Final Project

Dipa Rai

16/12/2022

Part 1:

working directory

```
setwd("~/Desktop/R Programming final ")
```

The data is of daily conformed Covid_19 cases and deaths from Covid-19 recorded by mainly a 10-year age group intervals and gender from the 2nd of February 2020 to the 13th of December 2022 in Ireland. It contains 1018 observation and 44 variable. It is accessed from data.gov.ie.

```
covid_profile <- read.csv("COVID-19_HPSC_Detailed_Statistics_Profile.csv")
```

```
library('tidyverse')
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library('dplyr')
library("ggplot2")
```

Selected only 13 variables, which contains date, daily conformed cases, total conformed cases, sex, and age profiles. The missing values are omitted and left with 1016 observations

```
select_variables <- covid_profile %>%
  select( Date,ConfirmedCovidCases,TotalConfirmedCovidCases ,Male,Female,HospitalisedAged15to24, Hospit
 na.omit()
```

The age profiles columns are renamed for the better readability, which I am interested in investigating

```
select_variables <-  select_variables %>%
  rename("Aged15_24" = "HospitalisedAged15to24",
         "Aged25_34" = "HospitalisedAged25to34",
         "Aged35_44" = "HospitalisedAged35to44",
         "Aged45_54" = "HospitalisedAged45to54",
         "Aged55_64" = "HospitalisedAged55to64",
         "Ahed65_74" = "HospitalisedAged65to74",
         "Aged75_84" = "HospitalisedAged75to84",
         "Aged85_up"   = "Aged85up")
```

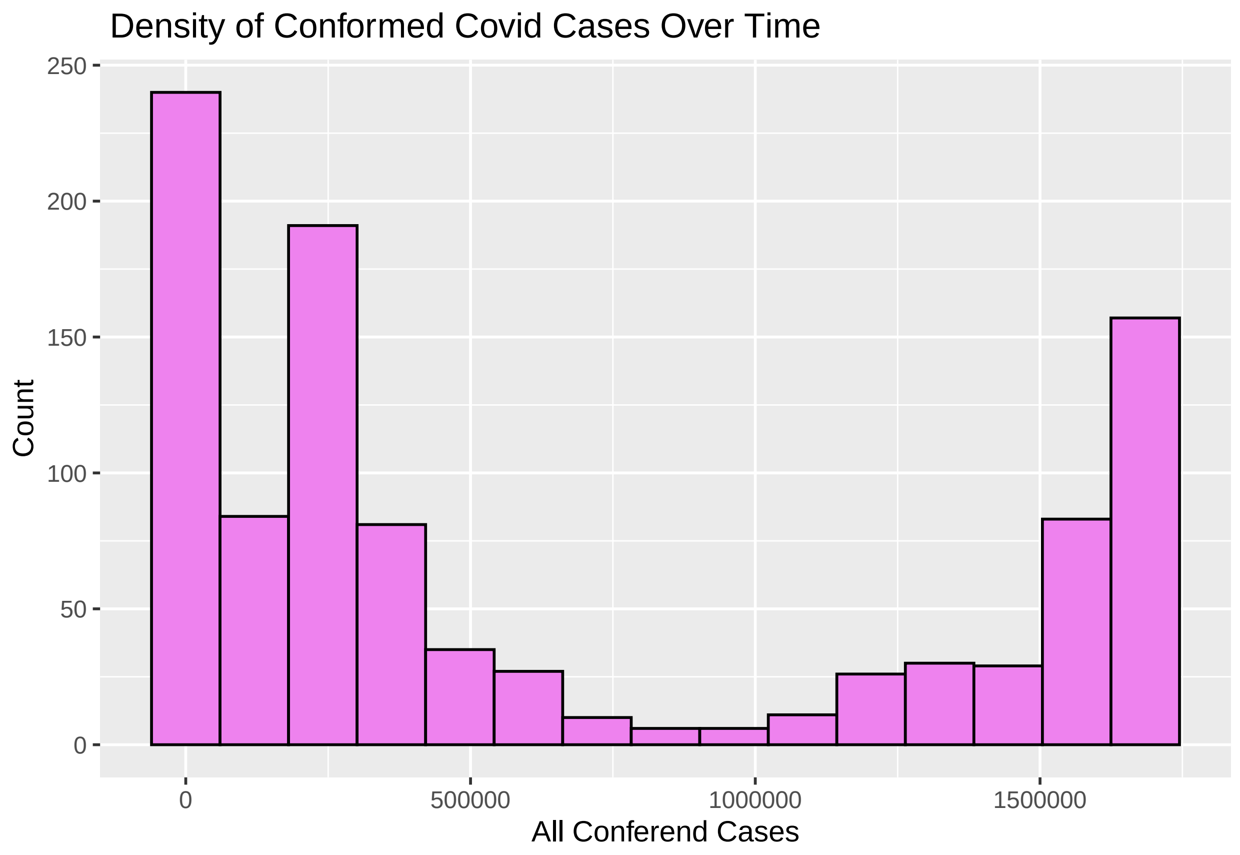
The summary statistics of male and female who were diseased by Covid-19. The mean value for male is 305255 and for female 339514. The median value for male is 140092 and for female it is 149920. Female numbers were slightly more than male who were diseased during the given time

```
select_variables %>%
  select(Male,Female) %>%
  summary
```

```
##       Male             Female
##  Min.   :     1   Min.   :     0
##  1st Qu.: 31204   1st Qu.: 35154
##  Median :140092   Median :149920
##  Mean   :305255   Mean   :339514
##  3rd Qu.:708031   3rd Qu.:787920
##  Max.   :785688   Max.   :898839
```

