Engineering Graduate Income Prediction

Sagar Apshankar

Introduction

Abstract

India produces ~1.5 M engineering graduates per year. This humongous number is due to the fact that Indians see professions such as medicine, engineering and law as secure futures for their children and bank on their degrees as a way into the middle class. However, there is significant variation in the incomes of graduates of a discipline. Students from populous modern cities are perceived to have an edge over those from the hinterland. The variability in competence between graduates combined with disparity in opportunities available has given rise to a new industry of job placement. Agencies test fresh graduates on their skills and help them get into jobs. The dataset we are going to analyze comes from one such training and placement agency called AMCAT. The following code chunks load the packages needed to run this report. The csv file of the data is available at this link and is also supplied without any changes with this report.

```r
packages_required <-c("dplyr", # Wrangling
  "ggplot2", #for graphics
  "plotly", #graphics 3D chart
  "stringr", #string operations
  "tidyr", # Wrangling
  "caret", #for CreateDataPartition and train()
  "ggthemes", #for graphics themes
  "reshape", #for wrangling, partucularly the melt function
  "lubridate", #for dealing with dates
  "tidyverse", # Wrangling
  "FactoMineR", #PCA analysis
  "factoextra", #PCA analysis
  "missMDA", #imputation of missing data
  "rpart", #ML DT algorithm
  "rpart.plot", #ML DT graphics algorithm
  "randomForest", #ML RF algorithm
  "xgboost",#ML RF algorithm
  "kableExtra") #ML algorithm

using<-function(...) {
  libs<-unlist(list(...))
  req<-unlist(lapply(libs,require,character.only=TRUE))
  need<-libs[req==FALSE]
  if(length(need)>0){
    install.packages(need)
    lapply(need,require,character.only=TRUE)
  }
}
```

```
using(packages_required)
```

Executive Summary

The dataset contains 34 columns. The descriptions for these columns, as found on https://www.kaggle.com/manishkc06/engineering-graduate-salary-prediction are as follows:

- `ID`: A unique ID to identify a candidate
- **`Salary:`** First annual salary offered to the candidate (in INR)
- `Gender`: Candidate's gender
- `DOB`: Date of birth of the candidate
- `10percentage`: Overall marks obtained in grade 10 examinations
- `10board`: The school board whose curriculum the candidate followed in grade 10
- `12graduation`: Year of graduation - senior year high school
- `12percentage`: Overall marks obtained in grade 12 examinations
- `12board`: The school board whose curriculum the candidate followed
- `CollegeID`: Unique ID identifying the university/college which the candidate attended for her/his undergraduate with 1173 levels.
- `CollegeTier`: Each college has been annotated as 1 or 2. The annotations have been computed from the average AMCAT scores obtained by the students in the college/university. Colleges with an average score above a threshold are tagged as 1 and others as 2.
- `Degree`: Degree obtained/pursued by the candidate
- `Specialization`: Specialization pursued by the candidate
- `CollegeGPA`: Aggregate GPA at graduation
- `CollegeCityID`: A unique ID to identify the city in which the college is located in.
- `CollegeCityTier`: The tier of the city in which the college is located in. This is annotated based on the population of the cities.
- `CollegeState`: Name of the state in which the college is located, a factor of 26 levels
- `GraduationYear`: Year of graduation (Bachelor's degree)
- `English`: Scores in AMCAT English section
- `Logical`: Score in AMCAT Logical ability section
- `Quant`: Score in AMCAT's Quantitative ability section
- `Domain`: Scores in AMCAT's domain module
- `ComputerProgramming`: Score in AMCAT's Computer programming section
- `ElectronicsAndSemicon`: Score in AMCAT's Electronics & Semiconductor Engineering section
- `ComputerScience`: Score in AMCAT's Computer Science section
- `MechanicalEngg`: Score in AMCAT's Mechanical Engineering section
- `ElectricalEngg`: Score in AMCAT's Electrical Engineering section
- `TelecomEngg`: Score in AMCAT's Telecommunication Engineering section
- `CivilEngg`: Score in AMCAT's Civil Engineering section
- `conscientiousness`: Scores in one of the sections of AMCAT's personality test. Conscientiousness is defined as the quality of wishing to do one's work or duty well and thoroughly.
- `agreeableness`: Scores in one of the sections of AMCAT's personality test. Agreeableness describes a person's ability to put other people's needs above their own.
- `extraversion`: Scores in one of the sections of AMCAT's personality test. Extraversion is the state of primarily obtaining gratification from outside oneself. Extraverts tend to enjoy human interactions and to be enthusiastic, talkative, assertive, and gregarious.
- `nueroticism`: Scores in one of the sections of AMCAT's personality test. Neuroticism, in psychology and development, a broad personality trait dimension representing the degree to which a person experiences the world as distressing, threatening, and unsafe.
- `openesstoexperience`: Scores in one of the sections of AMCAT's personality test.

Objective

This analysis aims primarily to make a model which can predict the first annual salary of the individual by using the all of supporting information in the rows. It also aims to understand with the help of the best model, the comparative importance of each variable in making a prediction of the Salary which is the outcome.

Key Steps

- Analyze nature of data in each column and load data converting to appropriate data type
- Standardize free-text columns into usable factors retaining only the minimum required variability
- Change non-useful variables into more useful ones using calculations where necessary.
- Minimize NA values by combining variables and then impute the rest while minimizing loss to data integrity
- Perform PCA on the dataset to understand broad tendencies
- Weed out erroneous and extreme values based on the 3 * sd rule
- Find optimum parameters to reduce complexity of data by training multiple lm algorithms and partition the data into 80/20 training and test set
- Introduce, motivate and then use algorithms RandomForest, XgBoost, NeuralNetwork to predict Salary and compare them

Data Pre-processing

Data Loading

Let us examine the contents of each column.

The ID columns including college IDs and College City IDs are likely to be interpreted as 'numeric' values by the read csv function. We should however treat them as factors as the number itself only acts as an identifier and no information is encoded in its magnitude. The following script will load the data from the csv taking into account the correct variable types.

```
url_file =  "https://raw.githubusercontent.com/sagar-aps/engineering_graduate_income_prediction/main/Eng

data<-read.csv2(url(url_file),
                  sep = ",",dec = ".",colClasses = c(
                                                  ID="factor",
                                                  DOB= "Date",
                                                  CollegeID="factor",
                                                  CollegeTier="factor",
                                                  CollegeCityID="factor",
                                                  CollegeCityTier="factor")
                ,stringsAsFactors = TRUE)

data%>% glimpse()
```

```
## Rows: 2,998
## Columns: 34
## $ ID                    <fct> 604399, 988334, 301647, 582313, 339001, 609356, ~
## $ Gender                <fct> f, m, m, m, f, f, f, f, m, f, m, m, f, f, m, m, ~
## $ DOB                   <date> 1990-10-22, 1990-05-15, 1989-08-21, 1991-05-04,~
## $ X10percentage         <dbl> 87.80, 57.00, 77.33, 84.30, 82.00, 83.16, 72.50,~
## $ X10board              <fct> "cbse", "cbse", "maharashtra state board,pune", ~
## $ X12graduation         <int> 2009, 2010, 2007, 2009, 2008, 2007, 2007, 2009, ~
## $ X12percentage         <dbl> 84.00, 64.50, 85.17, 86.00, 75.00, 77.00, 53.20,~
## $ X12board              <fct> "cbse", "cbse", "amravati divisional board", "cb~
## $ CollegeID             <fct> 6920, 6624, 9084, 8195, 4889, 10950, 14381, 1320~
## $ CollegeTier           <fct> 1, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ Degree                <fct> B.Tech/B.E., B.Tech/B.E., B.Tech/B.E., B.Tech/B.~
## $ Specialization        <fct> instrumentation and control engineering, compute~
## $ collegeGPA            <dbl> 73.82, 65.00, 61.94, 80.40, 64.30, 99.93, 68.00,~
## $ CollegeCityID         <fct> 6920, 6624, 9084, 8195, 4889, 10950, 14381, 1320~
## $ CollegeCityTier       <fct> 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, ~
## $ CollegeState          <fct> Delhi, Uttar Pradesh, Maharashtra, Delhi, Tamil ~
## $ GraduationYear        <int> 2013, 2014, 2011, 2013, 2012, 2013, 2013, 2013, ~
## $ English               <int> 650, 440, 485, 675, 575, 535, 510, 370, 510, 500~
## $ Logical               <int> 665, 435, 475, 620, 495, 595, 495, 470, 555, 410~
## $ Quant                 <int> 810, 210, 505, 635, 365, 620, 405, 280, 440, 560~
## $ Domain                <dbl> 0.6944793, 0.3423149, 0.8246664, 0.9900088, 0.27~
## $ ComputerProgramming   <int> 485, 365, -1, 655, 315, 455, -1, 465, 525, 385, ~
## $ ElectronicsAndSemicon <int> 366, -1, 400, -1, -1, 300, -1, -1, -1, -1, 260, ~
## $ ComputerScience       <int> -1, -1, -1, -1, -1, -1, -1, -1, 438, 407, -1, 37~
## $ MechanicalEngg        <int> -1, -1, -1, -1, -1, -1, 469, -1, -1, -1, -1, -1,~
## $ ElectricalEngg        <int> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, ~
## $ TelecomEngg           <int> -1, -1, 260, -1, -1, 313, -1, -1, -1, -1, -1, -1~
## $ CivilEngg             <int> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, ~
## $ conscientiousness     <dbl> -0.1590, 1.1336, 0.5100, -0.4463, -1.4992, 0.846~
## $ agreeableness         <dbl> 0.3789, 0.0459, -0.1232, 0.2124, -0.7473, -0.620~
## $ extraversion          <dbl> 1.2396, 1.2396, 1.5428, 0.3174, -1.0697, -0.7585~
## $ nueroticism           <dbl> 0.14590, 0.52620, -0.29020, 0.27270, 0.06223, -0~
## $ openess_to_experience <dbl> 0.2889, -0.2859, -0.2875, 0.4805, 0.1864, -0.285~
## $ Salary                <int> 445000, 110000, 255000, 420000, 200000, 440000, ~
```

For our factors, we will need to switch back temporarily to string datatype for cleaning operations but this is easily done with the 'as' family of functions inside 'mutate'.

A big challenge in analyzing this data comes from the free-text columns `X12board`, `X10board` and `specialization` which have multiple values representing the same idea.

The other big challenge is that some 'factors' have far too many levels to be useful. `CollegeID` with 1173 different factor levels, most having a single row of representation is exemplary of this problem. In this section, we will first deal with the free-text columns. Calculating the degree to which we need to simplify the other columns will be dealt with using cross-validation to compare multiple lm models.

A cursory look through the data tells us that `X10 board`, `X12board` and `Specialization` have many unique values and will require a lot of cleaning. Cleaning these fields necessitates some background knowledge of the education system in India. The rationale for the cleaning method employed is given in the following paragraph.

```
#next statement supresses warning :
#`summarise()` has grouped output by 'X10board'. You can override using the `.groups` argument.
```

```
options(dplyr.summarise.inform = FALSE)

data %>% group_by(X10board,X12board) %>%
  summarise(n=n()) %>%
  arrange(desc(n)) %>%
  mutate(perc = n/sum(n)*100) %>%
  filter(X10board != "cbse" & X10board != "icse")
```

```
## # A tibble: 349 x 4
## # Groups:   X10board [219]
##    X10board                     X12board                            n  perc
##    <fct>                        <fct>                           <int> <dbl>
##  1 state board                  state board                       835 94.8
##  2 0                            0                                 255 99.6
##  3 up board                     up board                           68 94.4
##  4 state board                  cbse                               26  2.95
##  5 ssc                          board of intermediate              20 20.8
##  6 matriculation                state board                        19 65.5
##  7 rbse                         rbse                               16 80
##  8 up                           up                                 13 92.9
##  9 board of secondary education board of intermediate education     9 64.3
## 10 mp board                     mp board                            9 81.8
## # ... with 339 more rows
```

Indian students have two significant exams in their life, the first being 10th standard exam (grade) which is generally taken at 16 years old. The next exam is the 12th standard exam which is taken at 18 years old and is accompanied by entrance examinations into professional courses such as engineering. Each Indian state has their own “State Education board”; a body which sets curricula and conducts the examinations. There are also two other significant national boards of education, namely, the CBSE (Central Board of Secondary Education) and the ICSE(Indian Certificate of Secondary Education) to which schools can be associated. Fewer schools are associated with ICSE and CBSE than the state boards. The ICSE was established to provide special education to mixed race British-Indians and the CBSE, to provide education for wards of central government and military wards who changed schools frequently because of transfers of their parents. These are known by many other acronyms such as “aissce” for CBSE. As an example of the nature of cleaning required, “central” refers to “Central Board for Secondary Education”, a.k.a “CBSE”. Moreover, some students have used the name of their school instead of the board names.

Since we don’t have many examples from each state board and sometimes we don’t have information about the state at all, we shall be grouping all State Boards into one category.

