Homework 1

ywx 3220101739

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 functio
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) 加载数据
iowa.df <- read.csv("data/Iowa.csv", sep = ";", header = TRUE)

# (b) 行数和列数
cat("Number of rows:", nrow(iowa.df), "\n")
```

```
## Number of rows: 33
```

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

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

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

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

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

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

```
# (e) 显示第 2 行
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")
max(vector1)
sort(vector1)
sum(vector1)
```

a. 分析

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

1. `max(vector1)`

   ```
   max(vector1)  # 返回 "7"
   ```

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

   解释：对字符型向量按字典序取最大值，“7” 是最大的。

2. `sort(vector1)`

   ```
   sort(vector1)  # 返回 c("12", "32", "5", "7")
   ```

   ```
   ## [1] "12" "32" "5"  "7"
   ```

解释：按字符的 ASCII 码顺序排序，有 “1”<“3”<“5”<“7”。

3. `sum(vector1)`

```r
sum(vector1)
```

```
## Error in sum(vector1): invalid 'type' (character) of argument
```

错误原因：字符型不能直接求和。

---

b. 分析

向量操作

```r
vector2 <- c("5", 7, 12)  # 自动转为字符型
class(vector2)            # 验证类型
```

```
## [1] "character"
```

1. 修正后的加法操作

```r
# 需要显式转换为数值型
as.numeric(vector2[2]) + as.numeric(vector2[3])  # 返回 19
```

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

修正：使用 `as.numeric()` 进行类型转换，原本的字符型无法相加（会显示错误）。

数据框操作

```r
dataframe3 <- data.frame(z1 = "5", z2 = 7, z3 = 12)
```

2. 数据框加法

```r
dataframe3[1,2] + dataframe3[1,3]  # 返回 19
```

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

原因：R 的 data.frame 允许不同列有不同的数据类型，由于 7 和 12 是数值型，所以可以相加。（保存的 “5” 是字符型）

列表操作

```
list4 <- list(z1 = "6", z2 = 42, z3 = "49", z4 = 126)
```

3. 列表元素加法

   ```
   list4[[2]] + list4[[4]]  # 返回 168
   ```

   ```
   ## [1] 168
   ```

4. 子列表加法（错误）

   ```
   list4[2] + list4[4]
   ```

   ```
   ## Error in list4[2] + list4[4]: non-numeric argument to binary operator
   ```

   原因：单中括号 [] 返回一个子列表，列表不能直接相加；双中括号 [[]] 直接提取元素本身（如果是数值，就是数值）。

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)`.

a

```
seq(from = 1, to = 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(from = 1, to = 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

1. 整体重复 3 次

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

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

2. 每个元素重复 3 次

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

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

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.

1. 数据准备

```r
# 加载 DAAG 包中的 orings 数据集
data(orings, package = "DAAG")
# 提取关键行 (1,2,4,11,13,18)
```

