R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac.

R is free software distributed under a GNU-style copy left, and an official part of the GNU project called **GNU S**.

Evolution of R

R was initially written by **Ross Ihaka** and **Robert Gentleman** at the Department of Statistics of the University of Auckland in Auckland, New Zealand. R made its first appearance in 1993.

- A large group of individuals has contributed to R by sending code and bug reports.
- Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive.

Features of R

As stated earlier, R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R –

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

As a conclusion, R is world's most widely used statistics programming language. It's the # 1 choice of data scientists and supported by a vibrant and talented community of contributors. R is taught in universities and deployed in mission critical business applications.

R Command Prompt

Once you have R environment setup, then it's easy to start your R command prompt by just typing the following command at your command prompt –

```
$ R
```

This will launch R interpreter and you will get a prompt > where you can start typing your program as follows –

```
> myString <- "Hello, World!"
> print ( myString)
[1] "Hello, World!"
```

Here first statement defines a string variable myString, where we assign a string "Hello, World!" and then next statement print() is being used to print the value stored in variable myString.

R Script File

Usually, you will do your programming by writing your programs in script files and then you execute those scripts at your command prompt with the help of R interpreter called **Rscript**. So let's start with writing following code in a text file called test.R as under –

```
# My first program in R Programming
myString <- "Hello, World!"

print ( myString)
```

Save the above code in a file test.R and execute it at Linux command prompt as given below. Even if you are using Windows or other system, syntax will remain same.

```
$ Rscript test.R
```

Comments

Comments are like helping text in your R program and they are ignored by the interpreter while executing your actual program. Single comment is written using # in the beginning of the statement as follows –

```
# My first program in R Programming
```

R does not support multi-line comments but you can perform a trick which is something as follows –

```
if(FALSE) {
  "This is a demo for multi-line comments and it should be put inside either a
    single OR double quote"
}

myString <- "Hello, World!"
print ( myString)
```

```
[1] "Hello, World!"
```

Though above comments will be executed by R interpreter, they will not interfere with your actual program. You should put such comments inside, either single or double quote.

Statistical analysis in R is performed by using many in-built functions. Most of these functions are part of the R base package. These functions take R vector as an input along with the arguments and give the result.

The functions we are discussing in this chapter are mean, median and mode.

Mean

It is calculated by taking the sum of the values and dividing with the number of values in a data series.

The function **mean()** is used to calculate this in R.

Syntax

The basic syntax for calculating mean in R is –

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Following is the description of the parameters used –

- **x** is the input vector.
- **trim** is used to drop some observations from both end of the sorted vector.
- **na.rm** is used to remove the missing values from the input vector.

Example

```
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find Mean.
result.mean <- mean(x)
print(result.mean)
```

When we execute the above code, it produces the following result –

```
[1] 8.22
```

Median

The middle most value in a data series is called the median. The **median()** function is used in R to calculate this value.

Syntax

The basic syntax for calculating median in R is –

```
median(x, na.rm = FALSE)
```

Following is the description of the parameters used –

- **x** is the input vector.
- **na.rm** is used to remove the missing values from the input vector.

```
# Create the vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find the median.
median.result <- median(x)
print(median.result)
```

Mode

The mode is the value that has highest number of occurrences in a set of data. Unike mean and median, mode can have both numeric and character data.

R does not have a standard in-built function to calculate mode. So we create a user function to calculate mode of a data set in R. This function takes the vector as input and gives the mode value as output.

Example

```
# Create the function.
```

```
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Create the vector with numbers.
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)

# Calculate the mode using the user function.
result <- getmode(v)
print(result)

# Create the vector with characters.
charv <- c("o","it","the","it","it")

# Calculate the mode using the user function.
result <- getmode(charv)
print(result)
```

When we execute the above code, it produces the following result –

```
[1] 2
[1] "it"
```

Information About the Data Set

You can use the question mark (`?`) to get information about the `mtcars` data set:

Example

```
# Use the question mark to get information about the data set

?mtcars
```

Result:

mtcars {datasets} R Documentation

Motor Trend Car Road Tests

Description

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models).

Usage

```
mtcars
```

Format

A data frame with 32 observations on 11 (numeric) variables.

[, 1] mpg Miles/(US) gallon

[, 2] cyl Number of cylinders

[, 3] disp Displacement (cu.in.)

