Assignment3

Yan Zichu,300476924

9/21/2020

Q1. (3 Mark)

```
this.list <- list(tags=LETTERS[1:20],1:10,names=c("Catherine","Maui"),diag(c(1,3,5)))
```

a. How long is the list? **ANS**

```
length(this.list)
```

```
## [1] 4
```

b. Write code to extract the vector 1:10 from the list. **ANS**

```
vec <- this.list[[2]]
vec
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

c. What type of object is produced by the following code? (this.list[c("tags","names")]) **ANS**

```
c <- this.list[c("tags","names")]
str(c)
```

```
## List of 2
##  $ tags : chr [1:20] "A" "B" "C" "D" ...
##  $ names: chr [1:2] "Catherine" "Maui"
```

Q2. (14 Marks)

```
library(dplyr)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tibble)
library(tidyr)
vehicles <- read.csv("motor_vehicle_modified.csv",stringsAsFactors = FALSE)
```

a. Find out how many vehicles have transmission type "4-gear auto" and are make Kia or Honda. [3 marks] **ANS**

```
n <- vehicles[(vehicles$make =="Kia"|vehicles$make =="Honda") & vehicles$transmission_type == "4-gear au
nrow(n)
```

```
## [1] 13
```

```
m <- filter(vehicles,make %in% c("Kia","Honda"), transmission_type == "4-gear auto")
nrow(m)
```

```
## [1] 13
```

b. Drop the columns vehicle_usage and vehicle_type from the dataset. [1 mark] **ANS**

```
vehicles$vehicle_usage <- NULL
vehicles$vehicle_type <- NULL
ncol(vehicles)
```

```
## [1] 25
```

```
vehicles <- read.csv("motor_vehicle_modified.csv",stringsAsFactors = FALSE)
ncol(vehicles)
```

```
## [1] 27
```

```
n <- select(vehicles,-vehicle_usage,-vehicle_type)
ncol(n)
```

```
## [1] 25
```

c. Use the cut() function to create a new column called cc_rating_group which groups the cc_rating column into three levels labelled low, medium and high. [2 marks] **ANS**

```r
new <- mutate(vehicles, cc_rating_group = cut(cc_rating, 3, labels=c("low", "medium", "high")))
head(new,3)
```

```
##   objectid basic_colour     body_type cc_rating chassis7
## 1     6001        white station wagon      2199
## 2     6002       silver station wagon      2354
## 3     6003         grey     hatchback      2261
##   first_nz_registration_year first_nz_registration_month gross_vehicle_mass
## 1                       2011                           9               2510
## 2                       2012                           1               2270
## 3                       2009                           4               2165
##   height import_status  make   model motive_power number_of_axles
## 1      0           new   Kia Sorento       diesel               0
## 2      0           new Honda Odyssey       petrol               2
## 3      0           new  Ford  Mondeo       petrol               2
##   number_of_seats      nz_assembled original_country power_rating
## 1               7 imported built-up      South Korea          145
## 2               7 imported built-up            Japan          133
## 3               5 imported built-up          Belgium          118
##   previous_country road_transport_code             submodel
## 1             None                     2.2 diesel auto ex
## 2             None                                      s
## 3             None                   mondeo zetec 2.3 aut
##                    TLA transmission_type vdam_weight      vehicle_type
## 1         Dunedin City       6-gear auto           0 passenger car/van
## 2     Hauraki District       5-gear auto           0 passenger car/van
## 3 Kapiti Coast District      6-gear auto           0 passenger car/van
##       vehicle_usage vehicle_year cc_rating_group
## 1 private passenger         2011             low
## 2 private passenger         2012             low
## 3 private passenger         2009             low
```

d. Produce a table summarising the vehicles from the dataset vehicles, showing the median NZ registration year for each cc rating group. [3 marks] **ANS**

```r
tapply(new$first_nz_registration_year,new$cc_rating_group, median)
```

```
##    low medium   high
##   2008   1995   1995
```

```r
new2 <- group_by(new, cc_rating_group)
summarise(new2, median(first_nz_registration_year),.groups = 'drop')
```