First, by row, we will identify all typing mistakes, alternate names and research school names to find the correct education board and add them to a variable storing alternate names. Next,we can replace the names and group to see the result. We will replace all the 0s with NA.

```
#We first make a list of all non-standard values in the columns X10board and X12board.

cbse_alt_names <- c("central","cbse","delhi public school",
                    "jawahar navodaya vidyalaya ","all india board " ,"cbsc","aissce","aisse")
icse_alt_names <- c("certificate", "icse", "cisce", "isc", "anglo")

data %>% mutate(
  X10board = as.character(X10board),
  X12board = as.character(X12board), #necessary for string functions
  X10board=ifelse(str_detect(X10board,paste(cbse_alt_names,collapse = "|")),"cbse",X10board),
  X10board=ifelse(str_detect(X10board,paste(icse_alt_names,collapse = "|")),"icse",X10board),
  X12board=ifelse(str_detect(X12board,paste(cbse_alt_names,collapse = "|")),"cbse",X12board),
  X12board=ifelse(str_detect(X12board,paste(icse_alt_names,collapse = "|")),"icse",X12board),
  X10board=ifelse(str_detect(X10board,paste(c("cbse","icse","0"),collapse = "|")),X10board,"state board
  X12board=ifelse(str_detect(X12board,paste(c("cbse","icse","0"),collapse = "|")),X12board,"state board
  X10board=ifelse(str_detect(X10board,"0"),NA,X10board),
  X12board=ifelse(str_detect(X12board,"0"),NA,X12board),
  X10board = as.factor(X10board),
  X12board = as.factor(X12board)
  ) %>%
  group_by(X10board,X12board) %>%
  summarise(n=n())
```

```
## # A tibble: 12 x 3
## # Groups:   X10board [4]
##    X10board    X12board        n
##    <fct>       <fct>       <int>
##  1 cbse        cbse          936
##  2 cbse        icse            5
##  3 cbse        state board   100
##  4 icse        cbse           65
##  5 icse        icse          126
##  6 icse        state board    31
##  7 state board cbse           47
##  8 state board icse            5
##  9 state board state board  1418
## 10 state board <NA>            9
## 11 <NA>        cbse            1
## 12 <NA>        <NA>          255
```

The above code worked but was cumbersome. In the next code chunk, we write a function to make it more readable and replace the boards in the data.

```
replace_by <- function(target, pattern, replaceby = deparse(substitute(pattern)), ifnotfound=target) {
  ifelse(str_detect(target, paste(pattern,collapse = "|")),replaceby,ifnotfound)
}



#replaceby defaults to the variable name of the variable containing values to be replaced.
#if the name is cbse_alt_names, all the alternative names will be replaced with "cbse_alt_names" by def

#ifnotfound defaults to the value of the target variable for the particular row
#if the value doesn't match our pattern, we want it remain unchanged. We do this by copying the origina
```

The next piece of code replaces the boards using our function.

```
noncbseicse0 <- paste(c("cbse","icse","0"))

data <- data %>% mutate(

  #Save original variables
  X10board_orig= X10board,
  X12board_orig = X12board,

  #convert to char to facilitate string functions
  X10board = as.character(X10board),
  X12board = as.character(X12board),

  #replace cbse and icse synonyms
  X10board= replace_by(X10board,cbse_alt_names,"cbse"),
  X10board= replace_by(X10board,icse_alt_names,"icse"),
  X12board= replace_by(X12board,cbse_alt_names,"cbse"),
  X12board= replace_by(X12board,icse_alt_names,"icse"),

  #replace non cbse & icse and non 0 by state board
  X10board= replace_by(X10board, noncbseicse0, X10board ,"state board"),
  X12board= replace_by(X12board,noncbseicse0, X12board, "state board"),

  X10board= replace_by(X10board, "0",NA), # Convert 0 values to NA
  X12board= replace_by(X12board, "0",NA),

  X10board = as.factor(X10board), #reconvert to factor
  X12board = as.factor(X12board)
  )


#We reconvert to factor as string values aren't useful for training ML algortihms.
#A finite list of values represented by factors is more suited to this purpose.
#These are automatically assigned numbers in the algorithms.
```

We can similarly see that Specialization has many unique values referring to all the different names of engineering courses across the country. Factors are only useful for prediction if they have many instances in the data. Grouping of similar courses, thus is mandatory if we want to use the variables.

The following code shows that 494 data-points for the column `Specialization` are from courses which have appear fewer than 150 times (5% of the data). A factor which has less than 5% of instances is difficult to use since we have very little data to train from and thus would run the risk of overfitting.

```
data %>%
  group_by(Specialization) %>%
  summarise(n=n()) %>%
  arrange(n) %>%
  filter(n<150) %>%
  pull(n) %>%
  sum()
```

```
## [1] 494
```

In order to simplify the data, we will make 8 groups separating engineering disciplines.

```
mechanical <- c("industrial & management engineering","mechanical & production engineering",
                "industrial engineering","automobile/automotive engineering",
                "mechanical and automation", "industrial & production engineering",
                "mechanical engineering")
```

```r
computer <-c("computer and communication engineering" ,"computer networking",
             "computer science and technology","computer engineering",
              "computer science & engineering")

#eintc stands for electronics, instrumentation and telecommunication

eintc <-c("electronics & instrumentation","control and instrumentation engineering",
          "instrumentation engineering","mechatronics","applied electronics and instrumentation",
          "electronics","embedded systems technology" ,"electronics and computer engineering",
          "telecommunication engineering","electronics engineering",
          "instrumentation and control engineering","electronics & instrumentation eng" ,
          "electronics and communication engineering", "electronics & telecommunications")

electrical <- c("electrical and power engineering","electrical engineering","electrical")

#IT is generally a separate discipline from computer science in Indian engineering schools

IT <- c("information & communication technology", "information science",
        "information science engineering", "computer application","information technology")

biotech <- c("biomedical engineering" ,"biotechnology")

other <- c("ceramic engineering","metallurgical engineering","aeronautical engineering",
           "chemical engineering","other","electronics and electrical engineering")


#check mutation spec

 data %>% mutate(

  Specialization = as.character(Specialization),
  spec_orig = Specialization,

  Specialization = replace_by(Specialization,mechanical),
  Specialization = replace_by(Specialization,computer),
  Specialization = replace_by(Specialization,eintc),
  Specialization = replace_by(Specialization,electrical),
  Specialization = replace_by(Specialization,IT),
  Specialization = replace_by(Specialization,biotech),
  Specialization = replace_by(Specialization,other),
  Specialization = as.factor(Specialization)) %>%
  group_by(Specialization, spec_orig) %>%
  summarise(n=n()) %>%
  arrange(Specialization)
```

```
## # A tibble: 42 x 3
## # Groups:   Specialization [8]
##    Specialization    spec_orig                                   n
##    <fct>             <chr>                                   <int>
##  1 biotech           biomedical engineering                      2
##  2 biotech           biotechnology                              12
##  3 civil engineering civil engineering                          15
##  4 computer          computer and communication engineering      1
##  5 computer          computer engineering                      415
##  6 computer          computer networking                         1
##  7 computer          computer science & engineering            557
##  8 computer          computer science and technology             4
##  9 computer          electronics and computer engineering        3
## 10 eintc             applied electronics and instrumentation     5
## # ... with 32 more rows
```

After adjusting the groups as necessary, we can mutate the dataset keeping the originals if we need them later.

```
#final mutate Specialization

data<-data %>% mutate(

  Specialization = as.character(Specialization), #convert to char
  spec_orig = Specialization,    #backup of original


  Specialization = replace_by(Specialization,mechanical),
  Specialization = replace_by(Specialization,computer),
  Specialization = replace_by(Specialization,eintc),
  Specialization = replace_by(Specialization,electrical),
  Specialization = replace_by(Specialization,IT),
  Specialization = replace_by(Specialization,other),

  Specialization = as.factor(Specialization) #reconvert to factor
  )
```

We see two year columns and the date of birth in the data. Some students may have been late in graduating because of having failed either in school or in their engineering exams. We will be able to analyze this by adding `time_to_grad` and `time_to_X12` columns to the data.

```
#Number of years to graduate

data <- data %>% mutate(time_to_grad = GraduationYear - year(DOB),
                        time_to_X12 = X12graduation -year(DOB)) %>%
   relocate(starts_with("time"), .after = CollegeState)
```

In the columns `computerprogramming` to `civilengg`, -1 denotes that the student hasn't taken the test. In other words, it denotes an NA. The following code will replace -1 with NA.

```
replace_minus_one <- function(x){
  ifelse(x==-1,NA,x)
}

data <- data%>%
  mutate_at(.vars = vars(ComputerProgramming:CivilEngg),replace_minus_one)
```

Finally, we make a dataframe with all our now clean data.

```
cleaned_data <- data %>% select(Gender:Salary)
```

Missing Data

As is common with real world datasets, we have many NAs in our data. Our choices with respect to this missing data will affect the accuracy of our prediction model and the quality of the insights that we get from it. Let us examine which columns have missing values and how many.

```
sapply(cleaned_data,  function(x) sum(is.na(x))) %>%
  as.data.frame() %>%
   `colnames<-` ("NAs") %>%
  filter(NAs>0) %>%
  mutate(percent = paste(round(NAs/nrow(cleaned_data)*100,1),"%"))%>%
  print.data.frame()
```

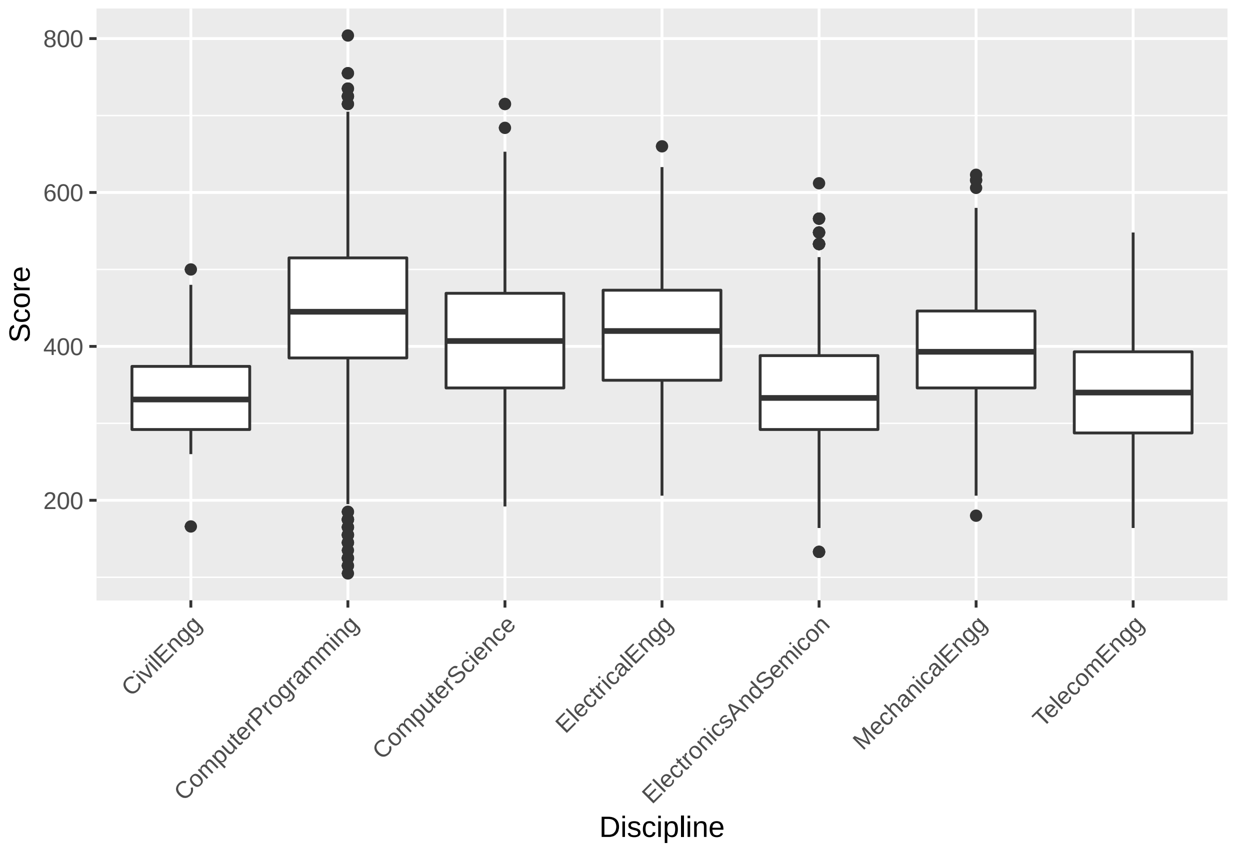
```
##                          NAs percent
## X10board                 256   8.5 %
## X12board                 264   8.8 %
## ComputerProgramming      650  21.7 %
## ElectronicsAndSemicon 2133  71.1 %
## ComputerScience         2298  76.7 %
## MechanicalEngg          2811  93.8 %
## ElectricalEngg          2876  95.9 %
## TelecomEngg             2724  90.9 %
## CivilEngg               2972  99.1 %
```

We see that more than 99% of the values from the variable `CivilEngg` are missing! One option is to impute the missing data directly. This can be done with the MissMDA package by running the following command. However, this is a bad idea because we then make an assumption of all students being able to score average (or another complex aggregate) marks in all of the tests increasing their impact on predictions.

```
#Takes hours to run so don't run unless you have the time

 cleaned_imputed <- imputeFAMD(cleaned_data , ncp = 2)
```

However, we see that the data in `ComputerProgramming:CivilEngg` is basically sparse and we may be able to capture the information by aggregating intelligently. A good approach would simplify this data while losing as little detail as is possible. A simple example would be to take the mean of all the columns (excluding NAs). This seems like a good approach if we consider all tests to be equal but this assumption needs to be verified.

```
cleaned_data %>% select(ComputerProgramming:CivilEngg) %>%
  gather("Discipline","Score")%>%
  na.omit()%>%
  ggplot(aes(Discipline,Score)) +
  geom_boxplot()+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



It is clear that some tests were clearly harder than others (or that students are systematically worse in certain disciplines). A simple average excluding NAs would thus not do.

In order to account for each tests' difficulty, we can scale the variables and then take a mean. However, one may ask, what if some students took multiple exams not linked to their disciplines, 'for fun'?

For these "jacks of all trades", we can include in a separate column, the highest scale attained in any of the tests as well as the number of tests taken.

The following code makes uses the data available in those columns to make a meaningful summary variable and reduce NAs in the process.

We also remove DOB as logically as we have already used it to calculate time to graduate and time to 12th standard. As such, our calculated variables are definitely better indicators for Salary than the DOB itself.

