DATA ANALYSIS

SALVADOR CASTRO

Contents

1 Data Cleaning & Screening **3**

1.1 Import data To R . . . 3

1.2 Data Screening . . . 3

1.3 Data Cleaning . . . 4

1.3.1 Categorical Variblaes: . . . 4

1.3.2 Continious Variblaes . . . 8

1.3.3 Ordinal Variblaes . . . 10

2 Data Query & Manipulation **17**

2.1 Display the number of cases that have missing data . . . 17

2.1.1 Number of incomplete cases . . . 17

2.1.2 Number of missing values . . . 17

2.1.3 Which cases (row numbers) are incomplete? . . . 17

2.1.4 Count of missing values per variable . . . 17

2.2 Display all missing values by data type. . . . 17

2.2.1 Nominal Variables . . . 17

2.2.2 Continious Variables . . . 18

2.2.3 Ordinal Variables . . . 18

3 Likert Items **19**

3.1 Calculate Cronbach's coefficient alpha . . . 20

3.1.1 Plots . . . 23

4 Data Analysis **30**

4.1 Descriptive Statistics . . . 30

4.2 Pearson product-moment correlation coefficient . . . 33

4.2.1 Assumptions of Pearson product-moment correlation . . . 33

4.2.2 Assumption #2: Test univariate normal distribution . . . 34

4.2.3 Assumption #3: Test pairwise linearity . . . 41

4.2.4 Assumption #4: Test Bivariate Outliers . . . 41

4.2.5 Quantile-Quantile Plots . . . 46

4.2.6 Assumption #5: Test Bivariate Normality . . . 46

4.2.7 Assumption #6: Test Homoscedasticity . . . 51
4.3 Pearson product-moment correlation coefficient . . . 54
4.3.1 Spearman's rank correlation coefficient . . . 57
4.4 Bivariate Linear Regression . . . 58
4.4.1 Assumptions of bivariate linear regression . . . 58
4.4.2 Test Univariate and Bivariate Outliers . . . 59
4.4.3 Mahalanobis' Distance . . . 61
4.4.4 Outliers based on adjusted quantile plots . . . 65
4.4.5 Normality of Residuals . . . 67
4.4.6 Leverage Plots . . . 68
4.4.7 Cook's Distanece . . . 72
4.4.8 Test Non-independence of Errors . . . 74
4.4.9 Test Homoscedasticity . . . 74
4.4.10 Studentized Residuals vs. Fitted Values . . . 74
4.5 Chi-square test of goodness of fit . . . 76
4.6 One-sample Kolmogorov-Smirnov test . . . 80
4.7 Chi-square test of independence (cross-tabulation) . . . 80
4.8 One-sample t-test . . . 87
4.9 Dependent (paired samples) t-test . . . 89
4.10 Cohen's d (Effect Size) . . . 94
4.11 Wilcoxon signed-rank test . . . 96
4.12 Independent samples t-test, Mann–Whitney U test and Point biserial correlation . . . 96
4.12.1 Mann–Whitney U test . . . 99
4.12.2 Point-biserial correlation coefficient . . . 101
4.13 One-way ANOVA and Kruskal-Wallis . . . 102
4.14 Omnibus Tests: . . . 102
4.14.1 Bartlett test of homogeneity of variances . . . 102
4.14.2 Levene's Test for Homogeneity of Variance . . . 102
4.15 Post-Hoc Tests: . . . 106
4.15.1 Tukey's HSD (honest significant difference) . . . 106
4.15.2 Means Plot . . . 107
4.16 Kruskal-Wallis one-way analysis of variance . . . 107

1 Data Cleaning & Screening

A short survey was given to graduate students to study statistics anxiety, statistics achievement and their relationship to other variables. Project Grade and Final Exam were entered later by their statistics course instructors.

Online survey can be accessed here: https://ohio.qualtrics.com/jfe/form/SV_2rieFgVELzdFdSl

1.1 Import data To R

```
# read data file from the hard disk
mydata <- read.csv("/Users/salvadorcastro/Desktop/R Files/Data Analysis/anxietydata.csv",
                   header = TRUE)
dim(mydata)
```

```
## [1] 110  13
```

Use the sample( ) function to take a random sample of size n from a dataset. Sample takes a sample of the specified size from the elements of x using either with or without replacement.

```
mysample <- as.data.frame(mydata[sample(1:nrow(mydata), 100, replace = FALSE),])
dim(mysample) # size of the data farame
```

```
## [1] 100  13
```

```
# We will use mydata without random sampling
```

```
#save "mysample" to hard disk as csv file
write.csv(mysample, file = "mysample.csv")
```

1.2 Data Screening

Non-NA values cannot be interpreted as missing: Other packages allow you to designate values as "system missing" so that these values will be interpreted in the analysis as missing. In R, you would need to explicitly change these values to NA. The $ is.na() $ function can be used to make such a change.

Most statistical analyses use listwise deletion as a method for handling missing data. In this method, an entire record is excluded from analysis if any single value is missing. Instead, we will employ multiple data imputation as a method for handling missing data.

Even though it is possible to inspect 13 variables and 110 observations visually, we adopted a very large database VLDB analysis and avoided viewing the data as a whole. Instead, we employed relational operations to screen the data and detect possible problems.

Display the structure of the data: variable names, types and the dimensions of the data frame.

110 observations, 13 variables:

- 2 integer (int) variables: id, gre
- 2 real number (num) variables:

- 3 nominal variables (Factor): residence, major, computer
- 5 ordinal variables (Factor): general, affective, cognitive, value, difficulty, all Likert scales

```
str(mydata)
```

```
## 'data.frame':    110 obs. of  13 variables:
##  $ id        : int  3 34 40 66 67 74 81 85 87 92 ...
##  $ residence : Factor w/ 5 levels "Bad data","East",..: 5 4 2 2 4 5 2 3 2 4 ...
##  $ major     : Factor w/ 3 levels "Other Major",..: NA 3 3 1 2 3 3 2 2 2 ...
##  $ computer  : Factor w/ 4 levels "Mac","Missing Completely",..: 3 1 3 3 1 1 1 1 1 1 ...
##  $ gpa       : num  2.73 3.8 3.49 3.41 3.45 2.96 2.71 3.51 2.69 3.33 ...
##  $ gre       : int  134 154 156 162 162 147 148 158 141 150 ...
##  $ project   : num  97.2 86.8 93.8 89.3 75.8 79.9 93.2 75 84.2 85.3 ...
##  $ final     : num  73.8 98.7 70.4 87.5 66.8 41.9 100 66 68.1 77 ...
##  $ general   : Factor w/ 6 levels "A","D","Missing Completely",..: 2 1 4 4 1 3 2 4 2 4 ...
##  $ affective : Factor w/ 5 levels "A","D","N","SA",..: 1 3 3 3 1 1 4 2 4 3 ...
##  $ cognitive : Factor w/ 4 levels "A","D","N","SD": 4 3 3 3 1 2 2 3 2 2 ...
##  $ value     : Factor w/ 6 levels "A","D","Missing Completely",..: 1 2 4 1 2 1 4 2 5 4 ...
##  $ difficulty: Factor w/ 5 levels "A","D","N","SA",..: 1 2 3 3 2 1 3 5 4 2 ...
```

Display the umber of cases that have missing data labeled as "NA" or completely missing (blank). 12 records would be excluded from analysis in listwise deletion

```
sum(!complete.cases(mydata))
```

```
## [1] 12
```

1.3 Data Cleaning

1.3.1 Categorical Variblaes:

residence

- 1 = East
- 2 = West
- 3 = South
- 4 = North

Display all the unique values or factors

Notice:

(1) levels are not labelled correctly and

(2) missing or bad data are also considered to be a level

```
head(mydata$residence, 10)
```

```
##  [1] West  South East  East  South West  East  North East  South
## Levels: Bad data East North South West
```

```
as.integer(head(mydata$residence, 10))
```

```
##  [1] 5 4 2 2 4 5 2 3 2 4
```

Display rows in which variable “residence” has an unexpected value (bad data, missing data)

```
mydata[which(mydata$residence != 'East' &
               mydata$residence != 'West' &
               mydata$residence != 'South' &
               mydata$residence != 'North'),
       c("id", "residence")]
```

```
##     id residence
## 76 680  Bad data
```

```
# 76th case with an ID number 680 has a value coded as "Bad Data" in the "variance"residence" column
```

Restructure the variable “residence” as a nominal variable (factors) with levels (“NA” is not a category):

```
levels <- c("East", "West", "South", "North")
mydata$residence <- factor(mydata$residence, levels, labels = levels , exclude = NA)
```

Check to see the results

```
# Levels are labelled correctly and missing data is not among them
head(mydata$residence, 10)
```

```
##  [1] West  South East  East  South West  East  North East  South
## Levels: East West South North
```

```
head(as.integer(mydata$residence), 10)
```

```
##  [1] 2 3 1 1 3 2 1 4 1 3
```

Display the locations of missing values labelled as “NA”.

```
# Notice "bad data" has been relabelled as "NA"
mydata[which(is.na(mydata$residence)), c("id", "residence")]
```

```
##     id residence
## 76 680      <NA>
```

Alternative code to replace all bad data with “NA” (missing value)

```
is.na(mydata$residence) <- which((mydata$residence != "East") &
                                   (mydata$residence != "West") &
                                   (mydata$residence != "South") &
                                   (mydata$residence != "North")
)
```

major

*1 = Other Major

*2 = Physical Science

*3 = Social Science

Display all the unique values or factors

```
as.integer(head(mydata$major, 5))
```

```
## [1] NA  3  3  1  2
```

Select rows in which variable "major" has an unexpected value (bad data)

```
# There appears to be none
mydata[which(mydata$major != "Other Major" &
               mydata$major != "Physical Science" &
               mydata$major != "Social Science"),
       c("id", "major")]
```

```
## [1] id    major
## <0 rows> (or 0-length row.names)
```

Restructure the variable "major" as a nominal variable (factors) with levels ("NA" is not a category):

*1 = Other Major

*2 = Physical Science

*3 = Social Science

```
levels <- c("Other Major", "Physical Science", "Social Science")
mydata$major <- factor(mydata$major, levels, labels = levels, exclude = NA)
```

Check to see the results

```
# Levels are labelled correctly and missing data is not among them
head(mydata$major, 5)
```

```
## [1] <NA>             Social Science   Social Science   Other Major
## [5] Physical Science
## Levels: Other Major Physical Science Social Science
```

```
head(as.integer(mydata$major), 5)
```

```
## [1] NA  3  3  1  2
```

Display the locations of missing values in "major".

```
mydata[which(is.na(mydata$major)), c("id", "major")]
```

```
##     id major
## 1    3  <NA>
## 66 594  <NA>
## 76 680  <NA>
```

computer

*1 = Mac

*2 = PC

Display all the unique values or factors

```
# Notice, levels are not labelled correctly and missing
# or bad data are also considered a factor
head(mydata$computer, 10)
```

```
##  [1] PC  Mac PC  PC  Mac Mac Mac Mac Mac Mac
## Levels: Mac Missing Completely PC Some other problem
```

```
as.integer(head(mydata$computer, 10))
```

```
##  [1] 3 1 3 3 1 1 1 1 1 1
```

Select rows in which variable "computer" has an unexpected value (bad data)

```
mydata[which(mydata$computer != "Mac" &
               mydata$computer != "PC"),
       c("id", "computer")]
```

```
##      id           computer
## 49  448 Some other problem
## 100 916 Missing Completely
```

Restructure the variable "major" as a nominal variable (factors) with levels ("NA" is not a category):

```
levels <- c("Mac", "PC")
mydata$computer <- factor(mydata$computer, levels, labels = levels , exclude = NA)
```

Check to see the results

```
# Levels are labelled correctly and missing data is not among them
head(mydata$computer)
```

```
## [1] PC  Mac PC  PC  Mac Mac
## Levels: Mac PC
```

```
head(as.numeric(mydata$computer))
```

```
## [1] 2 1 2 2 1 1
```

Display the locations of missing values in "computer".

```
mydata[which(is.na(mydata$computer)), c("id", "computer")]
```

```
##      id computer
## 49  448     <NA>
## 85  752     <NA>
## 100 916     <NA>
```

1.3.2 Continious Variblaes

gpa

*min = 0.00

*max = 4.00

Display the range of values the variable "computer" takes

```
summary(mydata$gpa)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.710   2.870   3.195   3.165   3.418   4.440
```

```
# Possible values are: min=0.00 and max=4.00. So, gpa of 4.44 appears to be a bad data
```

Select rows in which variable "gpa" has unexpected values (bad data)

```
mydata[which((mydata$gpa < 0.00) | (mydata$gpa > 4.00)), c("id", "gpa")]
```

```
##     id  gpa
## 37 364 4.44
```

Replace all bad data with "NA" (missing value) and display the locations of missing values in "gpa".

```
is.na(mydata$gpa) <- which((mydata$gpa < 0.00) | (mydata$gpa > 4.00))
mydata[which(is.na(mydata$gpa)), c("id", "gpa")]
```

```
##     id gpa
## 37 364  NA
```

gre

*min = 130

*max = 170

Display the range of values the variable "gre" takes

```
summary(mydata$gre)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -2.0   142.0   149.5   149.0   156.8   170.0
```

Select rows in which variable "gre" has an unexpected values (bad data)

```
mydata[which((mydata$gre < 130) | (mydata$gre > 170)), c("id", "gre")]
```

```
##     id gre
## 72 644  -2
```

Replace all bad data with "NA" (missing value) and display the locations of missing values in "gre".

```
is.na(mydata$gre) <- which((mydata$gre < 130) | (mydata$gre > 170))
mydata[which(is.na(mydata$gre)), c("id", "gre")]
```

```
##     id gre
## 72 644  NA
```

project

*min = 0.0

*max = 100.0

Display the range of values the variable "computer" takes

```
# Possible values are: min=0 and max=100. So, gre of -1.0 appears to be a bad data
summary(mydata$project)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -1.00   84.12   87.60   86.66   91.07  100.00
```

Select rows in which variable "project" has an unexpected values (bad data)

```
mydata[which((mydata$project < 0.0) | (mydata$project > 100.0)), c("id", "project")]
```

```
##     id project
## 76 680      -1
```

Replace all bad data with "NA" (msising value) and display the locations of missing values in "project".

```
is.na(mydata$project) <- which((mydata$project < 0.0) | (mydata$project > 100.0))
mydata[which(is.na(mydata$project)), c("id", "project")]
```

```
##     id project
## 76 680      NA
```

final

Display the range of values the variable "computer" takes

```
# Possible values are: min=0 and max=100.
summary(mydata$final)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   41.90   70.65   80.05   80.18   92.28  100.00
```

Select rows in which variable "final" has an unexpected values (bad data)

```
# There appears to be none
mydata[which((mydata$final < 0.0) | (mydata$final > 100.0)), c("id", "final")]
```

```
## [1] id    final
## <0 rows> (or 0-length row.names)
```

Display the locations of missing values in "final".

```
mydata[which(is.na(mydata$final)), c("id", "final")]
```

```
## [1] id    final
## <0 rows> (or 0-length row.names)
```

1.3.3 Ordinal Variblaes

Likert Scale

Check the rating that most closely describes your feelings about these statements concerning statistics on the scale from Strongly Disagree to Strongly Agree (SD=1 and SA=5).

Variable	Likert Item	SD D N A SA
general	I will like statistics (+)	[ ] [ ] [ ] [ ] [ ]
affective	I am scared by statistics (–)	[ ] [ ] [ ] [ ] [ ]
cognitive	I can learn statistics (+)	[ ] [ ] [ ] [ ] [ ]
value	Statistics is irrelevant in my life (–)	[ ] [ ] [ ] [ ] [ ]
difficulty	Statistics is a complicated subject (–)	[ ] [ ] [ ] [ ] [ ]

general

Display all the unique values or factors. Notice there are 6 levels including missing data and they are not labelled correctly

```
# Labels
labels(levels(mydata$general))
```

```
## [1] "1" "2" "3" "4" "5" "6"
```

```
# Levels
levels(mydata$general)
```

```
## [1] "A"                  "D"                  "Missing Completely"
## [4] "N"                  "SA"                 "SD"
```

```
head(mydata$general, 5)
```

```
## [1] D A N N A
## Levels: A D Missing Completely N SA SD
```

```
as.integer(head(mydata$general, 5))
```

```
## [1] 2 1 4 4 1
```

Select rows in which variable "general" has an unexpected value (bad data)

```
mydata[which((mydata$general != "SA") &
               (mydata$general != "A") &
               (mydata$general != "N") &
               (mydata$general != "D") &
               (mydata$general != "SD")),
       c("id", "general")]
```

```
##   id            general
## 6 74 Missing Completely
```

Alternative code to replace bad data with missing values

```
# Replace all bad data with "NA" (misising value)
is.na(mydata$general) <- which((mydata$general != "SA") &
                                 (mydata$general != "A") &
                                 (mydata$general != "N") &
                                 (mydata$general != "D") &
                                 (mydata$general != "SD"))
```

Restructure *general* as an ordinal variable (factors) with 5 levels (NA is not a factor):

SD=1

D=2

N=3

A=4

SA=5

```
levels <- c("SD", "D", "N", "A", "SA")
mydata$general <- ordered(mydata$general, levels, labels = levels, exclude = NA)
```

Check to see if it worked

```
head(mydata$general, 10)
```

```
##  [1] D    A    N    N    A    <NA> D    N    D    N
## Levels: SD < D < N < A < SA
```

```
head(as.integer(mydata$general), 10)
```

```
##  [1]  2  4  3  3  4 NA  2  3  2  3
```

Display the locations of missing values in "general".

```
# Notice Missing Completely on row 6 converted to NA
mydata[which(is.na(mydata$general)), c("id", "general")]
```

```
##     id general
## 6   74    <NA>
## 38 367    <NA>
```

affective

Display all the unique values or factors. Notice there are 6 levels including missing data and they are not labelled correctly

```
# Labels
labels(levels(mydata$affective))
```

```
## [1] "1" "2" "3" "4" "5"
```

```
# Levels
levels(mydata$affective)
```

```
## [1] "A"  "D"  "N"  "SA" "SD"
```

```
head(mydata$affective, 10)
```

```
##  [1] A  N  N  N  A  A  SA D  SA N
## Levels: A D N SA SD
```

```
as.integer(head(mydata$affective, 10))
```

```
##  [1] 1 3 3 3 1 1 4 2 4 3
```

Select rows in which variable "affective" has an unexpected value (bad data)

```
mydata[which((mydata$affective != "SA") &
               (mydata$affective != "A") &
               (mydata$affective != "N") &
               (mydata$affective != "D") &
               (mydata$affective != "SD")),
       c("id", "affective")]
```

```
## [1] id        affective
## <0 rows> (or 0-length row.names)
```

Restructure *affective* as an ordinal variable (factors) with 5 levels (NA is not a factor):

```
levels <- c("SD", "D", "N", "A", "SA")
mydata$affective <- factor(mydata$affective, levels, labels = levels, ordered= TRUE, exclude = NA)
```

Check to see if it worked

```
head(mydata$affective, 10)
```

```
##  [1] A  N  N  N  A  A  SA D  SA N
## Levels: SD < D < N < A < SA
```

```
head(as.integer(mydata$affective), 10)
```

```
##  [1] 4 3 3 3 4 4 5 2 5 3
```

Display the locations of missing values in "affective". Notice Missing Completely on row 6 converted to NA

```
mydata[which(is.na(mydata$affective)), c("id", "affective")]
```

```
##     id affective
## 68 606      <NA>
## 79 716      <NA>
```

cognitive

Display all the unique values or factors. Notice there are 6 levels including missing data and they are not labelled correctly

```
# Labels
labels(levels(mydata$cognitive))
```

```
## [1] "1" "2" "3" "4"
```

```
# Levels
levels(mydata$cognitive)
```

```
## [1] "A"  "D"  "N"  "SD"
```

```
head(mydata$cognitive, 10)
```

```
##  [1] SD N  N  N  A  D  D  N  D  D
## Levels: A D N SD
```

```
as.integer(head(mydata$cognitive, 10))
```

```
##  [1] 4 3 3 3 1 2 2 3 2 2
```

Select rows in which variable "cognitive" has an unexpected value (bad data)

```
mydata[which((mydata$cognitive != "SA") &
               (mydata$cognitive != "A") &
               (mydata$cognitive != "N") &
               (mydata$cognitive != "D") &
               (mydata$cognitive != "SD")),
       c("id", "cognitive")]
```

```
## [1] id        cognitive
## <0 rows> (or 0-length row.names)
```

Restructure *cognitive* as an ordinal variable (factors) with 5 levels (NA is not a factor):

```
levels <- c("SD", "D", "N", "A", "SA")
mydata$cognitive <- ordered(mydata$cognitive, levels, labels = levels, exclude = NA)
```

Check to see if it worked

```
head(mydata$cognitive, 10)
```

```
##  [1] SD N  N  N  A  D  D  N  D  D
## Levels: SD < D < N < A < SA
```

```
head(as.integer(mydata$cognitive), 10)
```

```
##  [1] 1 3 3 3 4 2 2 3 2 2
```

Display the locations of missing values in "cognitive". Notice Missing Completely on row 6 converted to NA

```
mydata[which(is.na(mydata$cognitive)), c("id", "cognitive")]
```

```
##     id cognitive
## 16 172      <NA>
```

value

Display all the unique values or factors. Notice there are 6 levels including missing data and they are not labelled correctly

```
# Labels
labels(levels(mydata$value))
```

```
## [1] "1" "2" "3" "4" "5" "6"
```

```
# Levels
levels(mydata$value)
```

```
## [1] "A"                  "D"                  "Missing Completely"
## [4] "N"                  "SA"                 "SD"
```

```
head(mydata$value, 10)
```

```
##  [1] A  D  N  A  D  A  N  D  SA N
## Levels: A D Missing Completely N SA SD
```

```
as.integer(head(mydata$value, 10))
```

```
##  [1] 1 2 4 1 2 1 4 2 5 4
```

Select rows in which variable "value" has an unexpected value (bad data)

```
mydata[which((mydata$value != "SA") &
               (mydata$value != "A") &
               (mydata$value != "N") &
               (mydata$value != "D") &
               (mydata$value != "SD")),
       c("id", "value")]
```

```
##     id              value
## 92 801 Missing Completely
```

Restructure *value* as an ordinal variable (factors) with 5 levels (NA is not a factor):

```
levels <- c("SD", "D", "N", "A", "SA")
mydata$value <- factor(mydata$value, levels, labels = levels, ordered= TRUE, exclude = NA)
```

Check to see if it worked

```
head(mydata$value, 10)
```

```
##  [1] A  D  N  A  D  A  N  D  SA N
## Levels: SD < D < N < A < SA
```

```
head(as.integer(mydata$value), 10)
```

```
##  [1] 4 2 3 4 2 4 3 2 5 3
```

Display the locations of missing values in "value". Notice Missing Completely on row 6 converted to NA

```
mydata[which(is.na(mydata$value)), c("id", "value")]
```

```
##     id value
## 84 743  <NA>
## 92 801  <NA>
## 93 808  <NA>
```

difficulty

Display all the unique values or factors. Notice there are 6 levels including missing data and they are not labelled correctly

```
# Labels
labels(levels(mydata$difficulty))
```

```
## [1] "1" "2" "3" "4" "5"
```

```
# Levels
levels(mydata$difficulty)
```

```
## [1] "A"  "D"  "N"  "SA" "SD"
```

```
head(mydata$difficulty, 10)
```

```
##  [1] A  D  N  N  D  A  N  SD SA D
## Levels: A D N SA SD
```

```
as.integer(head(mydata$difficulty, 10))
```

```
##  [1] 1 2 3 3 2 1 3 5 4 2
```

Select rows in which variable "difficulty" has an unexpected value (bad data)

```
mydata[which((mydata$difficulty != "SA") &
               (mydata$difficulty != "A") &
               (mydata$difficulty != "N") &
               (mydata$difficulty != "D") &
               (mydata$difficulty != "SD")), c("id", "difficulty")]
```

```
## [1] id         difficulty
## <0 rows> (or 0-length row.names)
```

Restructure *difficulty* as an ordinal variable (factors) with 5 levels (NA is not a factor):

```
levels <- c("SD", "D", "N", "A", "SA")
mydata$difficulty <- factor(mydata$difficulty, levels, labels = levels, ordered= TRUE, exclude = NA)
```

Check to see if it worked

```
head(mydata$difficulty, 10)
```

```
##  [1] A  D  N  N  D  A  N  SD SA D
## Levels: SD < D < N < A < SA
```

```
head(as.integer(mydata$difficulty), 10)
```

```
##  [1] 4 2 3 3 2 4 3 1 5 2
```

Display the locations of missing values in *difficulty*. Notice Missing Completely on row 6 converted to NA

```
mydata[which(is.na(mydata$difficulty)), c("id", "difficulty")]
```

```
##     id difficulty
## 45 418       <NA>
## 50 460       <NA>
```

2 Data Query & Manipulation

2.1 Display the number of cases that have missing data

If the cases with missing values were to be deleted listwise 18 records would be lost

2.1.1 Number of incomplete cases

```
sum(!complete.cases(mydata))
```

```
## [1] 18
```

2.1.2 Number of missing values

```
sum(is.na(mydata))
```

```
## [1] 20
```

2.1.3 Which cases (row numbers) are incomplete?

```
which(!complete.cases(mydata))
```

```
##  [1]   1   6  16  37  38  45  49  50  66  68  72  76  79  84  85  92  93
## [18] 100
```

2.1.4 Count of missing values per variable

```
sapply(mydata, function(x) sum(is.na(x)))
```

```
##         id  residence      major   computer        gpa        gre
##          0          1          3          3          1          1
##    project      final    general  affective  cognitive      value
##          1          0          2          2          1          3
## difficulty
##          2
```

2.2 Display all missing values by data type.

2.2.1 Nominal Variables

```
subset(mydata[, c("id", "residence", "major", "computer")],
       !complete.cases(mydata[, c("id", "residence", "major", "computer")]))
```

```
##      id residence            major computer
## 1     3      West             <NA>       PC
## 49  448     South Physical Science     <NA>
## 66  594     North             <NA>       PC
## 76  680      <NA>             <NA>       PC
## 85  752     South      Other Major     <NA>
## 100 916      East Physical Science     <NA>
```

2.2.2 Continious Variables

```
subset(mydata[, c("id", "gre", "gpa", "project", "final")],
       !complete.cases(mydata[, c("id", "gre", "gpa", "project", "final")]))
```

```
##     id gre  gpa project final
## 37 364 149   NA    85.2  98.6
## 72 644  NA 3.65    84.9  64.5
## 76 680 150 3.24      NA  63.4
```

2.2.3 Ordinal Variables

```
subset(mydata[, c("id", "general", "affective", "cognitive", "value", "difficulty")], !complete.cases(my
```

```
##     id general affective cognitive value difficulty
## 6   74    <NA>         A         D     A           A
## 16 172       A         A      <NA>     N           N
## 38 367    <NA>         A         D     A           A
## 45 418      SD         A         D     A       <NA>
## 50 460       N         A         D     N       <NA>
## 68 606       N      <NA>         N     N           N
## 79 716       A      <NA>         A     D           D
## 84 743       N        SA         D  <NA>           N
## 92 801       N         A        SD  <NA>           A
## 93 808       D        SA         D  <NA>           A
```

Save the cleaned data

```
#save the cleaned data to hard disk as csv file
write.table(mydata,
            file = "mydata.clean.csv",
            sep = ",",
            row.names = F,
            col.names = T,
            na = "NA",
            append = FALSE)
```

3 Likert Items

Recode the Statistics Anxiety items (general, affective, cognitive, value, & difficulty) so they all have the same scale direction. That is, either recode general and cognitive, or recode affective, value, and difficulty. In doing the recoding, you will need to decide whether higher Statistics Anxiety total scores represent higher anxiety or higher SA total scores represent lower anxiety (i.e., better attitude toward statistics). Be sure to create a really useful variable names, variable labels, value labels, and/or missing value codes for recoded variables you create (if any).

Likert Scale

Check the rating that most closely describes your feelings about these statements concerning statistics on the scale from Strongly Disagree to Strongly Agree (SD=1 and SA=5).