```
## # A tibble: 3 x 2
##   cc_rating_group `median(first_nz_registration_year)`
##   <fct>                                          <dbl>
## 1 low                                             2008
## 2 medium                                          1995
## 3 high                                            1995
```

e. Create, but DO NOT DISPLAY, a contingency table called vehicles_country_status, giving the number of vehicles and for every combination of original country and import status. [2 marks] **ANS**

```r
#vehicles_country_status <- group_by(vehicles, original_country, import_status) %>%
#                           select(original_country,import_status) %>%
#                           mutate(number = n())

#vehicles_country_status

vehicles_country_status <- group_by(vehicles, original_country, import_status)
r <- summarise(vehicles_country_status,number=n(),.groups = 'drop')
r
```

```
## # A tibble: 49 x 3
##    original_country import_status number
##    <chr>            <chr>          <int>
##  1 Australia        new              305
##  2 Australia        re-reg             8
##  3 Australia        used               4
##  4 Austria          new                4
##  5 Belgium          new               30
##  6 Canada           re-reg             3
##  7 China            new               25
##  8 China            used               1
##  9 Czech Republic   new                6
## 10 Czech Republic   used               1
## # ... with 39 more rows
```

f. Now sort the countries in that table in decreasing order of the number of used cars, and keep the top 3 countries (still list all import statuses for those countries), and display the resulting table. [3 marks]
ANS

```r
arrange(r, desc(number))
```

```
## # A tibble: 49 x 3
##    original_country import_status number
##    <chr>            <chr>          <int>
##  1 Japan            new             1318
##  2 Japan            used            1172
##  3 Not Known        new              660
##  4 New Zealand      new              355
##  5 Australia        new              305
##  6 Thailand         new              215
##  7 South Korea      new              178
##  8 Germany          new              175
##  9 Germany          used             137
## 10 United Kingdom   new               79
## # ... with 39 more rows
```

Q3. (14 Marks)

```r
library(ggplot2)
library(ggthemes)
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
earnings_wide <- read.csv("average_weekly_earnings.csv", stringsAsFactors = FALSE)
earnings_wide$Date <- as.Date(as.yearqtr(earnings_wide$Date, format="%YQ%q"))
```

a. Convert the data from wide format to long format. Use the pivot_longer() function from tidyr. Name the new long-form dataset earnings_long and name the new earnings column Average.Weekly.Earnings. [3 marks] **ANS**

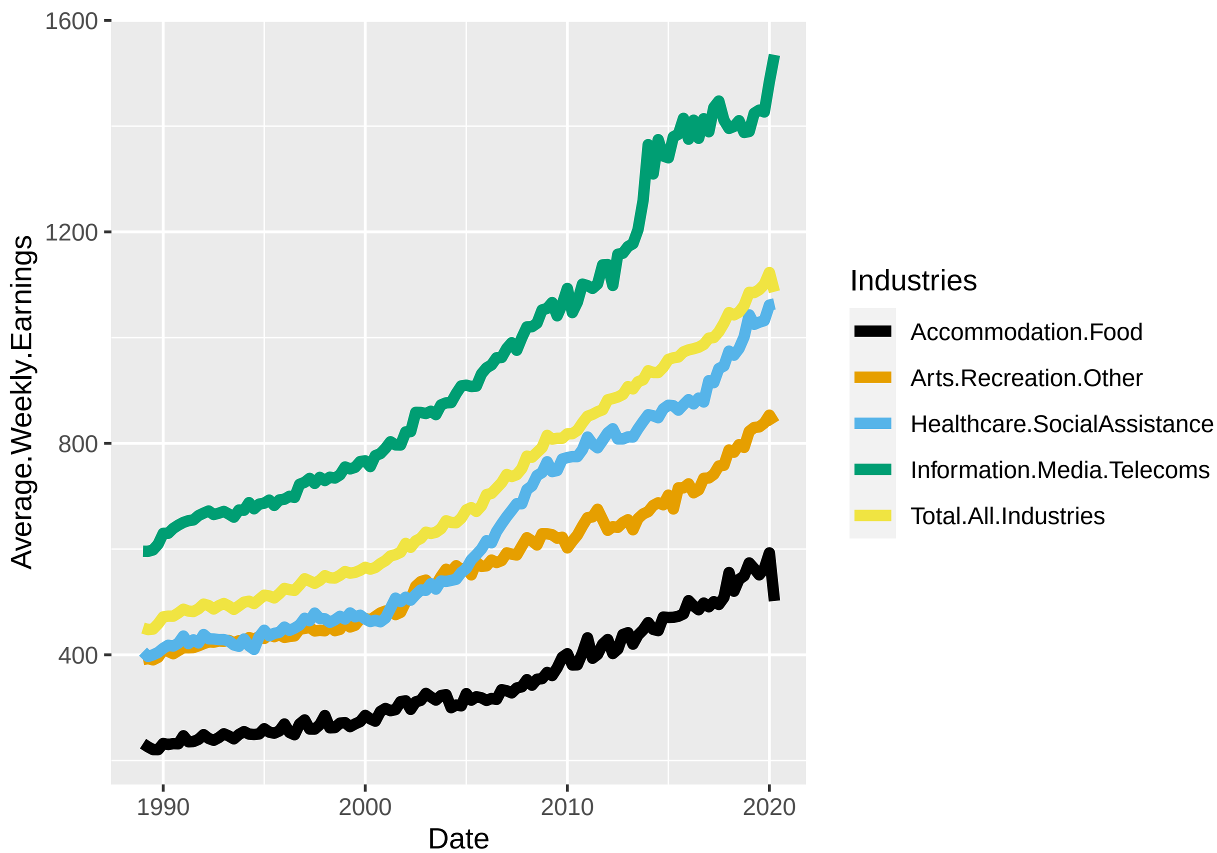
```r
earnings_long <- pivot_longer(earnings_wide, cols=c(Information.Media.Telecoms, Healthcare.SocialAssista