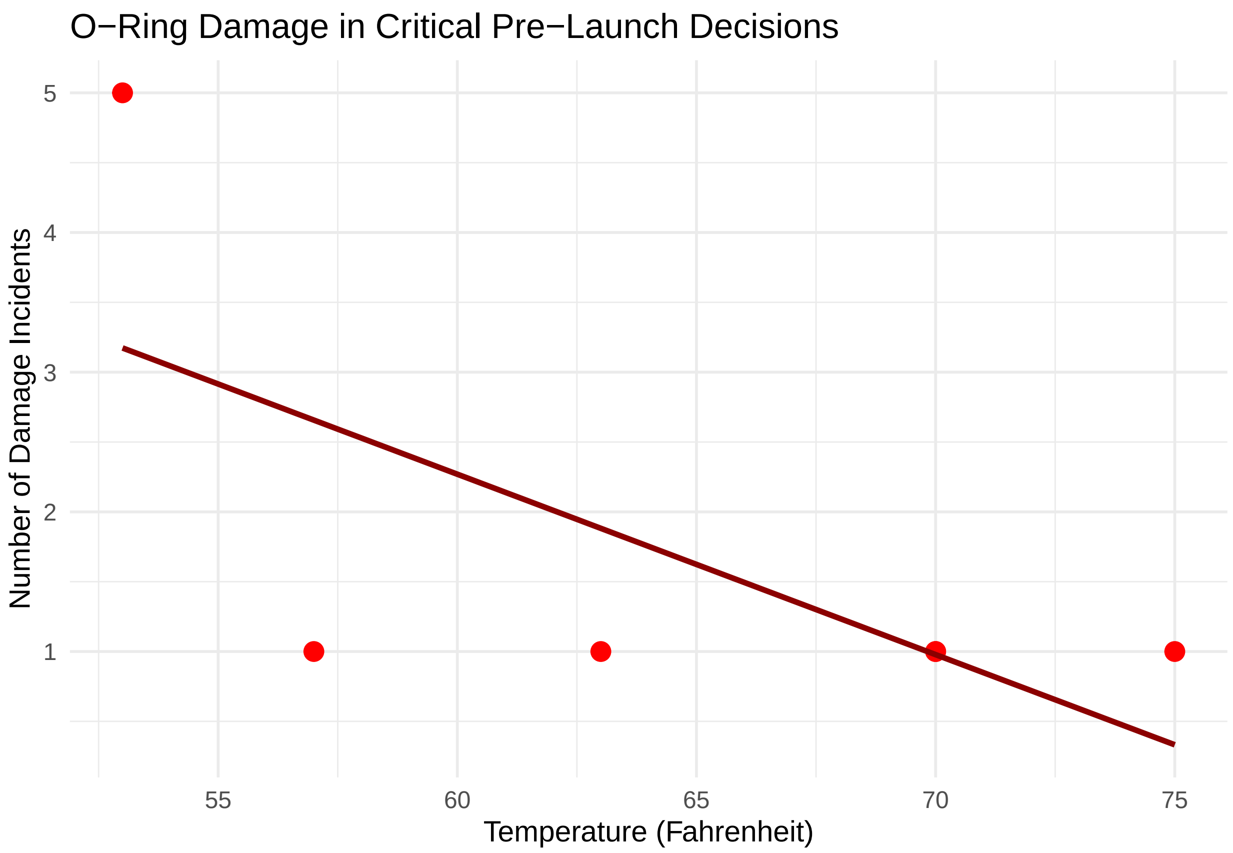
```
critical_rows <- c(1, 2, 4, 11, 13, 18)
orings_critical <- orings[critical_rows, ]
# 显示关键子集
head(orings_critical)
```

```
##    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
```

2. 关键子集数据可视化

```
ggplot(orings_critical, aes(x = Temperature, y = Total)) +
  geom_point(size = 3, color = "red") +
  geom_smooth(method = "lm", se = FALSE, color = "darkred") +
  labs(title = "O-Ring Damage in Critical Pre-Launch Decisions",
       x = "Temperature (Fahrenheit)",
       y = "Number of Damage Incidents") +
  theme_minimal()
```

