BUS2004 W11

Saher Manaseer

2022-10-06

```
invisible(library(tidyverse))
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

The dataset for this assignment is the anonymised Open University Learning Analytics Dataset (OULAD). OULAD contains data about courses, students, and their interactions with seven selected courses (also called modules) in a Virtual Learning Environment (VLE). Specifically, this dataset consists of:

- courses.csv, which contains the list of all available modules and their presentations;
- assessments.csv, which contains information about assessments in module-presentations;
- vle.csv, which contains information about the available materials in the VLE;
- studentInfo.csv, which contains demographic information about the students together with their results;
- studentRegistration.csv, which contains information about the time when the student registered for the module presentation;
- studentAssessment.csv, which contains the results of students' assessments;
- studentVle.csv, which contains information about each student's interactions with the materials in the VLE.

Q1 Read the data from studentInfo.csv. How many columns and rows are there in the data? Please display the first 10 records as well as the last 10 records of the data

```
studentRAW <- read_csv("studentInfo.csv")
```

```
## Rows: 32593 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): code_module, code_presentation, gender, region, highest_education, ...
## dbl (3): id_student, num_of_prev_attempts, studied_credits
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_dimensions = dim(studentRAW)
cat("The dataset contains (",data_dimensions[1],") rows and (",data_dimensions[2],
      ") Columns\n")
```

```
## The dataset contains ( 32593 ) rows and ( 12 ) Columns
```

```
head(studentRAW,10)
```

```
## # A tibble: 10 x 12
##    code_~1 code_~2 id_st~3 gender region highe~4 imd_b~5 age_b~6 num_o~7 studi~8
##    <chr>   <chr>     <dbl> <chr>  <chr>  <chr>   <chr>   <chr>     <dbl>   <dbl>
##  1 AAA     2013J     11391 M      East ~ HE Qua~ 90-100% 55<=          0     240
##  2 AAA     2013J     28400 F      Scotl~ HE Qua~ 20-30%  35-55         0      60
##  3 AAA     2013J     30268 F      North~ A Leve~ 30-40%  35-55         0      60
##  4 AAA     2013J     31604 F      South~ A Leve~ 50-60%  35-55         0      60
##  5 AAA     2013J     32885 F      West ~ Lower ~ 50-60%  0-35          0      60
##  6 AAA     2013J     38053 M      Wales  A Leve~ 80-90%  35-55         0      60
##  7 AAA     2013J     45462 M      Scotl~ HE Qua~ 30-40%  0-35          0      60
##  8 AAA     2013J     45642 F      North~ A Leve~ 90-100% 0-35          0     120
##  9 AAA     2013J     52130 F      East ~ A Leve~ 70-80%  0-35          0      90
## 10 AAA     2013J     53025 M      North~ Post G~ <NA>    55<=          0      60
## # ... with 2 more variables: disability <chr>, final_result <chr>, and
## #   abbreviated variable names 1: code_module, 2: code_presentation,
## #   3: id_student, 4: highest_education, 5: imd_band, 6: age_band,
## #   7: num_of_prev_attempts, 8: studied_credits
```

```
tail(studentRAW,10)
```

```
## # A tibble: 10 x 12
##    code_~1 code_~2 id_st~3 gender region highe~4 imd_b~5 age_b~6 num_o~7 studi~8
##    <chr>   <chr>     <dbl> <chr>  <chr>  <chr>   <chr>   <chr>     <dbl>   <dbl>
##  1 GGG     2014J   2508153 F      East ~ Lower ~ 10-20   0-35          0      30
##  2 GGG     2014J   2533195 F      South~ Lower ~ 10-20   0-35          0      30
##  3 GGG     2014J   2606765 F      Londo~ Lower ~ 80-90%  0-35          0      30
##  4 GGG     2014J   2608143 M      East ~ HE Qua~ 60-70%  35-55         0      30
##  5 GGG     2014J   2620947 F      Scotl~ A Leve~ 80-90%  0-35          0      30
##  6 GGG     2014J   2640965 F      Wales  Lower ~ 10-20   0-35          0      30
##  7 GGG     2014J   2645731 F      East ~ Lower ~ 40-50%  35-55         0      30
##  8 GGG     2014J   2648187 F      South~ A Leve~ 20-30%  0-35          0      30
##  9 GGG     2014J   2679821 F      South~ Lower ~ 90-100% 35-55         0      30
## 10 GGG     2014J   2684003 F      Yorks~ HE Qua~ 50-60%  35-55         0      30
## # ... with 2 more variables: disability <chr>, final_result <chr>, and
## #   abbreviated variable names 1: code_module, 2: code_presentation,
## #   3: id_student, 4: highest_education, 5: imd_band, 6: age_band,
## #   7: num_of_prev_attempts, 8: studied_credits
```

Q2 How many unique "region" values are there in the data file? How many "region" values contain the keywords "North" and "South", respectively?

