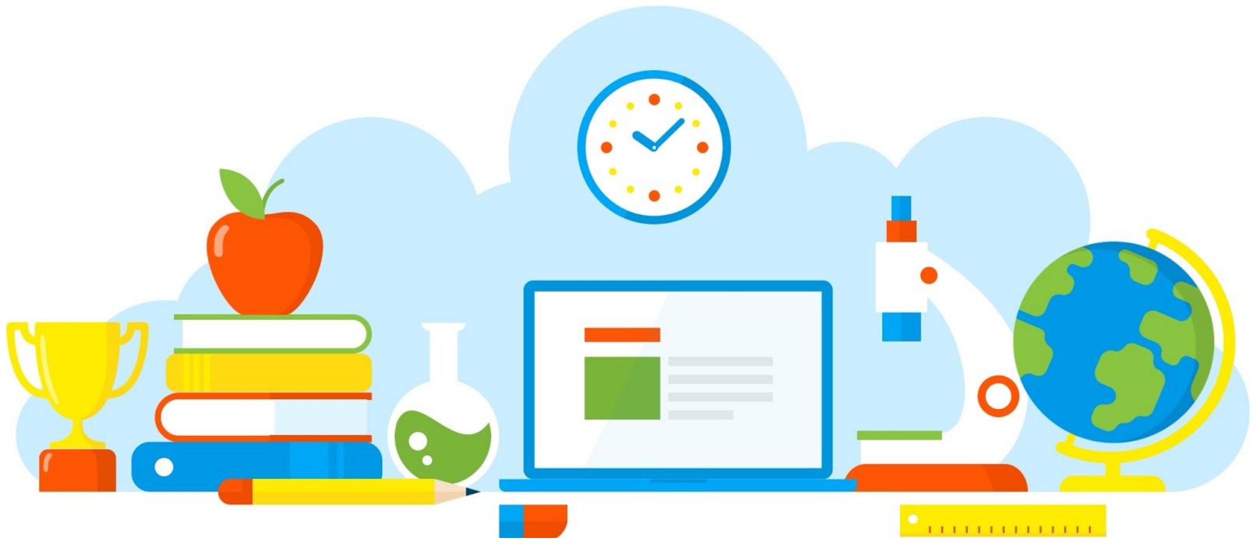


Data Science

Analysing Suicides in India

Aekansh Dixit
PES1201701808 (3C)

Introduction

I decided to analyse the Suicides In India from 2001 to 2012 and inferred some meaningful insights from the data gathered. The data set was downloaded from the internet and underwent cleaning in order to properly understand the analytics. There are both categorical data, as well as numeric data making it a challenge to properly interpret the same.

The Project Details

Dataset: Suicides In India (2001 - 2012)

Programming Language Used: R in RStudio

GitHub Link: Crazytics - Suicides In India

Total Rows: 238,000

Total Columns: 7

State	Year	Type_code	Type	Gender	Age_group	Total
A & N Islands	2001	Causes	Illness (Aids/STD)	Female	0-14	0
A & N Islands	2001	Causes	Bankruptcy or Sudden change in Economic	Female	0-14	0
A & N Islands	2001	Causes	Cancellation/Non-Settlement of Marriage	Female	0-14	0
A & N Islands	2001	Causes	Physical Abuse (Rape/Incest Etc.)	Female	0-14	0
A & N Islands	2001	Causes	Dowry Dispute	Female	0-14	0

The Dataset Downloaded in CSV

Gathering and Understanding the Dataset

The dataset can be downloaded from the given link above. Upon opening the **.csv** file, we realise the format of the dataset:

It seems to have 7 columns, namely:

1. **State** in which the data was recorded.
2. **Year** in which the data was recorded.
3. **Type_Code** is the type of data recorded, which can be broken into:
    a. **Causes:** The cause of the suicide.
    b. **Education Status:** The education status of the profile.
    c. **Means Adopted:** How did the profile commit suicide?
    d. **Professional Profile:** What was the professional status of the profile?
    e. **Social Status:** Was the profile married/divorced/Never Married, etc?
4. **Type** is the information of the categorical data presented in **3**.
5. **Gender** is the sex of the profile.
6. **Age Group** is the age group the profile falls under.
    a. **0-14**
    b. **15-29**
    c. **30-44**
    d. **45-59**
    e. **60+**
    f. **0-100+** is the special category for the profiles testing **Education Status** and **Social Status**.
7. **Total** is the number of profiles that committed suicide under that category.

