Week-1 HW Stat-560 QN-1

Sagar

2023-07-08

```
options(repos = "https://cran.rstudio.com/")
install.packages("tinytex")

## Installing package into 'C:/Users/Dell/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'tinytex' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Dell\AppData\Local\Temp\RtmpmyAPbv\downloaded_packages

library(tinytex)

## Warning: package 'tinytex' was built under R version 4.3.1
```

Week-1 HomeWork STAT 560

Question no 1

Packages in R make the computation and calculation much easier, so we will first install the packages and load the in the working environment. To install packages we can use the install.packages function and to load the package in the environment we can use the library function.

```
install.packages("readxl")

## Installing package into 'C:/Users/Dell/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

##
##   There is a binary version available but the source version is later:
##        binary source needs_compilation
## readxl  1.4.2  1.4.3              TRUE

## installing the source package 'readxl'

## Warning in install.packages("readxl"): installation of package 'readxl'
had
## non-zero exit status
```

```r
library(readxl)

install.packages("tidyverse")

## Installing package into 'C:/Users/Dell/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Dell\AppData\Local\Temp\RtmpmyAPbv\downloaded_packages

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.1

## Warning: package 'ggplot2' was built under R version 4.3.1

## Warning: package 'lubridate' was built under R version 4.3.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Data loading

(a) Download the data table and read the data into R. Take a look at the data in the data viewer. ANS:- we can load our data in the R working environment by a lot of different ways. Using read_excel function is one of them to load xlsx dataset into the R working environment, this function is the part of readxl library.

Viewing dataset

Once the dataset is loaded in our working environment, we can view our dataset using view function (R is case sensitive).

```r
View(TexasHousing)
```

Getting Familier with our data set

Before start working in our dataset, we need to know about our dataset and their data type so we can use str() function to get more familier with our data set.

```
str(TexasHousing)

## tibble [8,602 × 10] (S3: tbl_df/tbl/data.frame)
##  $ ...1     : num [1:8602] 1 2 3 4 5 6 7 8 9 10 ...
##  $ city     : chr [1:8602] "Abilene" "Abilene" "Abilene" "Abilene" ...
##  $ year     : num [1:8602] 2000 2000 2000 2000 2000 2000 2000 2000 2000
2000 ...
##  $ month    : num [1:8602] 1 2 3 4 5 6 7 8 9 10 ...
##  $ sales    : num [1:8602] 72 98 130 98 141 156 152 131 104 101 ...
##  $ volume   : num [1:8602] 5380000 6505000 9285000 9730000 10590000 ...
##  $ median   : num [1:8602] 71400 58700 58100 68600 67300 66900 73500 75000
64500 59300 ...
##  $ listings : chr [1:8602] "701" "746" "784" "785" ...
##  $ inventory: chr [1:8602] "6.3" "6.6" "6.8" "6.9" ...
##  $ date     : num [1:8602] 2000 2000 2000 2000 2000 ...
```

Creating new dataframe

(b) Write a code chunk to remove the inventory variable. Save the results in a data frame called txhousing. ANS:- we can create new dataset from given dataset by removing one column, for that we will use the select function which is the part of tidyverse library.

```
txhousing = select(TexasHousing, -inventory)
```

Creating a filtered dataframe

(c) Make a data set called dallas sub" that includes data only from the city of Dallas in 2012 and 2013. we can make a filtered dataset from the given dataset using filter function. If we use pipe whlie using filter function it make code more readable and easy to follow and easy to code. A pipe in R is denoted by %>% (ctrl + shift + M)

```
dallas_sub = txhousing %>% filter(city=='Dallas') %>% filter(year==2012 |
year==2013)
```

Checking work

Once we created a filtered dataset named dallas_sub. we can confirm our work by looking how many unique city the the dallas_sub dataframe has and how many unique year data does the dallas_sub dataset has. For this we are going to use a unique function.

```
unique(dallas_sub$year)

## [1] 2012 2013

unique(dallas_sub$city)

## [1] "Dallas"
```

Adding new collumn

(d) Add a column to the dallas sub data set called “prct sold” that calculates the percentage of listings that were sold (sales/listings × 100). ANS:- In order to do calculation with sales and listings, they both should have to be numeric. But the listings is a char type so we first need to change its type to numeric

```
dallas_sub=transform(dallas_sub, listings=as.numeric(listings))
```

And then we can usse mutate function to add new column to the dataset(obtained by some calculations with old column)

```
dallas_sub = mutate(dallas_sub, prct_sold=sales/listings *100)
head(dallas_sub)

##   ...1   city year month sales     volume median listings     date
prct_sold
## 1 2202 Dallas 2012     1  2555  509458081 150800    16721 2012.000
15.28019
## 2 2203 Dallas 2012     2  3085  634067291 157100    17173 2012.083
17.96425
## 3 2204 Dallas 2012     3  4068  898320563 167300    17433 2012.167
23.33505
## 4 2205 Dallas 2012     4  4291  983333297 168700    17632 2012.250
24.33643
## 5 2206 Dallas 2012     5  5004 1175419749 175100    17726 2012.333
28.22972
## 6 2207 Dallas 2012     6  5196 1209024869 177900    17587 2012.417
29.54455
```

(e) Calculate the average percentage of listings that were sold in Dallas in each month of the year based on your dallas_sub data set. Save the results of the calculation in a data frame called dallas summary ANS:- We need average percentage of listing over each month for that we will group our dataset by month and for average we will use mean function in summary and name the whole new dataset as dallas_summary.

```
dallas_summary = dallas_sub %>% group_by(month) %>%
summarise(Avg_per_listing=mean(prct_sold))
head(dallas_summary)

## # A tibble: 6 × 2
##   month Avg_per_listing
##   <dbl>           <dbl>
## 1     1            20.5
## 2     2            23.5
## 3     3            32.2
## 4     4            34.5
## 5     5            38.2
## 6     6            37.2
```

(f) Arrange dallas_summary in descending order based on the average percentage of listings. Which month had the greatest average percentage of listings sold? ANS:- To arrange dallas_summary subset in the descending order we will use -ve sign in front of the Avg_per_listing column. Clearly from the output we can see that the 8th month (August) has the greatest average percentage of listings sold.

```
dallas_summary = dallas_sub %>% group_by(month) %>%
summarise(Avg_per_listing=mean(prct_sold)) %>% arrange(-Avg_per_listing)
head(dallas_summary)

## # A tibble: 6 × 2
##   month Avg_per_listing
##   <dbl>           <dbl>
## 1     8            38.5
## 2     5            38.2
## 3     6            37.2
## 4     7            37.1
## 5    12            35.5
## 6     4            34.5
```

(g) In January of 2015, what city had the fewest houses listed for sale? ANS:- Waco and the number of house listed for sale are 1034.

```
r filletered_df=txhousing   %>%  filter(year==2015) %>% filter(month==1)
filter(filletered_df,listings==min(filletered_df$listings))
```

```
## # A tibble: 1 × 9 ##    ...1 city   year month sales   volume median
listings  date ##   <dbl> <chr> <dbl> <dbl> <dbl>    <dbl>  <dbl> <chr>
<dbl> ## 1  8409 Waco   2015     1   144 22645440 137500 1034      2015
```

(h) Generate a single table that shows the total number of houses sold in Austin in 2000 and 2001 (total over the entire period), and the total number of houses sold in Dallas in 2000 and 2001 (total over the entire period). ANS:- Austin 37013 and Dallas 92438

```
txhousing_1= txhousing %>%  group_by(city) %>% filter(city=='Austin' |
city=='Dallas') %>%  filter(year==2000 | year==2001) %>%
summarise(Total=sum(sales))
View(txhousing_1)
```

THE END

week-1 HW Stat-560 Qno-2

Sagar

2023-07-08

```
options(repos = "https://cran.rstudio.com/")
```

Installing and loading the tidyverse package in our working environmnet.

```
install.packages("tidyverse")

## Installing package into 'C:/Users/Dell/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Dell\AppData\Local\Temp\RtmpSEihp8\downloaded_packages

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.1

## Warning: package 'ggplot2' was built under R version 4.3.1

## Warning: package 'lubridate' was built under R version 4.3.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Installing and loading the nycflights13 package in our working environment.

```
install.packages("nycflights13")

## Installing package into 'C:/Users/Dell/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'nycflights13' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Dell\AppData\Local\Temp\RtmpSEihp8\downloaded_packages
```

```
library(nycflights13)
```

```
## Warning: package 'nycflights13' was built under R version 4.3.1
```

From nycflights13 package, assigning flights and planes dataset as flights_data and planes_data.

```
flights_data = flights
planes_data = planes
```

Looking at our data sets. [we can insert a code chunk by using shortcut ctrl+shift+I]

```
head(flights_data)     # head() gives first few rows of our data set.
```

```
## # A tibble: 6 × 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
## 1  2013     1     1      517            515         2      830
819
## 2  2013     1     1      533            529         4      850
830
## 3  2013     1     1      542            540         2      923
850
## 4  2013     1     1      544            545        -1     1004
1022
## 5  2013     1     1      554            600        -6      812
837
## 6  2013     1     1      554            558        -4      740
728
## # ℹ 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance
<dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
head(planes_data)
```

```
## # A tibble: 6 × 9
##   tailnum  year type               manufacturer model engines seats speed
engine
##   <chr>   <int> <chr>              <chr>        <chr>   <int> <int> <int>
<chr>
## 1 N10156   2004 Fixed wing multi … EMBRAER      EMB-…       2    55    NA
Turbo…
## 2 N102UW   1998 Fixed wing multi … AIRBUS INDU… A320…       2   182    NA
Turbo…
```

```
## 3 N103US   1999 Fixed wing multi … AIRBUS INDU… A320…       2   182    NA
Turbo…
## 4 N104UW   1999 Fixed wing multi … AIRBUS INDU… A320…       2   182    NA
Turbo…
## 5 N10575   2002 Fixed wing multi … EMBRAER      EMB-…       2    55    NA
Turbo…
## 6 N105UW   1999 Fixed wing multi … AIRBUS INDU… A320…       2   182    NA
Turbo…
```

```
tail(flights_data)     # tail() gives last few rows of our data set.
```

```
## # A tibble: 6 × 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
## 1  2013     9    30       NA           1842        NA       NA
2019
## 2  2013     9    30       NA           1455        NA       NA
1634
## 3  2013     9    30       NA           2200        NA       NA
2312
## 4  2013     9    30       NA           1210        NA       NA
1330
## 5  2013     9    30       NA           1159        NA       NA
1344
## 6  2013     9    30       NA            840        NA       NA
1020
## # ℹ 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance
<dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
tail(planes_data)
```

```
## # A tibble: 6 × 9
##   tailnum  year type               manufacturer model engines seats speed
engine
##   <chr>   <int> <chr>              <chr>        <chr>   <int> <int> <int>
<chr>
## 1 N996DL   1991 Fixed wing multi … MCDONNELL D… MD-88       2   142    NA
Turbo…
## 2 N997AT   2002 Fixed wing multi … BOEING       717-…       2   100    NA
Turbo…
## 3 N997DL   1992 Fixed wing multi … MCDONNELL D… MD-88       2   142    NA
Turbo…
## 4 N998AT   2002 Fixed wing multi … BOEING       717-…       2   100    NA
Turbo…
## 5 N998DL   1992 Fixed wing multi … MCDONNELL D… MD-88       2   142    NA
Turbo…
```

```
## 6 N999DN   1992 Fixed wing multi … MCDONNELL D… MD-88       2   142    NA
Turbo…
```

Getting familier with our data set

```
str(planes_data)
```

```
## tibble [3,322 × 9] (S3: tbl_df/tbl/data.frame)
##  $ tailnum     : chr [1:3322] "N10156" "N102UW" "N103US" "N104UW" ...
##  $ year        : int [1:3322] 2004 1998 1999 1999 2002 1999 1999 1999 1999
1999 ...
##  $ type        : chr [1:3322] "Fixed wing multi engine" "Fixed wing multi
engine" "Fixed wing multi engine" "Fixed wing multi engine" ...
##  $ manufacturer: chr [1:3322] "EMBRAER" "AIRBUS INDUSTRIE" "AIRBUS
INDUSTRIE" "AIRBUS INDUSTRIE" ...
##  $ model       : chr [1:3322] "EMB-145XR" "A320-214" "A320-214" "A320-214"
...
##  $ engines     : int [1:3322] 2 2 2 2 2 2 2 2 2 2 ...
##  $ seats       : int [1:3322] 55 182 182 182 55 182 182 182 182 182 ...
##  $ speed       : int [1:3322] NA NA NA NA NA NA NA NA NA NA ...
##  $ engine      : chr [1:3322] "Turbo-fan" "Turbo-fan" "Turbo-fan" "Turbo-
fan" ...
```

```
str(flights_data)
```

```
## tibble [336,776 × 19] (S3: tbl_df/tbl/data.frame)
##  $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013
2013 2013 ...
##  $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558
...
##  $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600
...
##  $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753
...
##  $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745
...
##  $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
##  $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79
301 ...
##  $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138
...
##  $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
##  $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
```

```
## $ time_hour      : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

(a) What month had the highest proportion of cancelled flights? What month had the lowest? ANS:- The heighest number of flights were cancled on the 2nd month(february) and the lowest number of flights were cancled on the month of november i.e 11th month

```
flights_data %>%  filter(is.na(dep_time) & is.na(arr_time)) %>%
group_by(month) %>%  summarise(Num_of_count=n())

## # A tibble: 12 × 2
##    month Num_of_count
##    <int>        <int>
##  1     1          521
##  2     2         1261
##  3     3          861
##  4     4          668
##  5     5          563
##  6     6         1009
##  7     7          940
##  8     8          486
##  9     9          452
## 10    10          236
## 11    11          233
## 12    12         1025
```

(b) How many planes have a missing date of manufacture? ANS:- So there were 70 planes which were missing their date of manufacture from the dataset.

```
planes_data %>% filter(is.na(year)) %>% summarise(Num_of_planes=n())

## # A tibble: 1 × 1
##   Num_of_planes
##           <int>
## 1            70
```

(c) What are the five most common manufacturers? ANS:- The five most common manufacturer were:- BOEING 1630
AIRBUS INDUSTRIE 400
BOMBARDIER INC 368
AIRBUS 336
EMBRAER 299

```
planes_data %>% group_by(manufacturer) %>% summarise(Number=n()) %>%
arrange(-Number)

## # A tibble: 35 × 2
##    manufacturer                  Number
##    <chr>                          <int>
##  1 BOEING                          1630
##  2 AIRBUS INDUSTRIE                 400
##  3 BOMBARDIER INC                   368
```

```
##   4 AIRBUS                          336
##   5 EMBRAER                         299
##   6 MCDONNELL DOUGLAS               120
##   7 MCDONNELL DOUGLAS AIRCRAFT CO   103
##   8 MCDONNELL DOUGLAS CORPORATION    14
##   9 CANADAIR                          9
##  10 CESSNA                            9
## # ℹ 25 more rows
```

Clearly, there were 35 total manufacture's, so cross-checking our work

```
unique(planes_data$manufacturer)
```

```
##  [1] "EMBRAER"                       "AIRBUS INDUSTRIE"
##  [3] "BOEING"                        "AIRBUS"
##  [5] "BOMBARDIER INC"                "CESSNA"
##  [7] "JOHN G HESS"                   "GULFSTREAM AEROSPACE"
##  [9] "SIKORSKY"                      "PIPER"
## [11] "AGUSTA SPA"                    "PAIR MIKE E"
## [13] "DOUGLAS"                       "BEECH"
## [15] "BELL"                          "AVIAT AIRCRAFT INC"
## [17] "STEWART MACO"                  "LEARJET INC"
## [19] "MCDONNELL DOUGLAS"             "CIRRUS DESIGN CORP"
## [21] "HURLEY JAMES LARRY"            "KILDALL GARY"
## [23] "LAMBERT RICHARD"               "BARKER JACK L"
## [25] "AMERICAN AIRCRAFT INC"         "ROBINSON HELICOPTER CO"
## [27] "FRIEDEMANN JON"                "LEBLANC GLENN T"
## [29] "MARZ BARRY"                    "DEHAVILLAND"
## [31] "CANADAIR"                      "CANADAIR LTD"
## [33] "MCDONNELL DOUGLAS CORPORATION" "MCDONNELL DOUGLAS AIRCRAFT CO"
## [35] "AVIONS MARCEL DASSAULT"
```

(d) What is the oldest plane (specified by the tailnum variable) that flew from New York City airports in 2013? ANS:- To get the oldest plane specified by the tailnum that flew from New york city airports in 2013, new need to look at both tables simultaniously. One of the method will be joining both table together by primary key tailnum. the other way will be fillter a table based on the condition that look for the data of tailnum in both the tables.

My final answer is plane with tail num= N381AA and which was manufactured in 1956.

```
filtered_planes_data= planes_data %>% filter(!is.na(year))
filtered_planes_data %>% summarise(tailnum, oldest_plane=min(year))
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was
deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that
`reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
```

```
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 3,252 × 2
##    tailnum oldest_plane
##    <chr>          <int>
##  1 N10156          1956
##  2 N102UW          1956
##  3 N103US          1956
##  4 N104UW          1956
##  5 N10575          1956
##  6 N105UW          1956
##  7 N107US          1956
##  8 N108UW          1956
##  9 N109UW          1956
## 10 N110UW          1956
## # ℹ 3,242 more rows
```

```
oldest_planes = filtered_planes_data %>% filter(filtered_planes_data$tailnum
%in% flights_data$tailnum)
oldest_plane= oldest_planes %>% arrange(year)
oldest_plane_by_tailnum= oldest_plane[1, ]
print(oldest_plane_by_tailnum)
```

```
## # A tibble: 1 × 9
##   tailnum  year type               manufacturer model engines seats speed
engine
##   <chr>   <int> <chr>              <chr>        <chr>   <int> <int> <int>
<chr>
## 1 N381AA   1956 Fixed wing multi … DOUGLAS      DC-7…       4   102   232
Recip…
```

Once your work is done try remove unnecessary dataframe from your environment.

```
rm(filtered_planes_data)
rm(oldest_planes)
rm(oldest_plane)
rm(oldest_plane_by_tailnum)
```

(e) What plane (specified by the tailnum variable) traveled the most times from New York City airports in 2013? ANS:- So the plane with tailnum= N725MQ flew the most number of time i.e 575

```
x=flights_data %>% filter(!is.na(tailnum)) %>%  group_by(tailnum) %>%
reframe(tailnum, No_of_count=n()) %>% arrange(-No_of_count)
Highest_flewed_plane=x[1,]
print(Highest_flewed_plane)
```

```
## # A tibble: 1 × 2
##   tailnum No_of_count
##   <chr>         <int>
## 1 N725MQ          575
```

Once your work is done try remove unnecessary dataframe from your environment

```
rm(x)
rm(Highest_flewed_plane)
```

(f) How many airplanes that flew from New York City are included in the planes table?
ANS:- There were 3322 number of planes that were included in the planes table which flew from the New york city

```
counts=planes_data %>% filter(planes_data$tailnum %in% flights_data$tailnum)
# filtering planes_data data set
count_1= counts %>% filter(!is.na(tailnum)) %>% reframe(tailnum,
Num_of_airplanes=n())
print(count_1[1,])                                      # code to look the
first row of our data frame
```

```
## # A tibble: 1 × 2
##   tailnum Num_of_airplanes
##   <chr>              <int>
## 1 N10156              3322
```

Once our work is done try to remove unnecessary dataframe from the environment

```
rm(count_1)
rm(counts)
```

The End

Week-2_HW_Stat_560_R_part

Sagar Kalauni

2023-07-15

```
options(repos = "https://cran.rstudio.com/")
```

R Practise

1. Consider the diamond data table in ggplot2 package. The data is loaded automatically when you load the ggplot2 or tidyverse library. The dataset containing the prices and other attributes of almost 54,000 diamonds. You may see the descriptions of the variables using command ?diamonds in R. Attach the plots generated for each question.

Installing the necessary packages and loading the the packages

```
install.packages("tidyverse")
```

```
## Installing package into
'C:/Users/Dell/AppData/Local/R/win-library/4.3' ## (as 'lib' is
unspecified)

## package 'tidyverse' successfully unpacked and MD5 sums
checked ##
## The downloaded binary packages are in
## C:\Users\Dell\AppData\Local\Temp\Rtmpm8Cpxe\downloaded_packages
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version

4.3.1 ## Warning: package 'ggplot2' was built under R version

4.3.1 ## Warning: package 'lubridate' was built under R

version 4.3.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr 1.1.2 ✓ readr 2.1.4
## ✓ forcats 1.0.0 ✓ stringr 1.5.0
## ✓ ggplot2 3.4.2 ✓ tibble 3.2.1
## ✓ lubridate 1.9.2 ✓ tidyr 1.3.0
## ✓ purrr 1.0.1

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag() masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Loading my diamond data set into the working environment.

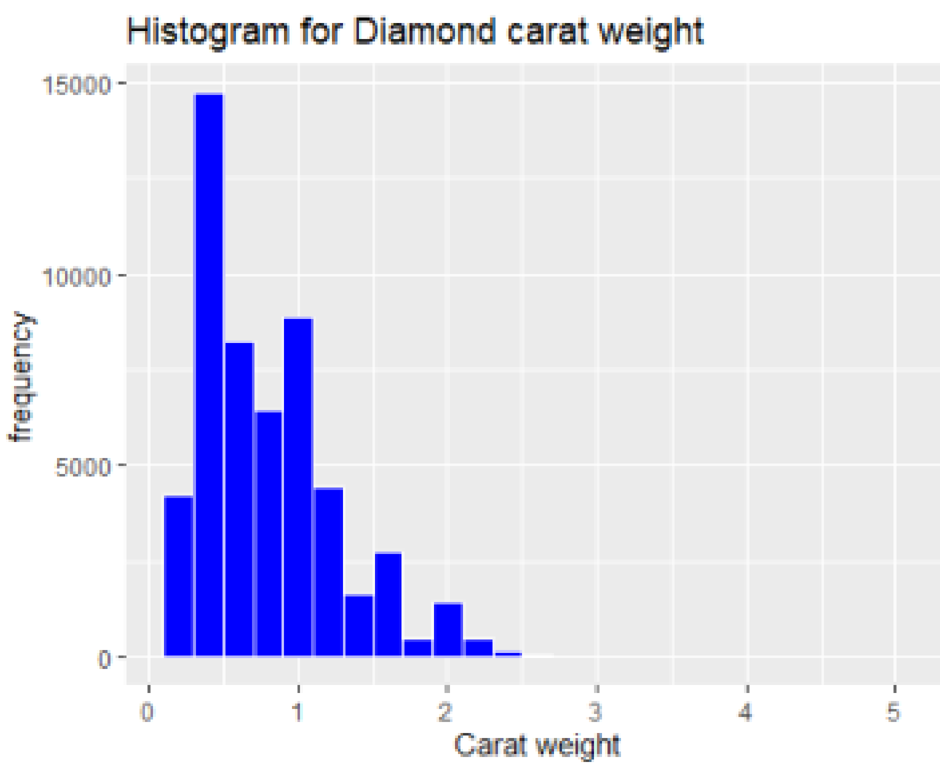
```
diamond_data= diamonds
```

(a) Construct a histogram for the caret weights of the diamonds. Describe the shape of the histogram.

