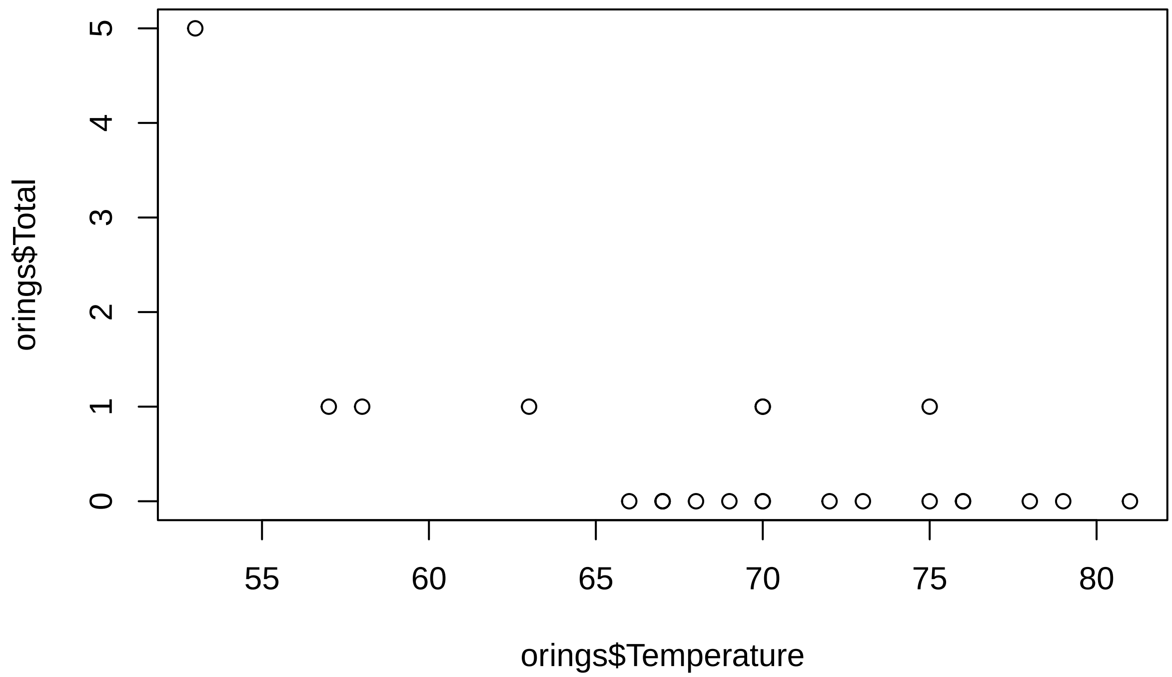
MB.Ch1.2. The orings data frame gives data on the damage that had occurred in US space shuttle launches prior to the disastrous Challenger launch of 28 January 1986. The observations in rows 1, 2, 4, 11, 13, and 18 were included in the pre-launch charts used in deciding whether to proceed with the launch, 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
data(orings)
prelaunch <- orings[c(1,2,4,11,13,18),]
plot(prelaunch$Temperature, prelaunch$Total) # new dataframe
```




```r
plot(orings$Temperature, orings$Total) #full data set
```



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.

```r
data(ais)
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 ...
```

```r
complete.cases(t(ais)) #columns
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

(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 %>% group_by(sport) %>%
  summarise(sex_f = sum(sex == "f"),sex_m = sum(sex == "m")) %>%
  mutate(ratio= sex_f / sex_m) %>%
  mutate(isbalance=(ratio > 0.5) & (ratio < 2))
```

```
## # A tibble: 10 x 5
##    sport   sex_f sex_m   ratio isbalance
##    <fct>   <int> <int>   <dbl> <lgl>
##  1 B_Ball     13    12   1.08  TRUE
##  2 Field       7    12   0.583 TRUE
##  3 Gym         4     0 Inf     FALSE
##  4 Netball    23     0 Inf     FALSE
##  5 Row        22    15   1.47  TRUE
##  6 Swim        9    13   0.692 TRUE
##  7 T_400m     11    18   0.611 TRUE
##  8 T_Sprnt     4    11   0.364 FALSE
##  9 Tennis      7     4   1.75  TRUE
## 10 W_Polo      0    17   0     FALSE
```

- Gym,Netball,T_Sprnt and W_Polo are not balance.

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

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

(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):

```
attach(Manitoba.lakes)
plot(log2(area) ~ elevation, pch=16, xlim=c(170,280))
```

