21205992 Vivek Bulani R Final Project

Vivek Bulani 21205992

12/20/2021

STAT40620 - Data Programming with R - Final Project

VIVEK MANMOHAN BULANI - 21205992

For this Final project, the dataset which has been used is Student Alcohol Consumption Dataset from Kaggle. It has the data which is obtained in a survey of students studying Mathematics and Portuguese language courses in secondary school. To limit the scope for this project, only the data of students studying mathematics has been analysed here.

The different attributes/columns in the student-mat.csv (Math course) dataset is :-

1. school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2. sex - student's sex (binary: 'F' - female or 'M' - male)
3. age - student's age (numeric: from 15 to 22)
4. address - student's home address type (binary: 'U' - urban or 'R' - rural)
5. famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6. Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7. Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary 8) education or 4 – higher education)
8. Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9. Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
10. Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
11. reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12. guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13. traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)

14. studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15. failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16. schoolsup - extra educational support (binary: yes or no)
17. famsup - family educational support (binary: yes or no)
18. paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19. activities - extra-curricular activities (binary: yes or no)
20. nursery - attended nursery school (binary: yes or no)
21. higher - wants to take higher education (binary: yes or no)
22. internet - Internet access at home (binary: yes or no)
23. romantic - with a romantic relationship (binary: yes or no)
24. famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25. freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26. goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27. Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28. Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29. health - current health status (numeric: from 1 - very bad to 5 - very good)
30. absences - number of school absences (numeric: from 0 to 93)

The grades in the Math subject :-

31. G1 - first period grade (numeric: from 0 to 20)
32. G2 - second period grade (numeric: from 0 to 20)
33. G3 - final grade (numeric: from 0 to 20, output target)

```
library(gtsummary)
library(GGally)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggplot2)
```

1. gtsummary package :-

• This package is used to display the descriptive statistics for (continuous, categorical, and dichotomous) variables in a beautiful way which is easy to understand and infer from.

• The function tbl_summary() of this package is used for achieving above desired results.

• It prints the count of each value in each column and its percentage of occurrence out of the total number of rows

2. GGally package :-

• This package is used to plot clean pair-wise scatter plots using ggpairs() function.

```
citation("gtsummary")
```

```
##
## To cite gtsummary in publications use:
##
##   Sjoberg DD, Whiting K, Curry M, Lavery JA, Larmarange J. Reproducible
##   summary tables with the gtsummary package. The R Journal
##   2021;13:570–80. https://doi.org/10.32614/RJ-2021-053.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{gtsummary,
##     author = {Daniel D. Sjoberg and Karissa Whiting and Michael Curry and Jessi
ca A. Lavery and Joseph Larmarange},
##     title = {Reproducible Summary Tables with the gtsummary Package},
##     journal = {{The R Journal}},
##     year = {2021},
##     url = {https://doi.org/10.32614/RJ-2021-053},
##     doi = {10.32614/RJ-2021-053},
##     volume = {13},
##     issue = {1},
##     pages = {570-580},
##   }
```

```
citation("GGally")
```

```
##
## To cite package 'GGally' in publications use:
##
##   Barret Schloerke, Di Cook, Joseph Larmarange, Francois Briatte,
##   Moritz Marbach, Edwin Thoen, Amos Elberg and Jason Crowley (2021).
##   GGally: Extension to 'ggplot2'. R package version 2.1.2.
##   https://CRAN.R-project.org/package=GGally
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {GGally: Extension to 'ggplot2'},
##     author = {Barret Schloerke and Di Cook and Joseph Larmarange and Francois B
riatte and Moritz Marbach and Edwin Thoen and Amos Elberg and Jason Crowley},
##     year = {2021},
##     note = {R package version 2.1.2},
##     url = {https://CRAN.R-project.org/package=GGally},
##   }
```

Part 1: Analysis

A] Pre-processing / Data Cleaning :-

```
# loading/importing the dataset of students studying math subject

maths_course_student <- data.frame(read.csv(file = "/Users/vivekbulani/Documents/U
CD Sem 1 Modules/UCD R Module/Final Project/student-mat.csv", header = TRUE))
```

```
# print the number of rows in the dataset

print(paste("Number of rows are" , nrow(maths_course_student)))
```

```
## [1] "Number of rows are 395"
```

```
# print the number of columns in the dataset

print(paste("Number of columns are" , ncol(maths_course_student)))
```

```
## [1] "Number of columns are 33"
```

```
# print the structure of the dataset. It tells the datatype of each column along w
ith some of its initial row values.

print("Structure of the Data set is")
```

```
## [1] "Structure of the Data set is"
```

```
cat("\n")
```

```
str(maths_course_student)
```

```
## 'data.frame':    395 obs. of  33 variables:
##  $ school    : chr  "GP" "GP" "GP" "GP" ...
##  $ sex       : chr  "F" "F" "F" "F" ...
##  $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address   : chr  "U" "U" "U" "U" ...
##  $ famsize   : chr  "GT3" "GT3" "LE3" "GT3" ...
##  $ Pstatus   : chr  "A" "T" "T" "T" ...
##  $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ Mjob      : chr  "at_home" "at_home" "at_home" "health" ...
##  $ Fjob      : chr  "teacher" "other" "other" "services" ...
##  $ reason    : chr  "course" "course" "other" "home" ...
##  $ guardian  : chr  "mother" "father" "mother" "mother" ...
##  $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup : chr  "yes" "no" "yes" "no" ...
##  $ famsup    : chr  "no" "yes" "no" "yes" ...
##  $ paid      : chr  "no" "no" "yes" "yes" ...
##  $ activities: chr  "no" "no" "no" "yes" ...
##  $ nursery   : chr  "yes" "no" "yes" "yes" ...
##  $ higher    : chr  "yes" "yes" "yes" "yes" ...
##  $ internet  : chr  "no" "yes" "yes" "yes" ...
##  $ romantic  : chr  "no" "no" "no" "yes" ...
##  $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ G1        : int  5 5 7 15 6 15 12 6 16 14 ...
##  $ G2        : int  6 5 8 14 10 15 12 5 18 15 ...
##  $ G3        : int  6 6 10 15 10 15 11 6 19 15 ...
```

```
# top 6 rows of the Data set are

head(maths_course_student)
```

```
##   school sex age address famsize Pstatus Medu Fedu     Mjob     Fjob     reason
## 1     GP   F  18       U     GT3       A    4    4  at_home  teacher     course
## 2     GP   F  17       U     GT3       T    1    1  at_home    other     course
## 3     GP   F  15       U     LE3       T    1    1  at_home    other      other
## 4     GP   F  15       U     GT3       T    4    2   health services       home
## 5     GP   F  16       U     GT3       T    3    3    other    other       home
## 6     GP   M  16       U     LE3       T    4    3 services    other reputation
##   guardian traveltime studytime failures schoolsup famsup paid activities
## 1   mother          2         2        0       yes     no   no         no
## 2   father          1         2        0        no    yes   no         no
## 3   mother          1         2        3       yes     no  yes         no
## 4   mother          1         3        0        no    yes  yes        yes
## 5   father          1         2        0        no    yes  yes         no
## 6   mother          1         2        0        no    yes  yes        yes
##   nursery higher internet romantic famrel freetime goout Dalc Walc health
## 1     yes    yes       no       no      4        3     4    1    1      3
## 2      no    yes      yes       no      5        3     3    1    1      3
## 3     yes    yes      yes       no      4        3     2    2    3      3
## 4     yes    yes      yes      yes      3        2     2    1    1      5
## 5     yes    yes       no       no      4        3     2    1    2      5
## 6     yes    yes      yes       no      5        4     2    1    2      5
##   absences G1 G2 G3
## 1        6  5  6  6
## 2        4  5  5  6
## 3       10  7  8 10
## 4        2 15 14 15
## 5        4  6 10 10
## 6       10 15 15 15
```

```
# print out all the column names present in the data

print("The different columns present in the Data set are")
```

```
## [1] "The different columns present in the Data set are"
```

```
cat("\n")
```

```
colnames(maths_course_student)
```

```
##  [1] "school"     "sex"        "age"        "address"    "famsize"
##  [6] "Pstatus"    "Medu"       "Fedu"       "Mjob"       "Fjob"
## [11] "reason"     "guardian"   "traveltime" "studytime"  "failures"
## [16] "schoolsup"  "famsup"     "paid"       "activities" "nursery"
## [21] "higher"     "internet"   "romantic"   "famrel"     "freetime"
## [26] "goout"      "Dalc"       "Walc"       "health"     "absences"
## [31] "G1"         "G2"         "G3"
```