ANSWER:-

The Shape of the histogram looks right skewed.

After examining the carat data in the diamond dataset, I have concluded that using a bin width of 0.2 would yield better results. Considering that the color blue is easily noticeable, I have opted to fill the histogram bars with blue. Furthermore, since the background is white, I have chosen white as the color (border) for the histogram bars to ensure clear distinction between the classes. These modifications have been implemented to ensure that the histogram classes can be easily distinguished and identified.



(b) Construct a side-by-side barplot of the color grade of the diamonds separated by different grades of cut. Describe what you observe.

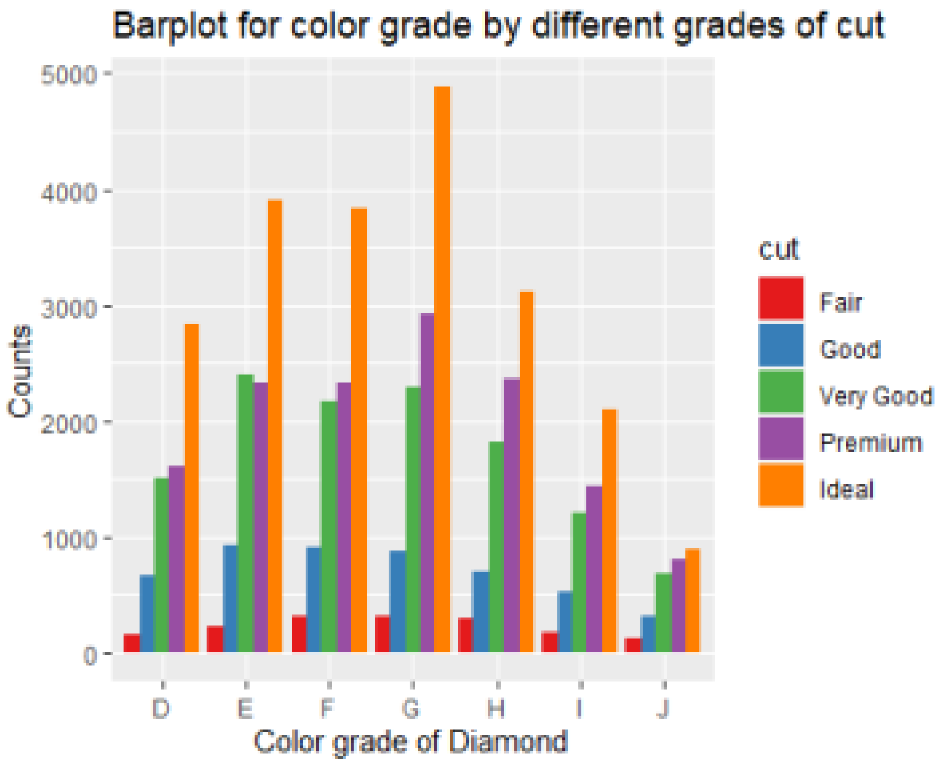
ANSWER:-

Based on the graph, it is evident that the ideal cut is most prevalent across all colors, with color G having the highest number of diamonds with an ideal cut and color J having the
lowest. Similarly, the Fair cut is consistently lower in color J compared to other colors. Overall, the Fair cut tends to be less common across all colors.

here position='dodge' has put side-by-side.

```
figure_barplot = ggplot(data = diamond_data, aes(x= color, fill=cut)) +
geom_bar(position = 'dodge') + labs(x='Color grade of Diamond', y='Counts',
```

```
title='Barplot for color grade by different grades of cut')+
scale_fill_brewer(palette='Set1')
figure_barplot
```

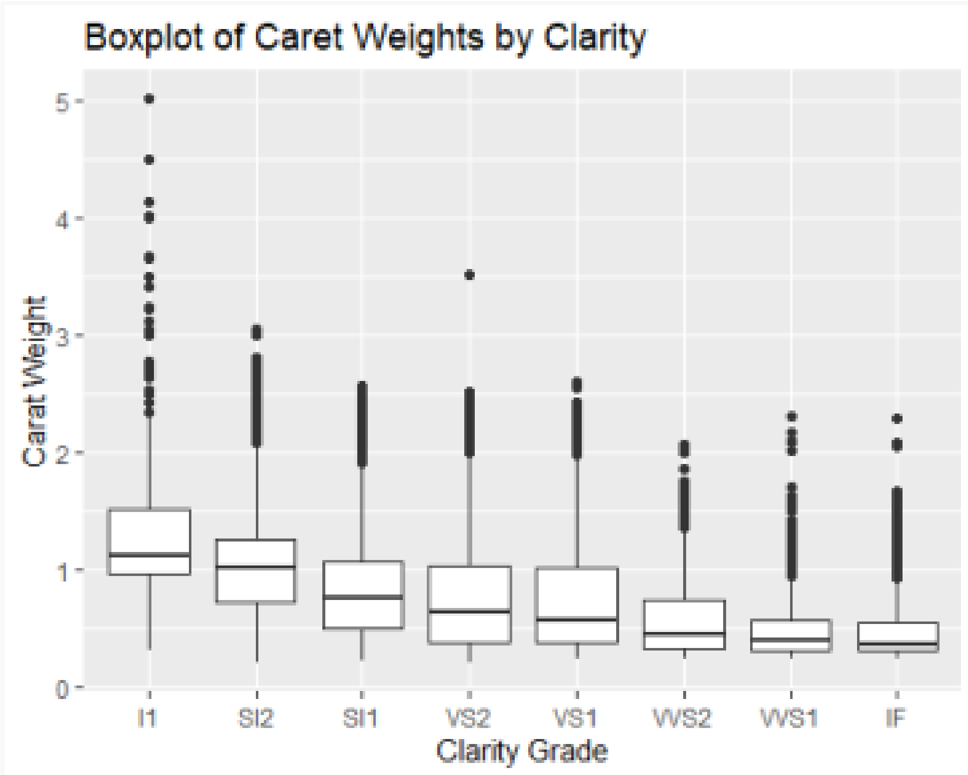


(c) Construct side-by-side boxplot for the caret weights of the diamonds by different grades of clarity. Describe what you observe.

ANSWER:-

Diamonds with clarity grade IF exhibit the lowest variability in their carat weight, and their mean carat weight is also the lowest among the means of carat weight for different clarity grades. Additionally, there is a noticeable high level of variance in carat weight for clarity grades VS1 and VS2.

```
figure_baxplot_new = ggplot(data = diamond_data, aes(x=clarity, y= carat)) +
geom_boxplot() + labs(x = "Clarity Grade", y = "Carat Weight", title='Boxplot
of Caret Weights by Clarity')
figure_baxplot_new
```



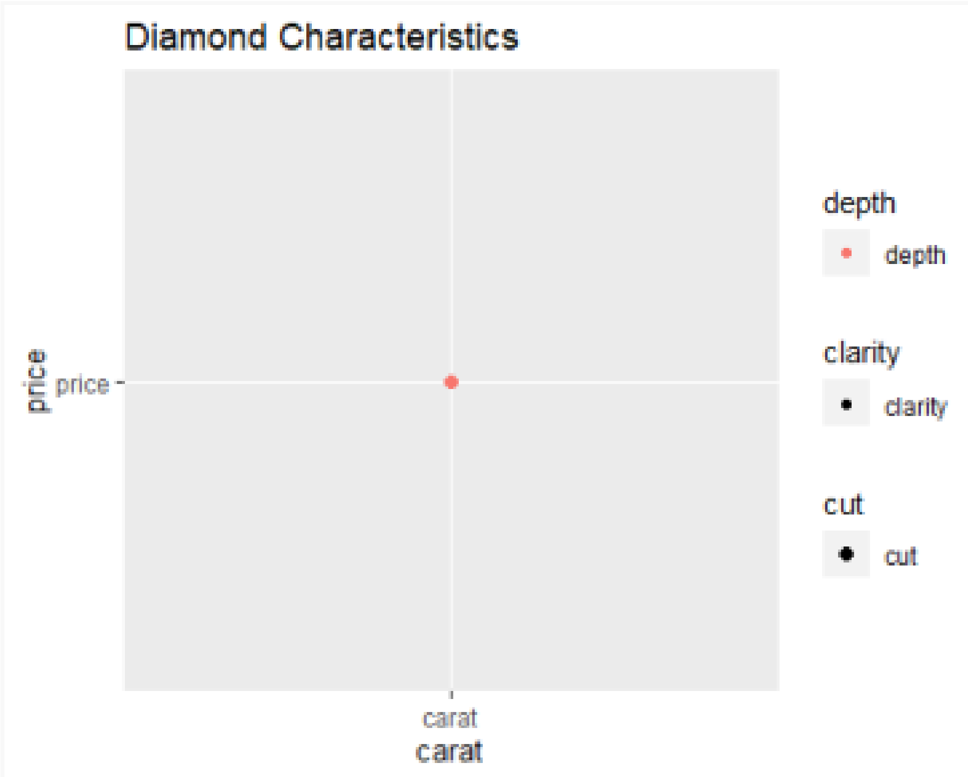
(d) Create a data graphic with at least five variables (either quantitative or categorical). For the purposes of this exercise, do not worry about making your visualization meaningful—just try to encode five variables into one plot.

ANSWER:-

The five variables used in my data graphs are:- Carat, Price, Depth, Clarity and cut.

```
figure_scattered= ggplot(data=diamond_data, aes(x='carat', y='price', color='depth', shape='clarity', size='cut'))+ geom_point() + labs(x='carat', y='price', color='depth', shape='clarity', size='cut', title = "Diamond Characteristics")
figure_scattered
```

```
## Warning: Using size for a discrete variable is not advised.
```



Week-2_HW_Stat_560_R_part_QN-3

Sagar Kalauni

2023-07-15

```
options(repos = "https://cran.rstudio.com/")
```

Installing library tidyverse

```
install.packages("tidyverse")
```

```
## Installing package into
'C:/Users/Dell/AppData/Local/R/win-library/4.3' ## (as 'lib' is
unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums
checked ##
## The downloaded binary packages are in
## C:\Users\Dell\AppData\Local\Temp\Rtmpqwg2tI\downloaded_packages
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version

4.3.1 ## Warning: package 'ggplot2' was built under R version

4.3.1 ## Warning: package 'lubridate' was built under R

version 4.3.1
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
```

```
## ✓ dplyr 1.1.2 ✓ readr 2.1.4
## ✓ forcats 1.0.0 ✓ stringr 1.5.0
## ✓ ggplot2 3.4.2 ✓ tibble 3.2.1
## ✓ lubridate 1.9.2 ✓ tidyr 1.3.0
## ✓ purrr 1.0.1
```

```
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag() masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

2. Using babynames data table from the babynames package to answer the following questions. You will need to install the package first and load the library. Attach the plots generated for each question.

Installing and loading the babynames packages.

```
install.packages("babynames")
```

```
## Installing package into
'C:/Users/Dell/AppData/Local/R/win-library/4.3' ## (as 'lib' is
unspecified)
```

```
## package 'babynames' successfully unpacked and MD5 sums
checked ##
## The downloaded binary packages are in
## C:\Users\Dell\AppData\Local\Temp\Rtmpqwg2tI\downloaded_packages
```

```
library(babynames)
```

```
## Warning: package 'babynames' was built under R version
```

`4.3.1` Loading the babynames dataset into our working environment.

```
babyname_data= babynames
babyname_data
```

```
## # A tibble: 1,924,665 × 5
## year sex name n prop
## <dbl> <chr> <chr> <int> <dbl>
## 1 1880 F Mary 7065 0.0724
## 2 1880 F Anna 2604 0.0267
## 3 1880 F Emma 2003 0.0205
## 4 1880 F Elizabeth 1939 0.0199
## 5 1880 F Minnie 1746 0.0179
## 6 1880 F Margaret 1578 0.0162
## 7 1880 F Ida 1472 0.0151
## 8 1880 F Alice 1414 0.0145
## 9 1880 F Bertha 1320 0.0135
## 10 1880 F Sarah 1288 0.0132
## # ℹ 1,924,655 more rows
```

(a) Generate a plot of the reported proportion of babies born with the name ‘Jessie’ over time and interpret the figure.

ANSWER:-

Extracting the subset from the babyname_data dataset having only name 'Jessie'

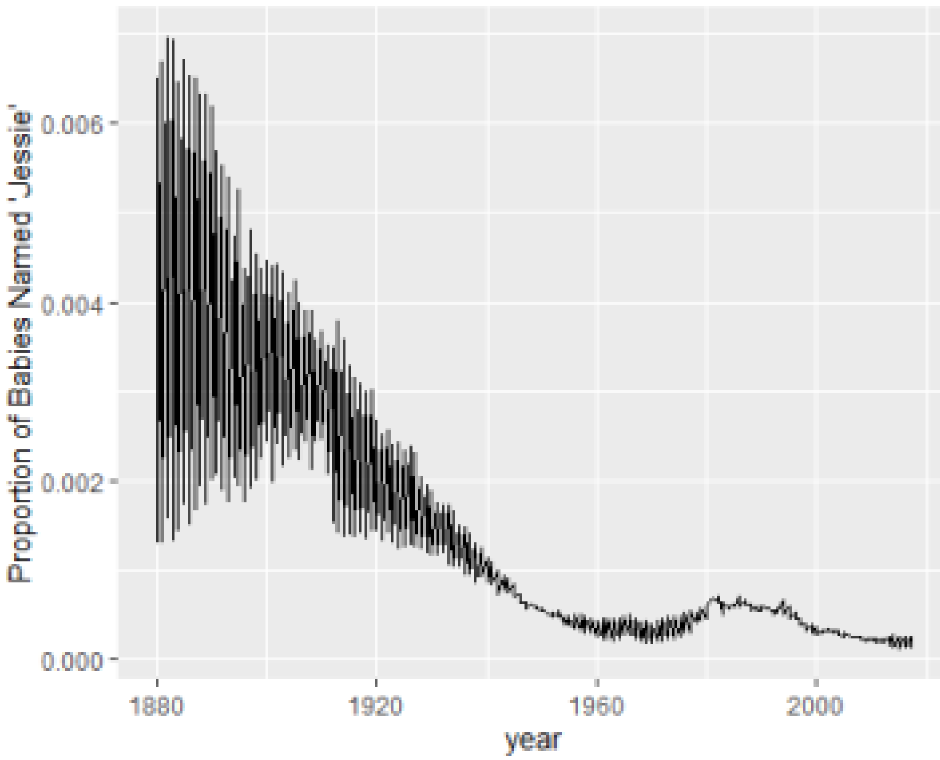
```
babyname_Jessie=babyname_data %>% filter(name=='Jessie')
babyname_Jessie
```

```
## # A tibble: 276 × 5
## year sex name n prop
## <dbl> <chr> <chr> <int> <dbl>
## 1 1880 F Jessie 635 0.00651
## 2 1880 M Jessie 154 0.00130
## 3 1881 F Jessie 661 0.00669
## 4 1881 M Jessie 143 0.00132
## 5 1882 F Jessie 806 0.00697
## 6 1882 M Jessie 192 0.00157
## 7 1883 F Jessie 833 0.00694
## 8 1883 M Jessie 151 0.00134
## 9 1884 F Jessie 888 0.00645
## 10 1884 M Jessie 177 0.00144
## # ℹ 266 more rows
```

Interpretation:-

From the figure we can cleaely see that the trend of boys having name 'Jessie' has decreasing proportation from 1980 to 1940 strictly. Then onward the decrease was not linear.

```
figure_plot=ggplot(data=babyname_Jessie,aes(x = year, y= prop)) + geom_line()
+
labs(y = "Proportion of Babies Named 'Jessie'",x="year")
figure_plot
```



Installing the mdsr package for multidimensional scaling.

```
install.packages("mdsr")
```

```
## Installing package into
'C:/Users/Dell/AppData/Local/R/win-library/4.3' ## (as 'lib' is
unspecified)

## package 'mdsr' successfully unpacked and MD5 sums
checked ##
## The downloaded binary packages are in
 ## C:\Users\Dell\AppData\Local\Temp\Rtmpqwg2tI\downloaded_packages
library(mdsr)

## Warning: package 'mdsr' was built under R version 4.3.1
```

(b) Recreate plots that resembles the graphic in following link for boys named "Jessie" and girls named "Jessie".

ANSWER:-

Plot for Age Distribution of American Boys Named Jessie.

```
# Create the babynames distance matrix
BabynamesDist= make_babynames_dist()

# Filter the data for boys named "Jessie"
Jessie_M= BabynamesDist %>%
 filter(name == "Jessie" & sex == "M")

# Create the plot for age distribution of boys named
"Jessie" name_plot= ggplot(data = Jessie_M, aes(x = year))
```

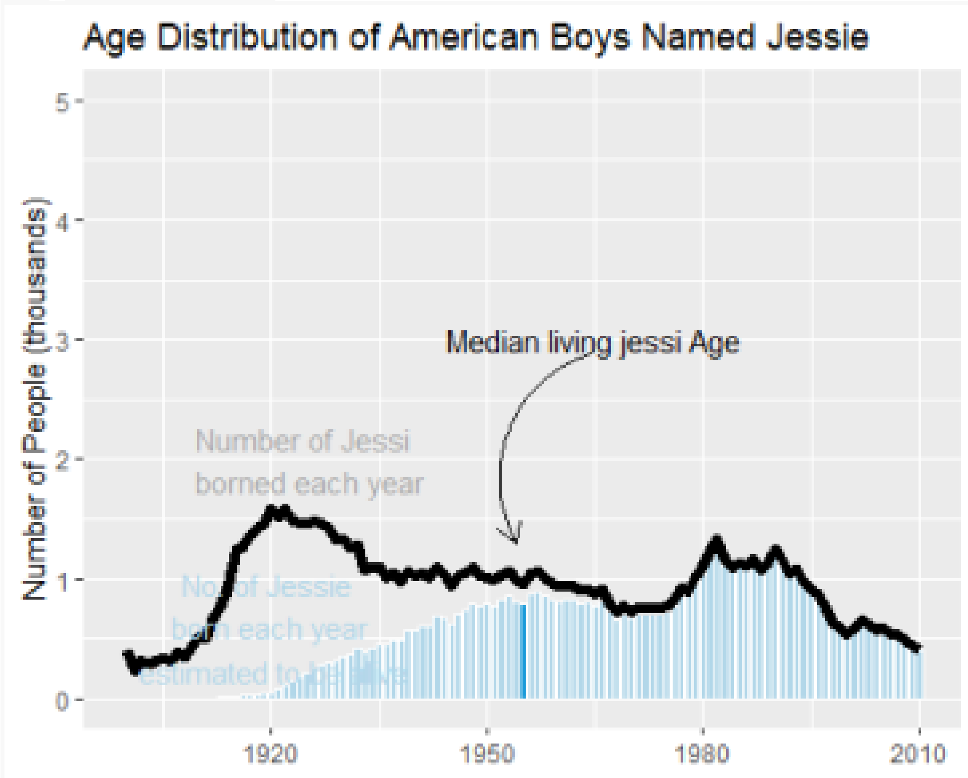
```
# Add columns representing the count of people and estimated alive probability
name_plot= name_plot +
 geom_col(aes(y = count_thousands * alive_prob), fill = "#b2d7e9", color =
"white", size = 0.3)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2
3.4.0. ## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning
was ## generated.
```

```
# Add a line representing the count of people
name_plot= name_plot +
 geom_line(aes(y = count_thousands), size = 2)

# Set axis labels
name_plot= name_plot +
 ylab("Number of People (thousands)") +
 xlab(NULL)

# Calculate the median year of birth
median_yob= median(Jessie_M$year)


# Create a context data frame for annotations

context= data.frame(
 year = c( 1965, 1920,1925),
 num_people = c(3, 0.6,2),
 label = c( "Median living jessi Age", 'No. of Jessie \n born each year \n
estimated to be alive','Number of Jessi \n borned each year') )

# Add a column for the estimated alive today and plot a column for the median
year of birth
name_plot= name_plot +
 geom_col(aes(y = ifelse(year == median_yob, est_alive_today / 1000, 0)),
color = "white", fill = "#008fd5")

# Add annotations and additional elements to the plot
name_plot +
 ggtitle("Age Distribution of American Boys Named Jessie") +
geom_text(data = context, aes(y = num_people, label = label, color =
label)) +
 geom_curve(x = 1965, xend = 1954, y = 2.9, yend = 1.3, arrow = arrow(length
= unit(0.3, "cm")), curvature = 0.5) +
 scale_color_manual(guide = FALSE, values = c("black" ,"#b2d7e9",
"darkgray")) +
 ylim(0, 5)
```

```
## Warning: The `guide` argument in `scale_*()` cannot be `FALSE`. This was
deprecated in
## ggplot2 3.3.4.
```

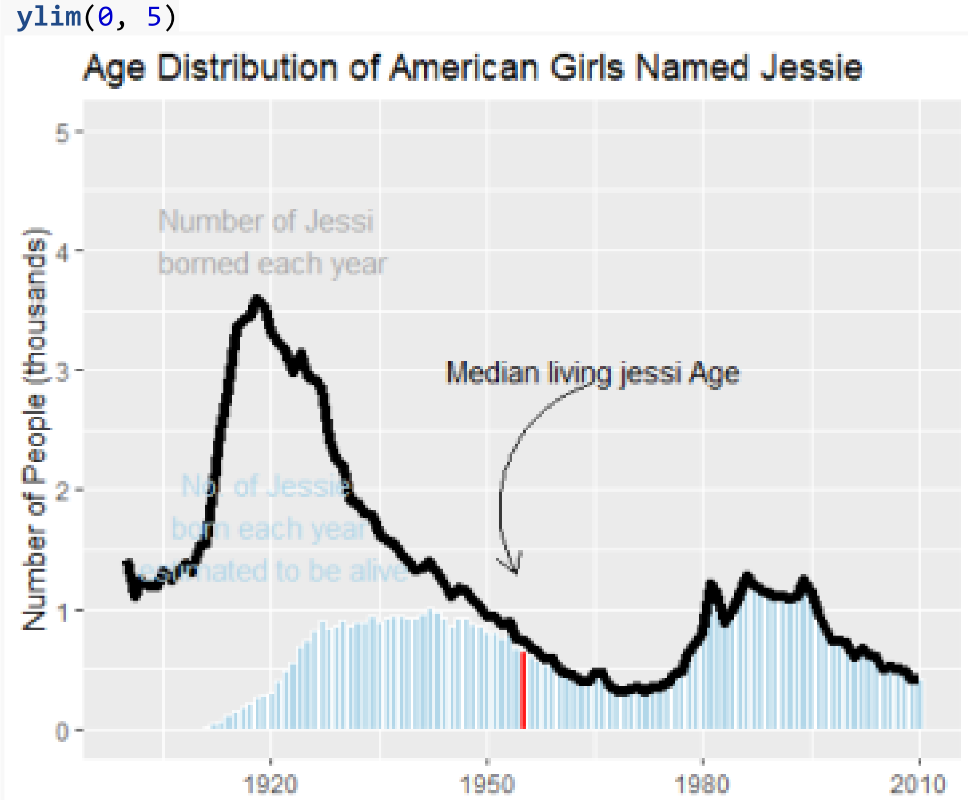
```
## ℹ Please use "none" instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning
was ## generated.
```



And Plot for Age Distribution of American girl Named Jessie.