earnings_long
```

```
## # A tibble: 630 x 3
##    Date       Industries                    Average.Weekly.Earnings
##    <date>     <chr>                                           <dbl>
##  1 1989-01-01 Information.Media.Telecoms                       596.
##  2 1989-01-01 Healthcare.SocialAssistance                      406.
##  3 1989-01-01 Arts.Recreation.Other                            391.
##  4 1989-01-01 Accommodation.Food                               232.
##  5 1989-01-01 Total.All.Industries                             451.
##  6 1989-04-01 Information.Media.Telecoms                       596.
##  7 1989-04-01 Healthcare.SocialAssistance                      397.
##  8 1989-04-01 Arts.Recreation.Other                            393.
##  9 1989-04-01 Accommodation.Food                               226.
## 10 1989-04-01 Total.All.Industries                             448.
## # ... with 620 more rows
```

b. Produce a line plot of the long-format data using the ggplot2 package. Colour the lines by industry, and use a colorblind-friendly colour palette. Relabel the plot axes to make them more readable. [5 marks] **ANS**

```r
ggplot(earnings_long, aes(x=Date, color=Industries, y=Average.Weekly.Earnings)) +
    geom_line(stat="identity",size = 2)+ scale_color_colorblind()
```



c. Copy the following code into your answer script, and run it. Then write code to summarise the data up to 2005 to find the maximum earnings value within each time period, for each industry separately. Repeat for the data since 2005. [2 marks]

```
## Method 1
earnings_long_to2005 <- earnings_long[earnings_long$Date < '2006-01-01',]
earnings_long_to2020 <- earnings_long[earnings_long$Date >= '2006-01-01',]

## Method 2
earnings_long_to2005 <- filter(earnings_long, Date < '2006-01-01')
earnings_long_to2020 <- filter(earnings_long, Date >= '2006-01-01')
```

ANS

```
s <-group_by(earnings_long_to2005,Industries) %>%
    filter(Average.Weekly.Earnings == max(Average.Weekly.Earnings)) %>%
    select(Date,maximum_earnings_value=Average.Weekly.Earnings)
```

```
## Adding missing grouping variables: `Industries`
```

```
s
```

```
## # A tibble: 5 x 3
## # Groups:   Industries [5]
##   Industries                  Date       maximum_earnings_value
```

