Project

Jason Ballantyne

21/12/2021

```
# Loading dataset for crime in Ireland between 2003 - 2019
# source: https://www.kaggle.com/sameerkulkarni91/crime-in-ireland/version/1
crime = read.csv("IRELAND_CRIME_GARDA_DIVISION_wise_2003-2019.csv")
```

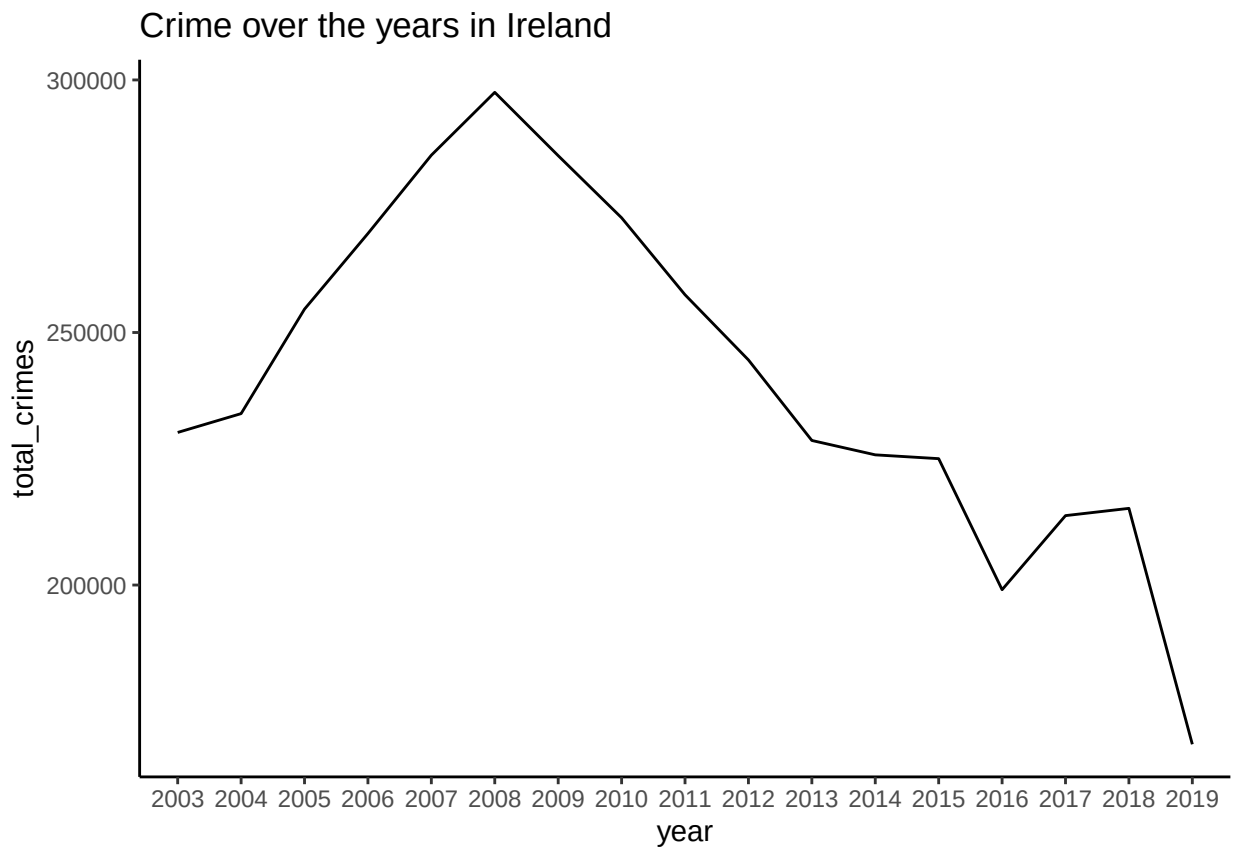
Part 1: Analysis

The data set chosen contains complete information of all types of crimes committed in Ireland from the year 2003 to 2019.This data set in its original form has been taken from StatBank, Central Statistics Office, Govt. of Ireland website - > https://statbank.cso.ie/

```
library(tidyverse)
library(reshape2)

# Convert dataframe from wide to long
clean = melt(crime, measure.vars = colnames(crime)[6:ncol(crime)])

# Remove preceding X and proceeding Q1,Q2,Q3
to_remove = c("X", "Q1", "Q2", "Q3")

clean = crime %>%
  melt(., measure.vars = colnames(crime)[6:ncol(crime)]) %>%
  mutate(variable = str_remove_all(variable, "X")) %>%
  mutate(variable = substr(variable,1,4)) %>%
  rename(year = variable)
```