```
# Create the babynames distance matrix
BabynamesDist= make_babynames_dist()

# Filter the data for girl named "Jessie"
Jessie= BabynamesDist %>%
 filter(name == "Jessie" & sex == "F")

# Create the plot for age distribution of boys named
"Jessie" name_plot= ggplot(data = Jessie, aes(x = year))

# Add columns representing the count of people and estimated alive
probability
name_plot= name_plot +
 geom_col(aes(y = count_thousands * alive_prob), fill = "#b2d7e9", color =
"white", size = 0.3)

# Add a line representing the count of people
name_plot= name_plot +
 geom_line(aes(y = count_thousands), size = 2)

# Set axis labels
name_plot= name_plot +
 ylab("Number of People (thousands)") +
 xlab(NULL)
```

```r
# Calculate the median year of birth
median_yob= median(Jessie$year)


# Create a context data frame for annotations

context= data.frame(
 year = c( 1965, 1920,1920),
 num_people = c(3, 1.7,4.1),
 label = c( "Median living jessi Age", 'No. of Jessie \n born each year \n
estimated to be alive','Number of Jessi \n borned each year') )

# Add a column for the estimated alive today and plot a column for the median
year of birth
name_plot= name_plot +
 geom_col(aes(y = ifelse(year == median_yob, est_alive_today / 1000, 0)),
color = "white", fill = "red")

# Add annotations and additional elements to the plot
name_plot +
 ggtitle("Age Distribution of American Girls Named Jessie") +
geom_text(data = context, aes(y = num_people, label = label, color =
label)) +
 geom_curve(x = 1965, xend = 1954, y = 2.9, yend = 1.3, arrow = arrow(length
= unit(0.3, "cm")), curvature = 0.5) +
 scale_color_manual(guide = FALSE, values = c("black" ,"#b2d7e9",
"darkgray")) +
 ylim(0, 5)
```



Week-2_HW_Stat_560_R_part_QN-3

Sagar Kalauni

2023-07-15

```
options(repos = "https://cran.rstudio.com/")
```

Installing library tidyverse

```
install.packages("tidyverse")
```

```
## Installing package into
'C:/Users/Dell/AppData/Local/R/win-library/4.3' ## (as 'lib' is
unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums
checked ##
## The downloaded binary packages are in
## C:\Users\Dell\AppData\Local\Temp\RtmpA9KcxX\downloaded_packages
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version

4.3.1 ## Warning: package 'ggplot2' was built under R version

4.3.1 ## Warning: package 'lubridate' was built under R

version 4.3.1
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr 1.1.2 ✓ readr 2.1.4
## ✓ forcats 1.0.0 ✓ stringr 1.5.0
## ✓ ggplot2 3.4.2 ✓ tibble 3.2.1
## ✓ lubridate 1.9.2 ✓ tidyr 1.3.0
## ✓ purrr 1.0.1
```

```
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag() masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Installing and loading the nasaweather package.

```
install.packages("nasaweather")
```

```
## Installing package into
'C:/Users/Dell/AppData/Local/R/win-library/4.3' ## (as 'lib' is
unspecified)
## package 'nasaweather' successfully unpacked and MD5 sums
checked ##
## The downloaded binary packages are in
```

```
## C:\Users\Dell\AppData\Local\Temp\RtmpA9KcxX\downloaded_packages
```

```
library(nasaweather)
```

```
##
## Attaching package: 'nasaweather'
```

```
## The following object is masked from
'package:dplyr': ##
## storms
```

Loading the storms data into the working environment.

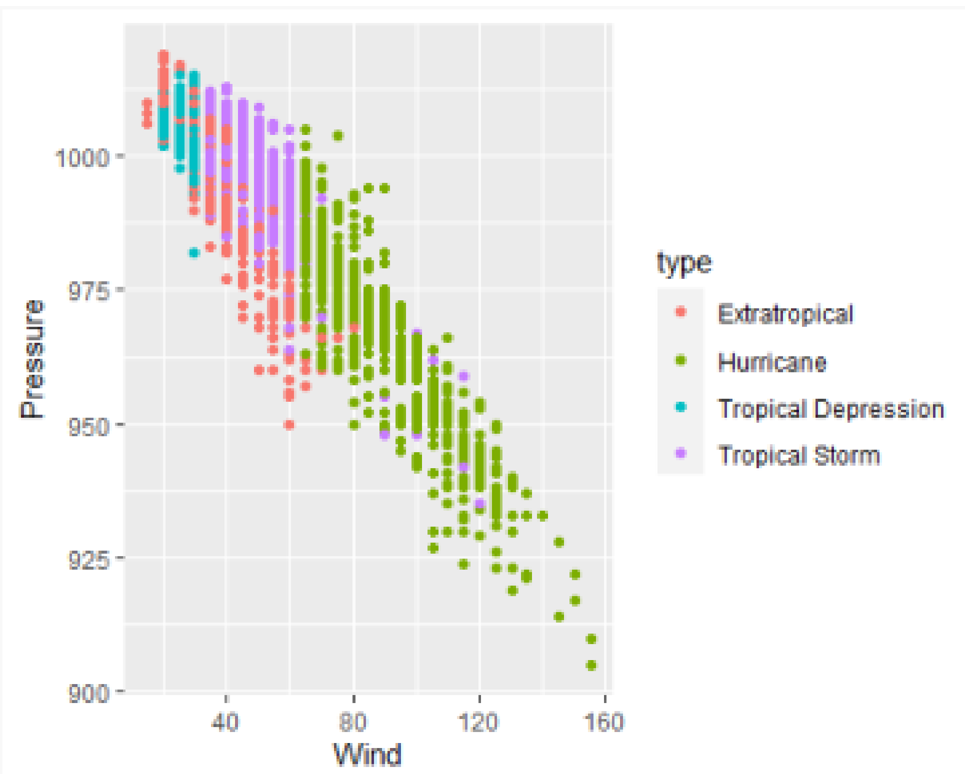
```
storm_data= storms
```

(a) Create a scatterplot between wind and pressure, with color being used to distinguish the type of storm.

ANSWER:-

The graph is below:

```
figure_scattered= ggplot(data=storm_data,aes(y = pressure,x
= wind,color=type)) + geom_point() +
labs(y = "Pressure",x="Wind")
figure_scattered
```



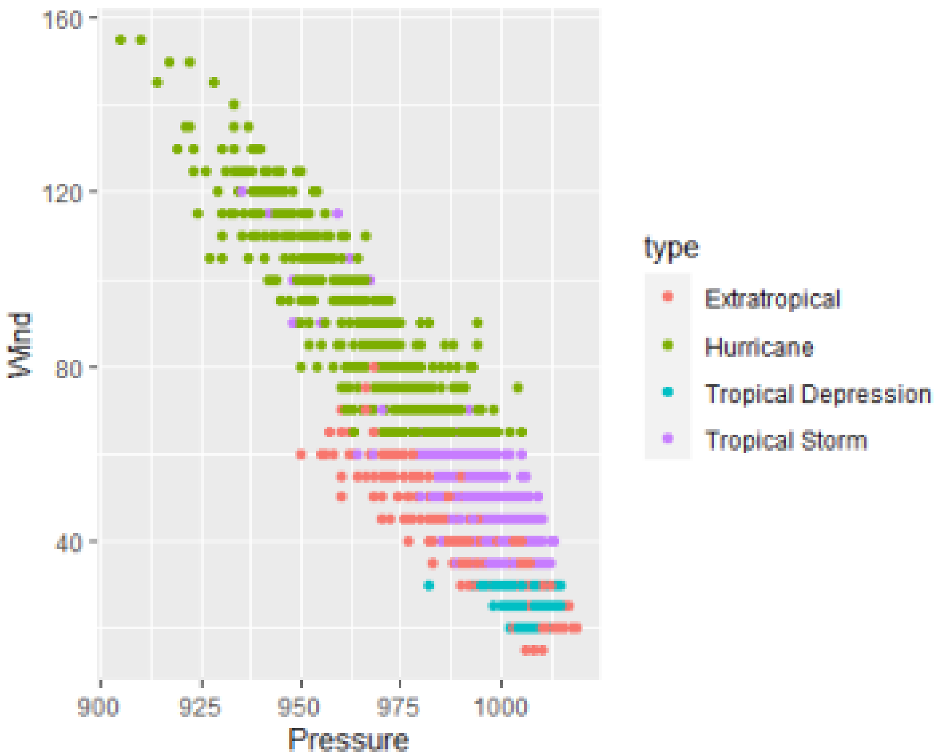
(b) From the scatterplot, what type of an association is apparent between wind and pressure? What type of an association would you expect to see if the axes of the plot were reversed?

ANSWER:-

Based on the scatterplot generated from the storm data, an analysis reveals a negative association between the variables "Wind" and "Pressure." This indicates that Where there is high Pressure, wind speed is low, i.e as pressure decreases, wind speed increases and vice versa. Also from scatter plot we can clearly see that Tropical Depression is occured only when the pressure is relatively high.

If axes of the plot were reversed:

```
figure_scattered= ggplot(data=storm_data,aes(x = pressure,
y= wind,color=type)) + geom_point() +
labs(x = "Pressure",y="Wind")
figure_scattered
```



No matter if your reversed the axes of the scatter plot, the negative co-relation between Wind and Pressure will still be there.

(c) (Extra Credit) Use the geom path function to plot the path of each tropical storm in the data table. Use color to distinguish the storms from one another, and use faceting to plot each year in its own panel.

ANSWER:-

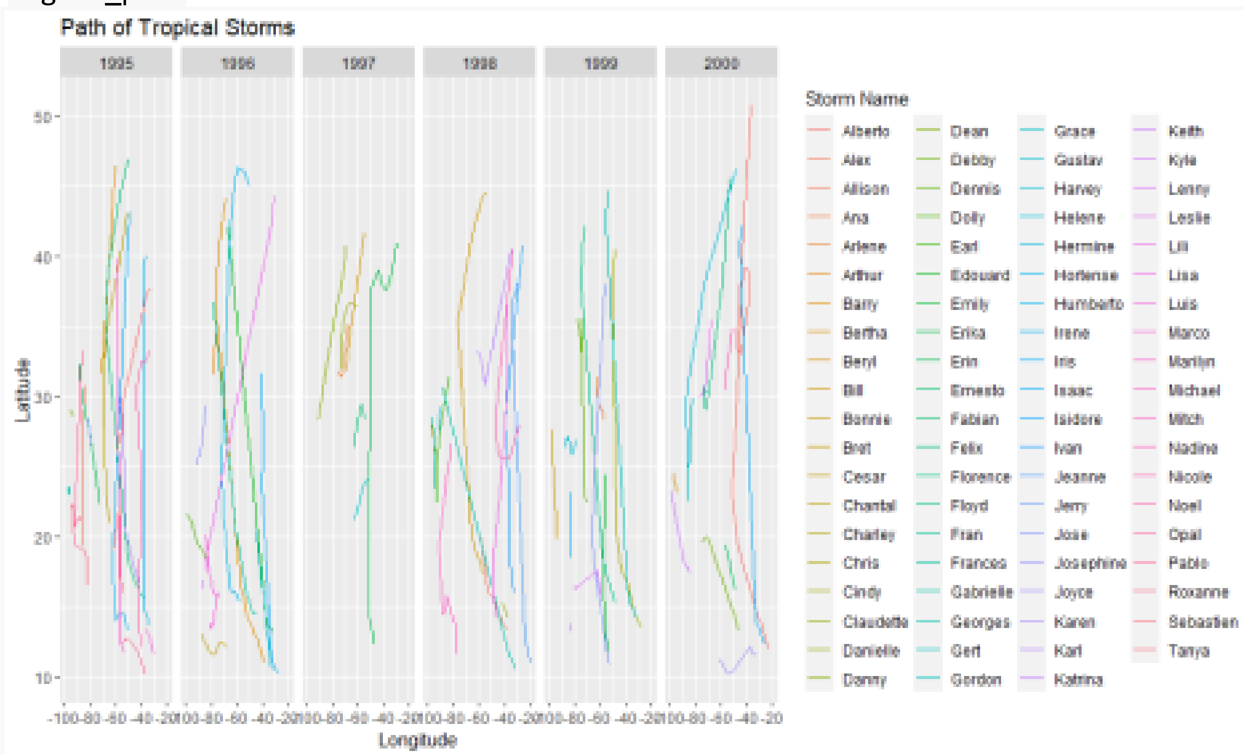
Extracting only Tropical Storm type data from whole strom dataset.

```
tropical_storms = storm_data %>% filter( type == "Tropical
Storm") tropical_storms
```

```
## # A tibble: 926 × 11
## name year month day hour lat long pressure wind type  seasday
## <chr> <int> <int> <int> <int> <dbl> <dbl> <int> <int> <chr>  <int>
## 1 Allison 1995 6 3 12 19.3 -85.7 1003 35 Tropical …  3
## 2 Allison 1995 6 3 18 20.6 -85.8 1001 40 Tropical …  3
## 3 Allison 1995 6 4 0 22 -86 997 50 Tropical …  4
## 4 Allison 1995 6 4 6 23.3 -86.3 995 60 Tropical …  4
## 5 Allison 1995 6 5 6 28.5 -85.6 990 60 Tropical …  5
## 6 Allison 1995 6 5 12 29.6 -84.7 990 60 Tropical …  5
## 7 Allison 1995 6 5 18 30.7 -83.8 993 45 Tropical …  5
```

```
## 8 Barry 1995 7 7 6 31.6 -71 1007 35 Tropical …  37
## 9 Barry 1995 7 7 12 32.2 -70.6 1004 40 Tropical …  37
## 10 Barry 1995 7 7 18 33.2 -70.2 1001 60 Tropical …  37
## # ℹ 916 more rows
```

```
figure_path= ggplot(tropical_storms, aes(x = long, y = lat, group = name,
color = name)) +
geom_path() + labs(title = "Path of Tropical Storms",x = "Longitude",y
= "Latitude",color = "Storm Name") +
facet_wrap(~year, nrow = 1)
figure_path
```



Since the I was not satisfied with the quality of the output shown in the word, so I export my graph separately and pasted over here. [graph = output]

Week-2_HW_Stat-560_QN-3(Theory Part)

Sagar

3. The following plot presents the prices and other attributes of almost 54,000 diamonds. You may see the descriptions of the variables using command? diamonds in R once you load the tidyverse library.

(a) Identify the visual cues, coordinate system, scales and context present in the plot? ANSWER: -

The visual cues, coordinate system, scales, and context present in the plot are as follows: Visual cues:

- Position: The position of the points on the x-axis represents the carat weight, and the position on the y-axis represents the price. The position of each point indicates the relationship between carat weight and price.

• Color: The color of the points represents the color of the diamonds. • Shape:

The points are represented by circles, indicating individual data points.

• Faceting: The plot is divided into separate panels based on the "cut" variable, allowing for comparisons between different cut categories.

Coordinate system: The plot uses a Cartesian coordinate system, with the x-axis representing the carat weight and the y-axis representing the price.

Scales:

• X-scale: The scale for the x-axis represents the carat weight of the diamonds. •

Y-scale: The scale for the y-axis represents the price of the diamonds. • Color

scale: The color scale represents the different colors of the diamonds.

Context: The plot includes faceting, which means the data is divided into separate panels based on the "cut" variable. Each panel shows a specific subset of the data, allowing for comparisons based on different levels of the "cut" variable.

(b) How many variables are depicted in the graphic? Explicitly link each variable to a visual cue that you listed above.

ANSWER:-

The graphic depicts the following variables:

1. Carat weight: This variable is represented by the x-coordinate position of the points. As the carat weight increases, the points move further along the x-axis.
2. Price: This variable is represented by the y-coordinate position of the points. As the price increases, the points move further along the y-axis.
3. Color: This categorical variable is represented by the color of the points. Different colors indicate different diamond colors.
4. Cut: This categorical variable is depicted through faceting. Each panel represents a specific level of the "cut" variable, allowing for comparisons between different cut categories.

The visual cues listed above are linked to these variables as follows:

• Position (x-coordinate and y-coordinate) represents carat weight and price. •

Color represents the color variable.

• Faceting represents the cut variable, creating separate panels for each cut category.

Sagar

2023-07-12

Concepts

Queson-1

1. For each of the following, state whether you expect the distribution to be symmetric, right skewed, or left skewed. Also specify whether the mean or median would best represent a typical observation in the data, and whether the variability of observations would be best represented using the standard deviation or IQR. Explain your reasoning.

(a) Housing prices in a country where 25% of the houses cost below $300,000, 50% of the houses cost below $600,000, 75% of the houses cost below $900,000 and very few houses that cost more than $1,200,000.

ANSWER:-

The distribution of housing prices is likely to be right skewed. The presence of very few houses that cost more than $1,200,000 suggests a long tail on the right side of the distribution. The median would best represent a typical observation since it divides the data into two equal halves. The variability of observations would be best represented using the interquartile range (IQR) since it provides a measure of spread that is not affected by extreme values in the tails of the distribution.

(b) Number of alcoholic drinks consumed by college students in a given week. Assume that most of these students don't drink since they are under 21 years old, and only a few drink excessively.

ANSWER:-

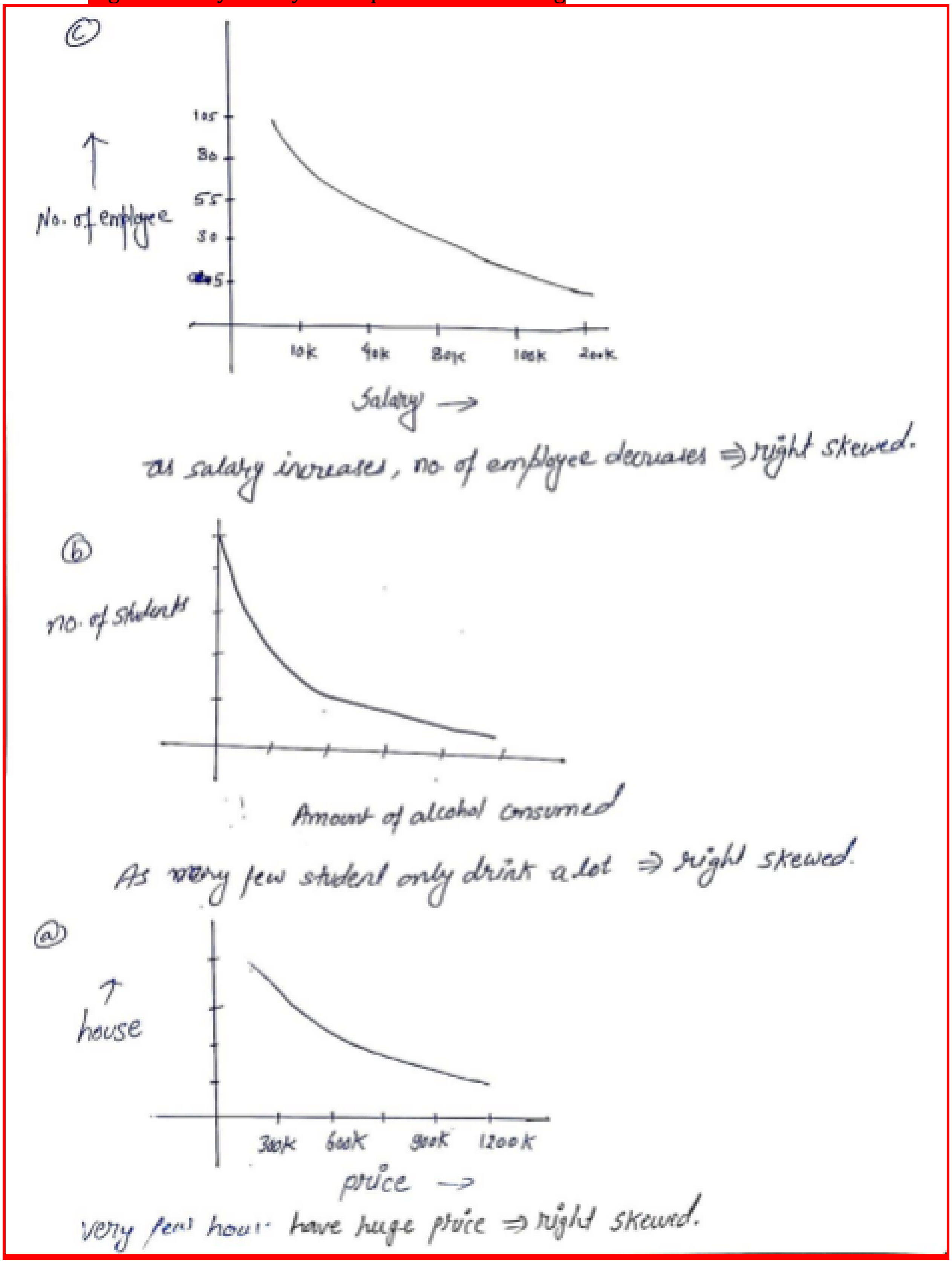
The distribution of the number of alcoholic drinks consumed by college students in a given week is likely to be right skewed. Since most students don't drink due to being under 21 years old, the majority of observations would be clustered around zero. However, there would be a small number of students who drink excessively, leading to a long tail on the right side of the distribution. In this case, the median would best represent a typical observation since it is less affected by extreme values. The variability of observations would be best represented using the interquartile range (IQR), as it focuses on the middle 50% of the data and is not inluenced by extreme values.