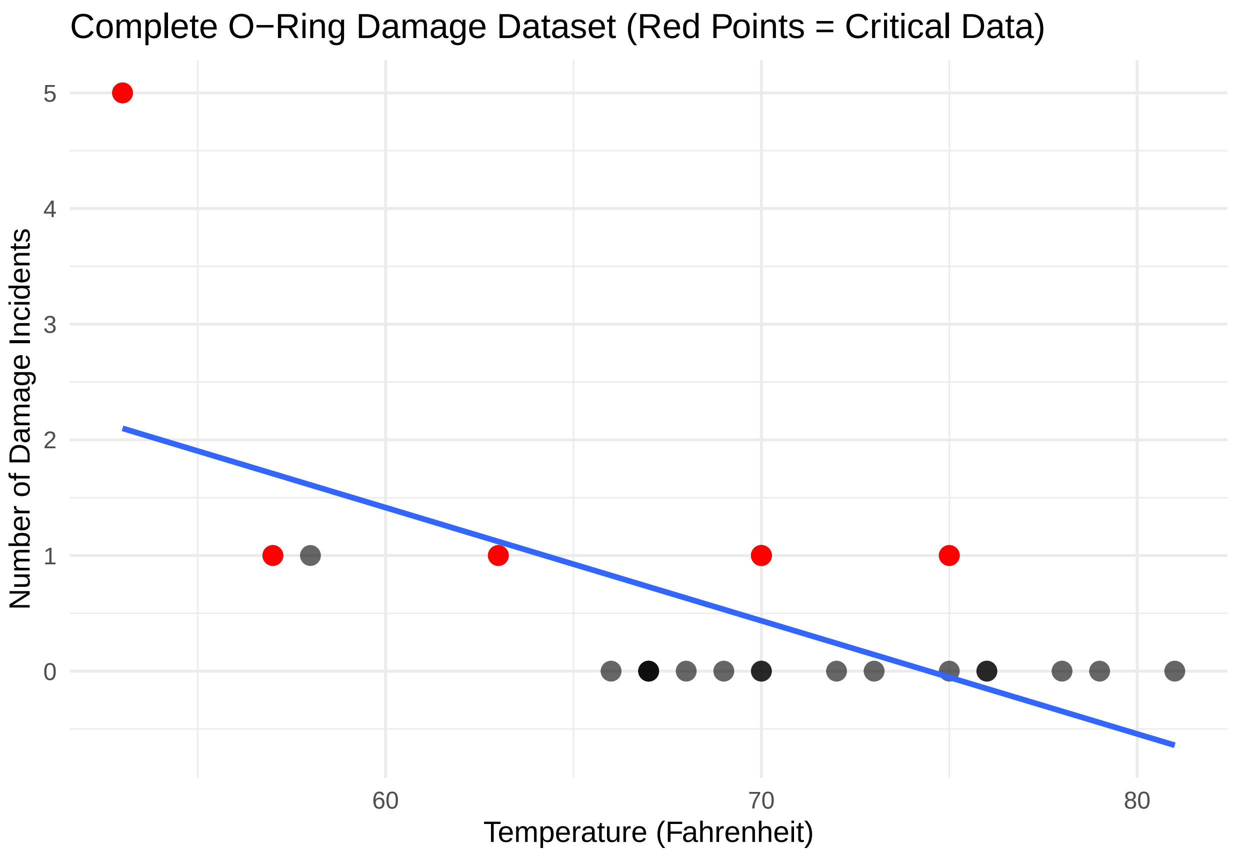
```
## `geom_smooth()` using formula = 'y ~ x'
```



3. 完整数据集可视化

```r
ggplot(orings, aes(x = Temperature, y = Total)) +
  geom_point(size = 3, alpha = 0.6) +
  geom_point(data = orings_critical,
             color = "red", size = 3) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Complete O-Ring Damage Dataset (Red Points = Critical Data)",
       x = "Temperature (Fahrenheit)",
       y = "Number of Damage Incidents") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



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.

(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?

a.

```
# 查看数据结构
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
# 检查缺失值
missing_values <- sapply(ais, function(x) sum(is.na(x)))

# 创建中文数据框（确保文件保存为UTF-8编码）
missing_df <- data.frame(
  变量名 = names(missing_values),
  缺失值数量 = unname(missing_values),  # 移除names属性
  row.names = NULL,
  stringsAsFactors = FALSE
)

# 显示美观表格
knitr::kable(missing_df,
             caption = "各变量缺失值统计",
             col.names = c("变量名称", "缺失值数量"))
```

表 1: 各变量缺失值统计

变量名称	缺失值数量
rcc	0
wcc	0
hc	0
hg	0
ferr	0
bmi	0
ssf	0
pcBfat	0
lbm	0
ht	0

变量名称	缺失值数量
wt	0
sex	0
sport	0

b. 性别比例分析

```r
# 创建交叉表
gender_table <- table(ais$sex, ais$sport)