Variable	**Likert Item**	SD D N A SA
general	I will like statistics (+)	[ ] [ ] [ ] [ ] [ ]
affective	I am scared by statistics (–)	[ ] [ ] [ ] [ ] [ ]
cognitive	I can learn statistics (+)	[ ] [ ] [ ] [ ] [ ]
value	Statistics is irrelevant in my life (–)	[ ] [ ] [ ] [ ] [ ]
difficulty	Statistics is a complicated subject (–)	[ ] [ ] [ ] [ ] [ ]

Import data

```
mydata.clean <- read.table(file = "mydata.clean.csv",
                           header = TRUE,
                           sep = ",",
                           numerals = "no.loss",
                           na.strings = "NA",
                           colClasses = NA,
                           nrows = -1,
                           skip = 0,
                           check.names = TRUE,
                           strip.white = FALSE,
                           allowEscapes = FALSE,
                           flush = FALSE,
                           stringsAsFactors = default.stringsAsFactors(),
                           skipNul = FALSE)
```

```
# structure of the data
str(mydata.clean)
```

```
## 'data.frame':    110 obs. of  13 variables:
##  $ id        : int  3 34 40 66 67 74 81 85 87 92 ...
##  $ residence : Factor w/ 4 levels "East","North",..: 4 3 1 1 3 4 1 2 1 3 ...
##  $ major     : Factor w/ 3 levels "Other Major",..: NA 3 3 1 2 3 3 2 2 2 ...
##  $ computer  : Factor w/ 2 levels "Mac","PC": 2 1 2 2 1 1 1 1 1 1 ...
##  $ gpa       : num  2.73 3.8 3.49 3.41 3.45 2.96 2.71 3.51 2.69 3.33 ...
```

```
##  $ gre       : int  134 154 156 162 162 147 148 158 141 150 ...
##  $ project   : num  97.2 86.8 93.8 89.3 75.8 79.9 93.2 75 84.2 85.3 ...
##  $ final     : num  73.8 98.7 70.4 87.5 66.8 41.9 100 66 68.1 77 ...
##  $ general   : Factor w/ 5 levels "A","D","N","SA",..: 2 1 3 3 1 NA 2 3 2 3 ...
##  $ affective : Factor w/ 5 levels "A","D","N","SA",..: 1 3 3 3 1 1 4 2 4 3 ...
##  $ cognitive : Factor w/ 4 levels "A","D","N","SD": 4 3 3 3 1 2 2 3 2 2 ...
##  $ value     : Factor w/ 5 levels "A","D","N","SA",..: 1 2 3 1 2 1 3 2 4 3 ...
##  $ difficulty: Factor w/ 5 levels "A","D","N","SA",..: 1 2 3 3 2 1 3 5 4 2 ...
```

Restructure the Likert items as ordinal variavles

```
levels <- c("SD", "D", "N", "A", "SA")

mydata.clean$general    <- factor(mydata.clean$general,
                                  levels, labels = levels,
                                  ordered= TRUE,
                                  exclude = NA)

mydata.clean$affective  <- factor(mydata.clean$affective,
                                  levels, labels = levels,
                                  ordered= TRUE,
                                  exclude = NA)

mydata.clean$cognitive  <- factor(mydata.clean$cognitive,
                                  levels, labels = levels,
                                  ordered= TRUE,
                                  exclude = NA)

mydata.clean$value      <- factor(mydata.clean$value, levels,
                                  labels = levels,
                                  ordered= TRUE,
                                  exclude = NA)

mydata.clean$difficulty <- factor(mydata.clean$difficulty,
                                  levels, labels = levels,
                                  ordered= TRUE,
                                  exclude = NA)
```

3.1 Calculate Cronbach's coefficient alpha

From Wikipedia, the free encyclopedia

Cronbach's is a function of the number of items in a test, the average covariance between item-pairs, and the variance of the total score. It is used as a (lowerbound) estimate of the reliability of a psychometric test.

It was first named alpha by Lee Cronbach in 1951, as he had intended to continue with further coefficients. The measure can be viewed as an extension of the Kuder–Richardson Formula 20 (KR-20), which is an equivalent measure for dichotomous items. Alpha is not robust against missing data.

Cronbach's alpha will generally increase as the intercorrelations among test items increase, and is thus known as an internal consistency estimate of reliability of test scores. Because intercorrelations among test items are maximized when all items measure the same construct, Cronbach's alpha is widely believed to indirectly indicate the degree to which a set of items measures a single unidimensional latent construct.

A commonly accepted rule for describing internal consistency using Cronbach's alpha is as follows; however, a greater number of items in the test can artificially inflate the value of alpha and a sample with a narrow range can deflate it, so this rule should be used with caution:

Cronbach's α	**Internal Consistency (reliability)**
$\alpha \geq 0.9$	*Excellent (High-Stakes testing)*
$0.7 \leq \alpha < 0.9$	*Good (Low-Stakes testing)*
$0.6 \leq \alpha < 0.7$	*Acceptable*
$0.5 \leq \alpha < 0.6$	*Poor*
$\alpha < 0.5$	*Unacceptable*

Recode the positively-worded Statistics Anxiety items (general, affective, cognitive, value, & difficulty) so they all have the same scale direction and high scores mean high statistics anxiety.

Cronbach's coefficient alpha before recoding

```
# Notice reliability is negative
psy::cronbach(data.matrix(mydata.clean[, c("general", "affective", "cognitive", "value", "difficulty")]))
```

```
## $sample.size
## [1] 100
##
## $number.of.items
## [1] 5
##
## $alpha
## [1] -0.580286
```

Recode **general** and **cognitive**

```
# Reverse the sacale direction for general and cognitive
mydata.clean$general <- likert::reverse.levels(mydata.clean$general)
mydata.clean$cognitive <- likert::reverse.levels(mydata.clean$cognitive)
```

Alternative code for recoding likert.items*items*general <- likert::recode(likert.items*items*general, from=1:5, to=5:1) likert.items*items*cognitive <- likert::recode(likert.items*items*cognitive, from=1:5, to=5:1)

Cronbach's coefficient alpha after recoding

```
# Notice the reliability id positive and very high, 0.80
# after reversing the scale direction for the positively-worded items.
psy::cronbach(data.matrix(mydata.clean[, c("general", "affective", "cognitive", "value", "difficulty")]))
```

```
## $sample.size
## [1] 100
##
## $number.of.items
## [1] 5
##
## $alpha
## [1] 0.8786137
```

A likert class with the following elements: results, items, grouping, nlevels, and summary

```
library(likert)
```

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

```
likert.scale <- likert(mydata.clean[, c("general", "affective", "cognitive", "value", "difficulty")])
```

Summaries

```
summary(likert.scale, center = (likert.scale$nlevels - 1)/2 + 1, ordered = TRUE)
```

```
##         Item       low  neutral     high     mean        sd
## 2  affective  5.555556 38.88889 55.55556 3.601852 0.7846639
## 3  cognitive 11.009174 42.20183 46.78899 3.431193 0.7860705
## 4      value 21.495327 46.72897 31.77570 3.158879 0.8594594
## 5 difficulty 26.851852 42.59259 30.55556 3.046296 0.9411095
## 1    general 28.703704 45.37037 25.92593 2.981481 0.8644752
```

```
print(likert.scale, digits = 4)
```

```
##         Item     SA     A     N     D     SD
## 1    general 2.7778 25.93 45.37 22.22  3.704
## 2  affective 0.9259  4.63 38.89 44.44 11.111
## 3  cognitive 0.0000 11.01 42.20 39.45  7.339
## 4      value 0.9346 20.56 46.73 25.23  6.542
## 5 difficulty 4.6296 22.22 42.59 25.00  5.556
```

Optional: Prints the code of the likert items for a LaTeX table to be used in LaTeX documents.

```
xtable(likert.scale)
```

```
## % latex table generated in R 3.2.2 by xtable 1.8-0 package
## % Sun Dec  6 02:48:22 2015
## \begin{tabular}{lrrrrrr}
##   \hline
## Item & n & mean & sd & low & neutral & high \\
##   \hline
## affective & 108 & 3.60 & 0.78 & 5.56 & 38.89 & 55.56 \\
##   cognitive & 108 & 3.43 & 0.79 & 11.01 & 42.20 & 46.79 \\
##   value & 109 & 3.16 & 0.86 & 21.50 & 46.73 & 31.78 \\
##   difficulty & 107 & 3.05 & 0.94 & 26.85 & 42.59 & 30.56 \\
##   general & 108 & 2.98 & 0.86 & 28.70 & 45.37 & 25.93 \\
##    \hline
## \end{tabular}
```

Optional: Prints the code of the likert items for an HTML table.

```
print(xtable(likert.scale), tabular.environment = "longtable", type = "html", floating = FALSE)
```

```
## <!-- html table generated in R 3.2.2 by xtable 1.8-0 package -->
## <!-- Sun Dec  6 02:48:22 2015 -->
## <table border=1>
## <tr> <th> Item </th> <th> n </th> <th> mean </th> <th> sd </th> <th> low </th> <th> neutral </th> <th
##   <tr> <td> affective </td> <td align="right"> 108 </td> <td align="right"> 3.60 </td> <td align="rig
##   <tr> <td> cognitive </td> <td align="right"> 108 </td> <td align="right"> 3.43 </td> <td align="rig
##   <tr> <td> value </td> <td align="right"> 109 </td> <td align="right"> 3.16 </td> <td align="right"
##   <tr> <td> difficulty </td> <td align="right"> 107 </td> <td align="right"> 3.05 </td> <td align="ri
##   <tr> <td> general </td> <td align="right"> 108 </td> <td align="right"> 2.98 </td> <td align="right
##    </table>
```

3.1.1 Plots

```
likert.density.plot(likert.scale)
```

```
likert.heat.plot(likert.scale)
```

```
## Warning: Non Lab interpolation is deprecated
```

```
plot(likert.scale, type = "bar", include.histogram = TRUE)
```

Add a column (anxiety) to hold the total score:

```
# Compute total score for cases that have at least 4 responses
head(mydata.clean)
```

```
##   id residence            major computer  gpa gre project final general
## 1  3      West             <NA>       PC 2.73 134    97.2  73.8       D
## 2 34     South   Social Science      Mac 3.80 154    86.8  98.7       A
## 3 40      East   Social Science       PC 3.49 156    93.8  70.4       N
## 4 66      East      Other Major       PC 3.41 162    89.3  87.5       N
## 5 67     South Physical Science      Mac 3.45 162    75.8  66.8       A
## 6 74      West   Social Science      Mac 2.96 147    79.9  41.9    <NA>
##   affective cognitive value difficulty
## 1         A        SD     A          A
## 2         N         N     D          D
## 3         N         N     N          N
## 4         N         N     A          N
## 5         A         A     D          D
## 6         A         D     A          A
```

```
mydata.clean$anxiety <- rep(NA, nrow(mydata.clean))

#check
head(mydata.clean)
```

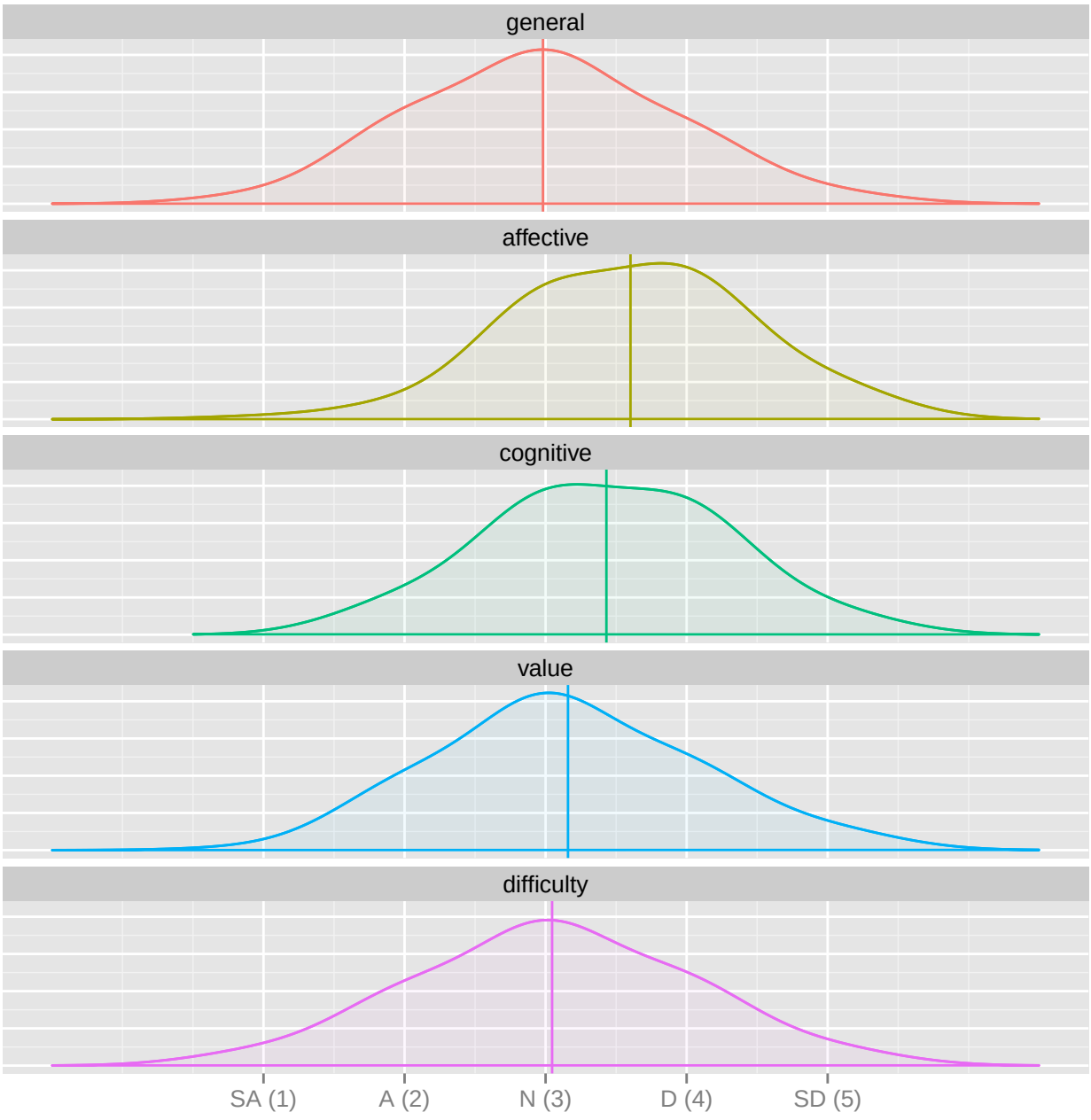
general
affective
cognitive
value
difficulty
SA (1)
A (2)
N (3)
D (4)
SD (5)

Figure 1:


value
3.16 (0.86)
0.93%
20.56%
46.73%
25.23%
6.54%
general
2.98 (0.86)
2.78%
25.93%
45.37%
22.22%
3.70%
difficulty
3.05 (0.94)
4.63%
22.22%
42.59%
25.00%
5.56%
cognitive
3.43 (0.79)
0.00%
11.01%
42.20%
39.45%
7.34%
affective
3.60 (0.78)
0.93%
4.63%
38.89%
44.44%
11.11%
Mean (SD)
SA
A
N
D
SD
Percent
100
75
50
25
0

Figure 2:

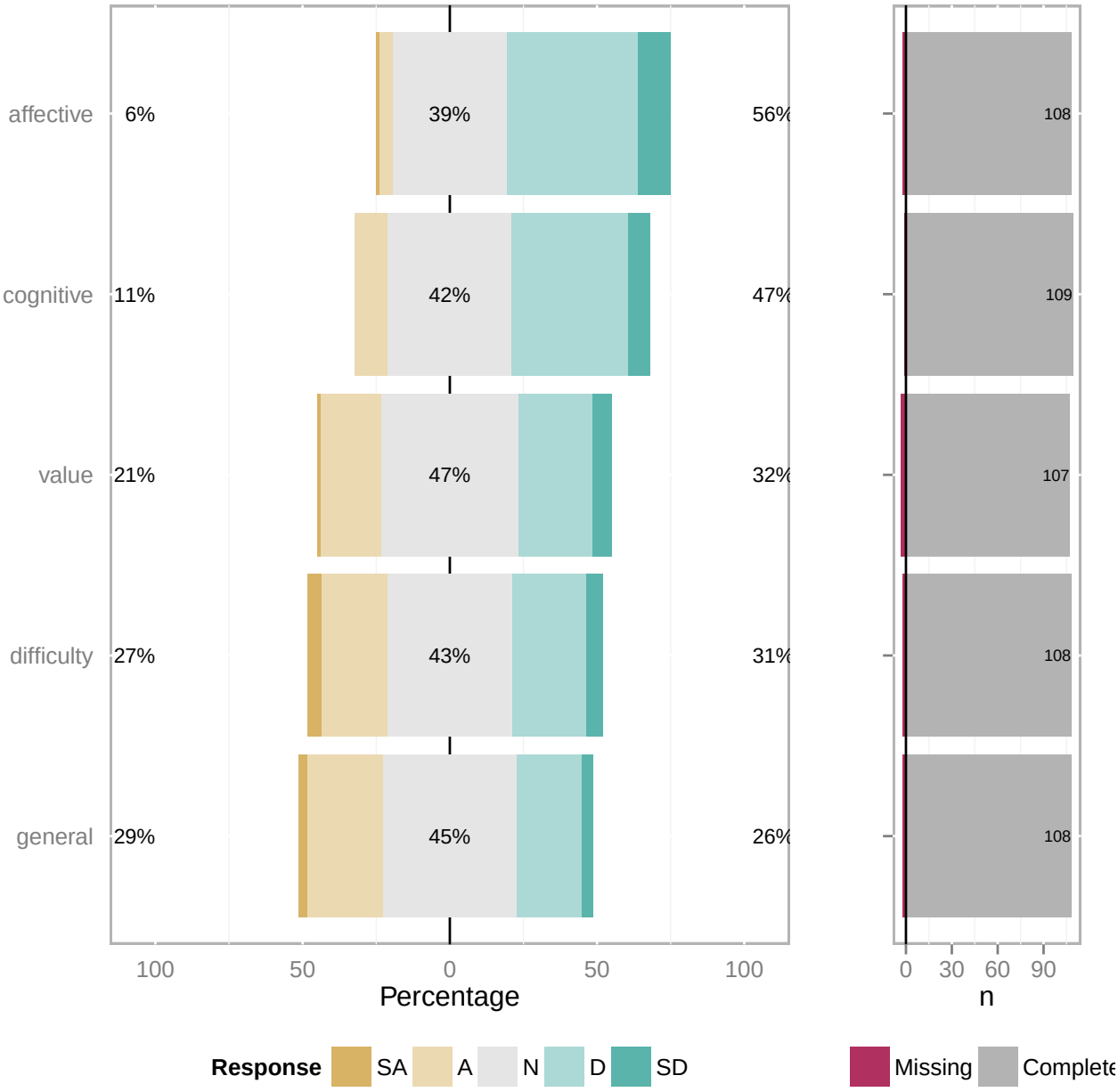
affective
cognitive
value
difficulty
general
6%
11%
21%
27%
29%
39%
42%
47%
43%
45%
56%
47%
32%
31%
26%
100
50
0
50
100
Percentage
108
109
107
108
108
0
30
60
90
n
Response
SA
A
N
D
SD
Missing
Complete

Figure 3:

```
##   id residence            major computer  gpa gre project final general
## 1  3      West             <NA>       PC 2.73 134    97.2  73.8       D
## 2 34     South   Social Science      Mac 3.80 154    86.8  98.7       A
## 3 40      East   Social Science       PC 3.49 156    93.8  70.4       N
## 4 66      East      Other Major       PC 3.41 162    89.3  87.5       N
## 5 67     South Physical Science      Mac 3.45 162    75.8  66.8       A
## 6 74      West   Social Science      Mac 2.96 147    79.9  41.9    <NA>
##   affective cognitive value difficulty anxiety
## 1         A        SD     A          A      NA
## 2         N         N     D          D      NA
## 3         N         N     N          N      NA
## 4         N         N     A          N      NA
## 5         A         A     D          D      NA
## 6         A         D     A          A      NA
```

Compute total score for cases that have at least 4 responses

```
for(i in 1:110){
  n <- 0 # number of responses
  sum <- 0 # total statistics anxiety

  for (j in 1:5){
    if(!is.na(mydata.clean[, c("general", "affective", "cognitive", "value", "difficulty")][i,j])){
      # calculate total statistics anxiety
      sum <- sum + as.numeric(mydata.clean[, c("general", "affective", "cognitive", "value", "difficulty
      # increment number of responses
      n <- n+1
    }

  }

  # if the number of responses is 4 or more score is given
  if(n >= 4) {mydata.clean[i, "anxiety"] <- sum}
}
```

```
#check
str(mydata.clean)
```

```
## 'data.frame':    110 obs. of  14 variables:
##  $ id        : int  3 34 40 66 67 74 81 85 87 92 ...
##  $ residence : Factor w/ 4 levels "East","North",..: 4 3 1 1 3 4 1 2 1 3 ...
##  $ major     : Factor w/ 3 levels "Other Major",..: NA 3 3 1 2 3 3 2 2 2 ...
##  $ computer  : Factor w/ 2 levels "Mac","PC": 2 1 2 2 1 1 1 1 1 1 ...
##  $ gpa       : num  2.73 3.8 3.49 3.41 3.45 2.96 2.71 3.51 2.69 3.33 ...
##  $ gre       : int  134 154 156 162 162 147 148 158 141 150 ...
##  $ project   : num  97.2 86.8 93.8 89.3 75.8 79.9 93.2 75 84.2 85.3 ...
##  $ final     : num  73.8 98.7 70.4 87.5 66.8 41.9 100 66 68.1 77 ...
##  $ general   : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 4 2 3 3 2 NA 4 3 4 3 ...
##  $ affective : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 3 3 3 4 4 5 2 5 3 ...
##  $ cognitive : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 5 3 3 3 2 4 4 3 4 4 ...
##  $ value     : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 4 2 4 3 2 5 3 ...
##  $ difficulty: Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 3 2 4 3 1 5 2 ...
##  $ anxiety   : num  21 12 15 16 12 16 19 11 23 15 ...
```

Add a column (rank) to hold the rank (high, medium, and low ntiles)

```
mydata.clean$rank <- rep(0, nrow(mydata.clean))

# check
str(mydata.clean)
```

```
## 'data.frame':    110 obs. of  15 variables:
##  $ id        : int  3 34 40 66 67 74 81 85 87 92 ...
##  $ residence : Factor w/ 4 levels "East","North",..: 4 3 1 1 3 4 1 2 1 3 ...
##  $ major     : Factor w/ 3 levels "Other Major",..: NA 3 3 1 2 3 3 2 2 2 ...
##  $ computer  : Factor w/ 2 levels "Mac","PC": 2 1 2 2 1 1 1 1 1 1 ...
##  $ gpa       : num  2.73 3.8 3.49 3.41 3.45 2.96 2.71 3.51 2.69 3.33 ...
##  $ gre       : int  134 154 156 162 162 147 148 158 141 150 ...
##  $ project   : num  97.2 86.8 93.8 89.3 75.8 79.9 93.2 75 84.2 85.3 ...
##  $ final     : num  73.8 98.7 70.4 87.5 66.8 41.9 100 66 68.1 77 ...
##  $ general   : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 4 2 3 3 2 NA 4 3 4 3 ...
##  $ affective : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 3 3 3 4 4 5 2 5 3 ...
##  $ cognitive : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 5 3 3 3 2 4 4 3 4 4 ...
##  $ value     : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 4 2 4 3 2 5 3 ...
##  $ difficulty: Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 3 2 4 3 1 5 2 ...
##  $ anxiety   : num  21 12 15 16 12 16 19 11 23 15 ...
##  $ rank      : num  0 0 0 0 0 0 0 0 0 0 ...
```

Create a new ordinal variable with 3 levels (i.e., high, medium, and low ntiles) for the rank of the students scores on SA.

```
mydata.clean$rank <- cut(mydata.clean$anxiety, 3, labels = c("High", "Medium", "Low"))

# check
str(mydata.clean)
```

```
## 'data.frame':    110 obs. of  15 variables:
##  $ id        : int  3 34 40 66 67 74 81 85 87 92 ...
##  $ residence : Factor w/ 4 levels "East","North",..: 4 3 1 1 3 4 1 2 1 3 ...
##  $ major     : Factor w/ 3 levels "Other Major",..: NA 3 3 1 2 3 3 2 2 2 ...
##  $ computer  : Factor w/ 2 levels "Mac","PC": 2 1 2 2 1 1 1 1 1 1 ...
##  $ gpa       : num  2.73 3.8 3.49 3.41 3.45 2.96 2.71 3.51 2.69 3.33 ...
##  $ gre       : int  134 154 156 162 162 147 148 158 141 150 ...
##  $ project   : num  97.2 86.8 93.8 89.3 75.8 79.9 93.2 75 84.2 85.3 ...
##  $ final     : num  73.8 98.7 70.4 87.5 66.8 41.9 100 66 68.1 77 ...
##  $ general   : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 4 2 3 3 2 NA 4 3 4 3 ...
##  $ affective : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 3 3 3 4 4 5 2 5 3 ...
##  $ cognitive : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 5 3 3 3 2 4 4 3 4 4 ...
##  $ value     : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 4 2 4 3 2 5 3 ...
##  $ difficulty: Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 3 2 4 3 1 5 2 ...
##  $ anxiety   : num  21 12 15 16 12 16 19 11 23 15 ...
##  $ rank      : Factor w/ 3 levels "High","Medium",..: 3 1 2 2 1 2 3 1 3 2 ...
```

Compute a new scale variable that represents the variable ACHIEVEMENT (ACH) using the average of PROJECT GRADE and FINAL EXAM, which will be considered the COURSE GRADE. Only calculate this average if the case has both grades.

```r
# Add a column (achievement) to hold the average of project and final
mydata.clean$achievement <-  rowMeans(mydata.clean[, c("project", "final")], na.rm = FALSE)

# check
str(mydata.clean)
```

```
## 'data.frame':    110 obs. of  16 variables:
##  $ id         : int  3 34 40 66 67 74 81 85 87 92 ...
##  $ residence  : Factor w/ 4 levels "East","North",..: 4 3 1 1 3 4 1 2 1 3 ...
##  $ major      : Factor w/ 3 levels "Other Major",..: NA 3 3 1 2 3 3 2 2 2 ...
##  $ computer   : Factor w/ 2 levels "Mac","PC": 2 1 2 2 1 1 1 1 1 1 ...
##  $ gpa        : num  2.73 3.8 3.49 3.41 3.45 2.96 2.71 3.51 2.69 3.33 ...
##  $ gre        : int  134 154 156 162 162 147 148 158 141 150 ...
##  $ project    : num  97.2 86.8 93.8 89.3 75.8 79.9 93.2 75 84.2 85.3 ...
##  $ final      : num  73.8 98.7 70.4 87.5 66.8 41.9 100 66 68.1 77 ...
##  $ general    : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 4 2 3 3 2 NA 4 3 4 3 ...
##  $ affective  : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 3 3 3 4 4 5 2 5 3 ...
##  $ cognitive  : Ord.factor w/ 5 levels "SA"<"A"<"N"<"D"<..: 5 3 3 3 2 4 4 3 4 4 ...
##  $ value      : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 4 2 4 3 2 5 3 ...
##  $ difficulty : Ord.factor w/ 5 levels "SD"<"D"<"N"<"A"<..: 4 2 3 3 2 4 3 1 5 2 ...
##  $ anxiety    : num  21 12 15 16 12 16 19 11 23 15 ...
##  $ rank       : Factor w/ 3 levels "High","Medium",..: 3 1 2 2 1 2 3 1 3 2 ...
##  $ achievement: num  85.5 92.8 82.1 88.4 71.3 ...
```

4 Data Analysis

4.1 Descriptive Statistics

Run appropriate procedures that will allow you to describe all your individual variables DESCRIPTIVE STATISTICS for all variables (e.g., EXPLORE is not usually appropriate for nominal variables).