[, 4] hp Gross horsepower

[, 5] drat Rear axle ratio

[, 6] wt Weight (1000 lbs)

[, 7] qsec 1/4 mile time

[, 8] vs Engine (0 = V-shaped, 1 = straight)

[, 9] am Transmission (0 = automatic, 1 = manual)

[,10] gear Number of forward gears

[,11] carb Number of carburetors

Note

Get Information

Use the `dim()` function to find the dimensions of the data set, and the `names()` function to view the names of the variables:

Example

```
Data_Cars <- mtcars # create a variable of the mtcars data set for better organization

# Use dim() to find the dimension of the data set
dim(Data_Cars)

# Use names() to find the names of the variables from the data set
names(Data_Cars)
```

Result:

```
[1] 32 11
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
"gear"
[11] "carb"
```

Use the `rownames()` function to get the name of each row in the first column, which is the name of each car:

Example

```
Data_Cars <- mtcars

rownames(Data_Cars)
```

Result:

```
 [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
 [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
 [7] "Duster 360"          "Merc 240D"           "Merc 230"
[10] "Merc 280"            "Merc 280C"           "Merc 450SE"
[13] "Merc 450SL"          "Merc 450SLC"         "Cadillac
Fleetwood"
```

```
[16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
[19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
[22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
[25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
[28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
[31] "Maserati Bora"
```

Print Variable Values

If you want to print all values that belong to a variable, access the data frame by using the `$` sign, and the name of the variable (for example `cyl` (cylinders)):

Example

```
Data_Cars <- mtcars

Data_Cars$cyl
```

Result:

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6
8 4
```

Sort Variable Values

To sort the values, use the `sort()` function:

Example

```
Data_Cars <- mtcars

sort(Data_Cars$cyl)
```

Result:

```
 [1] 4 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 8 8 8 8 8 8 8 8 8 8 8 8
8 8
```

Analyzing the Data

Now that we have some information about the data set, we can start to analyze it with some statistical numbers.

For example, we can use the `summary()` function to get a statistical summary of the data:

Example

```
Data_Cars <- mtcars

summary(Data_Cars)
```

Do not worry if you do not understand the output numbers. You will master them shortly.

The `summary()` function returns six statistical numbers for each variable:

- Min
- First quantile (percentile)
- Median
- Mean
- Third quantile (percentile)
- Max

We will cover all of them, along with other statistical numbers in the next chapters.

Max Min

In the previous chapter, we introduced the **mtcars** data set. We will continue to use this data set throughout the next pages.

You learned from the R Math chapter that R has several built-in math functions. For example, the `min()` and `max()` functions can be used to find the lowest or highest value in a set:

Example

Find the largest and smallest value of the variable `hp` (horsepower).

```
Data_Cars <- mtcars

max(Data_Cars$hp)
min(Data_Cars$hp)
```

Result:

```
[1] 335
[1] 52
```

Outliers

Max and min can also be used to detect **outliers**. An outlier is a data point that differs from rest of the observations.

Example of data points that could have been outliers in the **mtcars** data set:

- If maximum of forward gears of a car was 11
- If minimum of horsepower of a car was 0
- If maximum weight of a car was 50 000 lbs

Mean

To calculate the average value (mean) of a variable from the `mtcars` data set, find the sum of all values, and divide the sum by the number of values.

Sorted observation of wt (weight)

1.513	1.615	1.835	1.935	2.140	2.200
2.620	2.770	2.780	2.875	3.150	3.170
3.435	3.440	3.440	3.440	3.460	3.520
3.730	3.780	3.840	3.845	4.070	5.250

Luckily for us, the `mean()` function in R can do it for you:

Median

The median value is the value in the middle, after you have sorted all the values.

If we take a look at the values of the `wt` variable (from the `mtcars` data set), we will see that there are two numbers in the middle:

Sorted observation of wt (weight)

1.513	1.615	1.835	1.935	2.140	2.200
2.620	2.770	2.780	2.875	3.150	3.170
3.435	3.440	3.440	3.440	3.460	3.520
3.730	3.780	3.840	3.845	4.070	5.250

Note: If there are two numbers in the middle, you must divide the sum of those numbers by two, to find the median.

Luckily, R has a function that does all of that for you: Just use the `median()` function to find the middle value:

Example

Find the mid point value of weight (`wt`):

```
Data_Cars <- mtcars

median(Data_Cars$wt)
```

Result:

```
[1] 3.325
```

Mode

The mode value is the value that appears the most number of times.

R does not have a function to calculate the mode. However, we can create our own function to find it.

If we take a look at the values of the `wt` variable (from the `mtcars` data set), we will see that the numbers **3.440** are often shown:

Sorted observation of wt (weight)

1.513	1.615	1.835	1.935	2.140	2.200
2.620	2.770	2.780	2.875	3.150	3.170
3.435	3.440	3.440	3.440	3.460	3.520
3.730	3.780	3.840	3.845	4.070	5.250

Instead of counting it ourselves, we can use the following code to find the mode:

Example