# 转换为数据框并重命名列
gender_df <- as.data.frame(gender_table, responseName = "人数")
names(gender_df)[1:2] <- c("性别", "运动项目")

# 计算比例并筛选失衡项目
library(dplyr)
imbalance_details <- gender_df %>%
  group_by(运动项目) %>%
  mutate(
    总人数 = sum(人数),
    比例 = 人数/总人数
  ) %>%
  filter(比例 > 2/3) %>%  # 筛选比例 >66.7% 的项目
  mutate(
    对比性别 = ifelse(性别 == "m", "f", "m"),
    对比人数 = 总人数 - 人数,
    比例显示 = paste0(round(比例/(1-比例), 1), ":1")
  )

# 显示完整分布表
knitr::kable(gender_df, caption = "各运动项目男女运动员人数分布")
```

表 2: 各运动项目男女运动员人数分布

性别	运动项目	人数
f	B_Ball	13
m	B_Ball	12

性别	运动项目	人数
f	Field	7
m	Field	12
f	Gym	4
m	Gym	0
f	Netball	23
m	Netball	0
f	Row	22
m	Row	15
f	Swim	9
m	Swim	13
f	T_400m	11
m	T_400m	18
f	T_Sprnt	4
m	T_Sprnt	11
f	Tennis	7
m	Tennis	4
f	W_Polo	0
m	W_Polo	17