Going through the dataset, it is clear that a lot of categorical data is clustered and that we need to clean the data in order for us to find some meaningful analysis. The only column that needs to be focused on is **Total**.

Questions to Answer

We'll be looking to answer 3 different questions using this dataset:

1) What is the trend of suicides committed by the females in India over the years?
2) Which age group is more prone to commit suicides?
3) What is the major cause of suicides in India?

Getting Started

First Steps in R

As already guessed, the first step in R is to import the dataset. Thankfully, R makes it really easy to import CSV datasets. But before we do that, we need to install a few packages that we'll require later:

```
> install.packages("dplyr")
> install.packages("magrittr")
> install.packages("lubridate")
> install.packages("zoo")
> install.packages("ggplot2")
> install.packages("ggpubr")
> install.packages("car")
> install.packages("fastDummies")
```

Now that we have our packages ready, we're ready to begin. Let's go ahead and import our csv into a dataframe in R.

```
> library(dplyr)
> library(magrittr)

# Import Dataset
> dataset <- read.csv("dataset.csv")
```

Now, let's select a few columns and see their listing in RStudio:

```
# Select State (Karnataka), Type, and Total Deaths
> dataset %>%
+     select(State,Type,Total) %>%
+     filter(State == "Karnataka") %>%
+     head()
```

This will give us the following output:

	State	Type	Total
1	Karnataka	Insanity/Mental Illness	11
2	Karnataka	Causes Not known	42
3	Karnataka	Property Dispute	0
4	Karnataka	Drug Abuse/Addiction	5
5	Karnataka	Cancer	0
6	Karnataka	Fall in Social Reputation	0

Now that our CSV is working properly, let's move on to the next step.

Checking for missing value

In order to check for the missing data, we will take the help of this code:

```
# Check for missing values
> apply(dataset, 2, function(x) any(is.na(x)))
```

This will show the following output:

State	Year	Type_code	Type	Gender	Age_group	Total
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

What this means is that our dataset is complete and there are no missing values. However, for the sake of this project demonstration, we will **deliberately** delete a few values in our dataset.

The best column that qualifies for this manipulation is the gender since

1. Age_group is a groups together multiple profiles. (Grouped data)
2. Total counts the number of deaths by that type_code. (Grouped data)
3. Type_Code in turn is used to find out the profile of the person. (Grouped data)

Let's first find out the number of entries in our dataset:

```
# Find out number of entries
> nrow(dataset)
[1] 237519
```

We will now remove **10% of our data randomly**, which means we will remove 23715 rows of data.

To select 10% of data randomly, we execute this:

```
# take 10% sample
> sampleData <- dataset[sample(nrow(dataset), 0.1*nrow(dataset)), ]
> sampleDataRowNumbers <- as.numeric(rownames(sampleData))
```

The second line extracts the row numbers of the sampled data. We will use these row numbers to delete data from the original dataset. But before we proceed, let's check the size of this dataset.

```
# Find out number of entries in sample
nrow(sampleData)
[1] 23751
```

This number is in sync with the number we calculated earlier, so we are on the right track. Let's proceed by deleting these values from the original dataset.

```
# Delete the data from 5th column
> dataset[sampleDataRowNumbers,5] <- NA
```

This line will make all the sampled rows' **gender** column as NA. Let's go ahead and test for the missing values once again.

```
# Check for missing values
> apply(dataset, 2, function(x) any(is.na(x)))
```

State	Year	Type_code	Type	Gender	Age_group	Total
FALSE	FALSE	FALSE	FALSE	**TRUE**	FALSE	FALSE

Notice how the "Gender" column now shows TRUE, which means we have some missing values.