```
##   <chr>                        <date>                          <dbl>
## 1 Accommodation.Food           2003-01-01                       327.
## 2 Arts.Recreation.Other        2005-07-01                       575.
## 3 Information.Media.Telecoms   2005-10-01                       931.
## 4 Healthcare.SocialAssistance  2005-10-01                       600.
## 5 Total.All.Industries         2005-10-01                       682.
```

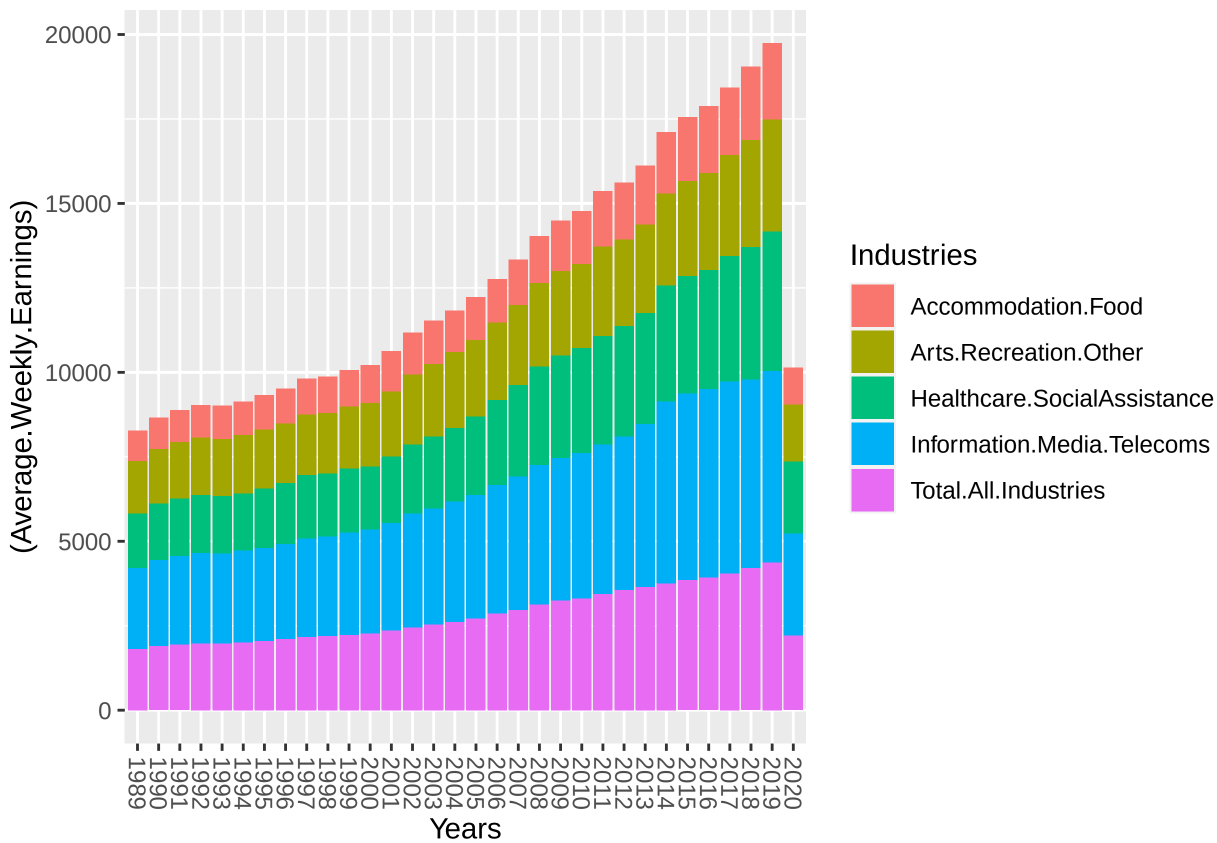
d. Add an extra column Years to each of the summaries (up to 2005 and after 2005) to indicate which time period it covers (as a character variable), then combine the two datasets using rbind(). [3 marks] **ANS**

```
a <- mutate(earnings_long_to2005,Years = substring(Date,1,4))
b <- mutate(earnings_long_to2020,Years = substring(Date,1,4))
c <- rbind(a,b)
head(c,3)
```