```r
# 显示详细的失衡项目信息
# cat("\n\n** 性别比例失衡 (>2:1) 的运动项目详情：**\n")
for(i in 1:nrow(imbalance_details)){
  item <- imbalance_details[i,]
  cat(sprintf(
    "%s：%s %d 人 vs %s %d 人（比例 %s）\n",
    item$运动项目,
    ifelse(item$性别=="m", " 男性", " 女性"),
    item$人数,
    ifelse(item$对比性别=="m", " 男性", " 女性"),
    item$对比人数,
    item$比例显示
  ))
}
```

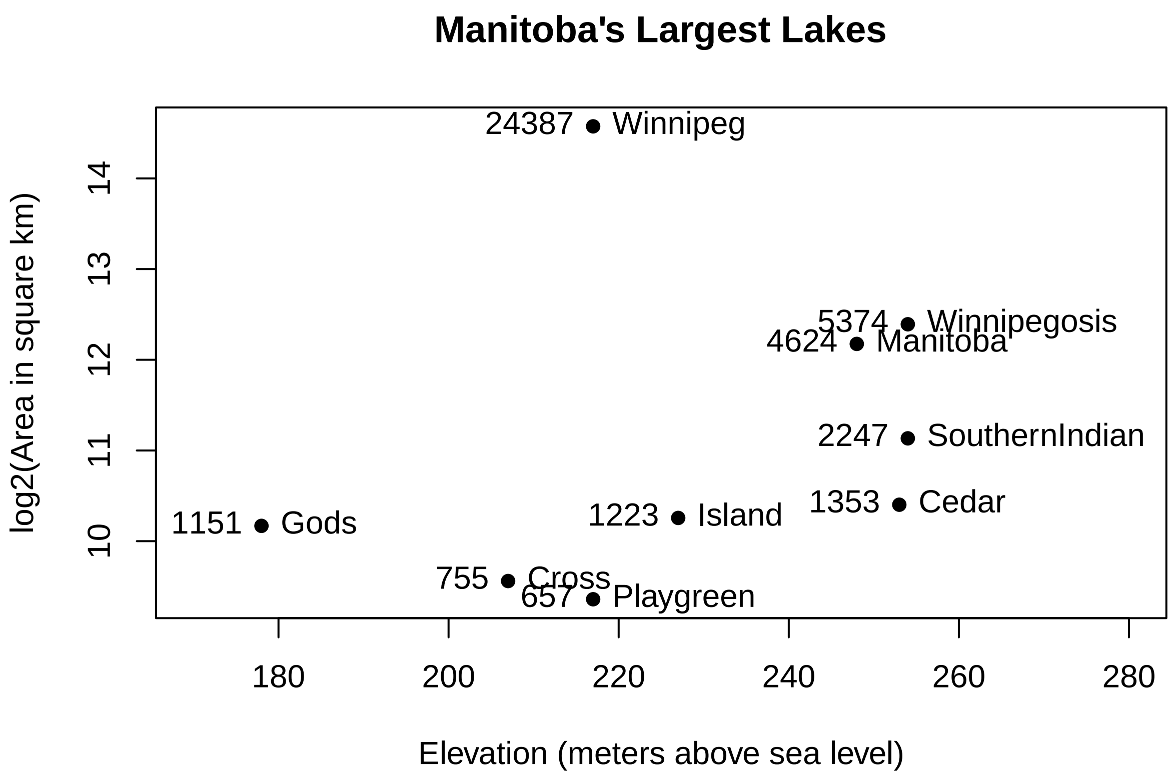
```
## Gym： 女性 4人 vs 男性 0人（比例 Inf:1）
## Netball： 女性 23人 vs 男性 0人（比例 Inf:1）
## T_Sprnt： 男性 11人 vs 女性 4人（比例 2.7:1）
## W_Polo： 男性 17人 vs 女性 0人（比例 Inf:1）
```

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):

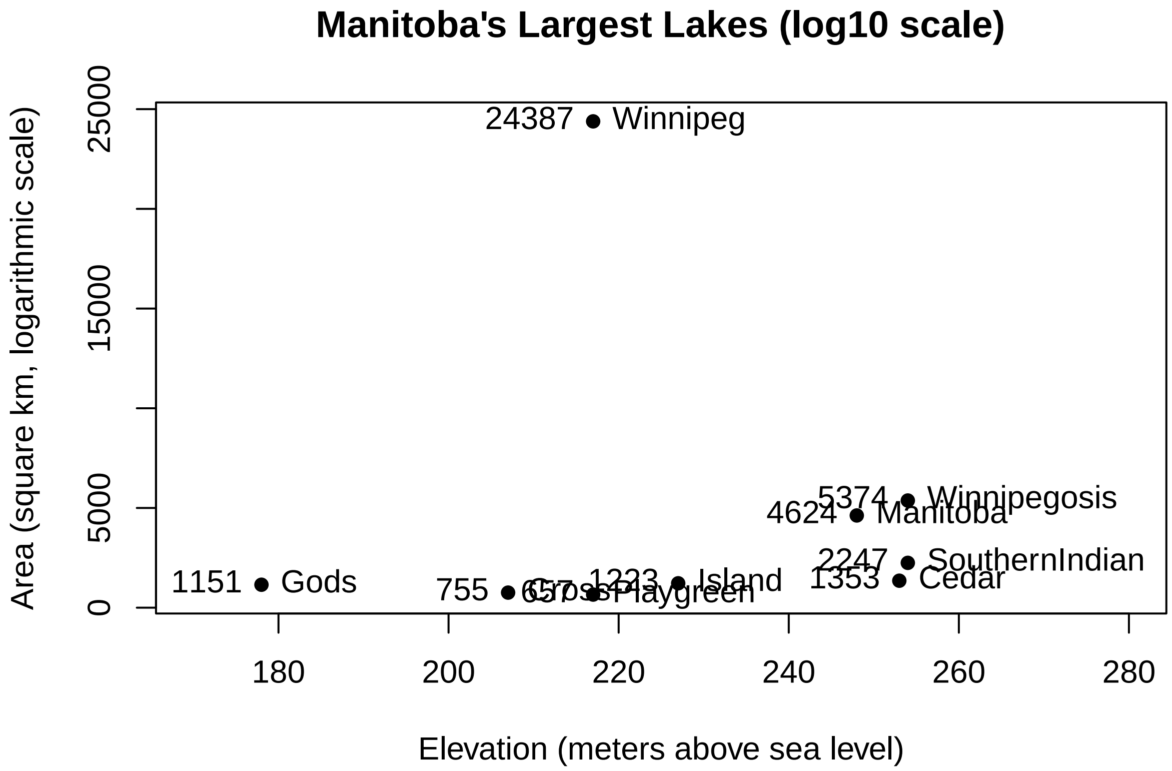
```r
# 创建 Manitoba.lakes 数据框
Manitoba.lakes <- data.frame(
  elevation = c(217, 254, 248, 254, 253, 227, 178, 207, 217),
  area = c(24387, 5374, 4624, 2247, 1353, 1223, 1151, 755, 657)
)