```r
# 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")
```

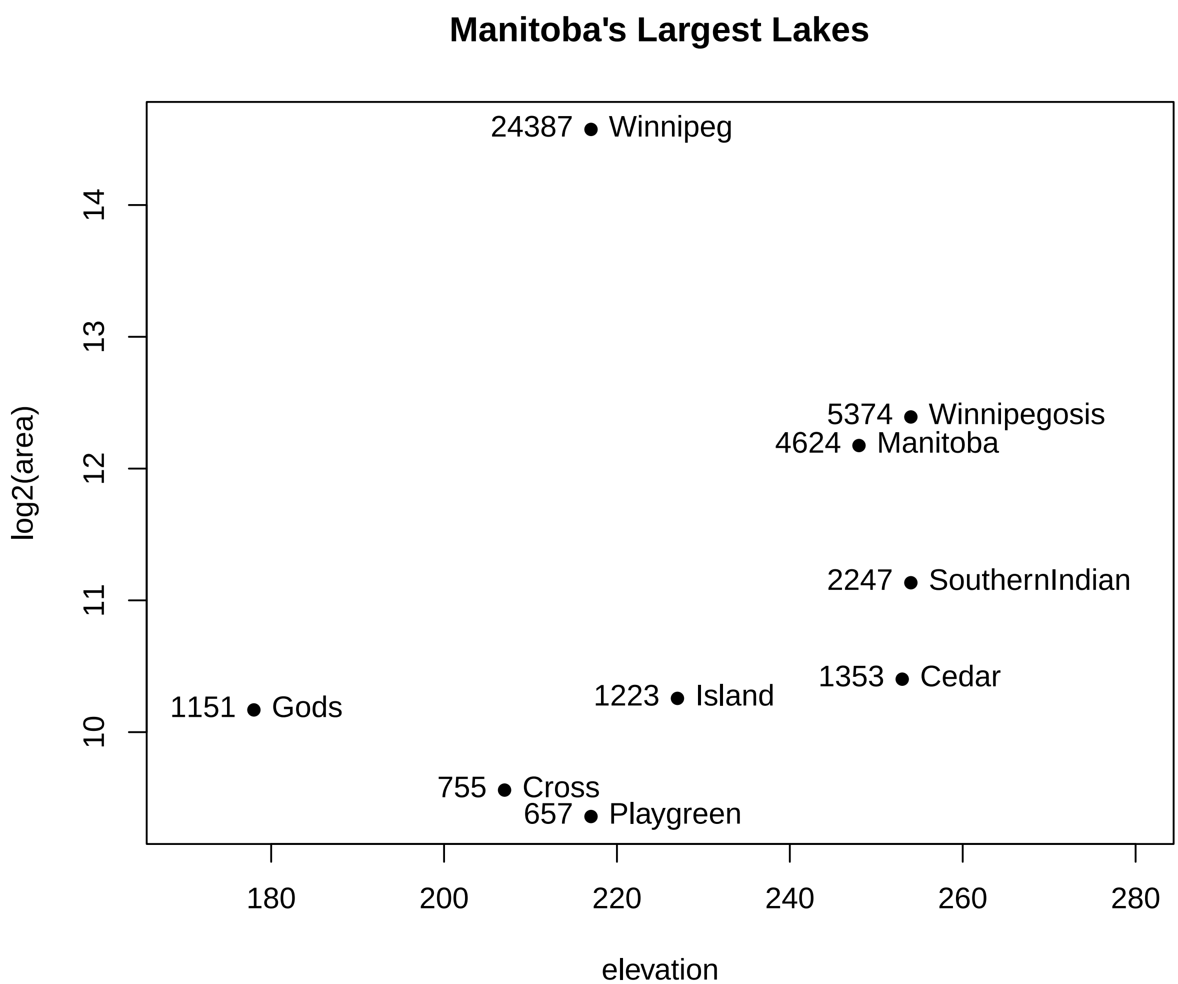


Figure 1: Use log2(area) to respond to skewness towards large values.The y-axis shows the base-2 logarithm of lake area, so that each unit increase corresponds to a doubling in actual surface area. Labels to the left of each point indicate the lake's surface area in square kilometers; labels to the right show the lake's name.

Devise 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 `ylog=TRUE` in order to obtain a logarithmic y-scale.