```
uniq_Regions<-studentRAW%>%
  select(region)%>%
  unique()

cat("There are :",dim(uniq_Regions)[1], "unique Regions")
```

```
## There are : 13 unique Regions
```

```
paste(" The number of regions with 'North' is", sum(grepl("North", uniq_Regions$region)))
```

```
## [1] " The number of regions with 'North' is 2"
```

```
paste(" The number of regions with 'South' is", sum(grepl("South", uniq_Regions$region)))
```

```
## [1] " The number of regions with 'South' is 3"
```

Q3 How many columns contain missing values? List each of these columns (i.e., column name) with the corresponding missingness percentages.

```
library(naniar)
miss_var_summary(studentRAW)%>%
  filter(n_miss>0)
```

```
## # A tibble: 1 x 3
##   variable n_miss pct_miss
##   <chr>     <int>    <dbl>
## 1 imd_band   1111     3.41
```

```
# students should only produce one column
```

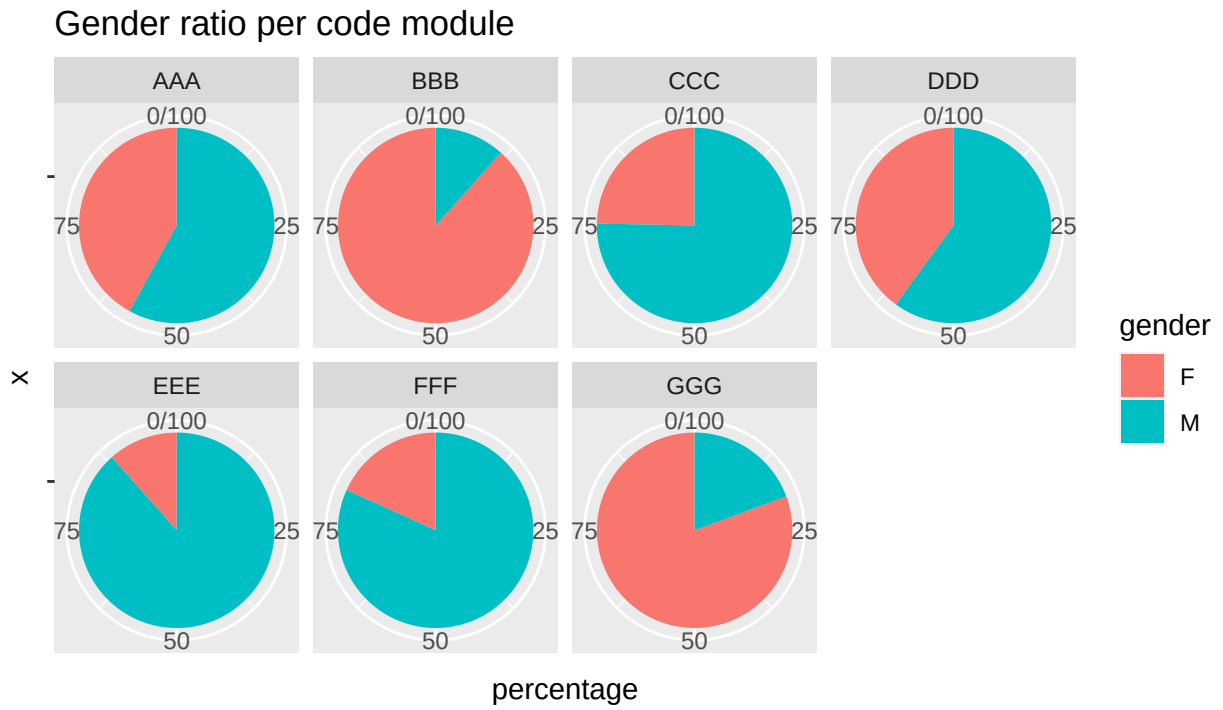
Q4 How many unique students are there in the data file? How many male and female students are there, respectively?