```
cleaned_NA_treated_data <- cleaned_data %>% select(ComputerProgramming:CivilEngg) %>%
  mutate_all(funs(scale)) %>%
  mutate(MaxScaledScore = apply(X=., MARGIN=1,FUN = max, na.rm=TRUE)) %>%
  mutate(MeanScale = rowMeans(.,na.rm = TRUE),
         TestsTaken = rowSums(!is.na(.))-1)%>%
          select(MaxScaledScore:TestsTaken) %>%
  mutate_all(.,.funs = function(x){ifelse(is.infinite(x),NA,x)})%>%
  cbind(cleaned_data %>% select(-c(ComputerProgramming:CivilEngg))) %>%
  relocate(1:3, .after = Domain)%>%
  select(-DOB)%>% #We included information in time_to_graduate
  select(-CollegeID)%>% #We remove factors with too many levels before PCA
  select(-CollegeCityID)%>%
  mutate(Salary_Cat = cut(Salary,c(seq(0,1000000,200000),10000000)))
```

We can see from the following code chunk that the NAs in the data have now drastically reduced.

```
sapply(cleaned_NA_treated_data,  function(x) sum(is.na(x))) %>%
  as.data.frame() %>%
   `colnames<-` ("Nas") %>%
  filter(Nas>0) %>%
  mutate(percent = paste(round(Nas/nrow(cleaned_data)*100,1),"%"))%>%
  print.data.frame()
```

```
##                 Nas percent
## X10board        256   8.5 %
## X12board        264   8.8 %
## MaxScaledScore   94   3.1 %
## MeanScale        94   3.1 %
```

Principal Components

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. The data is reduced to a set of principal components (PC1, PC2, PC3 and so on) out of which the lower PCs account for the maximum variability. Hence, by studying just the first two dimensions, we can build an idea of where the vriability in our data comes from.

With so few rows having missing data (3% of scores and 8% of boards are missing) , we can impute the missing values and run FAMD (MCA + PCA). Although principal component methods are more suited to prediction of categorical variables from numeric variables, they can still be useful in for identifying patterns in data such as ours. We can hack around this by making groupings of the Salary using the ‘`cut`‘ command.

Moreover, the MissMDA package allows for imputing unknown data while minimizing the damage to data integrity.

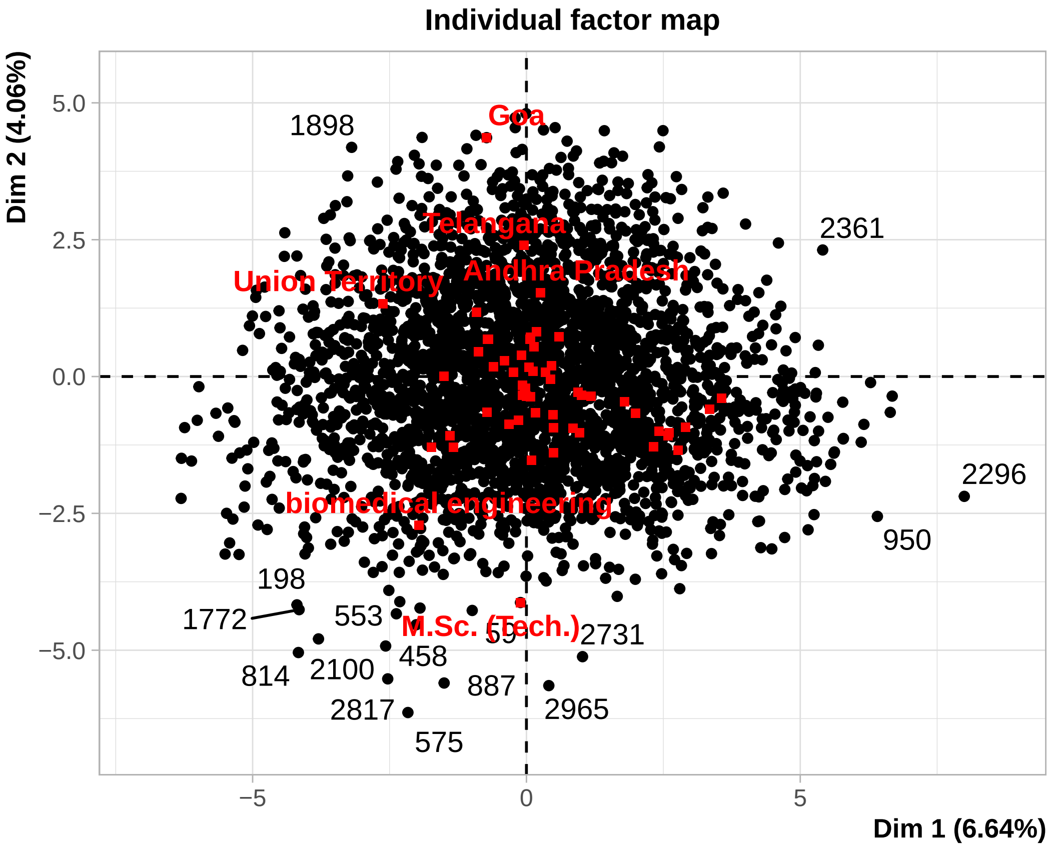
FAMD will help us identify which variables are the cause of maximum variability in our data. This in turn can allow us to eliminate variables from future analyses.

We shouldn’t include factor variables which have a very high number of levels like `CollegeID`(>1000 factors) in the PCA as those columns then will become the source of the maximum variability overshadowing all other subtle variations that we are trying to uncover.

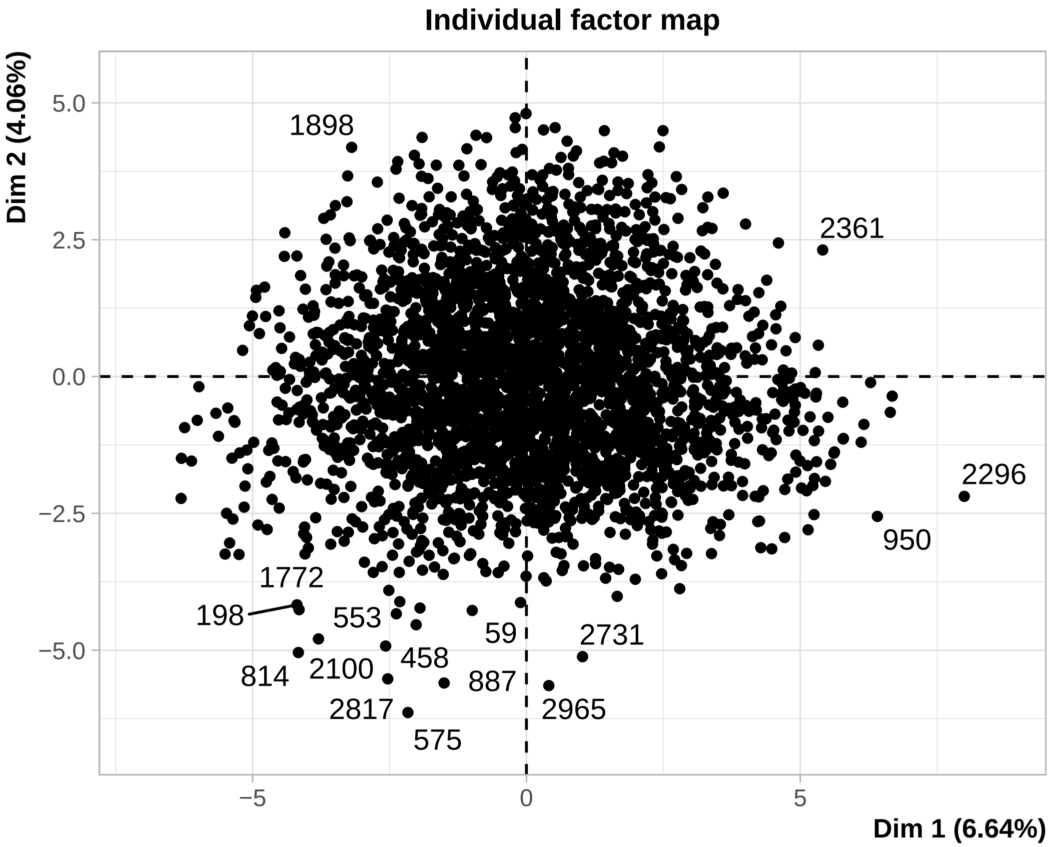
```
knitr::opts_chunk$set(cache = TRUE) #Don't re-run code unless it has changed

cl_NA_tr_imputed <- imputeFAMD(cleaned_NA_treated_data , ncp = 5)

imputed_data<-cl_NA_tr_imputed$completeObs

dat.famd <- FAMD(cleaned_NA_treated_data, tab.disj = cl_NA_tr_imputed$tab.disj)
```

```
## Warning: ggrepel: 2982 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

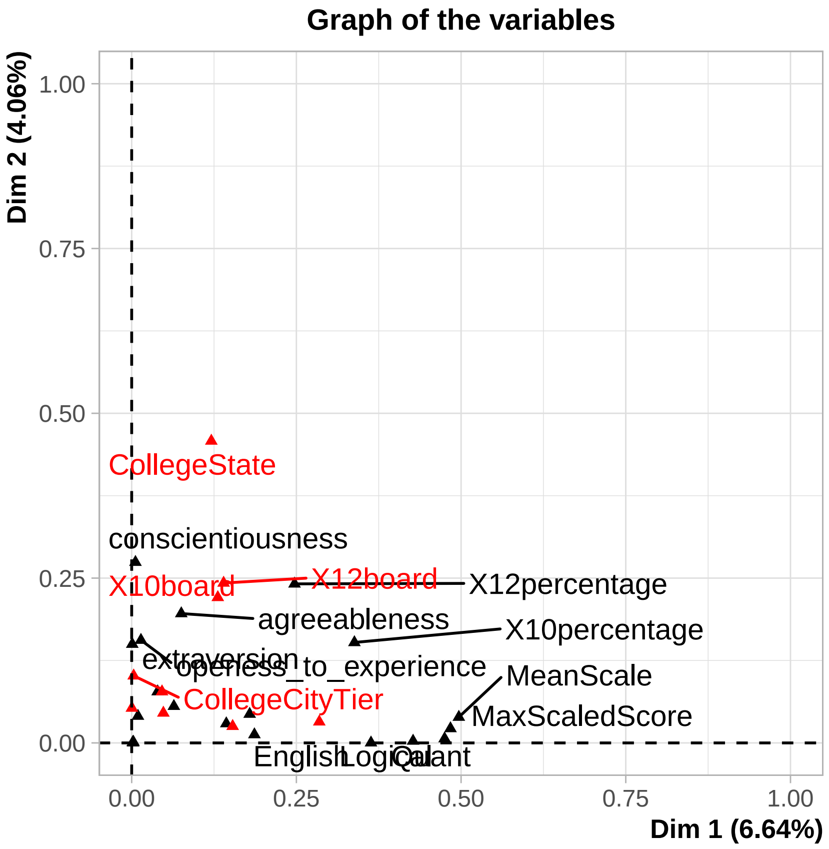
```
## Warning: ggrepel: 51 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



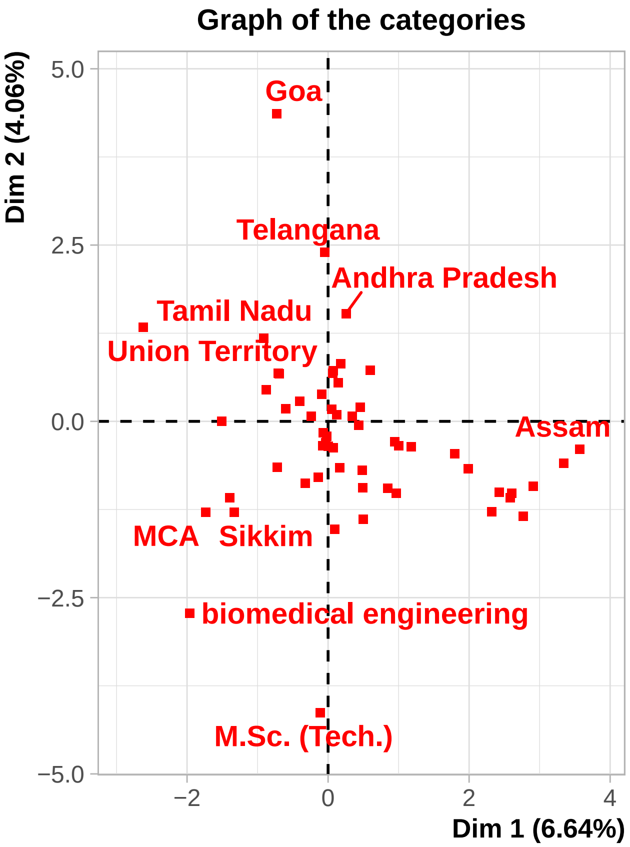
```
## Warning: ggrepel: 2982 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