As for some columns, the column names and values within columns are not proper and not giving proper information, lets rename some columns and also replace values within columns with more meaningful ones.

```
# replacing the short form of school names in the "school" column with their full
names

maths_course_student$school[maths_course_student$school == 'GP'] <- 'Gabriel Perei
ra'
maths_course_student$school[maths_course_student$school == 'MS'] <- 'Mousinho da S
ilveira'
```

```
# replacing the short form of gender in the "sex" column with their full names

maths_course_student$sex[maths_course_student$sex == 'M'] <- 'male'
maths_course_student$sex[maths_course_student$sex == 'F'] <- 'female'
```

```
# replacing the short form of area type in the "address" column with their full na
mes

maths_course_student$address[maths_course_student$address == 'U'] <- 'urban'
maths_course_student$address[maths_course_student$address == 'R'] <- 'rural'
```

```
# replacing the short form of family size description in the "famsize" column with
their full names

maths_course_student$famsize[maths_course_student$famsize == 'LE3'] <- 'less or eq
ual to 3'
maths_course_student$famsize[maths_course_student$famsize == 'GT3'] <- 'greater th
an 3'
```

```
# replacing the short form of status of parents in the "Pstatus" column with their
full names

maths_course_student$Pstatus[maths_course_student$Pstatus == 'T'] <- 'living toget
her'
maths_course_student$Pstatus[maths_course_student$Pstatus == 'A'] <- 'apart'
```

```
# converting "Medu" column to an ordered factor with the labels as mentioned in de
scription of dataset at the top.
# the order is "none < 4th grade < 5th to 9th grade < secondary education < higher
education"

maths_course_student$Medu = factor(maths_course_student$Medu, levels=c(0,1,2,3,4),
labels=c("none","4th grade","5th to 9th grade","secondary education","higher educa
tion"), ordered=TRUE)
```

```
# converting "Fedu" column to an ordered factor with the labels as mentioned in de
scription of dataset at the top.
# the order is "none < 4th grade < 5th to 9th grade < secondary education < higher
education"

maths_course_student$Fedu = factor(maths_course_student$Fedu, levels=c(0,1,2,3,4),
labels=c("none","4th grade","5th to 9th grade","secondary education","higher educa
tion"), ordered=TRUE)
```

```
# converting "traveltime" column to an ordered factor with the labels as mentioned
in description of dataset at the top.
# the order is "less than 15 min < 15 to 30 min < 30 min to 1 hour < more than 1 h
our"

maths_course_student$traveltime = factor(maths_course_student$traveltime, levels=c
(1,2,3,4), labels=c("less than 15 min","15 to 30 min","30 min to 1 hour","more tha
n 1 hour"), ordered=TRUE)
```

```
# converting "studytime" column to an ordered factor with the labels as mentioned
in description of dataset at the top.
# the order is "less than 2 hours < 2 to 5 hours < 5 to 10 hours < more than 10 ho
urs"

maths_course_student$studytime = factor(maths_course_student$studytime, levels=c(1
,2,3,4), labels=c("less than 2 hours","2 to 5 hours","5 to 10 hours","more than 10
hours"), ordered=TRUE)
```

```
# converting "famrel" column to an ordered factor with the labels as mentioned in
description of dataset at the top.
# the order is "very bad < bad < neutral < good < excellent"

maths_course_student$famrel = factor(maths_course_student$famrel, levels=c(1,2,3,4
,5), labels=c("very bad","bad","neutral","good","excellent"), ordered=TRUE)
```

```
# converting "freetime" column to an ordered factor with the labels as mentioned i
n description of dataset at the top.
# the order is "very low < low < medium < high < very high"

maths_course_student$freetime = factor(maths_course_student$freetime, levels=c(1,2
,3,4,5), labels=c("very low","low","medium","high","very high"), ordered=TRUE)
```

```
# converting "goout" column to an ordered factor with the labels as mentioned in d
escription of dataset at the top.
# the order is "very low < low < medium < high < very high"

maths_course_student$goout = factor(maths_course_student$goout, levels=c(1,2,3,4,5
), labels=c("very low","low","medium","high","very high"), ordered=TRUE)
```

```
# converting "Dalc" column to an ordered factor with the labels as mentioned in de
scription of dataset at the top.
# the order is "very low < low < medium < high < very high"

maths_course_student$Dalc = factor(maths_course_student$Dalc, levels=c(1,2,3,4,5),
labels=c("very low","low","medium","high","very high"), ordered=TRUE)
```

```
# converting "Walc" column to an ordered factor with the labels as mentioned in de
scription of dataset at the top.
# the order is "very low < low < medium < high < very high"

maths_course_student$Walc = factor(maths_course_student$Walc, levels=c(1,2,3,4,5),
labels=c("very low","low","medium","high","very high"), ordered=TRUE)
```

```
# converting "health" column to an ordered factor with the labels as mentioned in
description of dataset at the top.
# the order is "very bad < bad < neutral < good < very good"

maths_course_student$health = factor(maths_course_student$health, levels=c(1,2,3,4
,5), labels=c("very bad","bad","neutral","good","very good"), ordered=TRUE)
```

```
# renaming the short column names with their full names(description)

names(maths_course_student)[names(maths_course_student) == "G1"] <- "First Period
Grade"
names(maths_course_student)[names(maths_course_student) == "G2"] <- "Second Period
Grade"
names(maths_course_student)[names(maths_course_student) == "G3"] <- "Final Grade"
```

```
# print the structure of the dataset after converting all column names and values
to proper meaningful/structured names and datatypes.

str(maths_course_student)
```

```
## 'data.frame':    395 obs. of  33 variables:
##  $ school             : chr  "Gabriel Pereira" "Gabriel Pereira" "Gabriel Perei
ra" "Gabriel Pereira" ...
##  $ sex                : chr  "female" "female" "female" "female" ...
##  $ age                : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address            : chr  "urban" "urban" "urban" "urban" ...
##  $ famsize            : chr  "greater than 3" "greater than 3" "less or equal t
o 3" "greater than 3" ...
##  $ Pstatus            : chr  "apart" "living together" "living together" "livin
g together" ...
##  $ Medu               : Ord.factor w/ 5 levels "none"<"4th grade"<..: 5 2 2 5 4
5 3 5 4 4 ...
##  $ Fedu               : Ord.factor w/ 5 levels "none"<"4th grade"<..: 5 2 2 3 4
4 3 5 3 5 ...
##  $ Mjob               : chr  "at_home" "at_home" "at_home" "health" ...
##  $ Fjob               : chr  "teacher" "other" "other" "services" ...
##  $ reason             : chr  "course" "course" "other" "home" ...
##  $ guardian           : chr  "mother" "father" "mother" "mother" ...
##  $ traveltime         : Ord.factor w/ 4 levels "less than 15 min"<..: 2 1 1 1 1
1 1 2 1 1 ...
##  $ studytime          : Ord.factor w/ 4 levels "less than 2 hours"<..: 2 2 2 3
2 2 2 2 2 2 ...
##  $ failures           : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup          : chr  "yes" "no" "yes" "no" ...
##  $ famsup             : chr  "no" "yes" "no" "yes" ...
##  $ paid               : chr  "no" "no" "yes" "yes" ...
##  $ activities         : chr  "no" "no" "no" "yes" ...
##  $ nursery            : chr  "yes" "no" "yes" "yes" ...
##  $ higher             : chr  "yes" "yes" "yes" "yes" ...
##  $ internet           : chr  "no" "yes" "yes" "yes" ...
##  $ romantic           : chr  "no" "no" "no" "yes" ...
##  $ famrel             : Ord.factor w/ 5 levels "very bad"<"bad"<..: 4 5 4 3 4 5
4 4 4 5 ...
##  $ freetime           : Ord.factor w/ 5 levels "very low"<"low"<..: 3 3 3 2 3 4
4 1 2 5 ...
##  $ goout              : Ord.factor w/ 5 levels "very low"<"low"<..: 4 3 2 2 2 2
4 4 2 1 ...
##  $ Dalc               : Ord.factor w/ 5 levels "very low"<"low"<..: 1 1 2 1 1 1
1 1 1 1 ...
##  $ Walc               : Ord.factor w/ 5 levels "very low"<"low"<..: 1 1 3 1 2 2
1 1 1 1 ...
##  $ health             : Ord.factor w/ 5 levels "very bad"<"bad"<..: 3 3 3 5 5 5
3 1 1 5 ...
##  $ absences           : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ First Period Grade : int  5 5 7 15 6 15 12 6 16 14 ...
##  $ Second Period Grade: int  6 5 8 14 10 15 12 5 18 15 ...
##  $ Final Grade        : int  6 6 10 15 10 15 11 6 19 15 ...
```

```
# find if there are any missing/na values in the dataset

sum(is.na(maths_course_student))
```

```
## [1] 0
```

Hence there are no missing values in dataset. So no pre-processing related to handling of missing data is needed here.

```
# find if there are any duplicate rows in dataset

maths_course_student[duplicated(maths_course_student)==TRUE , ]
```

```
##  [1] school              sex                 age
##  [4] address             famsize             Pstatus
##  [7] Medu                Fedu                Mjob
## [10] Fjob                reason              guardian
## [13] traveltime          studytime           failures
## [16] schoolsup           famsup              paid
## [19] activities          nursery             higher
## [22] internet            romantic            famrel
## [25] freetime            goout               Dalc
## [28] Walc                health              absences
## [31] First Period Grade  Second Period Grade Final Grade
## <0 rows> (or 0-length row.names)
```

As there are 0 rows in the output, it means there are no duplicate rows present in the dataset. So no pre-processing related to removal of duplicate rows is needed here.

B] Analysing the cleaned and structured data :-

```
# display the descriptive statistics for (continuous, categorical, and dichotomous
) variables in a more meaningful way

tbl_summary(maths_course_student)
```

Characteristic	**N = 395**[1]
school	
Gabriel Pereira	349 (88%)
Mousinho da Silveira	46 (12%)
sex	

female	208 (53%)
male	187 (47%)
age	
15	82 (21%)
16	104 (26%)
17	98 (25%)
18	82 (21%)
19	24 (6.1%)
20	3 (0.8%)
21	1 (0.3%)
22	1 (0.3%)
address	
rural	88 (22%)
urban	307 (78%)
famsize	
greater than 3	281 (71%)
less or equal to 3	114 (29%)
Pstatus	
apart	41 (10%)
living together	354 (90%)
Medu	
none	3 (0.8%)
4th grade	59 (15%)

5th to 9th grade	103 (26%)
secondary education	99 (25%)
higher education	131 (33%)
Fedu	
none	2 (0.5%)
4th grade	82 (21%)
5th to 9th grade	115 (29%)
secondary education	100 (25%)
higher education	96 (24%)
Mjob	
at_home	59 (15%)
health	34 (8.6%)
other	141 (36%)
services	103 (26%)
teacher	58 (15%)
Fjob	
at_home	20 (5.1%)
health	18 (4.6%)
other	217 (55%)
services	111 (28%)
teacher	29 (7.3%)
reason	
course	145 (37%)

home	109 (28%)
other	36 (9.1%)
reputation	105 (27%)
guardian	
father	90 (23%)
mother	273 (69%)
other	32 (8.1%)
traveltime	
less than 15 min	257 (65%)
15 to 30 min	107 (27%)
30 min to 1 hour	23 (5.8%)
more than 1 hour	8 (2.0%)
studytime	
less than 2 hours	105 (27%)
2 to 5 hours	198 (50%)
5 to 10 hours	65 (16%)
more than 10 hours	27 (6.8%)
failures	
0	312 (79%)
1	50 (13%)
2	17 (4.3%)
3	16 (4.1%)
schoolsup	51 (13%)

famsup	242 (61%)
paid	181 (46%)
activities	201 (51%)
nursery	314 (79%)
higher	375 (95%)
internet	329 (83%)
romantic	132 (33%)
famrel	
very bad	8 (2.0%)
bad	18 (4.6%)
neutral	68 (17%)
good	195 (49%)
excellent	106 (27%)
freetime	
very low	19 (4.8%)
low	64 (16%)
medium	157 (40%)
high	115 (29%)
very high	40 (10%)
goout	
very low	23 (5.8%)
low	103 (26%)
medium	130 (33%)

high	86 (22%)
very high	53 (13%)
Dalc	
very low	276 (70%)
low	75 (19%)
medium	26 (6.6%)
high	9 (2.3%)
very high	9 (2.3%)
Walc	
very low	151 (38%)
low	85 (22%)
medium	80 (20%)
high	51 (13%)
very high	28 (7.1%)
health	
very bad	47 (12%)
bad	45 (11%)
neutral	91 (23%)
good	66 (17%)
very good	146 (37%)
absences	4 (0, 8)
First Period Grade	11 (8, 13)
Second Period Grade	11 (9, 13)

Final Grade	11 (8, 14)

[1] n (%); Median (IQR)

Using this output, now we can get the different values in each column and their proportion in respective columns of dataset. Some inferences are as follows :-

1. The most common age in the dataset is 15-18 years old (each having more than 20% proportion).
2. Most of the students involved in the survey are from Urban area (approximately 80%).
3. Most of the students have their parents living together (90% of cases).
4. Highest proportion of Mother Education Status is for “Higher Education” category (33%), whereas for Father Education Status, it is “5th to 9th Grade” category which has highest proportion (29%).
5. More than half of the students (approximately 70%) have guardian as their Mother.
6. Most of the students (approximately 80$) have never failed in their previous classes.
7. Most of the students (approximately 80$) have attended their nursery school.
8. 95% students want to pursue higher education.

These values can also be obtained by using prop.table() function as follows :-

```
# getting proportion values(fractional values) from the table for school variable

prop_school = prop.table(table(maths_course_student$school))

print(paste("The proportion of students who study at Gabriel Pereira are", round(p
rop_school['Gabriel Pereira'], 2)))
```

```
## [1] "The proportion of students who study at Gabriel Pereira are 0.88"
```

```
print(paste("The proportion of students who study at Mousinho da Silveira are", ro
und(prop_school['Mousinho da Silveira'], 2)))
```

```
## [1] "The proportion of students who study at Mousinho da Silveira are 0.12"
```

Hence there are very high number of students from “Gabriel Pereira” as compared to “Mousinho da Silveira”.

```
# getting proportion values(fractional values) from the table for Dalc variable wh
ich represents workday alcohol consumption

prop_Dalc = prop.table(table(maths_course_student$Dalc))

print(paste("The proportion of students who consume very low amount of alcohol on
workdays are", round(prop_Dalc['very low'], 2)))
```

```
## [1] "The proportion of students who consume very low amount of alcohol on workd
ays are 0.7"
```

```
print(paste("The proportion of students who consume low amount of alcohol on workd
ays are", round(prop_Dalc['low'], 2)))
```

```
## [1] "The proportion of students who consume low amount of alcohol on workdays a
re 0.19"
```

```
print(paste("The proportion of students who consume neutral/medium amount of alcoh
ol on workdays are", round(prop_Dalc['medium'], 2)))
```

```
## [1] "The proportion of students who consume neutral/medium amount of alcohol on
workdays are 0.07"
```

```
print(paste("The proportion of students who consume high amount of alcohol on work
days are", round(prop_Dalc['high'], 2)))
```

```
## [1] "The proportion of students who consume high amount of alcohol on workdays
are 0.02"
```

```
print(paste("The proportion of students who consume very high amount of alcohol on
workdays are", round(prop_Dalc['very high'],2)))
```

```
## [1] "The proportion of students who consume very high amount of alcohol on work
days are 0.02"
```

Hence most of the students consume very low alcohol on weekdays.

```
# getting proportion values(fractional values) from the table for Walc variable wh
ich represents weekend alcohol consumption

prop_Walc = prop.table(table(maths_course_student$Walc))

print(paste("The proportion of students who consume very low amount of alcohol on
weekends are", round(prop_Walc['very low'], 2)))
```

```
## [1] "The proportion of students who consume very low amount of alcohol on weeke
nds are 0.38"
```

```
print(paste("The proportion of students who consume low amount of alcohol on weeke
nds are", round(prop_Walc['low'], 2)))
```

```
## [1] "The proportion of students who consume low amount of alcohol on weekends a
re 0.22"
```

```
print(paste("The proportion of students who consume neutral/medium amount of alcoh
ol on weekends are", round(prop_Walc['medium'], 2)))
```

```
## [1] "The proportion of students who consume neutral/medium amount of alcohol on
weekends are 0.2"
```

```
print(paste("The proportion of students who consume high amount of alcohol on week
ends are", round(prop_Walc['high'], 2)))
```

```
## [1] "The proportion of students who consume high amount of alcohol on weekends
are 0.13"
```

```
print(paste("The proportion of students who consume very high amount of alcohol on
weekends are", round(prop_Walc['very high'],2)))
```

```
## [1] "The proportion of students who consume very high amount of alcohol on week
ends are 0.07"
```