Histogram shows the distribution daily of conformed cases of Covid_19 during the covid period.

```
ggplot(data = select_variables, aes(x = TotalConfirmedCovidCases)) +
  geom_histogram(fill = "violet",
               color = "black" , bins = 15)+
  ggtitle(" Density of Conformed Covid Cases Over Time")+
  labs(x = "All Conferend Cases", y = "Count")
```



The date variable in the original data is in character format so now it is changed to Date format and plotted the same variable again below

```
class(select_variables$Date)
```

```
## [1] "character"
```

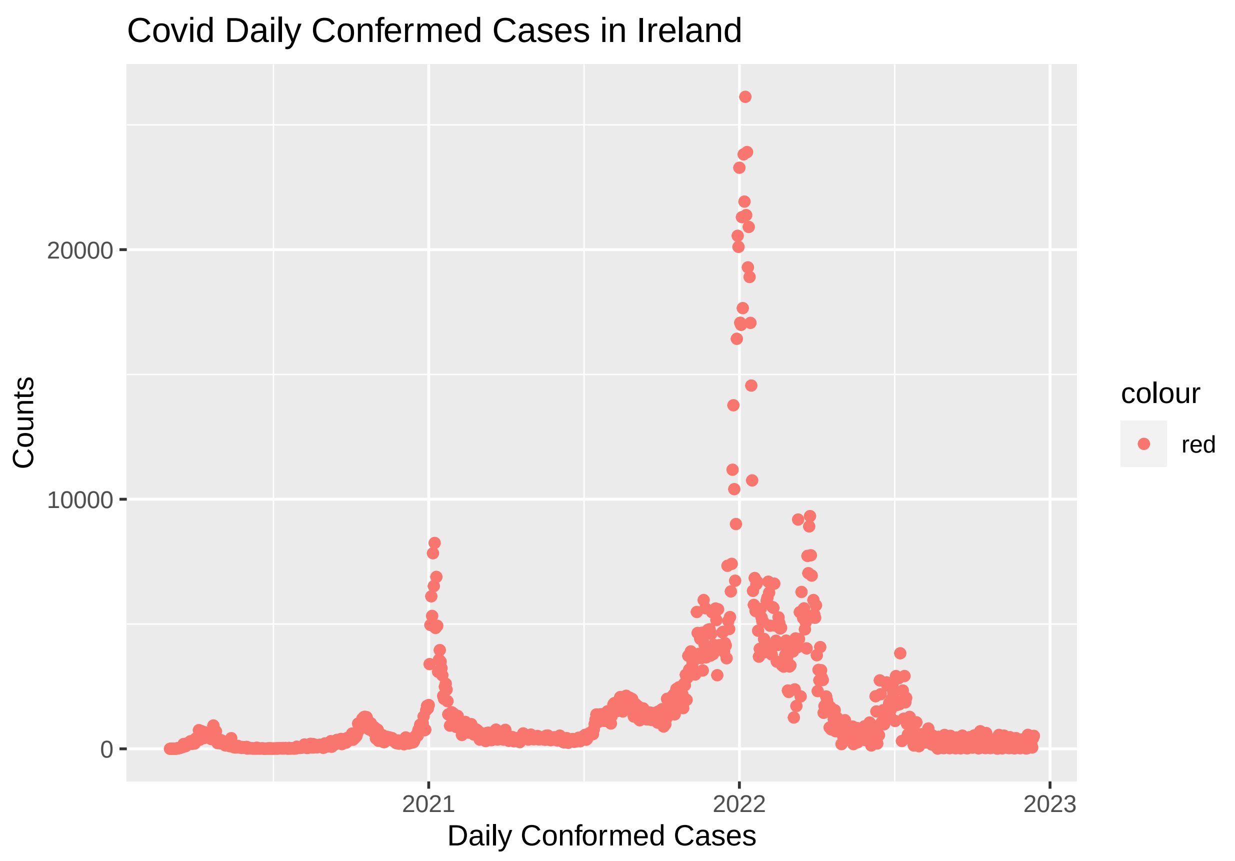
```
select_variables$format_date <- as.Date(select_variables$Date)

class(select_variables$format_date)
```

```
## [1] "Date"
```

