Data Preparation & Exploration Using R

Introduction:

This document describes a high level process for data preparation and exploration in R. Document has been prepared in relevance to contest "Business Analytics for Beginners Using R-Part I"
https://www.crowdanalytix.com/contests/business-analytics-for-beginners-using-r---part-i

R codes are also given as an example working with some variable for the solvers to get them started. Below mentioned sections describe the process in brief:

Contents

CrowdANALYTIX

Section 1: Understanding the business Objective and variables

Solvers should start with downloading all relevant data and data dictionary and reading through the contest description thoroughly. It is important to keep end objective in mind while preparing your data for analysis. Solvers should go through each of the variables to understand its meaning and develop a general hypothesis as how the independent variable could impact the dependent variable in question. Types of technique for modeling as well as data treatment can vary depending on, if variable is continuous, nominal or ordinal.

Please go through the link below to understand about different types of variables
https://statistics.laerd.com/statistical-guides/types-of-variable.php

Section 2: Importing data in R and getting sense of inconsistencies

Analysis will start with importing data in R and getting sense of inconsistencies and missing values in data.

```
# importing data in R
startup<- read.csv(file="CAX_Startup_Data.csv", header=TRUE, as.is=T)
```

Here 'as.is=T' is added to read dates as character which can later be converted to dates. Also please note the variable name with spaces has been converted to variable name with dots in 'R' after importing. R will also convert any special characters in variable name to dot (.) after importing.

One of the prominent inconsistency is missing value are represented as "No Info" as well as "blanks". So, the first step should be to replace "No Info" and "blanks" with "NA" to get better sense of missing values in data.

```
# R code for replacing 'No Info' and 'blanks' with NA
startup[startup=="No Info"]<- NA
startup[startup==""]<- NA
```

While importing file many of the continuous values and dates were imported as character vector that needs to be converted respective data type

```
# R code for converting column as date
startup$Est..Founding.Date <- as.Date(startup$Est..Founding.Date, "%m/%d/%Y")
startup$Last.Funding.Date <- as.Date(startup$Last.Funding.Date, "%m/%d/%Y")

# display column header of data
colnames(startup)
```

```
# R code for converting character vector to numeric
# noting columns that needs to be converted to numeric
col<- c(3:5,10,11,18:23,25,61,66,68:70,72,74,88,92,94:96,98,99,102:116)

# using for loop for converting column as numeric
for(i in col)
 {
  startup[,i]<-as.numeric(startup[,i])
 }
```

Above conversion will give one warning message. This is because in variable “Last round of funding received (in milionUSD)” (column 96) there is string “unknown amount” which got converted to ‘NA’ that we could have done anyway by specifically replacing it with ‘NA’. So warning message is not cause for concern here.

We can use str(startup) command to see if the changes have been implemented. This will show structure of data frame with data type of variables.

Next step would be to understand amount of missing values in data. Any variable missing more than 50% should not be used in modeling as it can give false impression of relationship with dependent and can pollute the model. Being on conservative side generally we prefer to keep variable with less than 40% missing value only for treatment and anything above 40% missing values are to be used in testing only to give additional insights.

```
# Percent missing value for each variable
mis_val<-sapply(startup, function(x) sum(is.na(x)))
percent_mis<-round((mis_val/nrow(startup))*100,1)

# making data frame with variable and missing value percent for filtering
name<-row.names(percent_mis)
pcnt_mis_var<-cbind(name,percent_mis)
row.names(pcnt_mis_var)<-NULL
colnames(pcnt_mis_var)<-c("variable","Percent.Missing")
```

Now we can filter out the data with more than 40% missing which we can later use for testing and additional insights and keep only less than 40% variable for modeling.

```
# keeping only variables with less than 40% missing
new_var<-pcnt_mis_var$variable[which(pcnt_mis_var$Percent.Missing<=40)]
new_startup<-startup[new_var]

# separate data frame for more than 40% missing
other_var<-pcnt_mis_var$variable[which(pcnt_mis_var$Percent.Missing>40)]
other_data<-startup[other_var]
```

It would be better if we separate out the numeric and character variables from data frame. It would help in performing operations going forward.

```
# Separate data frame for numeric variables
cnt_df<-new_startup[,c(1:3,15,18,20:22,24,26,30,31,33,37,42:44,48,50,51,54:56,60,
                       63:65,70,71,73,74,78,80,82,86,87,90,94:96,98,110,113)
# Separate data frame for character variables
cnt_var<-colnames(cnt_df)
var <- colnames(startup) %in% cnt_var
char_df <- startup[!var]
```

Now each of the variables can be separately explored individually to check for further inconsistency, type of treatment to apply, to create additional feature or to explore graphically.