```
## # A tibble: 3 x 4
##   Date       Industries                  Average.Weekly.Earnings Years
##   <date>     <chr>                                         <dbl> <chr>
## 1 1989-01-01 Information.Media.Telecoms                     596. 1989
## 2 1989-01-01 Healthcare.SocialAssistance                    406. 1989
## 3 1989-01-01 Arts.Recreation.Other                          391. 1989
```

e. Plot a column chart of the highest earnings values by Years, again using ggplot2. The bars should be coloured by industry and positioned side-to-side, not stacked on top of each other. [4 marks] **ANS**

```
ggplot(c, aes(x=Years, fill=Industries, y = (Average.Weekly.Earnings))) +
    geom_bar(stat='identity')+  theme(axis.text.x=element_text(angle = -90, vjust=0.5))
```



Q4. (4 Marks) Copy the following code into your RMarkdown answers document. It defines a function, and then runs that function.

Run the code – it should stop with an error. Use the browser() command or the R option options(error=recover) or the debug() function to debug the function and find the two mistakes in it. The mistakes may include code that is incorrect, code that needs to be removed or code that is missing and needs to be added.

If the 'Stop' button doesn't close browser mode, go to the console and use the Esc key to escape from browser mode.

For each of the two, either change the code to fix the bug or write a comment in your RMarkdown explaining where you think the bug is. Each mistake is worth 1 marks for finding it and 1 mark for fixing it, and partial credit will be given for incomplete answers. **ANS**

```
library(tidyr)
library(dplyr)
food_prices <- read.csv("food_prices_yearmonth.csv", stringsAsFactors = FALSE)

get_food_summary <- function(food_prices,
                           latest_year = max(food_prices$Year)) {
    food_prices <- filter(food_prices, Year <= latest_year)

    fruit <- c("Oranges","Bananas","Apples","Kiwifruit",
               "Sultanas (supermarket only)",
               "Peaches - canned (supermarket only)",
               "Apricots", "Grapes","Mandarins","Berries","Pineapple")
```

```r
    vegetables <- c("Lettuce","Broccoli","Cabbage","Tomatoes",
                    "Carrots","Mushrooms","Potatoes",
                    "Peas - frozen (supermarket only)",
                    "Avocado","Beans","Capsicums","Cauliflower",
                    "Celery","Courgettes","Cucumber","Kumara",
                    "Mixed vegetables","Pnions","Parsnips",
                    "Pears","Pumpkin")

    food_prices$Fruit_veg <- rep(FALSE, nrow(food_prices))                          ## change NA to FALSE
    food_prices$Liquid <- rep(FALSE, nrow(food_prices))

    item_parts <- strsplit(food_prices$Item, split = ", ")

    for (i in 1:nrow(food_prices)) {
        this_item_parts <- item_parts[[i]]
        item_type <- this_item_parts[1]
        item_units <- this_item_parts[length(this_item_parts)]
        if (item_type %in% fruit) {
            food_prices$Fruit_veg[i] <- "Fruit"
        } else if (item_type %in% vegetables) {
            food_prices$Fruit_veg[i] <- "Vegetables"
        } else  food_prices$Fruit_veg[i] <- "Other"                                 ## add else

        if (food_prices$Fruit_veg[i] == "Other") {
            if (length(grep("ml",item_units)) != 0 |
                length(grep("litres",item_units)) != 0) {
                    food_prices$Liquid <- TRUE                                      ## Capital TRUE
            }

        }
    }

    fruit_veg_summary <- food_prices %>%
        group_by(Fruit_veg, Year, Month) %>%
        filter(Fruit_veg == "Fruit"||Fruit_veg == "Vegetables") %>%                 ## add a filter
        summarise(Max_price = max(Data_value))

    liquids_summary <- food_prices %>%
        ungroup() %>%
        filter(Liquid == TRUE) %>%
        group_by(Liquid, Year, Month) %>%
        summarise(Median_Price = median(Data_value))

    list(fruit_veg_summary = fruit_veg_summary,
         liquids_summary = liquids_summary)
}

food_summary <- get_food_summary(food_prices, latest_year = 2015)
```

```
## `summarise()` regrouping output by 'Fruit_veg', 'Year' (override with `.groups` argument)