Save and reload data frame

```
save(mydata.clean, file = "numeric.df.RData")

# To A Tab Delimited Text File
write.table(mydata.clean, "mydata.clean.csv",
            sep=",",
            append = FALSE,
            quote = TRUE,
            eol = "\n",
            na = "NA",
            dec = ".",
            row.names = TRUE,
            col.names = TRUE,
            qmethod = c("escape", "double")
)

mydata.clean <- read.table("mydata.clean.csv",
                           header = TRUE,
                           sep = ","
)

numeric.df <- mydata.clean[,!(names(mydata.clean) %in% c("general", "affective", "cognitive", "value",

str(numeric.df)
```

```
## 'data.frame':    110 obs. of  11 variables:
##  $ id         : int  3 34 40 66 67 74 81 85 87 92 ...
##  $ residence  : Factor w/ 4 levels "East","North",..: 4 3 1 1 3 4 1 2 1 3 ...
##  $ major      : Factor w/ 3 levels "Other Major",..: NA 3 3 1 2 3 3 2 2 2 ...
##  $ computer   : Factor w/ 2 levels "Mac","PC": 2 1 2 2 1 1 1 1 1 1 ...
##  $ gpa        : num  2.73 3.8 3.49 3.41 3.45 2.96 2.71 3.51 2.69 3.33 ...
##  $ gre        : int  134 154 156 162 162 147 148 158 141 150 ...
##  $ project    : num  97.2 86.8 93.8 89.3 75.8 79.9 93.2 75 84.2 85.3 ...
##  $ final      : num  73.8 98.7 70.4 87.5 66.8 41.9 100 66 68.1 77 ...
##  $ anxiety    : int  21 12 15 16 12 16 19 11 23 15 ...
##  $ rank       : Factor w/ 3 levels "High","Low","Medium": 2 1 3 3 1 3 2 1 2 3 ...
##  $ achievement: num  85.5 92.8 82.1 88.4 71.3 ...
```

```
Hmisc::describe(mydata.clean[,c("residence", "major", "computer", "gpa", "gre", "anxiety", "achievement
```

```
## mydata.clean[, c("residence", "major", "computer", "gpa", "gre", "anxiety", "achievement")]
##
##  7  Variables      110  Observations
## ---------------------------------------------------------------------------
## residence
```

```
##       n missing  unique
##     109       1       4
##
## East (31, 28%), North (31, 28%), South (23, 21%)
## West (24, 22%)
## ------------------------------------------------------------------------
## major
##       n missing  unique
##     107       3       3
##
## Other Major (32, 30%), Physical Science (42, 39%)
## Social Science (33, 31%)
## ------------------------------------------------------------------------
## computer
##       n missing  unique
##     107       3       2
##
## Mac (50, 47%), PC (57, 53%)
## ------------------------------------------------------------------------
## gpa
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##     109       1      80       1   3.153   2.562   2.658   2.870   3.190
##     .75     .90     .95
##   3.410   3.650   3.806
##
## lowest : 1.71 2.29 2.42 2.44 2.51, highest: 3.81 3.91 3.94 3.99 4.00
## ------------------------------------------------------------------------
## gre
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##     109       1      33       1   150.4   138.0   139.0   142.0   150.0
##     .75     .90     .95
##   157.0   163.0   166.6
##
## lowest : 134 135 137 138 139, highest: 165 166 167 169 170
## ------------------------------------------------------------------------
## anxiety
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##     110       0      18    0.99   15.93   11.00   11.00   13.25   16.00
##     .75     .90     .95
##   18.00   20.00   21.55
##
##           6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 25
## Frequency 1 1 1  2  7  7  9  5 15 14 14  8 12  5  3  1  4  1
## %         1 1 1  2  6  6  8  5 14 13 13  7 11  5  3  1  4  1
## ------------------------------------------------------------------------
## achievement
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##     109       1     104       1    83.9   67.58   71.26   77.35   84.75
##     .75     .90     .95
##   91.55   94.51   95.67
##
## lowest :  60.90  63.40  64.00  64.10  66.05
## highest:  96.30  96.60  97.00  97.80 100.00
## ------------------------------------------------------------------------
```

```
psych::describe(mydata.clean[,c("gpa", "gre", "anxiety", "achievement")])
```

```
##             vars   n   mean   sd median trimmed   mad    min max range
## gpa            1 109   3.15 0.41   3.19    3.16  0.44   1.71   4  2.29
## gre            2 109 150.40 9.11 150.00  150.02 10.38 134.00 170 36.00
## anxiety        3 110  15.93 3.47  16.00   15.92  2.97   6.00  25 19.00
## achievement    4 109  83.90 8.86  84.75   84.47 10.23  60.90 100 39.10
##              skew kurtosis   se
## gpa         -0.28     0.38 0.04
## gre          0.29    -0.74 0.87
## anxiety     -0.05     0.01 0.33
## achievement -0.52    -0.46 0.85
```

```
psych::describeBy(numeric.df[,-c(1:4, 7:8, 10)], numeric.df$rank, mat = FALSE)
```

```
## group: High
##             vars  n   mean   sd median trimmed   mad    min    max range
## gpa            1 19   3.52 0.35    3.5    3.54  0.44   2.82   4.00  1.18
## gre            2 19 157.95 8.78  158.0  158.24 10.38 141.00 170.00 29.00
## anxiety        3 19  10.74 1.59   11.0   10.94  1.48   6.00  12.00  6.00
## achievement    4 19  84.90 9.57   87.8   85.25  7.64  68.15  95.75 27.60
##              skew kurtosis   se
## gpa         -0.13    -1.17 0.08
## gre         -0.18    -1.12 2.01
## anxiety     -1.54     1.80 0.37
## achievement -0.59    -1.35 2.20
## ------------------------------------------------------------
## group: Low
##             vars  n   mean   sd median trimmed  mad    min    max range
## gpa            1 26   2.92 0.40   2.90    2.95 0.32   1.71   3.56  1.85
## gre            2 26 146.04 7.80 146.50  145.64 8.15 134.00 165.00 31.00
## anxiety        3 26  20.38 1.75  20.00   20.18 1.48  19.00  25.00  6.00
## achievement    4 26  85.53 8.79  84.78   85.96 9.97  64.00 100.00 36.00
##              skew kurtosis   se
## gpa         -0.76     1.20 0.08
## gre          0.40    -0.41 1.53
## anxiety      1.03    -0.13 0.34
## achievement -0.39    -0.52 1.72
## ------------------------------------------------------------
## group: Medium
##             vars  n   mean   sd median trimmed  mad    min    max range
## gpa            1 64   3.14 0.35   3.20    3.14 0.36   2.42   3.81  1.39
## gre            2 64 149.94 8.42 150.00  149.58 9.64 134.00 169.00 35.00
## anxiety        3 65  15.66 1.55  16.00   15.70 1.48  13.00  18.00  5.00
## achievement    4 64  82.94 8.68  84.43   83.54 9.16  60.90  97.80 36.90
##              skew kurtosis   se
## gpa         -0.25    -0.76 0.04
## gre          0.26    -0.86 1.05
## anxiety     -0.25    -0.95 0.19
## achievement -0.57    -0.29 1.09
```

```
# To A Tab Delimited Text File
write.table(numeric.df, "numeric.df.csv",
            sep=",",
            append = FALSE,
            quote = TRUE,
            eol = "\n",
            na = "NA",
            dec = ".",
            row.names = TRUE,
            col.names = TRUE,
            qmethod = c("escape", "double")
)
```

4.2 Pearson product-moment correlation coefficient

The Pearson product-moment correlation coefficient (Pearson's correlation, for short) is a measure of the strength of a linear association between two variables and is denoted by r.

Research Question: Are there relationships between GPA, GRE, Statistics Anxiety (SA), and Achievement (ACH) among graduate students?

A Pearson correlation was performed to assess whether scores of GRE could be predicted from scores of achievement among 110 graduate students currently enrolled in an educational statistics course. Anxiety scores were obtained by averaging their project and final exam grades in this course; the range of possible scores was from 0 to 100. Examination of histograms indicated that the distribution shapes were close to normal for both variables. Both distributions were skewed. However, the skewness was not judged severe enough to require data transformation or removal of outliers. The scatter plot of GRE with GPA suggested a positive linear relationship. The correlation between GRE and GPA was statistically significant, r(107) = +0.478, p < 0.0001 (two-tailed). The r^2 was 0.23; thus, about 23% of the variance in GRE could be predicted from scores of GPA.

```
numeric.df <- read.table("numeric.df.csv",
                         header = TRUE,
                         sep = ",")

# row.names(numeric.df) <- numeric.df$id
```

4.2.1 Assumptions of Pearson product-moment correlation

- Assumption #1: Each score on X should be independent of other X scores (and each score on Y should be independent of other Y scores).
- Assumption #2: Scores on both X and Y should be quantitative and normally distributed.
- Assumption #3: Scores on Y should be linearly related to scores on X.
- Assumption #4: There shouldn't be any significant bivariate outliers.
- Assumption #5: X, Y scores should have a bivariate normal distribution.
- Assumption #6: Scores on Y should have roughly equal or homogeneous variance across levels of X and vice versa (homoscedasticity).

4.2.2 Assumption #2: Test univariate normal distribution

```
# Density plot of GPA vs a normal curve with the same parameters
x <- na.omit(numeric.df$gpa)

# Normal Curve
plot(seq(0, 4, length = 1000),
     dnorm(seq(0, 4, length = 1000), mean = mean(x), sd = sd(x)),
     col = adjustcolor("tomato", alpha.f = 0.8),
     lwd = 4,
     type = "l",
     xlab = "GPA",
     ylab = "Density",
     main = " Density Plot of GPA vs Normal Distribution")

text(2, .9, col = "tomato",
     expression(paste(Nu, "(", mu(GPA), ", ", sigma(GPA), ")")))

# Distribution lines
abline(v = c(mean(x)-2*sd(x),
             mean(x)-sd(x),
             mean(x),
             mean(x)+sd(x),
             mean(x)+2*sd(x)),
       lty = 2)

# Kernel Density Plot
lines(density(x),
      col = adjustcolor("dodgerblue", alpha.f = 0.8),
      lwd = 4)
```

```
# Density plot of GRE vs a normal curve with the same parameters
x <- na.omit(numeric.df$gre)

# Normal Curve
plot(seq(130, 170, length = 1000),
     dnorm(seq(130, 170, length = 1000), mean = mean(x), sd = sd(x)),
     col = adjustcolor("tomato", alpha.f = 0.8),
     type = "l",
     lwd = 4,
     xlab = "GRE",
     ylab = "Density",
     main = " Density Plot of GRE vs Normal Distribution")

text(160, .042, col = "tomato", expression(paste(Nu, "(", mu(GRE), ", ", sigma(GRE), ")")))

# Distribution lines
abline(v = c(mean(x)-2*sd(x),
             mean(x)-sd(x),
             mean(x),
             mean(x)+sd(x),
             mean(x)+2*sd(x)),
```

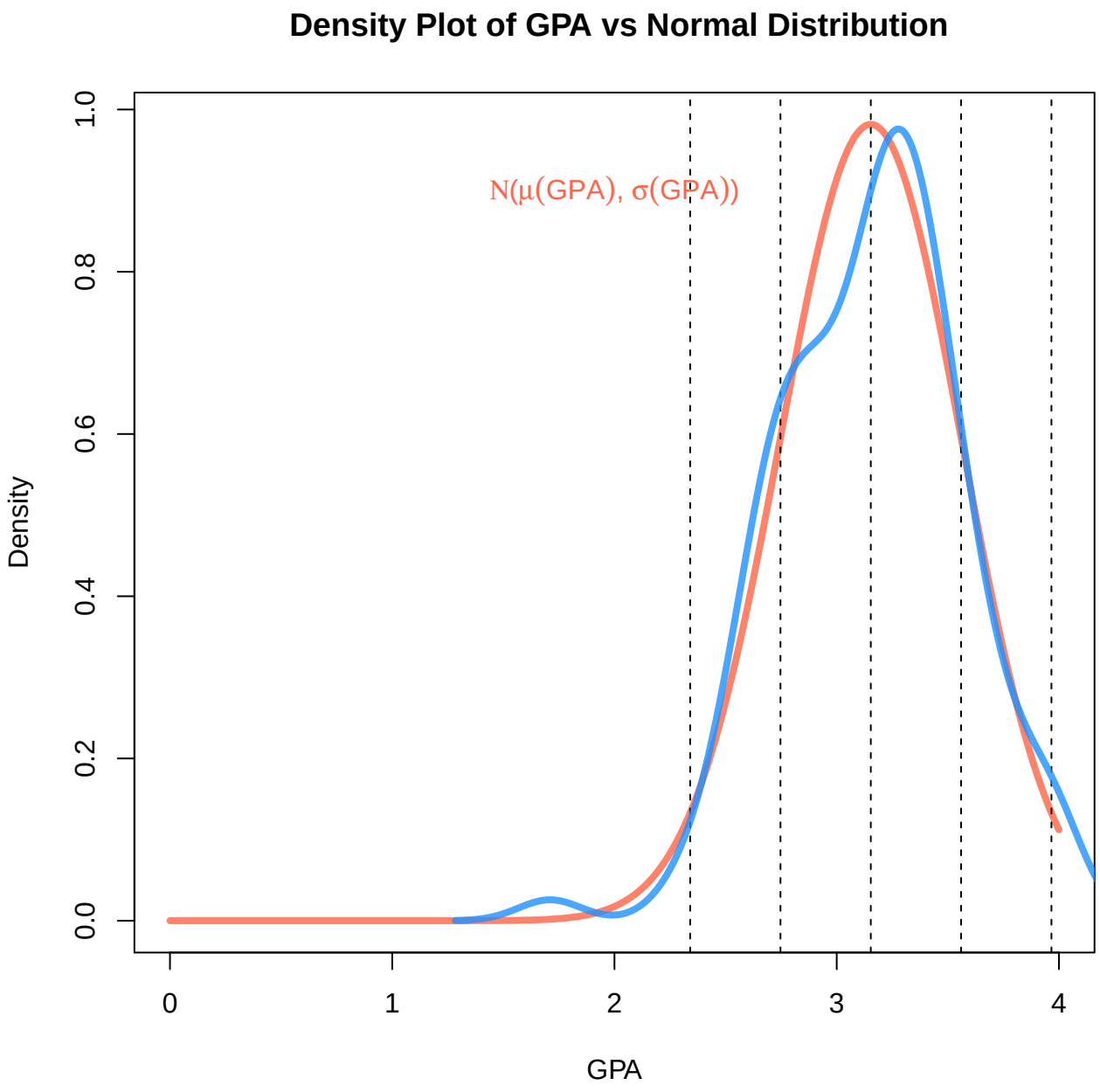
Density Plot of GPA vs Normal Distribution
N(μ(GPA), σ(GPA))
Density
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
GPA

Figure 4:

```
       lty = 2)

# Kernel Density Plot
lines(density(x),
      col = adjustcolor("dodgerblue", alpha.f = 0.8),
      lwd = 4)
```

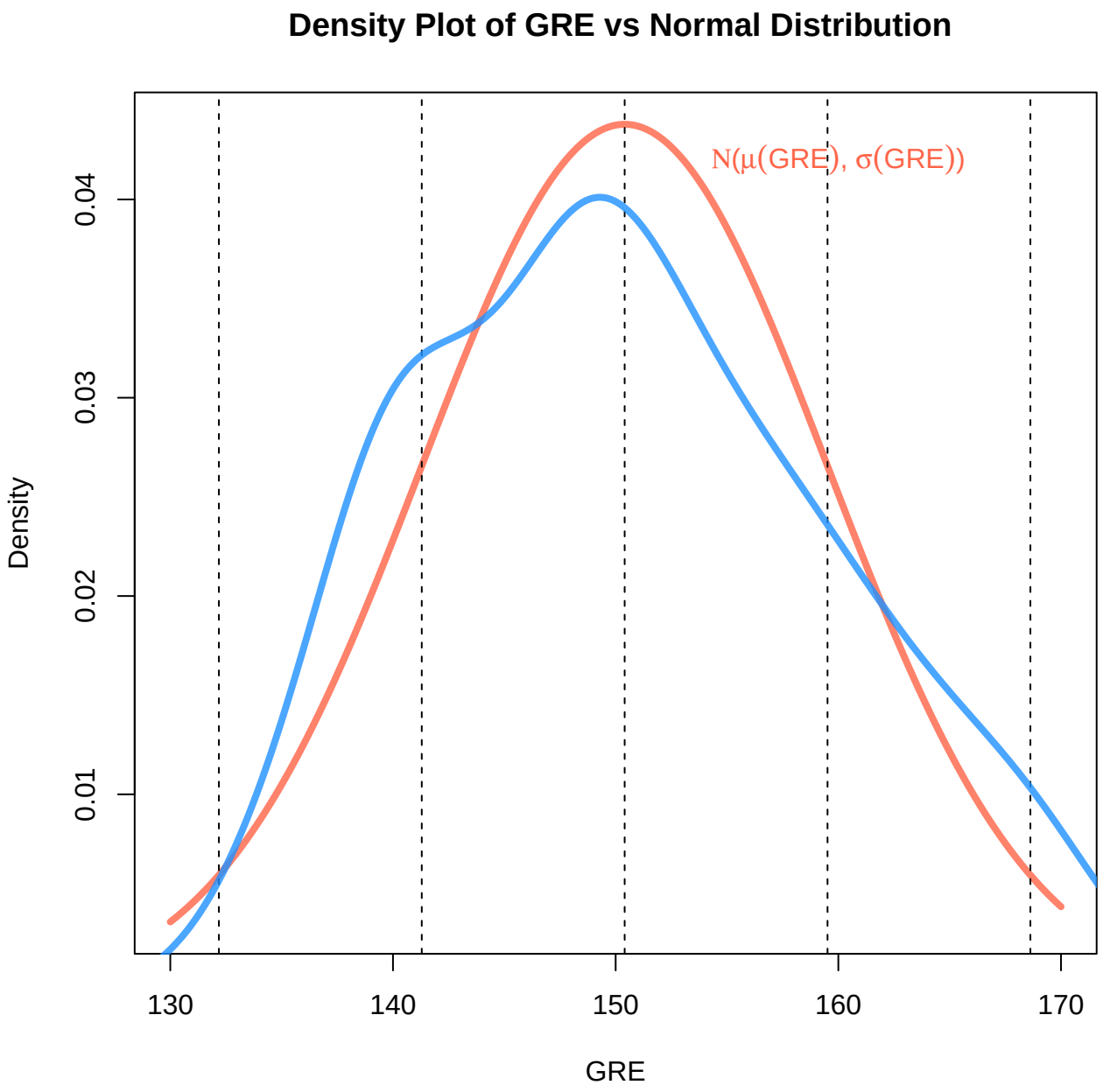

Figure 5:

```
# Histogram of GRE
hist(na.omit(numeric.df$gre), breaks = 12, col="red")
```

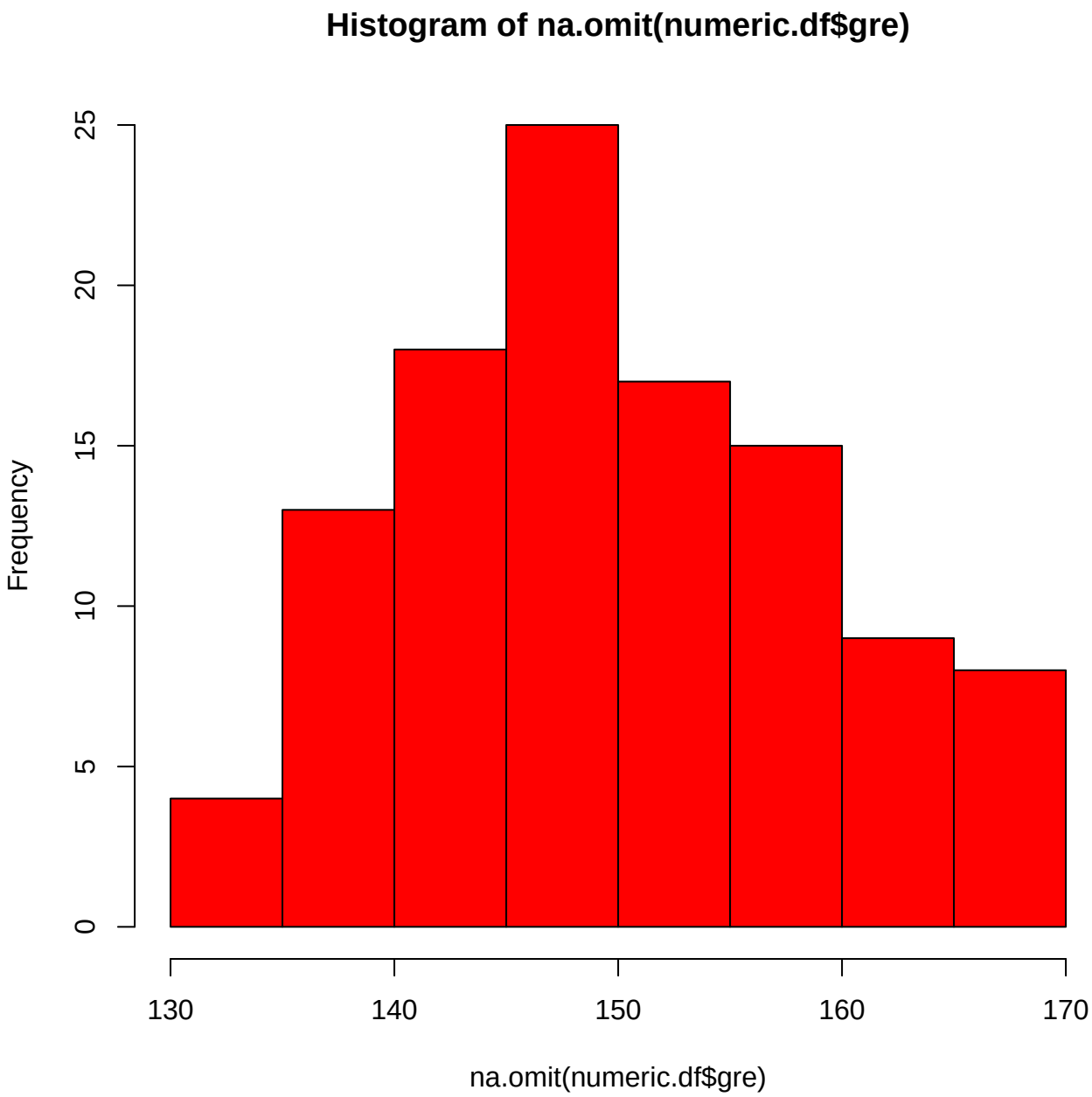
Histogram of na.omit(numeric.df$gre)
Frequency
0
5
10
15
20
25
130
140
150
160
170
na.omit(numeric.df$gre)

Figure 6:

```
# Add a Normal Curve (Thanks to Peter Dalgaard)
x <- na.omit(numeric.df$gre)
h <- hist(x, breaks= 12, col="red", xlab = "GRE", main="Histogram with Normal Curve")
xfit <- seq(min(x),max(x),length=40)
yfit <- dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col = "blue", lwd = 4)
```

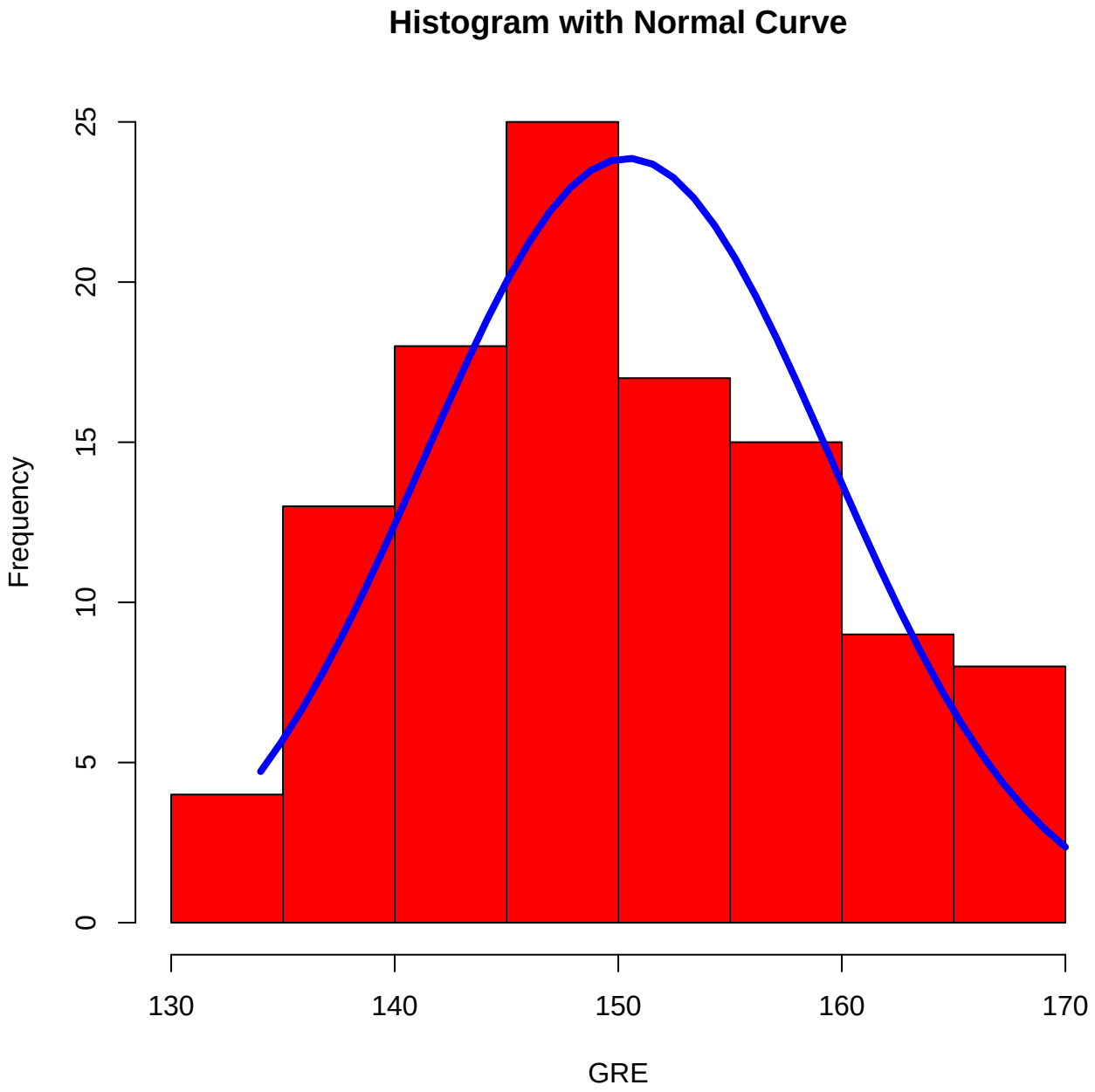

Figure 7:

```
# Histogram of GPA
hist(na.omit(numeric.df$gpa), breaks = 12, col="red")
```

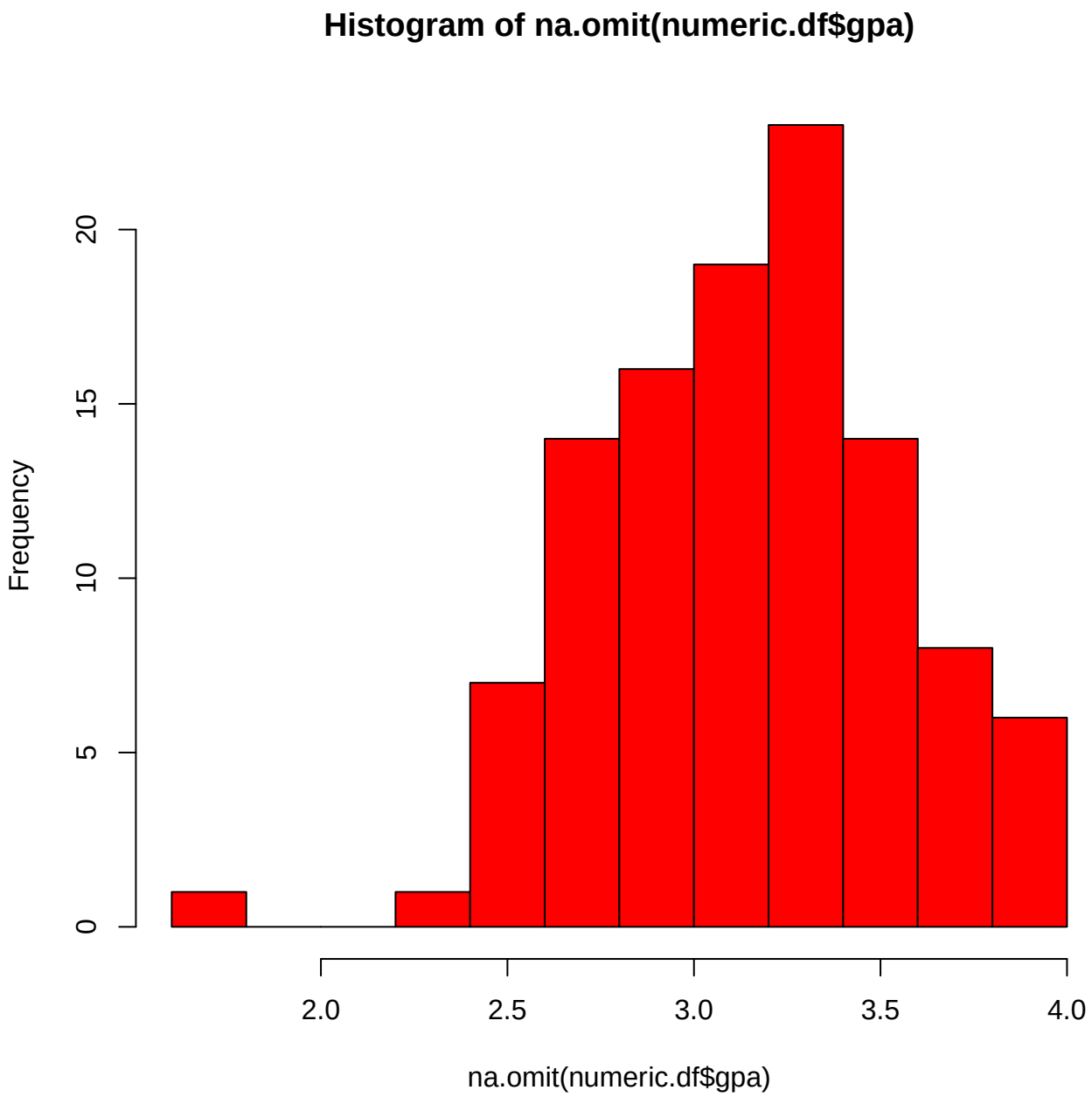
Histogram of na.omit(numeric.df$gpa)
Frequency
0
5
10
15
20
2.0
2.5
3.0
3.5
4.0
na.omit(numeric.df$gpa)