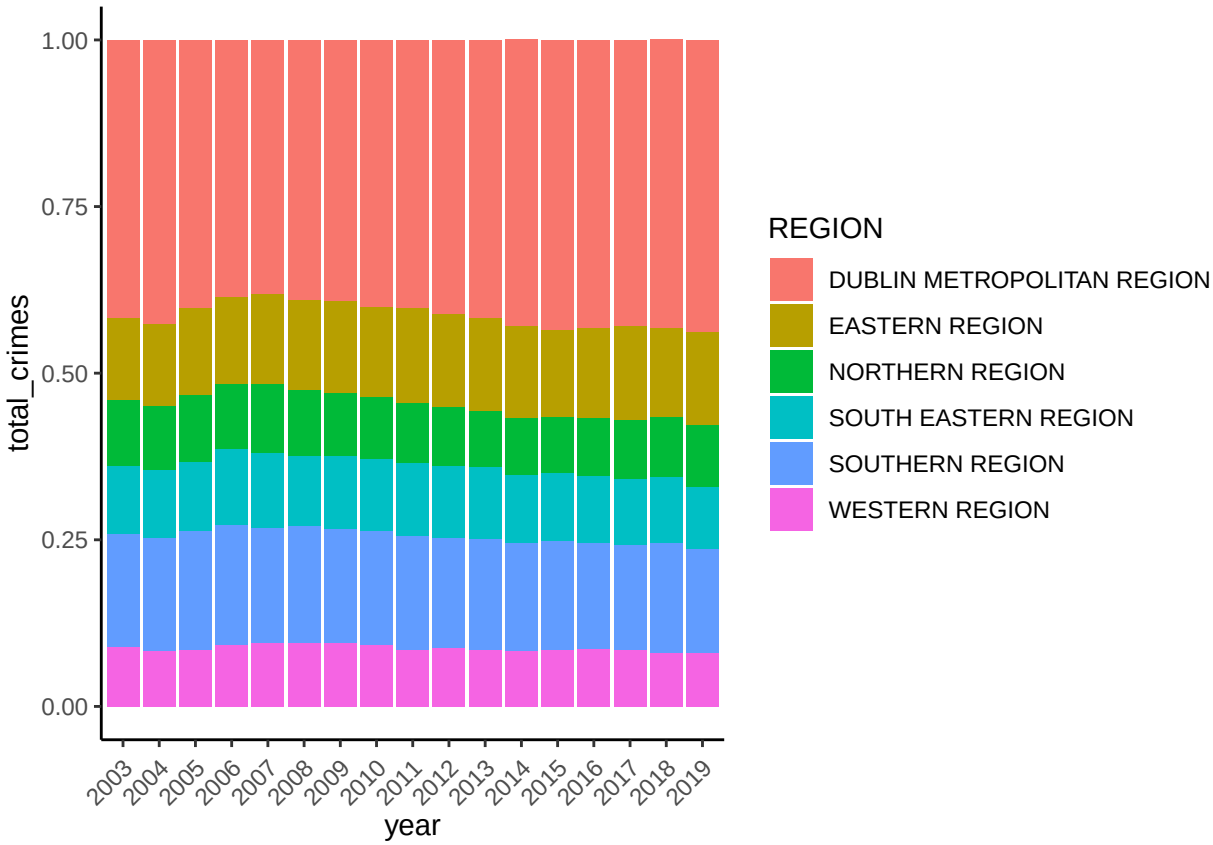
The purpose of the above step is to transform all value variables from a wide to a long dataframe. This will be essential in order to create summaries and visualizations. We also removed the preceding "X" and proceeding "Q1,Q2,Q3" so that only year is left.

```
# Overall crime for each year
clean %>%
  group_by(year) %>%
  summarize(total_crimes = sum(value)) %>%
  ggplot()+
  geom_line(aes(x = year, y = total_crimes, group =1)) +
  theme_classic() +
  ggtitle("Crime over the years in Ireland")
```