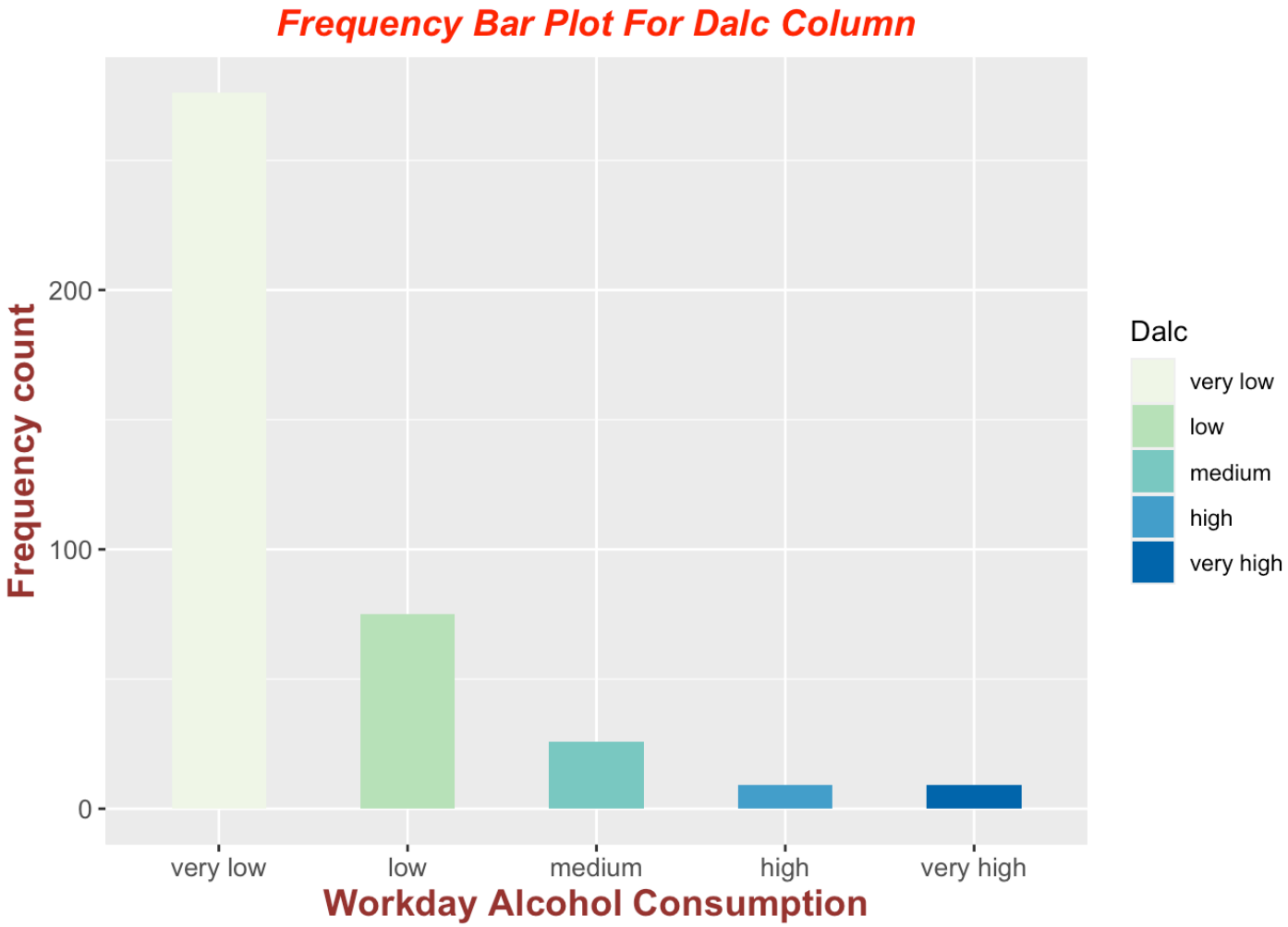
If we compare the alcohol consumption levels on weekdays and weekends, we see that at a broader level, students consume more amount of alcohol on weekends than weekdays (Because for weekends, the proportion of higher alcohol consumption levels is much more than the corresponding values for weekdays).

```
print(paste("The correlation between number of absences and Final Grades is",round
(cor(maths_course_student$absences, maths_course_student$`Final Grade`),3)))
```

```
## [1] "The correlation between number of absences and Final Grades is 0.034"
```

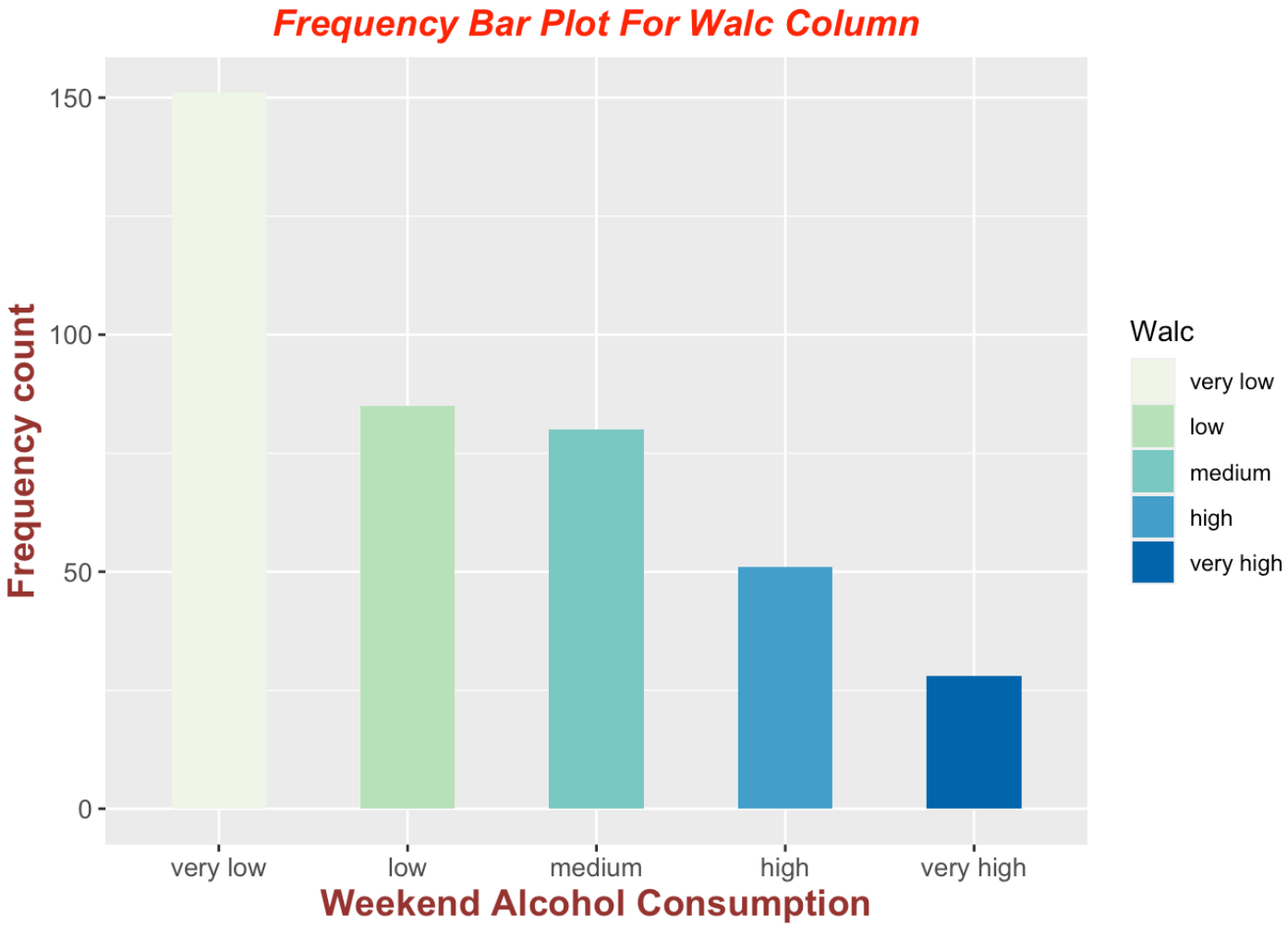
C] Graphical plots and summaries :-

```
# barplot for Dalc (workday alcohol consumption) column

ggplot(maths_course_student, aes(x = Dalc, fill = Dalc)) +
  geom_bar(width = 0.5) +
  xlab("Workday Alcohol Consumption") + ylab("Frequency count") + ggtitle("Frequen
cy Bar Plot For Dalc Column") +
  theme(
    plot.title = element_text(color="red", size=14, face="bold.italic", hjust = 0.
5),
    axis.text = element_text(size=10),
    axis.title.x = element_text(color="#993333", size=14, face="bold"),
    axis.title.y = element_text(color="#993333", size=14, face="bold")
    ) +
  scale_fill_brewer(palette="GnBu")
```



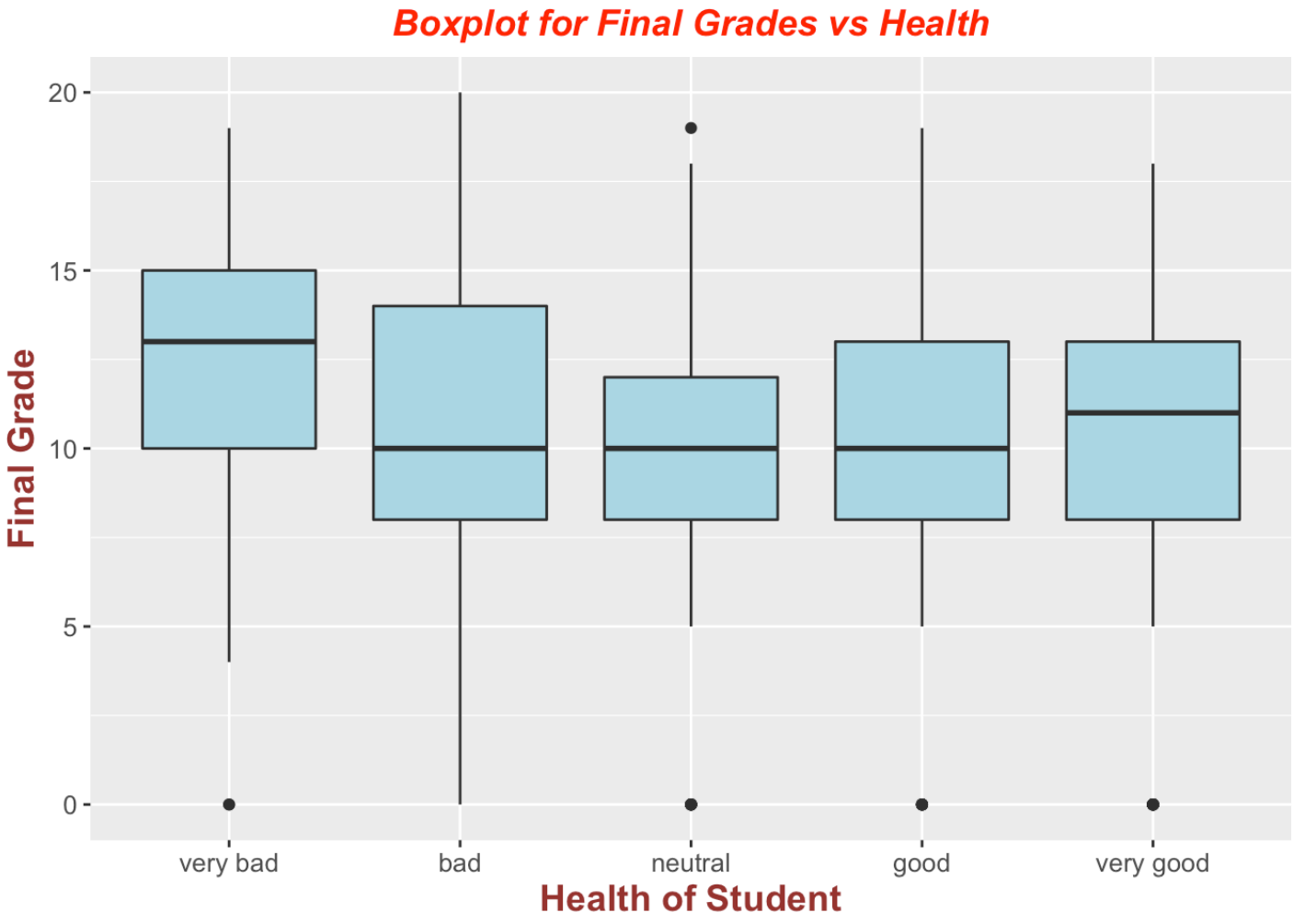
Thus we see that during Workdays, the alcohol consumption level is mostly very low (approximate count for very low category is 270 out of 395 records).

```
# barplot for Walc (weekend alcohol consumption) column

ggplot(maths_course_student, aes(x = Walc, fill = Walc)) +
  geom_bar(width = 0.5) +
  xlab("Weekend Alcohol Consumption") + ylab("Frequency count") + ggtitle("Frequen
cy Bar Plot For Walc Column") +
  theme(
    plot.title = element_text(color="red", size=14, face="bold.italic", hjust = 0.
5),
    axis.text = element_text(size=10),
    axis.title.x = element_text(color="#993333", size=14, face="bold"),
    axis.title.y = element_text(color="#993333", size=14, face="bold")
    ) +
  scale_fill_brewer(palette="GnBu")
```



Thus we see that during Weekends, the alcohol consumption levels are more as compared to weekdays, because the count for low, medium and high amount of alcohol consumption is more on weekends than on weekdays.