Dealing with missing values

The Project Guidelines states that:

- All the NAN's for categorical columns to be replaced with its previous row values. **(1 mark)**
- All the NAN's for numeric columns to be replaced with average of the column. **(1 mark)**

Before we do so, let's view our missing value data:

```
# View all the columns of the sample data
> dataset[sampleDataRowNumbers,]
```

This will output the following:

State	Year	Type_code	Type	Gender	Age_group	Total
A & N Islands	2007	Social_Status	Married	<NA>	0-100+	33
Kerala	2012	Causes	Suspected/Illicit Relation	<NA>	30-44	2
Assam	2012	Means_adopted	By Consuming Other Poison	<NA>	30-44	63

We can see that some of the genders are missing. We have many different ways of dealing with **NA** but for now, we will stick to the guidelines.

Luckily, R has a special function called **na.locf()** which is Last Observation Carried Forward. Let's quickly replace the **NA** values:

```
# Replace the value with the previous value
> fixedDataset <- dataset %>% mutate(Gender = na.locf(Gender))
```

This will replace all the missing values with the values that occurred just before them. Let's view the same rows again:

```
# View all the columns of the sample data
> fixedDataset[sampleDataRowNumbers,]
```

State	Year	Type_code	Type	Gender	Age_group	Total
A & N Islands	2007	Social_Status	Married	Female	0-100+	33
Kerala	2012	Causes	Suspected/Illicit Relation	Female	30-44	2
Assam	2012	Means_adopted	By Consuming Other Poison	Male	30-44	63

And that is how we fix our dataset.

Fixing Categorical Data

The categorical values often do not help in finding insights in a given dataset. It is best to then split them using dummy variables and then assign them numeric values of 0 or 1. This makes it easier for us to visualise our data and find meaningful insights.

```
# Encode all the categorical data in dummy variables
> library(fastDummies)
> encodedData <- fastDummies::dummy_cols(fixedDataset)
```

The code above quickly creates dummy variables for each level of category present in the data set. For example,

1. **Gender** is broken down into:
    a. **Gender_Female** and given a score of 1 if the initial category was Female, or 0 otherwise.
    b. **Gender_Male** and given a score of 1 if the initial category was Male, or 0 otherwise.

Let's have a look at the columns produced by this encoding (not all of them since my dataset is huge) each having **only 1 or 0** as their values:

1. State_Uttarakhand
2. State_West Bengal
3. Type_code_Causes
4. Type_code_Education_Status
5. Type_code_Means_adopted
6. Type_code_Professional_Profile
7. Type_code_Social_Status
8. Type_Illness (Aids/STD)
9. Type_Bankruptcy or Sudden change in Economic
10. Type_Cancellation/Non-Settlement of Marriage

Finding number of female suicides

To find the number of female suicides over the years, let's filter our dataset:

```
# Number of women committing suicide regardless of cause
> womenDied <- encodedData %>%
+  select(Year, Gender_Female, Type_code, Total) %>%
+  filter(Gender_Female == 1, Total > 0, Type_code == "Causes")
```

This code will produce the following output:

Year	Gender_Female	Type_code	Total
2001	1	Causes	1
2001	1	Causes	1
2001	1	Causes	8
2001	1	Causes	8
2001	1	Causes	6
2001	1	Causes	5
2001	1	Causes	2
2001	1	Causes	2
2001	1	Causes	3
2001	1	Causes	2
2001	1	Causes	1
2001	1	Causes	1
2001	1	Causes	2

This gives us the **total** number of women that commit suicide in each category.

The next step is to group this data by year, so we can find out the number of suicides in a each year.

```
# Number of women committing suicide per year
> aggWomen <- aggregate(womenDied$Total, by=list(Category=womenDied$Year),
FUN=sum)
```

This will give us the **total number of suicides per year.**

Category	x
2001	42307
2002	41531
2003	40748
2004	41196
2005	41476
2006	42753
2007	43597
2008	45791
2009	45685
2010	47434
2011	48158
2012	40783

Normalising and Standardising Data

It is important for us to normalise this data. Normalisation helps us be consistent in comparing our data with theoretical values. In a normalised data, the mean is zero, and the variance is 1.

To normalise our data, we use the following formula:

$$Zi = \frac{xi - min(x)}{max(x) - min(x)}$$

Where x = (x1, x2, … xn) and Zi is the ith normalised data.