```
unique_students_count <-studentRAW%>%
  select(id_student, gender)%>%
  unique()%>%
  count()


unique_female_count <-studentRAW%>%
  select(id_student, gender)%>%
  unique()%>%
  filter(gender == 'F')%>%
```

```
  count()

unique_male_count <-studentRAW%>%
  select(id_student, gender)%>%
  unique()%>%
  filter(gender == 'M')%>%
  count()
```

Q5 For each code_module value, write code to calculate the fractions of students of different gender values (M and F), and then use a pie chart to visualise the fraction numbers specific to the code_module value.

```
Q5<-studentRAW%>%
  select(code_module,id_student, gender)%>%
  group_by(code_module,gender)%>%
  unique()%>%
  count()

Q5<-Q5%>%
  group_by(code_module)%>%
  mutate(total = sum(n))%>%
  mutate(percentage = n/total * 100)%>%
  select(everything(), -total)

Q5%>%
  ggplot( aes(x="", y=percentage , fill=gender)) +
  geom_bar(width = 1, stat="identity") +
  ggtitle("Gender ratio per code module") +
  coord_polar("y") +
  facet_wrap(~code_module, ncol = 4)
```



Q6 In order to get insights on the study loads of students, let us look at the studied_credit attribute which describes the total number of credits for the modules the student is currently studying. What is the largest value in the column "studied_credits"? Please sort the data according to the column studied_credits in an ascending manner and display the first 10 records of the data.

```
cat("The maximum studied_credit is", max(studentRAW$studied_credits))
```

```
## The maximum studied_credit is 655
```

```
studentRAW%>%
  filter(!is.na(studied_credits))%>%
  arrange(studied_credits)%>%
  head(10)
```

