Homework 1: STAT5400

Dev Narayan Baiju

2024-09-04

Question 1

The volume of a sphere of radius r is given by $(4 * pi * r^3)/3$. For spheres having radii 3, 4, 5, . . . , 20 find the corresponding volumes and print the results out in a table. Use the technique of section 2.1.5 to construct a data frame with columns radius and volume.

The following is the code for creating the radii and volume vector

```
r = 3:20
v = (4*pi*r^3)/3
```

The next code will create the data frame and print the same.

```
df_volume = data.frame(Radii=r, Volumes=v)
print(df_volume)
```

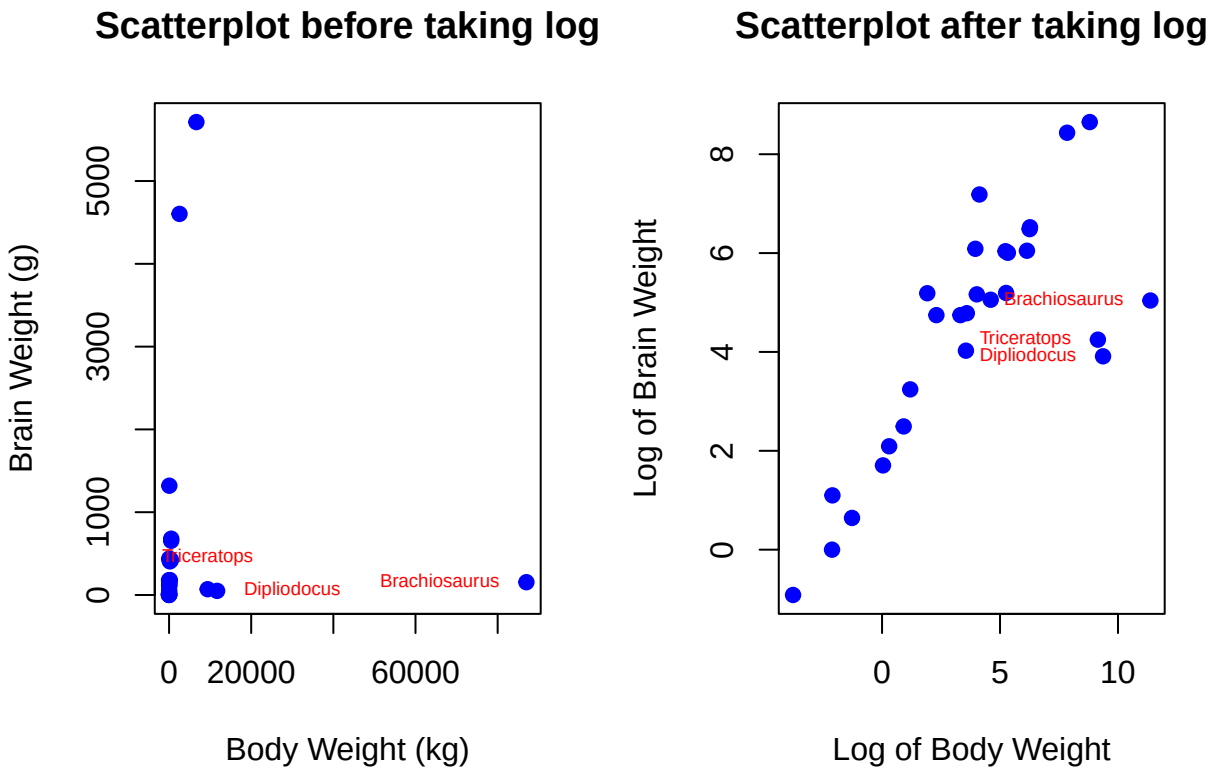
```
##    Radii    Volumes
## 1      3   113.0973
## 2      4   268.0826
## 3      5   523.5988
## 4      6   904.7787
## 5      7  1436.7550
## 6      8  2144.6606
## 7      9  3053.6281
## 8     10  4188.7902
## 9     11  5575.2798
## 10    12  7238.2295
## 11    13  9202.7721
## 12    14 11494.0403
## 13    15 14137.1669
## 14    16 17157.2847
## 15    17 20579.5263
## 16    18 24429.0245
## 17    19 28730.9120
## 18    20 33510.3216
```