The following code now helps us in normalising this data:

```
# Normalise the data using formula
> womenDiedNorm <- as.data.frame(apply(as.data.frame(aggWomen[,2]), 2, function(x)
(x-min(x))/(max(x)-min(x))))
```

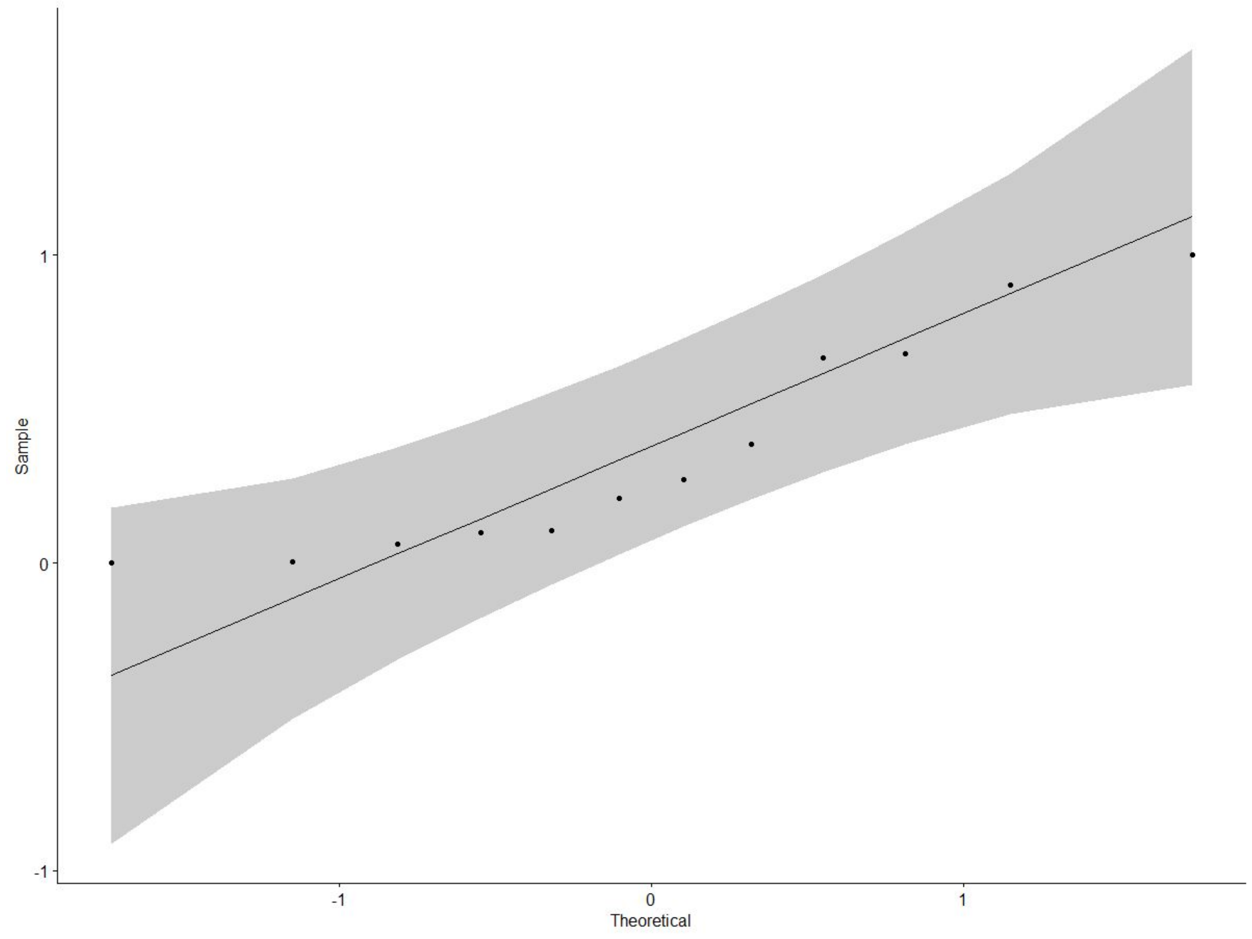
This will normalise our data between [0,1] so that it’s easier to scale it later, if needed.