From the above results, we can conclude that the overall crime rate had risen massively during the period of 2003-2008. The reported cases peaked at around 300K per year in 2008. After that the trend has been downwards. Crime rate in 2012 was back to circa 2003 numbers and fortunately 2019 has seen the lowest crime reports in the period of 17 years or ~ 2 decades.

```
# breakdown of crime by region
clean %>%
  group_by(REGION, year) %>%
  summarize(total_crimes = sum(value)) %>%
  ggplot() +
  geom_bar(aes(x = year, y = total_crimes, fill = REGION),
           stat = 'identity', position = "fill") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.9 , hjust=1))
```



In the above analysis, we have broken down crime by region. From these results, we can see that there has been relatively consistent percentages across all years. The highest percent of crimes happen in Dublin where ~35%-40% of all crime occurs. This is followed by Southern Region where ~15%-20% of crime occurs. The least amount of crime happens in the North and West region (about 10%). Data such as this can aid law enforcement in identifying areas where it is necessary to deploy more resources.

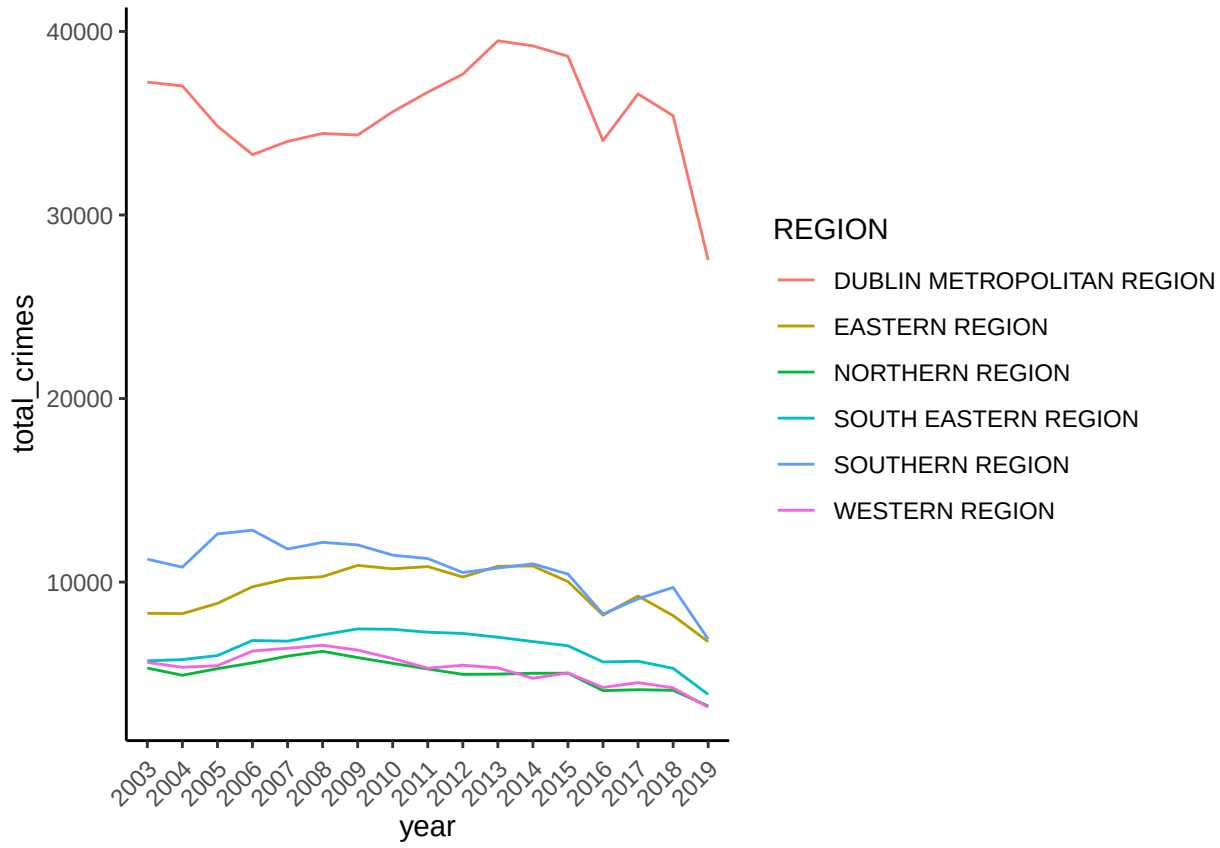
```
# breakdown of crime by offence
clean %>%
  group_by(TYPE.OF.OFFENCE) %>%
  summarize(total_crimes = sum(value)) %>%
  arrange(desc(total_crimes)) %>%
  top_n(total_crimes, 5)
```

```
## # A tibble: 14 x 2
##    TYPE.OF.OFFENCE                                                total_crimes
##    <chr>                                                                 <int>
##  1 THEFT AND RELATED OFFENCES                                          1235434
##  2 PUBLIC ORDER AND OTHER SOCIAL CODE OFFENCES                          752079
##  3 DAMAGE TO PROPERTY AND ENVIRONMENT                                   555281
##  4 BURGLARY AND RELATED OFFENCES                                        408499
##  5 ATTEMPTS/THREATS TO MURDER/ASSAULTS/ HARASSMENTS AND RELATED OF~      282305
##  6 CONTROLLED DRUG OFFENCES                                             278482
##  7 DANGEROUS OR NEGLIGENT ACTS                                          196678
##  8 OFFENCES AGAINST GOVERNMENT/ JUSTICE PROCEDURES AND ORGANISATIO~      181680
##  9 FRAUD/DECEPTION AND RELATED OFFENCES                                  87202
## 10 WEAPONS AND EXPLOSIVES OFFENCES                                       48642
```

```
## 11 ROBBERY/EXTORTION AND HIJACKING OFFENCES                    42990
## 12 SEXUAL OFFENCES                                             33776
## 13 KIDNAPPING AND RELATED OFFENCES                              1865
## 14 HOMICIDE OFFENCES                                            1729
```