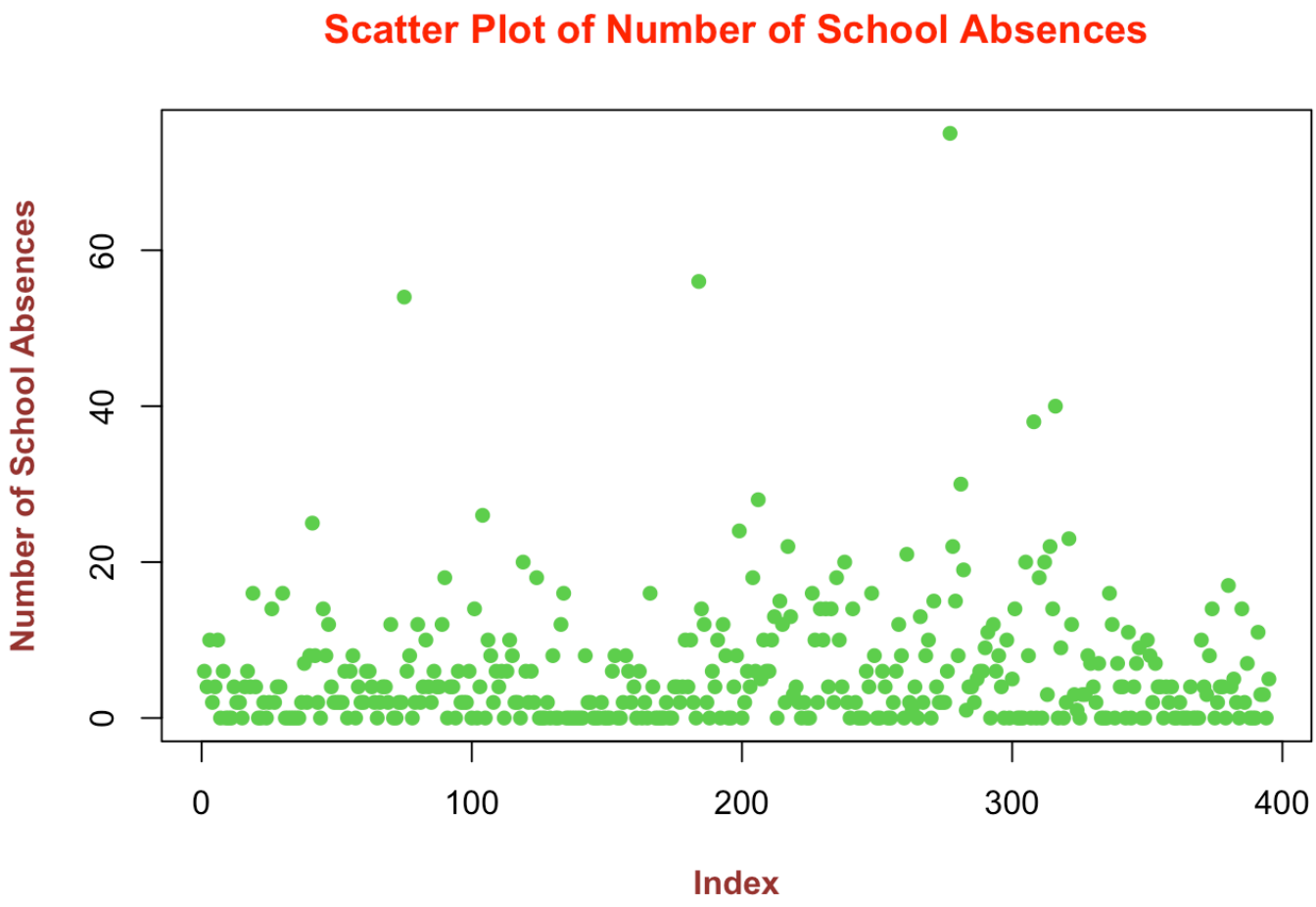
```
# boxplot for Health column
# x-axis is different categories of health and y-axis is distribution of Final Gra
de for each category of health

ggplot(maths_course_student, aes(x = health, y = `Final Grade`)) +
   geom_boxplot(fill = 'lightblue') +
  xlab("Health of Student") + ylab("Final Grade") + ggtitle("Boxplot for Final Gra
des vs Health") +
  theme(
    plot.title = element_text(color="red", size=14, face="bold.italic", hjust = 0.
5),
    axis.text = element_text(size=10),
    axis.title.x = element_text(color="#993333", size=14, face="bold"),
    axis.title.y = element_text(color="#993333", size=14, face="bold")
    )
```



1. Here we see that when the health of a student is very bad, then its Final Grades are, on an average, higher to those students whose health is better. But this seems to be strange and not logical. Maybe there might be some mistake while data collection / data entry. This aspect needs to be further investigated and verified and appropriate reasoning for such behavior needs to be found out if this data is correct.

2. There is high variation in Final Grades for those students whose health is “bad” and very low variation in Final Grades for those whose health is neutral.
3. Also there are a very few outliers almost in all categories.
4. Apart from this, we can infer that except for neutral condition of health, for all other categories of health, the final grades have a skewed distribution.

```
# scatter plot of Number of Absences

plot(maths_course_student$absences , col = 3 , pch = 16, ylab = "Number of School
Absences", main = "Scatter Plot of Number of School Absences", col.main = "red", c
ol.lab = "#993333", font.lab = 2)
```



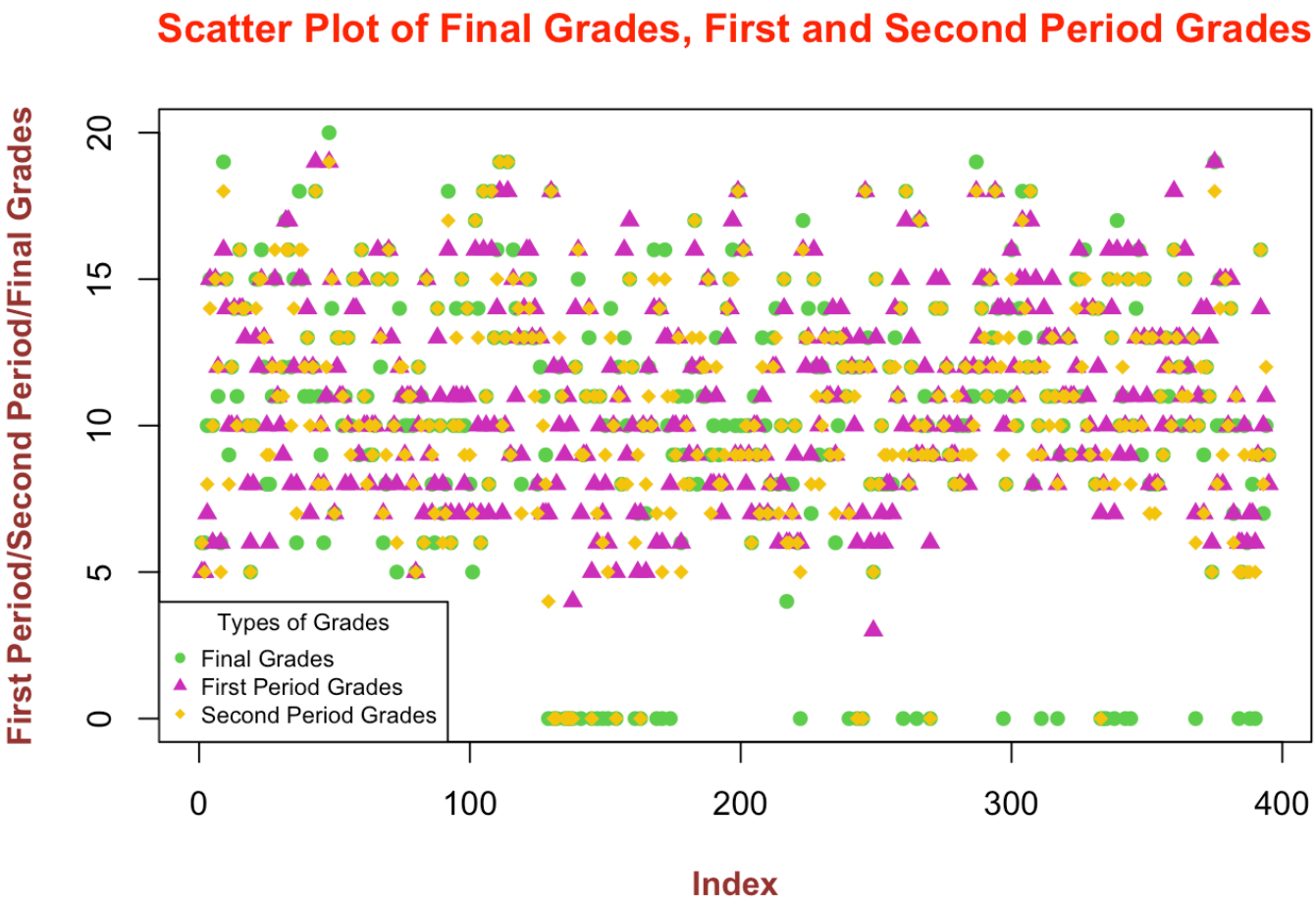
From above scatter plot, we can say that most of the students have 0-10 absences in school. But there are also a few outliers which show that 3 students have more that 50 absences. Hence it needs to be rechecked if this data is correct or not. If it is correct then we need to check the Final Grades of such students as follows :-

```
# finding out the final grades of students who have absences exceptionally large (
>50)

print(paste("The Final grades of students who have absences greater than 50 are",m
aths_course_student$`Final Grade`[maths_course_student$absences > 50]))
```

```
## [1] "The Final grades of students who have absences greater than 50 are 11"
## [2] "The Final grades of students who have absences greater than 50 are 8"
## [3] "The Final grades of students who have absences greater than 50 are 9"
```