Now, once we’re done, let’s test our normalised data by visual inspection.

Q-Q Plot

```
# Plot a Q-Q Graph for the normalised data
> library(ggpubr)
> ggqqplot(womenDiedNorm$`aggWomen[, 2]`)
```

Visually, we can study the impact of the parent distribution of any sample data, by using normal quantile plots. Normal Quantile-Quantile plot for sample ‘x’:



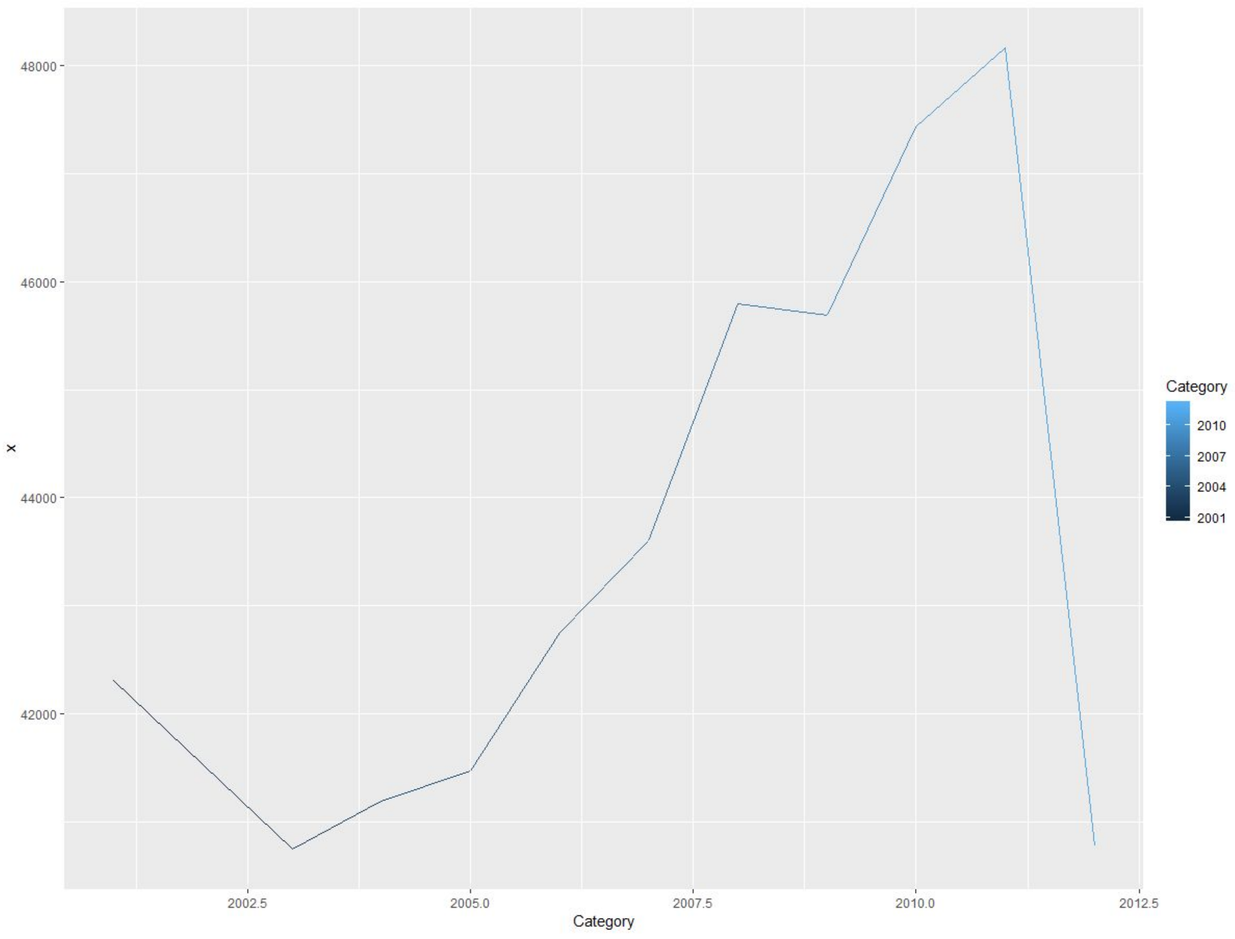
Our samples are very close to the line, and hence we can conclude that this is a normalised data.