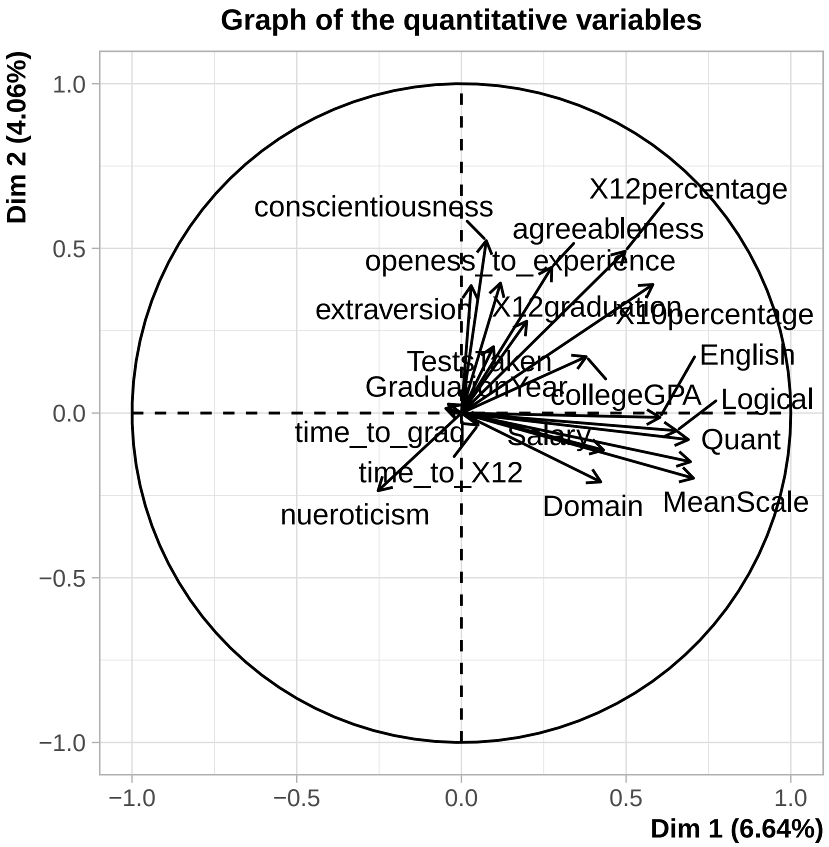
```
## Warning: ggrepel: 14 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```
## Warning: ggrepel: 47 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```
## Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

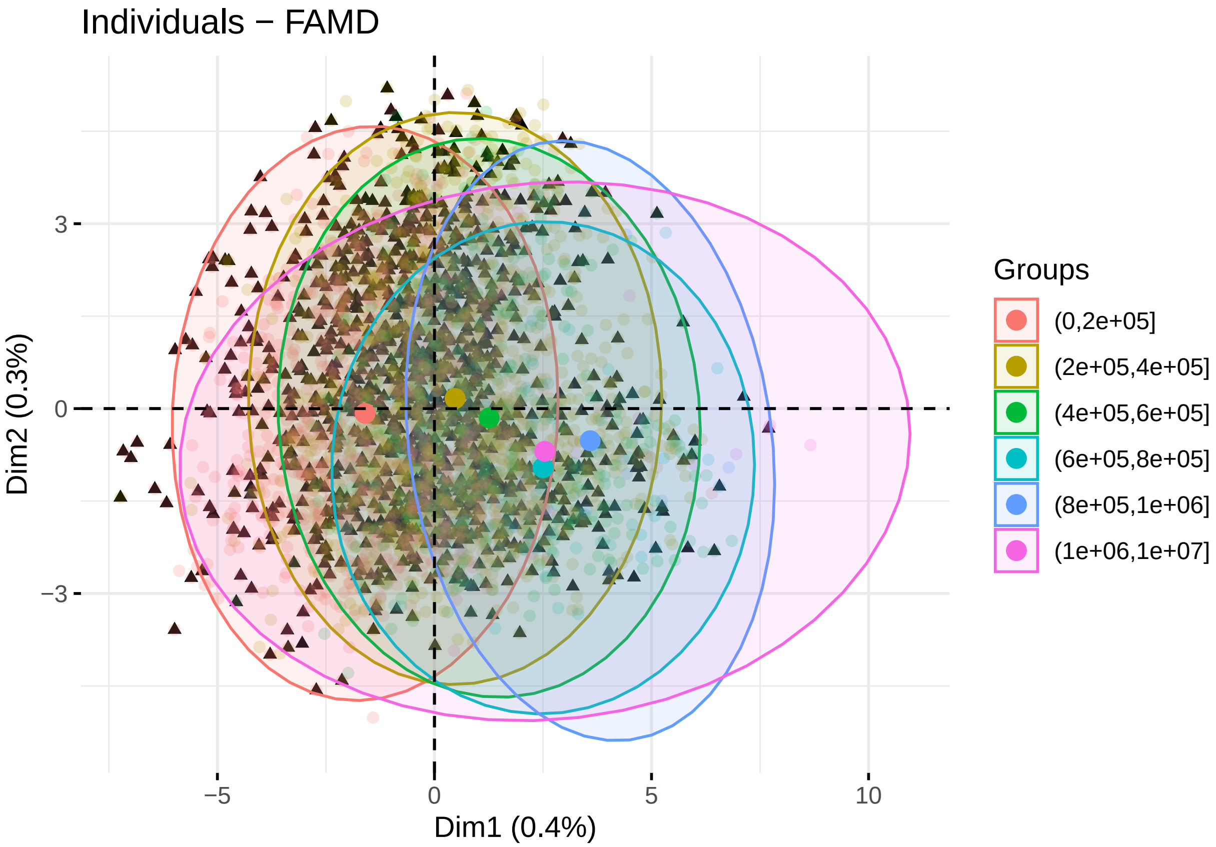


The first thing to note from the PCA correlation circle is that Dim 1 accounts for ~0.29% of the variability and Dim 2, 0.41%. This is low for a PCA analysis and tells us that there may be other latent factors (or missing variables) which are needed to fully explain the data.

The correlation circle also tells us that `MaxScaledScore` and `MeanScale` are very heavily correlated. Having multiple heavily correlated variables introduces noise in linear regression models so we will drop one of them. `neurotiscism`, as can be expected has a negative correlation with all variables linked to testing logical ability. conscientiousness, `openness_to_experience` and `extraversion` also seem to be correlated but less so than `X10score`and `X12score`.

We see that Salary is correlated with `Logical`, `Quant`, Scores and `Domain`. To examine this relationship more closely, we can colour the individuals factor map with salary buckets we created with `cut`.

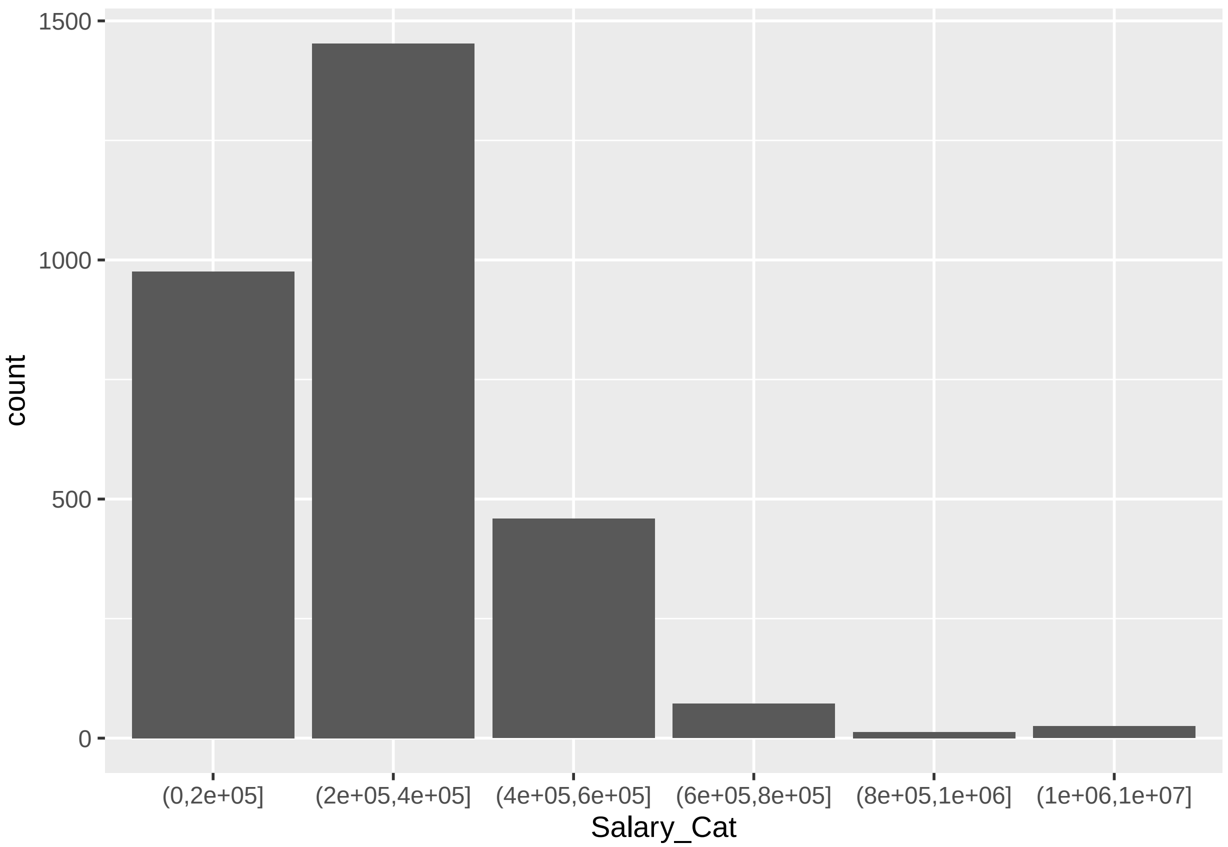
```
fviz_famd_ind(dat.famd,
              label = "none,",
              habillage = cleaned_NA_treated_data$Salary_Cat,
              addEllipses = TRUE,
              alpha.ind = 0.2
              )
```



In the above graph, individuals are plotted on the first two PCA dimensions with their salary grouping. The ellipses have too much overlap but their centers show that PCA1 has a positive relationship with increasing salary. It is interesting to note that mid-high salaries are linked to a decreasing PCA dimension 2, a dimension with a high contribution of `extraversion`, `conscientiousness` and `openness_to_experience`. The last two categories have very few observations linked to them and the large ellipses show that the principal components weren't able to predict for them at all. It may be prudent to remove them from the dataset so as not to bias algorithms.

```
cleaned_NA_treated_data %>% ggplot(aes(Salary_Cat))+
  geom_histogram(stat="count")
```

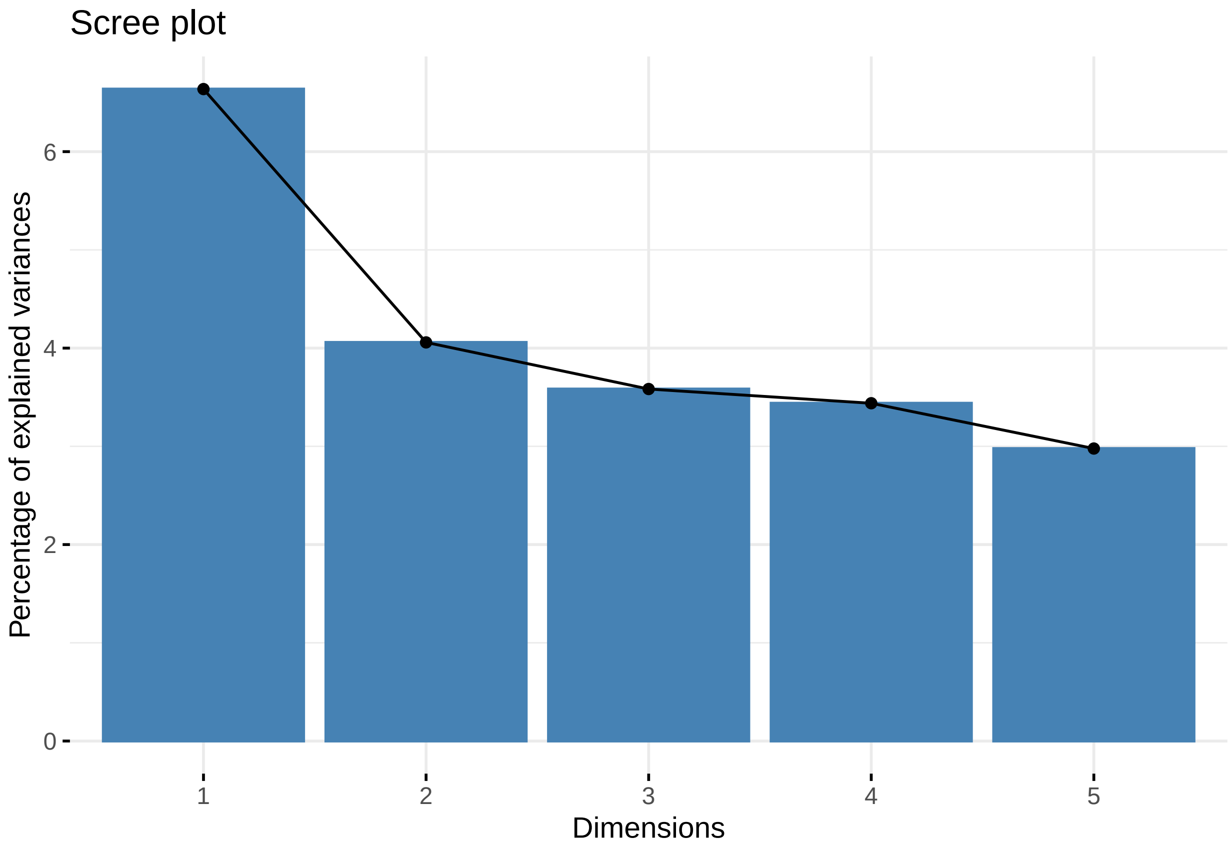
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