```
## # A tibble: 10 x 12
##    code_~1 code_~2 id_st~3 gender region highe~4 imd_b~5 age_b~6 num_o~7 studi~8
##    <chr>   <chr>     <dbl> <chr>  <chr>  <chr>   <chr>   <chr>     <dbl>   <dbl>
##  1 CCC     2014B     28418 F      West ~ A Leve~ 20-30%  0-35          0      30
##  2 CCC     2014B     40333 M      North~ HE Qua~ 0-10%   35-55         0      30
##  3 CCC     2014B     40604 M      Irela~ A Leve~ <NA>    35-55         0      30
##  4 CCC     2014B     59541 M      East ~ A Leve~ 40-50%  0-35          0      30
##  5 CCC     2014B     66254 F      Irela~ A Leve~ <NA>    0-35          0      30
```

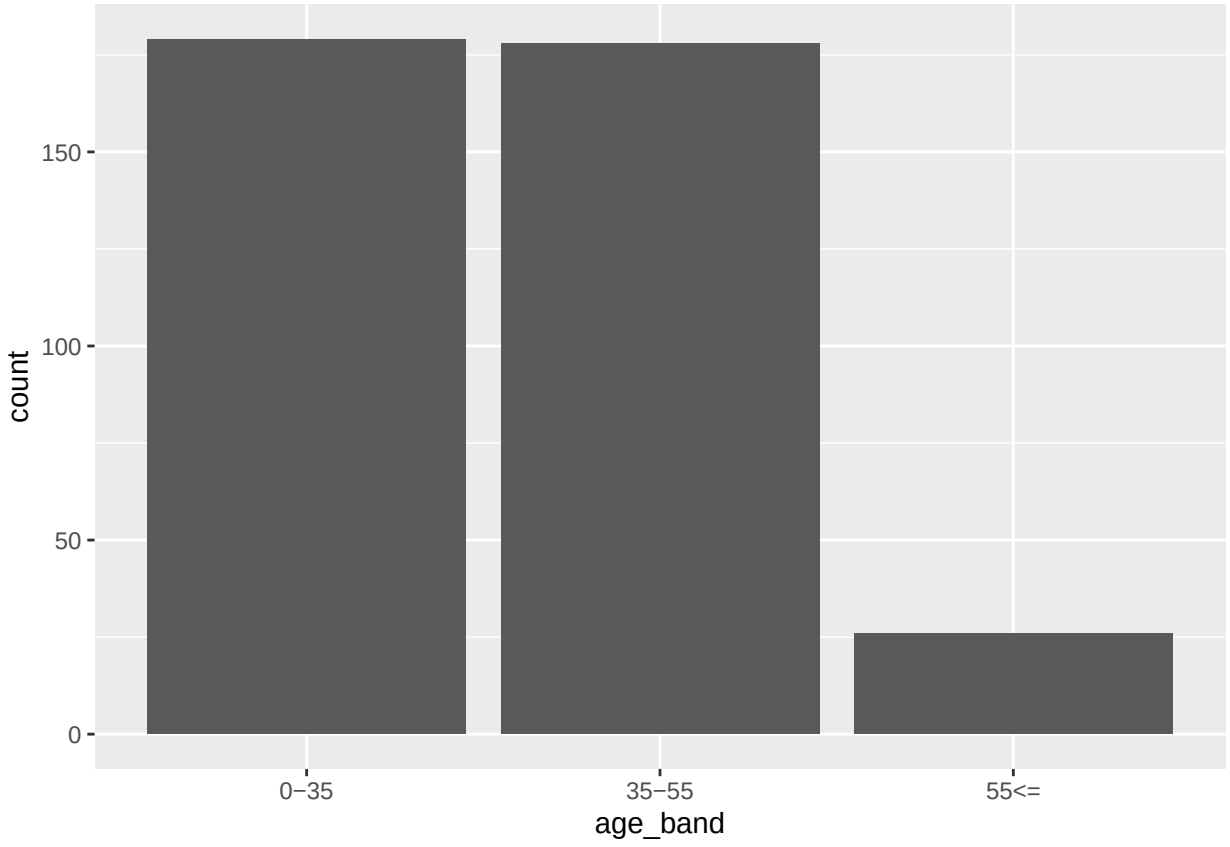
```
##  6 CCC     2014B     70381 M      Scotl~ HE Qua~ 30-40%  35-55         0      30
##  7 CCC     2014B     75728 F      South~ HE Qua~ 30-40%  35-55         0      30
##  8 CCC     2014B     82575 M      West ~ HE Qua~ 80-90%  0-35          0      30
##  9 CCC     2014B     83453 M      West ~ HE Qua~ 40-50%  35-55         0      30
## 10 CCC     2014B     86047 F      Wales  HE Qua~ 20-30%  0-35          0      30
## # ... with 2 more variables: disability <chr>, final_result <chr>, and
## #   abbreviated variable names 1: code_module, 2: code_presentation,
## #   3: id_student, 4: highest_education, 5: imd_band, 6: age_band,
## #   7: num_of_prev_attempts, 8: studied_credits
```

Q7 Write code to only keep data records that are about the code_module of AAA and code_presentation of 2013J and save them in a file named "studentInfo_AAA_2013J.csv".

```
studentInfo_AAA_2013J <- studentRAW%>%
  filter((code_module == 'AAA') & (code_presentation == '2013J'))
write_csv(studentInfo_AAA_2013J,"studentInfo_AAA_2013J.csv")
```

Use a bar chart to plot the number of students of different "age_band" values in the student cohort we stored in "studentInfo_AAA_2013J.csv".

```
studentInfo_AAA_2013J%>%
  ggplot(aes(x=age_band))+
  geom_bar()
```



Q8 Write code to read the data from studentAssessment.csv and write code to add a new column named "code_module" (i.e., the module that an id_assessment is related to) and fill it with corresponding values, which can be retrieved from the file assessments.csv. Tip: you may need to use the join function in R.

```
studentAssessment <- read_csv("studentAssessment.csv")
```

```
## Rows: 173912 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): id_assessment, id_student, date_submitted, is_banked, score
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
assesment <- read_csv("assessments.csv")
```

```
## Rows: 206 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): code_module, code_presentation, assessment_type
## dbl (3): id_assessment, date, weight
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
studentAssessment <- left_join(studentAssessment,assesment)[,1:7]
```

```
## Joining, by = "id_assessment"
```

Q9 Write code to display the statistical information, i.e., Min, Max, and Mean, of students' assessment scores (i.e., the column "score") for each module. Which module has the largest Mean value of assessment scores? What observations can you make?