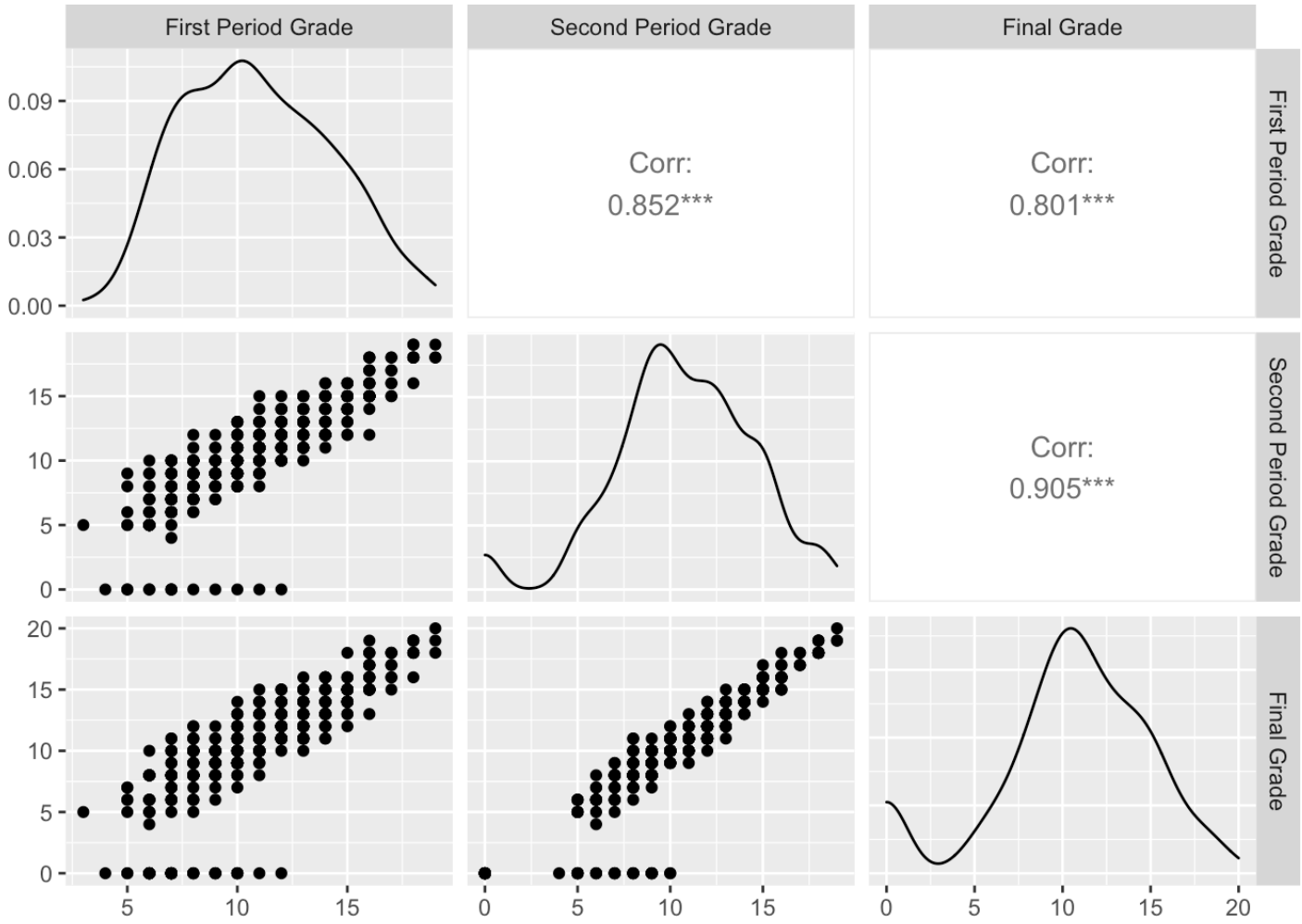
These grades are out of 20. Hence all these 3 students are still pass even after being absent for so many days.

```
# scatter plots of First and Second Period Grades and Final Grades


plot(maths_course_student$`Final Grade` , col = 3 , pch = 16, ylab = "First Period
/Second Period/Final Grades", main = "Scatter Plot of Final Grades, First and Seco
nd Period Grades", col.main = "red", col.lab = "#993333", font.lab = 2)

points(maths_course_student$`First Period Grade` , col = 6, pch = 17)
points(maths_course_student$`Second Period Grade` , col = 7 , pch = 18)

legend("bottomleft", c("Final Grades", "First Period Grades", "Second Period Grade
s"), cex=0.7, col=c(3,6,7), pch=c(16,17,18), title = "Types of Grades")
```



From above scatter plot, we can infer that for all students, First Period Grades are never 0 (i.e First Period Grades > 0 for all students), whereas Second Period and Final Grades are 0 for few students.

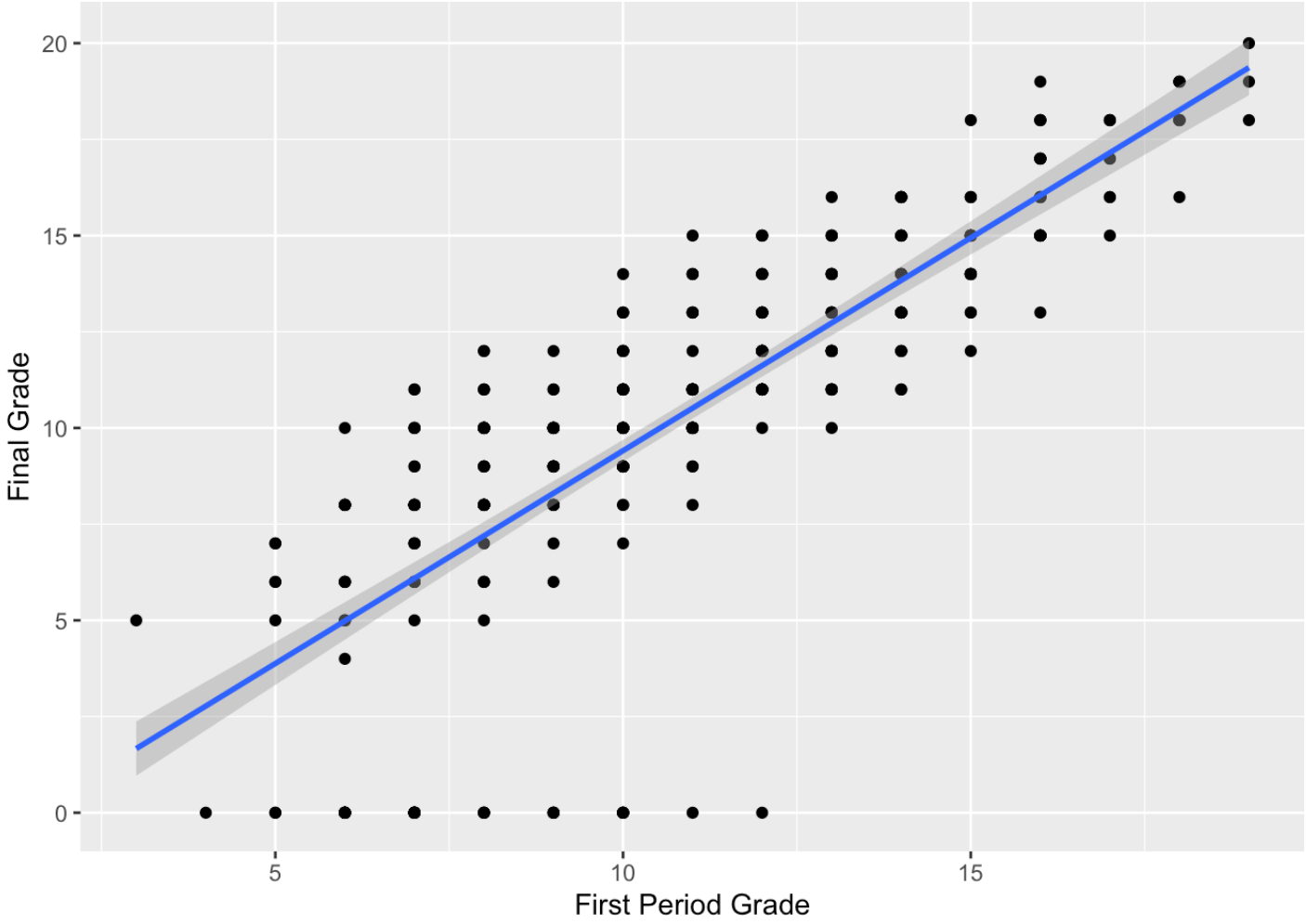
```
# pair-wise scatter plot for first period grades, second period grades and final g
rades

ggpairs(maths_course_student[31:33])
```



From above plot, we can infer that Final Grades has a high positive correlation with both First and Second Period Grades individually. Also this relationship appears to be linear (i.e as First / Second Period Grades increases, Final Grades increases for most of the students). This relationship can also be shown as follows :-

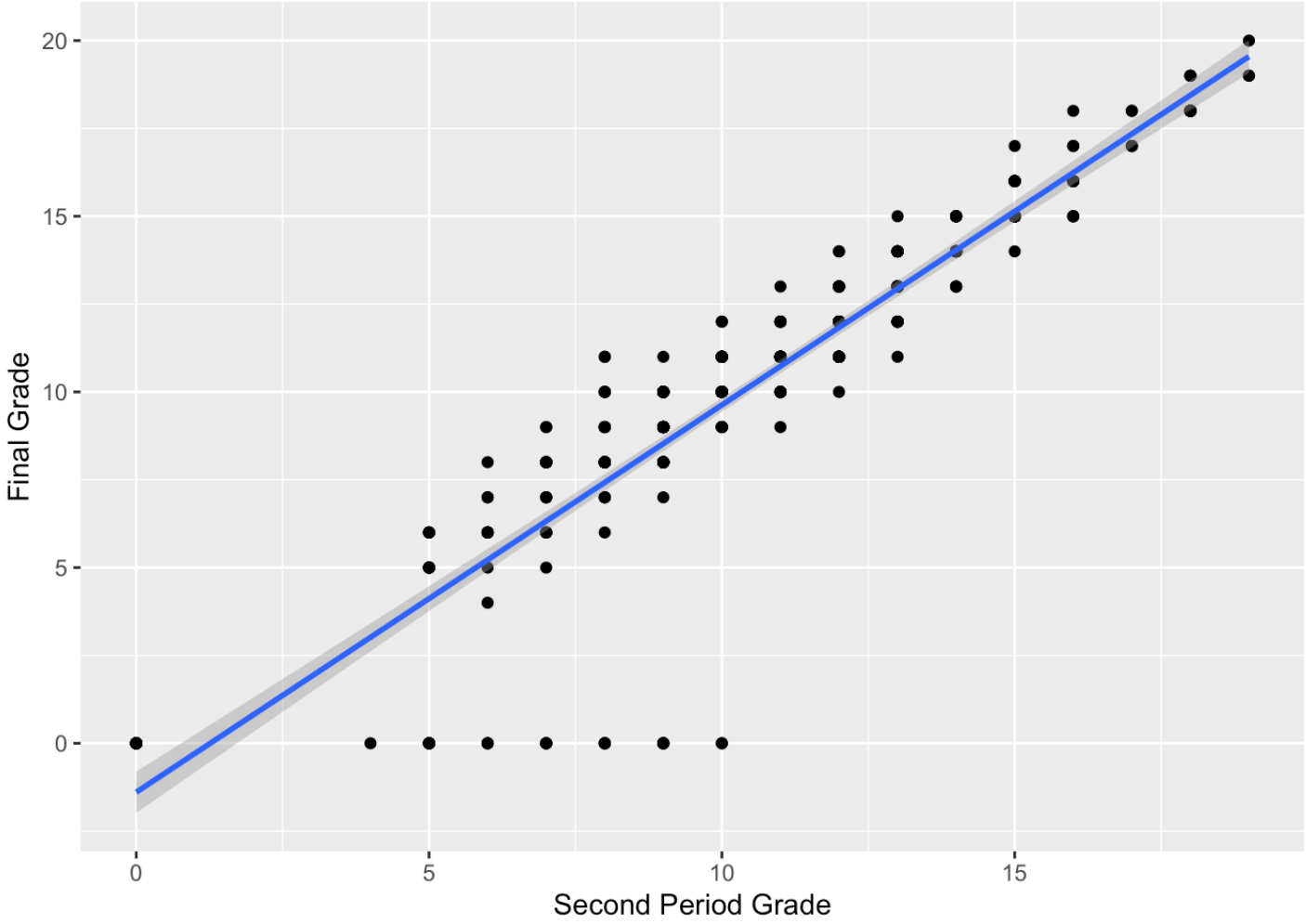
```
# showing and plotting the linear relationship between First Period Grades and Fin
al Grades

ggplot(maths_course_student, aes(x=`First Period Grade` , y=`Final Grade`)) +
  geom_point()+
  geom_smooth(method=lm)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# showing and plotting the linear relationship between Second Period Grades and Fi
nal Grades

ggplot(maths_course_student, aes(x=`Second Period Grade` , y=`Final Grade`)) +
  geom_point()+
  geom_smooth(method=lm)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Part 2: R Package

For this task, the selected package of interest is “caret” (short form for Classification And REgression Training). Though R has many inbuilt methods for different machine learning algorithms, but it is hard/challenging to remember in exactly which package does each method resides. Also different methods may have very different syntax, thus leading to confusion. Such problems further create difficulties in complex work such as hyper-parameter tuning.

All such problems have been solved by the introduction of caret package. The user just calls the functions of caret package, which have almost similar structure/syntax (hence called consistent modeling syntax). The caret internally calls the traditional inbuilt methods of respective machine learning algorithms. Hence with the help of caret, users/developers can now focus on more challenging/complex parts of model building.

```
# import the caret package

library(caret)
```

```
## Loading required package: lattice
```

```
citation("caret")
```

```
##
## To cite package 'caret' in publications use:
##
##   Max Kuhn (2021). caret: Classification and Regression Training. R
##   package version 6.0-90. https://CRAN.R-project.org/package=caret
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {caret: Classification and Regression Training},
##     author = {Max Kuhn},
##     year = {2021},
##     note = {R package version 6.0-90},
##     url = {https://CRAN.R-project.org/package=caret},
##   }
```

Now one of the methods of caret package is to partition/split the dataset into training and testing data. This is very important to ensure the model doesn't overfit the data and that it predicts well for unseen/testing data.

To partition the data, we have createDataPartition() method which takes the response variable as its first parameter, the proportion of split and whether you want partitioned data in the form of list or not.