(c) Annual salaries of the employees at a Fortune 500 company where only a few high level executives earn much higher salaries than all the other employees.

ANSWER:-

The distribution of annual salaries at the Fortune 500 company is likely to be right skewed. Since only a few high-level executives earn much higher salaries than all the other employees, there would be a long tail on the right side of the distribution. The majority of employees would have salaries clustered around a lower range, while a few outliers would have signiicantly higher salaries. In this case, the median would be a better representation of a typical observation since it is less inluenced by extreme values. The variability of observations would be best represented using the IQR.

#Note:- Figure is only for my concept not for checking



Queson-2

2. Poppy Pringle recorded the heights in centimeters of the sunlowers growing in her brother's and sister's gardens. Here are the heights of the 19 sunlowers in her brother's garden: 94, 103, 110, 111, 123, 143, 150, 150, 151, 156, 157, 160, 170,

182, 201, 220, 250, 255, 231.

(a) Compute the mean, standard deviation, and the ive-number summary to show information about this data.

ANSWER:-

Solution for Q.No. 2(a)

1> Mean

$$\text{Mean} = \frac{94+103+110+111+123+143+150+150+161+156+157+160+170+182+201+220+250+255+231}{19}$$

$$= \frac{3117}{19}$$

$$= 164.0526 \qquad \therefore \bar{X} = 164.0526$$

2> Standard Deviation

X	$X-\bar{X}$	$(X-\bar{X})^2$
94	−70.0526	4907.3661
103	−61.0626	3727.413
110	−54.0626	2921.618
111	−53.0626	2814.578
123	−41.0526	1685.315
143	−21.0526	443.211
150	−14.0526	197.475
150	−14.0526	197.475
151	−13.0526	170.370
156	−8.0526	64.844
157	−7.0526	49.739
160	−4.0526	16.423
170	5.9474	35.371
182	17.9474	322.103
201	36.9474	1365.110
220	55.9474	3130.111
250	85.9474	7386.355
255	90.9474	8271.429
231	66.9474	4481.354

Now, standard deviation is given by

$$SD = \sqrt{\frac{\Sigma (X-\bar{X})^2}{n-1}}$$

where n=19 given

$$= 48.41312$$

3.) Five-number summary

the five number summary consists of the minimum, first quartile (Q_1), median (Q_2), third quartile (Q_3) and maximum value of the data

- 94, 103, 110, 111, 123, 143, 150, 150, 151, 156, 157, 160, 170, 182, 201, 220, 231, 250, 255 (sorted)

- Minimum = 94
- Q_1 = The median of the lower half of the data

$$= \frac{123+143}{2} = 133$$

- Q_2 = Median

= 156

- Q_3 = The median of the upper half of the data

= 191.5

- Maximum = 255

Therefore, the five-number summary is {94, 133, 156, 19.5, 255}

The Same thing if we want to do in R with code

```
Data=c(94, 103, 110, 111, 123, 143, 150, 150, 151, 156, 157, 160, 170, 182,
201, 220, 250, 255, 231)
sum(Data)

## [1] 3117

mean(Data)

## [1] 164.0526
sd(Data)

## [1] 48.41312

data= sort(Data) (sorted)
```

```
## [1] 94 103 110 111 123 143 150 150 151 156 157 160 170 182 201 220 231
250 255
```

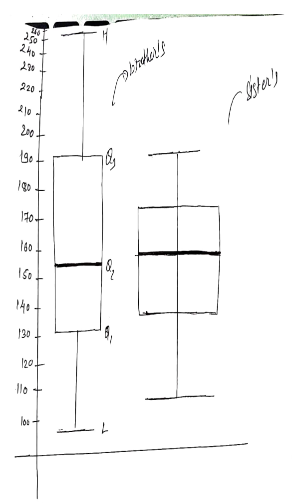
```r
quantile(data)
```

```
## 0% 25% 50% 75% 100%
## 94.0 133.0 156.0 191.5 255.0
```

(b) Here is some information about the sunlowers in Poppy's sister's garden. Construct a side-by-side boxplot by hand for the heights of sunlowers in each of the gardens. median(Q2):160 lower_quartile(Q1):143 upper_quartile(Q3):172 shortest:110 tallest:199.

ANSWER:-

For the answer, look at the attached igure.



(c) Poppy says that the sunlowers in her brother's garden are shorter than those in her sister's garden. Is Poppy correct? Explain.

ANSWER:-

brother's garden median = 156 < sister's garden median =160, so yes she is right.

(d) Whose garden has sunlowers varying more in height? Explain.

ANSWER:-

To determine whose garden has sunlowers varying more in height, we can compare the standard deviations of the heights in Poppy's brother's and sister's gardens. Unfortunately, the heights of the sunlowers in Poppy's sister's garden were not provided, so we don't have the necessary information to make a direct comparison between the two gardens.

But We can also compare the varying height by looking at their Interquartile range (IQR). so IQR for poppy's brother=(Q3-Q1)=(191.5 - 133)*1.5= 87.75

And IQR for poppy's sister=(Q3-Q1)=(172 - 143)*1.5=43.5

Therefore, brother's garden IQR > sister's garden IQR = 43.5

Hence, we can say that Poppy's brother's garden has sunlowers in more varying heights.

Solution for Q.No. -1

	Excellent	Very	Good	Fair	Poor	Total
No coverage	0.0230	0.0364	0.0427	0.0192	0.0050	0.1262
Insu. covered	0.2099	0.3123	0.2410	0.0817	0.0289	0.8738
	0.2329	0.3487	0.2837	0.1009	0.0339	1

a) Are being in excellent health and having health coverage mutually exclusive?

Solu: No. About 20.99 % have excellent health and health coverage.

b) What is the probability that randomly chosen individual has excellent health?

Solu: $P(\text{excellent health}) = 0.2329$

c) What is the probability that a randomly chosen individual has excellent health given that he has health coverage?

Solu: $$P(\text{excellent}/\text{covered}) = \frac{P(\text{excellent and covered})}{P(\text{covered})}$$

$$= \frac{0.2099}{0.8738}$$

$$= 0.24021.$$ □

d.) Do having excellent health and having health coverage appear to be independent?

Sol:

If they were independent, the probability of having excellent health and having health coverage would be equal to the probability of having excellent health times the probability of having health coverage.

The first quantity is 0.2099, and the second is $0.2329 \times 0.8738 = 0.2035$. Though these numbers are close, they are not equal, therefore the two events are not independent.

Solution for Q.NO - 2

let I be the event of identical twins. let M be the event the twins are both male, let F be the event that both twins are female, and let B be the event that they are mixed sex. From the information given in the question, we know that

$P(I) = 0.30 \Rightarrow P(I^c) = 0.70$

$P(F/I^c) = 0.25$.

we want to know $P(I/F)$. using Baye's theorem

$$P(I/F) = \frac{P(F/I) * P(I)}{P(F/I) \cdot P(I) + P(F/I^c) \cdot P(I^c)}$$

$$= \frac{(0.50) \cdot (0.30)}{(0.50) \cdot (0.30) + (0.25) \cdot (0.70)}$$

$$= 0.4615. \quad \square$$

Solution for Q.No.-3

let X : Amount win at the game

So,

$$P(X=50) = P(3\,Heart) = \frac{\binom{13}{3}}{\binom{52}{3}} = \frac{11}{850}$$

$$P(X=25) = P(3\,black) = \frac{\binom{26}{3}}{\binom{52}{3}} = \frac{100}{850}$$

$$P(X=0) = 1 - \left(\frac{11}{850} + \frac{100}{850}\right) = \frac{739}{850}$$

(a) Create a probability model for the amount you win at this game.

X	0	25	50
P(X)	$\frac{739}{850}$	$\frac{100}{850}$	$\frac{11}{850}$

(b) Find the expected winnings and compute the standard deviation of this distribution

soln

Expected value $E(X) = \Sigma x_i \cdot P(X=x_i)$

$$= 0\cdot\left(\frac{739}{850}\right) + 25\cdot\left(\frac{100}{850}\right) + 50\cdot\left(\frac{11}{850}\right)$$

$= \$\,3.59.$

Standard deviation $(\sigma_x) = \sqrt{\sigma_x^2} = \sqrt{Var(X)}$

$$= \sqrt{\sum (x_i - \mu_x)^2 \, P(X = x_i)}$$

$$= 9.6440 \, . \square$$

ⓒ If the game costs \$5 to play, what ~~should~~ would be the expected value and standard deviation of the net profit (or loss)?

solu

X	-5	20	45
$P(X)$	$\frac{739}{850}$	$\frac{100}{850}$	$\frac{11}{850}$

Expected value $E(X) = \sum x_i \cdot P(X = x_i)$

$$= \$ -1.41$$

Standard deviation $(\sigma_x) = \sqrt{\sigma_x^2} = \sqrt{var(X)}$

$$= \sqrt{\sum (x_i - \mu_x)^2 \cdot P(X = x_i)}$$

$= 9.64$ (not Changed)

d.) If the game costs \$5 to play, should you play this game ? Explain.

Solu

Since you are gettind \$-1.41 as your expected return which means in a long run you are expected to lose \$1.41 per game. So its good idea not to play a game

Solution for Q. No. 5

Since Robin is randomly guessing, the probability of getting any question right is 1/4.

a) The first question she gets right is the 3rd question?

Solu In order for her to get 3rd question right, she has to get first two question wrong. The probability of getting first two questions wrong is $= (3/4)^2$

And the probability of getting 3rd question right $= (1/4)$

$\therefore$ overall probability $= (3/4)^2 * (1/4)$

$= 0.140$

b) She gets exactly 3 or exactly 4 questions right?

Solu

$$P(X=3) = \binom{5}{3}\left(\frac{1}{4}\right)^3\left(\frac{3}{4}\right)^2$$

$$= 10 * \frac{1}{64} * \frac{9}{16}$$

$$= 0.08789$$

$$P(X=4) = \binom{5}{4}\left(\frac{1}{4}\right)^4\left(\frac{3}{4}\right)^1$$

$$= 5 * \frac{1}{256} * \frac{3}{4}$$

$$= 0.01464 . \square$$

(c) She get the majority (3 or more) of the questions right?

Solu

$P(\text{3 or more right}) = P(X \geq 3)$

$= P(X=3) + P(X=4) + P(X=5)$

Here

$P(X=5) = \binom{5}{5}\left(\frac{1}{4}\right)^5\left(\frac{3}{4}\right)^0$

$= 1 * \frac{1}{1024}$

$= 0.000976$

$\therefore$ $P(\text{3 or more right}) = 0.08789 + 0.01464 + 0.000976$

$= 0.1035$. $\square$

Week-3_QN_4(theory_part)

Sagar Kalauni

2023-07-19

4.(Normal distribution) Sophia who took the Graduate Record Examination (GRE) scored 160 on the Verbal Reasoning section and 157 on the Quantitative Reasoning section. The mean score for Verbal Reasoning section for all test takers was 151 with a standard deviation of 7, and the mean score for the Quantitative Reasoning was 153 with a standard deviation of 7.67. Suppose that both distributions are nearly normal.

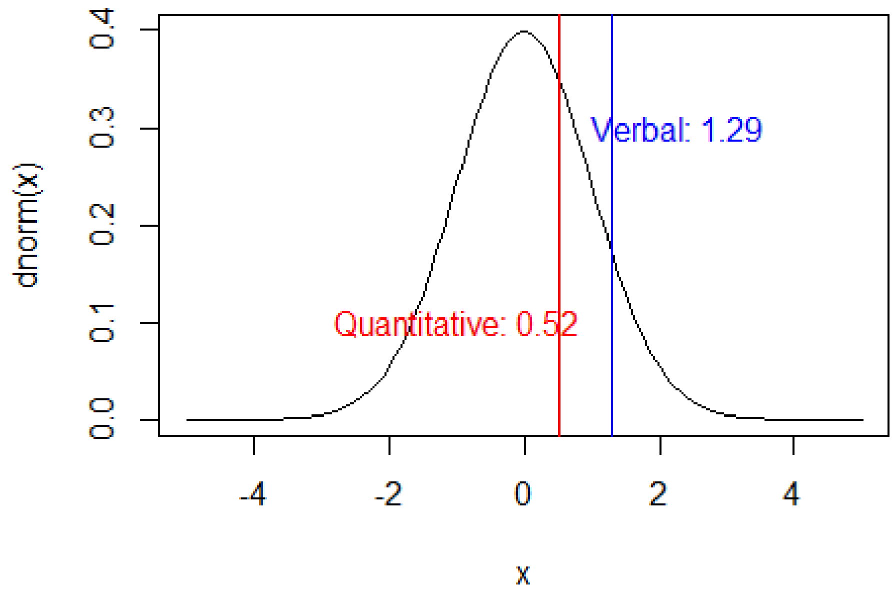
(a) What is Sophia's Z-score on the Verbal Reasoning section? On the Quantitative Reasoning section? Draw a standard normal distribution curve and mark these two Z-scores.

ANSWER:- for sophia, we are given Verbal: $N(\mu=151,\sigma=7)$ Quantitative: $N(\mu=153,\sigma=7.67)$

Verbal: $Z=(160-151)/7 = 1.285714$

Quantitative: $Z=(157-153)/7.67 = 0.5215124$

```r
curve(dnorm, from = -5, to=5)
abline(v=1.285714, col="blue")
abline(v=0.5215124, col="red")
text(1.285714+1, 0.3, "Verbal: 1.29",col="blue")
text(0.5215124-1.5, 0.1, "Quantitative: 0.52", col="red")
```



(b) What do these Z-scores tell you? Which section did she do better on?

ANSWER: - The Z-scores tell me how far above the average scores Sophia scored for both tests. That is, she scored 1.2857 standard deviations above the mean for the Verbal test and 0.5215 standard deviations above the mean for the Quantitative test. Sophia did better on the Verbal section than she did on the Quantitative section since that has the higher Z-score.

(c) What percent of the test takers did better than her on the Verbal Reasoning section? On the Quantitative Reasoning section?

ANSWER: - To know what percent of the test taker did better than her on the verbal and quantitative section, first we need to know her scoring percentile. To look at scoring percentile, we can either use table or R. using R to find her percentile on verbal.

```r
pnorm(1.29)

## [1] 0.9014747

# her percentile on verbal
```

using R to find her percentile on quantitative.

```r
pnorm(0.52)

## [1] 0.6984682
```

So clearly from result we can see that she lies on 90th percentile on verbal and approximately 70th percentile on quantitative.

Given the above, 10% of test takers did better than her on the Verbal Reasoning section (100-90) and 30% of test takers did better than her on the Quantitative Reasoning section (100-70).

(d) Explain why simply comparing raw scores from the two sections could lead to an incorrect conclusion as to which section a student did better on.

ANSWER: - I don't think it would, but it could. In the current problem, the raw Verbal score is higher than the raw quantitative score. However, it is possible for someone to have a higher raw score in one section than the other, but in fact do worse in that section. This can occur when the scales used are very different, or even slightly different as in this problem.

For example, if we lower her Verbal to 156, then her new Z-score for Verbal is (156-151)/7 = 0.7142857. Notice that even though the raw score is lower than her Quantitative score, the Z-score is still higher than the Z-score of her Quantitative score. In other words, she still would have done better on the Verbal with this score, even though the raw score is lower than the Quantitative score. This highlights the importance of normalizing values before doing comparisons so that the comparisons are done using the same units of measure (in this case, standard deviations from the mean).

(e) Calculate the score of a student who scored worse than 70% of the test takers in the Verbal Reasoning section.

ANSWER: - To calculate the score of the studnets who scored worse than 70% we can use the following code chunck:

```
mean_verbal= 151
sd_verbal= 7

desired_percentile= 0.70

score= qnorm(desired_percentile, mean = mean_verbal, sd = sd_verbal)
score

## [1] 154.6708
```

So their score is 154.6708, who score worser then 70% of the tast takers.

Week_3_HW

Sagar Kalauni

2023-07-21

```
options(repos = "https://cran.rstudio.com/")
```

R Practise

Attach all of the R code that you used to answer the following questions. You may use the method of your choice to write iterative operations, such as apply(), map(), or simply write a loop.

1. Simulate the distributions of a random variable using the following steps:

(a) Write a function called gen_exp to do the following: Generate a random number, y, between 0 and 1 such that each number between 0 and 1 is equally likely. This can be done using runif(0,1) in R. Then generate a x according to $x = -m \log(1 - y)$. Return the value of x.

ANSWER:- generating the function named gen_exp. In R we can define and generate our own function using the function() function.Where inside () we can give the argument we wanna pass inside the function and inside the{}, we will write what the function will actually do with the taken argument and provide the return.

```
gen_exp= function(m){
y= runif(1,0, 1)
x = -m*log(1-y)
return(x)
}
```

Calling the function.

```
gen_exp(2.5)         # lets try by calling the function with m-value of 2.5
and see the output.

## [1] 1.69277
```

The y used inside my function a random number between 0 and 1, and since it is a random number between 0 and 1, so all the numbers between 0 and 1 have equal probability of being y. You can check this by looking at this code chunck. This code chunck will produce differnt y values between 0 and 1 everytime exucated.

```
y= runif(1,0,1)
y

## [1] 0.001424506
```

(b) Repeat your gen_exp function 1000 times for m=10. Create a histogram of the simulation. Calculate the mean and standard deviation of the simulated values.

ANSWER:-

```
m=10                          # putting the m value as fixed 10
N=1000                        # Because I have to repete it for 1000 times so
putting simulation value to 1000.



values= numeric(N)
for (i in 1:N){                  # Reperting the gen_exp(m) function 1000
times for the m-value=10
values[i]= gen_exp(m)
}
```

Looking at the generated values:

```
values

##    [1]  2.604311890 18.639810775  7.517231407 13.788701935 29.002709908
##    [6]  1.742088310  6.528277993  1.946406533  6.466201384 30.834522167
##   [11]  2.818455655 26.854946624  5.171646472  2.118889908  9.672943898
##   [16]  4.755186949 11.834655272 25.990707886  3.682881635  0.305743249
##   [21]  7.504972506 10.278820433  3.504850949  0.019997293  4.061048724
##   [26] 18.847480321  3.431669932 30.399141576  4.988358727 24.817373621
##   [31]  7.136366015 14.282926451 29.376613336  0.788111612  3.944972773
##   [36]  3.483137677  1.620354247  9.168087229 24.917114876  6.670858500
##   [41]  9.165860748 18.579650622  3.388059679 12.631108746  6.331226328
##   [46]  2.811305690 27.738949502 11.466236964 12.095209898  4.629545567
##   [51] 23.735755130  4.262223405  3.430398189  2.479454310  4.035634196
##   [56]  2.298365416  4.230813917 31.014141229  8.203791289  6.119492685
##   [61]  1.114461531 12.780576821  1.591626234 10.845446288  0.216479503
##   [66]  2.022128990  5.893995283  2.673419643  0.276535664 10.614006330
##   [71]  9.153584019 12.220378453 23.732234660  4.205558441 13.360009011
##   [76]  4.971410877 18.279613107  8.331601655  1.240905343 14.273990307
##   [81] 11.988557971 11.036227213  8.533865053 18.928575839 16.579495195
##   [86] 28.894318266  8.215260056  4.055436348 10.769955949  3.643547435
##   [91]  0.300275043  9.772644209  7.530076262  7.387159794  7.839889578
##   [96]  4.860617055  9.568910590 13.203215470 59.728912544 21.402294199
##  [101] 20.702264848 15.352036077 18.876809984  6.381736891  5.503173458
##  [106]  2.356697797  3.622586661 19.539027435 49.141312738  3.437242375
##  [111]  7.582190780  1.037950164 24.503121887  2.745627571  8.453140551
##  [116]  2.084863291  5.583624755 20.552439733  4.484353249 21.978609944
##  [121]  3.796327985  6.921982784  8.641819039  2.399981760  9.964628092
##  [126]  8.283424889  3.980050988 13.517345185  0.407097678  0.355133815
##  [131]  7.507494071  1.645906181 12.672852375 11.303573550  0.024073885
##  [136] 32.607020974  5.242260406 18.458011508  0.006528287  4.632286105
##  [141]  1.226982242  3.835871271  0.569717334  6.014090612  6.151497007
##  [146]  2.217027903 36.978463795  9.734194019  1.909472391 35.504697565
##  [151]  2.908603200  4.531279744  1.159767686  0.660179910 18.381047920
##  [156]  0.512500690 13.756426282  5.168437704  4.853000842  3.125566542
```

```
## [161] 22.232417573  4.962462923  5.625462740  2.107107672  0.626348252
## [166] 21.178054574 21.458848235  0.841083016  7.752388392  5.742679584
## [171]  0.979813451  1.991271041  3.514005164  1.913540635  3.469410953
## [176]  0.236031755 15.332958660 14.020821119  5.213552893  5.068515856
## [181]  2.877544013  2.367297289 15.620419030  3.335561222 14.919477322
## [186]  1.057310594  9.080819480  5.516689037 29.626410118  6.591529685
## [191]  3.982823937  5.380378070  9.702920202  8.484501105  5.044491636
## [196]  7.234689960 13.597531961  1.085439876  8.007697355  5.350262318
## [201]  9.240336933  1.526353910  1.677073393 13.061310725 17.791386113
## [206]  5.961248033 26.312983133  4.899302185  1.652029894  2.470464218
## [211]  9.410870704  6.304026073 19.916735949 20.118719366 16.350352084
## [216]  4.697338455 23.259203203  3.101180447  6.296519142 16.008907014
## [221]  3.097430917 10.654894327  6.753268911  8.566151955 15.626126695
## [226]  3.884320232 12.444098222 15.666817626  5.988564832  5.522693992
## [231]  1.013318494 13.021624513  3.028178334  2.907534196  0.912641028
## [236]  4.850036936  5.134476512  3.473570402  4.578839504  0.969128496
## [241] 29.813301558  5.318618395  8.881183869 12.300871705 12.395957413
## [246]  0.841677576  8.633813316  9.167217497  5.950177566  1.400403634
## [251] 17.371067221  1.060855948  1.544766503 17.751771653 11.209401827
## [256]  0.629470704 23.459052729 18.782845963  4.800843332  2.219728507
## [261]  5.414079287 13.060699953 23.347116860 21.193292461 27.387692615
## [266] 28.482782240 24.105654709 16.441409493 10.018956980 12.322007059
## [271]  0.669419668 28.449820793  0.743618477  4.897818959  2.702115052
## [276]  6.008985031  2.655523722  0.156099192  2.523277086  5.836303021
## [281] 19.229472861 17.008318140  1.007279589  2.476973631  1.399827801
## [286]  7.959835040 24.560533659 28.008789821 19.307298085  3.130961441
## [291] 10.372206735  0.006582728  2.004323783  1.484014338  9.801914856
## [296]  1.391570565  3.269116347  1.456614655  7.449072469 24.953378673
## [301] 21.841044169 25.744685386  5.030080561 56.927131242  3.806572837
## [306] 19.705978113  2.957498041  8.984870398 26.265362449  1.528527563
## [311]  2.399767454 14.686231643  3.087403550  8.156629738  1.756371910
## [316]  8.644141044  3.183915400 32.394234721  3.327016865  7.679390686
## [321] 11.526835650  7.656443956  0.552546529  5.254338801 15.838649354
## [326]  0.394580748 20.708550529  6.555569432  1.533485645  2.726752912
## [331] 26.171174883  5.500792342  2.193972383  1.866813834 26.850121708
## [336] 12.859368309  7.000688055  3.181600569 35.986484757 23.457486237
## [341]  0.436642466  3.927653545 11.306028690  2.895797715  4.138699283
## [346]  9.156646085 19.081195402 23.881648539  0.160382720 20.250487017
## [351]  2.795239430 28.618344380 29.139336420 40.156384568  9.918391443
## [356]  4.396080415  3.311048090 12.256119327  7.589184619 15.235825789
## [361]  6.354412189  2.896770697  3.096930127 14.914661316 19.985598380
## [366] 29.732397459  3.542158057  5.278847099 15.763453769  2.095227513
## [371]  1.606949078  1.076202255  4.149542723 10.448299637  1.626854407
## [376]  6.669029597 16.197725721  4.515620796 13.666408008  0.443272450
## [381]  2.175349269 15.775587073  4.435075753  3.069726453  5.190077965
## [386]  7.662093147 11.642443630  0.261839167  5.818429978 11.145323750
## [391]  1.930921646  0.057774584  2.364788234  8.153301254  1.077402181
## [396] 47.972455159  2.856141642 14.454426109 13.061401067  8.323050564
## [401]  1.315143642  3.493877084  8.252368692  6.313884187  1.704954360
## [406]  2.444954348 60.026635205  4.927780365  7.713015749  7.888130149
```