```
#Method 1
studentAssessment %>%
  group_by(code_module) %>%
  filter(!is.na(score))%>%
  summarize(min = min(score),
            median = median(score),
            mean = mean(score),
            max = max(score))
```

```
## # A tibble: 7 x 5
##   code_module   min median  mean   max
##   <chr>       <dbl>  <dbl> <dbl> <dbl>
## 1 AAA             0     71  69.0    98
## 2 BBB             0     80  76.7   100
## 3 CCC             0     78  73.3   100
```

```
## 4 DDD             0     74  70.1   100
## 5 EEE             0     85  81.2   100
## 6 FFF             0     80  77.7   100
## 7 GGG             0     80  79.7   100
```

```
#Method 2
  tapply(studentAssessment$score, studentAssessment$code_module, summary)
```

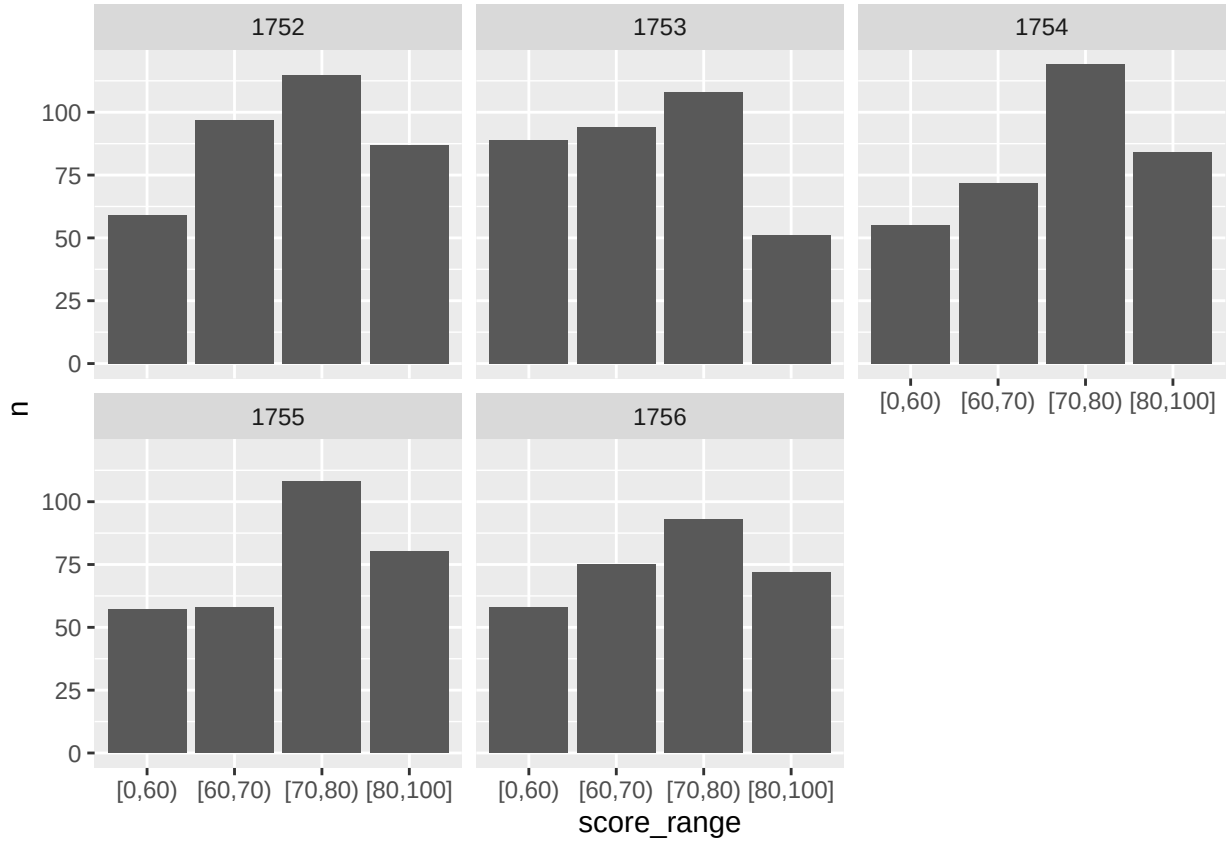
```
## $AAA
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   62.00   71.00   69.03   78.00   98.00       3
##
## $BBB
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   65.00   80.00   76.71   95.00  100.00      53
##
## $CCC
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   59.00   78.00   73.26   92.00  100.00      11
##
## $DDD
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   58.00   74.00   70.09   86.00  100.00      49
##
## $EEE
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   74.00   85.00   81.18   92.00  100.00       7
##
## $FFF
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   70.00   80.00   77.71   88.00  100.00      46
##
## $GGG
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0    65.0    80.0    79.7   100.0   100.0       4
```

Q10 Write code to only keep assessment data relevant to students from the code_module of AAA and code_presentation of 2013J, then write code to add a new column named "score_range" and fill it with one of the following values based on the corresponding "score" value:

```
studentAssessment<-studentAssessment%>%
  mutate(score_range = paste(ifelse(studentAssessment$score <60,'[0,60)',
                                    ifelse(studentAssessment$score <70,'[60,70)',
                                           ifelse(studentAssessment$score <80,
                                                  '[70,80)','[80,100]') ))))%>%filter(!is.na(score))
```

For each id_assessment value, use a bar chart to plot the number of students of different score_range values. What do you observe?

```
DF<-studentAssessment%>%
  filter(!is.na(score_range))%>%
  filter(code_module == 'AAA')%>%
  filter (code_presentation == '2013J')%>%
  group_by(id_assessment, score_range)%>%
  count()

DF%>%
  ggplot(aes(x=score_range, y = n))+
  geom_bar(stat = "identity")+
  facet_wrap(DF$id_assessment)
```



Now, Lets try some regression work

```
library(tidyverse)
library(Metrics)
library(rpart)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
```

```
##   method from
##   +.gg   ggplot2
```

```
library(explore)
```

```
##
## Attaching package: 'explore'
```

```
## The following object is masked from 'package:GGally':
##
##     rescale01
```

```
#Read data
All<-read_csv("predictive_twitter_data.csv")
```

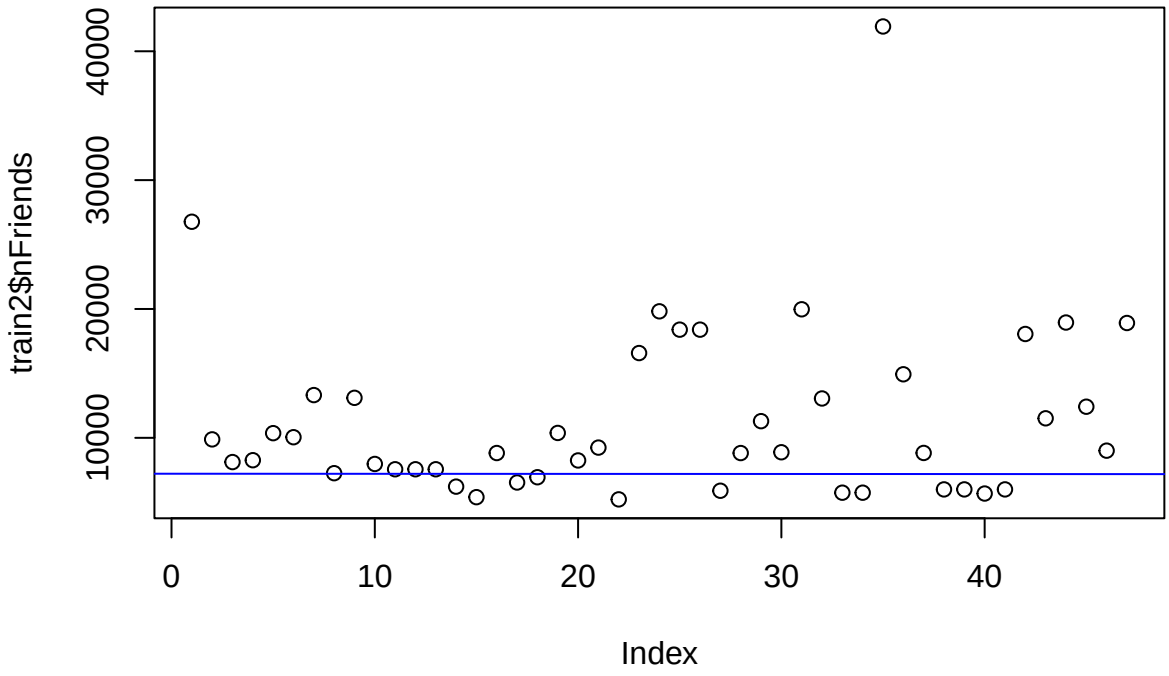
```
## Rows: 39955 Columns: 25
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (25): text_score, text_score_expansion, hashtag, hasURL, isReply, length...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
All <- All[1:1000,]
All<- filter(All, nFriends>5000)
# Adding an index column
All$record_ID <- seq.int(nrow(All))