```
summary(dat.famd)
```

```
##
## Call:
## FAMD(base = cleaned_NA_treated_data, tab.disj = cl_NA_tr_imputed$tab.disj)
##
##
## Eigenvalues
##                        Dim.1  Dim.2  Dim.3  Dim.4  Dim.5
## Variance               4.499  2.751  2.429  2.331  2.019
## % of var.              6.637  4.058  3.584  3.439  2.978
## Cumulative % of var.   6.637 10.695 14.279 17.718 20.695
##
## Individuals (the 10 first)
##                           Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2
## 1                      |  8.463 |  4.803  0.171  0.322 | -0.032  0.000  0.000 |
## 2                      |  6.389 | -3.019  0.068  0.223 | -0.962  0.011  0.023 |
## 3                      |  5.269 | -0.244  0.000  0.002 |  1.173  0.017  0.050 |
## 4                      |  8.379 |  5.616  0.234  0.449 | -1.400  0.024  0.028 |
## 5                      | 16.882 | -2.123  0.033  0.016 |  0.014  0.000  0.000 |
## 6                      | 12.436 |  3.027  0.068  0.059 | -0.296  0.001  0.001 |
## 7                      |  7.386 | -1.434  0.015  0.038 |  0.063  0.000  0.000 |
## 8                      |  5.745 | -1.531  0.017  0.071 |  2.523  0.077  0.193 |
## 9                      |  5.515 | -0.127  0.000  0.001 |  0.932  0.011  0.029 |
## 10                     |  5.208 | -1.958  0.028  0.141 | -0.355  0.002  0.005 |
##                       Dim.3    ctr   cos2
## 1                     0.534  0.004  0.004 |
## 2                     3.260  0.146  0.260 |
## 3                    -0.165  0.000  0.001 |
## 4                    -0.622  0.005  0.006 |
## 5                     0.604  0.005  0.001 |
## 6                    -0.213  0.001  0.000 |
## 7                    -0.693  0.007  0.009 |
## 8                    -1.268  0.022  0.049 |
## 9                    -1.292  0.023  0.055 |
## 10                   -0.085  0.000  0.000 |
##
## Continuous variables (the 10 first)
##                          Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
## X10percentage        |  0.581  7.515  0.338 |  0.390  5.538  0.152 | -0.161
## X12graduation        |  0.198  0.875  0.039 |  0.279  2.826  0.078 |  0.213
## X12percentage        |  0.497  5.494  0.247 |  0.491  8.770  0.241 | -0.307
## collegeGPA           |  0.379  3.192  0.144 |  0.171  1.065  0.029 | -0.219
## time_to_grad         | -0.048  0.050  0.002 |  0.014  0.007  0.000 | -0.072
## time_to_X12          |  0.051  0.057  0.003 | -0.037  0.048  0.001 |  0.153
## GraduationYear       | -0.040  0.036  0.002 |  0.027  0.027  0.001 | -0.066
## English              |  0.603  8.074  0.363 | -0.013  0.006  0.000 |  0.182
## Logical              |  0.654  9.494  0.427 | -0.054  0.104  0.003 |  0.037
## Quant                |  0.689 10.557  0.475 | -0.081  0.236  0.006 | -0.061
##                         ctr   cos2
## X10percentage         1.072  0.026 |
## X12graduation         1.873  0.045 |
## X12percentage         3.875  0.094 |
## collegeGPA            1.969  0.048 |
## time_to_grad          0.212  0.005 |
## time_to_X12           0.963  0.023 |
## GraduationYear        0.180  0.004 |
## English               1.360  0.033 |
## Logical               0.057  0.001 |
## Quant                 0.151  0.004 |
##
## Categories (the 10 first)
##                          Dim.1     ctr    cos2  v.test     Dim.2     ctr
## f                    |   0.066   0.005   0.001   0.958 |   0.681   1.463
## m                    |  -0.021   0.002   0.001  -0.958 |  -0.214   0.459
## X10board_cbse        |   0.849   1.313   0.208  16.742 |  -0.947   4.365
## X10board_icse        |   1.005   0.392   0.058   7.575 |  -0.343   0.122
## X10board_state board |  -0.709   1.373   0.278 -20.345 |   0.680   3.378
## X12board_cbse        |   0.966   1.711   0.254  19.153 |  -1.023   5.133
## X12board_icse        |   0.943   0.211   0.031   5.472 |  -0.289   0.053
## X12board_state board |  -0.696   1.388   0.291 -21.125 |   0.678   3.526
## CollegeTier_1        |   2.908   3.149   0.405  21.430 |  -0.922   0.848
## CollegeTier_2        |  -0.237   0.257   0.405 -21.430 |   0.075   0.069
##                         cos2  v.test     Dim.3     ctr    cos2  v.test
## f                      0.125  12.586 |   0.034   0.005   0.000   0.674 |
## m                      0.125 -12.586 |  -0.011   0.001   0.000  -0.674 |
## X10board_cbse          0.259 -23.874 |   0.790   3.897   0.180  21.198 |
## X10board_icse          0.007  -3.308 |   1.360   2.466   0.107  13.959 |
## X10board_state board   0.256  24.955 |  -0.720   4.859   0.287 -28.124 |
## X12board_cbse          0.285 -25.943 |   0.859   4.642   0.201  23.184 |
## X12board_icse          0.003  -2.147 |   1.664   2.257   0.098  13.138 |
## X12board_state board   0.276  26.331 |  -0.687   4.644   0.284 -28.397 |
## CollegeTier_1          0.041  -8.694 |   0.097   0.012   0.000   0.972 |
## CollegeTier_2          0.041   8.694 |  -0.008   0.001   0.000  -0.972 |
```

```
fviz_screeplot(dat.famd)
```



We can see more clearly from the output of the summary that the first two dimensions account for 10.6 % of the variability in the data. `ctr` is the contribution to the principal component. `cos2` is how well the PC is capturing the variability.

```
dimres <- dimdesc(dat.famd, axes = 1, proba = 0.05)

  head(dimres$Dim.1$quanti, n=10)
```

```
##                 correlation       p.value
## MeanScale         0.6877762  0.000000e+00
## Quant             0.6833813  0.000000e+00
## MaxScaledScore    0.6781335  0.000000e+00
## Logical           0.6469752  0.000000e+00
## English           0.5934698 8.357225e-285
## X10percentage     0.5673049 5.623714e-255
## X12percentage     0.4847016 1.808120e-176
## Salary            0.4290528 1.491252e-134
## Domain           0.4064396 1.229456e-119
## collegeGPA       0.3782254 1.409093e-102
```

```
  head(dimres$Dim.1$quali, n=10)
```

```
##                         R2       p.value
## Salary_Cat      0.281610903 8.492751e-212
## CollegeTier     0.196763589 9.364079e-145
## CollegeCityID   0.685756005 1.009203e-121
## X12board        0.151376666 3.290962e-106
## X10board        0.140486456  6.183076e-98
## CollegeState    0.155669686  7.748652e-91
## Degree          0.041561582  2.261797e-27
## Specialization  0.040413605  6.488829e-23
## CollegeCityTier 0.004133425  4.276506e-04
```

```
  head(dimres$Dim.1$category, n=10)
```

```
##                          Estimate       p.value
## CollegeTier=1           1.9019239 9.364079e-145
## X12board=cbse           0.9179121  2.343245e-91
## X10board=cbse           0.7882311  6.320601e-70
## CollegeState=Delhi      2.7515553  2.955597e-59
## Degree=B.Tech/B.E.      0.1710295  9.505817e-22
## Salary_Cat=(6e+05,8e+05] 1.0409673  1.156080e-21
## CollegeCityID=47        4.2665210  6.375333e-18
## CollegeCityID=8195      4.9142438  2.074141e-14
## X10board=icse           0.8716595  3.301086e-13
## Specialization=computer 0.5683722  3.503928e-13
```

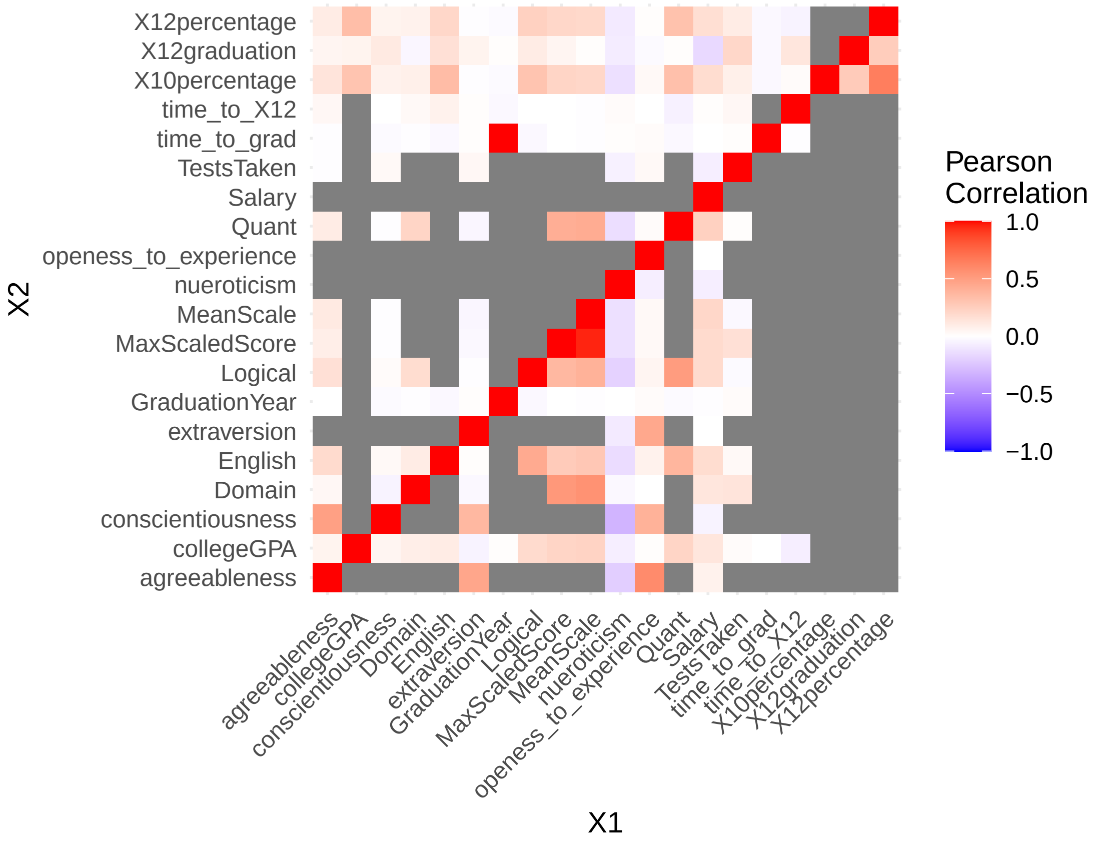
From the output of the dimension description (`dimdesc` command), we can see that the first dimension, built with `MeanScale`, `Quant` and `Logical` points towards academic acumen of the individual. The next biggest contribution is that of `English` suggesting that prowess in English is important for Salary. Among the categorical variables, we see that CollegeTier=1 and X12board and X10 board make the most contribution. The second dimension, with `agreeableness`, `conscientiousness` and `openess_to_experience` as the major weights seems to account more for the personality of the individual. Finally, we can check the `factosummarize` results and compare the effect of each variable on the first two principal components.