What is the trend of suicides committed by the females in India over the years?

Now, let's get ready to answer our first question. We have gathered enough data to make conclusions out of them.

```
# Plot the trend of women commiting suicide
> library(ggplot2)
> ggplot(data = aggWomen, aes(x=Category, y=x, col=Category)) +
+  geom_line()
```

This will give us the following graph:



Hence, we conclude that the female suicides had its peak in 2011, and then a drastic drop in 2012. This is, in fact, in sync with the report published by National Crime Records Bureau.

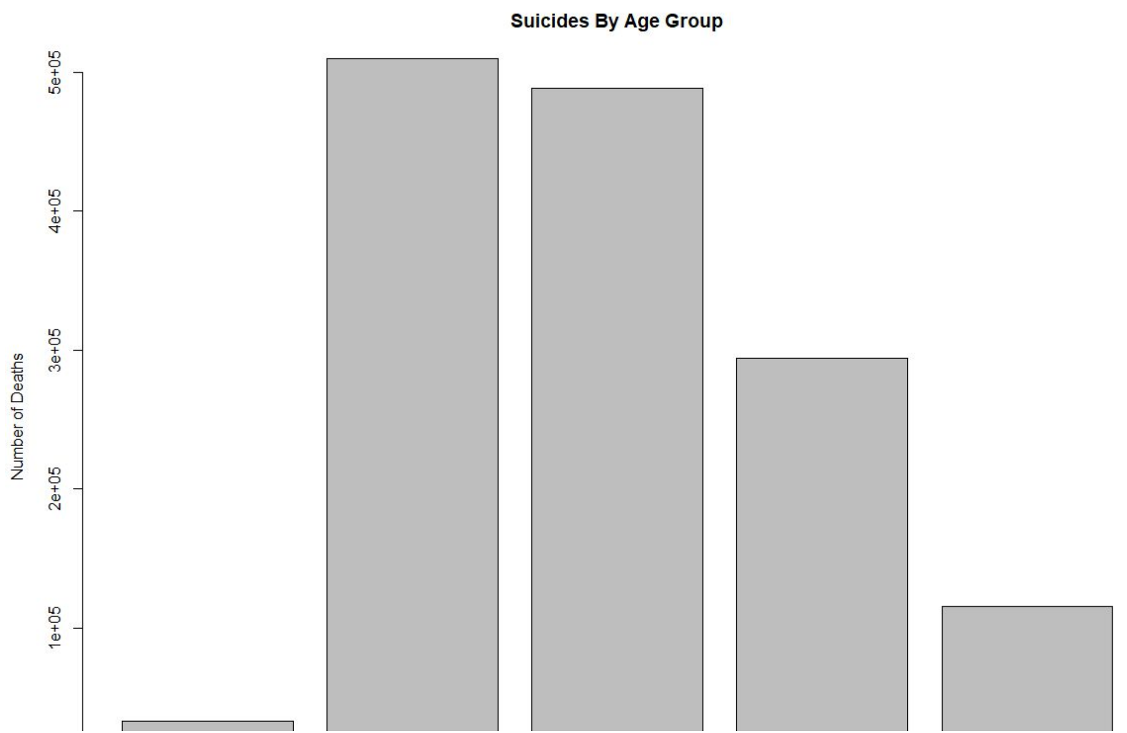
Which age group is more prone to commit suicides?

To answer this question, we need to consider both the genders, throughout the years. And then draw conclusions from this data.

```
# Number of people committing suicide regardless of thier age
> ageDied <- encodedData %>%
+  select(Year, Type_code, Age_group, Total) %>%
+  filter(Total > 0, Type_code == "Causes")
```

```
# Number of people committing suicide per age group per year
> aggAgeDied <- as.data.frame(aggregate(ageDied$Total,
by=list(Category=ageDied$Age_group), FUN=sum))
```

```
# Plot the graph of the trends of suicide by age groups
> barplot(aggAgeDied$x, main="Suicides By Age Group",
+                 names.arg = aggAgeDied$Category,
+                 ylab = "Number of Deaths",
+                 xlab = "Age Group")
```



Hence, we conclude that the age group **15-29** and **30-44** are prone to commit suicides the most.