#Reorder columns
All <- select(All, record_ID,everything())


#Create training and testing data
dt = sample(nrow(All),nrow(All)*0.80)
train <- All[dt,]
test <- All[-dt,]

#Remove some of the columns
train2<- select(All, record_ID, twitterAge, nFollowers, nFavorties, nFriends )


# Display the remaining columns
#colnames(train2)

#explore(train2)
```

```
#First model : Trivial
modl1 <- lm(nFriends ~ ., data=train2)
train2$modl1 <-  predict(modl1, new_data = test)
plot(train2$nFriends)
abline(modl1, col="blue")
```

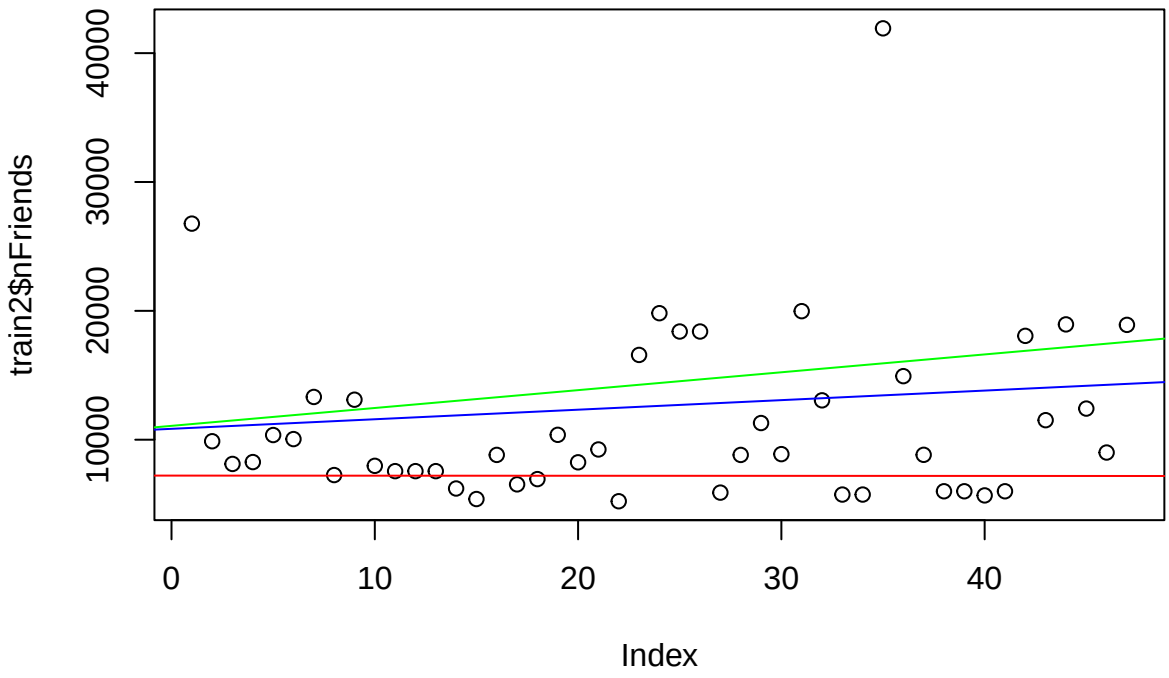
```
## Warning in abline(modl1, col = "blue"): only using the first two of 5 regression
## coefficients
```



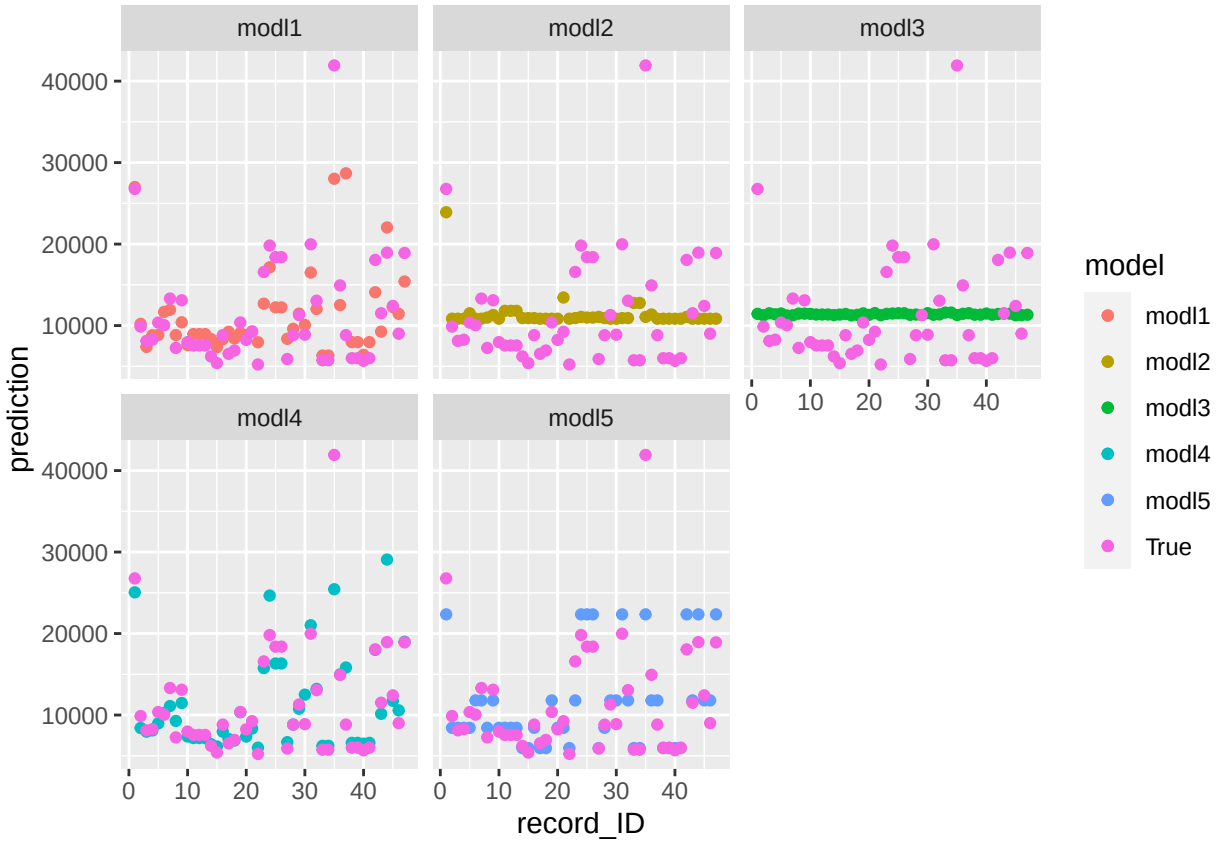