```
facto_summarize(dat.famd, element = "var") %>%
  arrange(-contrib)
```

```
##                                           name        Dim.1        Dim.2
## CollegeCityID                    CollegeCityID 0.6857560050 0.8911669485
## CollegeState                      CollegeState 0.1556696864 0.6939887929
## X12percentage                    X12percentage 0.2349356653 0.3179932392
## X10percentage                    X10percentage 0.3218348706 0.1856684774
## MeanScale                            MeanScale 0.4778022792 0.0158598554
## X12board                              X12board 0.1456220058 0.3360076568
## MaxScaledScore                  MaxScaledScore 0.4650190731 0.0060578763
## Quant                                    Quant 0.4670100015 0.0026193024
## X10board                              X10board 0.1358636630 0.3033153446
## Logical                                Logical 0.4185768595 0.0016149660
## English                             English 0.3522064123 0.0011791233
## Salary_Cat                       Salary_Cat 0.2816109035 0.0128162037
## CollegeTier                     CollegeTier 0.1967635887 0.0232365577
## CollegeCityTier             CollegeCityTier 0.0041334249 0.1944918878
## collegeGPA                       collegeGPA 0.1430544692 0.0461546198
## Salary                               Salary 0.1840862763 0.0023671723
## Domain                               Domain 0.1651931869 0.0192302984
## agreeableness                 agreeableness 0.0698457286 0.0629588181
## conscientiousness         conscientiousness 0.0042419261 0.1175545806
## Specialization               Specialization 0.0404136051 0.0625200643
## nueroticism                     nueroticism 0.0584941407 0.0278583724
## X12graduation                 X12graduation 0.0345944494 0.0500899758
## Degree                               Degree 0.0415615821 0.0294727616
## openess_to_experience openess_to_experience 0.0130235312 0.0419176851
## TestsTaken                       TestsTaken 0.0081722433 0.0399292801
## extraversion                   extraversion 0.0006540583 0.0437489506
## Gender                               Gender 0.0002709393 0.0407411449
## time_to_X12                     time_to_X12 0.0025047651 0.0025234898
## time_to_grad                   time_to_grad 0.0015014085 0.0002028986
## GraduationYear               GraduationYear 0.0010051054 0.0006442359
##                              coord         cos2     contrib
## CollegeCityID         1.264440e+00 1.078874e-03 18.12639572
## CollegeState          5.058535e-01 2.023414e-02  9.76664446
## X12percentage         1.563145e-01 1.563145e-01  6.35580077
## X10percentage         1.380505e-01 1.380505e-01  5.83364361
## MeanScale             2.285466e-01 2.285466e-01  5.67454179
## X12board              1.341069e-01 6.705346e-02  5.63913847
## MaxScaledScore        2.162794e-01 2.162794e-01  5.41492987
## Quant                 2.181052e-01 2.181052e-01  5.39828949
## X10board              1.104591e-01 5.522957e-02  5.10907372
## Logical               1.752092e-01 1.752092e-01  4.83001613
## English               1.240507e-01 1.240507e-01  4.06209197
## Salary_Cat            7.946896e-02 1.589379e-02  3.38437730
## CollegeTier           3.925585e-02 3.925585e-02  2.52885514
## CollegeCityTier       3.784418e-02 3.784418e-02  2.28315595
## collegeGPA            2.259483e-02 2.259483e-02  2.17491845
## Salary                3.389336e-02 3.389336e-02  2.14324294
## Domain                2.765859e-02 2.765859e-02  2.11990895
## agreeableness         8.842239e-03 8.842239e-03  1.52656017
## conscientiousness     1.383707e-02 1.383707e-02  1.40002508
## Specialization        5.542018e-03 6.927522e-04  1.18320075
## nueroticism           4.197653e-03 4.197653e-03  0.99260387
## X12graduation         3.705782e-03 3.705782e-03  0.97342955
## Degree                2.596009e-03 8.653363e-04  0.81652475
## openess_to_experience 1.926705e-03 1.926705e-03  0.63153766
## TestsTaken            1.661133e-03 1.661133e-03  0.55291683
## extraversion          1.914398e-03 1.914398e-03  0.51040319
## Gender                1.659914e-03 1.659914e-03  0.47142523
## time_to_X12           1.264185e-05 1.264185e-05  0.05779873
## time_to_grad          2.295395e-06 2.295395e-06  0.01959065
## GraduationYear        1.425277e-06 1.425277e-06  0.01895883
```

We have seen that variables `time_to_X12` , `time_to_grad`, `CollegeCityTier` and `Tests_Taken` have very low contribution in PC1 and PC2. Moreover, the contribution of variables `MeanScale` and `MaxScaledScore` was about the same, indicating that they are heavily correlated. We can confirm this by mapping correlations between all the quantitative variables. We can thus, drop some of them while making subsequent models.

```
# Correlation Matrix

cormat<-imputed_data %>%
  select(where(is.numeric))%>%
  cor()%>%
  round(digits = 2)


# Get lower triangle of the correlation matrix
get_lower_tri<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}

upper_tri <- get_lower_tri(cormat)

upper_tri%>%
  melt(na.rm = TRUE) %>%
  ggplot(aes(X1,X2, fill=value))+
  geom_tile()+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   hjust = 1))+
  coord_fixed()
```



Variable choice and outlier treatment

In order to run algorithms like linear regression, we will first remove the variables we saw to have very low effect or heavily correlated effect in the PCA. We will also add back variables with too many factors that we had removed for PCA.

```
total_imputed_set <- imputed_data %>%
  cbind(CollegeID = cleaned_data$CollegeID)%>%
  cbind(CollegeCityID = cleaned_data$CollegeCityID)%>%
  select(-TestsTaken)%>%
  select(-MaxScaledScore)%>%
  select(-Salary_Cat)
```

In order for any algorithm to give accurate predictions, and especially with linear regression, we need to filter out erroneous values and extremely high values. Such exceptions will skew the algorithm and produce a much less reliable result.

There are many methods to identify such values. here we will us on of the simpler ones: to filter out values greater than 3 times the sd.

```
upper <- mean(data$Salary)+3*sd(data$Salary)
lower <- mean(data$Salary)-3*sd(data$Salary)

total_imputed_set <- total_imputed_set %>%
  filter(Salary<upper | Salary>lower)
```

In the next section, we split data into test and training sets and apply algorithms to find the Salary

Methods

In order to comparatively compare methods we are using here, we will define a function to calculate the residual mean squared error (RMSE).

```
RMSE <- function(true_salary, predicted_salary){
  sqrt(mean((true_salary - predicted_salary)^2))
}
```

Linear regression

Linear regression attempts to model the relationship between variables based on a line. It assigns a value to each level of a factor so as to minimize the error of the predicted variable. Thus, for each variable in the test set, we need to have all levels of factors available in the train set. This however leads to a problem with out dataset because we have a factor with far too many levels : `CollegeID`.

First, we will add back from to the test set from the test set any rows which don't have a factor present in the train set. (if the algorithm doesn't know the value of its coefficient for a particular level of a variable, it can't predict for it).

```
set.seed(755, sample.kind = "Rounding")
```

```
## Warning in set.seed(755, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <- createDataPartition(y = total_imputed_set$Salary, times = 1, p = 0.2,
                                  list = FALSE)

train_set <- total_imputed_set[-test_index,]
test_set <- total_imputed_set[test_index,]


#In order to perform linear regression , we ensure that all factor variables in the test set
#are present in the train set.

factor_names <- total_imputed_set %>%
  select(-where(is.numeric))%>%
  colnames()

removed <- test_set %>%
  anti_join(train_set, by = factor_names)

test_set <- test_set %>%
  semi_join(train_set, by = factor_names)

train_set <- train_set %>%
  rbind(removed)


nrow(test_set)+nrow(train_set)
```

```
## [1] 2998
```

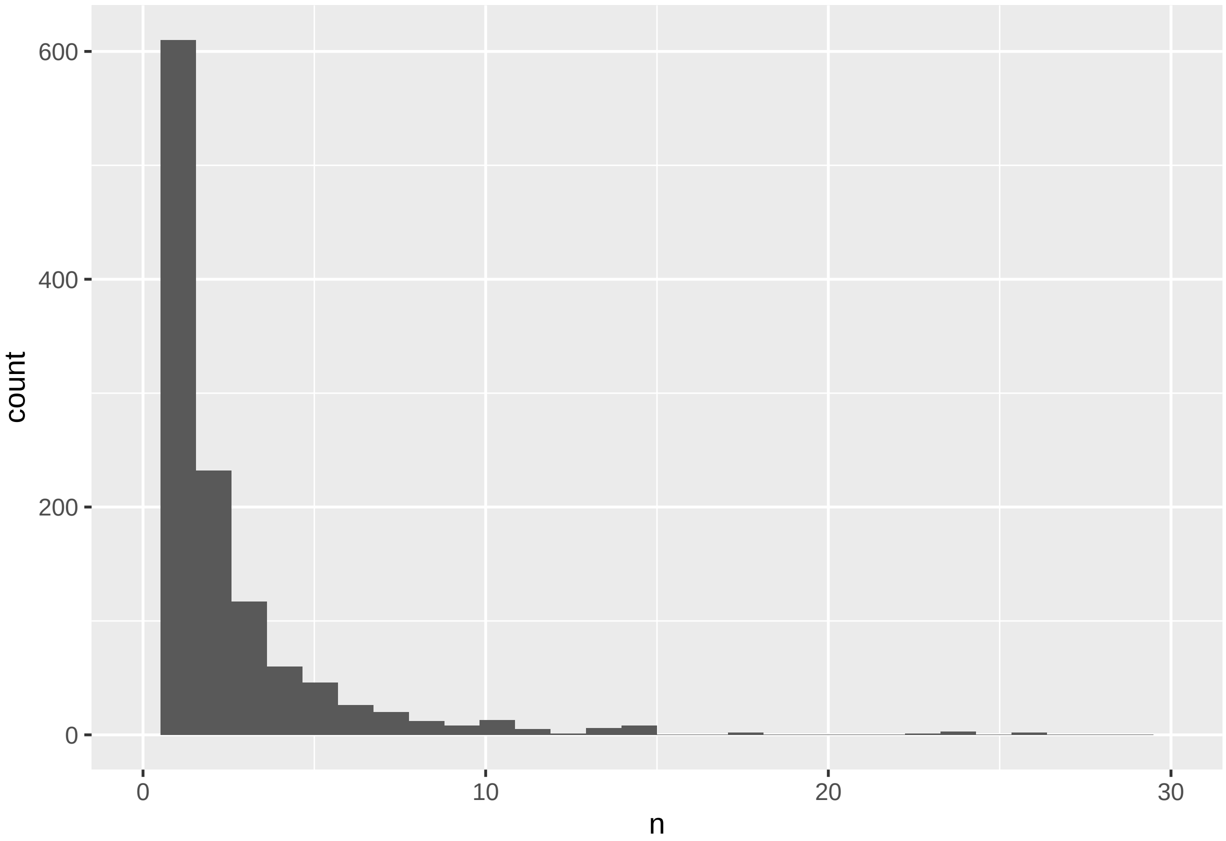
```
#By this, we ensure that all data is being considered.


nrow(test_set)
```

```
## [1] 159
```

We are left with less than 100 of the 901 we started with! This calls for a closer examination of the variable with the most levels, `CollegeID`.

```
total_imputed_set %>% group_by(CollegeID)%>%
  summarise(n=n()) %>%
  arrange(-n)%>%
  ggplot(aes(n))+
  geom_histogram(bins=30)+
  xlim(0,30)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).

## Warning: Removed 2 rows containing missing values (geom_bar).
```



```
total_imputed_set[total_imputed_set$CollegeID %in% names(
  which(table(total_imputed_set$CollegeID)>10)),
  ] %>%
  nrow()
```

```
## [1] 509
```

We see that if we were to select rows with `CollegeID` repeated at least 10 times, we would be left with only 509 of the 2998 observations. One option is to drop the column altogether but in this case, we lose the information about common colleges too. The other is to group together 'obscure' colleges under a 'fake' `CollegeID` (9999). We will also simplify the states at the same time grouping ones which have far too little representation. Too few instances of a college or state can't be considered a representative sample.

This grouping now can now be treated as a tuning parameter. The code to select the ideal `min_coll` and `min_state` has been provided bus isn't set to run as it is very time consuming.

```
res <- data.frame()


for(min_coll in 5:25)
{

  #first loop is for min_coll



  for(min_state in 9:20)

  {

    #second loop is for min_state
    print(paste("min_coll = ",min_coll))
    print(paste("min_state = ",min_state))

    college_ID_99999 <-  as.character(names(
      which(table(total_imputed_set$CollegeID)<min_coll)))