```
Data_Cars <- mtcars

names(sort(-table(Data_Cars$wt)))[1]
```

Result:

```
[1] "3.44"
```

Quartiles

Quartiles are data divided into four parts, when sorted in an ascending order:

1. The value of the first quartile cuts off the first 25% of the data
2. The value of the second quartile cuts off the first 50% of the data
3. The value of the third quartile cuts off the first 75% of the data
4. The value of the fourth quartile cuts off the 100% of the data

Use the `quantile()` function to get the quartiles.

1.2 Install R packages

Packages are the fundamental units created by the community that contains reproducible R code. These include reusable R functions, documentation that describes how to use them and sample data.

The directory where packages are stored is called the library. R comes with a standard set of packages. Others are available for download and installation. Once installed, they have to be loaded into the session to be used.

To install a package in R, we simply use the command

install.packages*("Name of the Desired Package")*

1.3 Loading the Data set

There are some data sets that are already pre-installed in R. Here, we shall be using **The Titanic** data set that comes built-in R in the Titanic Package.

While using any external data source, we can use the read command to load the files(Excel, CSV, HTML and text files etc.)

This data set is also available at Kaggle. You may download the data set, both train and test files. In this tutorial, we'd be just using the train data set.

*titanic <- **read.csv**("C:/Users/Desktop/titanic.csv", header=TRUE, sep=",")*

The above code reads the file titanic.csv into a dataframe **titanic**. With Header=TRUE we are specifying that the data includes a header(column names) and sep="," specifies that the values in data are comma separated.

2. Understanding the Data set

We have used the Titanic data set that contains historical records of all the passengers who on-boarded the Titanic. Below is a brief description of the 12 variables in the data set :

- PassengerId: Serial Number
- Survived: Contains binary Values of 0 & 1. Passenger did not survive — 0, Passenger Survived — 1.
- Pclass — Ticket Class | 1st Class, 2nd Class or 3rd Class Ticket
- Name — Name of the passenger
- Sex — Male or Female
- Age — Age in years — Integer
- SibSp — No. of Siblings / Spouses — brothers, sisters and/or husband/wife
- Parch — No. of parents/children — mother/father and/or daughter, son

- Ticket — Serial Number
- Fare — Passenger fare
- Cabin — Cabin Number
- Embarked — Port of Embarkment | C- Cherbourg, Q — Queenstown, S — Southhampton

2.1 Peek at your Data

Before we begin working on the dataset, let's have a good look at the raw data.

view(titanic)

This helps us in familiarising with the data set.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.00	1	0	A/5 21171	7.2500		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.00	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.00	0	0	STON/O2. 3101282	7.9250		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.00	0	0	373450	8.0500		S
6	0	3	Moran, Mr. James	male	NA	0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54.00	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2.00	3	1	349909	21.0750		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.00	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.00	1	0	237736	30.0708		C

head(titanic,n) | tail(titanic,n)

In order to have a quick look at the data, we often use the head()/tail().

```
head(titanic,10)
 PassengerId Survived Pclass                                                Name    Sex Age SibSp Parch           Ticket    Fare Cabin Embarked
           1        0      3                             Braund, Mr. Owen Harris   male  22     1     0        A/5 21171  7.2500              S
           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0         PC 17599 71.2833   C85        C
           3        1      3                              Heikkinen, Miss. Laina female  26     0     0 STON/O2. 3101282  7.9250              S
           4        1      1        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0           113803 53.1000  C123        S
           5        0      3                            Allen, Mr. William Henry   male  35     0     0           373450  8.0500              S
           6        0      3                                    Moran, Mr. James   male  NA     0     0           330877  8.4583              Q
           7        0      1                             McCarthy, Mr. Timothy J   male  54     0     0            17463 51.8625   E46        S
           8        0      3                       Palsson, Master. Gosta Leonard   male   2     3     1           349909 21.0750              S
           9        1      3    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg) female  27     0     2           347742 11.1333              S
          10        1      2                  Nasser, Mrs. Nicholas (Adele Achem) female  14     1     0           237736 30.0708              C
```

Top 10 rows of the data set.