# 设置行名为湖泊名称
row.names(Manitoba.lakes) <- c("Winnipeg", "Winnipegosis", "Manitoba",
                              "SouthernIndian", "Cedar", "Island",
                              "Gods", "Cross", "Playgreen")

# 绘制图形
attach(Manitoba.lakes)
plot(log2(area) ~ elevation, pch=16, xlim=c(170,280),
     xlab="Elevation (meters above sea level)",
     ylab="log2(Area in square km)")
# 在点右侧添加湖泊名称标签
text(log2(area) ~ elevation, labels=row.names(Manitoba.lakes), pos=4)
# 在点左侧添加实际面积值
text(log2(area) ~ elevation, labels=area, pos=2)
title("Manitoba's Largest Lakes")
```



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.

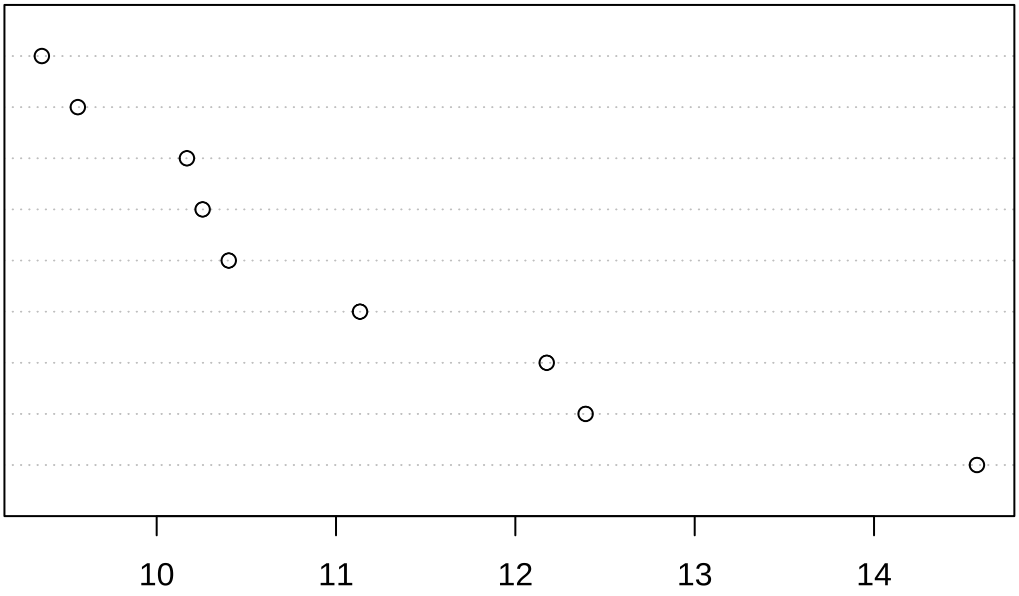
(b) Repeat the plot and associated labeling, now plotting area versus elevation, but specifying `ylog=TRUE` in order to obtain a logarithmic y-scale.