```
#Second model : Selected attributes
modl2 <- lm(nFriends ~ nFavorties , data=train2)
modl3 <- lm(nFriends ~ twitterAge, data=train2)
train2$modl2 <-  predict(modl2, new_data = test)
train2$modl3 <-  predict(modl3, new_data = test)

plot(train2$nFriends)
abline(modl1, col="red")
```

```
## Warning in abline(modl1, col = "red"): only using the first two of 5 regression
## coefficients
```

```
abline(modl2, col="blue")
abline(modl3, col="green")
```




```
#Second model : Selected attributes
#Third model : High degree polynomial of selected attributes
modl4 <- lm(nFriends ~ poly(nFollowers, degree= 3), data=train2)
train2$modl4 <-  predict(modl4, new_data = test)

# Fifth model : Regression Tree
modl5 <- rpart(nFriends ~ ., data=train2)
train2$modl5 <-  predict(modl5, new_data = test)

xxd<-train2%>%
  gather(model, prediction,-record_ID, -twitterAge, -nFollowers,-nFavorties, -nFriends)

ggplot(data = xxd,aes(x = record_ID,y=prediction, col = model))+
  geom_point()+
  geom_point(data = xxd, aes(y =nFriends, col = "True"))+
  facet_wrap(~model)
```


modl1
modl2
modl3
modl4
modl5
prediction
record_ID
model
modl1
modl2
modl3
modl4
modl5
True