Section 3: Explore individual variables

We will examine the values and in consistencies in each of the variable to better understand it and rectify for the same. There are two ways to examine any variable depending on type of variable. For continuous variables you can use 'summary' and 'quantile' function and for categorical variables use 'table' function.

```
# checking distribution of continuous variable for outlier detection and missing values
summary(cnt_df$Team.size.all.employees)
quantile(cnt_df$Team.size.all.employees, probs = seq(0, 1, by= 0.05),na.rm=T)
```

output:

```
  0%      5%     10%     15%     20%     25%     30%     35%     40%
45%     50%     55%
   1.0     3.0     4.0     6.0     8.0    10.0    10.0    10.0    11.0
14.0    16.5    23.0
   60%     65%     70%     75%     80%     85%     90%     95%    100%
  30.0    40.0    50.0    50.0    50.0    50.0    80.0   200.0  5000.0
```

Here a big jump in values seem to be occurring after 90th percentile. It can be further examined to identify cut off point of relatively larger jump in values that can be capped before analysis. Capping means very large outlier type values will be replaced with relatively meaningful values.

```
# further exploration to determine cutoff for capping
quantile(cnt_df$Team.size.all.employees, probs = seq(.9, 1, by= 0.01),na.rm=T)
```

output:

```
90%      91%      92%      93%      94%      95%      96%      97%
98%      99%     100%
  80.00    89.46   103.80   127.90   168.86   200.00   200.00   284.2
4   321.58   612.16 5000.00
```

Here relatively very large jump in values is observed after 92nd percentile. So the values above 103.8 can be capped at 103.8

```
# capping values
cnt_df$Team.size.all.employees[cnt_df$Team.size.all.employees>103.8]<-103.8
```

Similarly, categorical variable can be explored individually to check for inconsistency, missing value and appropriate treatment.

```
# checking distribution of categorical variable
table(char_df$Local.or.global.player,useNA="always")
```

output:

```
global  Global  GLObaL  GLOBAL   local   Local   LOCAL local
<NA>
     90      93       2      52      95      99      16
1      24
```

Here there are only two level of values but they are written in different ways which is causing inconsistency. Solver can convert all levels of variable to uppercase for any variable in data frame.

```
# convert a variable to uppercase
char_df$Local.or.global.player<-toupper(char_df$Local.or.global.player)
# check again for any inconsistency
table(char_df$Local.or.global.player,useNA="always")
```

output:

```
GLOBAL    LOCAL  LOCAL      <NA>
   237      210        1       24
```

Still one level seems to be different which is due to whitespaces. So we will use 'trimws' function to remove whitespaces.

```
# trimming whitespaces
char_df$Local.or.global.player<-trimws(char_df$Local.or.global.player)
```

output:

```
GLOBAL   LOCAL    <NA>
   237     211      24
```

Although R can handle character variable in analysis, if solvers want they can recode the variable also.

```
# Recoding variable levels and converting to factor variable
char_df$Local.or.global.player[char_df$Local.or.global.player=='LOCAL']<-0
char_df$Local.or.global.player[char_df$Local.or.global.player=='GLOBAL']<-1

char_df$Local.or.global.player<- as.factor(char_df$Local.or.global.player)
```

Section 4: Feature Creation

Solver can create additional features on top of given data to make it more meaningful which will help in analysis/ modeling also. For example, variable "Investors" has list of investors for the company separated by 'pipline' symbol. They can create 'Count.of.investors' variable which will help in analysis.

```
# Create additional features like counting number of investors for company
char_df$Investor.count<-length(strsplit(char_df$Investors, "|",fixed=T))
for (i in (1:length(char_df$Investors)))
  {
   if(is.na(char_df$Investors[i])==T){
    char_df$Investor.count[i]<- NA}
   else{
    lst<-strsplit(char_df$Investors[i], "|", fixed=T)
    char_df$Investor.count[i]<-length(lst[[1]])
   } }
```

Section 5: Missing Value Treatment

Missing value treatment is one of the critical aspect which impact your analysis to a great extent. Idea of treatment is based on nearest neighbour approach where treatment should be done at most granular to higher level. For example, if employee size is missing for a company of USA you can fill the data by taking averages of employee size of all USA based company or you can fill by simply taking average of variable. Treatment can vary based on if variable is continuous or categorical, skewed or normally distributed, time series or cross sectional and also on what type of modeling techniques you are intending to use.