```
# 使用 ylog=TRUE 绘制对数坐标图形
plot(area ~ elevation,
     pch = 16,
     xlim = c(170, 280),
     ylog = TRUE,  # 使用对数 y 轴 (默认以 10 为底)
     xlab = "Elevation (meters above sea level)",
     ylab = "Area (square km, logarithmic scale)",
     main = "Manitoba's Largest Lakes (log10 scale)")

# 添加湖泊名称标签 (右侧)
text(area ~ elevation,
     labels = row.names(Manitoba.lakes),
     pos = 4,
     ylog = TRUE)  # 注意这里也需要指定 ylog=TRUE

# 添加面积数值标签（左侧）
text(area ~ elevation,
     labels = area,
     pos = 2,
     ylog = TRUE)  # 注意这里也需要指定 ylog=TRUE
```



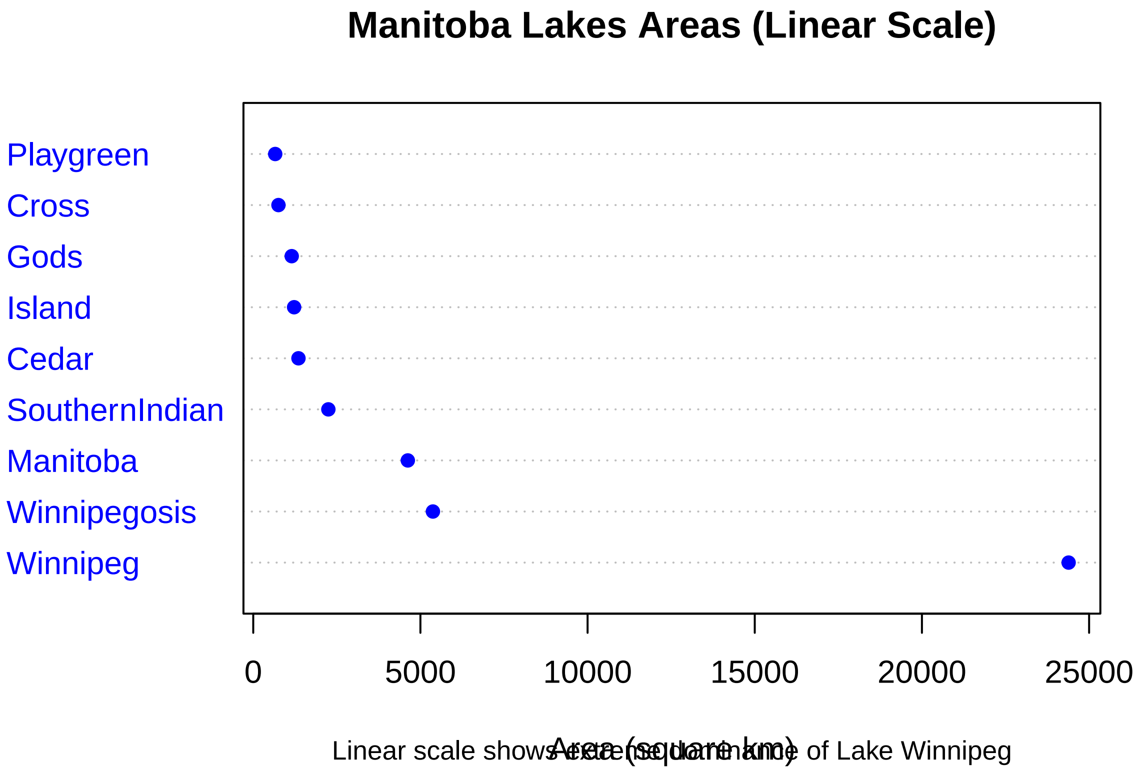
MB.Ch1.7. Look up the help page for the R function `dotchart()`. Use this function to display the areas of the Manitoba lakes (a) on a linear scale, and (b) on a logarithmic scale. Add, in each case, suitable labeling information.