```
# define/partition the data into training and testing data. Partition into 80:20 r
atio.

partitioned_data <- createDataPartition(maths_course_student$`Final Grade`, p=0.8,
list=FALSE)
training_data <- maths_course_student[partitioned_data, ]
testing_data <- maths_course_student[-partitioned_data, ]
```

Another major benefit of caret is that for training any machine learning model, we have the same train() method. Just change the value of parameter "method" to specify which algorithm you need to use. This train() function returns a "train" object which has many different attributes as shown below.

A] Linear Regression using Caret

```
# train a linear regression model (to predict Final Grades from few predictor vari
ables/columns which seem to be relevant and important) by specifying method="lm".
Internally caret calls the traditional lm() method defined in stats package

caret_linear_regression <- train(`Final Grade` ~ `school` + `sex` +  `age` +  `tra
veltime` +  `studytime` +  `failures` +  `activities` +  `freetime` + `absences` +
`First Period Grade` + `Second Period Grade`, data = training_data, method = "lm")
```

```
# check the class of linear regression model built using caret package

class(caret_linear_regression)
```

```
## [1] "train"         "train.formula"
```

```
# find out all the attributes available for train object

attributes(caret_linear_regression)
```

```
## $names
##  [1] "method"       "modelInfo"    "modelType"    "results"      "pred"
##  [6] "bestTune"     "call"         "dots"         "metric"       "control"
## [11] "finalModel"   "preProcess"   "trainingData" "ptype"        "resample"
## [16] "resampledCM"  "perfNames"    "maximize"     "yLimits"      "times"
## [21] "levels"       "terms"        "coefnames"    "contrasts"    "xlevels"
##
## $class
## [1] "train"         "train.formula"
```

```
# check out the results obtained from linear regression model

caret_linear_regression$results
```

```
##   intercept     RMSE Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1      TRUE 1.965381 0.811214 1.305273 0.1950256 0.03492323 0.1253872
```

“finalModel” attribute prints out the details/summary of the final one model which is the best. As linear regression has only 1 model in the result, the same model is considered as the final model.

```
caret_linear_regression$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
##                   (Intercept)  `schoolMousinho da Silveira`
##                       1.25863                       0.31799
##                       sexmale                           age
##                       0.07245                      -0.15347
##                  traveltime.L                 traveltime.Q
##                       0.83345                       0.69552
##                  traveltime.C                  studytime.L
##                       0.18725                      -0.22567
##                   studytime.Q                  studytime.C
##                      -0.30937                      -0.24124
##                      failures                 activitiesyes
##                      -0.24360                      -0.29089
##                    freetime.L                    freetime.Q
##                       0.59628                      -0.04582
##                    freetime.C                   `freetime^4`
##                       0.09465                      -0.16794
##                      absences    `\\`First Period Grade\\``
##                       0.03872                       0.14580
##  `\\`Second Period Grade\\``
##                       0.96592
```

So from above output, we can find the estimated values of all coefficients (intercept and slope parameters).

If we cross check the values of coefficients derived using caret package with those obtained in Part 3 (below), we see that coefficients are similar (close to each other) in both cases.

B] Resampling using Caret

```
# extract the R-squared value for the model

print(paste("The R-squared value for Linear Regression using Caret is", summary(ca
ret_linear_regression$finalModel)$r.squared))
```

```
## [1] "The R-squared value for Linear Regression using Caret is 0.852171416295222
"
```

But this R-squared value is when modelling is done on entire dataset. This may give false interpretation because this value is not the true measure of how well the model will perform for new unseen data. The real measure of the quality of model is obtained by the R-squared value of model based on resampling/splitting of data. To extract this R-squared value, use following code :-

```
caret_linear_regression
```

```
## Linear Regression
##
## 318 samples
##  11 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 318, 318, 318, 318, 318, 318, ...
## Resampling results:
##
##   RMSE      Rsquared  MAE
##   1.965381  0.811214  1.305273
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Here we see that R-squared value is approximately 82%. This is the true measure of quality of model because here we see that resampling (Bootstrap resampling) has been performed consisting of 25 repetitions/iterations. This means caret builds 25 models and calculates R-squared value for each model. At last it again runs the model having highest R-squared value and stores this model as Final Model (which can then be displayed using finalModel attribute as seen above.)

If we want to use Cross-validation instead of Bootstrap, then we can do as follows :-

```
# setting up a 10-fold cross validation

ten_fold_cv <- trainControl(method = "cv", number = 10)


# train the linear regression model using 10-fold cross validation

caret_linear_regression_cv <- train(`Final Grade` ~ `school` + `sex` +  `age` +  `
traveltime` +  `studytime` +  `failures` +  `activities` +  `freetime` + `absences
` + `First Period Grade` + `Second Period Grade`, data=training_data, method = "lm
", trControl = ten_fold_cv)
```

```
# printing out the trained model. Here we can also see for R-squared value (which
is based on 10-fold cross validation)

caret_linear_regression_cv
```

```
## Linear Regression
##
## 318 samples
##  11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 287, 287, 286, 286, 285, 286, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   1.80703  0.8523083  1.207455
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

10-fold cross validation means splitting the data into 10 approximately equal chunks. Then develop a model based on 9 chunks and predict the 10th. Perform this multiple times each time choosing a different set of 9 chunks and predict the 10th.

We can see that we get higher R-squared value and lower RMSE value when we use 10-fold cross validation instead of Bootstrap.

C] Random Forest using Caret

As said before, the same train() method can be used to build different machine learning models, just update the “method” parameter. So for random forest, we pass “rf” to the “method” attribute to specify that we need to build a random forest.

```
caret_random_forest <- train(`Final Grade` ~ `school` + `sex` +  `age` +  `travelt
ime` +  `studytime` +  `failures` +  `activities` +  `freetime` + `absences` + `Fi
rst Period Grade` + `Second Period Grade`, data = training_data, method = "rf")
```

```
# check out the results obtained from random forest model

caret_random_forest$results
```

```
##   mtry     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1    2 2.573540 0.7544259 1.770896 0.2178175 0.04504266 0.1417931
## 2   10 1.642647 0.8686556 1.067567 0.2097950 0.03637609 0.1177606
## 3   18 1.640668 0.8660383 1.045557 0.2223258 0.04165364 0.1192317
```

The idea behind random forest is that it creates multiple decision trees and then averages them to give the final prediction. In this case the tuning parameter is the number of randomly selected predictors used to make the trees different (i.e to help create uncorrelated trees). This parameter is called “mtry” in caret.

Caret automatically performs auto-tuning of “mtry” parameter and builds a model for each of this parameter value using resampling (Bootstrap resampling by default).

Hence we get 3 different model summaries contrary to linear regression where we were getting only 1 model as output.

If we want to specify the values of mtry that the model should consider, then we can do this using tuneGrid parameter as follows :-

```
# create a sequence of values to be considered for mtry parameter
mtry_value_list <- data.frame(mtry = seq(2, 20, by= 4))


caret_random_forest_1 <- train(`Final Grade` ~ `school` + `sex` +  `age` +  `trave
ltime` +  `studytime` +  `failures` +  `activities` +  `freetime` + `absences` + `
First Period Grade` + `Second Period Grade`, data = training_data, method = "rf",
tuneGrid = mtry_value_list)
```

```
# get the summary of each random forest model built using specified values of mtry

caret_random_forest_1$results
```

```
##   mtry     RMSE  Rsquared      MAE    RMSESD RsquaredSD      MAESD
## 1    2 2.517372 0.7673597 1.758766 0.2098018 0.02726735 0.15189256
## 2    6 1.771722 0.8537737 1.179731 0.1821328 0.03000316 0.08910201
## 3   10 1.669048 0.8641492 1.071216 0.2094908 0.03406264 0.08963896
## 4   14 1.653645 0.8649193 1.038491 0.2362422 0.03831885 0.09476641
## 5   18 1.674392 0.8609364 1.045057 0.2648276 0.04363044 0.10879923
```

D] Comparison of different models using Caret

Another useful tool of caret is that we can easily compare different models built using caret and hence decide which model/algorithm works best for the specified data.

```
model_list <- list(lm = caret_linear_regression, rf = caret_random_forest)
results_of_2_models <- resamples(model_list)
summary(results_of_2_models)
```

```
##
## Call:
## summary.resamples(object = results_of_2_models)
##
## Models: lm, rf
## Number of resamples: 25
##
## MAE
##         Min.   1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm 1.0739956 1.2313824 1.314524 1.305273 1.350987 1.658989    0
## rf 0.8140707 0.9706688 1.097002 1.045557 1.115067 1.220807    0
##
## RMSE
##        Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm 1.649883 1.848273 1.962864 1.965381 2.105942 2.434079    0
## rf 1.190067 1.498408 1.668540 1.640668 1.805142 1.970313    0
##
## Rsquared
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm 0.7296386 0.7930669 0.8102897 0.8112140 0.8329927 0.8748639    0
## rf 0.7560730 0.8404754 0.8785606 0.8660383 0.8885084 0.9353400    0
```

Hence we clearly see that Random Forest has much higher R-squared value and much lower RMSE (Root Mean Squared Error) value as compared to that of Linear Regression. Thus Random Forest is much better than Linear Regression for our dataset.

Part 3: Functions/Programming

```
# define a function that creates the list of standardized residuals from the linea
r model object passed as parameter. It also assigns the object (storing standardiz
ed residuals) the appropriate class

get_standardised_residuals <- function(my_mod){              # my_mod is the line
ar model object
  my_model_residuals <- rstandard(my_mod)                    # find standardized
residuals
  class(my_model_residuals) <- "my_resid"                    # assign the class
  return(my_model_residuals)
}
```

```
# train a linear regression model with predictor/independent variables as few colu
mns of dataset which seem to be important and relevant to the Final Grades. The re
sponse/dependent variable is "Final Grades"

my_model <- lm(`Final Grade` ~ `school` + `sex` +  `age` +  `traveltime` +  `study
time` +  `failures` +  `activities` +  `freetime` + `absences` + `First Period Gra
de` + `Second Period Grade`, data = maths_course_student)
```

```
# print out the summary of linear model trained

summary(my_model)
```

```
##
## Call:
## lm(formula = `Final Grade` ~ school + sex + age + traveltime +
##     studytime + failures + activities + freetime + absences +
##     `First Period Grade` + `Second Period Grade`, data = maths_course_student)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7632 -0.4404  0.2812  0.9603  3.8786
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2.00312    1.48345   1.350 0.177730
## schoolMousinho da Silveira  0.30630    0.34400   0.890 0.373822
## sexmale                     0.20016    0.21527   0.930 0.353086
## age                        -0.21012    0.08735  -2.405 0.016636 *
## traveltime.L                0.94580    0.47684   1.983 0.048042 *
## traveltime.Q                0.72700    0.41807   1.739 0.082859 .
## traveltime.C                0.22968    0.33720   0.681 0.496202
## studytime.L                -0.35661    0.30129  -1.184 0.237316
## studytime.Q                -0.35782    0.25725  -1.391 0.165071
## studytime.C                -0.27747    0.20783  -1.335 0.182660
## failures                   -0.26644    0.14601  -1.825 0.068820 .
## activitiesyes              -0.34969    0.19700  -1.775 0.076688 .
## freetime.L                  0.32537    0.36670   0.887 0.375490
## freetime.Q                 -0.06667    0.31178  -0.214 0.830799
## freetime.C                  0.01975    0.25951   0.076 0.939368
## freetime^4                 -0.20137    0.19295  -1.044 0.297303
## absences                    0.04582    0.01261   3.635 0.000317 ***
## `First Period Grade`        0.14555    0.05781   2.518 0.012229 *
## `Second Period Grade`       0.97278    0.05022  19.370  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.896 on 376 degrees of freedom
## Multiple R-squared:  0.8365, Adjusted R-squared:  0.8287
## F-statistic: 106.9 on 18 and 376 DF,  p-value: < 2.2e-16
```

From this output we see that age, traveltime, absences, First Period Grades and Second Period Grades are significant (because their p-values are less than 0.05)

```
# call the function created above and passing our trained linear model object as p
arameter

obj <- get_standardised_residuals(my_model)
```

```
# check the class of obj (object storing standardized residuals)

class(obj)
```

```
## [1] "my_resid"
```

```
# check the available/defined methods for class "my_resid"

methods(class="my_resid")
```

```
## no methods found
```

As we haven't declared any methods for this class as of now, hence output shown is "no methods found"

```
# define a "summary" method for class "my_resid". It returns values of few statist
ical parameters (such as mean, median, minimum, maximum) of the standardized resid
uals.


summary.my_resid <- function(object){
  cat("The mean of standardised residuals is", mean(object))
  cat("\nThe median of standardised residuals is", median(object))
  cat("\nThe minimum of standardised residuals is", min(object))
  cat("\nThe maximum of standardised residuals is", max(object))
}
```

```
# call the above defined summary object

summary(obj)
```

```
## The mean of standardised residuals is -0.000598314
## The median of standardised residuals is 0.1549142
## The minimum of standardised residuals is -4.698289
## The maximum of standardised residuals is 2.114425
```