    NorthEasternStates <- c("Sikkim","Meghalaya","Assam")
    UnionTerritory <- c("Union Territory","Jammu and Kashmir")

    total_imputed_set_2 <- total_imputed_set %>%
      mutate(CollegeState = as.character(CollegeState),
             CollegeState = replace_by(CollegeState,c("Maharashtra","Goa"),"Maharashtra & Goa"),
             CollegeState = replace_by(CollegeState,NorthEasternStates),
             CollegeState = replace_by(CollegeState,UnionTerritory),
             CollegeState = as.factor(CollegeState),
             GraduationYear = as.character(GraduationYear),
             GraduationYear = replace_by(GraduationYear,"GraduationYear_0","GraduationYear_2013"),
             GraduationYear = as.factor(GraduationYear)
      )
    lm_set <- total_imputed_set_2 %>%
      mutate(
        CollegeID = as.character(CollegeID),
        CollegeID2 = replace_by(CollegeID,college_ID_99999,replaceby = "9999"),
        CollegeID2 = as.factor(CollegeID2),
        Degree = as.character(Degree),
        Degree = ifelse(Degree=="M.Sc. (Tech.)","M.Tech./M.E.",Degree),
        Degree = as.factor(Degree)
      )%>%
      select(-CollegeID)%>%
      select(-CollegeCityID)

    state_other <-  as.character(names(
      which(table(lm_set$CollegeState)<min_state)))
    lm_set <- lm_set %>%
      mutate(
        CollegeState = as.character(CollegeState),
        CollegeState2 = replace_by(CollegeState,state_other),
        CollegeState2 = as.factor(CollegeState2)
      )%>%
      select(-CollegeState)
    set.seed(755, sample.kind = "Rounding")
    test_index <- createDataPartition(lm_set$CollegeID2, times = 1, p = 0.2,
                                      list = FALSE)
    train_set <- lm_set[-test_index,]
    test_set <- lm_set[test_index,]
    nrow(test_set)/(nrow(train_set)+nrow(test_set))

      factor_names <- lm_set %>%
        select(-where(is.numeric))%>%
        colnames()
      removed <- test_set %>%
        anti_join(train_set, by = factor_names)
      nrow(removed)
      test_set <- test_set %>%
        semi_join(train_set, by = factor_names)
      train_set <- train_set %>%
        rbind(removed)
      n=0
      while(nrow(test_set)/(nrow(train_set)+nrow(test_set))<0.2 & n<=100){
      n=n+1
      if(n%%10==0)
      {print(paste("Adjusting test set, test at",round((
        nrow(test_set)/(nrow(train_set)+nrow(test_set))
        )*100,digits = 2),"pc"))}

      set.seed(nrow(removed),sample.kind = "Rounding")

      readd_index <- sample(nrow(train_set),size = nrow(removed))
      test_set <- rbind(test_set, train_set[readd_index,])
      train_set<- train_set[-readd_index,]

      removed <- test_set %>%
        anti_join(train_set, by = factor_names)
      nrow(removed)
      test_set <- test_set %>%
        semi_join(train_set, by = factor_names)
      train_set <- train_set %>%
        rbind(removed)

      nrow(test_set)/(nrow(train_set)+nrow(test_set))
    }

if(nrow(test_set)/(nrow(train_set)+nrow(test_set))>=0.2){
        lm_obj <- train(Salary~. ,
                        data=train_set,
                        method="lm",
                        trControl = trainControl(method = "repeatedcv",
                                                 number = 10,
                                                 repeats = 10,
                                                 allowParallel = TRUE,
                                                 savePredictions = TRUE,
                                                 verboseIter = FALSE)
        )

        predicted_sal_lm = predict(lm_obj , test_set)
    res <- rbind(res,data.frame(MinColl= min_coll,
                                MinState=min_state,                           Partition=nrow(test_set)/(n
    RMSE=RMSE(predicted_sal_lm,test_set$Salary)))
} else {print("Minimum test set size not achieved. Skipping")}

}
}

res%>% view

res[which.min(res$RMSE),]

plot<-plot_ly(data = res,
              x=~MinColl,
              y=~MinState,
              z=~RMSE,
              color=~Partition,
              type="scatter3d",
              mode="markers")
plot
```

The above code finds the `min_coll` and `min_state` at 24 and 14. This means that we will not consider as discrete any colleges having less than 24 data points and any states having less than 14. This simplification is necessary to train algorithms. Without this, cross validation will fail often as the test data may contain all of the representation of a college or state and thus the algorithm may not have had a chance to train on the particular instance of the factor.

```
#25 and 9 were chosen by cross validation, code supplied in project

min_coll=25
min_state=9
```

```
  #Group together very similar states with very low representation
  #Also take care of stray graduation year = 0, replace by median (2013)

    NorthEasternStates <- c("Sikkim","Meghalaya","Assam")
    UnionTerritory <- c("Union Territory","Jammu and Kashmir")

#Make a list of colleges with under min_coll instances in data

  college_ID_99999 <-  as.character(names(
     which(table(total_imputed_set$CollegeID)<min_coll)))

  #Replace CollegeID with 9999 if they have under min_coll representation
  #Also take care of stray alternative representation of M.Tech degree.
  #Also take care of Graduation year = 0 and 2007 in single instances

    lm_set  <- total_imputed_set %>%
      mutate(CollegeState = as.character(CollegeState),
             CollegeState = replace_by(CollegeState,c("Maharashtra","Goa"),"Maharashtra & Goa"),
             CollegeState = replace_by(CollegeState,NorthEasternStates),
             CollegeState = replace_by(CollegeState,UnionTerritory),
             CollegeState = as.factor(CollegeState),
             GraduationYear =ifelse(GraduationYear %in% c(0,2007),2013,GraduationYear),
             GraduationYear =as.factor(GraduationYear),
             CollegeID = as.character(CollegeID),
             CollegeID2 = replace_by(CollegeID,college_ID_99999,replaceby = "9999"),
             CollegeID2 = as.factor(CollegeID2),
             Degree = as.character(Degree),
             Degree = ifelse(Degree=="M.Sc. (Tech.)","M.Tech./M.E.",Degree),
             Degree = as.factor(Degree)
      )%>%
      select(-CollegeID)

   #Make a list of states with under min_state instances in data

    state_other <-  as.character(names(
      which(table(lm_set$CollegeState)<min_state)))

    #Replace CollegeState by "other" in case of low representation

    lm_set <- lm_set %>%
      mutate(
        CollegeState = as.character(CollegeState),
        CollegeState2 = replace_by(CollegeState,state_other),
        CollegeState2 = as.factor(CollegeState2)
      )%>%
      select(-CollegeState)%>%
      select(-CollegeCityID) #Copy of college ID, needs to be removed

    #Set seed and replace

    set.seed(755, sample.kind = "Rounding")
```

```
## Warning in set.seed(755, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
#We use an 80/20 split in data per standard data science practice
test_index <- createDataPartition(lm_set$CollegeID2, times = 1, p = 0.2,
                                  list = FALSE)
train_set <- lm_set[-test_index,]
test_set <- lm_set[test_index,]
nrow(test_set)/(nrow(train_set)+nrow(test_set))
```

```
## [1] 0.200467
```

```r
#To ensure we don't have factors with instances in test_set that don't exist in train_set,
#we perform an antijoin on the factor names
  factor_names <- lm_set %>%
    select(-where(is.numeric))%>%
    colnames()


  removed <- test_set %>%
    anti_join(train_set, by = factor_names)
  nrow(removed)
```

```
## [1] 189
```

```r
  test_set <- test_set %>%
    semi_join(train_set, by = factor_names)
  train_set <- train_set %>%
    rbind(removed)


  #Since we reduce the test_set, we don't have 20% anymore in it.
  #The following code readds from the train_set to the test_set until it has at least 20% data

  n=0 # Iteration var to break out incase of infinite loop

  while(nrow(test_set)/(nrow(train_set)+nrow(test_set))<0.2 & n<=100){

  n=n+1

  #To give us an update of the current test set %
  if(n%%10==0) #to avoid apamming the cosole, we print at every 10th iteration
  {print(paste("Adjusting test set, test at",round((
    nrow(test_set)/(nrow(train_set)+nrow(test_set))
    )*100,digits = 2),"pc"))}


  #Seed set to number of removed rows to add repeatability while avoiding loss of randomness
  suppressWarnings(set.seed(nrow(removed) , sample.kind = "Rounding"))

  #select number of removed rows at random from train set
  readd_index <- sample(nrow(train_set),size = nrow(removed))
```

```
      #add rows back to test set
      test_set <- rbind(test_set, train_set[readd_index,])

      #remove rows from train set
      train_set<- train_set[-readd_index,]

      #check for representations with no match in train set
      removed <- test_set %>%
        anti_join(train_set, by = factor_names)
      nrow(removed)
      test_set <- test_set %>%
        semi_join(train_set, by = factor_names)
      train_set <- train_set %>%
        rbind(removed)

      #repeat process because we may have removed rows.
      nrow(test_set)/(nrow(train_set)+nrow(test_set))


    }

 #At this point we have the train and test set we will use for all subsequent models.

      gc()
```

```
##           used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells 2818131 150.6    4424824 236.4  4424824 236.4
## Vcells 5477940  41.8   12270676  93.7 12270676  93.7
```

The lm code is relatively simple. We use 10 fold cross validation using the `trcontrol` parameter in the caret train model.

```
knitr::opts_chunk$set(cache = TRUE)
 #Train lm algorithm using caret. lm doesn't need tuning parameters.
  lm_obj <- lm(Salary~. , data=train_set)
  lm_obj <- train(Salary~. ,
                  data=train_set,
                  method="lm",
                  trControl = trainControl(method = "repeatedcv",
                                                   number = 10,
                                                   repeats = 10,
                                                   allowParallel = TRUE,
                                                   savePredictions = TRUE,
                                                   verboseIter = FALSE)
                  )


  predicted_sal_lm = predict(lm_obj , test_set)
  #RMSE(predicted_sal_lm,test_set$Salary)
  RMSE_lm=  RMSE(predicted_sal_lm,test_set$Salary)
  RMSE_lm
```

```
## [1] 120890.6
```

Decision Trees and Random Forest

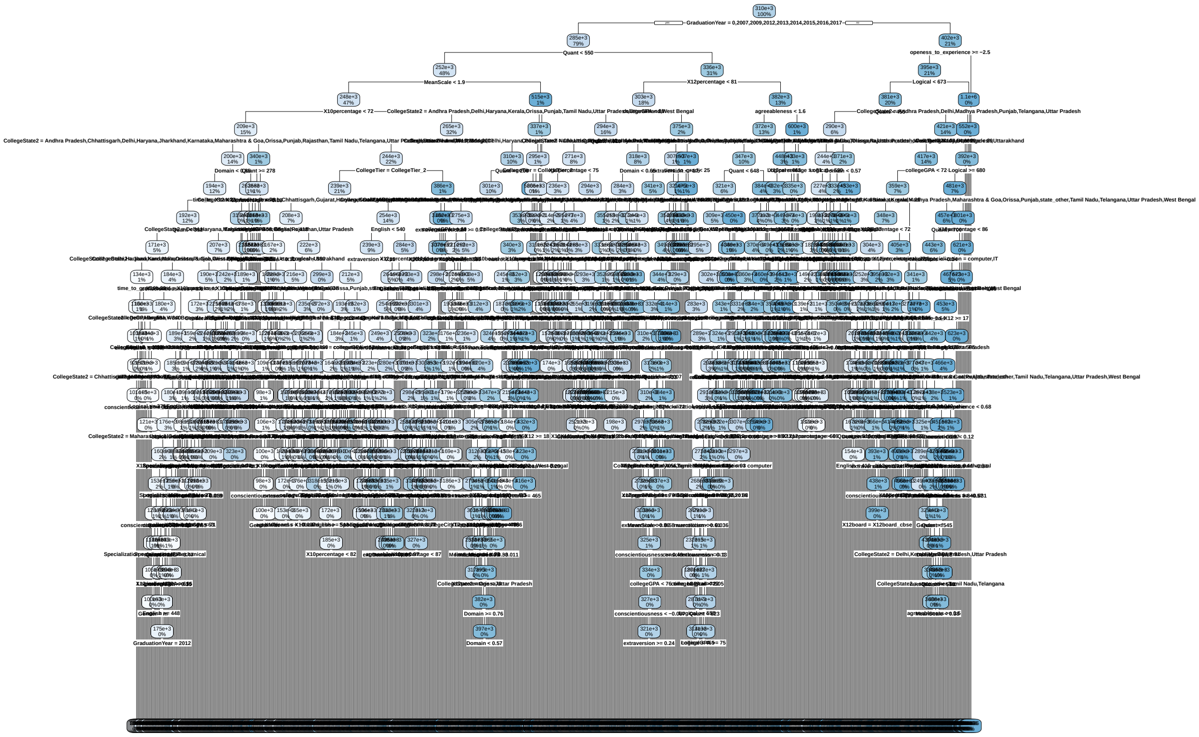
In the models for which we have tuning parameters available, like those in this section, we will need to find the ideal value for them. A simple way to achieve this is to extend the range on the side in case out best tune is chosen as one of the limits of the tune grid that we supply.

Decision Trees

Decision tree is a simple machine learning algorithm used for both classification as well as regression. The goal is to create a model that predicts the value of the target variable (here, Salary) by learning simple decision rules inferred from the data. Each feature of the data is split multiple times. Each split is associated with a cost function which the algorithm tries to minimize to try and yield combinations of predictors which give the minimum deviation from the target value of the prediction. Decision trees, when small can be easily represented visually. In our case, the data is too complex to get an intuitive understanding from the tree as you will see in the following unwieldy plot. The rules closer to the root of the tree (on top) are more important. It is thus possible to 'prune' the trees and get rid of more complex rules near the bottom which may be overfitting. However, basic DT is too simplistic to give an accurate prediction so we will not be fine-tuning our DT model here.