Now this graph displays the distribution of people who caught the disease. The X axis shows the the time by years and the Y axis is the number of people in 10 thousand, who had suffered from covid. The high picks indicate that there were two main times the disease spread very densely.

```
qplot(x=format_date, y = select_variables$ConfirmedCovidCases,
       data = select_variables,
      xlab = "Daily Conformed Cases", ylab = "Counts" ,  main = "Covid Daily Confermed Cases in Ireland
```



Removed five columns from the dataset left with only age groups variables. The records of these numbers grouped by age is that people who were hospitalized due to Covid_19

```
exclude_date <- select_variables [, ! (names(select_variables) %in% c ("Date", "ConfirmedCovidCases", "
head(exclude_date, 4)
```

```
##   Aged15_24 Aged25_34 Aged35_44 Aged45_54 Aged55_64 Ahed65_74 Aged75_84
## 3         0         0         0         0         0         0         0
## 4         0         0         0         1         0         0         0
## 5         0         0         2         1         0         0         0
## 6         0         0         4         1         0         0         0
##   Aged85_up
## 3         0
## 4         0
## 5         0
## 6         0
```

Mean values of each age group is calculated using sapply function. Then assigned them to a new set of data and saved on to local environment.

```
neam_of_All <-print(sapply(exclude_date, mean))
```

```
##  Aged15_24  Aged25_34  Aged35_44  Aged45_54  Aged55_64  Ahed65_74  Aged75_84
##   977.2421  1592.7736  1803.0344  2117.7057  2571.1880  3353.8720  4182.0276
##  Aged85_up
## 10355.4852
```

The sum of each age group is calculated using colSums function. Than it is assigned to a new data name all_ColSums.

```
all_ColSum <- colSums(exclude_date)
```

The all_ColSums data is saved in the R local environment and brought it on to the global environment using double arrows converting it to a data frame to manipulate it further

```
all_ColSum_Glob <<- as.data.frame(all_ColSum)
```

