Code

Castellano_CS636_Lab03

Question #1

Please read the train.csv file into R and store the data in a variable called “X”.

Hide

```
library(readr)
X <- read_csv('C:/Users/Castellano/Documents/Spring2020/CS636/Home Depot/train.csv/tra
in.csv')
```

Question #2

Write a function, called “distinct_relevance”, to count how many distinct values are in the column “relevance”? So when we call the function, it returns the desired results: distinct_relevance (vect = X$relevance);

Hide

```
distinct_relevance <- function(vector) {
  elementos <- c(rep(0,length(vector)))
  for (i in 1:length(vector)) {
    boo <- vector[i] %in% elementos
   if( boo == FALSE) {
    #print('Not in')
     elementos[i] <- vector[i]
   } else {
      next
    }
  }
  elementos <- elementos[!elementos %in% 0]
  return(elementos)
}

relevant <- distinct_relevance(X$relevance)
print(relevant)
```

```
 [1] 3.00 2.50 2.33 2.67 2.00 1.00 1.67 1.33 1.25 2.75 1.75 1.50 2.25
```

3, Write a function, called “count”, to count the number of appearances of a value, e.g. 3, in the column “relevance”, so when we call the function, it returns the desired results: count(vect = X$relevance, value=3); (For Q2 and Q3, please do not use existing R packages or functions.)

Hide

```
count <- function(vector, value) {
  cuenta <- 0
  for (i in 1:length(vector)) {
   if( vector[i] == value) {
     cuenta <- cuenta + 1
   }
  }
  return(cuenta)
}

count(X$relevance,3.00)
```

```
[1] 19125
```

Hide

```
table(X$relevance)
```

```


    1  1.25  1.33   1.5  1.67  1.75     2  2.25  2.33   2.5  2.67  2.75     3 
 2105     4  3006     5  6780     9 11730    11 16060    19 15202    11 19125 
```

Question #4

Compare the results with R function: table()

Hide

```
system.time(relevance_values <- distinct_relevance(X$relevance))
```

```
   user  system elapsed 
  11.75    9.10   20.84 
```

Hide

```
relevance_counts <- count(X$relevance,3.00)
system.time(table(X$relevance))
```

```
   user  system elapsed 
   0.08    0.00    0.08 
```

Hmmmmmm, the table() function is A LOT more efficient. Wondering why.

5. Pi can be computed by adding the following terms (http://en.wikipedia.org/wiki/Pi (http://en.wikipedia.org/wiki/Pi)): How many terms does it take to get the first 3 digits to be correct, 3.14? Write an R function getPi(N) to compute it, where N specifies the first N digits to be correct, and returns #terms.

Hide

```
getPi <- function(N){
  'This function only works for N < 10.
  If a higher precision is desired, this function can be modified as
  an implementation of a while loop instead.'
  old_pie <- 4
  for (k in 1:10^N){
    new_pie <- old_pie + 4*((-1)^k)/(2*k+1)
     #print(new_pie)
  if ( abs(new_pie-old_pie) < 1*10^(-(N-1)  )) {
    k <- paste(toString(k), "Iterations Required")
    new_pie <- round(new_pie, digits = N + 1)
    Y <- list(k, new_pie)
    break
  }
    old_pie <- new_pie
  }

  return(Y)
}
N <- getPi(3)
print(N)
```

```
[[1]]
[1] "200 Iterations Required"

[[2]]
[1] 3.1466
```

Hide

```
class(N)
```

```
[1] "list"
```