## `summarise()` regrouping output by 'Liquid', 'Year' (override with `.groups` argument)
```

```r
head(food_summary$fruit_veg_summary)
```

```
## # A tibble: 6 x 4
## # Groups:   Fruit_veg, Year [1]
##   Fruit_veg  Year Month    Max_price
##   <chr>     <int> <chr>        <dbl>
## 1 Fruit      2006 August        2.43
## 2 Fruit      2006 December      3.29
## 3 Fruit      2006 July          2.78
## 4 Fruit      2006 June          3.11
## 5 Fruit      2006 November      3.24
## 6 Fruit      2006 October       3.04
```

```r
head(food_summary$liquids_summary)
```

```
## # A tibble: 6 x 4
## # Groups:   Liquid, Year [1]
##   Liquid  Year Month    Median_Price
##   <lgl>  <int> <chr>           <dbl>
## 1 TRUE    2006 August           2.55
## 2 TRUE    2006 December         2.88
## 3 TRUE    2006 July             2.73
## 4 TRUE    2006 June             2.86
## 5 TRUE    2006 November         2.68
## 6 TRUE    2006 October          2.64
```

#Q5. (22 Marks) #In this question, we will simulate the drawing of random cards from a standard 52-card deck of playing cards. A standard 52-card deck of playing cards consist of 13 cards (which we will consider as being numbered from 1 to 13, so 11 = jack, 12 = queen, 13 = king) for four suits/types (clubs, diamonds, hearts, spades).

a. [2 marks] For a single random draw from the deck, what is the probability of selecting a card that is a club? What is the probability of selecting a card that is 7 or higher? **ANS**

```r
n <- 13/52
cat("The probability of selecting a card that is a club is " , n, "\n")
```

```
## The probability of selecting a card that is a club is  0.25
```

```r
n2 <- (7 * 4) / 52
cat("The probability of selecting a card that is 7 or highe is " , n2)
```

```
## The probability of selecting a card that is 7 or highe is  0.5384615
```

b. [8 marks] Write a custom function called card.draw to simulate the random drawing of cards with replacement from a standard 52-card deck of playing cards. **ANS**

```r
# 52-card deck
deck <- data.frame(
  Number <- rep(1:13,4),
  Suit <- c(rep("C",13), rep("D",13),rep("H",13),rep("S",13))
)
colnames(deck) <- c("Number", "Suit")
```

```r
card.draw<- function(n , seed){

################### check n  ###################
  if(!is.numeric(n))
  {
    stop("'n' must be an integer.")
  }else if(length(n) > 1)
   {
    stop("'n' consists of more than one element.")
   }
    # Check to see if n is 0 or negative or not a whole number.
    else if((n <= 0) | (ceiling(n) != n))
    {
      stop("'n' must be a positive integer (i.e., whole number).")
   }
################### check seed  ###################
 if(!is.numeric(seed))
  {
    stop("'seed' must be an integer.")
  }else if(length(seed) > 1)
   {
    stop("'seed' consists of more than one element.")
   }
    # Check to see if seed is 0 or negative or not a whole number.
    else if((n < 0) | (ceiling(seed) != seed))
    {
    stop("'seed' must be a positive integer (i.e., whole number).")
   }
################### check end  ###################
  set.seed(seed)
  hand <- sample( 52 , n , replace= TRUE)
  num<-list()
  suit<-list()
  for (i in 1:length(hand)) {
    num[[i]] <- deck$Number[hand[i]]
    suit[[i]] <- deck$Suit[hand[i]]
    #each <- data.frame(num,suit)
    #drawn <- rbind(drawn, each)
  }

  drawn <- do.call(rbind, Map(data.frame,Number = num,Suit=suit))


  return(drawn)
}
```

c.[2 marks] Show output for your code when it is run for the following function specifications:

```r
card.draw(n = 3.7, seed = 2) # Q1(c)i.
```

```
## Error in card.draw(n = 3.7, seed = 2): 'n' must be a positive integer (i.e., whole number).
```

