Homework 1

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
iowa.df<-read.csv("data/Iowa.csv", sep = ',', header=T)
# b.
cat("b. Number of rows:", nrow(iowa.df), "\n")
```

```
## b. Number of rows: 33
```

```r
cat("   Number of columns:", ncol(iowa.df), "\n\n")
```

```
##    Number of columns: 10
```

```r
# c.
cat("c. Column names:", paste(names(iowa.df), collapse = ", "), "\n\n")
```

```
## c. Column names: Year, Rain0, Temp1, Rain1, Temp2, Rain2, Temp3, Rain3, Temp4, Yield
```

```r
# d.
cat("d. Value at row 5, column 7:", iowa.df[5,7], "\n\n")
```

```
## d. Value at row 5, column 7: 79.7
```

```r
# e.
cat("e. Second row:\n")
```

```
## e. Second row:
```

```r
print(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
```

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")
[1] "5"  "12" "7"  "32"
max(vector1)
[1] "7"
sort(vector1)
[1] "12" "32" "5"  "7"
sum(vector1)
Error: invalid 'type' (character) of argument
```

```
sum() requires numeric input
```

```
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: cannot add strings
When creating the vector vector2, since there is a string "5", R will coerce the entire vector to the cl

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

list4 <- list(z1="6", z2=42, z3="49", z4=126)
list4[[2]]+list4[[4]]
42 + 126 = 168
list4[2]+list4[4]
Error: cannot add sublists
list4[2] and list4[4] return sublists, not individual elements
```

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.

```r
cat("Results:\n")
```

```
## Results:
```

```r
seq_by_372 <- seq(1, 10000, by = 372)
cat("Sequence by 372 (first 5 values):", head(seq_by_372, 5), "...\n")
```

```
## Sequence by 372 (first 5 values): 1 373 745 1117 1489 ...
```

```r
seq_length_50 <- seq(1, 10000, length.out = 50)
cat("Sequence with 50 elements (first/last):",
    head(seq_length_50, 2), "...", tail(seq_length_50, 2), "\n\n")
```

```
## Sequence with 50 elements (first/last): 1 205.0612 ... 9795.939 10000
```

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

```r
cat("Results:\n")
```

```
## Results:
```

```r
cat("rep(1:3, times=3):", rep(1:3, times = 3), "\n")
```

```
## rep(1:3, times=3): 1 2 3 1 2 3 1 2 3
```

```r
cat("rep(1:3, each=3):", rep(1:3, each = 3), "\n")
```

```
## rep(1:3, each=3): 1 1 1 2 2 2 3 3 3
```

```r
cat("Difference: 'times' repeats whole vector, 'each' repeats each element\n\n")
```

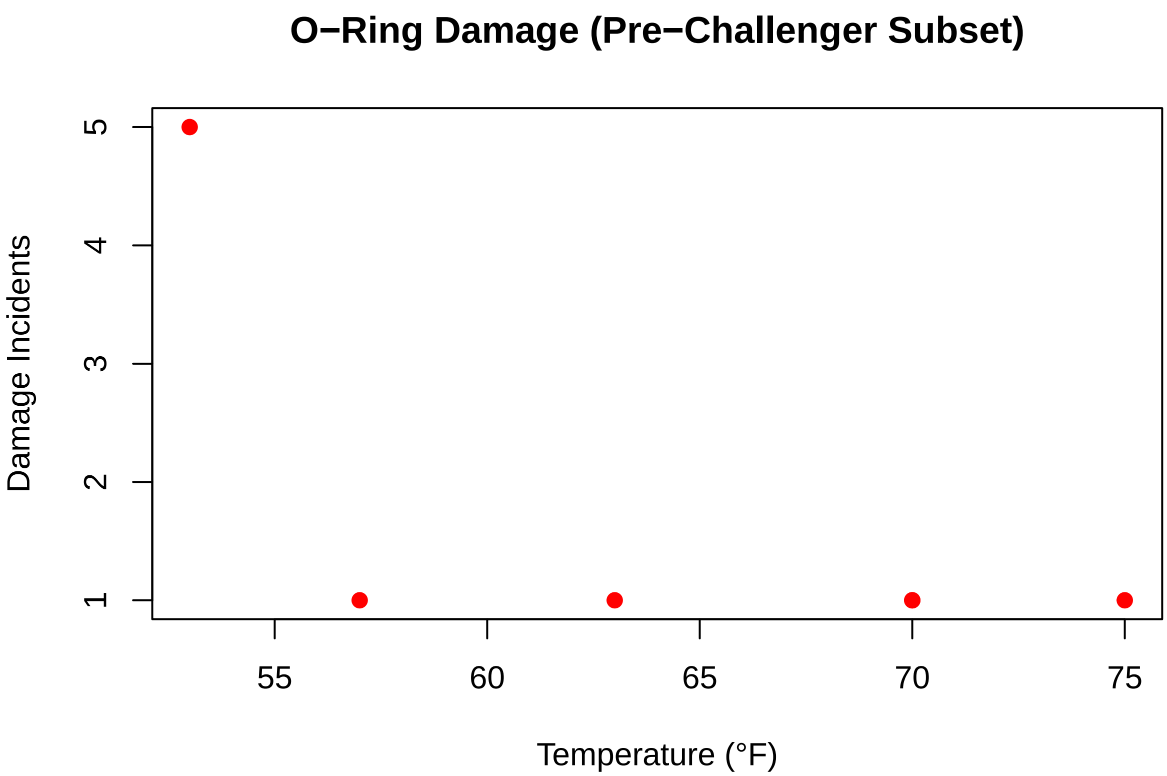
```
## Difference: 'times' repeats whole vector, 'each' repeats each element
```