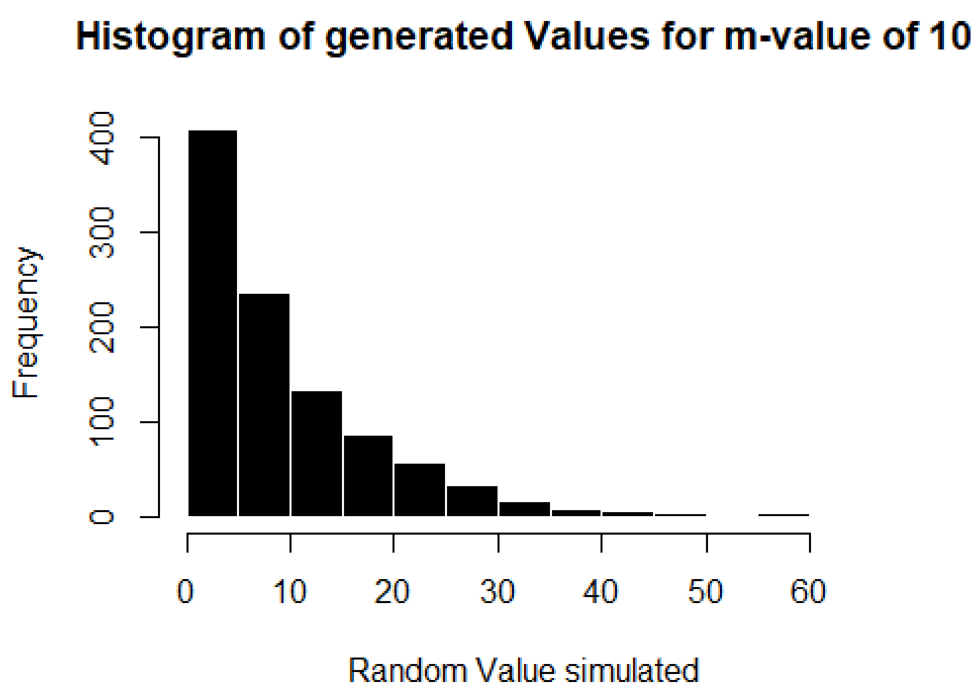
```
##  [411] 22.833102613  2.056515184  4.216096214 15.123518772  4.369595254
##  [416] 15.558002490  4.985950713  1.007032974  1.207348340  2.117756325
##  [421]  0.938481753 34.430126699 21.927432837  7.080983150 22.257580742
##  [426]  6.795903327  1.536708861  1.232255778 42.679507947  8.665835603
##  [431] 18.949547696  0.604220325 14.832872839 41.901318737  1.741539792
##  [436]  1.872632147  1.357471030  8.270767673 23.760352897  4.589262779
##  [441] 17.920151076  3.328550309 37.029106837  1.208131750  0.062719647
##  [446]  4.773097181 16.507541112  5.275300673 12.415418305  0.813135262
##  [451]  3.918706561 13.562617353  1.194524272  5.521389067  5.890407190
##  [456]  2.782156201 24.717894930 13.469193003  6.562475590  0.139667259
##  [461]  8.894741740  4.303078776  3.919712037 10.800774507 17.758087469
##  [466] 10.826026333 15.351559493  4.416089019  3.288296861 11.685083270
##  [471]  9.134962326 36.593911263  3.067317545 11.251571145  0.573445523
##  [476]  3.207519161 39.033202765  1.704491541 10.589679607  0.108187570
##  [481] 25.565293465 14.357434667 10.065288370 27.226468962  3.475142471
##  [486] 15.786145806  1.413598934  3.395079279 33.383297560 12.869131580
##  [491]  6.693501271  4.813894754  7.913295186 12.843888514  4.603681415
##  [496] 13.564740051 10.122448047  6.619920454 16.551935038  3.192014169
##  [501]  0.650888477  1.276631115 41.368228695 24.216962919  0.689764418
##  [506] 29.253564364  7.026442666  4.744628554  0.973502043 13.282809557
##  [511] 21.209244183  0.990102469  4.550307856  6.544937101  4.306993013
##  [516]  9.561290447 31.762836239 14.260266881 12.845272541 10.001260927
##  [521]  5.521412430 11.153942132 25.975575298  2.223390126 17.193254256
##  [526]  3.872849921  3.966752357  0.186885900 17.996058714  8.311922207
##  [531] 12.638514776  2.545681068  6.056361060 23.614318094  1.291917666
##  [536] 16.001177746  1.906824345  2.422465531  3.663343852 14.880087190
##  [541] 14.965555449  2.467508563  3.403821751 20.332589768  2.559391622
##  [546]  2.606495411 12.322299601  9.548855026  3.590308521 13.666804198
##  [551]  7.061367856  9.148482701  9.942479581 35.388386849  1.230531030
##  [556]  0.162676265 11.120779548  6.630070708  7.612765244  8.069442916
##  [561] 16.333978273 27.934280253  1.564241138 23.711846519 15.719113510
##  [566] 10.631705646  6.211118245 12.663216338  7.788432002  2.900681705
##  [571] 10.056498161  3.649001236 14.263175864  2.732886431 10.762097971
##  [576]  7.729477233  6.267928503  3.592634682  7.957770339 10.342658291
##  [581]  5.172047199 13.661458990  6.701386763  7.230836171  0.845558058
##  [586]  8.860606505  9.792083066  1.812916819 14.297832662 11.186216301
##  [591]  1.812661351  9.467855306 30.862544768  3.451315563  0.164425021
##  [596]  1.802613036  7.773557384 17.530508159 14.903887587  0.573656359
##  [601] 22.346737566  0.300715186  8.660263619  6.883748113  6.163888135
##  [606]  0.104892763 12.247051119  1.451314655 19.310892025 10.923406847
##  [611]  1.072063591  0.329669041  3.103108679 16.956094744 11.692381399
##  [616] 17.814182246  3.695019052  0.690493364 15.472633039 15.499733715
##  [621]  3.288518186  4.016222781 26.050942491 15.217844518  8.764681022
##  [626] 13.703789229  0.071924450  9.806690417  2.538147705 22.325362779
##  [631]  7.031775517  3.202977676  2.386656100  3.511398380  0.479832551
##  [636]  2.063654178 11.246941555  0.616724275  1.801373945  2.276839967
##  [641]  1.447134886  2.596066144 27.548858632  0.640818356  0.561348728
##  [646]  5.402130979  4.389108900 14.094650639  2.408649514 11.460145934
##  [651] 15.512052900  3.997777410  0.815807145  9.285806023 13.422198745
##  [656] 23.902097014  7.205252472 10.613859056  6.922607575 19.428261739
```

```
## [661]  4.793738600  5.001636778 19.210501558  0.811151185 13.568112437
## [666] 36.476603271  0.661867309 10.919862648  2.146842159 55.982445266
## [671] 15.975428768  8.168912484  1.226945425 10.530820165  6.034349950
## [676]  3.032382667 16.388711734 15.048475909  7.065103965 10.777777891
## [681]  0.166509171  7.445257807 21.159327269 24.024276961  0.601535200
## [686]  9.866350623  1.404910811  1.244784467  2.791961857  2.663769747
## [691]  4.280984388  9.528277678 11.940505555 16.678249079  2.575841913
## [696]  8.155690277  1.214953875  5.114400704  7.658653144  9.285792853
## [701]  7.220028939 25.607709525  8.289055603  2.431521368  5.090764909
## [706]  4.581610716 14.733643163  8.789298651 20.022771807  1.926127154
## [711]  1.609088642  6.644303967 10.672817549  3.058426556  6.620986867
## [716]  6.103483912  4.177259313 20.681345216  5.562188005  6.357545436
## [721]  6.486237123 24.136808349 10.755705417 18.896767014  2.379547421
## [726]  5.482970678  1.400877506  1.399566800  1.467841772  5.205764509
## [731]  5.984885265  3.292453194  2.599611921  1.324514321  3.552497487
## [736] 18.239451564  4.729154696  3.957031876  2.204840505  0.676615962
## [741]  0.751416046 18.789845072 16.412318387 21.244828344  3.064142836
## [746]  7.753389370  1.379997529 56.539254202 14.969003444  7.072251702
## [751]  8.516895714  2.851325483  7.389995740  2.392966642  5.215907059
## [756] 17.927395839  4.159382745  4.602348260  1.767050089  5.999351319
## [761]  1.494464475  8.278099775  0.470182059  1.699688973 12.420656502
## [766]  2.793670191  2.859711433  8.835400135 18.674905150  6.904820462
## [771]  0.136086683  5.631612557  0.328864965  2.295110653 10.668578735
## [776]  3.250944619 25.185237388  7.668393907 25.238473642 15.866483178
## [781]  3.964860450  8.002749421  0.202607836  2.152369175 24.251437160
## [786] 33.469632047 15.547118767 30.135348351  6.071426182  4.353854672
## [791]  0.250197023 29.325677191  0.801942927 17.042884593  0.047759071
## [796]  5.333763280 15.469670902 19.642481780 20.318074365 34.815940746
## [801] 14.803005504 19.537147797 31.391648564  0.761949687 12.576822766
## [806] 13.090797529  2.906695306  8.347028354  0.763310264 14.675402132
## [811] 15.035457335  7.564079104 20.567787611 10.900908471  4.622755821
## [816]  2.230618072  1.848020579  6.003970461 10.908418336  5.326497078
## [821]  6.895777024 11.223265882 14.855356470 13.250308843  9.883816684
## [826]  0.067120590  9.900308273 16.764802822  2.052929180  7.237515916
## [831] 14.519196570  4.919077714 20.118528560  6.380759577 35.186286534
## [836] 19.419928368  0.128474154 19.063204103  6.351002606 13.316624742
## [841] 18.013500207 13.093888162  4.829880777  1.040993553 12.575400246
## [846] 10.270904528  0.846597795  1.407844850  3.815082396  2.453619097
## [851]  7.416492562 16.583949425  5.212988153  8.127296754 24.052004099
## [856]  5.930358380  1.813917118  0.405633503  7.600240987 11.929528882
## [861]  3.633230814  7.073585632  5.425096847  3.131690679 24.057598868
## [866]  4.715938910  5.191236276  1.714378146  8.665321638  5.739076155
## [871] 30.644976579  5.661064559  1.031549106 25.007642510 41.415892201
## [876]  6.362428891  8.043752174 14.599499494 32.517355273 31.141569002
## [881] 16.783691147  3.471942454 14.971163382 20.871189738  1.195393168
## [886]  8.286373843  1.775025865  9.148015362  7.212375775  3.052472253
## [891] 11.153319704 11.323216166  3.260001345  2.308934645 41.768435941
## [896]  1.204490790  3.583389390  3.042599329  7.614974666  8.488283798
## [901]  1.195365176  8.900091805  1.061208328  7.583886626  4.073377699
## [906] 14.393823696  1.085738527 23.103589048  8.165882233  2.556005292
```

```
## [911]  7.338248650  8.669226331  9.091206719  0.307760144  9.143682674
## [916]  2.515271295 46.360019384  1.379927491  6.739727464 14.079180951
## [921]  0.133103022  9.839043562 59.257409618  0.042195465 18.759170834
## [926]  2.695275929  6.372838532  6.664956057 34.396204802  0.278933933
## [931]  6.050046209 45.952327575  4.898276864  3.331589514 12.134577246
## [936]  4.164932717  9.503579337  3.681650785 24.202096074 18.471911131
## [941]  1.392388208  2.532910702 18.631242182  1.287793140 11.526311819
## [946] 22.571968254  1.564529173 13.488456325  2.239354260  7.754162272
## [951]  2.651598190 25.570193493 11.130125590  2.683266918  3.305233277
## [956]  2.173945237 10.141661381  2.539635451  2.450116800  1.767818822
## [961]  0.350264713 12.311327806  4.029096568  3.438877028  4.331365760
## [966]  6.376405789  3.915757586  1.861003405 17.948531356  3.991624317
## [971]  6.927337928  7.533128629  8.223574955 11.496202384  3.618063206
## [976]  4.344406860  2.824716751  7.686384836 11.226312077  9.265261430
## [981]  6.948947605  1.704727829 20.232785520  4.290659605 16.348204933
## [986] 17.130673662 14.163644929 10.826784308  2.944064554 23.700221627
## [991]  7.138102620  0.488572645 20.512730387  0.462972568  2.396557445
## [996]  6.116343833  6.825955448 21.109771319  4.653529499  1.751929150
```

Creating the histogram of the generated values:

```r
hist(values, col = 'black', main = 'Histogram of generated Values for m-value
of 10', xlab = 'Random Value simulated', ylab = 'Frequency', breaks= 20,
border='white')
```



Calculation for mean and standard deviation of the generated values:

```
Mean_value= mean(values)
sd_of_Values= sd(values)

Mean_value

## [1] 9.725762

sd_of_Values

## [1] 9.607643
```

(c) Repeat the simulation in part (b) with m = 1 and m = 1000. Discuss your observations of the distribution of the simulated values. ANSWER:-

```
# Repeting the simulation part for the m-value of 1.
m=1                          # putting the m value as fixed 1
N=1000                        # Because I have to repeat it for 1000 times so
putting simulation value to 1000.



simulated_values_for_m1= numeric(N)
for (i in 1:N){                   # Repeating the gen_exp(m) function 1000
times for the m-value=10
simulated_values_for_m1[i]= gen_exp(m)
}
```

looking for the random values generated(simulated) when m=1 by repeating 1000 times'

```
simulated_values_for_m1

##    [1] 5.881529e-01 1.048526e+00 9.875806e-01 5.909503e-01 1.693203e-01
##    [6] 2.545719e-01 4.494942e-01 8.533817e-01 2.398883e-01 3.339727e-01
##   [11] 1.213876e+00 6.412324e-02 1.794659e-01 2.371807e-01 1.111195e+00
##   [16] 1.971116e+00 1.488340e+00 1.079763e+00 1.118482e+00 1.351566e+00
##   [21] 8.535421e-01 1.217947e+00 1.828123e-01 2.805271e+00 1.200425e+00
##   [26] 3.082463e+00 7.642576e-01 6.242894e-01 3.014740e-01 2.234575e-01
##   [31] 1.110883e+00 4.093906e-01 1.438774e-01 3.613888e-01 1.554036e+00
##   [36] 2.353986e+00 5.851905e-01 1.427095e+00 2.612182e-01 4.068704e+00
##   [41] 3.415566e-01 1.203139e+00 1.606509e+00 1.026719e-01 6.135823e-01
##   [46] 1.759617e+00 1.368907e-01 1.081757e-01 1.126321e+00 3.719115e-01
##   [51] 2.431011e-01 9.669344e-01 2.579376e+00 1.397042e-04 4.163423e-01
##   [56] 8.258738e-01 1.177627e+00 1.728230e+00 1.502665e+00 9.545448e-01
##   [61] 2.213866e+00 7.245964e-02 1.267484e+00 8.537153e-01 3.076002e+00
##   [66] 1.111061e+00 1.165499e+00 1.770397e+00 9.792967e-01 1.379339e+00
##   [71] 1.936882e-01 1.784960e+00 3.293687e-02 1.919180e+00 5.799400e-01
##   [76] 2.219206e-01 1.638741e+00 2.571502e+00 5.111019e-01 3.180963e-01
##   [81] 6.434062e-02 1.879524e+00 2.092141e-01 1.637006e+00 1.375078e+00
##   [86] 2.175812e+00 3.084184e-03 7.707730e-01 1.581533e+00 1.558310e+00
##   [91] 6.422433e-01 8.201223e-01 2.212769e-01 3.231255e+00 2.171335e+00
##   [96] 7.678543e-01 2.959148e-01 1.367156e-01 4.396439e-01 2.260183e+00
##  [101] 6.782811e-01 1.807030e-01 6.823694e-01 1.539562e+00 3.825735e-01
##  [106] 1.106858e+00 1.173925e+00 2.913313e+00 1.547643e+00 5.576169e-01
```

```
##  [111] 8.956752e-01 4.202674e-01 3.403712e+00 6.442085e-01 3.485689e+00
##  [116] 9.954718e-02 1.531212e+00 8.031694e-01 8.302761e-02 1.346431e+00
##  [121] 4.163297e+00 5.070688e-01 6.383519e-01 5.036597e-01 4.985297e-01
##  [126] 1.294184e-01 5.769959e-02 1.535003e-01 4.285492e-01 5.340333e-01
##  [131] 3.850785e+00 1.120694e+00 5.143382e-01 1.027655e-01 5.722159e-01
##  [136] 2.587688e+00 1.061830e+00 2.690535e+00 6.071151e-01 8.639397e-01
##  [141] 9.057496e-01 4.939033e+00 1.994287e+00 1.065403e+00 2.307555e-01
##  [146] 3.772126e-01 3.950047e-01 3.602737e-01 9.931285e-01 1.364150e+00
##  [151] 4.713723e-01 1.925527e-01 2.806038e+00 1.912326e-01 1.704300e-01
##  [156] 1.868124e+00 6.208219e-01 7.707612e-01 7.794761e-01 5.463169e-03
##  [161] 2.259734e+00 6.624577e-01 2.345677e+00 1.566941e+00 3.876117e-01
##  [166] 5.741139e-01 9.590702e-01 3.166321e+00 1.328559e-01 9.980713e-01
##  [171] 2.479781e+00 2.962155e+00 1.403551e+00 3.125425e-01 6.460013e-01
##  [176] 9.208559e-01 1.022260e+00 8.732389e-01 1.035571e+00 7.069728e-01
##  [181] 5.400534e+00 2.687930e-01 9.995796e-01 3.838433e-01 1.706312e+00
##  [186] 6.999574e-01 1.178220e-01 2.294926e+00 9.807866e-02 2.088608e-01
##  [191] 2.734764e-01 2.340348e+00 2.372376e-01 1.446206e+00 1.911710e+00
##  [196] 2.701064e+00 1.460315e-01 1.688356e+00 6.995148e-01 2.410611e-01
##  [201] 6.495371e-01 1.088180e-01 1.530512e-02 4.212473e+00 8.296234e-01
##  [206] 2.466133e-01 1.733191e-01 2.730947e-01 9.158061e-01 2.385605e-01
##  [211] 7.228622e-01 1.117013e+00 7.153700e-01 8.923958e-01 5.552403e-01
##  [216] 3.105243e+00 7.741411e-01 6.204057e-01 4.692690e-01 7.779935e+00
##  [221] 4.012280e+00 3.200875e+00 1.528127e+00 2.866845e-01 6.820192e-01
##  [226] 3.062499e+00 1.888493e-01 3.919199e-01 2.520260e-01 2.809649e-01
##  [231] 9.516715e-01 8.791300e-01 5.111331e-01 8.069855e-01 1.352184e+00
##  [236] 1.007066e+00 4.280459e-01 4.532558e-01 1.084160e+00 7.081134e-01
##  [241] 2.428864e+00 1.469407e+00 4.135271e-01 7.939198e-01 5.932512e-01
##  [246] 7.993236e-01 7.287667e-04 1.744848e+00 6.124653e-01 8.073167e-01
##  [251] 1.346408e-01 1.684361e+00 7.287069e-01 1.823800e+00 2.328850e-01
##  [256] 6.257957e-01 1.104358e+00 1.302205e-01 1.059869e-01 4.265368e-01
##  [261] 2.346139e+00 8.731399e-01 3.943158e-01 9.520274e-01 2.219827e-01
##  [266] 1.847526e-01 4.886461e-01 1.795652e-01 8.138903e-01 8.574603e-01
##  [271] 3.258242e+00 4.170847e-01 5.711907e-01 1.425671e+00 5.920035e-01
##  [276] 2.040453e+00 1.357321e+00 6.620485e-01 6.510344e-01 3.487826e+00
##  [281] 2.372048e-01 6.464412e-01 3.703512e-01 1.509470e-02 4.457098e-03
##  [286] 5.385572e-01 2.238929e+00 2.426580e+00 1.594685e+00 1.657470e+00
##  [291] 7.545425e-01 4.107340e-01 6.295888e-01 7.882235e-01 7.020024e-01
##  [296] 1.751254e+00 4.729410e+00 3.754804e+00 1.566222e+00 3.298552e-01
##  [301] 6.602024e-01 3.723507e-01 6.827578e-01 4.250561e+00 6.454372e-01
##  [306] 1.746924e-01 4.025893e-01 2.597842e-01 1.257200e+00 4.607307e-01
##  [311] 4.688761e-01 1.008933e+00 2.770729e-01 2.050509e-01 6.301726e-01
##  [316] 1.279887e-01 1.310224e+00 1.370718e+00 1.975436e+00 1.685359e+00
##  [321] 5.455266e-01 1.124585e-02 4.660597e-01 8.861303e-01 4.006527e-01
##  [326] 6.094610e-01 1.121480e-01 2.400692e-01 1.333501e+00 7.578265e-01
##  [331] 2.486521e-01 1.247616e+00 1.107544e+00 2.408472e-01 1.136830e+00
##  [336] 1.399015e-01 1.751662e-01 3.628899e-01 1.388662e+00 9.554050e-01
##  [341] 5.405611e-01 4.237973e-01 1.506487e-01 1.453543e+00 1.975088e-01
##  [346] 4.144181e-01 5.946703e-02 2.574739e-02 2.197388e+00 1.510749e-01
##  [351] 2.691419e+00 1.768321e+00 2.014966e-01 1.916494e-01 3.449579e-02
##  [356] 3.857929e-01 2.799500e-01 1.575036e-02 1.783064e+00 2.948636e+00
```