Figure 8:

```
# Add a Normal Curve (Thanks to Peter Dalgaard)
x <- na.omit(numeric.df$gpa)
h <- hist(x, breaks= 12, col="red", xlab = "GPA", main="Histogram with Normal Curve")
xfit <- seq(min(x),max(x),length=40)
yfit <- dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col = "blue", lwd = 4)
```

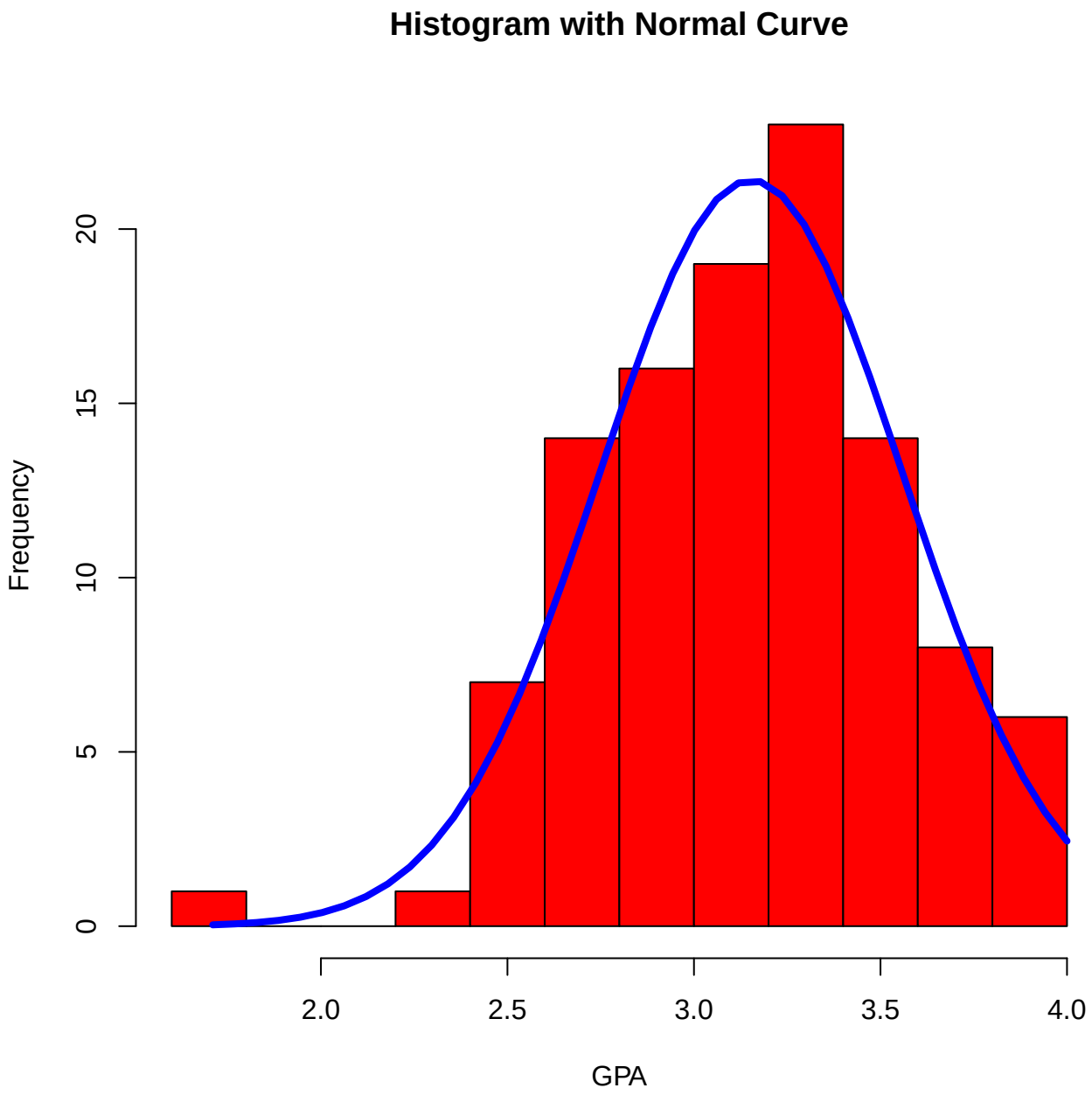

Figure 9:

4.2.3 Assumption #3: Test pairwise linearity

There should be a linear relationship between each pair of variables. If the variables are not linearly related, the power of the test is reduced. You can test for this assumption by plotting a scatterplot matrix for each group of the independent variable.

```
# Simple Scatterplot
plot(numeric.df$gre, numeric.df$gpa,
     main = "Scatterplot GRE vs GPA",
     xlab = "GRE",
     ylab = "GPA",
     pch = 1,
     cex = .1,
     col = "red")

# Labels
text(numeric.df$gre, numeric.df$gpa,
     labels = numeric.df$id,
     cex = .8,
     col = "blue")

# Add fit lines
abline(lm(numeric.df$gpa ~ numeric.df$gre), col = "red", lwd = 2) # regression line (y~x)
```

4.2.4 Assumption #4: Test Bivariate Outliers

Pearson's r is sensitive to outliers, which can have a very large effect on the line of best fit and the Pearson correlation coefficient, leading to very difficult conclusions regarding your data. Therefore, it is best if there are no outliers or they are kept to a minimum. If you cannot justify removing the data point(s), you can run a non-parametric test such as Spearman's rank-order correlation or Kendall's Tau Correlation instead, which are much less sensitive to outliers.

For a linear model, p-values reported use the t distribution with degrees of freedom one less than the residual df for the model. For a generalized linear model, p-values are based on the standard-normal distribution. The Bonferroni adjustment multiplies the usual two-sided p-value by the number of observations. The lm method works for glm objects. To show all of the observations set cutoff = Inf and n.max = Inf

```
fit <- lm(gpa ~ gre, data = numeric.df)

# Assessing Outliers
# There aren't any
car::outlierTest(fit, n.max = 5, cutoff = Inf) # Bonferonni p-value for most extreme obs
```

```
##      rstudent unadjusted p-value Bonferonni p
## 109 -3.767997         0.00027201     0.029377
## 68   2.506680         0.01372000           NA
## 110 -2.188884         0.03082000           NA
## 40   2.076472         0.04029200           NA
## 31   2.012982         0.04667600           NA
```

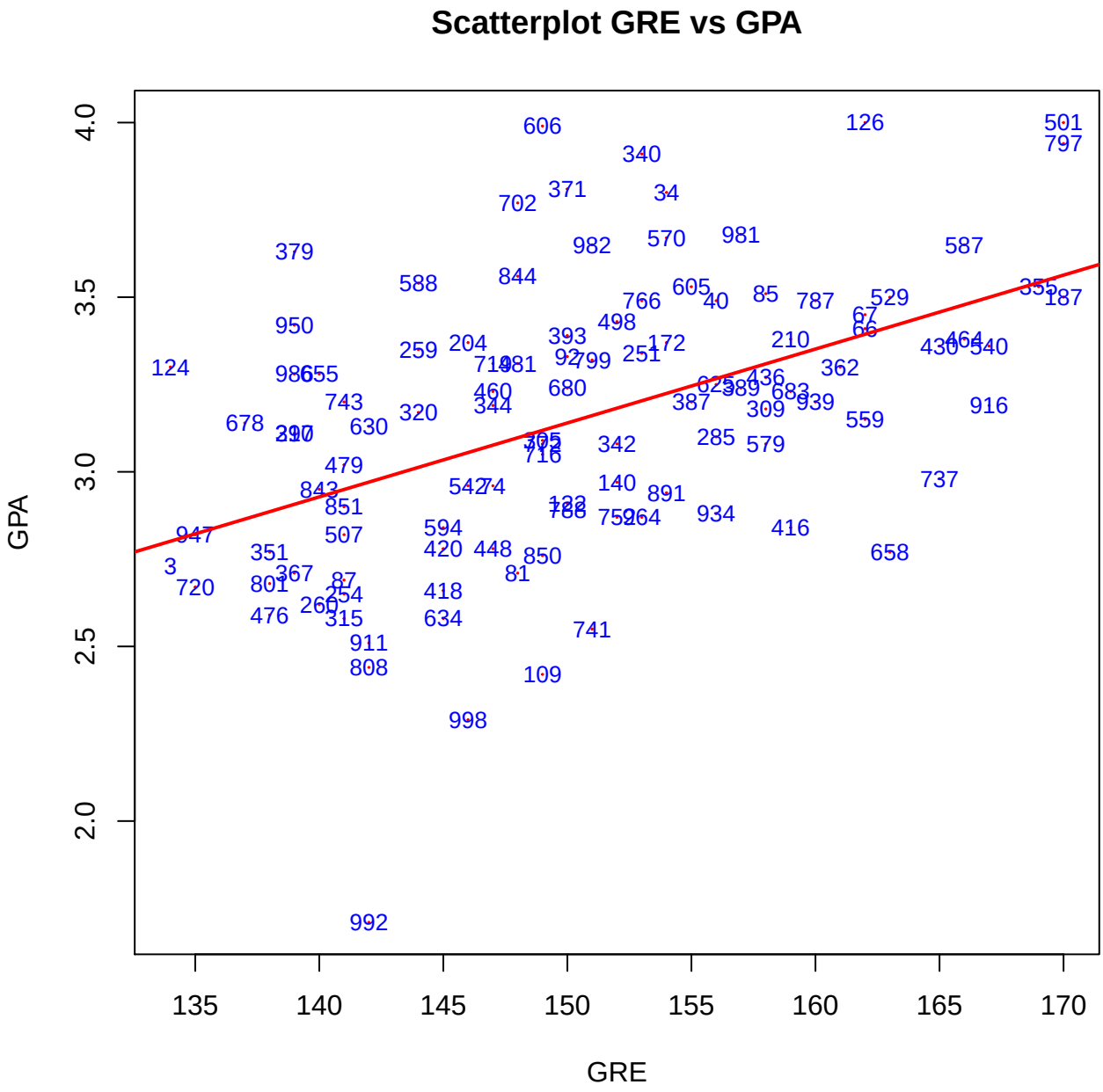
Scatterplot GRE vs GPA
GPA
GRE
4.0
3.5
3.0
2.5
2.0
135
140
145
150
155
160
165
170

Figure 10:

```
# leverage plots

# id.methods:

#Label all observations with Pearson residuals greater than 2 in absolute value
# which(abs(residuals(fit, type = "pearson")) > 2)

# label largest values of Cook's distance
# cooks.distance(fit)

# select cases according to their Mahalanobis distance from (mean(x), mean(y))
# "mahal"

car::leveragePlots(fit,
                   main = "Regression Leverage Plots",
                   id.method = cooks.distance(fit),
                   id.n = 10,
                   id.col = "red",
                   labels = numeric.df$id)
```

Cook's Distance is a commonly used estimate of the influence of a data point when performing least squares regression analysis. Cook's distance measures the effect of deleting a given observation. Data points with large residuals (outliers) and/or high leverage may distort the outcome and accuracy of a regression. Points with a large Cook's distance are considered to merit closer examination in the analysis.

There are different opinions regarding what cut-off values to use for spotting highly influential points. A simple operational guideline of $D_i > 1$ has been suggested [1]. Others have indicated that $D_i > 4/n$, where n is the number of observations, might be used [2].

[1] Cook, R. Dennis; and Weisberg, Sanford (1982); Residuals and influence in regression, New York, NY: Chapman & Hall [2] Bollen, Kenneth A.; and Jackman, Robert W. (1990); Regression diagnostics: An expository treatment of outliers and influential cases, in Fox, John; and Long, J. Scott (eds.); Modern Methods of Data Analysis (pp. 257-91). Newbury Park, CA: Sage

```
# Cook's D plot
# identify D values > 4/(n-k-1), where k = 1
cutoff <- 4/((nrow(numeric.df) - 2))

layout(matrix(c(1,2,3,4),2,2)) #divide graph
plot(fit, which = 1, cook.levels = cutoff)
plot(fit, which = 2, cook.levels = cutoff)
plot(fit, which = 3, cook.levels = cutoff)
plot(fit, which = 4, cook.levels = cutoff)
```

```
dim(na.omit(numeric.df))
```

```
## [1] 102  11
```

```
# Influence Plot
car::influencePlot(fit,
                   id.method = "noteworthy",
                   main = "Influence Plot \n achievement ~ gpa",
                   col = "red",
```

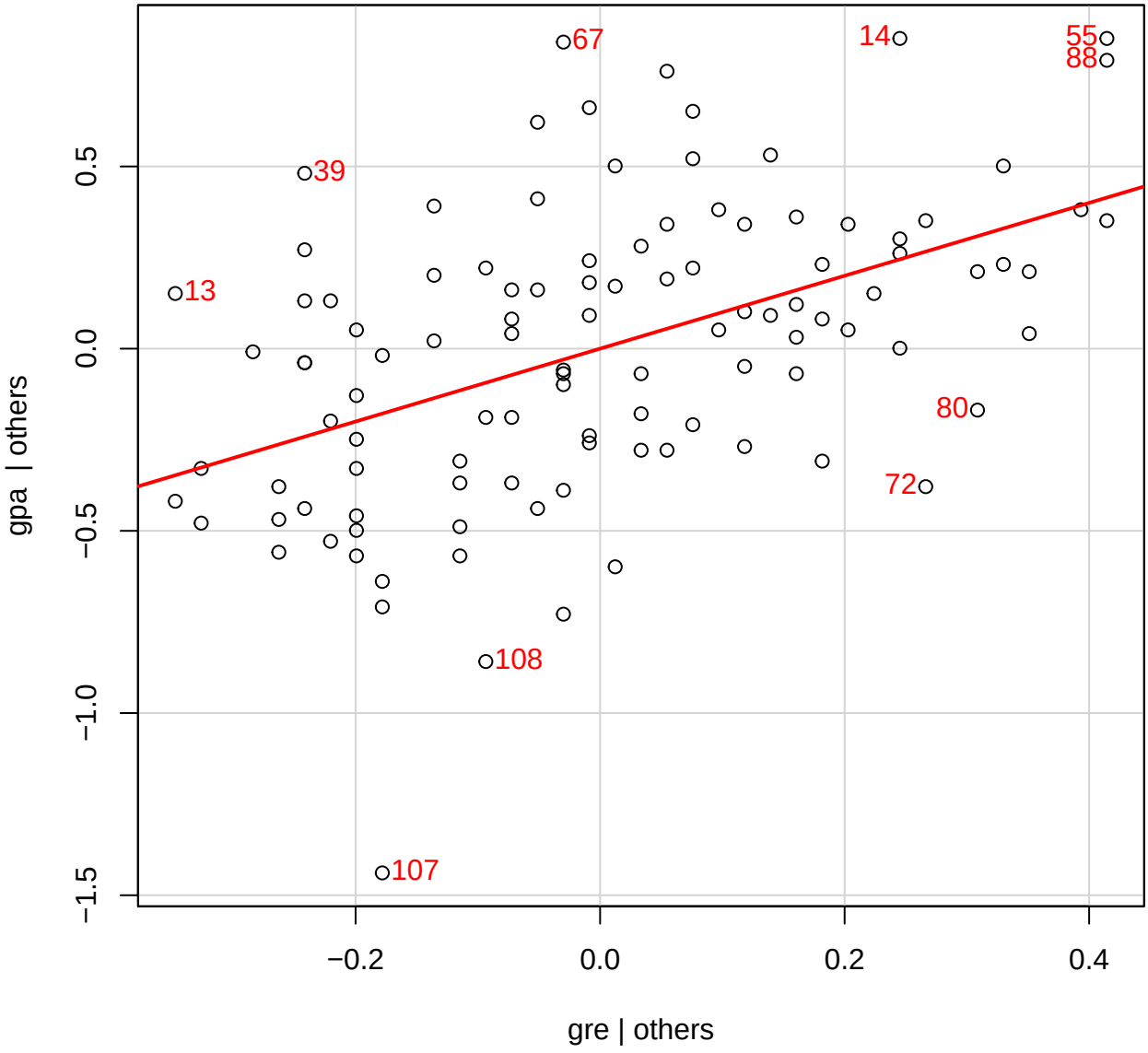
67
14
55
88
39
13
80
72
108
107
0.5
0.0
−0.5
−1.0
−1.5
gpa | others
−0.2
0.0
0.2
0.4
gre | others

Figure 11:

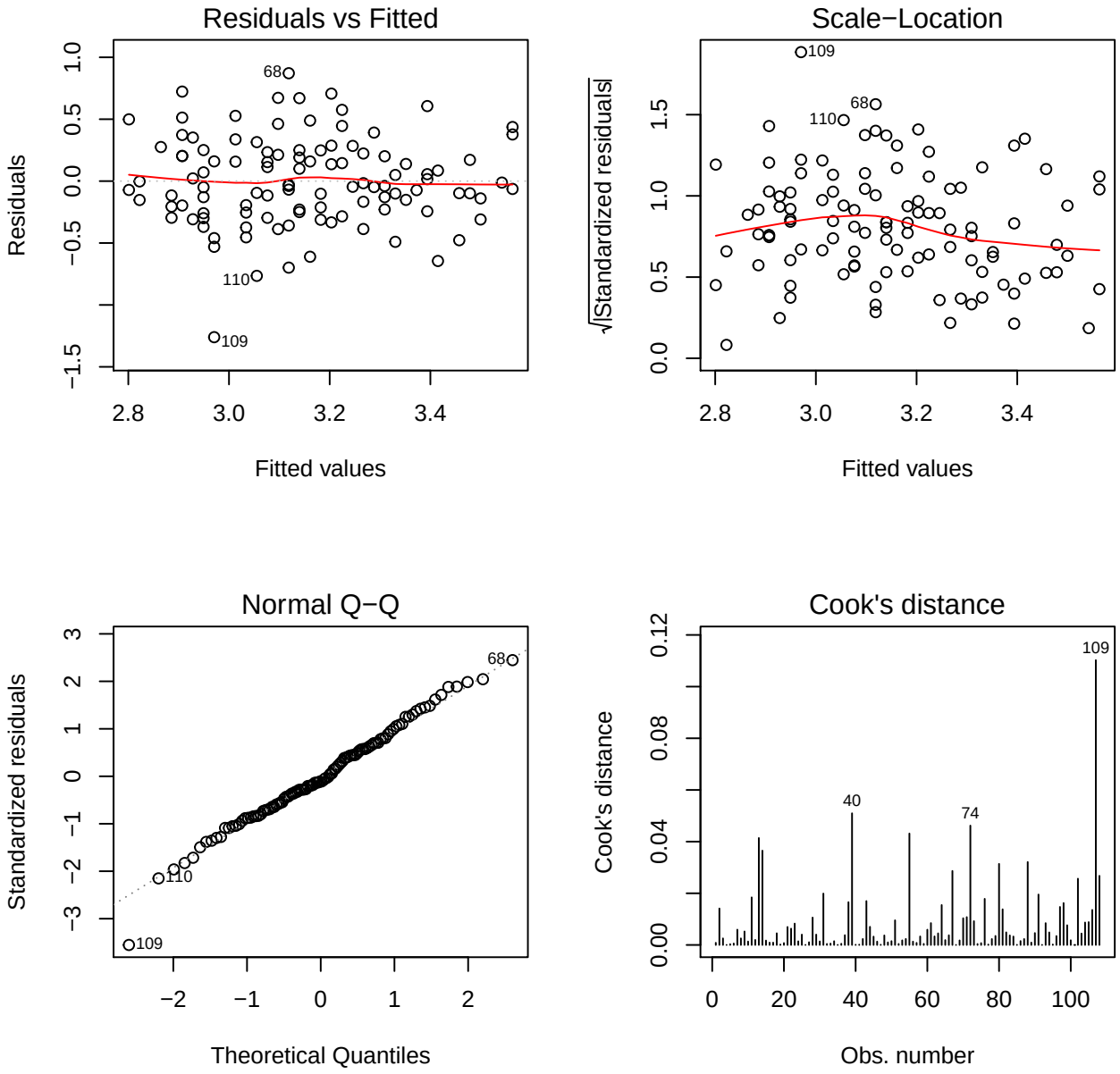
Residuals vs Fitted
Residuals
Fitted values
68
110
109
Scale−Location
√|Standardized residuals|
Fitted values
109
68
110
Normal Q−Q
Standardized residuals
Theoretical Quantiles
68
110
109
Cook's distance
Cook's distance
Obs. number
109
40
74

Figure 12:

```
                  col.sub = "red",
                  id.col = "blue",
                  id.n = 5,
                  id.cex = .7,
                  sub = "Circle size is proportial to Cook's Distance")
```

```
##         StudRes         Hat       CookD
## 13   1.42906950 0.039330608 0.203466092
## 17  -0.18060254 0.052050628 0.030062156
## 31   2.01298172 0.010003895 0.141068022
## 35  -0.03433296 0.047792025 0.005464669
## 40   2.07647182 0.023802513 0.225773594
## 56   1.25710795 0.052050628 0.207726743
## 59  -0.39688413 0.039944293 0.057472576
## 68   2.50667994 0.009483192 0.169265285
## 74  -1.84807676 0.026926729 0.214947057
## 90   1.08234160 0.052050628 0.179192075
## 109 -3.76799686 0.017163555 0.332030018
## 110 -2.18888445 0.011435827 0.163571761
```

4.2.5 Quantile-Quantile Plots

```
# Quantile-Quantile Plots
car::qqPlot(fit, main="QQ Plot", xlab = deparse(substitute(GRE)),
            ylab = deparse(substitute(GPA))) #qq plot for studentized resid
```

```
# Normal quantile-quantile plot
qqnorm(numeric.df$gpa, main = "GPA \n Normal Q-Q Plot", pch = 16)
qqline(numeric.df$gpa, col = "red", lwd = 2)
```

```
# Normal quantile-quantile plot
qqnorm(numeric.df$gre, main = "GPA \n Normal Q-Q Plot", pch = 16)
qqline(numeric.df$gre, col = "red", lwd = 2)
```

4.2.6 Assumption #5: Test Bivariate Normality

In order to assess the statistical significance of the Pearson correlation, the data need to have bivariate normality.

```
mvnormtest::mshapiro.test(t(na.omit(numeric.df$gre, numeric.df$gpa)))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.97309, p-value = 0.02618
```

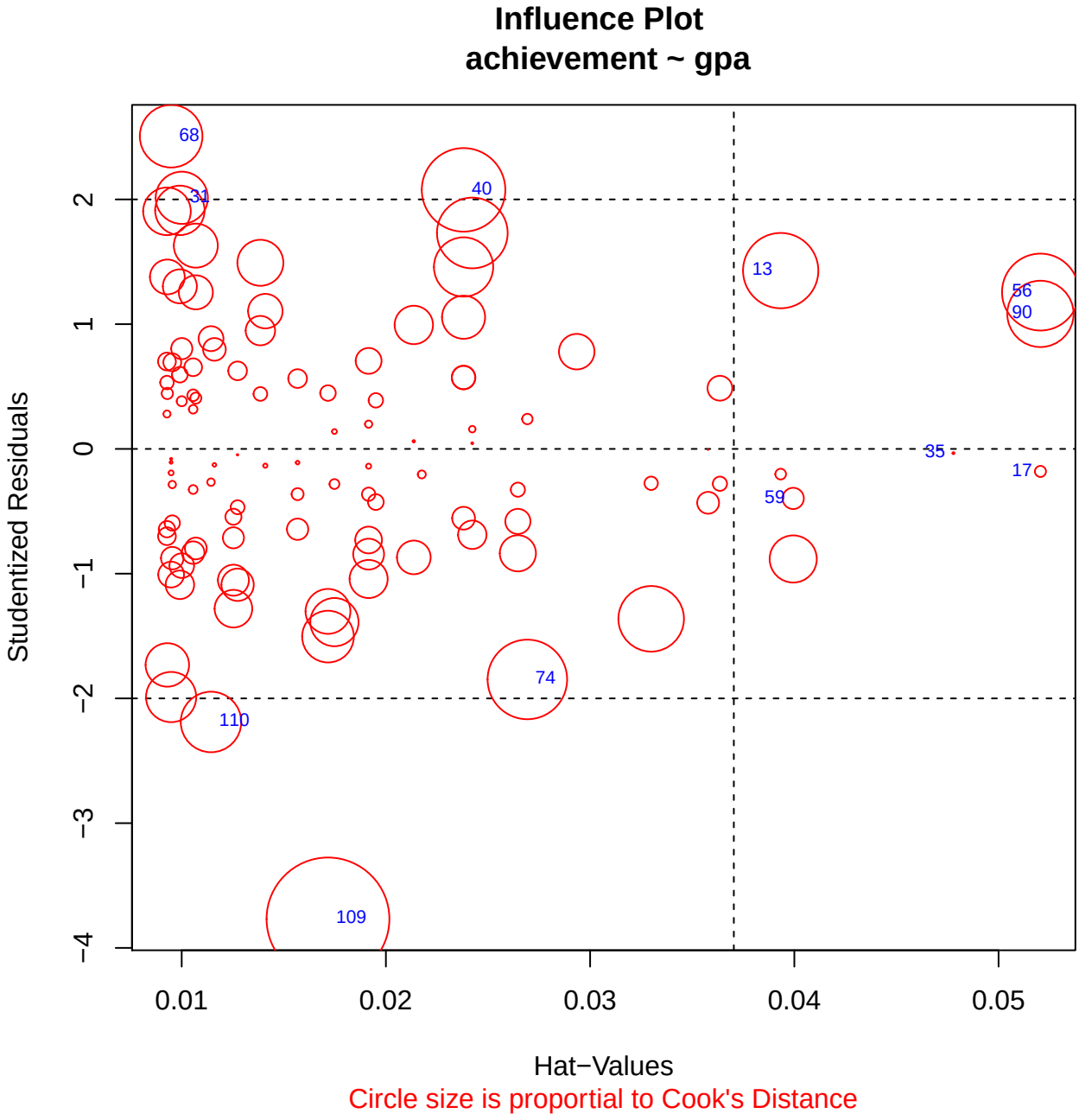
Influence Plot
achievement ~ gpa
Studentized Residuals
Hat−Values
Circle size is proportial to Cook's Distance
68
31
40
13
56
90
35
17
59
74
110
109
2
1
0
−1
−2
−3
−4
0.01
0.02
0.03
0.04
0.05

Figure 13:

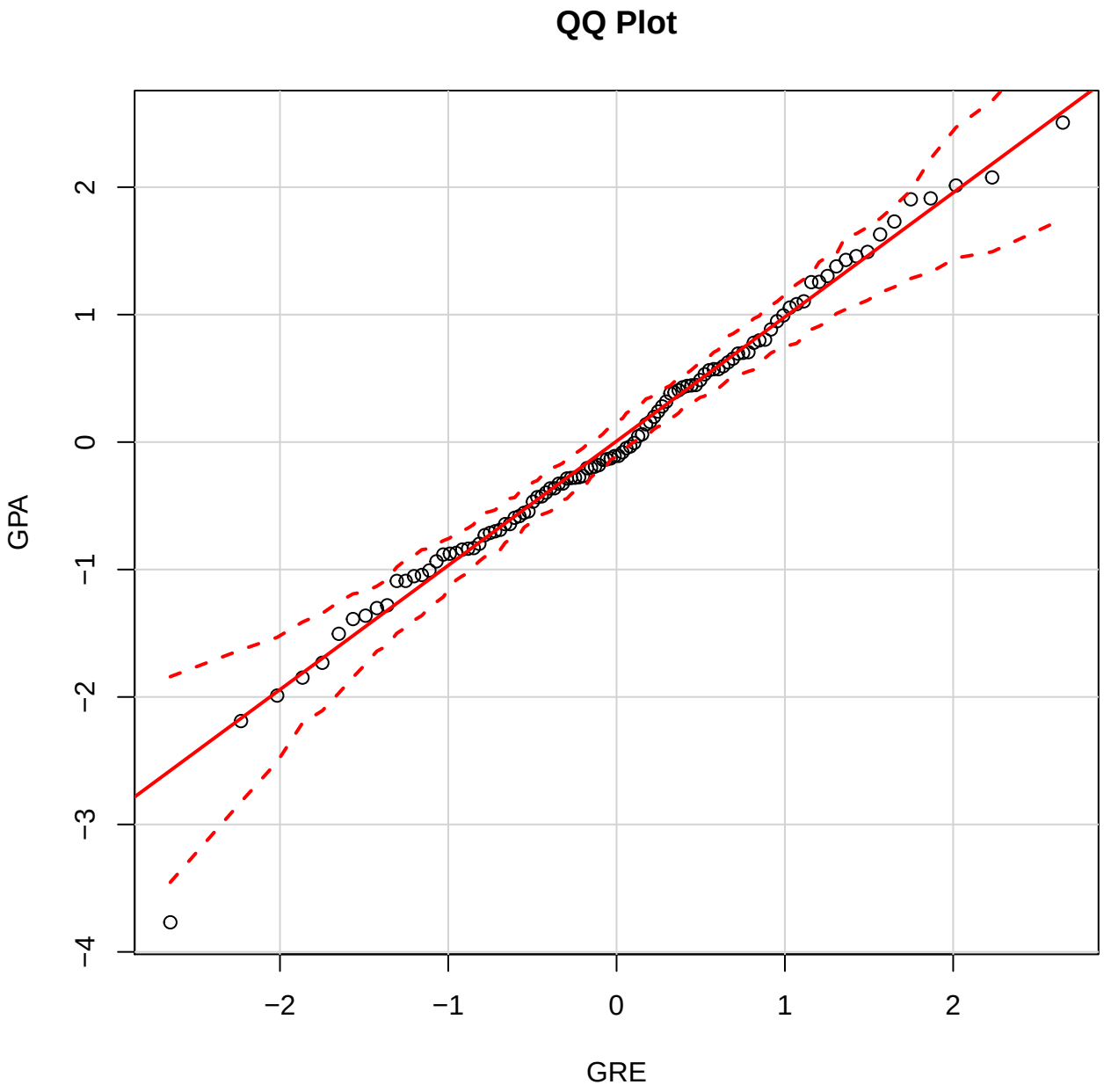

Figure 14:

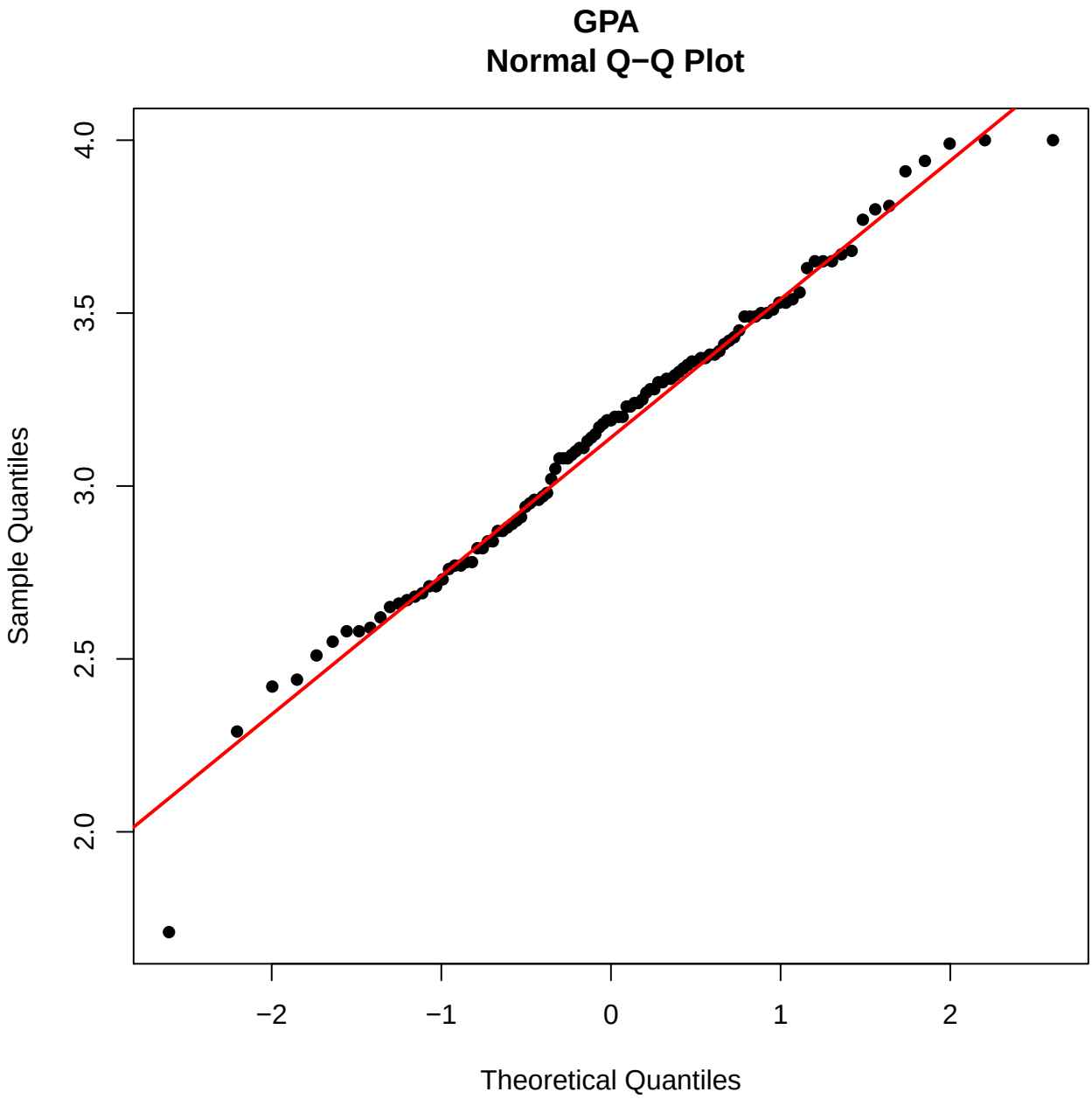
GPA
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
2.0
2.5
3.0
3.5
4.0
−2
−1
0
1
2

Figure 15:

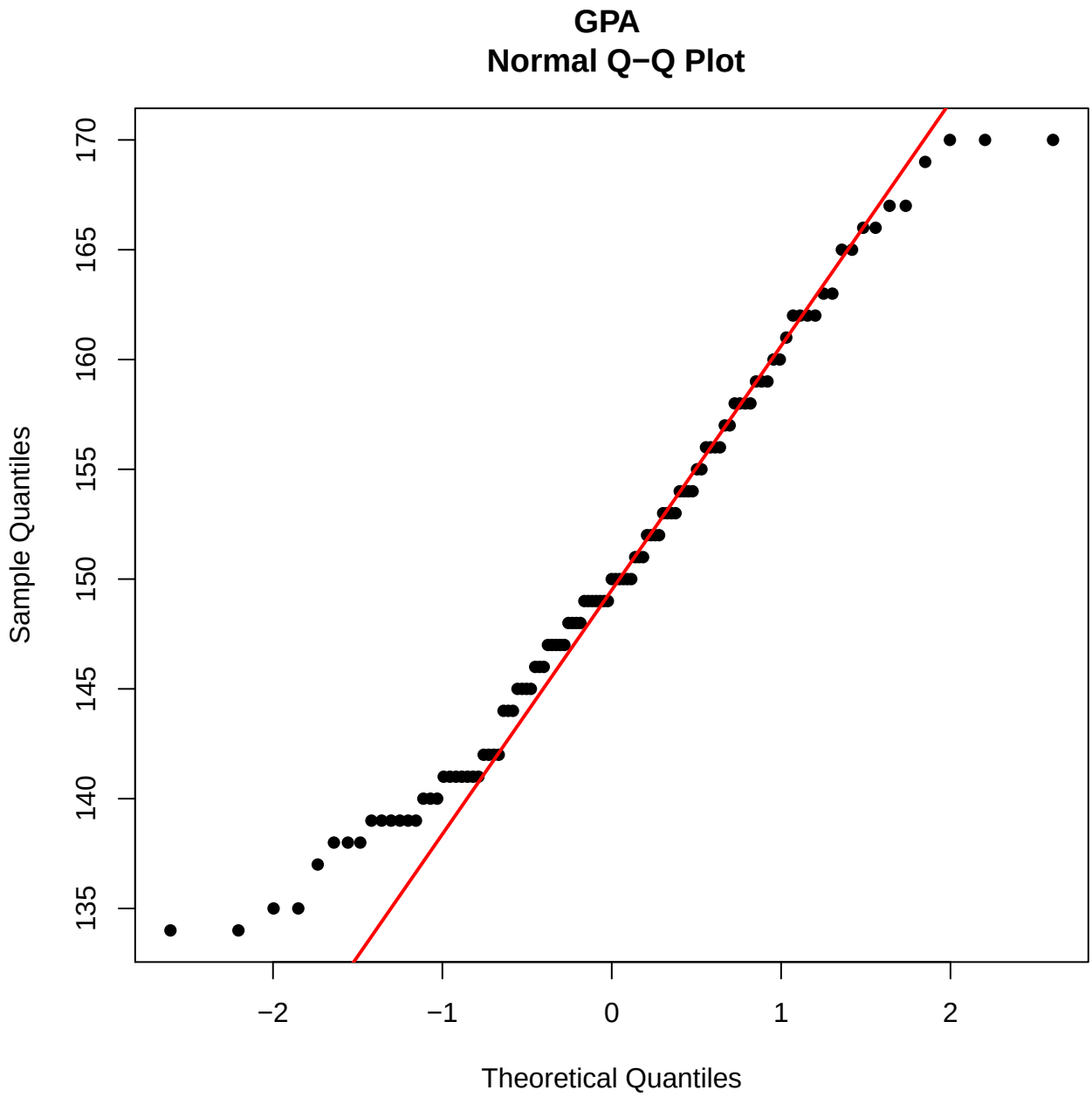
GPA
Normal Q−Q Plot
Sample Quantiles
135
140
145
150
155
160
165
170
−2
−1
0
1
2
Theoretical Quantiles

Figure 16:

The Histogram of the Residual can be used to check whether the variance is normally distributed. A symmetric bell-shaped histogram which is evenly distributed around zero indicates that the normality assumption is likely to be true. If the histogram indicates that random error is not normally distributed, it suggests that the model's underlying assumptions may have been violated.

```
# distribution of studentized residuals
sresid1 <- MASS::studres(fit)

# studentized residuals histogram
hist(sresid1,
     freq = FALSE,
     main = "Distribution of Studentized Residuals",
     col = "dodgerblue",
     border = "white",
     breaks= 24,
     xlim = c(-3, 3))

# normal curve
xfit <- seq(-3, 3, length = 100)
yfit <- dnorm(xfit)
lines(xfit, yfit, col = "tomato", lwd = 5)

# gridlines
grid(nx = NULL,
     ny = NULL,
     col = "lightgray",
     lty = "dotted",
     lwd = par("lwd"),
     equilogs = TRUE)
```

4.2.7 Assumption #6: Test Homoscedasticity

Computes a score test of the hypothesis of constant error variance against the alternative that the error variance changes with the level of the response (fitted values), or with a linear combination of predictors.

```
# Evaluate homoscedasticity
# non-constant error variance test
car::ncvTest(fit)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.733984    Df = 1     p = 0.187903
```

```
# plot studentized residuals vs. fitted values
car::spreadLevelPlot(fit)
```

```
##
## Suggested power transformation:  3.271271
```

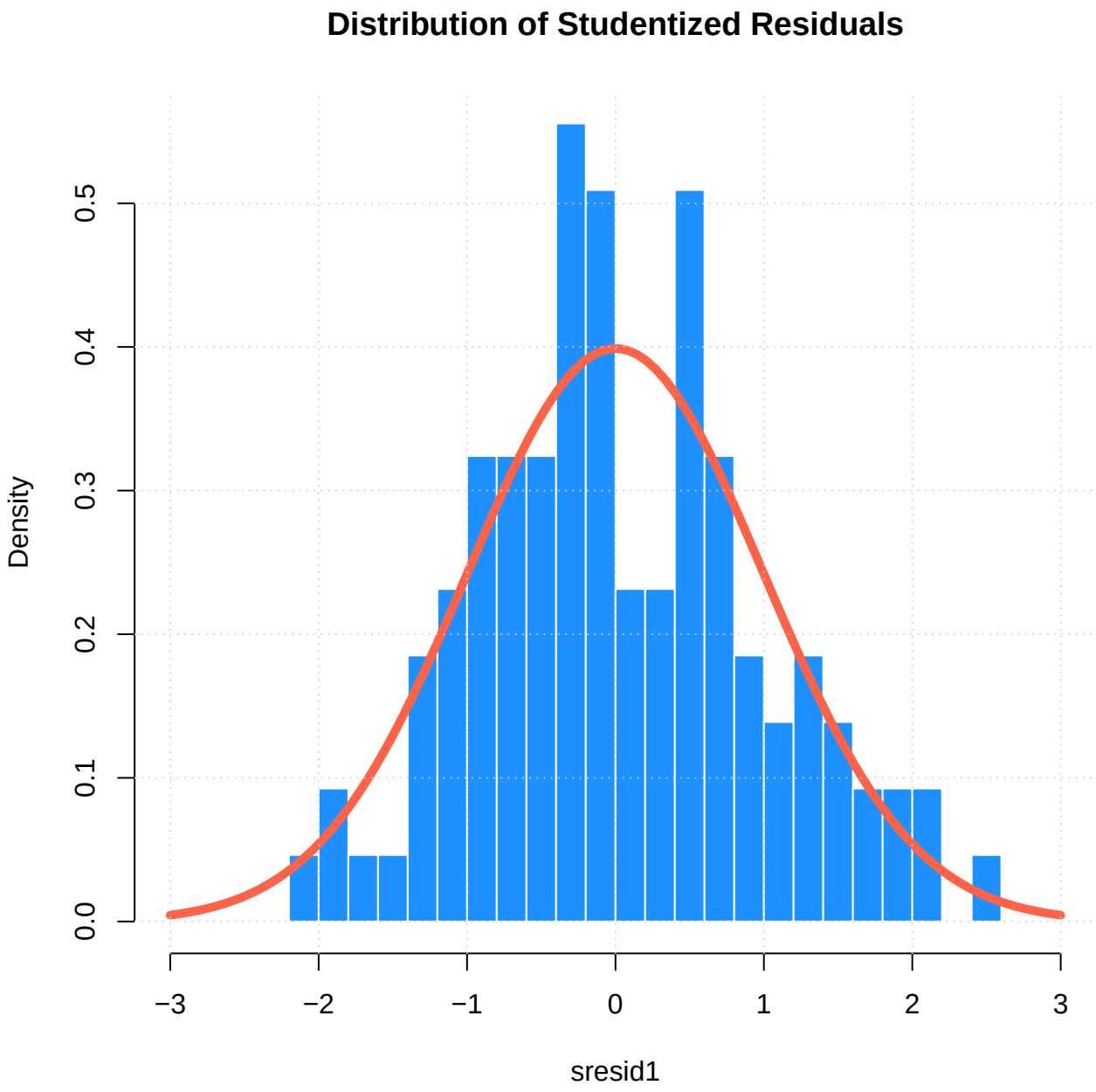
Distribution of Studentized Residuals
Density
0.0
0.1
0.2
0.3
0.4
0.5
−3
−2
−1
0
1
2
3
sresid1

Figure 17:

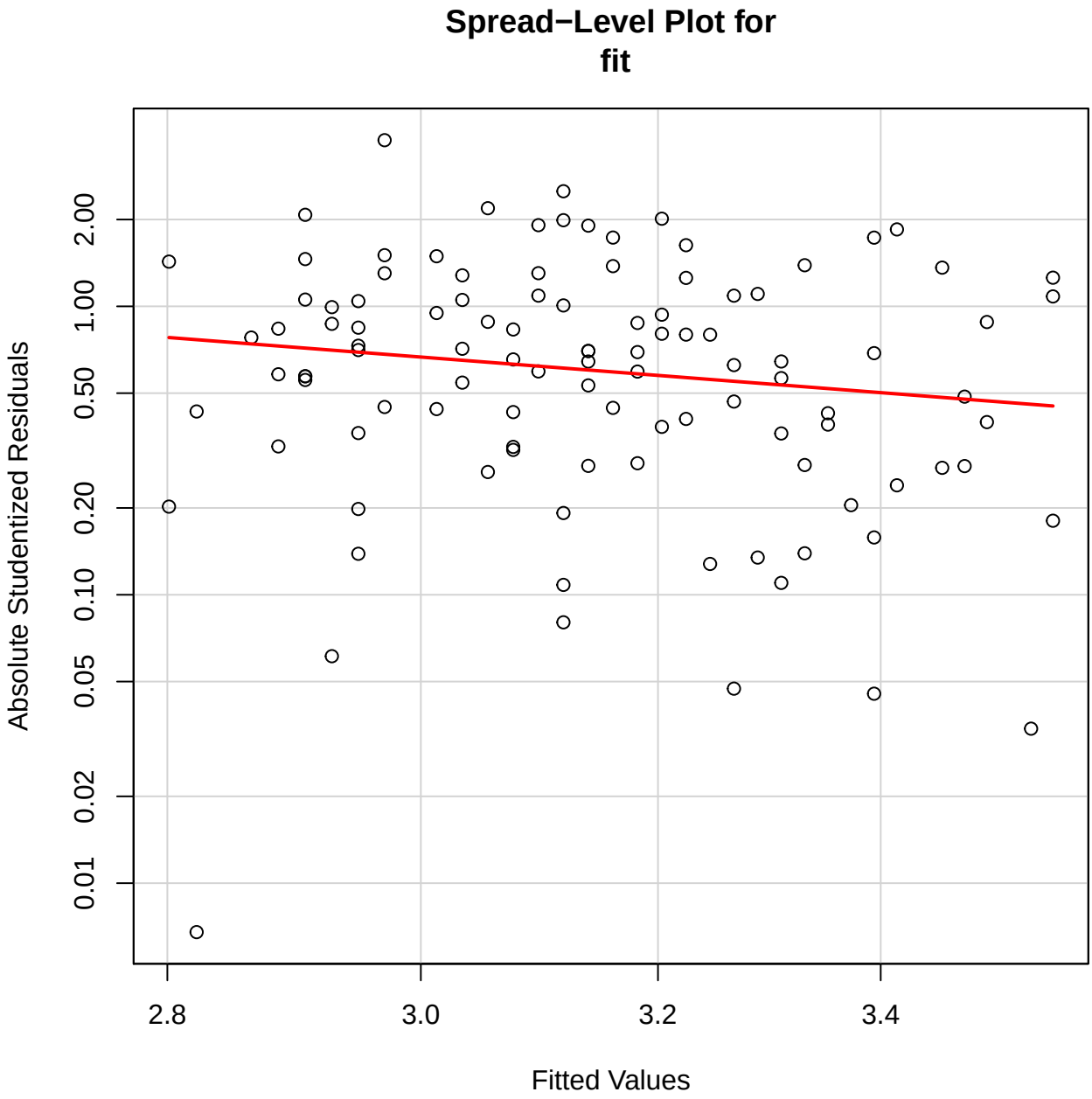
Spread−Level Plot for
fit
Absolute Studentized Residuals
0.01
0.02
0.05
0.10
0.20
0.50
1.00
2.00
2.8
3.0
3.2
3.4
Fitted Values

Figure 18:

4.3 Pearson product-moment correlation coefficient

rcorr omputes a matrix of Pearson's r or Spearman's rho rank correlation coefficients for all possible pairs of columns of a matrix. Missing values are deleted in pairs rather than deleting all rows of x having any missing variables.

```
# Correlations with significance levels
Hmisc::rcorr(data.matrix(numeric.df[, c("gre", "gpa", "anxiety", "achievement")]),
             type = "pearson")
```

```
##                gre   gpa anxiety achievement
## gre           1.00  0.48   -0.43        0.04
## gpa           0.48  1.00   -0.43        0.20
## anxiety      -0.43 -0.43    1.00       -0.02
## achievement   0.04  0.20   -0.02        1.00
##
## n
##              gre gpa anxiety achievement
## gre          109 108     109         108
## gpa          108 109     109         108
## anxiety      109 109     110         109
## achievement 108 108     109         109
##
## P
##              gre    gpa    anxiety achievement
## gre                 0.0000 0.0000  0.6648
## gpa          0.0000        0.0000  0.0362
## anxiety      0.0000 0.0000         0.8269
## achievement 0.6648 0.0362 0.8269
```

```
# type can be pearson or spearman
```

```
# Correlation and Covariance
cor(numeric.df$gpa, numeric.df$gre,
    use = "complete.obs",
    method = "pearson")
```

```
## [1] 0.4777923
```

```
cov(numeric.df$gpa, numeric.df$gre,
    use = "complete.obs")
```

```
## [1] 1.772882
```

Correlation Matrix

```
# histograms on the diagonal
panel.hist.density <-
  function(x,...)
  {
    usr <- par("usr"); on.exit(par(usr))
```

```
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "dodgerblue")
    tryd <- try( d <- density(x, na.rm = TRUE, bw = "nrd", adjust = 1.2), silent = TRUE)
    if (class(tryd) != "try-error")
    {
      d$y <- d$y/max(d$y)
      lines(d, lwd = 2, col = "tomato")
    }
  }
```

```
# scatter plot with tolerance ellipsoids and lowess line on the lower-triangle
panel.smooth <-
  function(x, y,
           col = par("col"),
           bg = NA,
           pch = 21,
           cex = .5,
           col.smooth = "red",
           span = 2/3,
           iter = 3, ...){
    require(cluster)

    points(x, y,
           pch = pch,
           col = col,
           bg = bg,
           cex = cex)

    ok <- is.finite(x) & is.finite(y)
    if (any(ok))
      lines(stats::lowess(x[ok], y[ok], f = span, iter = iter),
            col = col.smooth,
            lwd = 2, ...)

    # classical and robust cov.matrix ellipsoids
    X <- cbind(x,y)
    C.ls <- cov(X)
    m.ls <- colMeans(X)

    # calculates a 99% ellipsoid
    d2.99 <- qchisq(0.99, df = 2)

    # classical cov.matrix ellipsoids
    lines(ellipsoidPoints(C.ls, d2.99, loc = m.ls),
          lwd = 2,
          lty = 3,
          col = "purple")

    if (require(MASS)) {
      Cxy <- cov.rob(cbind(x,y))
```

```
      # robust cov.matrix ellipsoids
      lines(ellipsoidPoints(Cxy$cov, d2 = d2.99, loc = Cxy$center),
            lty = 3,
            lwd = 2,
            col = "brown")
    }
  }
```

```
# correlations with significance on the upper panel
panel.cor <-
  function(x, y,
           method = "pearson",
           digits = 3,
           cex.cor = 1.2,
           no.col = FALSE) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- cor(x, y, method = method)
    ra <- cor.test(x, y, method = method)$p.value
    txt <- format(c(r, 0.123456789), digits = digits)[1]

    # mark significant correlations
    prefix <- ""
    if (ra <= 0.1) prefix <- "."
    if (ra <= 0.05) prefix <- "*"
    if (ra <= 0.01) prefix <- "**"
    if (ra <= 0.001) prefix <- "***"
    if (no.col)
    {
      color <- "red"
      if (r < 0) {if (ra <= 0.001) sig <- 4 else sig <- 3}
      else {if (ra <= 0.001) sig <- 2 else sig <- 1}
    }
    else
    {
      sig <- 1
      if (ra <= 0.001) sig <- 2
      color <- 2
      if (r < 0) color <- 4
    }
    txt <- paste(txt, prefix, sep = "\n")
    if (missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r, col = "maroon")
  }
```

```
# Correlation Matrix
pairs(~gpa + gre + anxiety + achievement,
      data = numeric.df,
      lower.panel = panel.smooth,
      upper.panel = panel.cor,
      diag.panel = panel.hist.density,
      main = "Scatterplot Matrix with Correlations",
```

```
      na.action = stats::na.omit
      #bg = c("yellow", "green3", "blue")[unclass(numeric.df$rank)]
      )
```

```
## Loading required package: cluster
## Loading required package: MASS
```

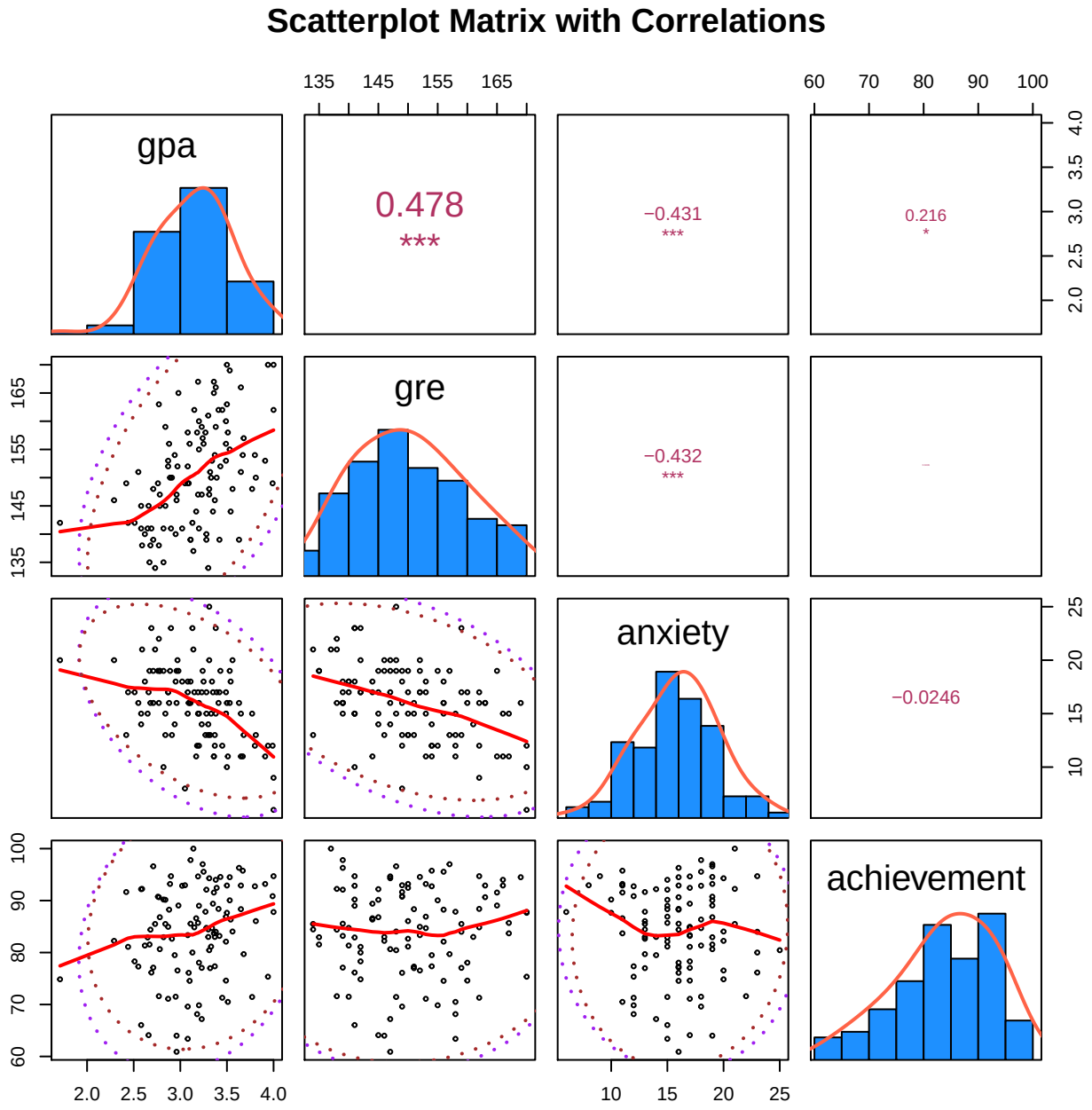

Figure 19:

4.3.1 Spearman's rank correlation coefficient

Spearman's rank correlation coefficient is a nonparametric measure of statistical dependence between two variables. It assesses how well the relationship between two variables can be described using a monotonic

function. If there are no repeated data values, a perfect Spearman correlation of +1 or -1 occurs when each of the variables is a perfect monotone function of the other.