```r
plot(area ~ elevation, pch=16, xlim=c(170,280), ylog=T)
text(area ~ elevation, labels=row.names(Manitoba.lakes), pos=4, ylog=T)
```

```r
text(area ~ elevation, labels=area, pos=2, ylog=T)
title("Manitoba's Largest Lakes")
```

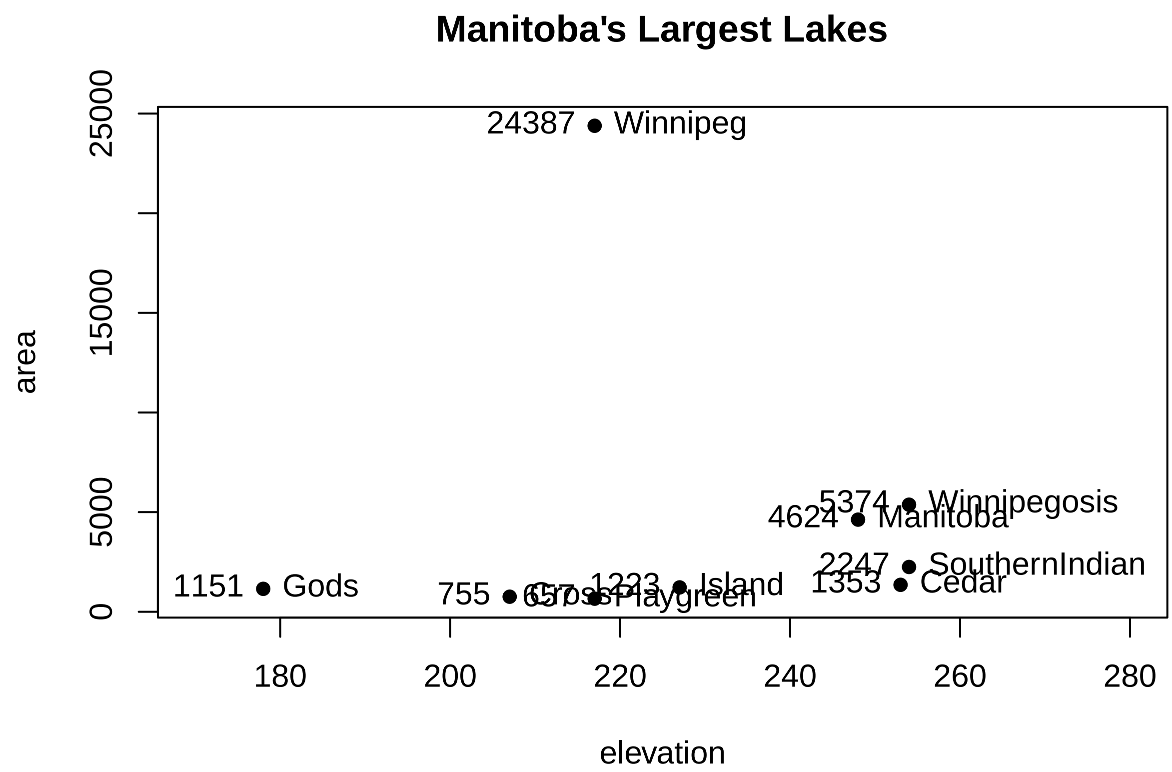
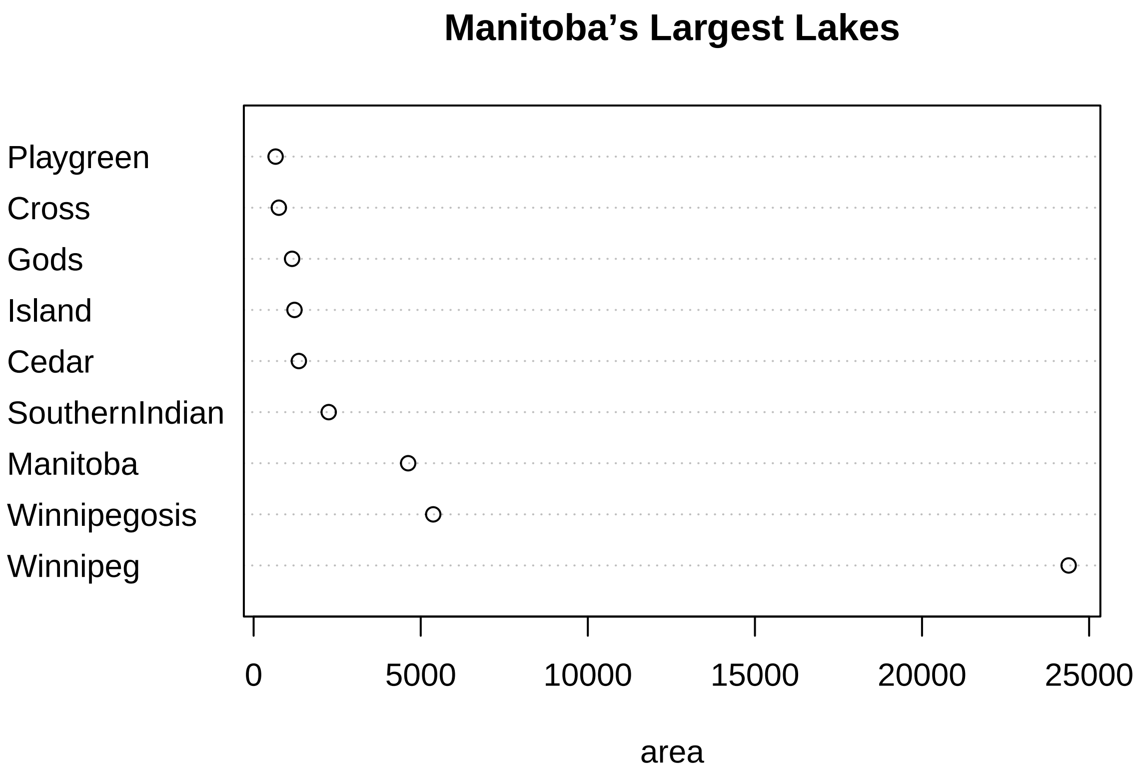


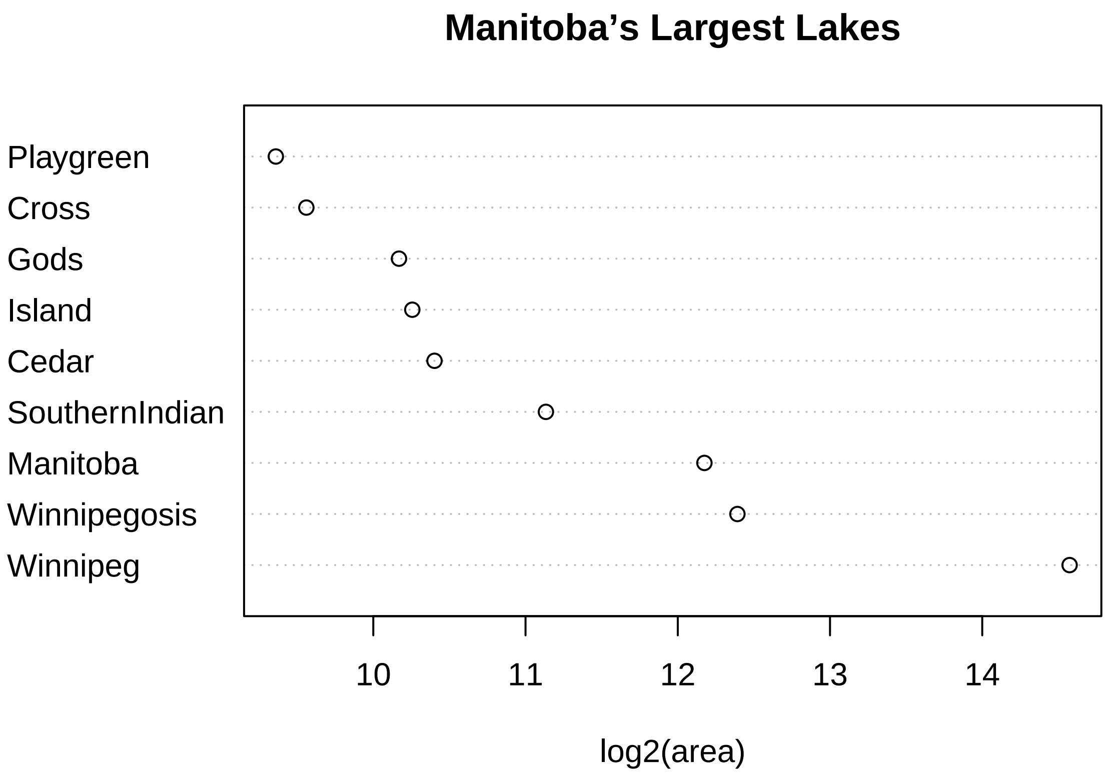
Figure 2: The y-axis now displays real area values but uses a logarithmic scale to accommodate the large range of sizes. As before, labels to the left indicate surface area, and labels to the right show lake names. This plot is more intuitive than the log-transformed axis but retains the advantage of visualizing both small and large lakes effectively.

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.

```r
dotchart(area, labels = rownames(Manitoba.lakes),
         main = "Manitoba's Largest Lakes",
         xlab = "area")
```




```
dotchart(log2(area), labels = rownames(Manitoba.lakes),
         main = "Manitoba's Largest Lakes",
         xlab = "log2(area)")
```



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

```
sum(area)
```

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