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.

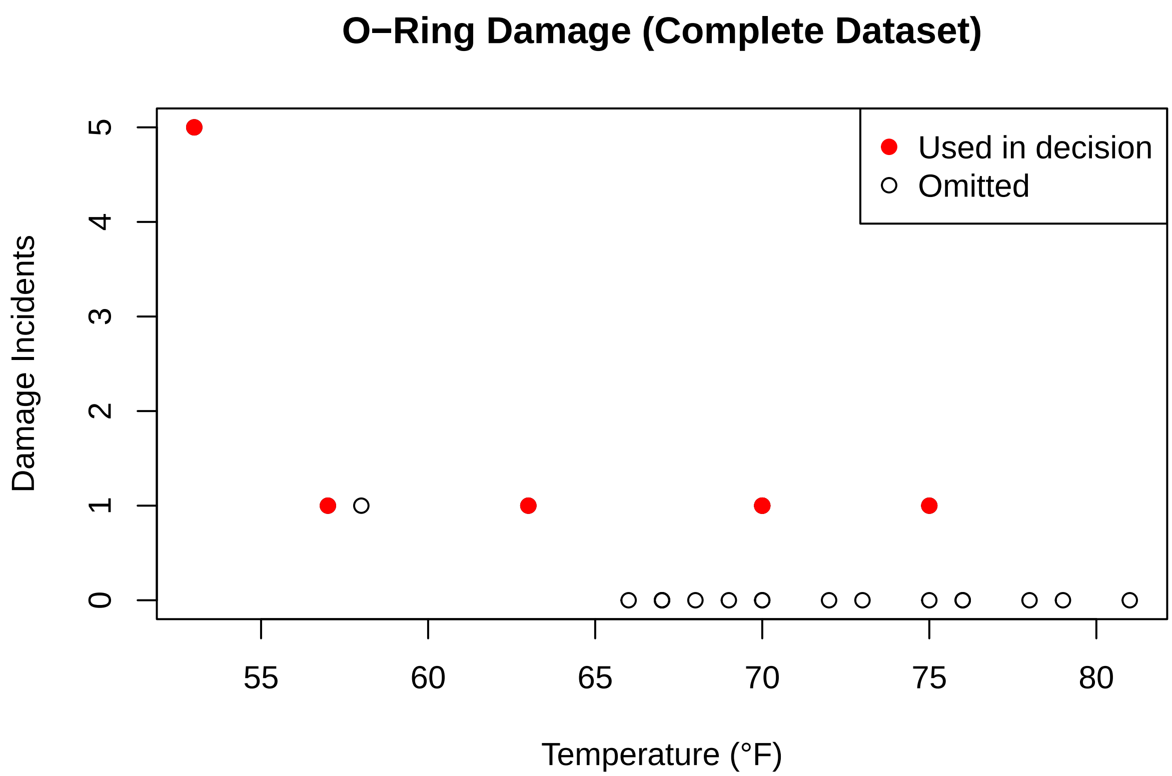
Create a new data frame by extracting these rows from orings, and plot total incidents against temperature for this new data frame. Obtain a similar plot for the full data set.

```r
data(orings)
selected_rows <- c(1, 2, 4, 11, 13, 18)
orings_sub <- orings[selected_rows, ]

plot(Total ~ Temperature, data = orings_sub,
     main = "O-Ring Damage (Pre-Challenger Subset)",
     xlab = "Temperature (°F)", ylab = "Damage Incidents",
     pch = 19, col = "red")
```




```r
plot(Total ~ Temperature, data = orings,
     main = "O-Ring Damage (Complete Dataset)",
     xlab = "Temperature (°F)", ylab = "Damage Incidents")
points(orings_sub$Temperature, orings_sub$Total,
       pch = 19, col = "red")
legend("topright", legend = c("Used in decision", "Omitted"),
       pch = c(19, 1), col = c("red", "black"))
```