From the results above, we can see the most common type of crime is theft followed by public order offences and then damage to property. These type of detailed insights into crime can help improve training within An Garda Siochana. The training can be tailored to dealing with theft, for example. Although rigorous and thorough training is essential, having an understanding of the most commonly committed crimes can help prepare enforcement adequately.

```
# breakdown of crime by region filtered for theft and related offences
clean %>%
  filter(TYPE.OF.OFFENCE == "THEFT AND RELATED OFFENCES") %>%
  group_by(year, REGION) %>%
  summarize(total_crimes = sum(value), .groups = "drop") %>%
  arrange(desc(total_crimes)) %>%
  ggplot() +
  geom_line(aes(x = year, y = total_crimes, color = REGION, group = REGION)) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.9 , hjust=1))
```



Given that theft and related offences was the most common crime, I have filtered for this type of crime across all regions to analyse how it performs across the years. From the graph, we can see that the Dublin Metropolitan Region is by far the greatest offender of this type of crime. The Northern and Western Region appear to be the least common offenders of this type of crime. Trends for areas outside of the Dublin

Metropolitan Region have seen a slow but steady decline since 2003 to 2019. However, for the region of Dublin, we can see a steady increase from 2005 - 2013, where there appears to be a shift as a sharp decline occurs leaving 2019 with the lowest count of crime for the Dublin Metropolitan Region between 2003 - 2019. The root of this cannot be determined from the graph, and could have been caused by a variety of factors such as reduced social issues due to increased employment, increase of enforcement, increase of quality of life etc. Various theories can be identified but we would need much more data to prove our hypothesis.

To conclude our analysis, from the crime data in Ireland between 2003 - 2019, we have looked at overall crime for each year, a breakdown of crime by region, a breakdown of crime by offence and ultimately a breakdown of crime by region filtered for theft and related offences. For each of these graphs, we summarised the results of the graphs which generated valuable insights of the data.

Part 2: R Package

The package that I will be demonstrating will be **dplyr** which is a part of Tidyverse family. This package is essential as it helps the user transform the data using variety of techniques. The package is efficient and works with a lot of speed. The syntax is different from base R and resembles more of a SQL like structure. The pipe operator %>% helps maintain the flow of the code and allows readability. The package contains a set of functions (or "verbs") that perform common data manipulation operations such as filtering for rows, selecting specific columns, adding new columns and summarizing data which we will now discuss in detail.

The essential functions of this package that we will be looking at more closesly are: **filter(), mutate(), select() and summarize()**.