```
dotchart(log2(area))
```



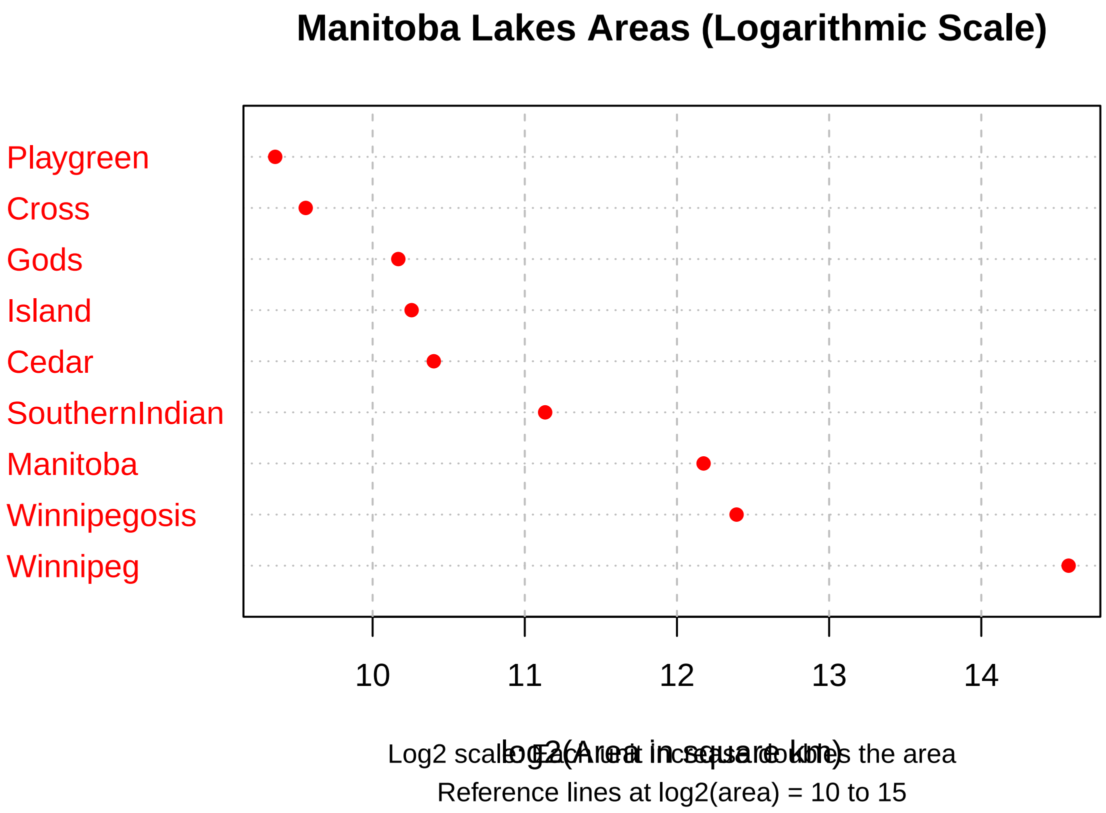
a. 线性

```
# (a) 线性尺度点图
dotchart(Manitoba.lakes$area,
         labels = row.names(Manitoba.lakes),
         xlab = "Area (square km)",
         main = "Manitoba Lakes Areas (Linear Scale)",
         pch = 16,
         color = "blue")

# 添加说明
mtext("Linear scale shows extreme dominance of Lake Winnipeg",
      side = 1, line = 3, cex = 0.8)
```



对数

```r
# (b) 对数尺度点图
dotchart(log2(Manitoba.lakes$area),
         labels = row.names(Manitoba.lakes),
         xlab = "log2(Area in square km)",
         main = "Manitoba Lakes Areas (Logarithmic Scale)",
         pch = 16,
         color = "red")

# 添加参考线
abline(v = seq(10, 15, by = 1), col = "gray", lty = 2)

# 添加解释性文本
mtext("Log2 scale: Each unit increase doubles the area",
      side = 1, line = 3, cex = 0.8)
mtext("Reference lines at log2(area) = 10 to 15",
      side = 1, line = 4, cex = 0.8)
```



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

```
# 计算所有湖泊面积总和
total_water_area <- sum(Manitoba.lakes$area)

# 打印结果
cat("Lower bound estimate for Manitoba's water-covered area:",
    format(total_water_area, big.mark = ","), "square kilometers\n")
```

```
## Lower bound estimate for Manitoba's water-covered area: 41,771 square kilometers
```