Spearman's coefficient, like any correlation calculation, is appropriate for both continuous and discrete variables, including ordinal variables. Spearman's ρ and Kendall's τ can be formulated as special cases of a more a general correlation coefficient.

```
# Correlation, Variance and Covariance (Matrices)
cor(data.matrix(numeric.df[,c("gpa", "gre", "anxiety", "achievement", "rank")]),
    use = "complete.obs",
    method = "spearman")
```

```
##                    gpa         gre      anxiety  achievement       rank
## gpa          1.0000000  0.48865167 -0.407807373  0.227602052 -0.1803416
## gre          0.4886517  1.00000000 -0.417146505  0.037079619 -0.1497408
## anxiety     -0.4078074 -0.41714650  1.000000000 -0.004813211  0.1028564
## achievement  0.2276021  0.03707962 -0.004813211  1.000000000 -0.1366120
## rank        -0.1803416 -0.14974082  0.102856445 -0.136611991  1.0000000
```

4.4 Bivariate Linear Regression

Run relevant descriptive statistics (e.g., means, standard deviations, correlations) and graphs (e.g., scatterplots, matrix of scatterplots) for these bivariate analyses. Screen the data for bivariate outliers (e.g., scatterplot). Test all appropriate assumptions for the analyses (e.g., normality, linearity).

Research Question: Can Statistics Anxiety (SA) be used to predict statistics Achievement (ACH) among graduate students?

A bivariate regression was performed to evaluate how well statistics achievement could be predicted from levels of statistics anxiety. Preliminary data screening indicated that the scores on achievement and statistics anxiety were reasonably normally distributed. A scatter plot indicated that the relation between X and Y was positive and reasonably linear and there were no bivariate outliers. The correlation between achievement and anxiety was not statistically significant, r(109) = -0.02, p = .876.} The regression equation for predicting achievement from statistics anxiety was found to be Y' = 84.519 - 0.039X. The adjusted r^2 for this equation was -0.009; that is, 0.9% of the variance in statistics achievement was predictable from scores of statistics anxiety. This is a very weak relationship; increases in statistics anxiety tended not to be associated with increases in statistics achievement.

4.4.1 Assumptions of bivariate linear regression

Preliminary data screening for bivariate regression is the same as for Pearson correlation. Preliminary examination of the univariate distributions of X and Y tells the researcher whether these distributions are reasonably normal, whether there are univariate outliers, and whether the range of scores on both X and Y is wide enough to avoid problems with restricted range. Preliminary examination of the scatter plot enables the researcher to assess whether the X, Y relation is linear, whether the variance of Y scores is fairly uniform across levels of X, and whether there are bivariate outliers that require attention. In applications of Pearson's r, both X and Y variables could be either quantitative or dichotomous. For bivariate regression, the Y (dependent) variable is assumed to be quantitative; the X (predictor) variable can be either quantitative or dichotomous.

Warner, Rebecca (Becky) M. (Margaret) (2012-04-10). Applied Statistics: From Bivariate Through Multivariate Techniques (Kindle Locations 8292-8296). SAGE Publications. Kindle Edition.

Furthermore, the sample must be representative of the population for the inference prediction and observations of the variables should be independent of each other (each score on X should be independent of other X scores

and each score on Y should be independent of other Y scores). In other words, the errors are uncorrelated, that is, the variance–covariance matrix of the errors is diagonal and each non-zero element is the variance of the error. The variance of the error is constant across observations (homoscedasticity). If not, weighted least squares or other methods might instead be used.

```
# import data
head(numeric.df)
```

```
##   id residence            major computer  gpa gre project final anxiety
## 1  3      West             <NA>       PC 2.73 134    97.2  73.8      21
## 2 34     South   Social Science      Mac 3.80 154    86.8  98.7      12
## 3 40      East   Social Science       PC 3.49 156    93.8  70.4      15
## 4 66      East      Other Major       PC 3.41 162    89.3  87.5      16
## 5 67     South Physical Science      Mac 3.45 162    75.8  66.8      12
## 6 74      West   Social Science      Mac 2.96 147    79.9  41.9      16
##     rank achievement
## 1    Low       85.50
## 2   High       92.75
## 3 Medium       82.10
## 4 Medium       88.40
## 5   High       71.30
## 6 Medium       60.90
```

4.4.2 Test Univariate and Bivariate Outliers

Outliers are simply single data points within your data that do not follow the usual pattern. Since outliers can severly affect normality and homogeneity of variance, methods for detecting disparate observerations are described first. First, there can be no (univariate) outliers in each group of the independent variable for any of the dependent variables. This is a similar assumption to the one-way ANOVA, but for each dependent variable that you have in your MANOVA analysis. Univariate outliers are often just called outliers and are the same type of outliers you will have comeacross if you have conducted t-tests or ANOVAs. Multivariate outliers are cases which have an unusual combination of scores on the dependent variables.

1. Detect univariate outliers using boxplots,
2. Check for multivariate outliers using a measure called Mahalanobis distance.

```
# Fast algorithm for identifying multivariate outliers in high-dimensional
# and/or large datasets, using the algorithm of Filzmoser, Maronna, and Werner (CSDA, 2007).
out <- mvoutlier::pcout(na.omit(data.matrix(numeric.df[,c("anxiety", "achievement")])),
                        makeplot = TRUE)
```

```
## sROC 0.1-2 loaded
```

```
out$wfinal01 ## "0" is outlier
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   1   1   1   1   1   0   1   0   0   1   1   1   1   0   1   1   1   1
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   0   1   1   1   1   1   1   1   0   1   1   1   1   1   1   0   1   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
```

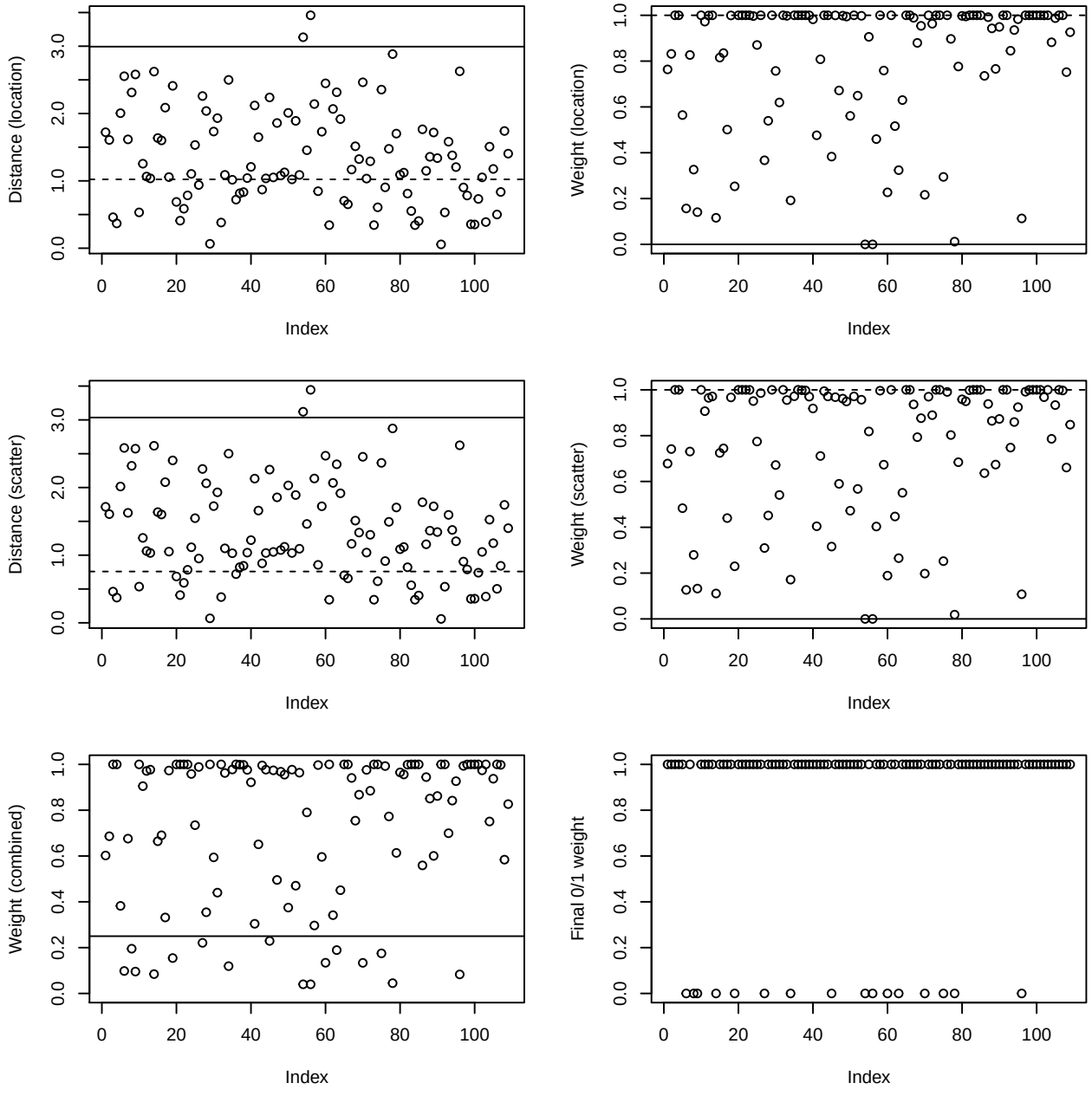
Distance (location)
Weight (location)
Distance (scatter)
Weight (scatter)
Weight (combined)
Final 0/1 weight
Index

Figure 20:

```
##   1   1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   0
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   1   0   1   1   1   0   1   1   0   1   1   1   1   1   1   0   1   1
##  73  74  75  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91
##   1   1   0   1   1   0   1   1   1   1   1   1   1   1   1   1   1   1
##  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109
##   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   1   1   1
## 110
##   1
```

4.4.3 Mahalanobis' Distance

Mahalanobis' distance (MD) is a statistical measure of the extent to which cases are multivariate outliers, based on a chi-square distribution, assessed using $p < .001$.

The critical chi-square values for 2 to 10 degrees of freedom at a critical alpha of .001 are shown below.

A maximum MD larger than the critical chi-square value for df = k (the number of predictor variables in the model) at a critical alpha value of .001 indicates the presence of one or more multivariate outliers.

df	Critical value
2	13.82
3	16.27
4	18.47
5	20.52
6	22.46
7	24.32
8	26.13
9	27.88
10	29.59

```
D2 <- mahalanobis(data.matrix(numeric.df[,c("anxiety", "achievement")]),
                  colMeans(data.matrix(numeric.df[,c("anxiety", "achievement")]),
                           na.rm = TRUE),
                  cov(data.matrix(numeric.df[,c("anxiety", "achievement")]),
                      y = NULL,
                      use = "pairwise.complete.obs"))
```

```
# Largest Mahalanobis' distance avlues (critical value = 13.82)
head(sort(D2, decreasing = TRUE), 5)
```

```
##       56       54        6       79       97
## 8.347715 6.963500 6.743081 6.178577 5.759521
```

```
plot(D2,
     cex = 2.5,
     main = expression("Outlier detection based on " * chi[df==2]^2 == 13.82),
     ylab = expression(D^2),
     col = "blue")
text(D2, cex = .6)
```

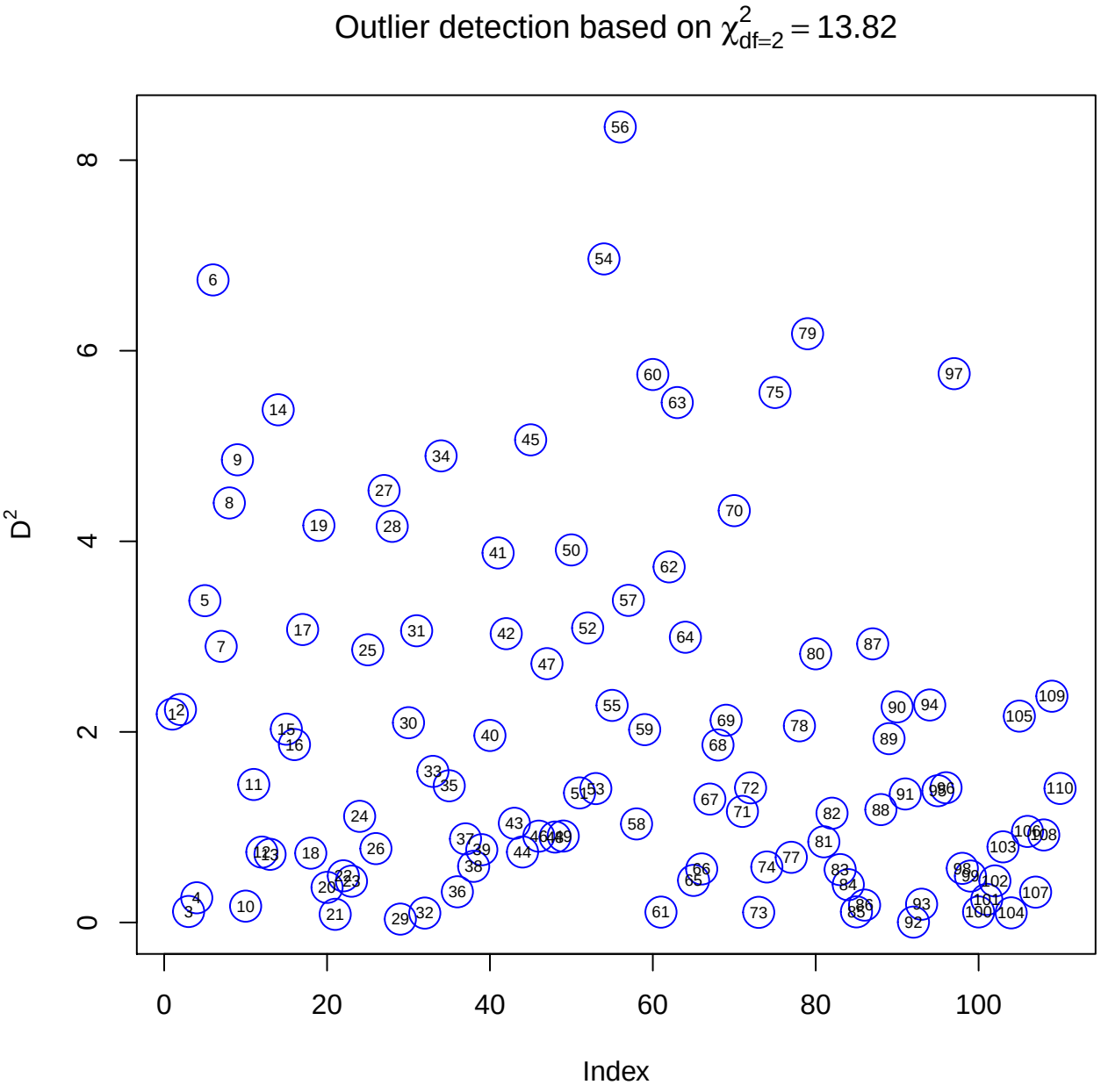
Outlier detection based on $\chi^2_{df=2} = 13.82$
D^2
0
2
4
6
8
0
20
40
60
80
100
Index

Figure 21:

```
plot(density(D2, bw = "nrd0", na.rm = T),
     main="Squared Mahalanobis distances, n = 100, p = 2")
rug(D2)
```

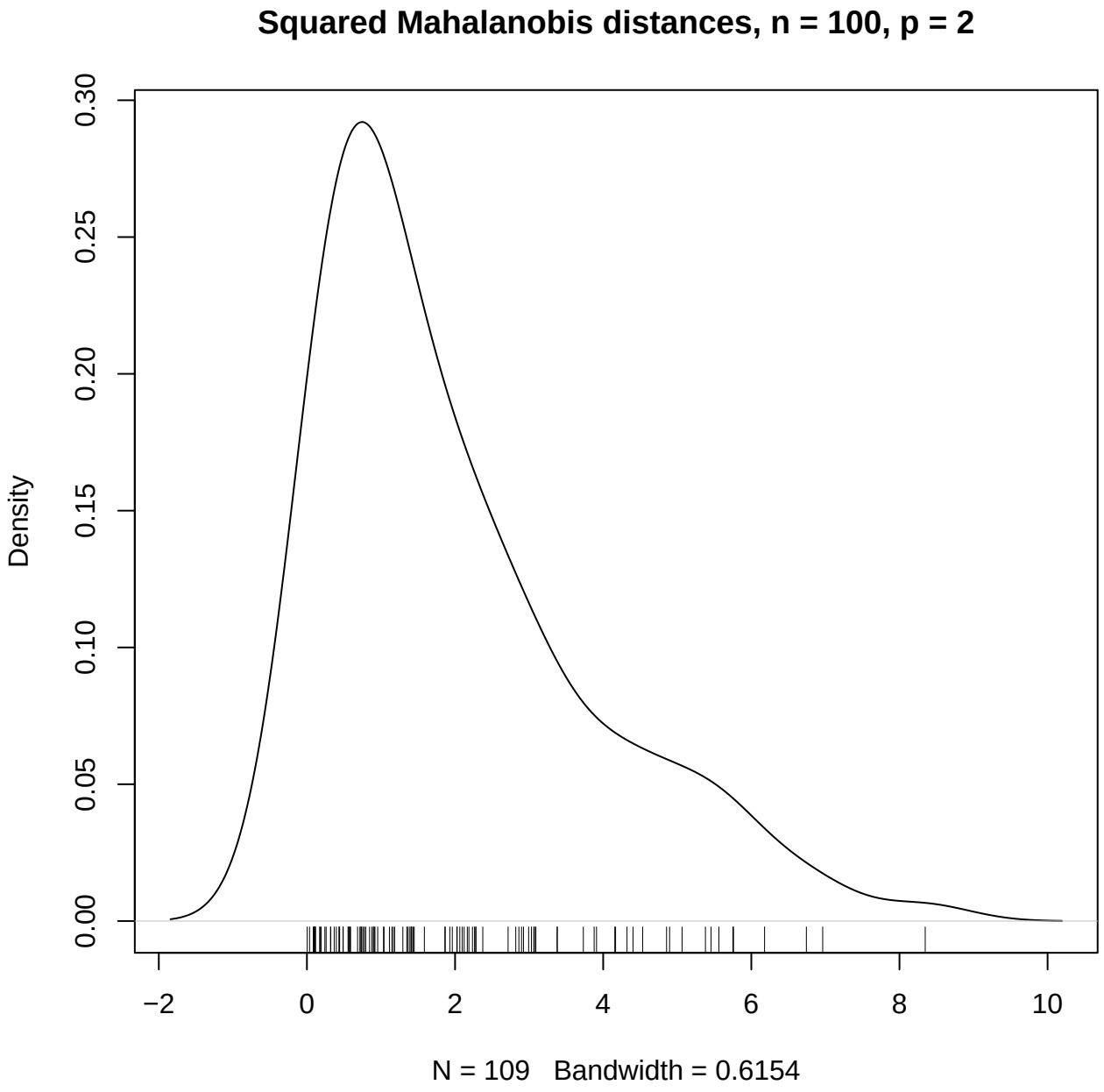

Figure 22:

```
qqplot(qchisq(ppoints(100), df = 2), D2,
       main = expression("Q-Q plot of Mahalanobis" * ~D^2 * " vs. quantiles of" * ~ chi[2]^2),
       xlab = expression(chi[2]^2 * ", probability points = 100"),
       ylab = expression(D^2),
       xlim = c(0, 15),
       ylim = c(0, 15))
abline(0, 1, col = 'red', lwd = 2)
```

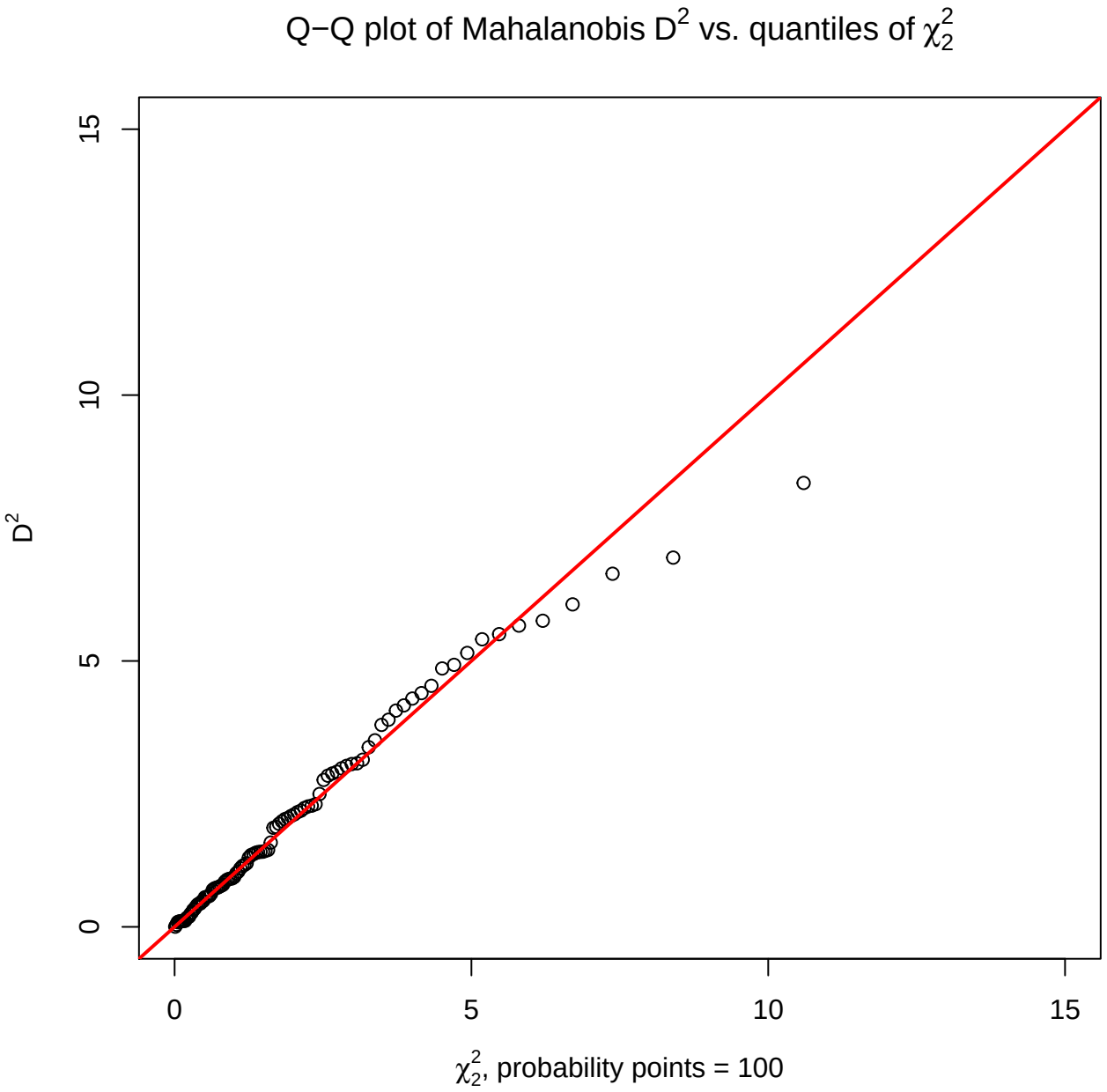
Q−Q plot of Mahalanobis D^2 vs. quantiles of χ^2_2
D^2
0
5
10
15
χ^2_2, probability points = 100

Figure 23:

```
plot(1/out$wfinal, cex = 2.5)
title("Outlier detection based on pcout")
text(50, 15, expression(chi[df==2]^2 == 13.83), cex = 1.5, col = "blue")
abline(h = 13.83, col = "blue", lwd = 2, lty = 2)
text(1/out$wfinal, names(out$wfinal), col = "red", cex = .6)
```

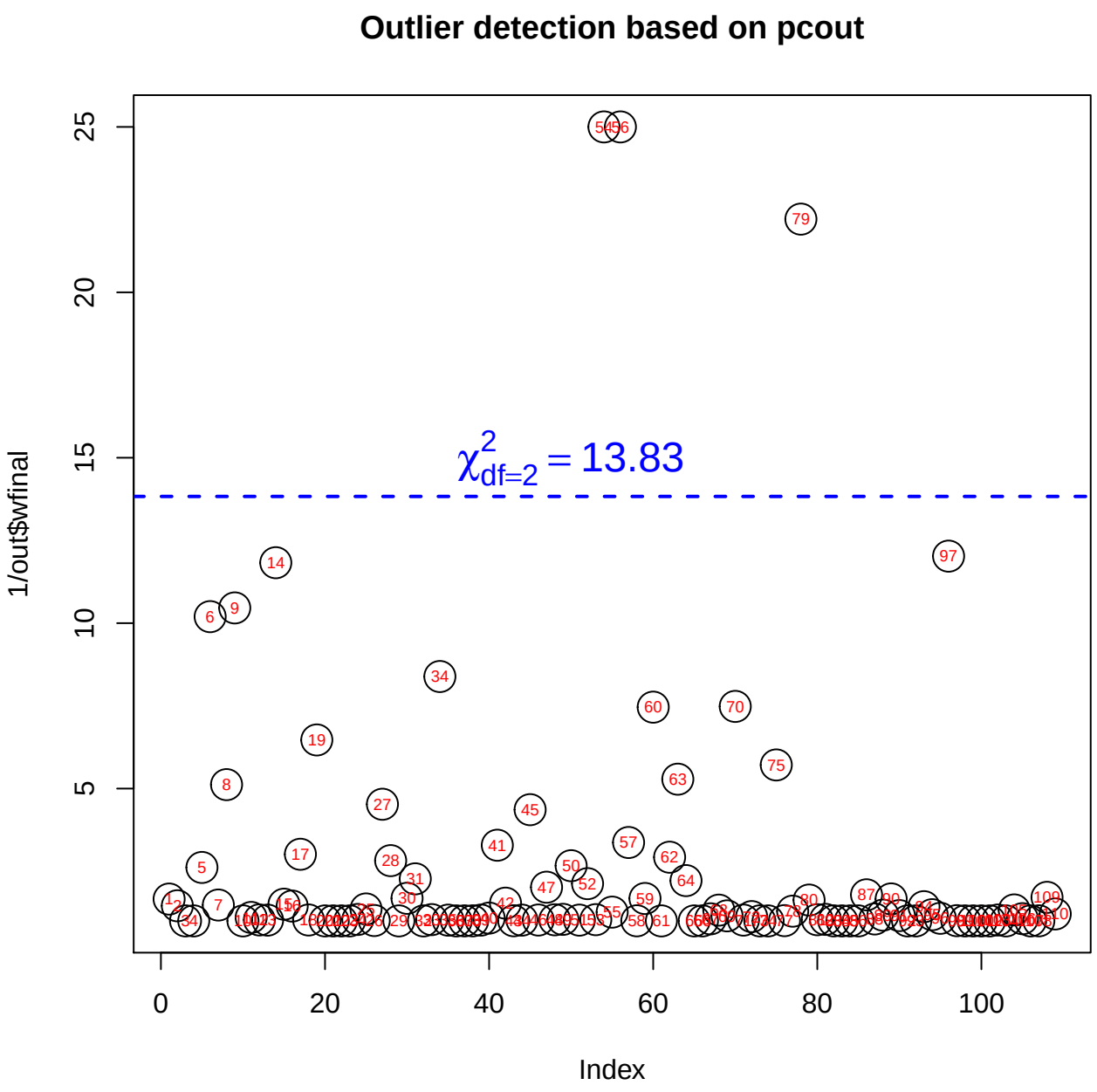

Figure 24:

4.4.4 Outliers based on adjusted quantile plots

```
outliers <-
  mvoutlier::aq.plot(na.omit(data.matrix(numeric.df[,c("anxiety", "achievement")])),
                     delta = qchisq(0.975, df = ncol(numeric.df[,c("anxiety", "achievement")])),
                     quan = 1/2, alpha = 0.05)
```