```
# define the "print" method for class "my_resid". It sorts the residuals in decrea
sing order and prints the top 10 and bottom 10 residuals. This maybe useful to fin
d out extreme values of residuals (worst prediction cases)


print.my_resid <- function(object){
  object = sort(object, decreasing = TRUE)
  print(object[1:10])                 # print top 10 residuals (highest positive v
alue of residuals)
  print(object[386:395])              # print bottom 10 residuals (highest negativ
e value of residuals)
}
```

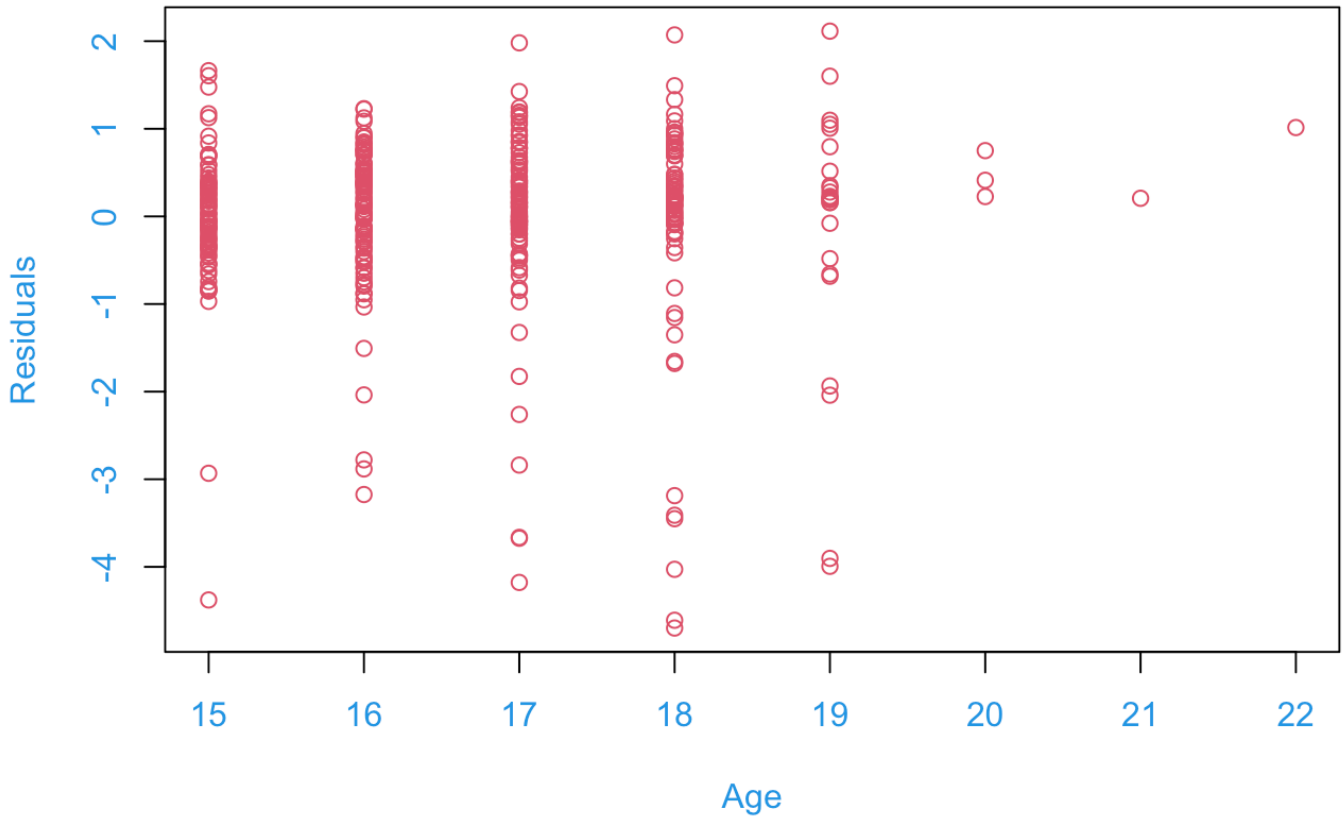
```
# call the above defined print object

print(obj)
```

```
##      371      158       79        3       44      128      218       34
## 2.114425 2.071394 1.980920 1.663381 1.605467 1.600163 1.491631 1.475890
##      189      339
## 1.425190 1.332575
##       334       338       344       311       297       335       260       141
## -3.450301 -3.663331 -3.673256 -3.903247 -3.993716 -4.028010 -4.177926 -4.377613
##       342       265
## -4.609391 -4.698289
```

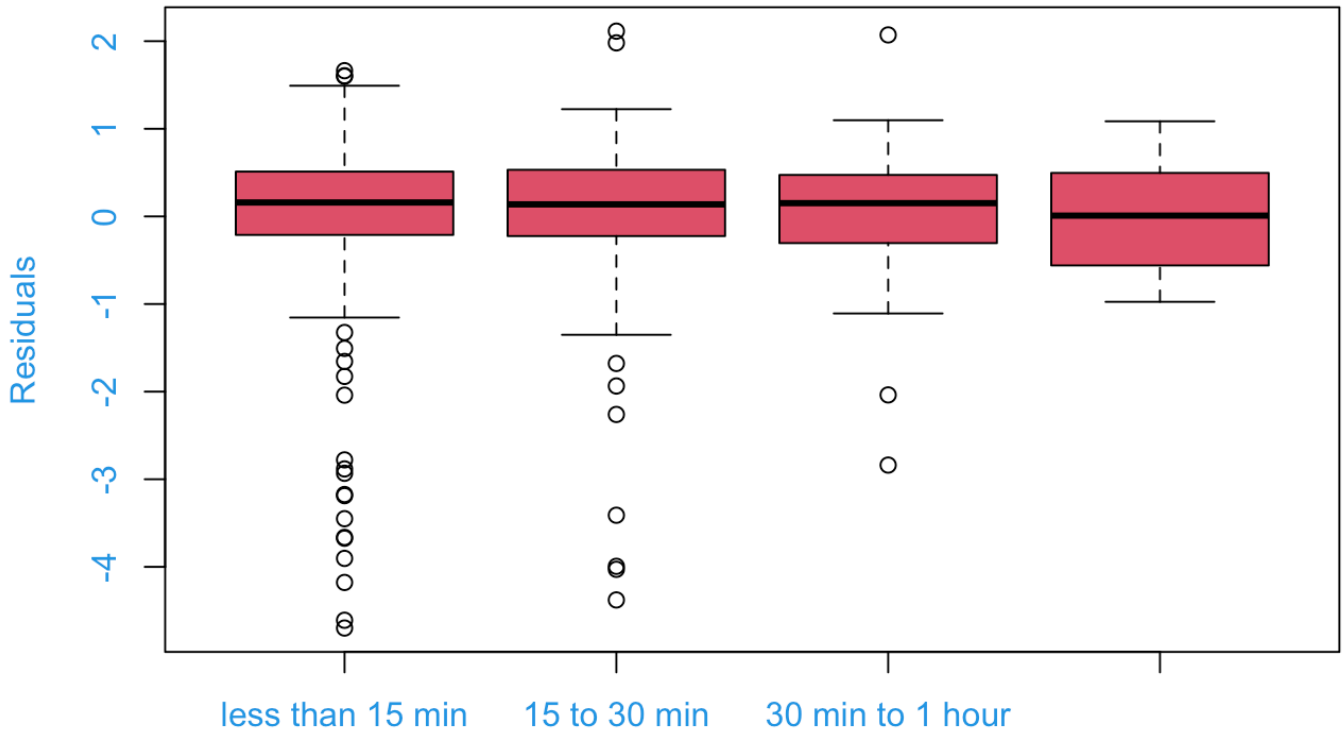
```
# define the "plot" method for class "my_resid". It plots the residuals with respe
ct to some predictor variables (the ones for which p-values are less than 0.05 i.e
the ones which are significant) and with respect to response variable (Final Grade
). Such plots are useful for determining the usefulness of trained linear models a
nd testing the assumptions of zero conditional mean, homoscedasticity for the line
ar model.


plot.my_resid <- function(object){

  plot(maths_course_student$age, object, xlab = "Age", ylab = "Residuals", main =
"Plot of Residuals vs Age Predictor", col = 2, col.main = 3, col.lab = 4, col.axis
= 4)

  plot(maths_course_student$traveltime, object, xlab = "Travel Time", ylab = "Resi
duals", main = "Plot of Residuals vs Travel Time Predictor", col = 2, col.main = 3
, col.lab = 4, col.axis = 4)

  plot(maths_course_student$absences, object, xlab = "Absences", ylab = "Residuals
", main = "Plot of Residuals vs Absences Predictor", col = 2, col.main = 3, col.la
b = 4, col.axis = 4)

  plot(maths_course_student$`First Period Grade`, object, xlab = "First Period Gra
de", ylab = "Residuals", main = "Plot of Residuals vs First Period Grade Predictor
", col = 2, col.main = 3, col.lab = 4, col.axis = 4)

  plot(maths_course_student$`Second Period Grade`, object, xlab = "Second Period G
rade", ylab = "Residuals", main = "Plot of Residuals vs Second Period Grade Predic
tor", col = 2, col.main = 3, col.lab = 4, col.axis = 4)

  plot(maths_course_student$`Final Grade`, object, xlab = "Final Grade", ylab = "R
esiduals", main = "Plot of Residuals vs Final Grade Response", col = 2, col.main =
3, col.lab = 4, col.axis = 4)
}
```