```r
knitr::opts_chunk$set(cache = TRUE)

set.seed(42)
fit_dt <- rpart(Salary~.,
             data = train_set,
             method = "anova",
             control = rpart.control(xval = 10,
                                     minbucket = 2,
                                     cp = 0),
             parms = list(split = "information"))

rpart.plot(fit_dt, extra = 100)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```r
predicted_dt <- rpart.predict(fit_dt,
                newdata = test_set)

RMSE_dt = RMSE(predicted_dt,test_set$Salary)
RMSE_dt
```

```
## [1] 191947.2
```

```r
gc()
```

```
##              used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5459793 291.6    9378301 500.9  9378301 500.9
## Vcells 13407275 102.3   36450541 278.1 56953970 434.6
```

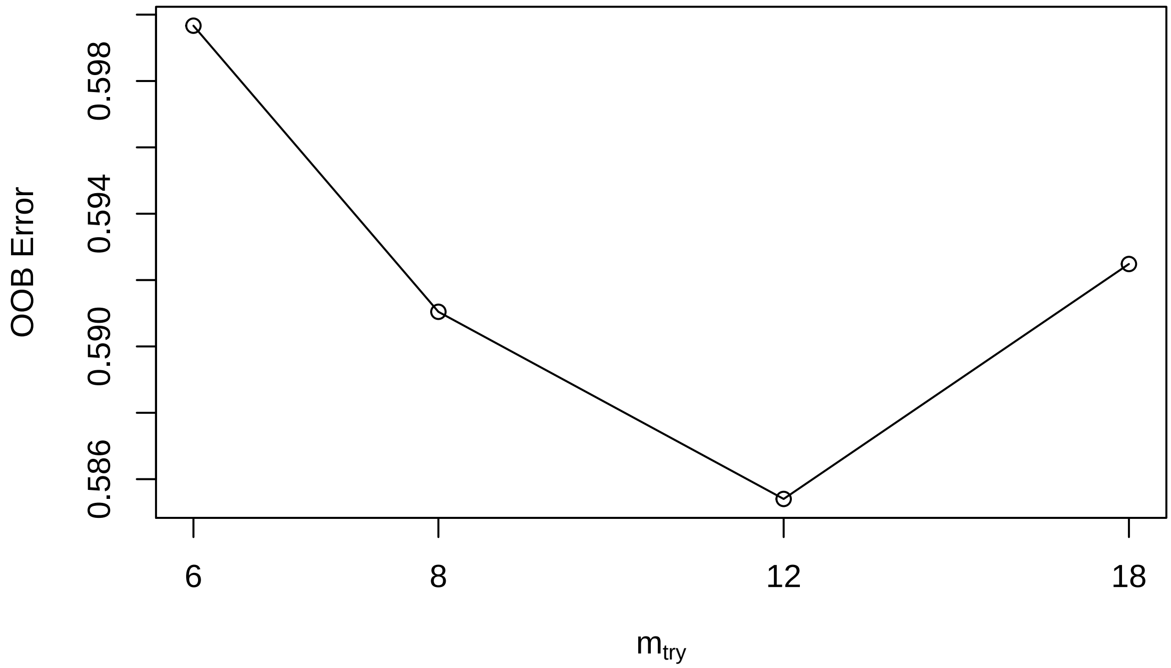
Random Forest

Decision trees are unstable meaning that small changes in the data can result in a completely different tree. To deal with this issue, we use boosting, a way of training and then de-correlating multiple trees in the algorithm RandomForest. The following code implements a random forest from a package of the same name.

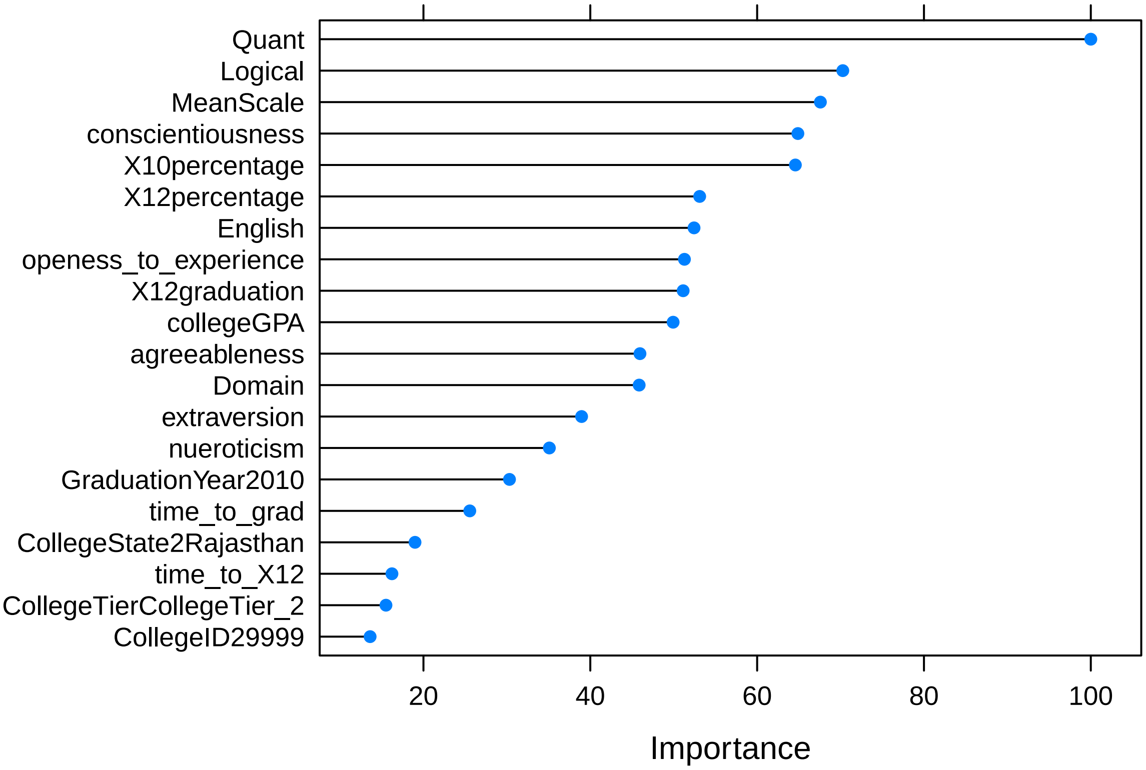
```r
knitr::opts_chunk$set(cache = TRUE)
set.seed(42)
```

```
#There is a function available to find the best value of mtry,
#the only tuning parameter available to #us in RandomForest.
#mtry is the number of predictors sampled for splitting at each node.

bestmtry <- tuneRF(x=train_set[,-24], #24th column is Salary. These are our predictors
                   y=train_set[,24], #This is only the Salary, so the predicted var
                   data = train_set
                   , stepFactor=1.5, improve=1e-5, ntree=500) %>%
  as.data.frame()
```

```
## mtry = 8  OOB error = 0.5910437
## Searching left ...
## mtry = 6     OOB error = 0.599667
## -0.01459001 1e-05
## Searching right ...
## mtry = 12    OOB error = 0.5854033
## 0.009543056 1e-05
## mtry = 18    OOB error = 0.5924842
## -0.01209575 1e-05
```



```
mtry = bestmtry[which.min(bestmtry$OOBError),1] #Select mtry with lowest out-of-box error (12).

mytuneGrid = expand.grid(mtry=mtry)

model_rf <- train(Salary ~ .,
                         data = train_set,
                         method = "rf",
                         preProcess = c("scale", "center"),
                         trControl = trainControl(method = "repeatedcv",
                                                  number = 5,
                                                  repeats = 3,
                                                  savePredictions = TRUE,
                                                  verboseIter = FALSE),
                  tuneGrid=mytuneGrid)

importance <- varImp(model_rf, scale = TRUE)
plot(importance,top = 20)
```



```
prediction_rf = predict(model_rf, newdata = test_set)
RMSE_rf <- RMSE(prediction_rf, test_set$Salary)
RMSE_rf
```

```
## [1] 117523.3
```

```
gc()
```

```
##           used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells 2933579 156.7    4424824 236.4  4424824 236.4
## Vcells 9249694  70.6   32861320 250.8 37881298 289.1
```

We see that RF dramatically improved the RMSE from DT and also that it beat the lm RMSE in prediction, if only by a small margin. The importance plot of the variables tells us exactly which variables contribute to the best Salary. Unsurprisingly, Quant and MeanScale are at the top. Engineers with good quant skills get paid higher. It is also interesting to note that the next two most important variables are performance in 10th and 12th standard, in that order. Hence, students who performed well then can expect to have their efforts rewarded in the future.

Random Forest XGBoost

No discussion about randomforest is complete without mentioning XGBoost, an algorithm which has had a lot of success in recent machine learning competitions on Kaggle (an ML resources website). XGBoost stands for “Extreme Gradient Boosting”. Random Forest trains multiple trees and aggregates them to create an more stable ensemble model. It does this by a process called bagging (also called bootstrap aggregating) by which it decreases variance (variance is the degree to which these predictions vary between model iterations) and avoids overfitting.

Gradient boosting, unlike bagging algorithms which only control for variance, also controls for bias(bias is how removed a model’s predictions are from correctness).

Extreme gradient boosting is an extension to gradient boosted decision trees specifically designed to improve speed and performance, in part by using the capacity of parallel processing available in most modern computers.

```
knitr::opts_chunk$set(cache = TRUE)

grid_xgb <- expand.grid(
  nrounds = c(65,70,80), #number of rounds, tunes to 70
  max_depth = c(5,6,7), #max depth of tree: tunes to 6, Range [0,Inf]
  eta = c(0,0.05,0.1,0.15), #step size shrinkage: tunes to 0.05 . Range [0,Inf]
  gamma = c(0,1), #minimum loss reduction for further partion: tunes to 0. Range [0,Inf]
  colsample_bytree = seq(0.3, 0.9, length.out = 5), #subsampling parameters, tunes to 0.5 Range[0,1]
  min_child_weight = c(0,1), #minimum sum of instance weight needed in a child, tunes to 1.
  subsample = c(0.4,0.5,0.75,1) #subsampling of instance to control overfitting.Tunes to 0.5 Range(0,1]
)

set.seed(42)
model_xgb <- train(Salary ~ .,
                          data = train_set,
                          method = "xgbTree",
                          preProcess = c("scale", "center"),
                          trControl = trainControl(method = "repeatedcv",
                                                   number = 5,
                                                   repeats = 3,
                                                   allowParallel = TRUE,
                                                   savePredictions = TRUE,
                                                   verboseIter = FALSE),
                   objective = "reg:squarederror",
                   tuneGrid = grid_xgb)


prediction_xgb = predict(model_xgb, newdata = test_set)
RMSE_xgb <- RMSE(prediction_xgb, test_set$Salary)



RMSE_xgb
```

```
## [1] 116031.4
```

We can observe that XGboost performs well and executes fast doesn't fare as well as RF in the RMSE. At this scale (of 3000 rows), we are better off without sacrificing accuracy for performance.

Neural Network

Artificial Neural Networks are a family of algorithms inspired from the human brain. Artificial Neural Networks (ANNs in short), in their most common implementation, are made up of stacked layers of 'neurons' with the input layer on the left and output layer on the right(feedforward network). Multiple 'hidden layers' of neurons exist between the input and output layer with adjustable 'weights'. When it is being 'trained', Patterns of input information are fed into the input units and are subsequently multiplied (the operation can be a more complex mathematical function). If the resulting value is greater than a threshold, the neuron transmits the result to the next layer. The thresholds and weights are chosen at random to begin with. This process continues until the output layer which is made up of the expected outputs (in our case, the Salary). The neuron paths which resulted in minimum error are retained and others are adjusted to minimize the error by a feedback process called "backpropagation". Neural networks are computationally very expensive but have been wildly successful for some tasks such as image recognition.

```
model_nn$bestTune
```

```
##   size decay
## 4   13 0.025
```

```
#Best Tune acheived within search range. Model is tuned.



predicted_salaries = predict(model_nn, newdata=test_set )

RMSE_nn = RMSE(predicted_salaries, test_set$Salary )
RMSE_nn
```

```
## [1] 124021.4
```

We achieve an RMSE of 124 K.

Results

```
rmse_results <- rbind(RMSE_lm,RMSE_dt,RMSE_rf,RMSE_xgb,RMSE_nn) %>%
  as.data.frame()%>%
  `colnames<-`("RMSE")%>%
  mutate(RMSE_by_sd = paste(round(RMSE/sd(lm_set$Salary)*100,1),"%"))


kable(rmse_results) %>%
```

	RMSE	RMSE_by_sd
RMSE_lm	**120890.6**	56.9 %
RMSE_dt	**191947.2**	90.4 %
RMSE_rf	**119104.8**	56.1 %
RMSE_xgb	**121693.6**	57.3 %
RMSE_nn	**124021.4**	58.4 %

```
kable_styling(bootstrap_options = "striped" , full_width = F , position = "center") %>%
kable_styling(bootstrap_options = "bordered", full_width = F , position ="center") %>%
column_spec(1,bold = T ) %>%
column_spec(2,bold =T ,color = "white" , background ="#D7261E")
```

We see that both linear regression and random forest are very succesful in predicting the Salary with RMSEs of around 120 K Rs by year. This amounts to 160 $ by today's exchange rate (Jul 7 2021). I have included the calculation RMSE/sd to make it more easy to understand how effectively we were able to predict the Salary of these individuals.

This project has explored the challenges of applying machine learning techniques to raw dirty data. The data had to be standardized (as in the case of various allied engineering disciplines and boards) and then reduced in complexity (with our min_coll and min_state) parameters. We found innovative ways of reducing the NAs by first encoding the information from various tests into one variable and also by imputation for fringe cases. To make our model robust, we also excluded outliers and changed variables like date of birth into other variables such as time_to_graduate which would logically have a greater impact on the salary outcome. For data such as ours, we have found that the RandomForest algorithm gives the best results by bagging different decision trees but also that Linear Regression gives results almost at par in a much shorter time span. We can also conclude that feed-forward neural network with 10 hidden layers is not worth the computational cost.

Our analysis has also resulted in a deeper understanding of the important factors for getting a high salary for aspiring engineering graduates. From the importance plot of our RF model, reproduced here, we can clearly see that Quantitative ability followed by Conscientiousness and English are the decisive elements for higher salaries.

This analysis could be further improved by trying out more of the 233 algorithms that caret implements. We could for instance, use ridge regression to offset the multicollinearity between our variables (conscientiousness, MaxScaledScore, MeanScaledScore etc.). A more elegant way could be found to split the test set and train set so as to include all levels of factors in each. Moreover, we could train the same algorithms while leaving out certain predictors which have the least effect in PCA1 and PCA2 or which appear at the bottom in variable importance plot from RF.