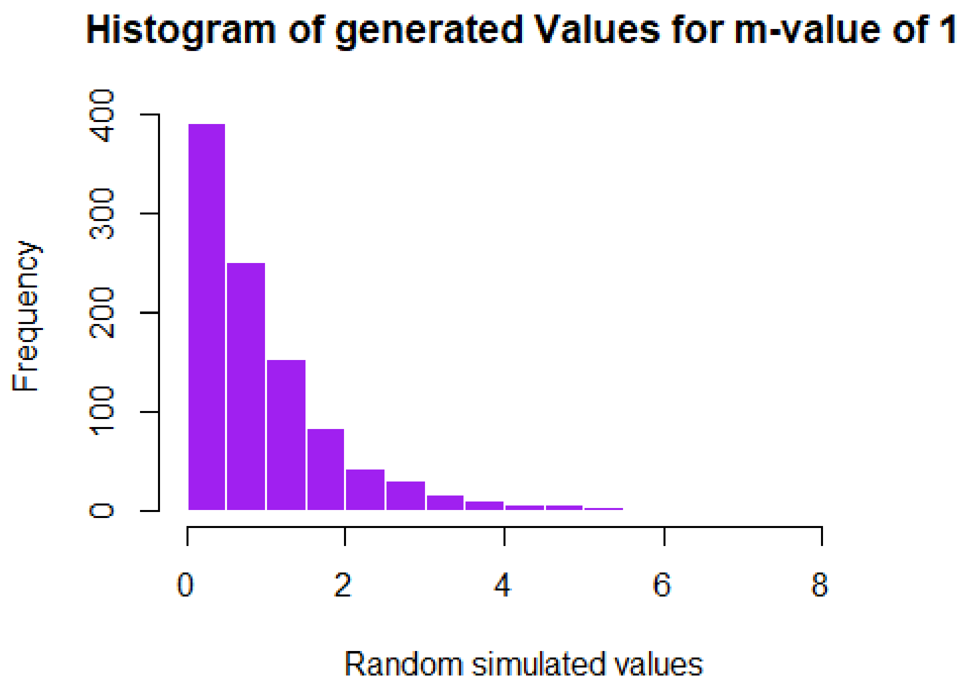
```
## [361] 1.178936e+00 1.685260e+00 7.366367e-01 4.658462e-01 1.735805e-01
## [366] 6.202622e-02 5.496611e-03 1.958449e-01 4.136071e-01 1.568723e-03
## [371] 1.752767e+00 9.215116e-01 1.102714e+00 1.466374e+00 7.392825e-01
## [376] 2.462786e+00 6.300624e-01 1.216836e+00 1.336338e-01 1.142165e+00
## [381] 1.120459e+00 1.228362e+00 1.222377e+00 3.369169e-01 1.766563e+00
## [386] 4.288832e-01 9.601246e-01 4.671539e-01 1.508816e+00 2.845786e+00
## [391] 1.681733e-03 9.265997e-01 2.799461e-01 9.360326e-01 9.071017e-01
## [396] 2.467087e-03 5.838353e-01 9.960536e-01 1.365569e+00 6.015079e-01
## [401] 1.387616e+00 8.586079e-02 9.147820e-02 2.053988e-01 8.000832e-01
## [406] 5.987538e-01 1.862367e+00 5.230664e-01 1.688436e-01 1.618481e+00
## [411] 5.782920e-01 7.645135e-01 2.008030e+00 7.304102e-01 1.286168e-01
## [416] 1.934171e-01 6.678565e-01 1.747086e-01 9.877508e-02 1.478871e-01
## [421] 5.642232e-01 4.219473e-01 6.503328e-01 2.686914e-01 1.216955e+00
## [426] 2.326800e-01 4.363742e-02 1.560154e+00 7.506242e-01 2.785296e-01
## [431] 3.619043e-01 1.456474e-01 1.650011e+00 1.099419e+00 7.001148e-01
## [436] 9.607163e-01 3.487715e-01 2.238528e-01 7.739768e-01 4.508854e-01
## [441] 1.076922e+00 1.170313e+00 1.497176e+00 1.818730e-01 4.686993e-01
## [446] 8.310279e-01 1.511084e+00 6.704785e-02 1.559092e+00 1.479667e+00
## [451] 1.440385e+00 4.071850e-01 1.525873e+00 7.933491e-01 3.128743e-01
## [456] 1.067570e-01 5.247465e-02 1.653197e+00 8.486907e-01 8.050612e-01
## [461] 4.686680e-01 6.815314e-01 2.965251e+00 1.038341e+00 9.954258e-06
## [466] 1.899006e+00 3.393373e-01 5.813689e-01 5.882030e-01 9.030371e-01
## [471] 2.277196e+00 6.068466e-01 3.823301e-01 5.580176e-01 4.787385e-01
## [476] 4.394355e-01 3.968172e-01 2.386364e-01 3.506593e-01 2.222762e-01
## [481] 3.641952e+00 3.714878e-01 1.083044e-01 9.259105e-01 3.035930e-01
## [486] 2.298546e+00 2.306449e+00 1.182430e+00 1.266371e+00 7.960988e-02
## [491] 2.219705e+00 1.644712e+00 2.625741e+00 5.033040e-02 2.198506e+00
## [496] 3.297643e-02 7.187916e-01 4.636829e-01 8.733290e-01 1.029754e+00
## [501] 3.377229e-02 5.305321e+00 6.882874e-01 3.494491e-01 4.353230e-01
## [506] 6.932489e-01 5.396774e-01 5.145267e-01 5.944637e-02 1.314643e+00
## [511] 1.129795e+00 1.208875e+00 2.912212e+00 1.096173e+00 6.956107e-01
## [516] 1.490434e+00 2.206136e+00 1.890679e+00 1.205351e+00 9.399736e-02
## [521] 6.437774e-02 3.550464e-01 3.740045e-01 8.324330e-02 1.410260e-02
## [526] 3.132345e+00 6.703678e-01 2.148965e+00 5.580841e-01 2.359286e-01
## [531] 7.322849e-01 3.882294e-01 8.252783e-01 1.363950e+00 2.292506e+00
## [536] 4.663336e-01 1.807487e-01 4.535214e-02 5.182868e-01 1.162532e+00
## [541] 2.065885e+00 3.584017e-01 2.020473e-01 6.147672e-01 6.074183e-01
## [546] 7.606685e-01 1.300580e-01 4.326847e-01 4.661584e-01 2.635548e-01
## [551] 1.417154e+00 1.745986e-01 1.402380e+00 7.082203e-01 5.644267e-02
## [556] 1.908167e+00 6.078408e+00 1.539838e-01 4.021825e-01 7.689580e-01
## [561] 2.915275e+00 1.445802e+00 1.134644e+00 1.822853e+00 4.273029e+00
## [566] 1.576006e+00 1.563677e-03 1.033326e+00 5.315641e-01 6.182638e-01
## [571] 3.192719e-01 5.199401e-01 4.901044e-01 2.083717e-01 1.698425e+00
## [576] 1.871202e+00 5.269151e-01 3.996969e-01 2.245128e+00 3.160676e-02
## [581] 5.146539e-01 2.128722e-01 1.596348e-01 1.345020e+00 1.713879e+00
## [586] 1.005155e+00 1.640539e-01 4.715602e-01 9.091027e-01 7.788322e-01
## [591] 7.822736e-01 6.464305e-01 5.066093e-01 7.372822e-01 1.669204e+00
## [596] 2.048778e+00 1.890592e-01 2.451288e+00 9.104997e-03 1.549748e-02
## [601] 1.356512e-01 2.796421e-01 2.802912e-01 8.169793e-02 7.056713e-01
## [606] 8.427640e-01 6.588534e-01 2.494286e-01 2.686718e+00 8.781727e-02
```

```
## [611] 5.258896e-01 3.102278e-01 2.693050e-01 5.840745e-01 1.396127e+00
## [616] 3.563910e-01 1.785316e+00 7.826854e-01 3.404027e-01 5.281709e-01
## [621] 2.835980e-02 3.581921e-01 4.392022e-01 3.898062e-01 1.988934e+00
## [626] 8.563304e-01 2.285630e-02 3.346895e+00 1.380137e+00 1.445627e+00
## [631] 4.060044e-02 6.907009e-01 8.657680e-01 1.484741e+00 3.915950e+00
## [636] 2.354858e+00 1.020283e+00 1.582458e+00 8.572048e-04 7.326552e-01
## [641] 7.935182e-01 9.018710e-01 3.828692e+00 1.903850e+00 2.000491e+00
## [646] 2.677242e+00 9.577650e-01 6.494308e-01 6.120993e-01 1.124998e+00
## [651] 1.604570e-01 8.396311e-02 4.832780e-01 6.062941e-01 1.194647e+00
## [656] 2.355966e-01 2.853857e+00 1.611775e-02 5.368339e-02 8.776619e-01
## [661] 1.306742e-01 2.907036e-01 3.328449e-01 1.286148e+00 1.484463e+00
## [666] 1.138406e-01 3.849613e+00 1.859237e+00 1.361766e+00 1.087694e+00
## [671] 4.576511e-01 1.631270e-02 1.567360e+00 8.762398e-02 3.655617e+00
## [676] 3.753533e-01 1.334053e+00 5.534245e-01 6.209643e-01 9.539473e-01
## [681] 1.708741e+00 3.713476e-01 7.457703e-01 1.297619e+00 7.443193e-01
## [686] 6.166492e-01 5.247329e-01 5.337257e-03 2.458427e-01 7.347137e-01
## [691] 1.759705e+00 3.334140e-01 1.002950e+00 6.862512e-02 6.689885e-01
## [696] 1.439675e+00 3.330864e+00 1.106357e+00 4.532558e-02 4.450512e-01
## [701] 5.635213e-01 7.951868e-01 1.128843e+00 2.394134e-01 2.439793e-01
## [706] 5.087947e-01 1.647017e-01 1.246961e+00 1.538536e-01 8.566666e-01
## [711] 2.094205e-01 2.969681e-01 4.688780e-01 1.800461e+00 1.083645e+00
## [716] 1.823766e+00 1.440762e+00 3.453758e-01 1.908206e+00 3.254272e-02
## [721] 3.301006e-01 1.188791e+00 4.692096e-01 6.960231e-02 6.996613e-01
## [726] 1.485137e+00 5.931126e-01 3.472396e+00 8.003229e-01 5.138963e-01
## [731] 9.532665e-01 7.626925e-01 1.399591e+00 1.870379e+00 2.555863e+00
## [736] 2.774309e+00 3.421869e-01 1.115804e+00 1.218410e+00 3.278831e-01
## [741] 1.407290e+00 6.038716e-01 1.205902e-01 1.019321e+00 7.640794e-01
## [746] 2.707858e+00 3.937221e+00 1.048407e+00 7.175983e-01 2.097018e-01
## [751] 1.353554e+00 1.883990e-01 4.881219e-01 2.250831e-01 5.844233e-01
## [756] 6.206167e-01 2.611753e-01 1.050103e+00 3.036298e-01 4.472852e-01
## [761] 1.194847e+00 8.001303e-01 1.040486e-01 3.975500e-01 3.323543e-01
## [766] 1.215303e+00 1.046098e+00 6.591566e+00 2.131535e+00 3.356277e-01
## [771] 4.828496e-01 2.823603e-02 1.051255e+00 3.930099e+00 1.758956e-01
## [776] 2.526527e+00 4.274612e-01 3.286378e+00 7.285390e-01 6.477669e-03
## [781] 1.700328e-01 1.637709e+00 2.036916e+00 1.004840e+00 1.174220e+00
## [786] 7.128172e-01 1.309300e+00 4.871159e-01 3.174719e-01 1.206676e-01
## [791] 2.674019e-01 8.359112e-02 1.566052e-01 9.744593e-01 3.942988e+00
## [796] 1.204782e+00 7.435280e-01 1.042764e-01 2.413223e-01 1.518351e+00
## [801] 2.428326e-01 2.862880e+00 2.673195e-01 1.086807e+00 5.565564e-01
## [806] 9.670681e-01 8.161862e-01 4.680897e-02 1.523538e+00 7.849837e-01
## [811] 3.425788e-01 6.166676e-01 8.529654e-01 1.126012e+00 1.465802e+00
## [816] 8.162055e-01 1.557528e-01 6.291584e-02 2.009433e+00 9.104110e-01
## [821] 1.405636e-01 6.094413e-01 2.481973e+00 1.381762e+00 3.629147e+00
## [826] 1.062472e+00 2.736611e-02 7.851478e-01 5.174824e+00 2.003723e-01
## [831] 1.182823e+00 1.840956e+00 5.085483e-01 3.806955e-01 2.172669e+00
## [836] 1.201357e+00 4.460708e-01 3.049934e-01 1.111126e+00 3.191270e-01
## [841] 4.644001e-01 1.974390e-01 9.308604e-01 9.335753e-01 8.108922e-01
## [846] 2.123254e-01 5.358709e-01 4.037588e-01 4.722693e-01 6.371426e-01
## [851] 9.822206e-01 3.072545e-02 7.901527e-01 9.562779e-01 3.600206e-02
## [856] 3.271870e-01 1.200555e+00 6.252242e-02 4.730034e+00 2.979690e-01
```

```
## [861] 1.293337e-01 2.360715e-01 1.461711e+00 4.705328e-02 1.743157e-01
## [866] 1.183707e+00 1.450363e+00 1.414167e+00 3.445928e-01 3.177004e-02
## [871] 1.569828e+00 1.205547e-01 9.149085e-02 4.592766e-01 2.511020e-01
## [876] 1.143223e+00 3.608357e-01 1.208492e+00 1.562718e-01 3.231094e+00
## [881] 2.373328e+00 1.415365e+00 5.547705e-01 1.521298e-01 1.524976e+00
## [886] 1.734108e+00 5.581575e-01 2.450859e+00 2.704275e-01 1.204755e-01
## [891] 3.067365e-01 1.345335e+00 3.041968e-01 5.039093e-01 7.397962e-02
## [896] 1.287448e-02 2.736888e-01 2.206196e+00 2.086903e+00 1.983280e+00
## [901] 4.166808e-01 1.803098e-01 1.198319e-01 1.865481e+00 2.937005e-01
## [906] 3.363227e-01 1.112034e+00 2.545388e+00 4.624513e+00 1.678324e+00
## [911] 8.192949e-01 2.644394e+00 2.623750e+00 6.208238e-01 6.198848e-01
## [916] 2.581162e+00 4.015140e-01 4.870963e-01 8.869868e-01 2.694958e+00
## [921] 1.003276e+00 1.921925e+00 3.915957e-01 9.286440e-01 5.507506e-01
## [926] 2.842061e-01 7.691135e-02 4.675673e+00 9.610279e-02 4.850853e-01
## [931] 2.488235e-01 1.923609e+00 4.704567e-01 2.740310e-02 5.955943e-01
## [936] 9.901202e-01 5.871721e-01 6.567009e-01 5.661722e-01 8.325351e-01
## [941] 3.161087e-01 1.284707e-01 3.358098e-01 8.938830e-02 4.555456e-02
## [946] 4.634741e-02 1.139499e+00 2.270073e-02 8.828897e-01 1.235106e+00
## [951] 1.177909e+00 8.357597e-01 2.930863e+00 2.939936e-01 2.164747e+00
## [956] 1.730744e-01 3.491271e-01 1.109339e+00 3.544880e-01 1.141113e+00
## [961] 4.794493e-01 6.108858e-01 8.355171e-01 1.274976e+00 1.657255e+00
## [966] 2.779083e-01 2.438709e-01 7.132824e-02 2.577029e-02 1.395675e+00
## [971] 1.350035e+00 4.019852e-01 3.537371e-01 1.859245e+00 1.852651e-02
## [976] 1.026168e+00 1.980288e-01 8.123477e-02 1.556773e-01 3.447332e-01
## [981] 6.939724e-01 1.927130e+00 7.309318e-01 7.425249e-01 5.008195e-01
## [986] 8.263641e+00 4.922814e-01 1.496668e+00 1.556913e-01 2.644975e+00
## [991] 1.275330e+00 8.538783e-01 1.829645e-01 4.648531e-01 1.248453e+00
## [996] 6.063505e-03 1.966658e-01 5.476075e-01 5.887257e-01 3.127135e-01
```

looking at the distribution of the generated(simulated) values for m=1.

```
hist(simulated_values_for_m1, col = 'purple', main = 'Histogram of generated Values for m-value of 1',
     xlab = 'Random simulated values', ylab = 'Frequency', border='white', breaks = 20)
```



Clearly, from the graph we can see that the graph of the random values generated by the gen_exp() function for m value of 1 is Right skewed.