MB.Ch1.4. For the data frame **ais** (DAAG package) data(ais) (a) Use the function `str()` to get information on each of the columns. Determine whether any of the columns hold missing values.

```
cat("(a) Data structure:\n")
```

```
## (a) Data structure:
```

```
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 ...
```

```
cat("\nMissing values per column:\n")
```

```
##
## Missing values per column:
```

```
print(colSums(is.na(ais)))
```

```
##    rcc    wcc     hc     hg   ferr    bmi    ssf pcBfat    lbm     ht     wt
##      0      0      0      0      0      0      0      0      0      0      0
```

```
##    sex  sport
##      0      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?

```
gender_table <- as.data.frame.matrix(table(ais$sport, ais$sex))
gender_table$Ratio <- round(gender_table$m / gender_table$f, 2)
imbalanced <- gender_table[gender_table$Ratio > 2 | gender_table$Ratio < 0.5, ]
cat("\n(b) Sports with gender imbalance (>2:1 or <1:2 ratio):\n")
```

```
##
## (b) Sports with gender imbalance (>2:1 or <1:2 ratio):
```

```
print(imbalanced)
```

```
##          f  m Ratio
## Gym      4  0  0.00
## Netball 23  0  0.00
## T_Sprnt  4 11  2.75
## W_Polo   0 17   Inf
```

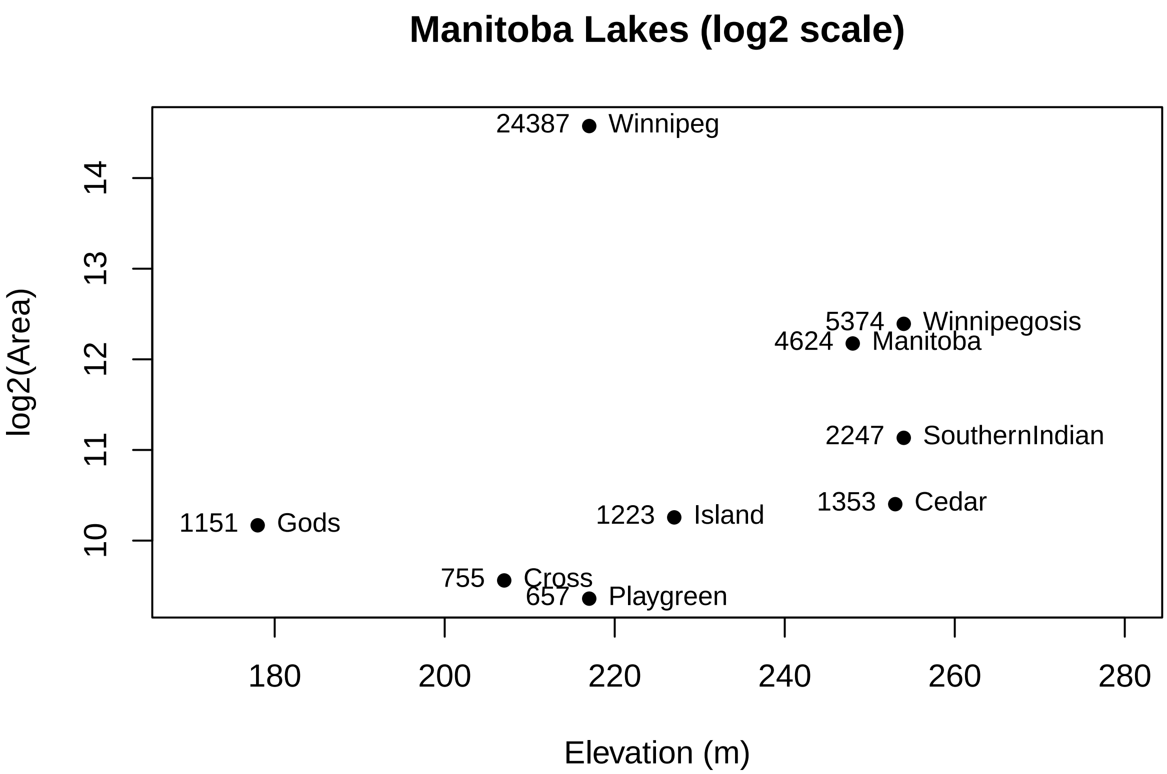
MB.Ch1.6.Create a data frame called Manitoba.lakes that contains the lake's elevation (in meters above sea level) and area (in square kilometers) as listed below. Assign the names of the lakes using the `row.names()` function.

	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):