```r
card.draw(n = 3, seed = 'a') # Q1(c)ii.
```

```
## Error in card.draw(n = 3, seed = "a"): 'seed' must be an integer.
```

```r
card.draw(n = c(3, 2), seed = c(1, 2)) # Q1(c)iii.
```

```
## Error in card.draw(n = c(3, 2), seed = c(1, 2)): 'n' consists of more than one element.
```

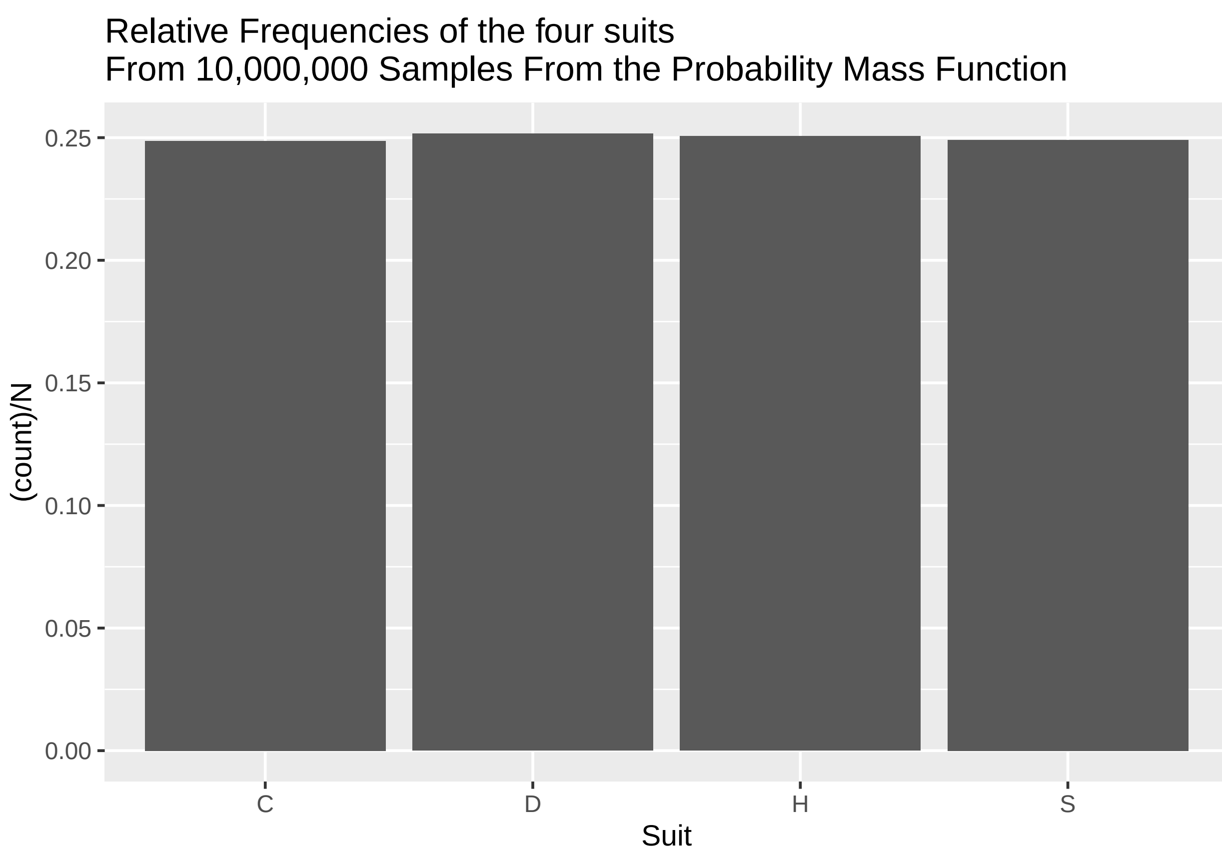
```r
card.draw(n = 4, seed = 0.3) # Q1(c)iv.
```