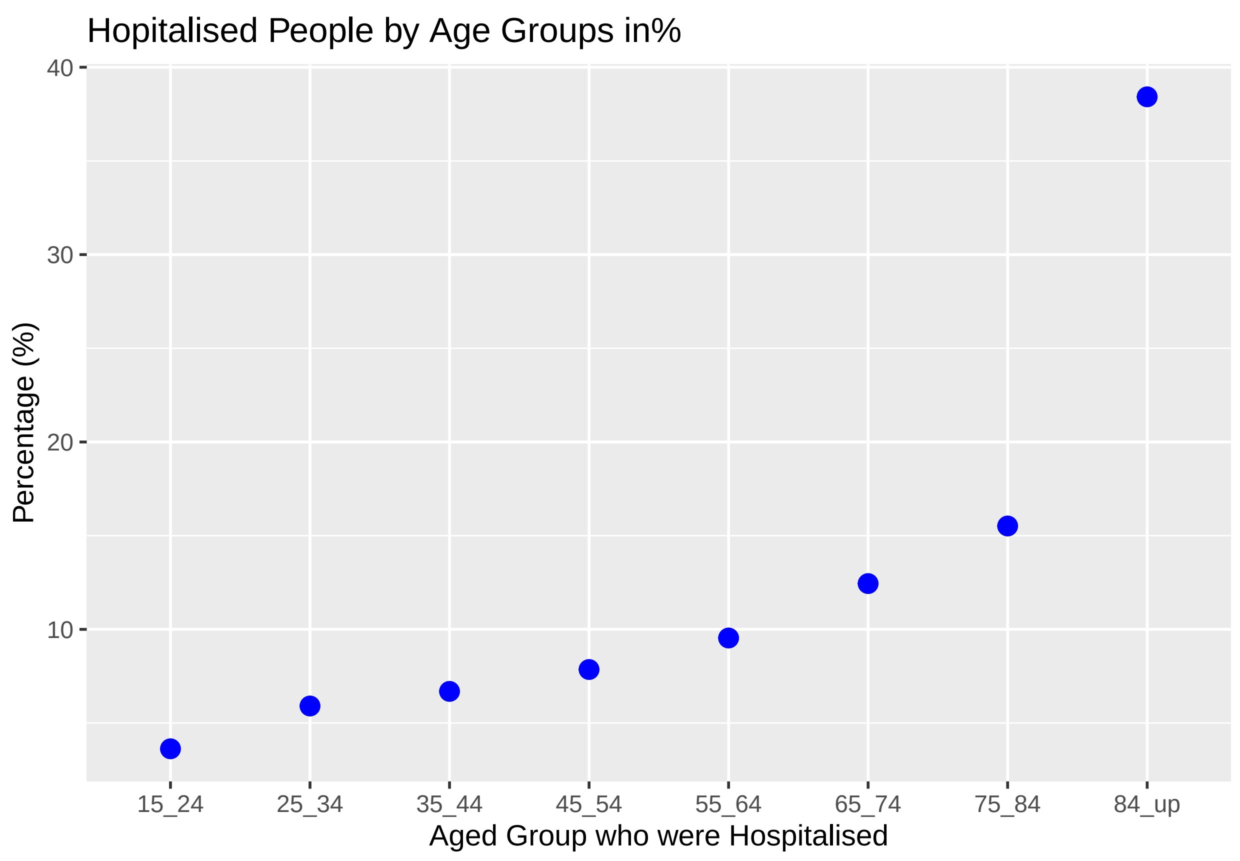
Calculated the proportion of each age group

```
all_ColSum_Glob$Per <- all_ColSum_Glob$all_ColSum/sum(all_ColSum_Glob$all_ColSum)
all_ColSum_Glob$Per <- all_ColSum_Glob$Per*100
```

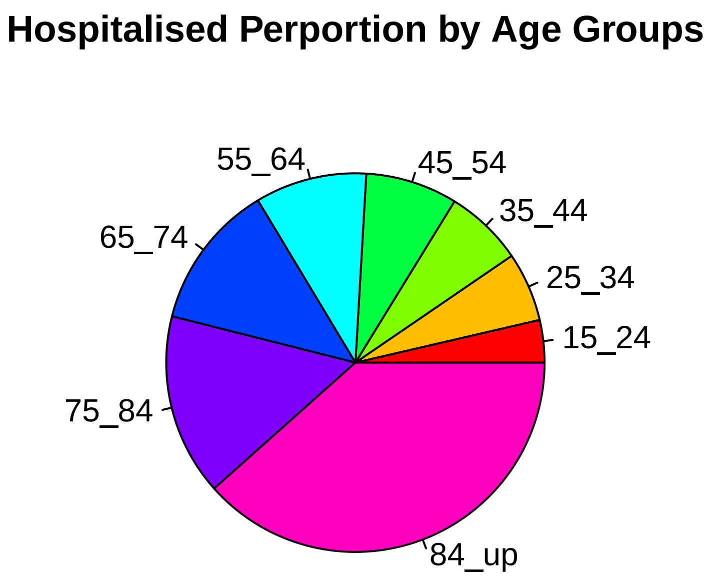
The point graph and the pie chart below display the percentage of people by age groups who were hospitalized due to covid. The graph shows that older people particularly who were above 84 years old were highly effected by the disease. comparatively, younger people were less effected. Looking at the graph that there is a linear positive correlation between age and hospitalized numbers. It indicates that older people are highly vulnerable to the disease.

```
Age_groups <- c("15_24", "25_34", "35_44", "45_54", "55_64", "65_74", "75_84","84_up")
all_CoSum_Glob2 <- cbind(all_ColSum_Glob, Age_groups) #This code is used to create labels for the varia
```

```
ggplot(all_ColSum_Glob, aes(x = Age_groups,
                            y = Per)) +
   geom_point(size = 3, color = "blue")+
  labs(title = "Hopitalised People by Age Groups in%", x = "Aged Group who were Hospitalised", y = "Perc
```



```
z <- c(2,4,6,8,10,12,14,16)
pie(all_CoSum_Glob2$Per, labels = all_CoSum_Glob2$Age_groups, main = "Hospitalised Perportion by Age Gro
```