```
> tail(titanic)
    PassengerId Survived Pclass                                      Name    Sex Age SibSp Parch     Ticket   Fare Cabin Embarked
886         886        0      3     Rice, Mrs. William (Margaret Norton) female  39     0     5     382652 29.125              Q
887         887        0      2                     Montvila, Rev. Juozas   male  27     0     0     211536 13.000              S
888         888        1      1              Graham, Miss. Margaret Edith female  19     0     0     112053 30.000   B42        S
889         889        0      3 Johnston, Miss. Catherine Helen "Carrie" female  NA     1     2 W./C. 6607 23.450              S
890         890        1      1                     Behr, Mr. Karl Howell   male  26     0     0     111369 30.000  C148        C
891         891        0      3                       Dooley, Mr. Patrick   male  32     0     0     370376  7.750              Q
```

Bottom 5 rows of the data set.

In case we do not explicitly pass the value for n, it takes the default value of 5, and displays 5 rows.

names(titanic)

This helps us in checking out all the variables in the data set.

```
> names(titanic)
 [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"         "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
[11] "Cabin"       "Embarked"
```

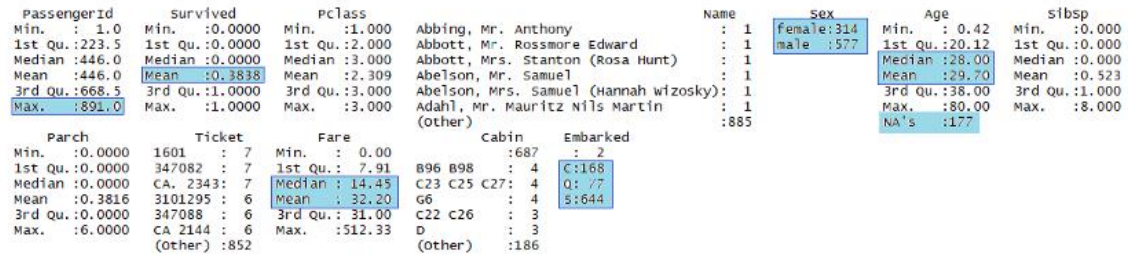
Familiarising with all the Variables/Column Names

str(titanic)

This helps in understanding the structure of the data set, data type of each attribute and number of rows and columns present in the data.

```
> str(titanic)
'data.frame':   891 obs. of  12 variables:
 $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
 $ Pclass     : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 629 417 581 ...
 $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
 $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
 $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345 133 ...
 $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
 $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

summary(titanic)



A cursory look at the data

Summary() is one of the most important functions that help in summarising each attribute in the dataset. It gives a set of descriptive statistics, depending on the type of variable:

- In case of a Numerical Variable -> Gives Mean, Median, Mode, Range and Quartiles.
- In case of a Factor Variable -> Gives a table with the frequencies.
- In case of Factor + Numerical Variables -> Gives the number of missing values.
- In case of character variables -> Gives the length and the class.

In case we just need the summary statistic for a particular variable in the dataset, we can use

summary(datasetName$VariableName) -> summary(titanic$Pclass)

as.factor(dataset$ColumnName)

There are times when some of the variables in the data set are factors but might get interpreted as numeric. For example, the Pclass(Passenger Class) tales the values 1, 2 and 3, however, we know that these are not to be considered as numeric, as these are just levels. In order to such variables treated as factors and not as numbers we need explicitly convert them to factors using the function **as.factor()**

```
titanic$Survived <- as.factor(titanic$Survived)
titanic$Pclass <- as.factor(titanic$Pclass)
titanic$Sex <- as.factor(titanic$Sex)
titanic$Embarked <- as.factor(titanic$Embarked)
```

3. Analysis & Visualisations

Data Visualisation is an art of turning data into insights that can be easily interpreted. In this tutorial, we'll analyse the survival patterns and check for factors that affected the same.