```
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)
)
rownames(Manitoba.lakes) <- c("Winnipeg", "Winnipegosis", "Manitoba",
                              "SouthernIndian", "Cedar", "Island",
                              "Gods", "Cross", "Playgreen")
#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")
plot(log2(area) ~ elevation, data = Manitoba.lakes, pch = 16,
     xlim = c(170, 280), xlab = "Elevation (m)", ylab = "log2(Area)")
text(Manitoba.lakes$elevation, log2(Manitoba.lakes$area),
     labels = rownames(Manitoba.lakes), pos = 4, cex = 0.8)
```

```r
text(Manitoba.lakes$elevation, log2(Manitoba.lakes$area),
     labels = Manitoba.lakes$area, pos = 2, cex = 0.8)
title("Manitoba Lakes (log2 scale)")
```




```r
cat("Y-axis : Each unit increase in log2(area) represents a doubling of area\n\n")
```

```
## Y-axis : Each unit increase in log2(area) represents a doubling of area
```

```r
cat("Labels : Lake names (right) link points to lakes; raw area values (left) show actual sizes, clarif
```

```
## Labels : Lake names (right) link points to lakes; raw area values (left) show actual sizes, clarifyi
```

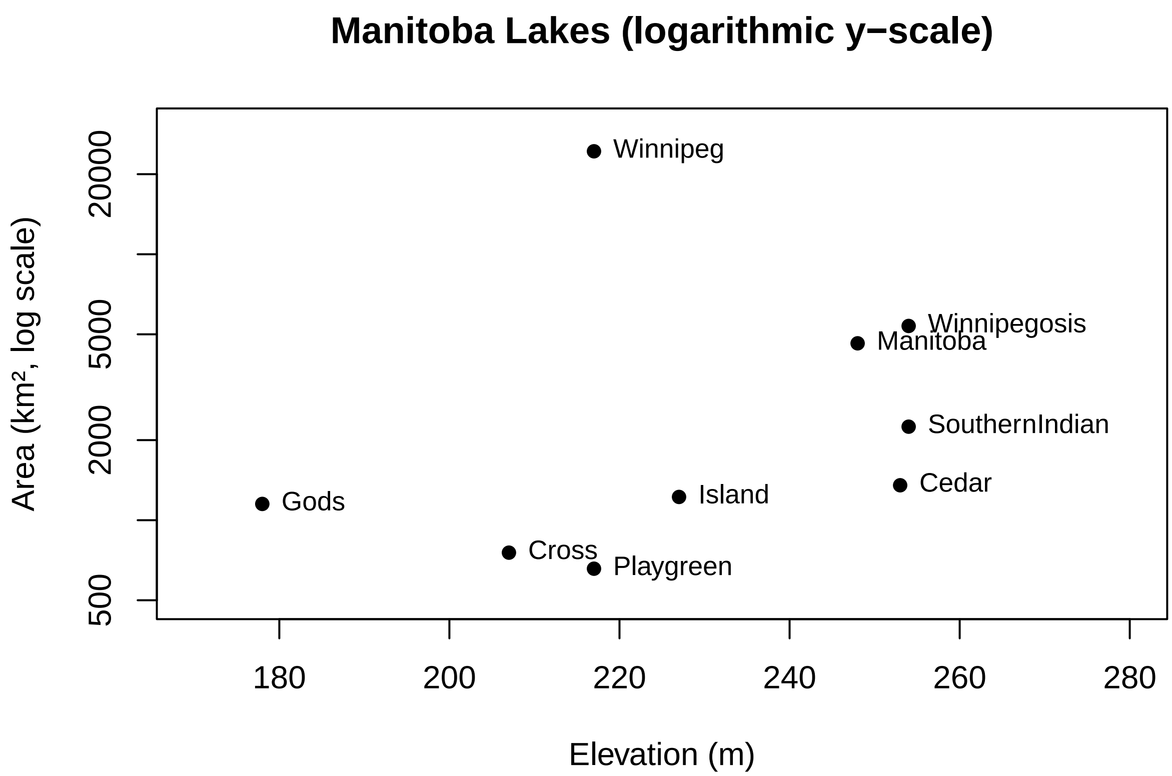
```r
cat("This setup uses vertical distance to represent multiplicative (doubling) area changes, making extr
```

