HW 1 Introduction to R

STAT 5400 Wylan Gao

Due: Sep 6, 2024. 09:30 AM

```
### Problems
1. The following problems from the online book called Using R for Data Analysis and Graphics b
y Maindonald. <https://cran.r-project.org/doc/contrib/usingR.pdf>.



* Problem 5, page 20. Note pi is missing in the formula calculating the volume of a sphere.
```

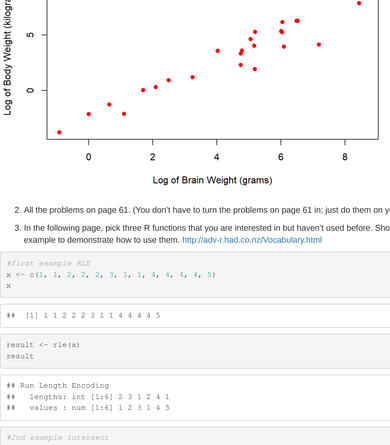
```
#this is for problem 1
Radius <- 3:20

sphere <- 4 * (Radius^3/3)

conversion <- data.frame(radius = Radius, Volume = sphere)

conversion
```

radius <int>	Volume <dbl>
3	36.00000
4	85.33333
5	166.66667
6	288.00000
7	457.33333
8	682.66667
9	972.00000
10	1333.33333
11	1774.66667
12	2304.00000

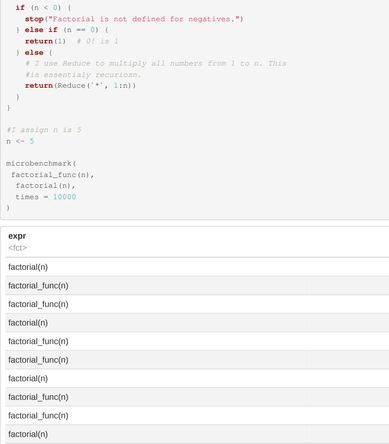
1-10 of 18 rows Previous 1 2 Next

- Problem 2, page 30.

```
library(MASS)


plot(Animals$brain, Animals$body,
     xlab = "Brain Weight (grams)",
     ylab = "Body Weight (kilograms)",
     main = "Brain vs. Body Weight of Animals",
     pch = 16, col = "red")

largest_indices <- sort(Animals$body, decreasing = TRUE)[1:3]

text(Animals$brain[largest_indices], Animals$body[largest_indices],
     labels = rownames(Animals)[largest_indices],
     pos = 4, cex = 0.8, col = "blue")
```




```
plot(log(Animals$brain), log(Animals$body),
     xlab = "Log of Brain Weight (grams)",
     ylab = "Log of Body Weight (kilograms)",
     main = "Log Version of Brain vs. Body Weight of Animals",
     pch = 16, col = "red")

text(log(Animals$brain[largest_indices]), log(Animals$body[largest_indices]),
     labels = rownames(Animals)[largest_indices],
     pos = 4, cex = 0.8, col = "blue")
```



2. All the problems on page 61. (You don't have to turn the problems on page 61 in; just do them on your own.)
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

```
#first example RLE
x <- c(1, 1, 2, 2, 2, 3, 1, 1, 4, 4, 4, 4, 5)
x
```

```
##  [1] 1 1 2 2 2 3 1 1 4 4 4 4 5
```

```
result <- rle(x)
result
```

```
## Run Length Encoding
##   lengths: int [1:6] 2 3 1 2 4 1
##   values : num [1:6] 1 2 3 1 4 5
```

```
#2nd example intersect
y <- c(1, 1, 2, 2, 2, 3, 1, 4, 5)
common_elements <- intersect(x, y)
common_elements
```

```
## [1] 1 2 3 4 5
```

```
#3rd example is union
amazon <- union(x, y)
amazon
```

```
## [1] 1 2 3 4 5
```

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)

factorial_func <- function(n) {
  # base case
  if (n < 0) {
    stop("Factorial is not defined for negatives.")
  } else if (n == 0) {
    return(1)  # 0! is 1
  } else {
    # I use Reduce to multiply all numbers from 1 to n. This
    #is essentialy recurison.
    return(Reduce('*', 1:n))
  }
}

#I assign n is 5
n <- 5

microbenchmark(
 factorial_func(n),
 factorial(n),
 times = 10000
)
```

expr <fct>	time <dbl>
factorial(n)	28900
factorial_func(n)	250400
factorial_func(n)	22974600
factorial(n)	2700
factorial_func(n)	11200
factorial_func(n)	8500
factorial(n)	1000
factorial_func(n)	8000
factorial_func(n)	7800
factorial(n)	900

1-10 of 10,000 rows Previous 1 2 3 4 5 6 … 1000 Next

```
#this concludes that factorial funciton is more efficient. Probably written in C
```

5. Write a R function to pick the unique element that appears an odd number of times in a sequence.

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

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

```
# myfunction(myseq) should output 2
```