Part 2

Lubridate Package

I am going to be investigating the package called 'lubridate' The lubricate package is used for manipulating and working with date and time more effectively. Below I will demonstrate some of its functions and their functionality

```
library("lubridate")
```

```
## Loading required package: timechange
```

```
##
## Attaching package: 'lubridate'
```

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

```
#function dmy
dmy(09112020) #creating a date by day, month, and year.
```

```
## [1] "2020-11-09"
```

```r
#function ymd
ymd(20201109) #creating a date by year, month, and day.
```

```
## [1] "2020-11-09"
```

```r
#function mdy
mdy(11092020) #crating a date by month, day, and year.
```

```
## [1] "2020-11-09"
```

We can also extract only day or month or year from a set of date. It is demonstrated below

```r
# A randomdata date_exmpl is created using ymd()function
date_exmpl <- ymd("2021/05/01")
day(date_exmpl) #extracting only the day.
```

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

```r
month(date_exmpl) #extracting only the month.
```

```
## [1] 5
```

```r
year(date_exmpl) #extracting only the year.
```

```
## [1] 2021
```

The function wday() outputs 7 days from the 2021/05/01 which was assigned to the date data 'date_exmpl'

```r
wday(date_exmpl)
```

```
## [1] 7
```

```r
wday(date_exmpl, label = TRUE) #outputs the name of the days.
```

```
## [1] Sat
## Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat
```

```r
hm("5 40") # The hm() outputs the hours and minutes.
```

```
## [1] "5H 40M 0S"
```

```r
hms("5 40 15") #The hms() outputs the hours, minutes and the second.
```

```
## [1] "5H 40M 15S"
```

```
hms("5:40:15")
```

```
## [1] "5H 40M 15S"
```

The duration() function breaks down time into preferable units for example minutes into seconds and hours into minutes and so on. It is very cool.

```
duration(minute = 60)
```

```
## [1] "3600s (~1 hours)"
```

```
dminutes(30)
```

```
## [1] "1800s (~30 minutes)"
```

```
dhours(5)
```

```
## [1] "18000s (~5 hours)"
```

```
duration(week=1)
```

```
## [1] "604800s (~1 weeks)"
```

```
duration(month=1)
```

```
## [1] "2629800s (~4.35 weeks)"
```

```
duration(year=1)
```

```
## [1] "31557600s (~1 years)"
```

The period () function converts the long formatting of times into shorter format that the way we use time parameters.

```
period(hour = 3, minute = 20, second = 10)
```

```
## [1] "3H 20M 10S"
```

```
time_data <-  "2022, September,21"
time_data <- lubridate::ymd(time_data)
print(time_data)
```

```
## [1] "2022-09-21"
```

Part 3

Create Functions

I created function for measuring a river flow using the water discharge equation. Discharge = water velocity times channel width times channel depth. The name of the function is river_flow. There are three inputs or arguments are given to the function. A subset is set as discharge which, holds the mathematical operations for the function's functionality

```
river_flow <- function(velocity, width, depth){
  discharge <- velocity * width * depth
  return(discharge)
}
```

```
result_1st <- river_flow(velocity = 0.845, width = 4.25, depth = 2.5) # vectors' names and their values
print(result_1st)
```

```
## [1] 8.978125
```

The function can also give the output without the names assigned in the function's input above

```
result_2nd <- river_flow(0.845, 4.25, 2.5) #without the vectors' name as it returns the output based on
```

```
result_1st <- river_flow(depth = 2.5, velocity = 0.845, width = 4.25) # one of the benefit using names
print(result_1st)
```

```
## [1] 8.978125
```

#Default Setting

A default value for the parameter depth is set to equals to 2.

```
defult_setting <-function(velocity, width, depth = 2){
  discharge <- velocity * width *depth
  return(discharge)
}
```

```
result_3rd <- defult_setting(0.845, 3.75) # Only velocity and width values are give in the code but, it
result_3rd
```

```
## [1] 6.3375
```

#Combining or nesting multiple functions.