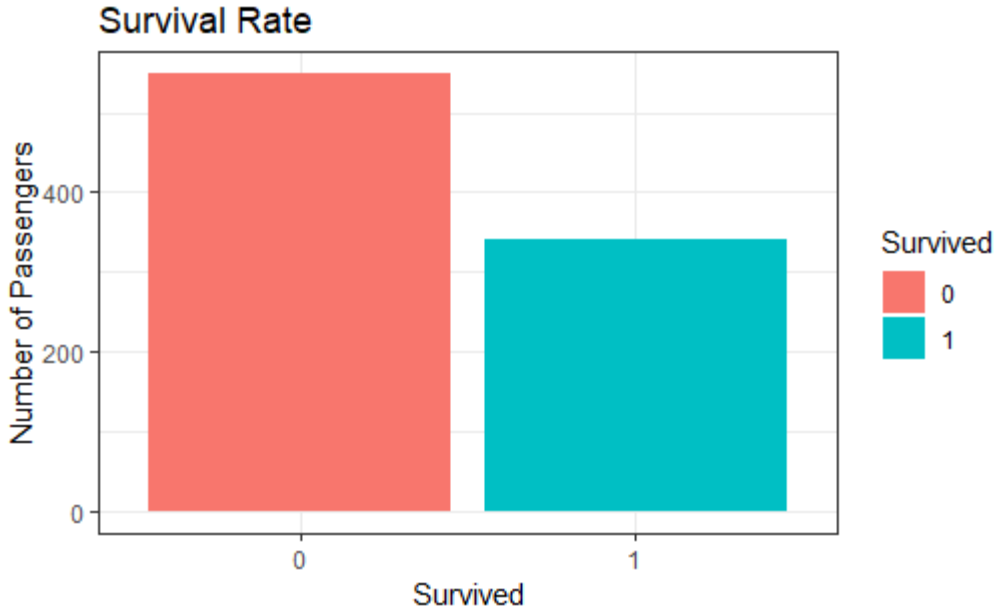
Points to think about

Now that we have an understanding of the dataset, and the variables, we need to identify the variables of interest. Domain knowledge and the correlation between variables help in choosing these variables. To keep it simple, we have chosen only 3 such variables, namely Age, Gender, Pclass.

What was the survival rate?

When talking about the Titanic data set, the first question that comes up is "How many people did survive?". Let's have a simple Bar Graph to demonstrate the same.

ggplot(titanic, aes(x=Survived)) + geom_bar()



On the X-axis we have the survived variable, 0 representing the passengers that did not survive, and 1 representing the passengers who survived. The Y -axis represents the number of passengers. Here we see that over 550 passenger did not survive and ~ 340 passengers survived.

Let's make is more clear by using checking out the percentages

prop.table(table(titanic$Survived))

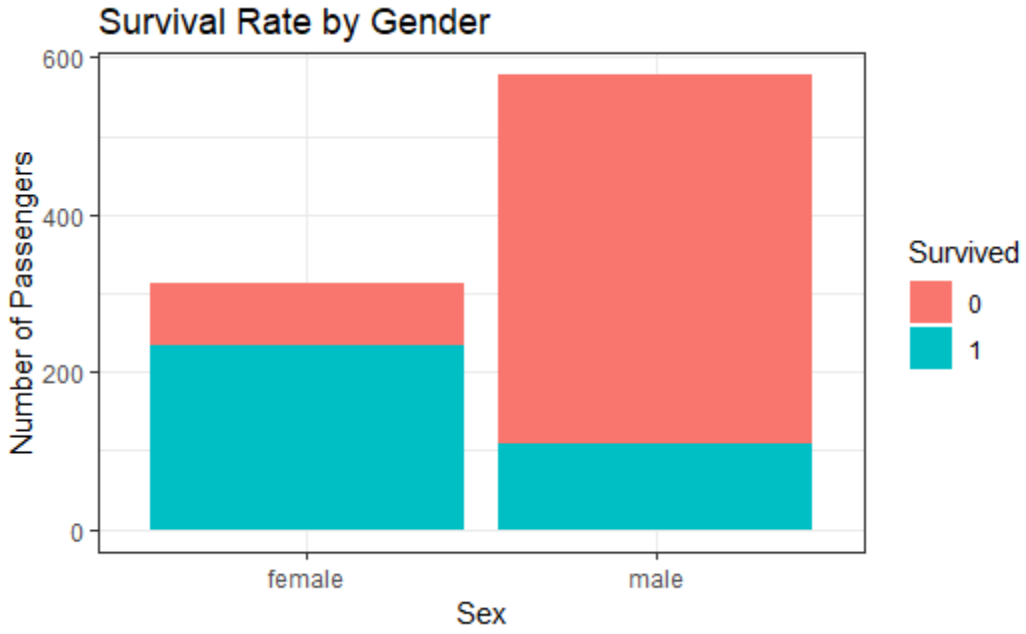
```
> prop.table(table(titanic$Survived))

        0         1
0.6161616 0.3838384
```

Only 38.38% of the passengers who on-boarded the titanic did survive.

Survival rate basis Gender

It is believed that in case of rescue operations during disasters, woman's safety is prioritised. Did the same happen back then?