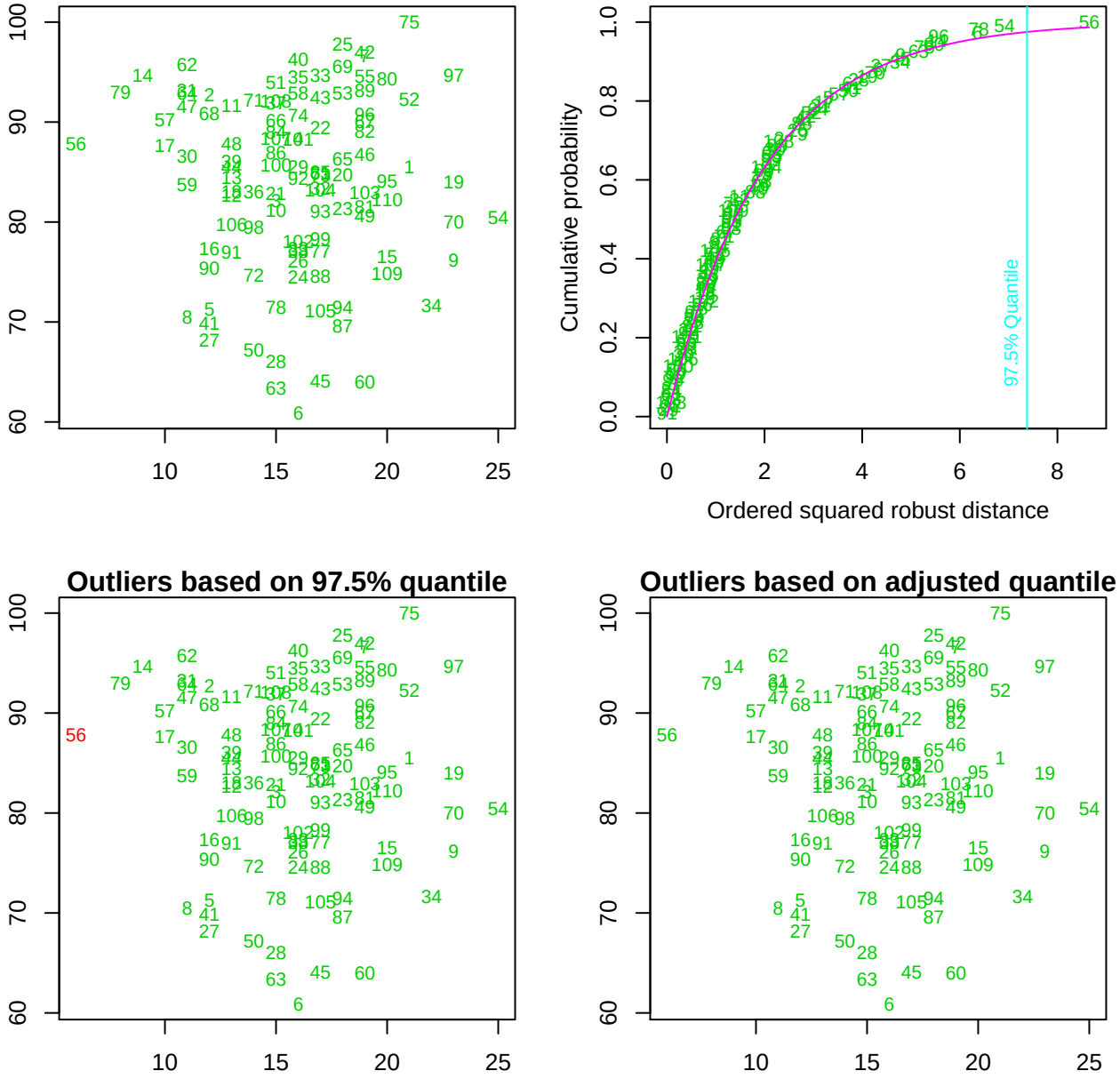

Figure 25:

```
outliers # show list of outliers
```

```
## $outliers
##     1     2     3     4     5     6     7     8     9    10    11    12
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    13    14    15    16    17    18    19    20    21    22    23    24
```

```
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    25    26    27    28    29    30    31    32    33    34    35    36
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    37    38    39    40    41    42    43    44    45    46    47    48
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    49    50    51    52    53    54    55    56    57    58    59    60
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    61    62    63    64    65    66    67    68    69    70    71    72
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    73    74    75    77    78    79    80    81    82    83    84    85
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    86    87    88    89    90    91    92    93    94    95    96    97
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    98    99   100   101   102   103   104   105   106   107   108   109
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   110
## FALSE
```

```
fit <- lm(achievement ~ anxiety, data = numeric.df)
summary(fit)
```

```
##
## Call:
## lm(formula = achievement ~ anxiety, data = numeric.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.9956  -6.5956   0.9122   7.3848  16.3740
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 84.75833    4.01097  21.132   <2e-16 ***
## anxiety     -0.05392    0.24595  -0.219    0.827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.897 on 107 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.000449,   Adjusted R-squared:  -0.008893
## F-statistic: 0.04806 on 1 and 107 DF,  p-value: 0.8269
```

4.4.5 Normality of Residuals

Assumption: Finally, you need to check that the residuals (errors) of the regression line is approximately normally distributed (we explain these terms in our enhanced linear regression guide). Two common methods to check this assumption include using either a histogram (with a superimposed normal curve) or by using a Normal P-P Plot.

This will produce a set of four plots: residuals versus fitted values, a Q-Q plot of standardized residuals, a scale-location plot (square roots of standardized residuals versus fitted values, and a plot of residuals versus leverage that adds bands corresponding to Cook's distances of 0.5 and 1.

Residuals versus fitted values gives an idea of whether there is any curvature in the data. If the red line is strongly curved, a quadratic or other model may be better. In this case, the curvature is not strong (so a quadratic component in the model is not necessary).

Q-Q plot of standardized residuals is to check whether the residuals are normally distributed.

Scale-location plot is used to check if the variance is contant (ie, if the standard deviation among the residuals appears to be about constant). If the red line is strongly tilted up/down, that is a red flag. There are no issues with that in this example - the variance appears constant. (The red line will always move up/down a little because of inherent randomness.)

Residuals versus leverage plot is used to check to see if there were any overly influential points.

```
op = par(mfrow=c(2,2), family="serif")
plot(fit,
     main = "Plot fit",
     col = rgb(0, 100, 0, 50, maxColorValue = 255),
     pch = 16)
```

```
par(op)
```

```
#Plots empirical quantiles of a variable, or of studentized residuals from a linear model, against theo
car::qqPlot(fit, main="Assessing Multivariate Outliers") #qq plot for studentized resid
```

4.4.6 Leverage Plots

The leverage of an observation measures its ability to move the regression model all by itself by simply moving in the y-direction. The leverage measures the amount by which the predicted value would change if the observation was shifted one unit in the y- direction.

The leverage always takes values between 0 and 1. A point with zero leverage has no effect on the regression model. If a point has leverage equal to 1 the line must follow the point perfectly.

These functions display a generalization, due to Sall (1990) and Cook and Weisberg (1991), of added-variable plots to multiple-df terms in a linear model. When a term has just 1 df, the leverage plot is a rescaled version of the usual added-variable (partial-regression) plot.

```
car::leveragePlots(fit) # leverage plots
```

These functions construct **added-variable (also called partial-regression) plots** for linear and generalized linear models.

In applied statistics, a partial regression plot attempts to show the effect of adding another variable to a model already having one or more independent variables. Partial regression plots are also referred to as added variable plots, adjusted variable plots, and individual coefficient plots.

When performing a linear regression with a single independent variable, a scatter plot of the response variable against the independent variable provides a good indication of the nature of the relationship. If there is more than one independent variable, things become more complicated. Although it can still be useful to generate scatter plots of the response variable against each of the independent variables, this does not take into account the effect of the other independent variables in the model.

Partial regression plots are formed by:

1. Computing the residuals of regressing the response variable against the independent variables but omitting X_i
2. Computing the residuals from regressing X_i against the remaining independent variables
3. Plotting the residuals from (1) against the residuals from (2).

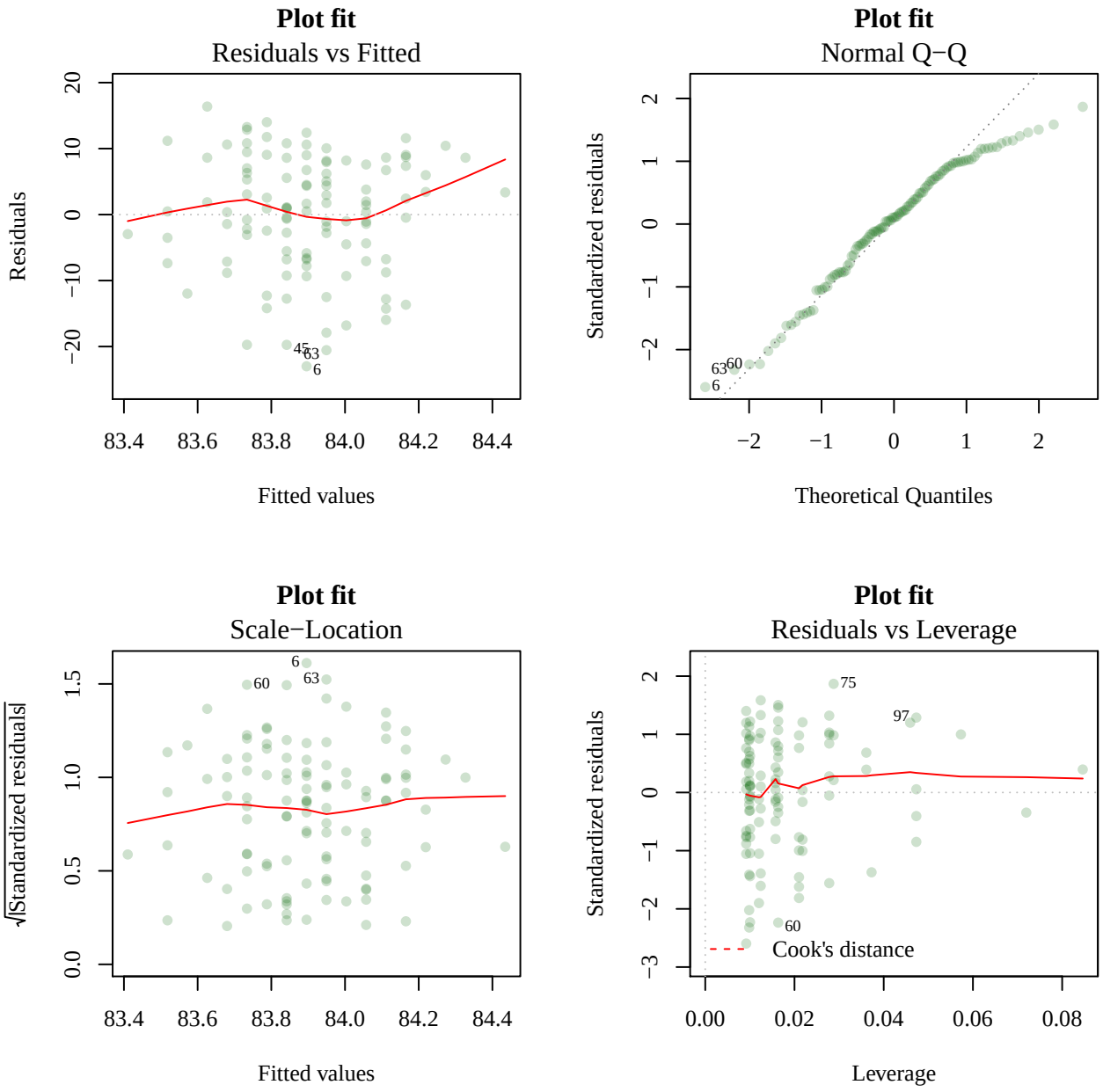
Plot fit
Residuals vs Fitted
Residuals
Fitted values
Plot fit
Normal Q−Q
Standardized residuals
Theoretical Quantiles
Plot fit
Scale−Location
√|Standardized residuals|
Fitted values
Plot fit
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

Figure 26:


Assessing Multivariate Outliers
Studentized Residuals(fit)
t Quantiles
2
1
0
−1
−2
−2
−1
0
1
2

Figure 27:

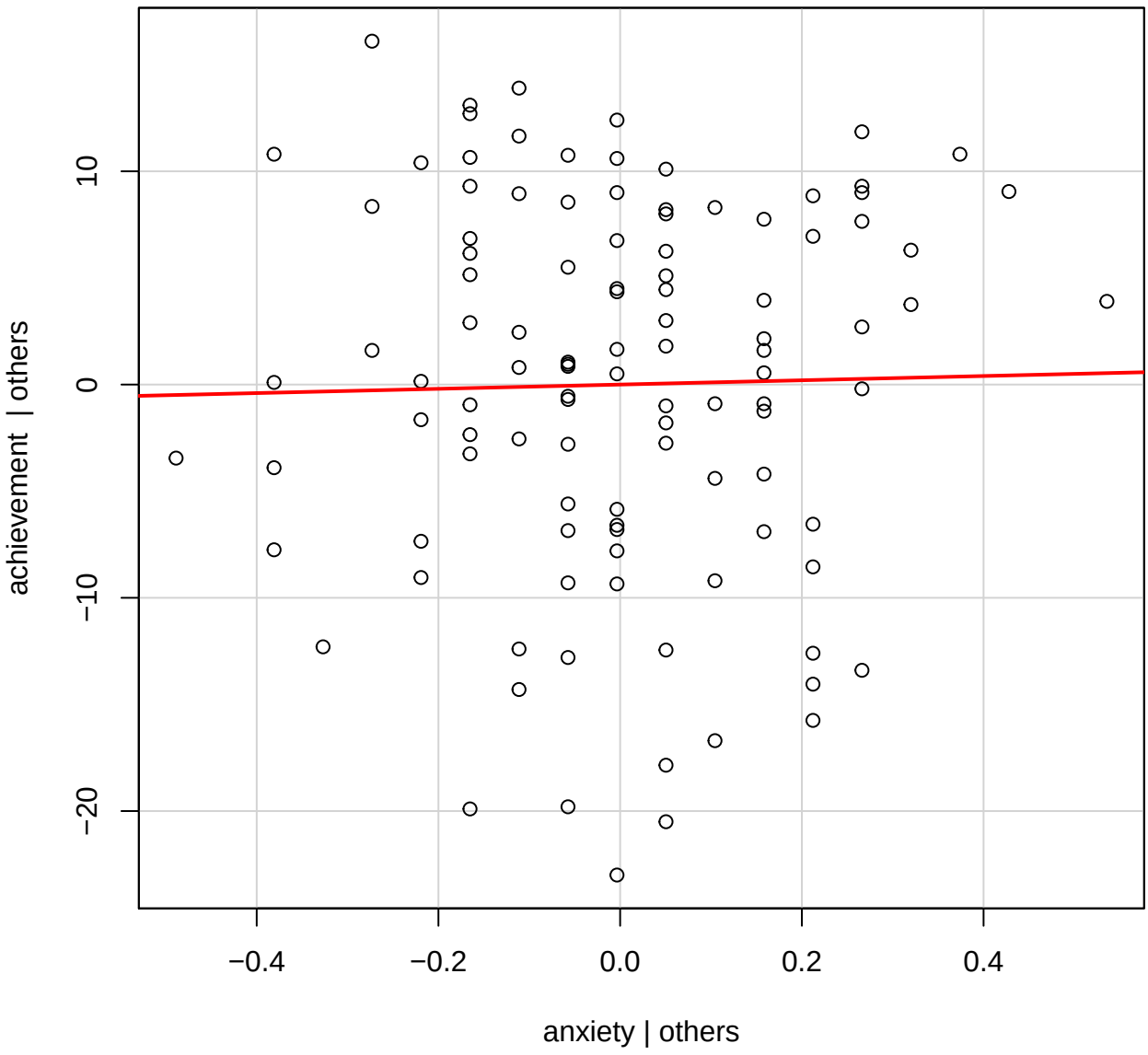

Figure 28:

```
car::avPlots(fit) # leverage plots
```

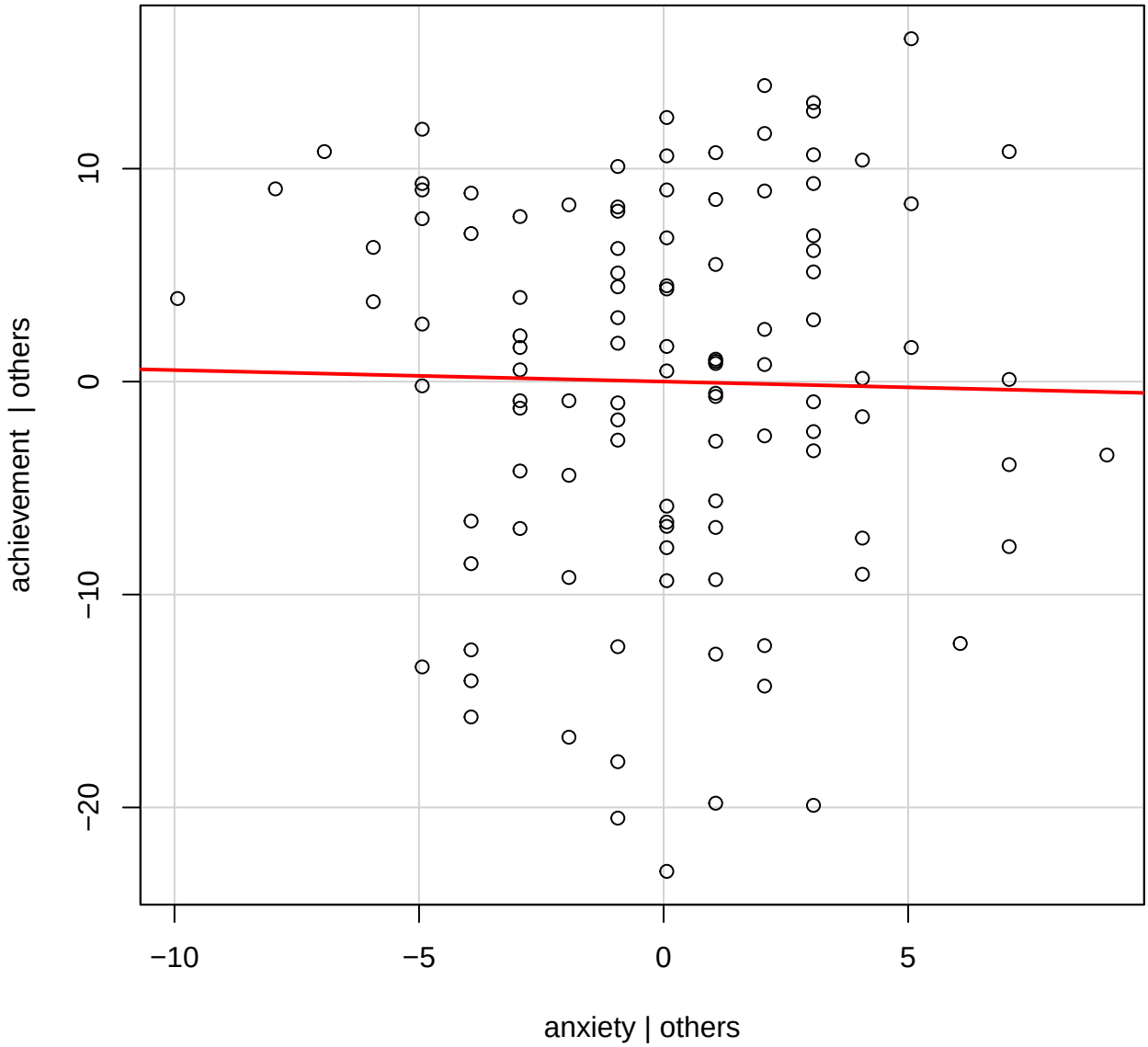

Figure 29:

4.4.7 Cook's Distanece

```
# Cook's D plot identify D values > 4/(n-k-1)
(cutoff <- round(4/((nrow(numeric.df)-length(fit$coefficients)-1)), 3))
```

```
## [1] 0.037
```

```
plot(fit, which=4, cook.levels=cutoff)
abline(h = cutoff, col = "red", lty = 2)
text(15, 0.04, expression(frac(4, n-k-1) * " = 0.037"), cex = 1, col = "red")
```

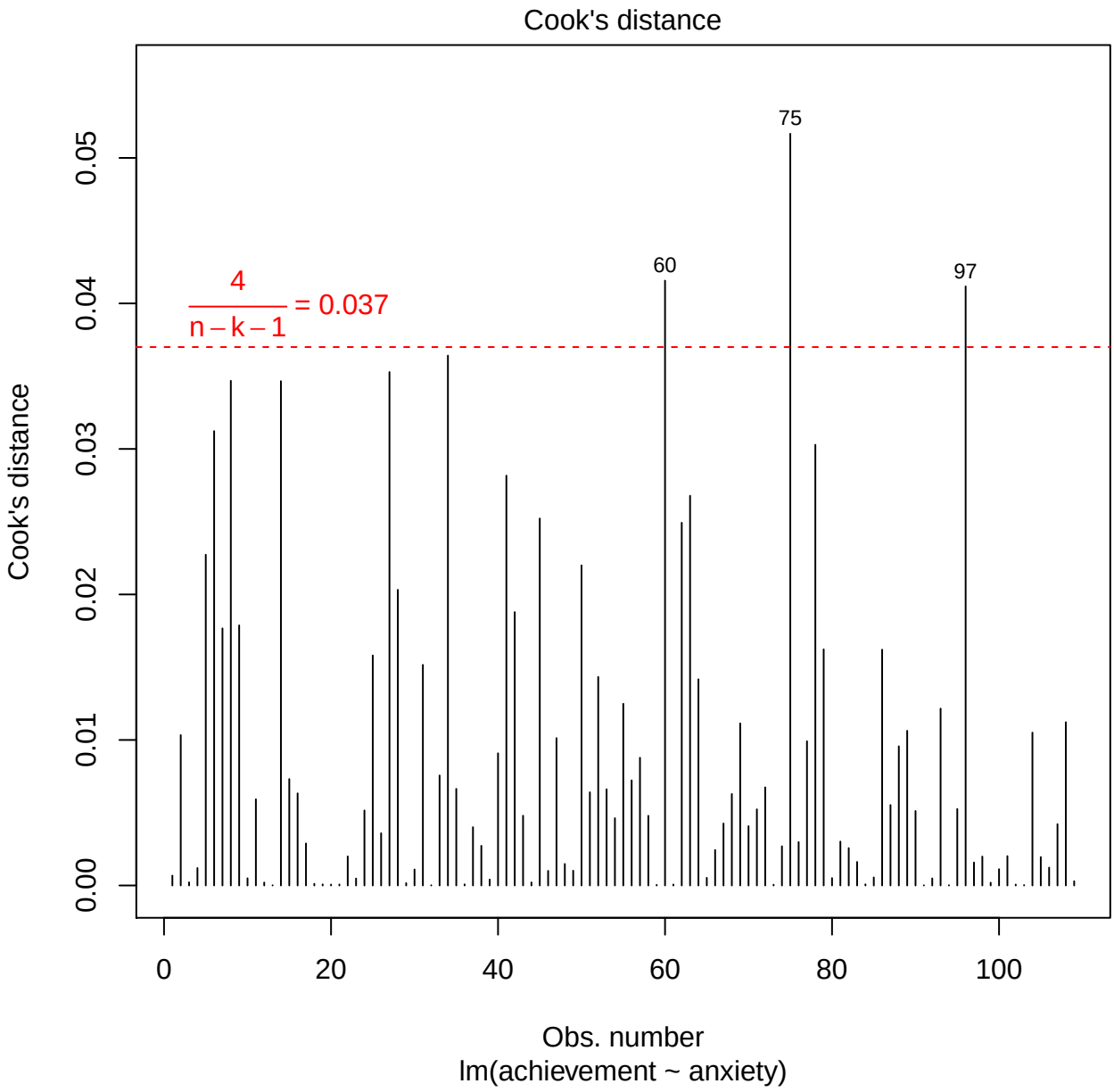
Cook's distance
75
60
97
4/(n−k−1) = 0.037
Cook's distance
0.00
0.01
0.02
0.03
0.04
0.05
0
20
40
60
80
100
Obs. number
lm(achievement ~ anxiety)

Figure 30:

4.4.8 Test Non-independence of Errors

Assumption: observations are independent, which you can easily check using the Durbin-Watson statistic Test for Autocorrelated Errors In statistics, the Durbin–Watson statistic is a test statistic used to detect the presence of autocorrelation (a relationship between values separated from each other by a given time lag) in the residuals (prediction errors) from a regression analysis. Durbin and Watson (1950, 1951) applied this statistic to the residuals from least squares regressions, and developed bounds tests for the null hypothesis that the errors are serially uncorrelated against the alternative that they follow a first order autoregressive process. Later, John Denis Sargan and Alok Bhargava developed several von Neumann–Durbin–Watson type test statistics for the null hypothesis that the errors on a regression model follow a process with a unit root against the alternative hypothesis that the errors follow a stationary first order autoregression (Sargan and Bhargava, 1983). Positive serial correlation is serial correlation in which a positive error for one observation increases the chances of a positive error for another observation. Negative serial correlation implies that a positive error for one observation increases the chance of a negative error for another observation and a negative error for one observation increases the chances of a positive error for another.

```
# null hypothesis: the errors are serially uncorrelated against the alternative that they follow a firs
car::dwt(fit,
         max.lag = 1,
         simulate = TRUE, # if TRUE p-values will be estimated by bootstrapping
         reps = 1000,
         method = "normal",
         alternative = "two.sided"
         )
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      -0.1050191      2.209382    0.31
##  Alternative hypothesis: rho != 0
```

4.4.9 Test Homoscedasticity

Assumption 5: Your data needs to show homoscedasticity, which is where the variances along the line of best fit remain similar as you move along the line. non-constant error variance test (Breusch-Pagan test) Computes a score test of the hypothesis of constant error variance against the alternative that the error variance changes with the level of the response (fitted values), or with a linear combination of predictors.

```
car::ncvTest(fit)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.03731139    Df = 1     p = 0.8468324
```

4.4.10 Studentized Residuals vs. Fitted Values

Creates plots for examining the possible dependence of spread on level, or an extension of these plots to the studentized residuals from linear models. spread-stabilizing power transformation, calculated as 1 - slope of the line fit to the plot.

```
car::spreadLevelPlot(fit)
```

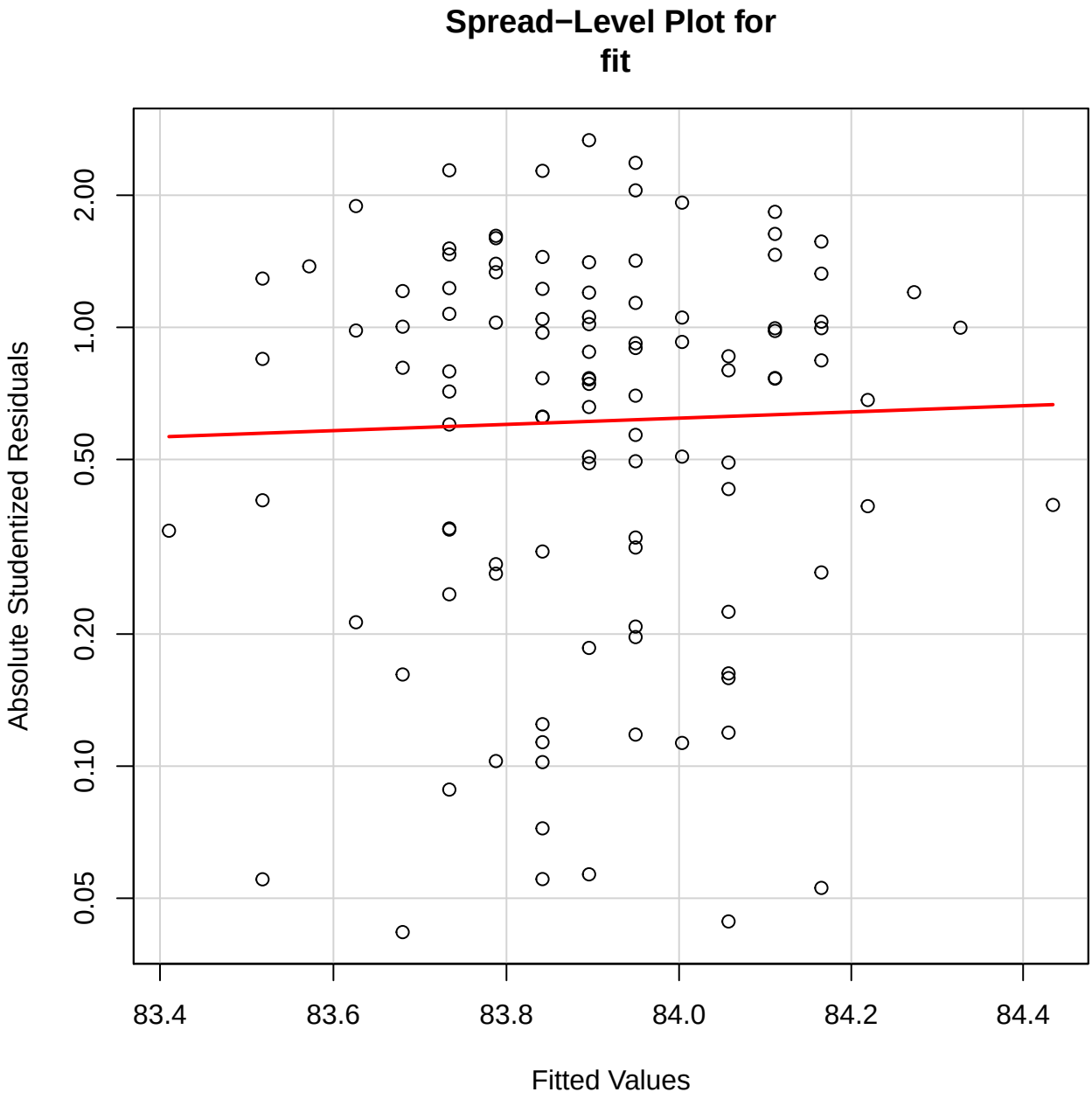
Spread−Level Plot for
fit
Absolute Studentized Residuals
0.05
0.10
0.20
0.50
1.00
2.00
83.4
83.6
83.8
84.0
84.2
84.4
Fitted Values

Figure 31:

```
##
## Suggested power transformation:  -12.74197
```

```
library(MASS)

# studentized residuals
sresid <- studres(fit)

# studentized residuals histogram
hist(sresid,
     freq = FALSE,
     main = "Distribution of Studentized Residuals",
     xlim = c(-3, 3)
)

# normal curve
xfit <- seq(-3, 3, length = 100)
yfit <- dnorm(xfit)
lines(xfit, yfit)
```

```
# These functions construct component+residual plots (also called partial-residual plots) for linear an
car::crPlots(fit)
```

```
summary(fit)
```

```
##
## Call:
## lm(formula = achievement ~ anxiety, data = numeric.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.9956  -6.5956   0.9122   7.3848  16.3740
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 84.75833    4.01097  21.132   <2e-16 ***
## anxiety     -0.05392    0.24595  -0.219    0.827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.897 on 107 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.000449,   Adjusted R-squared:  -0.008893
## F-statistic: 0.04806 on 1 and 107 DF,  p-value: 0.8269
```

4.5 Chi-square test of goodness of fit

Research Question: Are the proportions of Mac and PC users the same among graduate students?