similarly lets look for the m value of 1000.

```
# Repeting the simulation part for the m-value of 1.
m=1000                              # putting the m value as fixed 1
N=1000                            # Because I have to repeat it for 1000 times so
putting simulation value to 1000.


simulated_values_for_m1000= numeric(N)
for (i in 1:N){                      # Repeating the gen_exp(m) function 1000
times for the m-value=10
simulated_values_for_m1000[i]= gen_exp(m)
}
```

looking for the random values generated(simulated) when m=1 by repeating 1000 times'

```
simulated_values_for_m1000

##    [1] 1479.4308227   11.2865593 1000.0421018  965.4725081 1083.4304997
##    [6]   50.5125410 3623.3435844  253.0280002   98.1727717  793.6765435
##   [11] 1323.7907871 2537.8420397 1609.3705279  347.1285028 1627.0874809
##   [16]   61.2662158   46.1066879  375.1834377  603.3496333  750.5026146
##   [21] 1631.5087293  281.7177457  358.9198765  593.9178738  622.0061639
##   [26] 1007.4769697  479.7520505  325.6051700 2487.3065485 2057.1269118
##   [31]  364.1538712 1278.9142517  819.5886454 1610.1088844 1305.5989501
```

```
##  [36] 1065.8371087   25.2587089 1171.7314073  571.6282480 1017.2599512
##  [41]  347.6693713  761.4940631  860.8089130  494.9535719  671.9147316
##  [46]  365.4541323  104.2213257  344.9968708  659.1786852  333.1833180
##  [51] 4857.0198579 3246.0534067 3617.3575025 1085.9690367  444.6320058
##  [56] 1406.1004716  205.1734320 1580.7503055  638.9840260 2518.7650721
##  [61]  374.2626257  176.8347882   99.0077338 1212.4573216  218.4272792
##  [66] 2033.8804491  491.0429601  423.9090913  254.1625595  979.6925998
##  [71]  393.9697461 2813.5527099 5919.1342352    8.1308927   24.7067158
##  [76]  411.6619033   31.4825810   54.8259386 3495.0812021 1067.4868902
##  [81]   12.2530397  226.1502512 5298.0433848   65.6186261  922.3689431
##  [86]  394.2357515 1192.1115442   38.3407929 1643.0240110  141.8683960
##  [91]  143.3392052 3182.1001589  886.4359377  222.8003145  521.9517385
##  [96] 1000.2355085  690.3597468 4970.2764157  367.9056360   95.6957013
## [101]  848.5745692 1764.1848663 1536.8779693   26.4780607  996.4016845
## [106]  187.7298677 1606.0570323  311.5367935  139.8225754 1915.5968980
## [111] 3074.3717881 1854.7284288 2370.9351876  420.0748612 2182.9230877
## [116]  505.0404198  504.9405961  524.4578050  612.9571444 1906.7610435
## [121]  132.3451578  718.3664274 1594.2342014  840.5335776   44.2099272
## [126]  103.4785606  542.8608323  279.9186422 5555.3184461  587.0491812
## [131]  275.2131779  522.2768004 1039.2886792  651.3978273  231.0020138
## [136]  694.6407202 1348.0464450 1726.7709340  146.6023102   54.9544004
## [141] 3833.3823663   76.0463095 1024.8353036 2349.2530776  226.8627018
## [146]  904.0640795 2031.1010477 1229.3455165 6450.8712946  338.5914854
## [151] 1258.0552867   11.4450278  228.2993389 3017.1726498  590.8179558
## [156]  271.4619622  132.7121657  494.1123360  873.9291442 1734.3614214
## [161] 3122.9754614 1177.2622061 1129.8897817 2533.2357460   84.6937205
## [166]  193.7183491  821.0805617 1312.1731965  308.2374355 1091.5706503
## [171] 2414.9870699  226.5404494  209.0135577 1582.5567246  341.0944926
## [176]  186.0356718  904.7830383 1249.6968955  815.2073501  218.7740052
## [181] 2635.5768399 2082.3662838   61.4511384  551.7670111 6046.9372538
## [186]  600.2001308  354.7143796 1496.0218023  261.4795998  890.5439241
## [191]  278.9625985  346.7166749  207.1865706 1698.8615263 2775.9010079
## [196] 1142.6284873  997.7953051 1055.2060300  107.6051295  292.9279141
## [201]  188.2556697  156.0029702  224.7374191  284.7449981  639.1019546
## [206]  114.3561940 1810.4393869 1005.1705341 1685.5910643 1149.7258302
## [211]  376.8385446 2005.9982911  147.0940284  683.8514612  642.8376432
## [216]  277.9383446  487.6670727  812.5920727  435.7530589  899.8552972
## [221]  278.7809034  244.2834764  469.8059536 1198.5734804   73.8676986
## [226]  842.4290727 3268.8667181  805.9306157 3637.4313584  608.8823611
## [231]  268.5058088   44.3670483 1104.2141111  353.1761044  836.7231830
## [236]   26.0183438 4485.1816064 1833.0289645  386.9965460 1519.4627933
## [241] 1574.6478671 1212.9338643 3120.7804967  208.1337955  188.9609917
## [246]  275.6068467  367.4453025  376.2579623 1219.9200663 2712.4272060
## [251] 2556.0150025 1732.2820264  999.5837710 1074.3576197  678.2093453
## [256]  750.7834000  166.0372958 1423.3097384    2.1895106  835.7049668
## [261] 1034.0049951  134.6747160 1569.8281339  893.4982037  697.5481951
## [266]  568.8844271 1169.5201685  577.6295667 1018.8116380 1718.3074411
## [271] 1563.3460132 1110.5716698  661.4963900  677.0477726 1490.5885499
## [276]  413.4077777  253.8807686  556.0568681  311.8076869  142.7410880
## [281] 3286.6404464  617.0791542 1006.5130053   24.2659486 3642.2268963
```

```
## [286] 2349.5211457  116.8636149  771.4200131  528.2294237  220.8820779
## [291] 1097.3702004  113.0215898 1339.2076187   41.6114079  409.7689108
## [296]  274.0428590   81.1991852  690.1828258  219.6223873  758.8937540
## [301] 2170.7217763 1151.8124146  129.3921380  732.5378719 1392.8663237
## [306] 5916.7599971 1286.6211313 2487.7311883 1853.4575233 1006.0653096
## [311] 2354.7139050 2093.0550800  735.8169512  620.2364406 1298.7028851
## [316]  200.3049965  382.2054004   78.0155737    0.5900763  430.1433717
## [321] 1017.1064427 1601.3674063  978.9317276  667.0821202  868.6032284
## [326]  977.6280838 1227.6953121 1224.0842415  355.1438390 2143.8950245
## [331]  385.5119902 1421.2708450   35.0448787  100.9775546 1110.3485857
## [336]  475.8802715  591.4522644  471.6936166 1177.6549035 3224.5546678
## [341] 2382.8453291 1306.0674115 1272.1129811 8656.3941218  915.4881006
## [346]  328.5687711  831.1342902  825.1636255 1256.3112671  197.8110170
## [351]  567.8094048  509.7591584  356.1845357 2249.9543547  107.1240447
## [356]  846.5703671   79.1124533 1392.8172065  956.4204166  216.0375178
## [361]  285.0667944  126.0078190  133.7935802 2053.1984679 2336.0324726
## [366] 1781.1562236  729.0478543  919.0065216 3996.7948991   45.6154566
## [371]  695.9591514 1774.2262382   29.3407510   54.8285332 1044.2407155
## [376] 3170.8062397  343.8323632  656.9672835 2283.8953057  418.3656305
## [381] 2524.1374299 1057.5230973    9.5413577 2330.6972215  287.0085818
## [386] 1258.4572710 2995.8449969  411.3397713 2182.4159332  835.5685719
## [391]  859.7746329  144.8470016  613.9229312  857.4773369 3069.0008319
## [396]  447.8527383 1858.4617117 1536.2580295  766.7578409 2461.7326027
## [401]  940.3203433 1009.7307479   96.8303481  235.3398986  142.6351334
## [406]   86.2541365   50.4364459 2821.9210428   92.7080834  163.9605457
## [411] 2701.0586101  472.7473453 1520.7251483  645.6950426 2991.4667288
## [416]  241.2301365  119.9614671 2913.9241334 2446.1522242 1150.2060599
## [421] 1631.6529005  452.2819090  213.5400329 1538.6724662  359.1776158
## [426]   92.3667350  254.8469816 1244.2635043 3985.9147290  963.9040577
## [431] 1666.1401816  415.2500496  851.5867269  566.4556096  401.0629269
## [436]  290.2503741  106.8580134  269.7174666 1077.4939342 1063.0129245
## [441]  458.0780455  209.0738600 1463.1761960  327.7830158   43.2271996
## [446] 1963.3167220  390.7207582  598.8359505  641.5438002 1256.2446837
## [451] 3379.6376406   80.2861729 1724.3680896  486.7621005 1352.0184484
## [456]   75.2380076 1545.1810791  867.8430262  390.0306141  448.1931487
## [461]  278.4061262  276.7427570 1135.9118960 2552.1953674  147.2196518
## [466]  923.5467552  499.5585019 2163.5509221 2165.8251097 1415.9388218
## [471]  291.2610037 1364.5366467  302.2705951  383.5786533  655.7008535
## [476] 1704.7529862  947.0604761  903.9772538 4993.4112877  602.8655099
## [481]  133.8179314  619.4484837  672.8044147  851.2057294  692.7656798
## [486] 2108.4132255 5013.4069515  533.3666230 1210.3314910  467.9727093
## [491] 3938.1732162  185.5643583  980.3841445  116.2382117 1331.1922683
## [496]   86.7542397    5.8315465  306.5849887  918.1520438  438.2837257
## [501]  519.4941742  399.6762679 1095.8977394  118.6362072    2.2587525
## [506]  951.0657086    2.4473021  960.0541167  644.6563286 1970.6708221
## [511]   74.8782700 3611.6405983 1138.4570691  199.1109829 1350.5110482
## [516] 3764.2696728  341.0285305  174.2348841 2370.5113758 4796.1884907
## [521] 1567.0594418 1733.1868431 1458.7244144  535.9576510 1206.8373408
## [526] 2216.1585259 1048.1866483  459.7172885   13.1345761  441.3024106
## [531]  556.1950797  677.5216454  800.4094087 2651.7062650 1143.4741802
```

```
##  [536] 1176.5776435 1099.1876814   58.6306304  856.7928817  384.2614881
##  [541] 1714.8162479 1256.1997494 2131.0964201 2051.9818896 2313.1839291
##  [546] 1796.6276930  462.7733250  402.8443686 2048.9517859  343.1602245
##  [551] 1153.7371494  309.8937559  876.4978758  551.9196292 1834.2696169
##  [556]   73.6210284  763.8364214   65.1200985  334.2773273  106.2044034
##  [561]   99.3937089 1781.2335289  567.8561137 1932.8533471 1203.2788395
##  [566]  365.4264868  641.1237099  265.5778089 1068.9631404 1841.4449968
##  [571]  147.9420263 1237.0431251 2785.7533060  695.6490097 1510.3642466
##  [576]  877.7694470  649.7874487   14.0696561 2135.0293367   26.9582972
##  [581]  691.6341950  733.4972500  250.4826250  164.9612836  637.6811823
##  [586]  800.5908003  129.4488783  356.6589117  674.0321015 1150.9351253
##  [591]  257.3769051 3355.7410784  522.7916420  392.6685343  863.7993324
##  [596]  617.9635929   35.4266253  273.4931087 2426.0073783   86.9660588
##  [601] 1299.2723427  628.6807907  622.0492123   91.8777844  775.8738996
##  [606] 1694.0023771  495.2203219  253.2967110    3.2764693 2558.9342121
##  [611]  374.0401640  111.8373586  403.1062861  676.6130245  492.6185686
##  [616] 3332.3702183 3753.9621051  814.7021912  292.6361307  880.7844772
##  [621] 1248.5347480 2730.0111355  754.4839692 1501.6379059    9.1727546
##  [626]  114.9757344   81.2165323  756.8752218   50.5786728 1480.8576982
##  [631]  138.4123652  560.6276148  367.4779574  869.3928588 1788.9870420
##  [636]  264.8881921 1545.6087626   22.0814569 4127.8992691  190.8338556
##  [641] 2195.7334478  822.9766948 1144.2917305 3250.9968134 1611.4289272
##  [646] 1082.9918638  119.1750588  432.1780057  302.8870924 1314.9396761
##  [651]  180.0340370 1547.8279016 3393.2653024   76.4511230  512.3933302
##  [656]  783.4105511  193.4361141 1547.3571912   43.5629125 1508.1102362
##  [661] 1247.7028783 3915.3885825   58.8292367 1676.6235740  523.3804264
##  [666] 1508.9158608  276.0646583 2382.9417860 1287.6168940  176.4966050
##  [671] 3058.9443291 1787.4964005  195.3734938  144.8912621  548.1517591
##  [676] 4279.4604650 2153.3093726 1328.2666722 1306.3137866  416.4628065
##  [681] 3117.6713063  806.9441465 1389.3314488 3422.0648608  316.4913387
##  [686]  100.8859080  670.9240927 3884.3182842  791.9383576  429.4123401
##  [691]  441.4444768  525.1021427  507.8037755 3460.4631940  600.9064398
##  [696]  216.8334719  632.0948896 3531.7851276  141.6536148  393.9869631
##  [701]  174.9895213  726.2726475  413.8758487 1545.6492009  370.5118436
##  [706]  406.9933958   65.7548899 1777.3260397  935.5074553 1115.9823718
##  [711] 2175.4338219  224.4470646  585.1167410   19.5204676 2797.2381447
##  [716]  792.1960164  907.0253487  354.7970483  836.1931693 1594.7620553
##  [721] 2377.9397694  562.5627627 1788.5317220 1560.5636236 1683.9515757
##  [726]  206.2471208  155.0654217  519.3491628  248.3047995  566.5757097
##  [731] 2444.0087147  464.1062317   95.9854742  126.8824832   14.2238219
##  [736]  956.0661673  250.2940731  222.2843531  358.8646428  548.3652156
##  [741]   52.8250314  590.1016157 1737.4628069  202.8518540  940.1815595
##  [746]  296.0635324 2693.7060594  295.8426232  635.1602204  224.1417711
##  [751] 1553.9457731  116.8808999    5.9826050  972.8550667  382.0341057
##  [756] 1288.0464564  966.7416962  601.2994776  721.8760832 1929.4641269
##  [761] 1499.1130807  989.1170748  596.9325334 1291.1121857  741.4003356
##  [766]  298.9977512  218.9658003 2525.6500923  837.4038498  290.4322306
##  [771]  319.8089289  110.3391866  613.0951065  355.9353962  129.1446619
##  [776] 1314.0925718   79.8828085 1371.8548427  545.6314129 1145.2237318
##  [781] 3915.6714959  312.6050140 1052.4719491  712.5756395  623.2096625
```

```
##  [786]  673.4407119 1530.8024935 1555.8199419  232.7317062   77.7980254
##  [791]   17.7148311  143.2462015   83.5832594  897.9362057  582.7210246
##  [796] 3034.3456289  136.0737740 1664.5261133 1661.2818622  734.7208734
##  [801]  293.9914826  238.3088698  587.9096363   52.8516208  362.1350495
##  [806]  460.8570412  203.6470572 2380.8265609  379.7156379  108.7368726
##  [811] 2696.4143077  704.0355773  599.5409558 2171.2181046  748.1348820
##  [816]   68.2928298  800.0396867  433.0669599 1106.6683884  293.2393606
##  [821]   95.8557413 2139.6918173  803.3846011 1514.6298575 1408.9763021
##  [826] 1646.1886996  185.7457240 2235.7702254  970.8652661 1789.1499704
##  [831] 3725.0896843 1121.3563991  447.1488797 2477.4562656   86.1692223
##  [836]  130.8210470  976.4538792 1093.3835444  586.5331629 1044.1350883
##  [841]  132.9803210 3146.9207078   78.0853045 1093.0466513  187.8402767
##  [846]  749.4805811 1130.6102866  607.0143889  157.9672478  172.8840581
##  [851]  801.0990962  200.6286539  529.7535951  126.0888596  601.2718866
##  [856] 2643.3616244  576.2278878  295.5600321  166.7251554  968.4890081
##  [861] 1050.1139036 1031.8440268  946.1395626  290.8099022  443.5791052
##  [866] 1551.4291086 1053.7223926  239.8205701 3096.0863020    1.8480548
##  [871]  513.3157726  422.3249084  393.4400003  431.8113166  841.5737433
##  [876] 2861.8346212 1293.2958703 3246.1361291   24.4659455  866.1727436
##  [881]  691.5482615  690.1879437  348.2894841  978.9918606   29.8545993
##  [886] 1019.2662330  175.5841935 2855.3082890  270.1631122  402.5236903
##  [891]  311.6596846  567.4089444 1598.2694713 1000.9720662  258.4796141
##  [896]   67.4786564  598.3934107 1148.4293848  322.7884832   40.2201026
##  [901]  303.7638232 1059.5334372  640.8468823  652.3075995 1650.2932773
##  [906]  339.9635910  687.4658777 1410.9354894  680.9339061 1545.1640640
##  [911]  447.0736498 1552.9362271 1173.9191631  623.5829286  218.2907134
##  [916] 2025.5931875  981.8992723 1315.2173809  910.3541065    8.9843623
##  [921]  231.7026770   48.1993218  168.1077881 1593.3914363  653.1793928
##  [926]   85.5944824  874.0659561 2847.4791343  366.7540730 1441.3478145
##  [931] 1051.2152038 4496.4892048   64.3622199  781.7983212  571.5415597
##  [936] 2151.6429565  646.9286784  720.7509271 3702.0245853  967.4640047
##  [941] 1388.8238604  163.7491669  246.7961728  773.2475808  240.4740745
##  [946]  506.9673492  973.6395604 2100.6899919 1721.1571202   15.6728795
##  [951] 1418.5245742  442.1815697  730.4049834   13.1316077   72.9181749
##  [956]  739.6629442  436.1483430 1183.5128866 1191.2057184  269.9471375
##  [961]  223.9485594  597.6154411  679.6996268  904.0712964 1434.2262458
##  [966] 2326.4206476  479.5475718  505.3915350  112.2128076 1077.1409456
##  [971] 1557.4794099 2389.6727418 1945.1541595  221.2056689  778.3978526
##  [976] 1111.5635820 1895.4987766    6.9836988   14.8477493  118.1496269
##  [981]  161.2164151 2015.9225650  687.5991341 2085.7078367 1944.5399572
##  [986] 1942.8948856 1196.2261943 1715.7706014  616.1128513  277.9223535
##  [991]  541.7464246   98.4175754  159.0678214  266.4886080  216.8226102
##  [996]  209.5933786  787.2728964   54.3957429  326.8369943  741.0223383
```

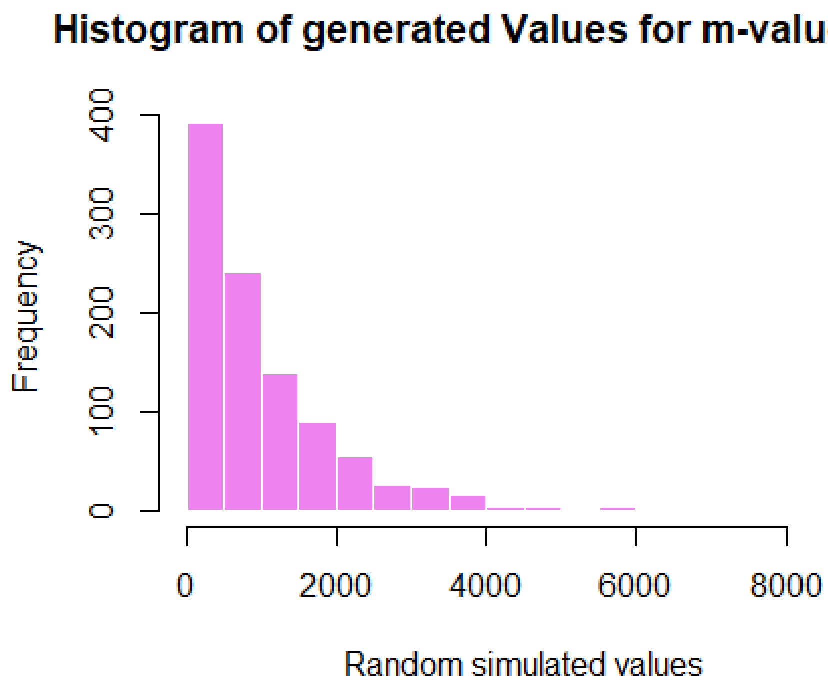
looking for the data summary of the random values generated when m=1000

```
summary(simulated_values_for_m1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## 0.00001 0.29544 0.67432 0.97665 1.34733 8.26364
```

looking at the distribution of the generated(simulated) values for m=1.

```
hist(simulated_values_for_m1000, col = 'violet', main = 'Histogram of
generated Values for m-value of 1000',
     xlab = 'Random simulated values', ylab = 'Frequency', border='white',
breaks = 30)
```



Clearly, from the graph we can see that the graph of the random values generated(simulated) by the gen_exp() function for m value of 1000 is also Right skewed.

2. About 52% of American adults are women. Their height is approximately normally distributed with a mean of 63.7 inches with a standard deviation of 2.7 inches. The average height of adult American men is 69.1 inches with a standard deviation of 2.9 inches.

(a) Write a R function that returns the mean height of a random sample of 10 American adults. You may use rnorm() function from R base package. the What is the average height of your 10 Americans?

ANSWER:- Since 52% of American adults are women, I will use 5 from male and 5 from female to make balance in sample.

```
# Define the function to generate a random sample of heights
get_mean_height= function(n, is_woman = TRUE) {
  if (is_woman) {
    mean_height= rnorm(n, mean = 63.7, sd = 2.7)
  } else {
    mean_height= rnorm(n, mean = 69.1, sd = 2.9)
  }
```

```
  return(mean_height)
}

# Get a random sample of 5 women and 5 men (as to make the sample balanced
from both male and female)
n=5
women_heights= get_mean_height(n)
men_heights= get_mean_height(n, is_woman = FALSE)

# Combine the heights of women and men into a single sample
sample_heights= c(women_heights, men_heights)

# Calculate the average height of the random sample
average_height= mean(sample_heights)
average_height

## [1] 66.14998
```

(b) Repeat this random sample 1000 times and calculate the average height for each simulation. Create a histogram of your set of 1000 average heights.

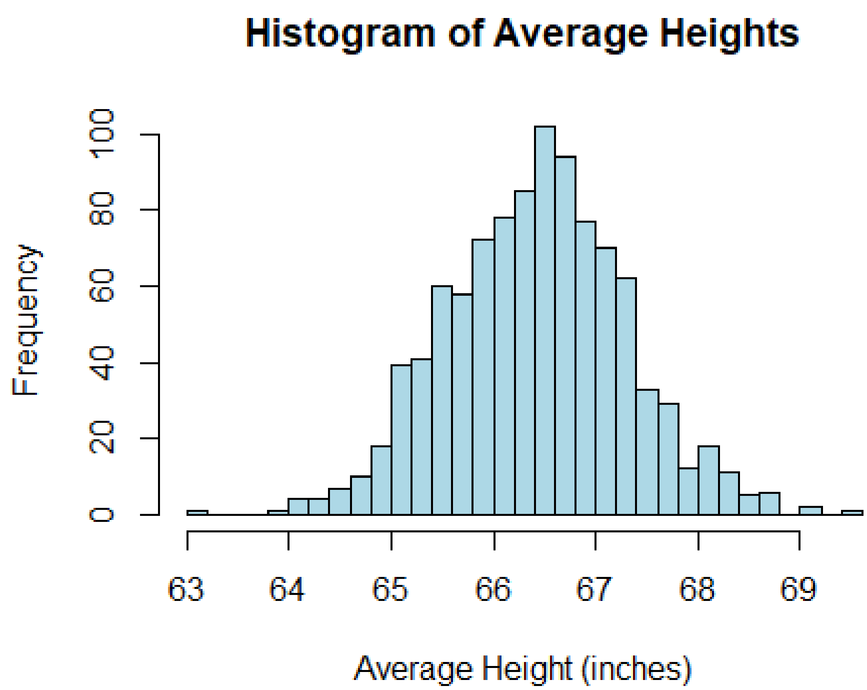
ANSWER:-

```
# Define the function to generate a random sample of heights
get_mean_height= function(n, is_woman = TRUE) {
  if (is_woman) {
    mean_height= rnorm(n, mean = 63.7, sd = 2.7)
  } else {
    mean_height= rnorm(n, mean = 69.1, sd = 2.9)
  }
  return(mean_height)
}

# Number of simulations
num_simulations= 1000

# Vector to store the average heights for each simulation
average_heights= numeric(num_simulations)

# Perform 1000 simulations
for (i in 1:num_simulations) {
  # Get a random sample of 5 women and 5 men
  n= 5
  women_heights <- get_mean_height(n)
  men_heights <- get_mean_height(n, is_woman = FALSE)

  # Combine the heights of women and men into a single sample
  sample_heights= c(women_heights, men_heights)

  # Calculate the average height of the random sample and store it
```

```
  average_heights[i]= mean(sample_heights)
}
```

Create a histogram of the 1000 average heights

```
hist(average_heights, breaks = 30, col = "lightblue", main = "Histogram of
Average Heights",
     xlab = "Average Height (inches)", ylab = "Frequency")
```



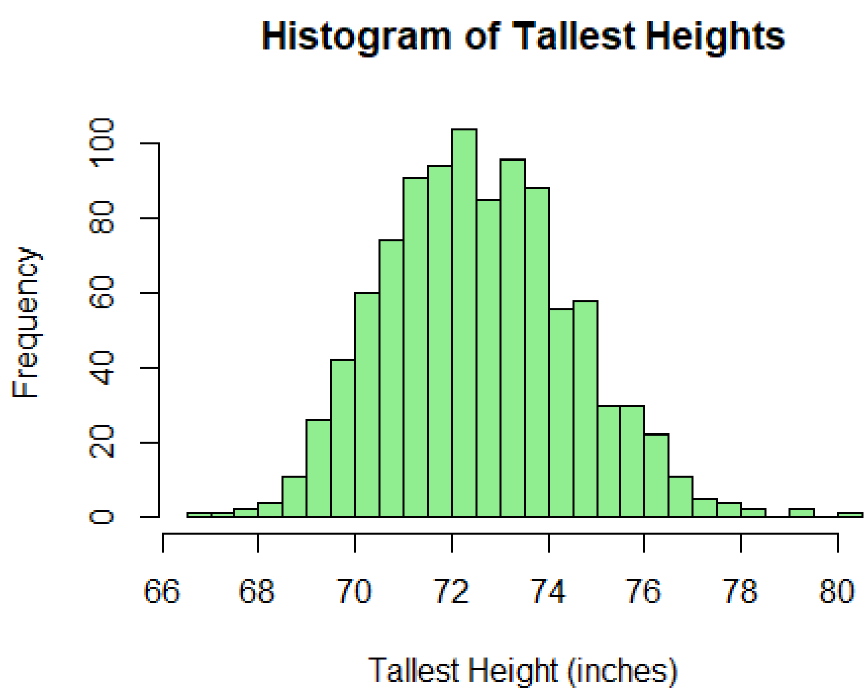
Clearly, the distribution of Average height is distributed Normally.