Velocity square is just a random calculation say speed of water flow is V^2.

```
velocity <- function(velocity){
  rate <- velocity^2
  return(rate)
}
```

The following codes combine multiple functions storing their outputs to intermediate variables

```
water_discharge <- river_flow(0.845, 4.25, 2.5)
print(water_discharge )
```

```
## [1] 8.978125
```

The intermediate variable is used to store the output of one function and then pass it as the input to the next function. This is a function nesting inside the other functions.

```
discharge_rate <- velocity(water_discharge)
print(discharge_rate)
```

```
## [1] 80.60673
```

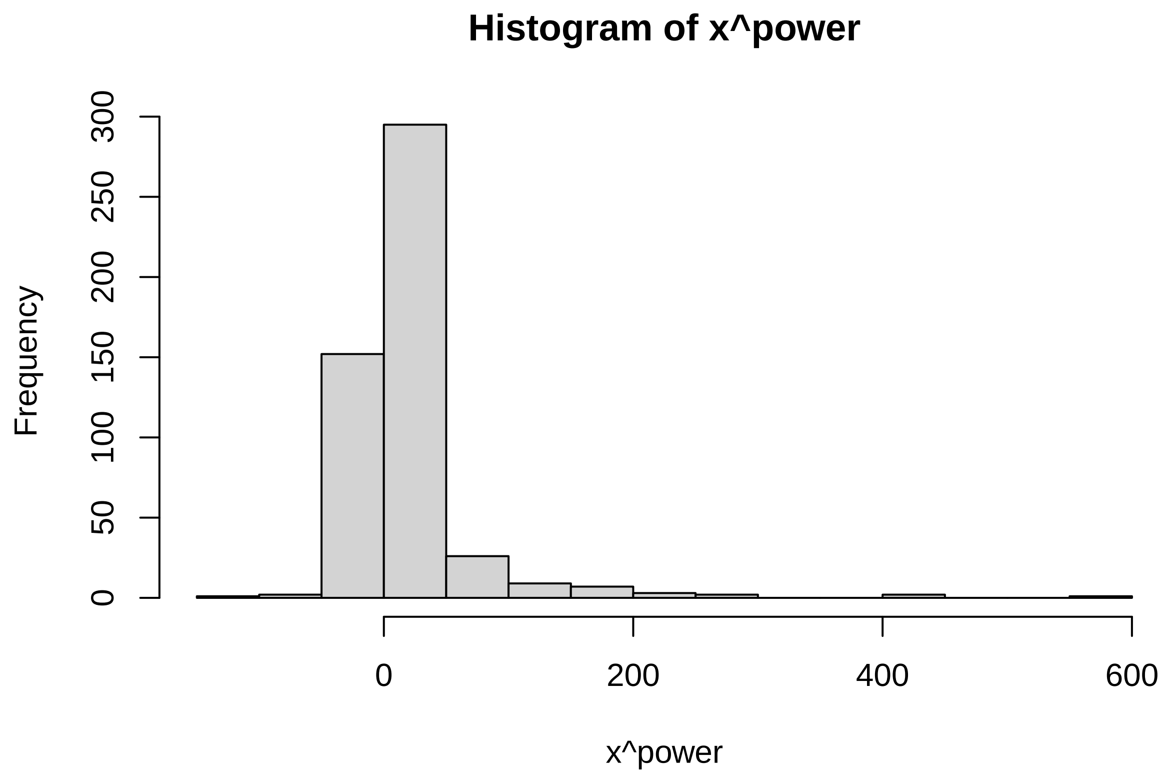
#Creating graphs

More functions, here 500,1,2 of random numbers are created as to create a histogram graph using a new function.

```
drtribution <- rnorm(500,1,2)
```

```
more_function <- function(x, power = 2){
  hist(x^power) # x to the power 2 is set for some mathematical operation and to plot that onto the arg
  return(hist)
}
```

```
Create_hist <- more_function(drtribution, 3)#The distribution holds the place of x and the power is set
```



```
print(Create_hist)
```

```
## function (x, ...)
## UseMethod("hist")
## <bytecode: 0x7fe8c63ae120>
## <environment: namespace:graphics>
```