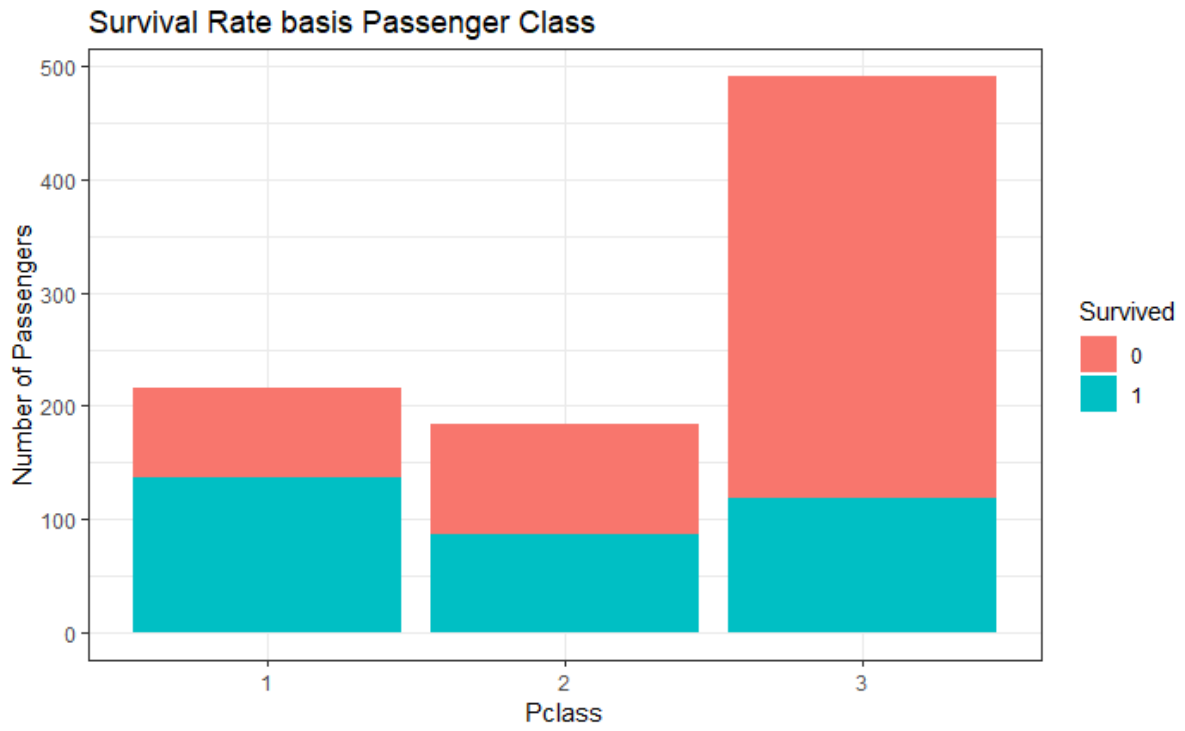



```
ggplot(titanic,aes(x=Sex, fill= Survived)) +
theme_bw() +
geom_bar() +
labs(y="Number of Passengers",
title="Survival Rate by Gender")
```

We see that the survival rate amongst the women was significantly higher when compared to men. The survival ratio amongst women was around 75%, whereas for men it was less than 20%.

Survival Rate basis Class of tickets (Pclass)

There were 3 segments of passengers, depending upon the class they were travelling in, namely, 1st class, 2nd class and 3rd class. We see that over 50% of the passengers were travelling in the 3rd class.



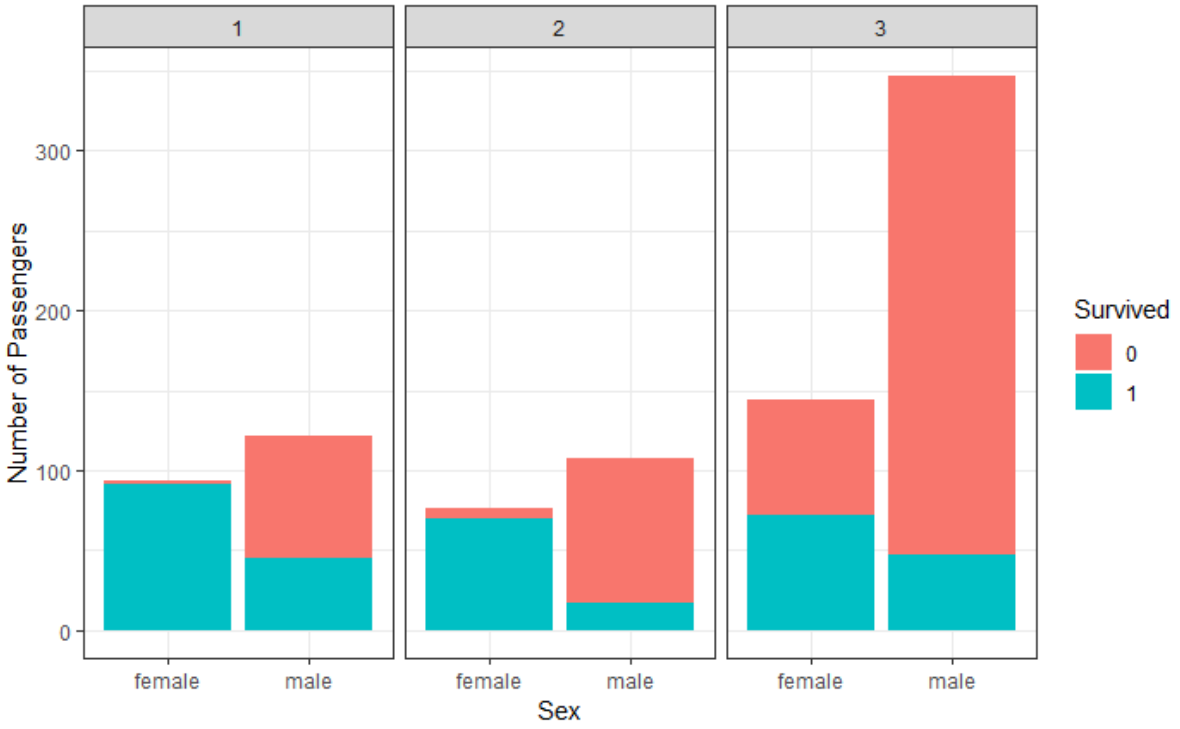
Survival Rate basis Passenger Class

```
ggplot(titanic,aes(x=Pclass,fill=Survived)) +
theme_bw() +
geom_bar() +
labs(y="Number of Passengers",
title="Survival Rate basis Passenger Class")
```