(c) Write another loop of 1000 height simulations of a 10-person sample. This time draw inferences on and plot the tallest heights.

ANSWER:-

```
# Define the function to generate a random sample of heights
get_mean_height= function(n, is_woman = TRUE) {
  if (is_woman) {
    mean_height= rnorm(n, mean = 63.7, sd = 2.7)
  } else {
    mean_height= rnorm(n, mean = 69.1, sd = 2.9)
  }
  return(mean_height)
}

# Number of simulations
num_simulations= 1000
```

```r
# Vector to store the tallest heights for each simulation
tallest_heights= numeric(num_simulations)

# Perform 1000 simulations
for (i in 1:num_simulations) {
  # Get a random sample of 5 women and 5 men
  n= 5
  women_heights= get_mean_height(n)
  men_heights= get_mean_height(n, is_woman = FALSE)

  # Combine the heights of women and men into a single sample
  sample_heights= c(women_heights, men_heights)

  # Find the tallest height in the random sample and store it
  tallest_heights[i]= max(sample_heights)
}

# Create a histogram of the tallest heights
hist(tallest_heights, breaks = 30, col = "lightgreen", main = "Histogram of
Tallest Heights",
     xlab = "Tallest Height (inches)", ylab = "Frequency")
```




```r
# Calculate and print the mean and standard deviation of the tallest heights
mean_tallest_height <- mean(tallest_heights)
sd_tallest_height <- sd(tallest_heights)
cat("Mean of Tallest Heights:", mean_tallest_height, "inches\n")
```

```
## Mean of Tallest Heights: 72.56593 inches
```

```r
cat("Standard Deviation of Tallest Heights:", sd_tallest_height, "inches\n")
```

```
## Standard Deviation of Tallest Heights: 1.958423 inches
```

3. The number of claims an insurance company receives in a day follow a Poisson distribution with rate 10 per day.The amount of a claim is a random variable that has an exponential distribution with mean $1000 (refer to Question 1 for how the distribution looks). The insurance company receives payments continuously in time at a constant rate of $11000 per day. Starting with an initial capital of $25000, use 10000 simulations to estimate the probability that the firm's capital is always positive throughout its first 365 days. You may use rpois() and rexp() from R base package to generate values from a poisson and an exponential distribution.

ANSWER:-

```r
# Number of simulations
num_simulations= 10000

# Number of days to simulate
num_days= 365

# Initial capital
initial_capital= 25000

# Rate of claims per day
rate_claims= 10

# Rate of payments received per day
rate_payments= 11000

# Function to simulate the company's capital over a given number of days
simulate_capital= function(num_days) {
  capital= numeric(num_days)
  capital[1]= initial_capital

  for (day in 2:num_days) {
    # Generate number of claims for the current day from Poisson distribution
    num_claims= rpois(1, lambda = rate_claims)

    # Generate claim amounts from exponential distribution
    claim_amounts= rexp(num_claims, rate = 1/1000)

    # Calculate the total claim amount for the day
    total_claims= sum(claim_amounts)

    # Calculate the net cash flow for the day
    net_cash_flow= rate_payments - total_claims
```

```
    # Update the capital for the day
    capital[day]= capital[day - 1] + net_cash_flow
  }

  return(capital)
}

# Vector to store whether the capital was always positive for each simulation
always_positive= logical(num_simulations)

# Perform 10000 simulations
for (i in 1:num_simulations) {
  capital_simulation= simulate_capital(num_days)

  # Check if the capital was always positive throughout the first 365 days
  always_positive[i]= all(capital_simulation > 0)
}

# Estimate the probability that the capital is always positive
probability_always_positive= sum(always_positive) / num_simulations
probability_always_positive

## [1] 0.9257
```

Week_4_HW_Stat_560

Sagar Kalauni

2023-07-29

```r
options(repos = "https://cran.rstudio.com/")
```

R Practise

In this exercise, we use a dataset from North Carolina, USA. In 2004, the state of North Carolina released a large data set containing information on births recorded in this state. We have observations on 13 different variables, some categorical and some numerical. The data set is available on Blackboard named ”data nc.csv”. Use functions in R to do the following. The variable in the data are:

```r
data=(data.frame(
  Variable = c("fage", "mage", "mature", "weeks", "premie", "visits",
"marital", "gained", "weight", "lowbirthweight", "gender", "habit",
"whitemom"),
  Description = c("father’s age in years", "mother’s age in years", "maturity
status of mother", "length of pregnancy in weeks", "whether the birth was
classified as premature (premie) or full-term", "number of hospital visits
during pregnancy", "whether mother is married or not married at birth",
"weight gained by mother during pregnancy in pounds", "weight of the baby at
birth in pounds", "whether baby was classified as low birthweight (low) or
not (not low)", "gender of the baby, female or male", "status of the mother
as a nonsmoker or a smoker", "whether mother is white or not white")
))

print(data)

##          Variable
## 1            fage
## 2            mage
## 3          mature
## 4           weeks
## 5          premie
## 6          visits
## 7         marital
## 8          gained
## 9          weight
## 10 lowbirthweight
## 11         gender
## 12          habit
## 13       whitemom
```

```
##                                                              Description
## 1                                                  father’s age in years
## 2                                                  mother’s age in years
## 3                                              maturity status of mother
## 4                                           length of pregnancy in weeks
## 5    whether the birth was classified as premature (premie) or full-term
## 6                            number of hospital visits during pregnancy
## 7                       whether mother is married or not married at birth
## 8                      weight gained by mother during pregnancy in pounds
## 9                                    weight of the baby at birth in pounds
## 10 whether baby was classified as low birthweight (low) or not (not low)
## 11                                      gender of the baby, female or male
## 12                         status of the mother as a nonsmoker or a smoker
## 13                                   whether mother is white or not white
```

ANSWERS:-

loading the necessary packages in R environment.

```
install.packages("tidyverse")

## Installing package into 'C:/Users/Dell/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Dell\AppData\Local\Temp\RtmpaQM80P\downloaded_packages

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.1

## Warning: package 'ggplot2' was built under R version 4.3.1

## Warning: package 'lubridate' was built under R version 4.3.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Loading the data into the R working environment.

```
ncdata= read_csv("C:/Users/Dell/Desktop/Foundation of data Science- STAT
560/Week-4 Stat-560/Data- Week 4/data_nc.csv")

## Rows: 1000 Columns: 13
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (7): mature, premie, marital, lowbirthweight, gender, habit, whitemom
## dbl (6): fage, mage, weeks, visits, gained, weight
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

Taking the first look at my data

```
View(ncdata)
```

1) In the data which variables are categorical and which are numerical? ANSWER:- let's look at the structure of my data variable

```
str(ncdata)

## spc_tbl_ [1,000 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ fage          : num [1:1000] NA NA 19 21 NA NA 18 17 NA 20 ...
##  $ mage          : num [1:1000] 13 14 15 15 15 15 15 15 16 16 ...
##  $ mature        : chr [1:1000] "younger mom" "younger mom" "younger mom"
"younger mom" ...
##  $ weeks         : num [1:1000] 39 42 37 41 39 38 37 35 38 37 ...
##  $ premie        : chr [1:1000] "full term" "full term" "full term" "full
term" ...
##  $ visits        : num [1:1000] 10 15 11 6 9 19 12 5 9 13 ...
##  $ marital       : chr [1:1000] "married" "married" "married" "married"
...
##  $ gained        : num [1:1000] 38 20 38 34 27 22 76 15 NA 52 ...
##  $ weight        : num [1:1000] 7.63 7.88 6.63 8 6.38 5.38 8.44 4.69 8.81
6.94 ...
##  $ lowbirthweight: chr [1:1000] "not low" "not low" "not low" "not low"
...
##  $ gender        : chr [1:1000] "male" "male" "female" "male" ...
##  $ habit         : chr [1:1000] "nonsmoker" "nonsmoker" "nonsmoker"
"nonsmoker" ...
##  $ whitemom      : chr [1:1000] "not white" "not white" "white" "white"
...
##  - attr(*, "spec")=
##   .. cols(
##   ..   fage = col_double(),
##   ..   mage = col_double(),
##   ..   mature = col_character(),
##   ..   weeks = col_double(),
```

```
##   ..   premie = col_character(),
##   ..   visits = col_double(),
##   ..   marital = col_character(),
##   ..   gained = col_double(),
##   ..   weight = col_double(),
##   ..   lowbirthweight = col_character(),
##   ..   gender = col_character(),
##   ..   habit = col_character(),
##   ..   whitemom = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

so my data variables are classified this way:

fage : numerical

mage : numerical

mature : categorical

weeks : numerical

premie : categorical

visits : numerical

marital : categorical

gained : numerical

weight : numerical

lowbirthweight: categorical

gender : categorical

habit : categorical

whitemom : categorical

2) let's make a side-by-side boxplot of habit and weight for smokers and non-smokers using ggplot(). What does the plot highlight about the relationship between these two variables?

ANSWER:-

```
# Removing all the na data from weight and habit column having null value in
it.
ncdata_cleaned= ncdata %>% filter(!is.na(habit)) %>% filter(!is.na(weight))
ncdata_cleaned

## # A tibble: 999 × 13
##     fage  mage mature   weeks premie visits marital gained weight
lowbirthweight
##    <dbl> <dbl> <chr>    <dbl> <chr>   <dbl> <chr>    <dbl>  <dbl> <chr>
```

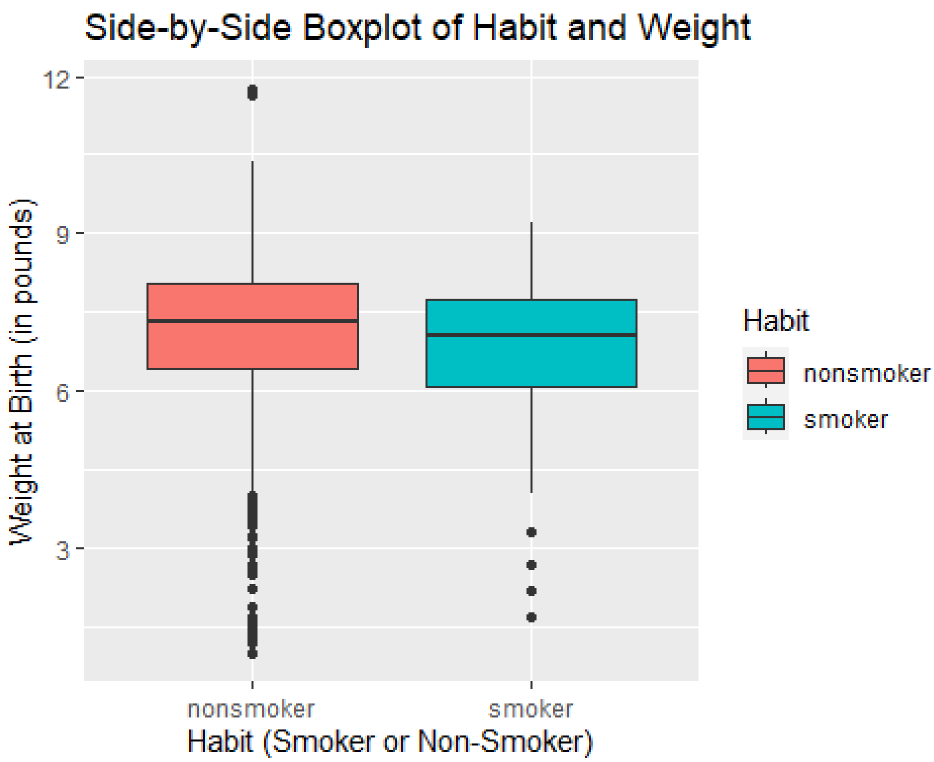
```
## 1     NA    13 younger…    39 full …     10 married     38    7.63 not low
## 2     NA    14 younger…    42 full …     15 married     20    7.88 not low
## 3     19    15 younger…    37 full …     11 married     38    6.63 not low
## 4     21    15 younger…    41 full …      6 married     34    8    not low
## 5     NA    15 younger…    39 full …      9 married     27    6.38 not low
## 6     NA    15 younger…    38 full …     19 married     22    5.38 low
## 7     18    15 younger…    37 full …     12 married     76    8.44 not low
## 8     17    15 younger…    35 premie      5 married     15    4.69 low
## 9     NA    16 younger…    38 full …      9 married     NA    8.81 not low
## 10    20    16 younger…    37 full …     13 married     52    6.94 not low
## # ℹ 989 more rows
## # ℹ 3 more variables: gender <chr>, habit <chr>, whitemom <chr>
```

```
ncdata_cleaned %>% group_by(habit) %>% summarise(N=n())
```

```
## # A tibble: 2 × 2
##   habit         N
##   <chr>     <int>
## 1 nonsmoker   873
## 2 smoker      126
```

From this result we can clearly see that the number of nonsmoker women is high more then smoker by the ratio of almost 7:1.

```
# Boxplot of habit and weight
ggplot(data = ncdata_cleaned, aes(x = habit, y = weight, fill=habit)) +
geom_boxplot() +
labs(x = "Habit (Smoker or Non-Smoker)", y = "Weight at Birth (in pounds)",
       title = "Side-by-Side Boxplot of Habit and Weight",
       fill = "Habit")
```



The plots show that the median birth weight for children born to nonsmoking mothers appears to be slightly greater than the median birth weight for children born to mothers who smoke. The sizable number of outliers at the bottom indicate that the distributions are left-skewed, as the means for each group should be lower than the respective medians. There are a much larger number of outliers associated with non-smoking mothers, but from the above box plots alone it is not apparent that the two subsets are of very different size, with 873 nonsmokers vs. 126 smokers, a ratio of nearly 7:1 .

3) Construct a 90% confidence interval for average number of hospital visits during pregnancy.Interpret your conclusion.

ANSWER:-

```
summary(ncdata$visits)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0    10.0    12.0    12.1    15.0    30.0       9

ncdata=drop_na(ncdata,visits)
t.test(ncdata$visits,mu=12.0, conf.level=0.90)

##
##  One Sample t-test
##
## data:  ncdata$visits
## t = 0.83533, df = 990, p-value = 0.4037
## alternative hypothesis: true mean is not equal to 12
## 90 percent confidence interval:
##  11.89810 12.31179
```

```
## sample estimates:
## mean of x
##  12.10494
```

So, 90% confidence interval for average number of hospital visits during pregnancy is: (11.89810, 12.31179)

Alternatively,

```r
visits_mean= mean(ncdata$visits, na.rm = TRUE)
visits_sd= sd(ncdata$visits, na.rm = TRUE)
n= length(ncdata$visits) - sum(is.na(ncdata$visits))

# Calculate the standard error of the mean
se= visits_sd / sqrt(n)

# Calculate the 90% confidence interval
conf_interval= visits_mean + c(-1, 1) * qt(0.95, df = n - 1) * se

# Print the confidence interval
conf_interval

## [1] 11.89810 12.31179
```

Conclusion: it means that there is a 90% chance that the average number of hospital visits during pregnancy will be somewhere between (11.89810, 12.31179).

4) Perform a hypotheses test to see there is evidence to believe majority (more than 50%) of the mothers are married at birth. Interpret your conclusion.

ANSWER:-

```r
table(ncdata$marital)

##
##     married not married
##         380         611

x=table(ncdata$marital)["married"]
n=sum(table(ncdata$marital))

# default two-sided test and two-sided confidence interval
prop_test= prop.test(x,n,p=0.5, alternative = "greater")
prop_test

##
##  1-sample proportions test with continuity correction
##
## data:  x out of n, null probability 0.5
## X-squared = 53.38, df = 1, p-value = 1
## alternative hypothesis: true p is greater than 0.5
## 95 percent confidence interval:
##  0.3578986 1.0000000
```

```
## sample estimates:
##         p
## 0.3834511
```

H_0: majority (more than 50%) of the mothers are married at birth.

H_1: minority (less than 50%) of the mothers are married at birth.

Since p-value=1 is greater than 0.05, we do not have enough evidence to reject the null hypothesis which says: the majority of mothers are married at birth.

5) Perform a hypotheses test to see if the proportion of low-birthweight babies is higher for smoking mothers. Interpret your conclusion.

ANSWER:-

```
summary(ncdata$lowbirthweight)

##    Length     Class      Mode
##       991 character character
```

lowbirthweight is categorical data.

```
summary(ncdata$habit)

##    Length     Class      Mode
##       991 character character

data_nc=drop_na(ncdata,lowbirthweight)
```

habit is categorical data.

```
tab_counts=ncdata %>% group_by(lowbirthweight,habit) %>%
summarize(N = n()) %>% spread(habit, N) %>% mutate(TOTAL=nonsmoker+smoker)

## `summarise()` has grouped output by 'lowbirthweight'. You can override
using
## the `.groups` argument.

prop.test(tab_counts$smoker,tab_counts$TOTAL, alternative = "greater")

##
##  2-sample test for equality of proportions with continuity correction
##
## data:  tab_counts$smoker out of tab_counts$TOTAL
## X-squared = 1.4174, df = 1, p-value = 0.1169
## alternative hypothesis: greater
## 95 percent confidence interval:
##  -0.02139698  1.00000000
## sample estimates:
##    prop 1    prop 2
## 0.1666667 0.1211778
```

```
smoking_low_count <- sum(ncdata$lowbirthweight == "low" & data_nc$habit ==
"smoker", na.rm = TRUE)
smoking_total <- sum(ncdata$habit == "smoker", na.rm = TRUE)
smoking_low_proportion <- smoking_low_count / smoking_total

non_smoking_low_count <- sum(ncdata$lowbirthweight == "low" & data_nc$habit
== "nonsmoker", na.rm = TRUE)
non_smoking_total <- sum(ncdata$habit == "nonsmoker", na.rm = TRUE)
non_smoking_low_proportion <- non_smoking_low_count / non_smoking_total

# Perform a two-sample proportion test
prop_test <- prop.test(c(smoking_low_count, non_smoking_low_count),
c(smoking_total, non_smoking_total), alternative = "greater")

# Print the test result
prop_test

##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(smoking_low_count, non_smoking_low_count) out of c(smoking_total,
non_smoking_total)
## X-squared = 1.4174, df = 1, p-value = 0.1169
## alternative hypothesis: greater
## 95 percent confidence interval:
##  -0.01889933  1.00000000
## sample estimates:
##    prop 1    prop 2
## 0.1440000 0.1039261
```

H_0: The proportion of low_birthweight babies is higher for smoking mother

H_a: The proportion of low_birthweight babies is lower for smoking mother

Since the p_value = 0.1169 is greater than the significance level (α = 0.05), we fail to reject the null hypothesis. Therefore, we do not have sufficient evidence to support the alternative hypothesis, which suggests that The proportion of low_birthweight babies is lower for smoking mother.

6) Perform a hypotheses test to see if the average weights of babies born to smoking and non-smoking mothers are different. Interpret your conclusion.

ANSWER: -

```
smoking_weight= ncdata$weight[ncdata$habit == "smoker"]
non_smoking_weight <- ncdata$weight[ncdata$habit == "nonsmoker"]

# Perform a two-sample t-test
t_test= t.test(smoking_weight, non_smoking_weight)
t_test
```

```
##
##  Welch Two Sample t-test
##
## data:  smoking_weight and non_smoking_weight
## t = -2.362, df = 168.53, p-value = 0.01932
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.58197500 -0.05205788
## sample estimates:
## mean of x mean of y
##  6.837360  7.154376
```

$ {H_0} $: Average weights of babies born to smoking and non-smoking mothers are the same $ {H_1} $: average weights of babies born to smoking and non-smoking mothers are different

The p-value = 0.01932 is less than 0.05, we reject the null hypothesis.

7) Determine if low birthweight is independent of the gender of the baby. Perform appropriate tests for support your conclusion.

ANSWER: -

```
contingency_table= table(ncdata$lowbirthweight, ncdata$gender)

# Perform a chi-square test for independence
chi_square_test= chisq.test(contingency_table)
chi_square_test
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 0.074252, df = 1, p-value = 0.7852
```

H_0: Low birthweight is independent of the gender of the baby. H_1: Low birthweight is dependent of the gender of the baby. The p-value = 0.7852 is greater than 0.05, we fail to reject the null hypothesis and conclude we don't have enough evidence to support the alternative hypothesis, which says Low birthweight is dependent of the gender of the baby.

```
$The \ End $
```