The **filter** function allows you to return a subset of rows meeting a particular condition (or not meeting a condition). So practically, you can remove unwanted rows containing NAs or any value of your choice. The first argument is the tibble whereas the second and subsequent arguments refer to the variables within that tibble or data frame, selecting the subset of rows where the expression is TRUE.

It is often useful to add new columns that are functions of existing columns, this is the job of the **mutate** function. It helps to create a new column or re-work an existing one. You can create a new column by adding col1 + col2 for example. Or you can pass on boolean values if a value meets certain criteria. It is similar to the base transform(), but instead, allows you to refer to columns that you've just created. This opens limitless opportunities to mutate data.

Often times you might work with a large dataset with many columns, but only a few columns are of interest to you, this is where **select** can come in handy.**Select** allows you to speedily zoom in on a useful subset using operations that usually only work on numeric variable positions. There are a number of helper functions you can use alongside **select**, some of these include: starts_with(), ends_with(), matches() and contains(). These helper functions allow you to quickly match larger blocks of variables that meet a certain criteria. It is important to note that **select** drops all variables that are not explicitly mentioned.

Finally, the **summarize** function is best utilized in conjunction with the group_by() function. After combining different columns, you may need aggregations like count, average, sum through which this function can help you. The output will have a single row summarizing all observations in the input.**Summarize** will contain one column for each grouping variable and one column for each of the summary statistics that you have specified.

For demonstration purposes, let's look at the most popular iris dataset.

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
```

```
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
iris %>%
  # using filter to only select setosa species
  filter(Species == "setosa") %>%

  # using mutate to add petal.length and width together
  mutate(Petal.Sum = Petal.Length + Petal.Width) %>%

  # using select to remove Sepal Length and Sepal Width from our table
  select(-Sepal.Length, -Sepal.Width)
```

```
##    Petal.Length Petal.Width Species Petal.Sum
## 1           1.4         0.2  setosa       1.6
## 2           1.4         0.2  setosa       1.6
## 3           1.3         0.2  setosa       1.5
## 4           1.5         0.2  setosa       1.7
## 5           1.4         0.2  setosa       1.6
## 6           1.7         0.4  setosa       2.1
## 7           1.4         0.3  setosa       1.7
## 8           1.5         0.2  setosa       1.7
## 9           1.4         0.2  setosa       1.6
## 10          1.5         0.1  setosa       1.6
## 11          1.5         0.2  setosa       1.7
## 12          1.6         0.2  setosa       1.8
## 13          1.4         0.1  setosa       1.5
## 14          1.1         0.1  setosa       1.2
## 15          1.2         0.2  setosa       1.4
## 16          1.5         0.4  setosa       1.9
## 17          1.3         0.4  setosa       1.7
## 18          1.4         0.3  setosa       1.7
## 19          1.7         0.3  setosa       2.0
## 20          1.5         0.3  setosa       1.8
## 21          1.7         0.2  setosa       1.9
## 22          1.5         0.4  setosa       1.9
## 23          1.0         0.2  setosa       1.2
## 24          1.7         0.5  setosa       2.2
## 25          1.9         0.2  setosa       2.1
## 26          1.6         0.2  setosa       1.8
## 27          1.6         0.4  setosa       2.0
## 28          1.5         0.2  setosa       1.7
## 29          1.4         0.2  setosa       1.6
## 30          1.6         0.2  setosa       1.8
## 31          1.6         0.2  setosa       1.8
```

```
## 32          1.5         0.4  setosa       1.9
## 33          1.5         0.1  setosa       1.6
## 34          1.4         0.2  setosa       1.6
## 35          1.5         0.2  setosa       1.7
## 36          1.2         0.2  setosa       1.4
## 37          1.3         0.2  setosa       1.5
## 38          1.4         0.1  setosa       1.5
## 39          1.3         0.2  setosa       1.5
## 40          1.5         0.2  setosa       1.7
## 41          1.3         0.3  setosa       1.6
## 42          1.3         0.3  setosa       1.6
## 43          1.3         0.2  setosa       1.5
## 44          1.6         0.6  setosa       2.2
## 45          1.9         0.4  setosa       2.3
## 46          1.4         0.3  setosa       1.7
## 47          1.6         0.2  setosa       1.8
## 48          1.4         0.2  setosa       1.6
## 49          1.5         0.2  setosa       1.7
## 50          1.4         0.2  setosa       1.6
```

From our results above, we have demonstrated the functionality of filter, mutate and select. Filter was used to only display the species "setosa". Mutate was used to add length and width together to create a new column called "Petal.sum". Select was used to remove "Sepal.Length" and "Sepal.Width" from our table.