1st and 2nd Class passengers disproportionately survived, with over 60% survival rate of the 1st class passengers, around 45–50% of 2nd class, and less than 25% survival rate of those travelling in 3rd class.

I'll leave you at the thought… Was it because of a preferential treatment to the passengers travelling elite class, or the proximity, as the 3rd class compartments were in the lower deck?

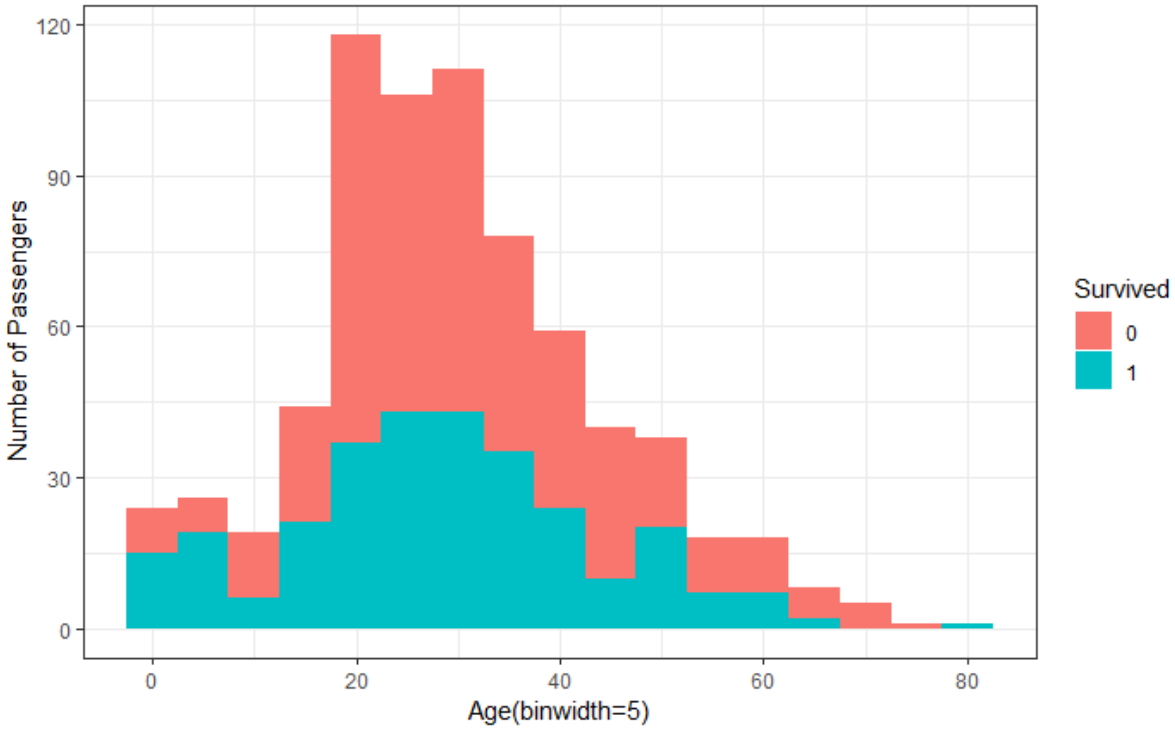
Survival Rate basis Class of tickets and Gender*(pclass)*



We see that the females in the 1st and 2nd class had a very high survival rate. The survival rate for the females travelling in 1st and 2nd class was 96% and 92% respectively, corresponding to 37% and 16% for men. The survival rate for men travelling 3rd class was less than 15%.