Question 2

Plot the graph of brain weight (brain) versus body weight (body) for the data set Animals from the MASS package. Label the axes appropriately. Alongside on the same graphics page, log(brain weight) versus log(body weight). Use the row labels to label the points with the three largest body weight values. Label the axes in untransformed units. [To access this data frame, specify library(MASS)]

```
library(MASS)
data("Animals")
larg_body_w <- order(Animals$body, decreasing = TRUE)
```

```
larg_body_w = larg_body_w[1:3]
par(mfrow = c(1,2))
plot(Animals$body, Animals$brain,
     xlab = "Body Weight (kg)", ylab = "Brain Weight (g)",
     main = "Scatterplot before taking log",
     pch = 19, col = "blue")
text(Animals$body[larg_body_w], Animals$brain[larg_body_w],
     labels =  rownames(Animals)[larg_body_w],
     pos = c(2,4,3), cex = 0.6, offset = 0.7, col = "red")
plot(log(Animals$body), log(Animals$brain),
     xlab = "Log of Body Weight", ylab = "Log of Brain Weight",
     main = "Scatterplot after taking log",
     pch = 19, col = "blue")
text(log(Animals$body[larg_body_w]), log(Animals$brain[larg_body_w]),
     labels = rownames(Animals)[larg_body_w],
     pos = c(2,2,2), cex = 0.6, offset = 0.7, col = "red")
```




```
par(mfrow = c(1,1))
```

Question 3

In the following page, pick three R functions that you are interested in but haven't used before. Show a simple example to demonstrate how to use them. http://adv-r.had.co.nz/Vocabulary.html

1. tapply() A quick demo of the tapply is as follows:

```
df = data.frame(Teams = c("A","A","A","A","B","B","B","C","C","C"),
                Score = c(10,6,8,5,9,7,8,4,7,10))
tapply(df$Score, df$Teams, median)
```

```
## A B C
```

```
## 7 8 7
```

tapply() function applies a given function to a vector and also takes into account an index vector, if you have to apply the function group-wise. The mean function is applied to the Score of the Teams, and the means are grouped by the Team names.

2. choose

```
choose(12,4)
```

```
## [1] 495
```

choose() function does the combination operation by inputting the values of n and k.

3. switch

```
switch("AB", "AA" = 10, "AB" = 20, "AC" = 30)
```

```
## [1] 20
```

switch() function returns the argument value that is equal with the expression(first argument).

Question 4

Write your own factorial function. Compare the run time of your function and the factorial function in R. (Besides writing a loop, try the Reduce function).

```
library(microbenchmark)
library(purrr)
Myfact <- function(q){
  f1 <- 1
  for(i in 1:q)
    f1 <- f1*i
return(f1)
}

reduce_fact <- function(q){
  f2 = reduce("*", 1:q, accumulate = FALSE)
  return(f2)
}
q = 8
microbenchmark(Myfact(q), reduce_fact(q), factorial(q))
```

```
## Unit: nanoseconds
##            expr   min      lq      mean median      uq      max neval
##       Myfact(q)   693   772.5 144673.65    919  1018.0 14373293   100
##  reduce_fact(q) 26986 27500.0  56417.15  27844 28547.0  2313518   100
##    factorial(q)   380   404.0    622.71    512   579.5    11433   100
```

The most efficient function is the in-built factorial function.

Question 5

Write a R function to pick the unique element that appears an odd number of times in a sequence. The following code is given:

```
set.seed(1)
n <- 5
p <- sample(n, n, replace=TRUE)
(myseq <- sample(c(p, p))[seq(2*n-1)])
```

```
## [1] 4 4 1 1 1 5 1 2 5
```

The following function will output the unique element that appears an odd number of times:

```
ret_odd <- function(myseq){
  df_freq = as.data.frame(table(myseq))
  indices_odd = which(df_freq$Freq %% 2 == 1)
  return(unique(df_freq$myseq)[indices_odd])
}
print(as.integer(ret_odd(myseq)))
```

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