```
# using summarize function now to demonstrate average petal length and
# sum petal length of all species
iris %>%
  group_by(Species) %>%
  summarize(Avg.Petal.Length = mean(Petal.Length),
            Sum.Petal.Length = sum(Petal.Length))
```

```
## # A tibble: 3 x 3
##   Species    Avg.Petal.Length Sum.Petal.Length
##   <fct>                 <dbl>            <dbl>
## 1 setosa                 1.46             73.1
## 2 versicolor             4.26            213
## 3 virginica              5.55            278.
```

In the demonstration above, we used a simple group_by followed by the summarize function. The summarize function was used to find the average petal length and sum of petal length by species.

To conclude, we have clearly summarised the purpose of the package. We have given a detailed description as well as demonstrated the functionality of some of the main functions using the iris dataset. Finally, the code and output was shown as well as a description of the demonstration examples.

Part 3: Functions

The function I have created to provide statistical analysis of interest is a function that calculates the confidence interval of a proportion where p is the sample proportion and n is the sample size. The confidence interval percentage is 95% by default.

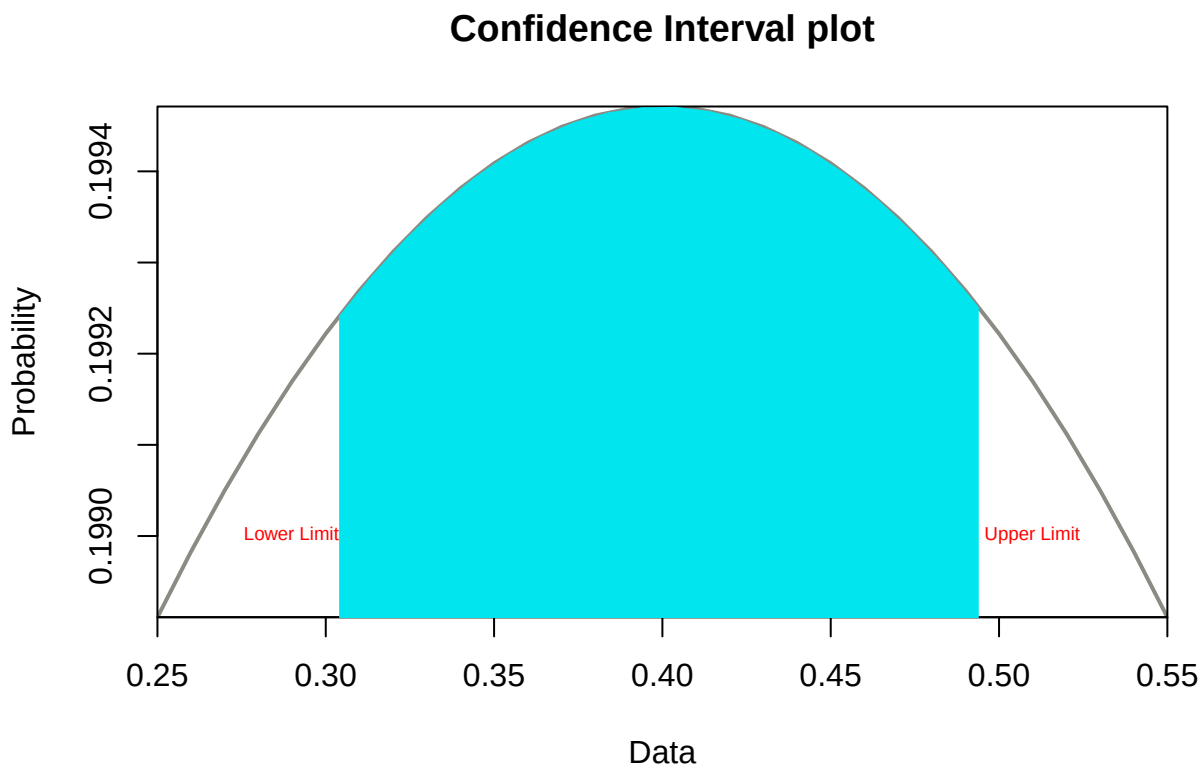
```r
# Sample proportion is set to 0.4 and the sample size is set to 100.
s = list(name = "Experiment1", p = 0.4, n = 100)