Till now it is evident that the Gender and Passenger class had significant impact on the survival rates. Let's now check the impact of passenger's Age on Survival Rate.

Survival rates basis age



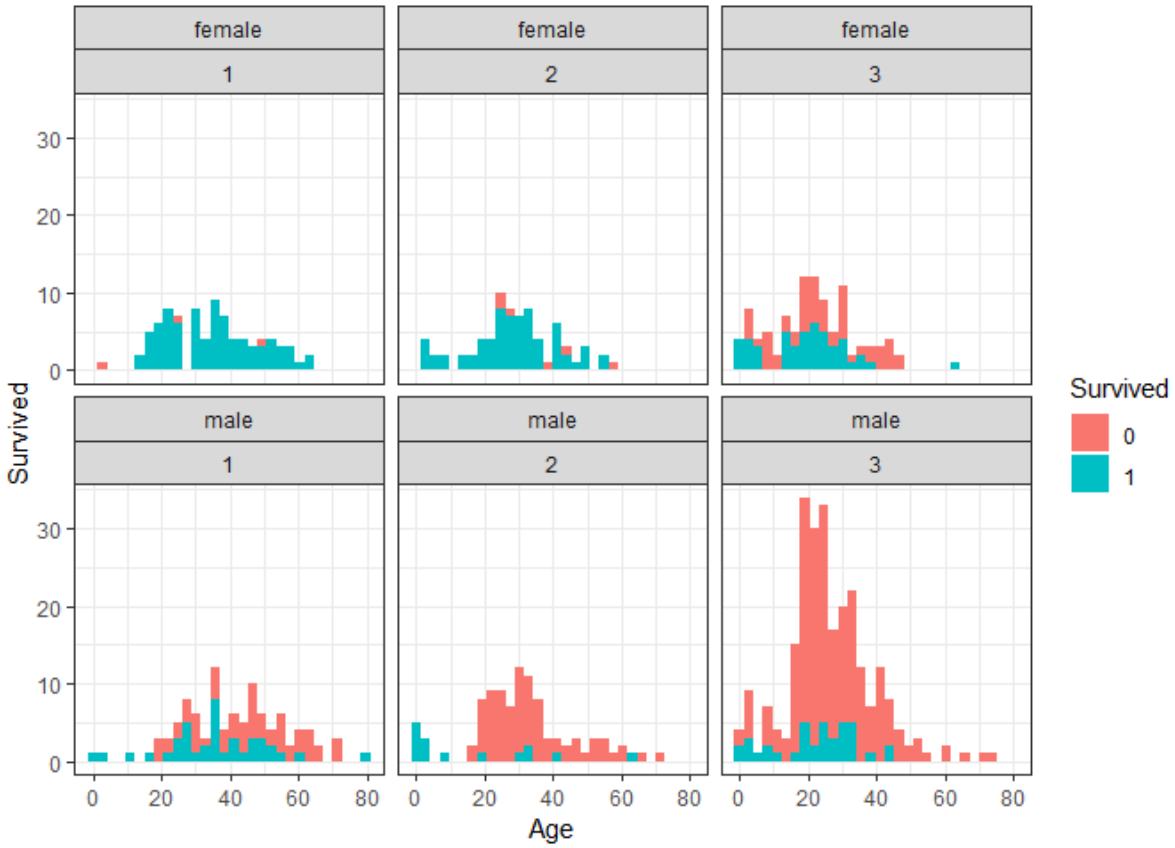

```
ggplot(titanic,aes(x=Age,fill=Survived)) +
theme_bw() +
geom_histogram(binwidth=5)+
labs(y="Number of Passengers",
x="Age")
```

Looking at the age<10 years section in the graph, we see that the survival rate is high. And the survival rate is low and drops beyond the age of 45.

Here we have used bin width of 5, you may try out different values and see, how the graph changes.

Survival Rate basis Age, Gender and Class of tickets

This graph helps identify the survival patterns considering all the three variables.




```
ggplot(titanic,aes(x=Age,fill=Survived)) +
theme_bw() +
facet_wrap(Sex~Pclass)+
geom_histogram(bnwidth=5) +
labs(y="Survived",
x="Age")
```

The top 3 sections depict the female survival patterns across the three classes, while the bottom 3 represent the male survival patterns across 3 classes. On the x-axis we have the Age.