```
## Error in card.draw(n = 4, seed = 0.3): 'seed' must be a positive integer (i.e., whole number).
```

d. [4 marks] Use your function to simulate 1,000,000 random card draws, and use ggplot to produce appropriate graphical displays to show relative frequencies (i.e. the distribution) of the outcomes of

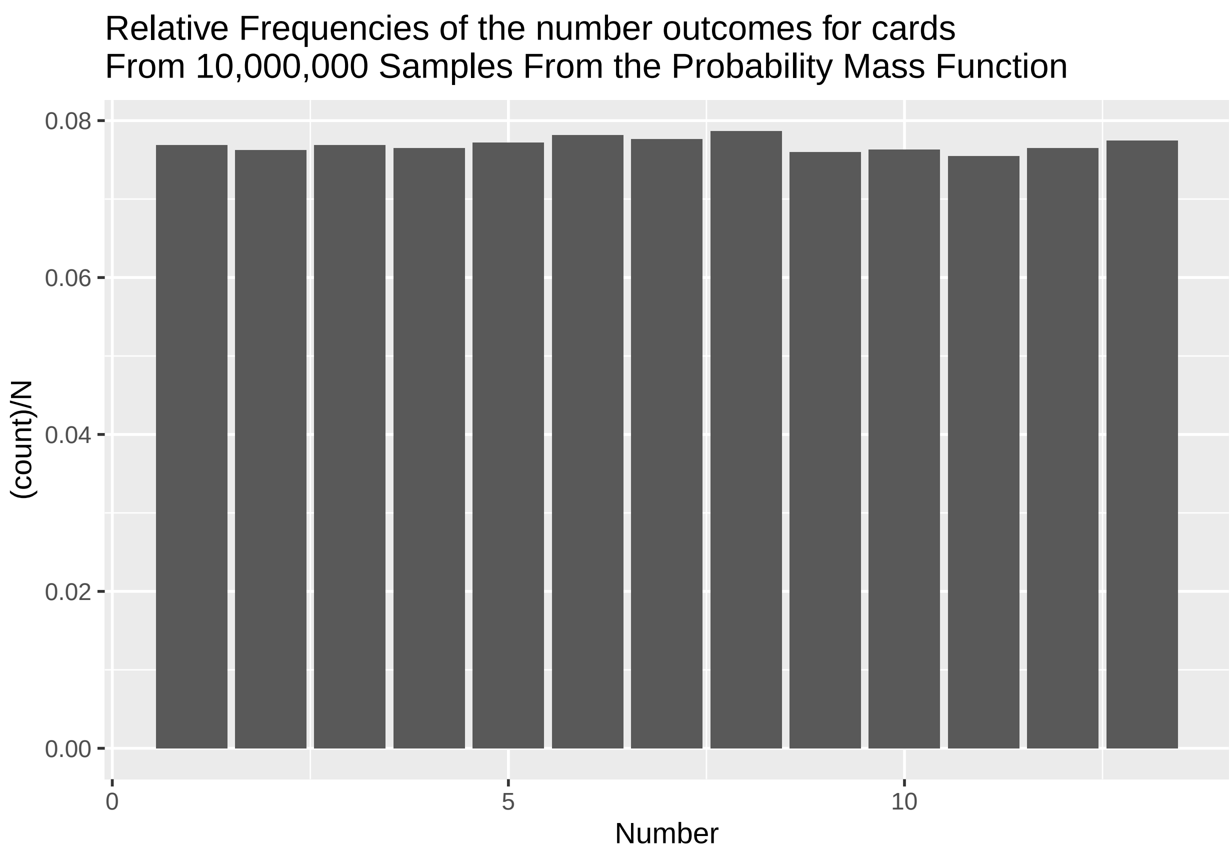
e. the four suits **ANS**

```r
N<-100000
outcomes <- card.draw(n=N,seed=0)
ggplot(as.data.frame(outcomes)) + geom_bar(aes(x=Suit, y = (..count..)/N))+ labs(title = "Relative Frequ
```



ii. the number outcomes for cards. **ANS**

```
ggplot(as.data.frame(outcomes)) + geom_bar(aes(x=Number, y = (..count..)/N))+ labs(title = "Relative Fre
```



e. [2 marks] Use your function to simulate 1,000,000 random card draws, and calculate the proportion of the cards drawn that are i. clubs **ANS**

```
n <- count(outcomes[outcomes$Suit =="C",])
n/N
```

```
##         n
## 1 0.24869
```

ii. 7 or higher **ANS**

```
n <- count(outcomes[outcomes$Number >=7,])
n/N
```

```
##        n
## 1 0.5381
```

f. [4 marks] Finally, calculate the exact expected value and variance for the number outcome for a randomly drawn card from the deck. Use a simulation of 1,000,000 random card draws to estimate the expected value and variance to verify the exact values you calculated.