```
## This setup uses vertical distance to represent multiplicative (doubling) area changes, making extrem
```

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.

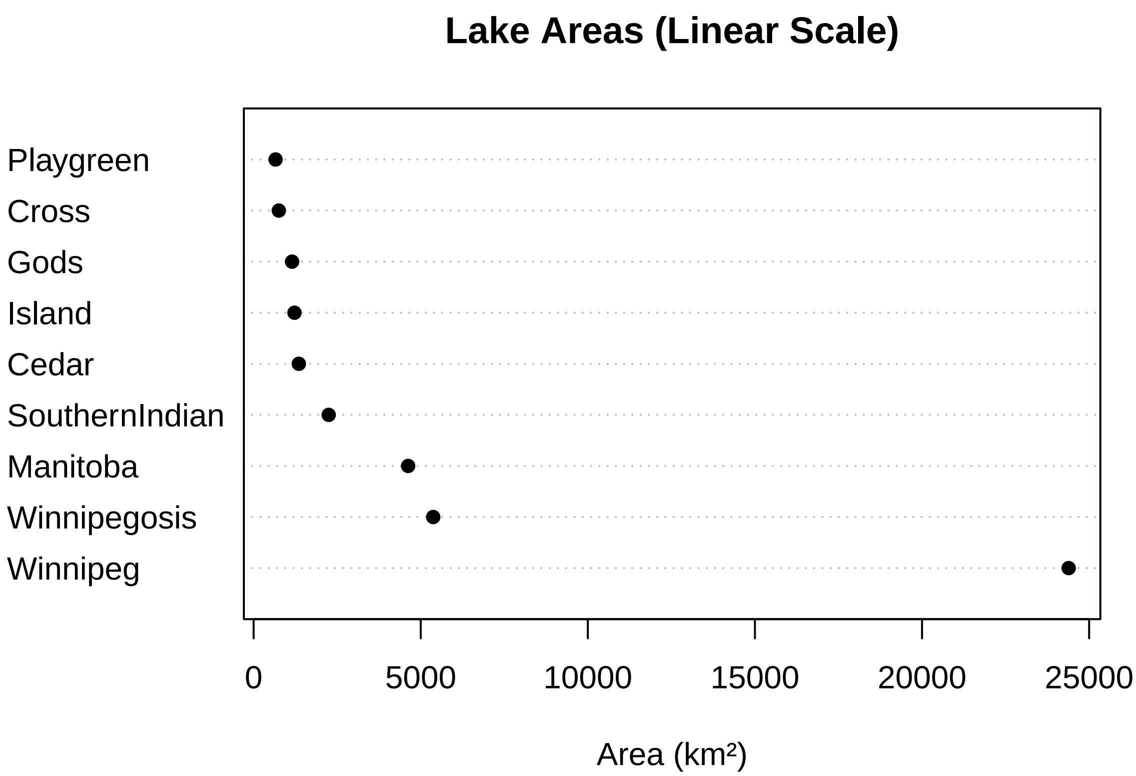
(b) Repeat the plot and associated labeling, now plotting area versus elevation, but specifying `ylog=TRUE` in order to obtain a logarithmic y-scale.

```r
#plot(area ~ elevation, pch=16, xlim=c(170,280), ylog=T)
#text(area ~ elevation, labels=row.names(Manitoba.lakes), pos=4, ylog=T)
#text(area ~ elevation, labels=area, pos=2, ylog=T)
#title("Manitoba's Largest Lakes")
plot(area ~ elevation, data = Manitoba.lakes, pch = 16, log = "y",
     xlim = c(170, 280), ylim = c(500, 30000),
     xlab = "Elevation (m)", ylab = "Area (km², log scale)")
text(Manitoba.lakes$elevation, Manitoba.lakes$area,
     labels = rownames(Manitoba.lakes), pos = 4, cex = 0.8)
title("Manitoba Lakes (logarithmic y-scale)")
```



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.

```
#dotchart(log2(area))
dotchart(Manitoba.lakes$area, labels = rownames(Manitoba.lakes),
         main = "Lake Areas (Linear Scale)",
         xlab = "Area (km²)", pch = 16)
```



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

```
total_water_area <- sum(Manitoba.lakes$area)
cat("Lower bound for Manitoba water area:",
    format(total_water_area, big.mark = ","), "km²\n")
```

```
## Lower bound for Manitoba water area: 41,771 km²
```

```
cat("(Sum of the 9 largest lakes)")
```

```
## (Sum of the 9 largest lakes)
```