```
# call the above defined plot object

plot(obj)
```

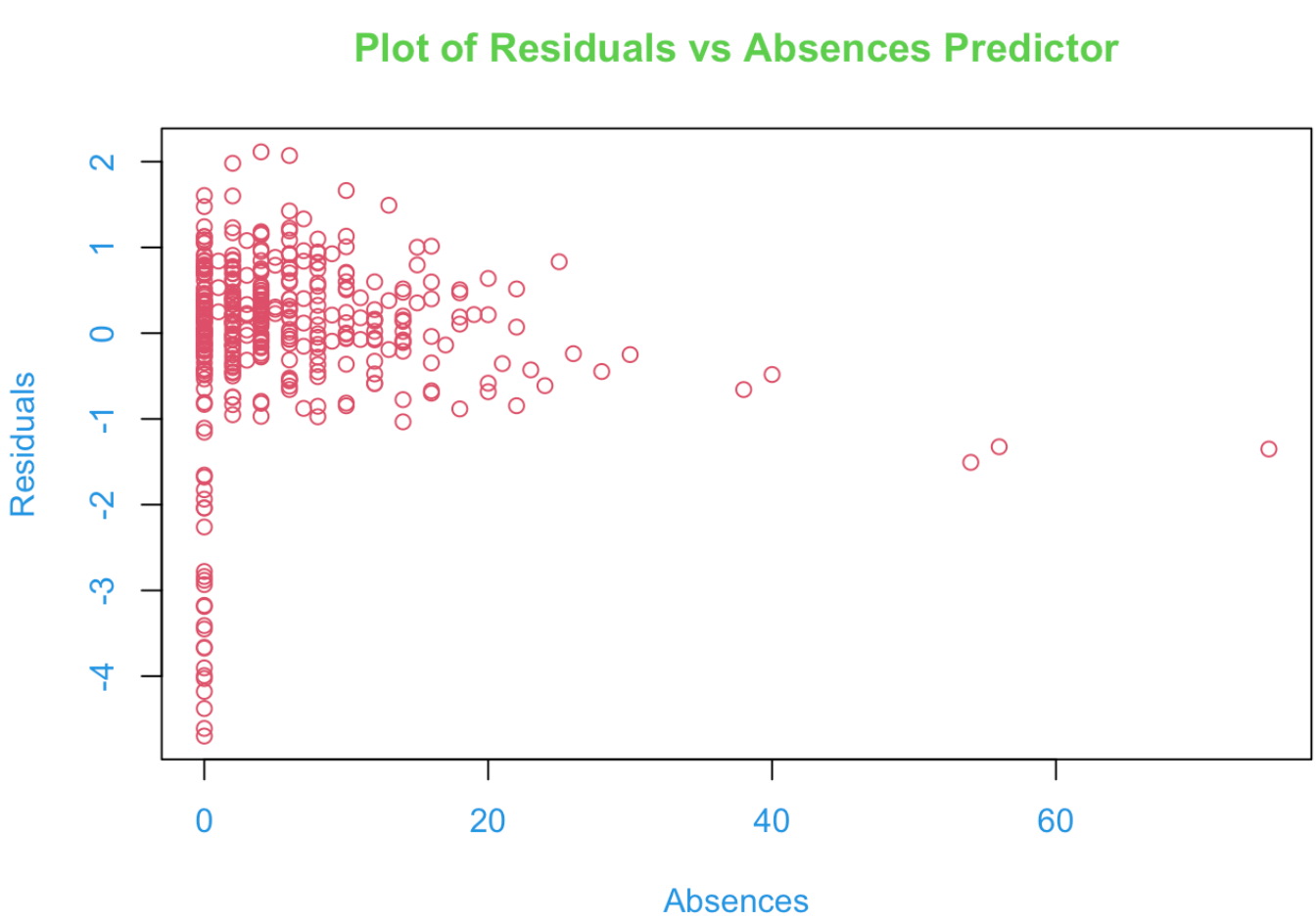
Plot of Residuals vs Age Predictor

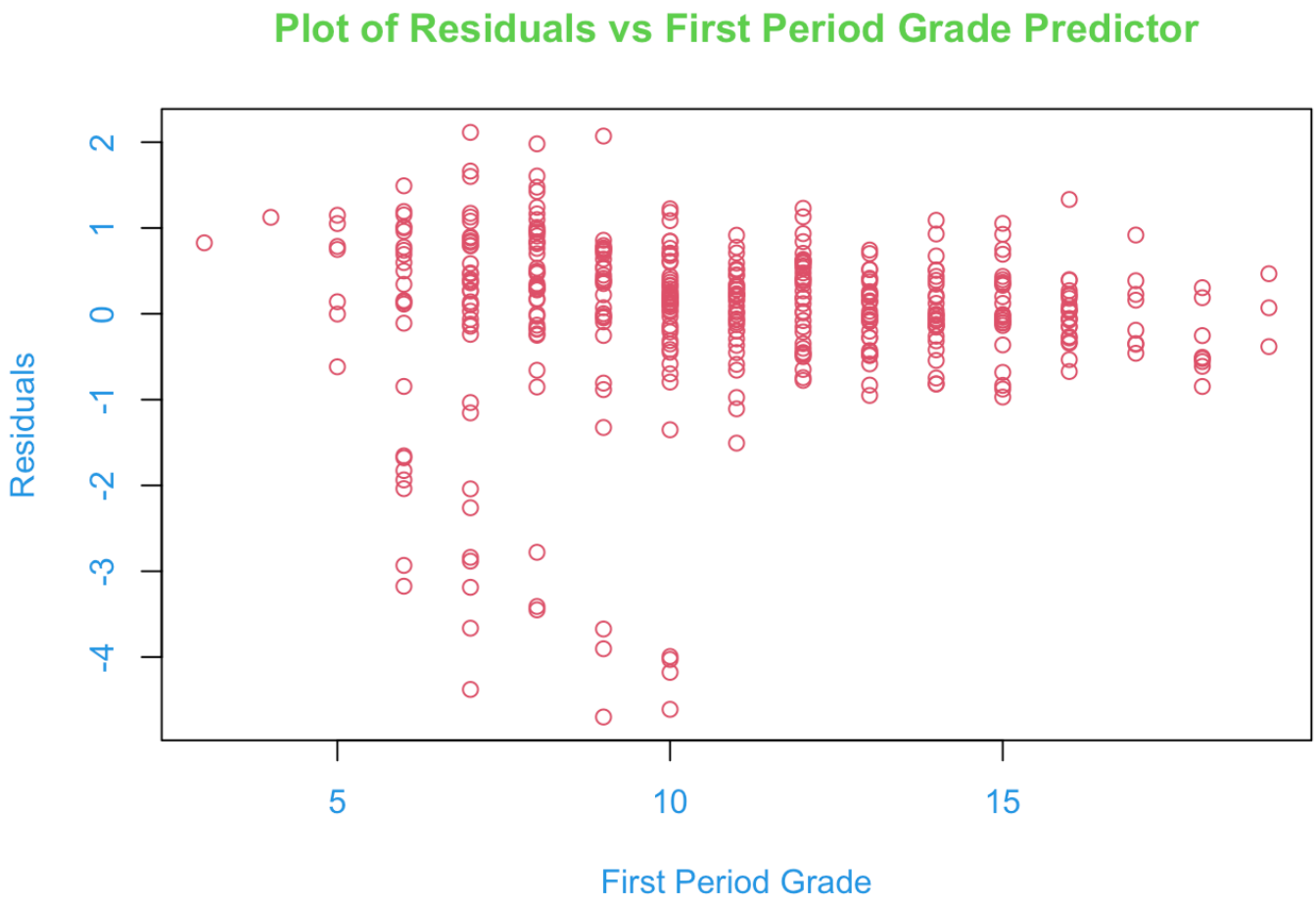


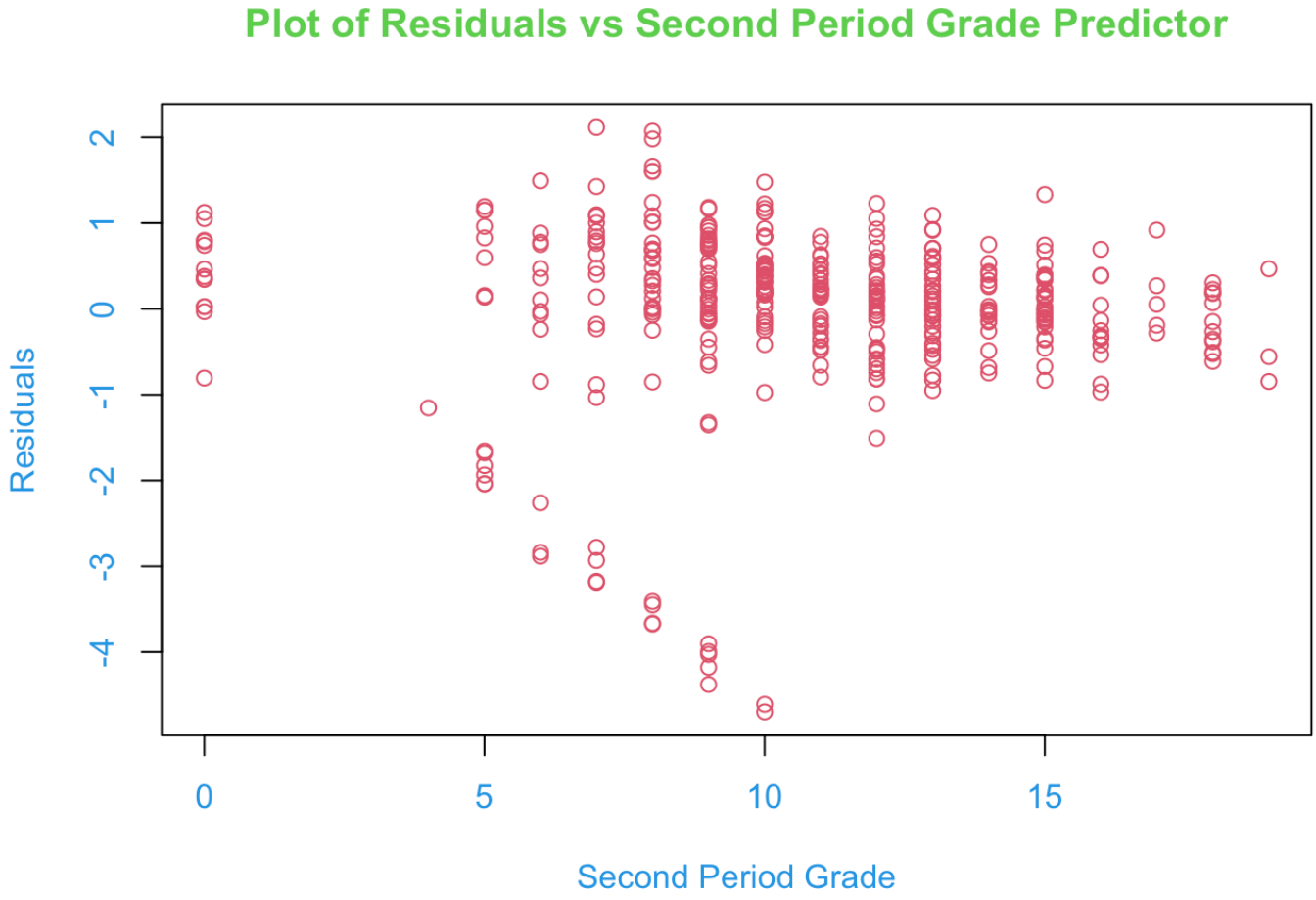
Plot of Residuals vs Travel Time Predictor

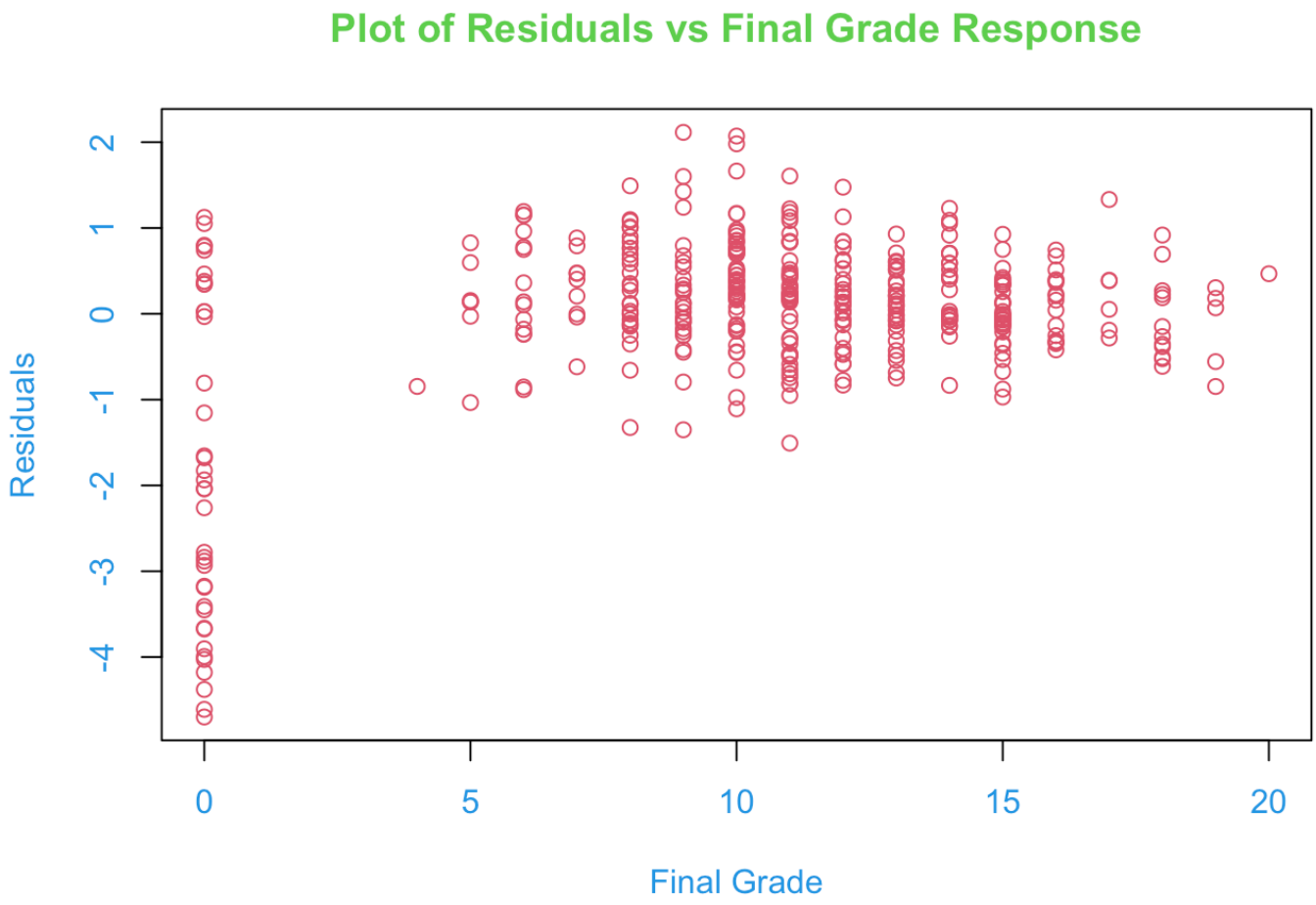


Travel Time


Plot of Residuals vs Absences Predictor
Residuals
2
1
0
-1
-2
-3
-4
0
20
40
60
Absences


Plot of Residuals vs First Period Grade Predictor
Residuals
2
1
0
-1
-2
-3
-4
5
10
15
First Period Grade


Plot of Residuals vs Second Period Grade Predictor
Residuals
2
1
0
-1
-2
-3
-4
0
5
10
15
Second Period Grade



Here we can see that for most of the predictor variables considered, the residuals are randomly distributed and more concentrated near 0. But for residuals vs response variable, there is some deviation for the values for Final Grade = 0.

```
# again check the available/defined methods for class "my_resid"

methods(class="my_resid")
```

```
## [1] plot    print   summary
## see '?methods' for accessing help and source code
```

Hence we get the 3 defined methods (plot, print, summary) for class “my_resid”