What is the major cause of suicides in India?

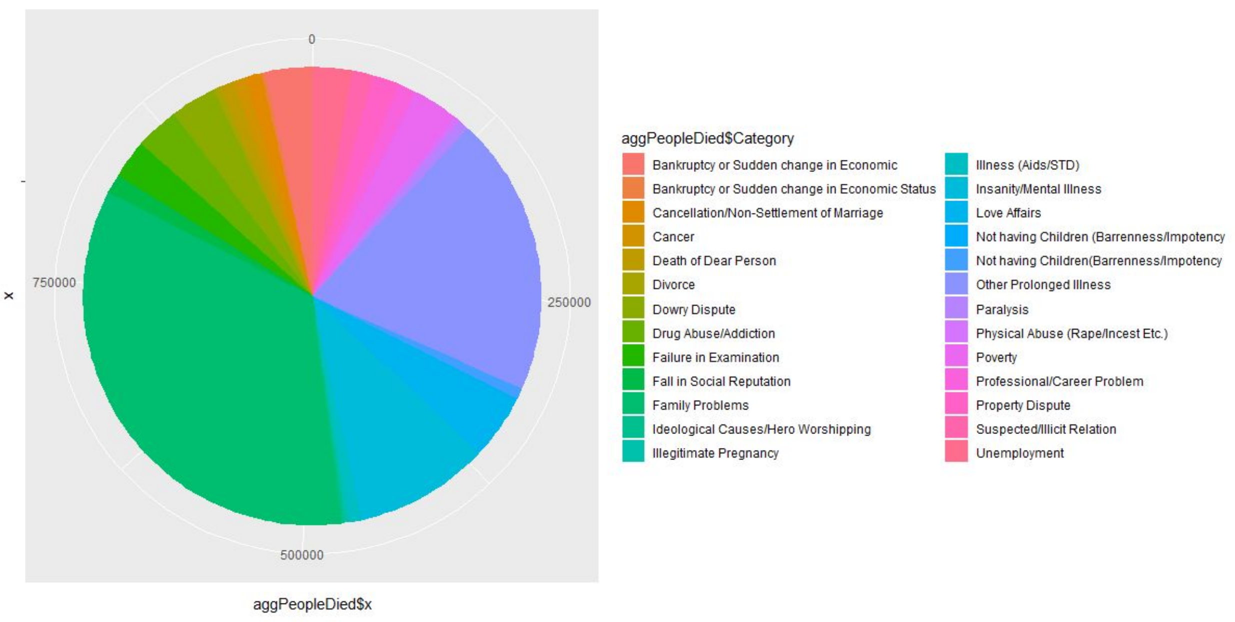
To answer this last question, we need to take a look at all the profiles we collected our data from, group them according to the causes, and plot a pie chart.

```
# Number of suicides regardless of thier cause
> peopleDied <- encodedData %>%
+  select(Year, Type_code, Type, Total) %>%
+  filter(Total > 0, Type_code == "Causes", Type != "Causes Not known", Type != "Other
Causes (Please Specity)")

# Number of students comitting suicide  regardless of thier cause per year
> aggPeopleDied <- as.data.frame(aggregate(peopleDied$Total, +
by=list(Category=peopleDied$Type), FUN=sum))
```

Let’s plot a barplot and a pie chart for our convenience to see which cause has affected the suicides most.

```
# Plot a bar plot
> bp<- ggplot(aggPeopleDied, aes(x="", y=aggPeopleDied$x,
fill=aggPeopleDied$Category))+
+  geom_bar(width = 1, stat = "identity")

# Plot a pie chart
> pie <- bp + coord_polar("y", start=0)
```

Next page, we show our graph.



Hence, we conclude that the "**Family Problems**", "**Other Prolonged Illness**" "**Insanity/Mental Illness**" and "**Love Affairs**" contribute towards suicide the most.