# Turn object into a class called CI_proportion
new_CI_proportion <- function(lst){
    structure(lst, class = "CI_proportion" )
}

# Turning s variable into a class s_data
s_data <- new_CI_proportion(s)

# Providing appropriate print method
print.CI_proportion <- function(s) {
  # Generating lower and upper limits
  p.lower = s$p - 1.96 * sqrt(s$p*(1-s$p)/s$n)
  p.upper = s$p + 1.96 * sqrt(s$p*(1-s$p)/s$n)
  result = c(p.lower,p.upper)

  # Concatenating and printing
  cat(s$name, "\n")
  cat("Lower bound:", result[1], "\n")
  cat("Upper bound:", result[2])
}

# Passing the s_data through our print method
print(s_data)
```

```
## Experiment1
## Lower bound: 0.30398
## Upper bound: 0.49602
```

The result above displays the lower and upper limits of the confidence interval.

```r
# Providing appropriate plot method
plot.CI_proportion <- function(s) {
  # Generating lower and upper limits
  p.lower = s$p - 1.96 * sqrt(s$p*(1-s$p)/s$n)
  p.upper = s$p + 1.96 * sqrt(s$p*(1-s$p)/s$n)
  result = c(p.lower,p.upper)

  # define the standard deviation
  std = 2

  # create vector for values of x that span a sufficient range of
  # standard deviations on either side of the mean
  x = seq(0.25, 0.55, 0.01)

  # use dnorm() to calculate probabilities for each x
  y = dnorm(x, mean = mean(x), sd = std)

  # plot normal distribution curve; the options xaxs = "i" and yaxs = "i"
  # force the axes to begin and end at the limits of the data
  plot(x, y, type = "l", lwd = 2, col = "ivory4",
```

```
       main = "Confidence Interval plot",ylab = "Probability",
       xlab = "Data", xaxs = "i", yaxs = "i")

  # Adding text to show the lower and upper limits of the graph
  text(0.27, 0.1990, "Lower Limit", cex=0.6, pos=4, col="red")
  text(0.49, 0.1990, "Upper Limit", cex=0.6, pos=4, col="red")

  # Setting the lower limit based on our p.lower variable
  lowlim = p.lower

  # Setting the upper limit based on our p.upper variable
  uplim = p.upper

  dx = seq(lowlim, uplim, 0.01)

  # use polygon to fill in area; x and y are vectors of x,y coordinates
  # that define the shape that is then filled using the turquoise2 color
  polygon(x = c(lowlim, dx, uplim),
          y = c(0, dnorm(dx, mean = mean(x), sd = 2), 0),
          border = NA, col = "turquoise2")
}

# Passing the s_data through our plot method
plot(s_data)
```



This method uses R to model the properties of a Normal Distribution. The upper and lower limits were

used to plot this graph.

```
# Providing appropriate summary method
summary.CI_proportion <- function(s) {
  # Generating lower and upper limits
  p.lower = s$p - 1.96 * sqrt(s$p*(1-s$p)/s$n)
  p.upper = s$p + 1.96 * sqrt(s$p*(1-s$p)/s$n)
  result = c(p.lower,p.upper)


  # Summary of our confidence interval function,concatenating and printing
  cat(s$name, "Summary: \n")
  cat("--------------------\n")
  cat("Minimum Value:", min(result), "\n")
  cat("1st Quartile:", quantile(result, 0.25), "\n")
  cat("Median Value:", median(result), "\n")
  cat("Mean Value:", mean(result), "\n")
  cat("3rd Quartile:", quantile(result, 0.75), "\n")
  cat("Max Value:", max(result), "\n")
}

# Passing the s_data through our summary method
summary(s_data)
```

```
## Experiment1 Summary:
## --------------------
## Minimum Value: 0.30398
## 1st Quartile: 0.35199
## Median Value: 0.4
## Mean Value: 0.4
## 3rd Quartile: 0.44801
## Max Value: 0.49602
```

The result above provides a summary of the confidence interval by displaying the Minimum Value, 1st Quartile, Median Value, Mean Value, 3rd Quartile and Max Value.

In conclusion, we have provided a working function of the confidence interval which is our analysis of interest. Methods have been created for print, summary and plot and the code has been clearly commented.