- **Time Series:** Here treatment should be done using moving averages of continuous time period. For example, if data is missing for 2013 you can take average of 2012 and 2014 to fill for 2013 or simply take moving average of 2010 to 2012 to fill for 2013. If data has seasonality you should try to capture it using averages of compounded growth rate.
- **Cross Sectional:** Usually cross sectional data is treating with mean for continuous variable and mode for categorical variable. If distribution of continuous variable is skewed, it is better to treat it with median.
- **Treatment For Tree Based Algorithms:** If you are using a tree based algorithm like 'CART', random forest or 'gradient boosting trees' you can simply put a missing value indicator like '-1' in place of missing value. Even if you don't do outlier treatment algorithm will take care of it and such value just get classified under different branches of trees based on the relationship among variables.

Here data is of cross sectional nature, so, treatment with mean or median for continuous variable and mode for categorical variable at overall level will do.
Since distribution of employee count seems skewed we will use median for treatment.

```
# Missing value treatment using median
cnt_df$Team.size.all.employees[is.na(cnt_df$Team.size.all.employees)]<-
median(cnt_df$Team.size.all.employees,na.rm=T)
```

For categorical variable we will use mode. We will calculate model using user defined function.

```
# Function to calculate mode
Mode <- function(x) {
 u <- unique(x)
 u[which.max(tabulate(match(x, u)))]
 }

# filling missing values with mode
char_df$Local.or.global.player[is.na(char_df$Local.or.global.player)]<-
Mode(char_df$Local.or.global.player)
```

Section 6: Graphical Exploration

Solvers can also explore each variable graphically to check for distribution or create any additional features. For continuous variable they can use boxplot or histogram and for categorical variable they can use bar charts to draw some inferences. It can also be used to understand what type of missing value treatment to apply for any variable, explore relationships with dependent or to present your findings.

```
# boxplot of employee count
boxplot(cnt_df$Employee.Count, main="box plot of employee count",
      ylab="Employee count")

# histogram with black outline, white fill and median line
library(ggplot2)
ggplot(cnt_df, aes(x=Employee.Count))+
 geom_histogram(binwidth=5, colour="black", fill="white")+
 geom_vline(aes(xintercept=median(Employee.Count, na.rm=T)),
         color="red", linetype="dashed", size=1)+
 ggtitle("Histogram of Employee count")+
 xlab("Employee Count") +
 ylab("Frequency")+
 theme_light()
```

Boxplot and histogram along with percentile distribution can be used to determine outlier and capping cut-offs. Here dependent variable is categorical so, boxplot and bar chart can also be used to see relationship of any independent continuous variable with dependent.

```
# box plot to see differnce in mean of team size w.r.t two categories of dependent
ggplot(cnt_df, aes(x=Dependent.Company.Status,y=Team.size.all.employees,
            fill=Dependent.Company.Status)) +
 geom_boxplot()

# data preparation for bar chart
avg_emp<-aggregate(as.numeric(cnt_df$Team.size.all.employees),
            by=list(as.factor(cnt_df$Dependent.Company.Status)),
                FUN=mean, na.rm=TRUE)
colnames(avg_emp)<-c("company.status","Avg.Employee.size")

# bar chart to check for difference in mean
ggplot(avg_emp, aes(x = company.status, y = Avg.Employee.size)) +
 geom_bar(stat = "identity")
```

Here employee size seems to be higher for successful companies. If the difference in employee's size of failed and successful companies is significant or not can tested using 't-test' before selecting for modeling.

Section 7: Hypothesis Testing and Variable Selection

Hypothesis testing can be used not only from point of view of insights after graphical exploration but also for variable selection. All the variable coming significant in test can be used to proceed for modeling. In general, there are lot of techniques for hypothesis testing based on data type of variable (continuous or categorical). Here we will mainly use 't-test' for testing difference in mean of an independent variable w.r.t to two categories of dependent and "chi-sq test" for testing interdependency for a categorical variable on categorical dependent.

```
# t-test for checking difference in mean
t.test(Team.size.all.employees~Dependent.Company.Status, data=cnt_df)
```

output:

```
Welch Two Sample t-test

data:  Team.size.all.employees by Dependent.Company.Status
t = -3.6299, df = 394.48, p-value = 0.0003209
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -14.294457  -4.250324
sample estimates:
 mean in group Failed mean in group Success
             23.38204              32.65443
```

Here we can reject the null hypothesis of equality with a strong p-value of 0.0003 towards null hypothesis, so, we can choose “Team.size.all.employees” for modeling.

```
# tabulating data for chi-sq test
tab<- table(char_df$Dependent.Company.Status,char_df$Local.or.global.player)

# chi-sq test for categorical variable
chisq.test(tab)
```

output:

```
        Pearson's Chi-squared test with Yates' continuity correc
tion

data:  tab
X-squared = 59.517, df = 1, p-value = 1.212e-14
```

Here, p-value is very small, so we can reject the null hypothesis of independence and conclude that whether company is operating locally or globally has impact on its success. Again this variable can be selected for modeling.