A Pearson's chi-square test of goodness-of-fit was performed to examine whether the proportions of Mac and PC users are the same among graduate students. Preference for the types of computer was equally distributed in the population, χ^2 (1, 107) = 0.4579, p = .5654.

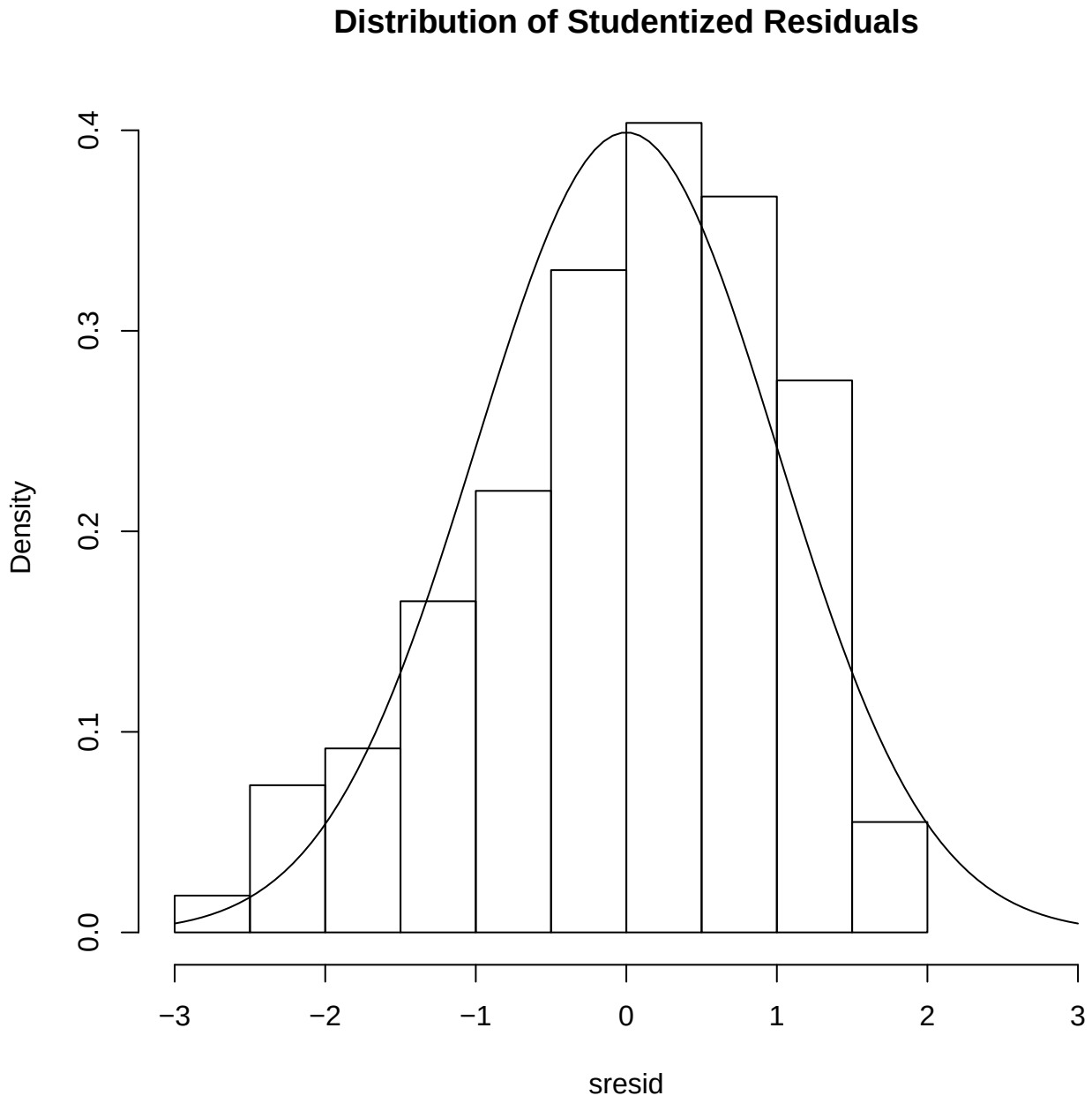
Distribution of Studentized Residuals
Density
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
sresid

Figure 32:

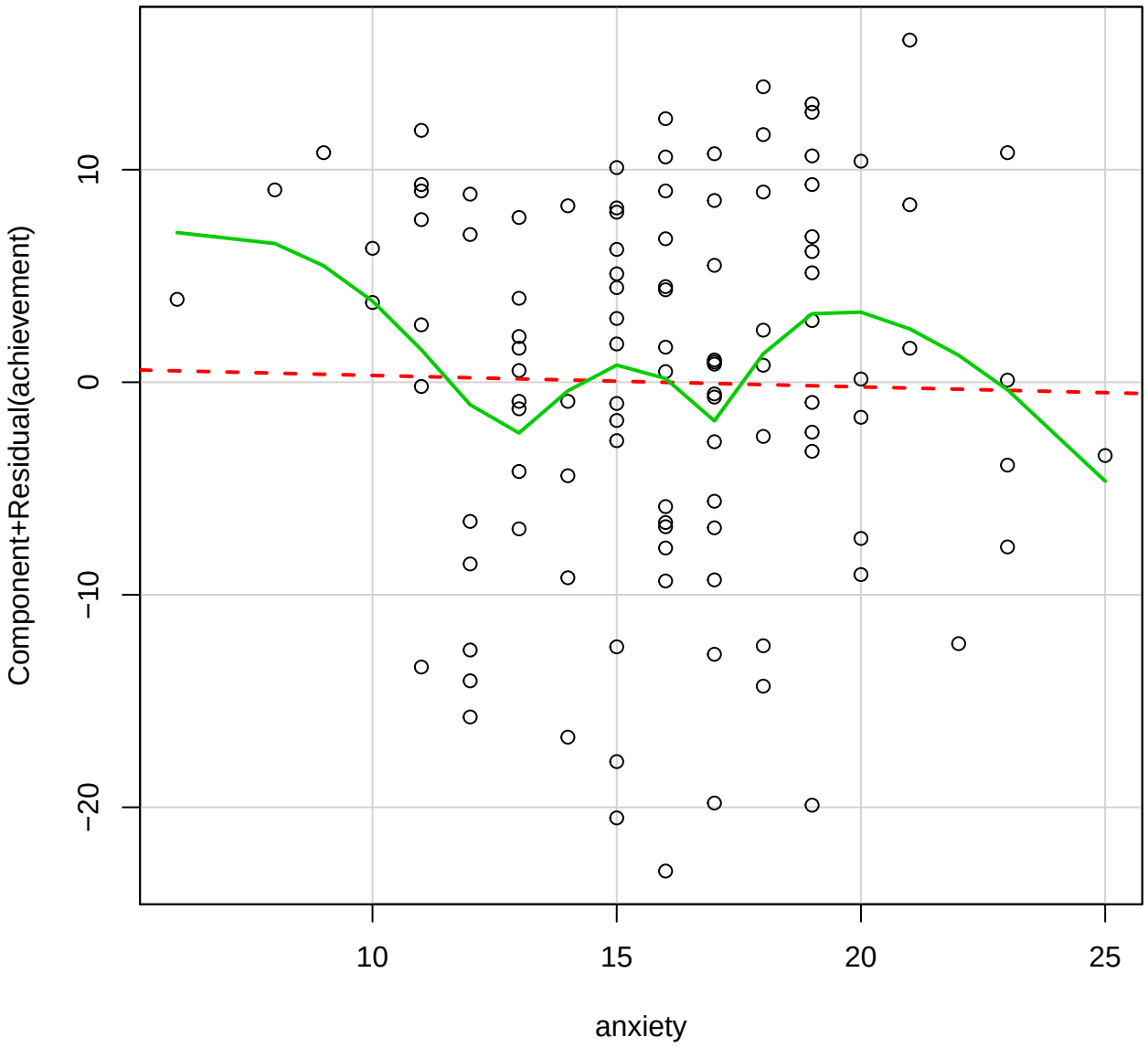
Component+Residual(achievement)
10
0
−10
−20
10
15
20
25
anxiety

Figure 33:

```
table(numeric.df$computer)
```

```
##
## Mac  PC
##  50  57
```

```
prop.table(table(numeric.df$computer))
```

```
##
##       Mac        PC
## 0.4672897 0.5327103
```

```
x <- na.omit(numeric.df$computer)
length(x)
```

```
## [1] 107
```

```
chisq.test(x = table(x),
           y = NULL,
           length(x),
           rescale.p = FALSE,
           simulate.p.value = TRUE,
           B = 1000
)
```

```
##
##  Chi-squared test for given probabilities with simulated p-value
##  (based on 1000 replicates)
##
## data:  table(x)
## X-squared = 0.45794, df = NA, p-value = 0.5764
```

prop.test can be used for testing the null that the proportions (probabilities of success) in several groups are the same, or that they equal certain given values.

```
prop.test(sum(na.omit(as.integer(numeric.df$computer)) == "1"), length(na.omit(numeric.df$computer)))
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  sum(na.omit(as.integer(numeric.df$computer)) == "1") out of length(na.omit(numeric.df$computer
## X-squared = 0.33645, df = 1, p-value = 0.5619
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.3710816 0.5658752
## sample estimates:
##         p
## 0.4672897
```

Goodness-of-fit test goodfit essentially computes the fitted values of a discrete distribution (either Poisson, binomial or negative binomial) to the count data given in x. If the parameters are not specified they are estimated either by ML or Minimum Chi-squared..

```
summary(gf <- vcd::goodfit(x = as.integer(numeric.df$computer),
                      type = "binomial",
                      method = c("ML", "MinChisq"),
                      par = NULL
                      )
        )
```

```
## Warning in vcd::goodfit(x = as.integer(numeric.df$computer), type =
## "binomial", : size was not given, taken as maximum count

##
##   Goodness-of-fit test for binomial distribution
##
##                       X^2 df P(> X^2)
## Likelihood Ratio 15.4893  0        0
```

```
plot(gf, main ="Count data vs Binomial distribution")
```

4.6 One-sample Kolmogorov-Smirnov test

If y is numeric, a two-sample test of the null hypothesis that x and y were drawn from the same continuous distribution is performed.

```
ks.test(numeric.df$gpa,"pweibull",
        shape = 2,
        scale = 1,
        alternative = "two.sided")
```

```
## Warning in ks.test(numeric.df$gpa, "pweibull", shape = 2, scale = 1,
## alternative = "two.sided"): ties should not be present for the Kolmogorov-
## Smirnov test

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  numeric.df$gpa
## D = 0.98555, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

4.7 Chi-square test of independence (cross-tabulation)

Research Question: Is there a relationship between Statistics Anxiety (as high, medium, and low ntiles) and major among graduate students?

A Pearson's chi-square test of independence was performed to examine the relation between the three statistics anxiety levels and the choice of academic majors among graduate students. The relation between these variables was not significant, $ χ^2 (4, 107) = 1.8479, p = 0.7637.$ There isn't any relationship between statistics anxiety and the choice of academic majors among the graduate students.

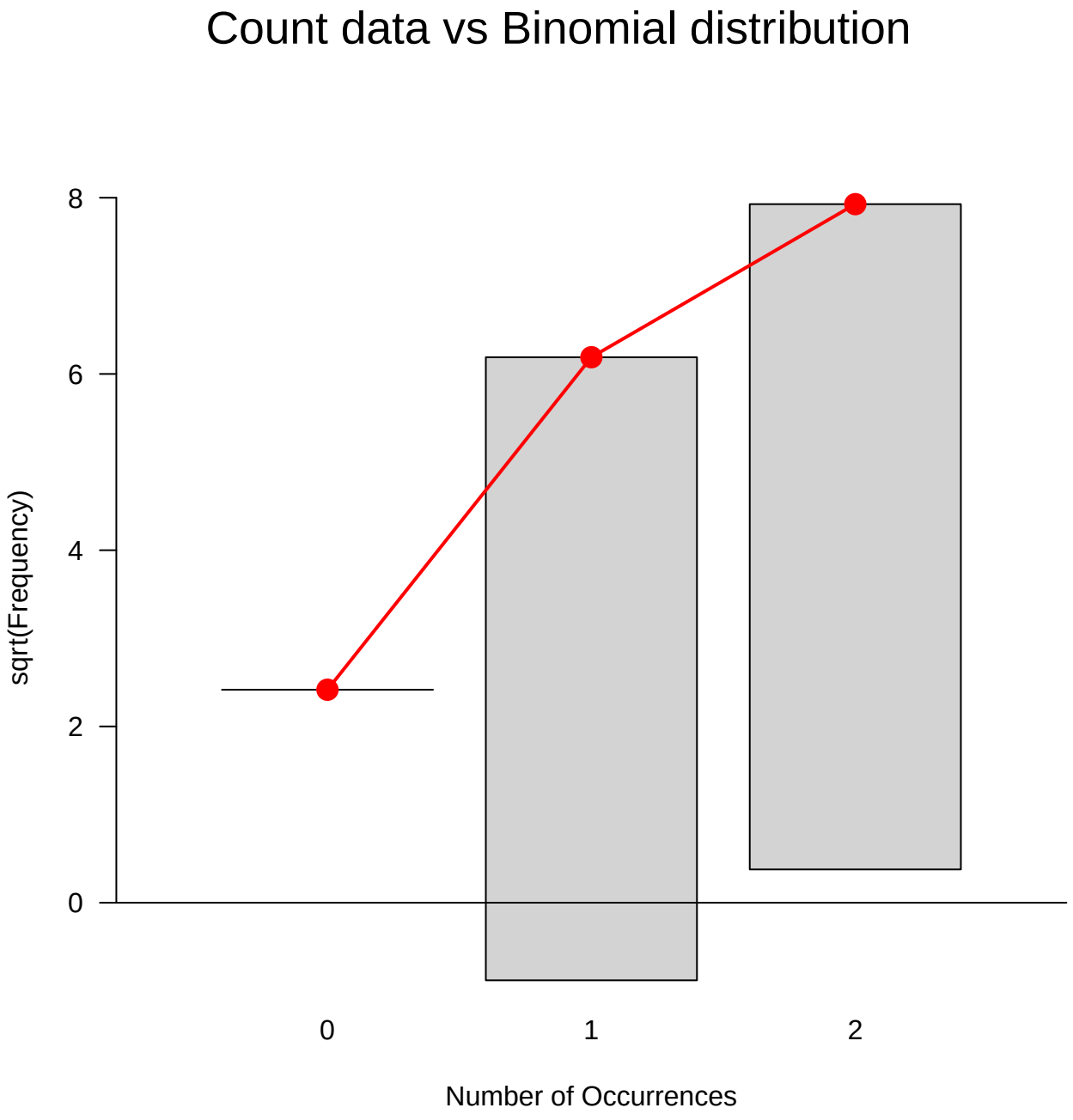
Count data vs Binomial distribution
sqrt(Frequency)
0
2
4
6
8
0
1
2
Number of Occurrences

Figure 34:

```r
(mytable <- table(numeric.df$rank, numeric.df$major))
```

```
##
##          Other Major Physical Science Social Science
##   High             5                7              7
##   Low              9               11              5
##   Medium          18               24             21
```

```r
# Create a contingency table
contingency.table <- xtabs(~rank + major,
                           data = numeric.df,
                           exclude = c(NA, NaN),
                           drop.unused.levels = FALSE
                           )
```

```r
margin.table(contingency.table, 1) # A frequencies (summed over major)
```

```
## rank
##   High    Low Medium
##     19     25     63
```

```r
margin.table(contingency.table, 2) # B frequencies (summed over rank)
```

```
## major
##      Other Major Physical Science   Social Science
##               32               42               33
```

```r
prop.table(contingency.table) # cell percentages
```

```
##         major
## rank     Other Major Physical Science Social Science
##   High    0.04672897       0.06542056     0.06542056
##   Low     0.08411215       0.10280374     0.04672897
##   Medium  0.16822430       0.22429907     0.19626168
```

```r
prop.table(contingency.table, 1) # row percentages
```

```
##         major
## rank     Other Major Physical Science Social Science
##   High     0.2631579        0.3684211      0.3684211
##   Low      0.3600000        0.4400000      0.2000000
##   Medium   0.2857143        0.3809524      0.3333333
```

```r
prop.table(contingency.table, 2) # column percentages
```

```
##         major
## rank     Other Major Physical Science Social Science
##   High     0.1562500        0.1666667      0.2121212
##   Low      0.2812500        0.2619048      0.1515152
##   Medium   0.5625000        0.5714286      0.6363636
```

```
gmodels::CrossTable(numeric.df$rank, numeric.df$major)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  107
##
##
##                  | numeric.df$major
## numeric.df$rank |      Other Major | Physical Science |   Social Science |        Row Total |
## ----------------|------------------|------------------|------------------|------------------|
##            High |                5 |                7 |                7 |               19 |
##                 |            0.082 |            0.028 |            0.222 |                  |
##                 |            0.263 |            0.368 |            0.368 |            0.178 |
##                 |            0.156 |            0.167 |            0.212 |                  |
##                 |            0.047 |            0.065 |            0.065 |                  |
## ----------------|------------------|------------------|------------------|------------------|
##             Low |                9 |               11 |                5 |               25 |
##                 |            0.310 |            0.144 |            0.953 |                  |
##                 |            0.360 |            0.440 |            0.200 |            0.234 |
##                 |            0.281 |            0.262 |            0.152 |                  |
##                 |            0.084 |            0.103 |            0.047 |                  |
## ----------------|------------------|------------------|------------------|------------------|
##          Medium |               18 |               24 |               21 |               63 |
##                 |            0.038 |            0.021 |            0.127 |                  |
##                 |            0.286 |            0.381 |            0.333 |            0.589 |
##                 |            0.562 |            0.571 |            0.636 |                  |
##                 |            0.168 |            0.224 |            0.196 |                  |
## ----------------|------------------|------------------|------------------|------------------|
##    Column Total |               32 |               42 |               33 |              107 |
##                 |            0.299 |            0.393 |            0.308 |                  |
## ----------------|------------------|------------------|------------------|------------------|
##
##
```

```
vcd::assocstats(mytable)
```

```
##                     X^2 df P(> X^2)
## Likelihood Ratio 2.0250  4  0.73117
## Pearson          1.9245  4  0.74965
##
## Phi-Coefficient   : NA
## Contingency Coeff.: 0.133
## Cramer's V        : 0.095
```

```
kappa(mytable)
```

```
## [1] 529.7152
```

```
vcd::mosaic(~rank + major,
            data = numeric.df,
            shade = TRUE,
            legend = TRUE
            )
```

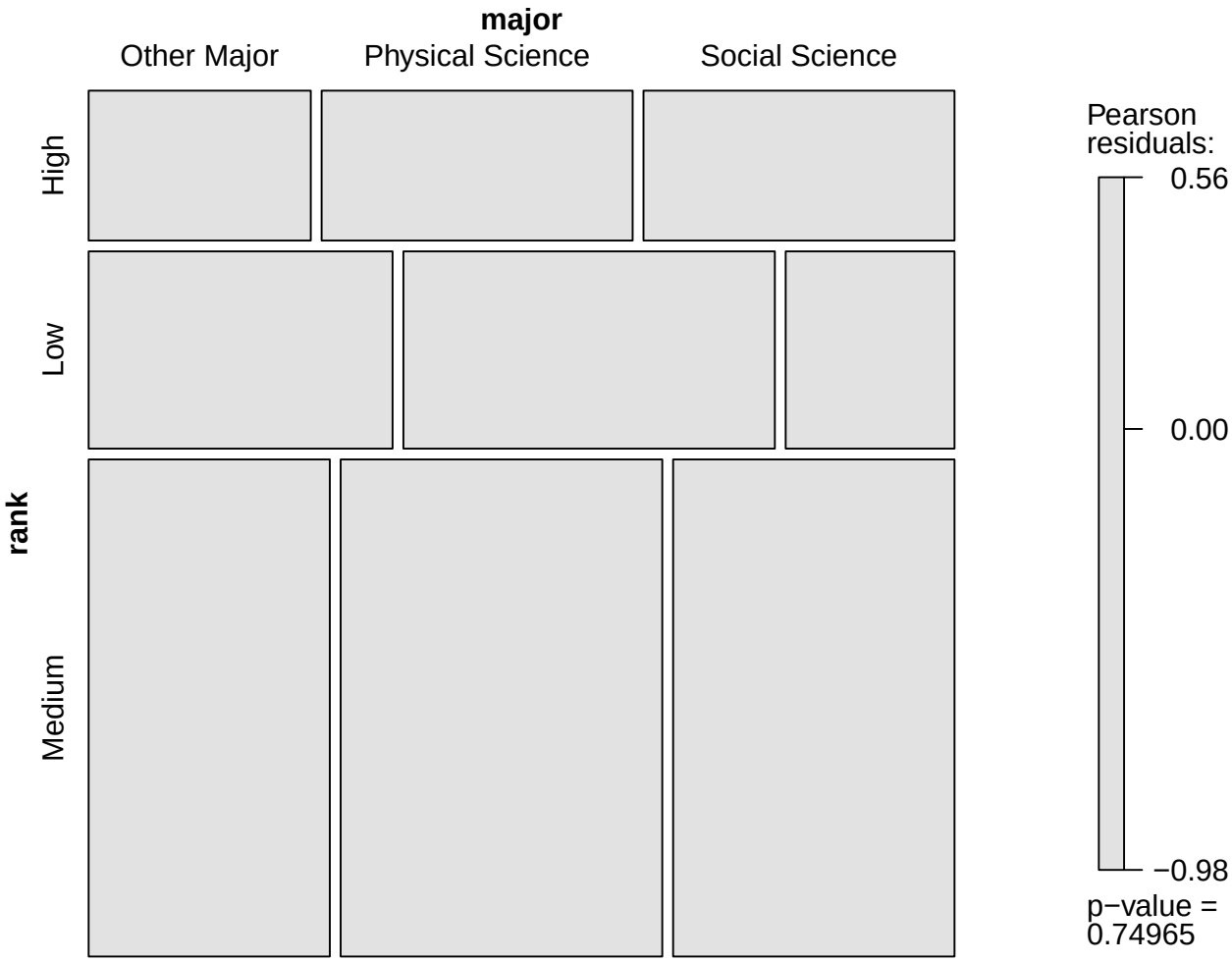

Figure 35:

```
vcd::assoc(~rank + major,
           data = numeric.df,
           shade = TRUE
           )
```

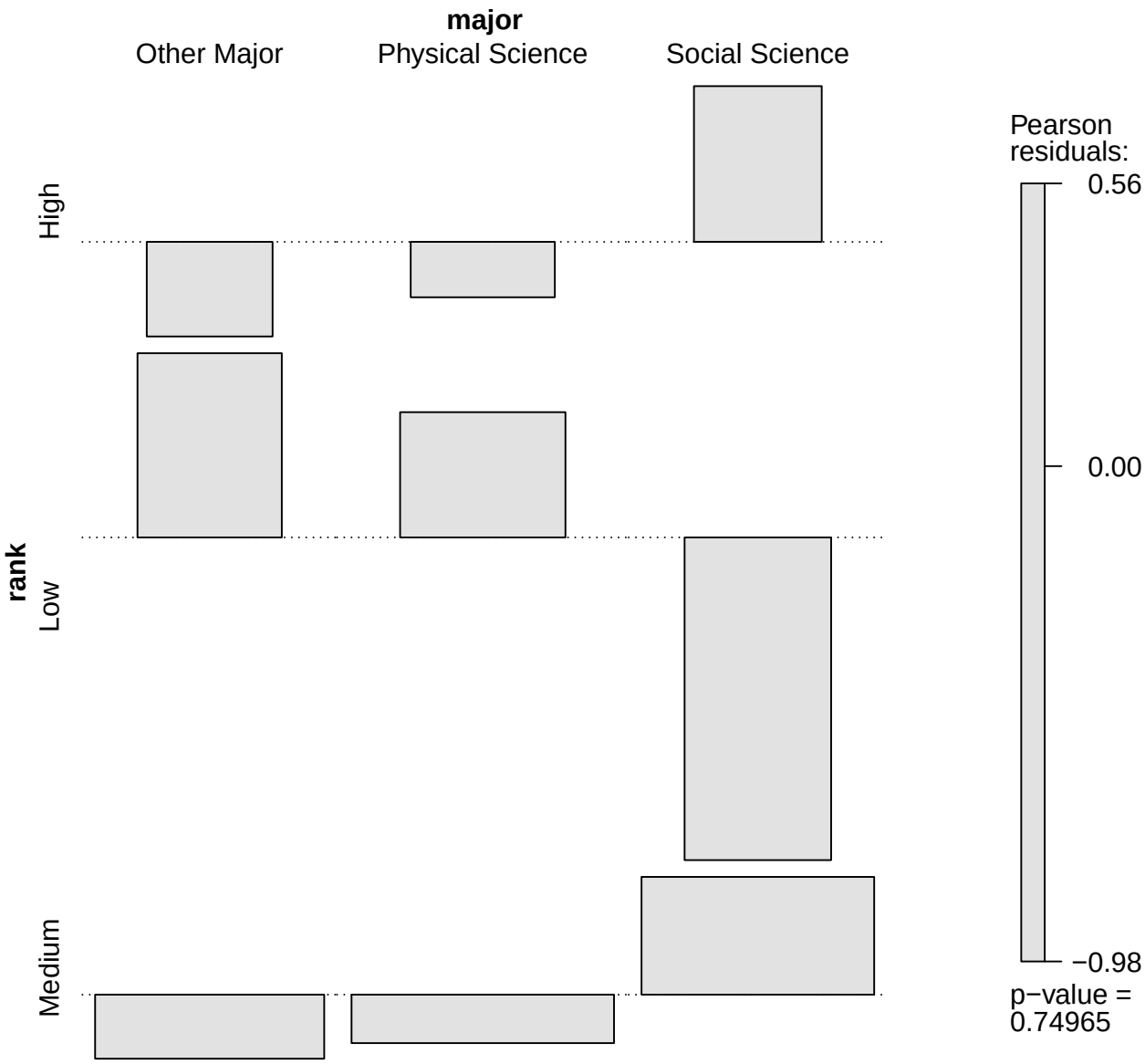
major
Other Major
Physical Science
Social Science
rank
High
Low
Medium
Pearson
residuals:
0.56
0.00
−0.98
p−value =
0.74965

Figure 36:

```
counts <- table(numeric.df$rank)
barplot(mytable,
        main = "Major Distribution",
        xlab = "Majors"
        )
```

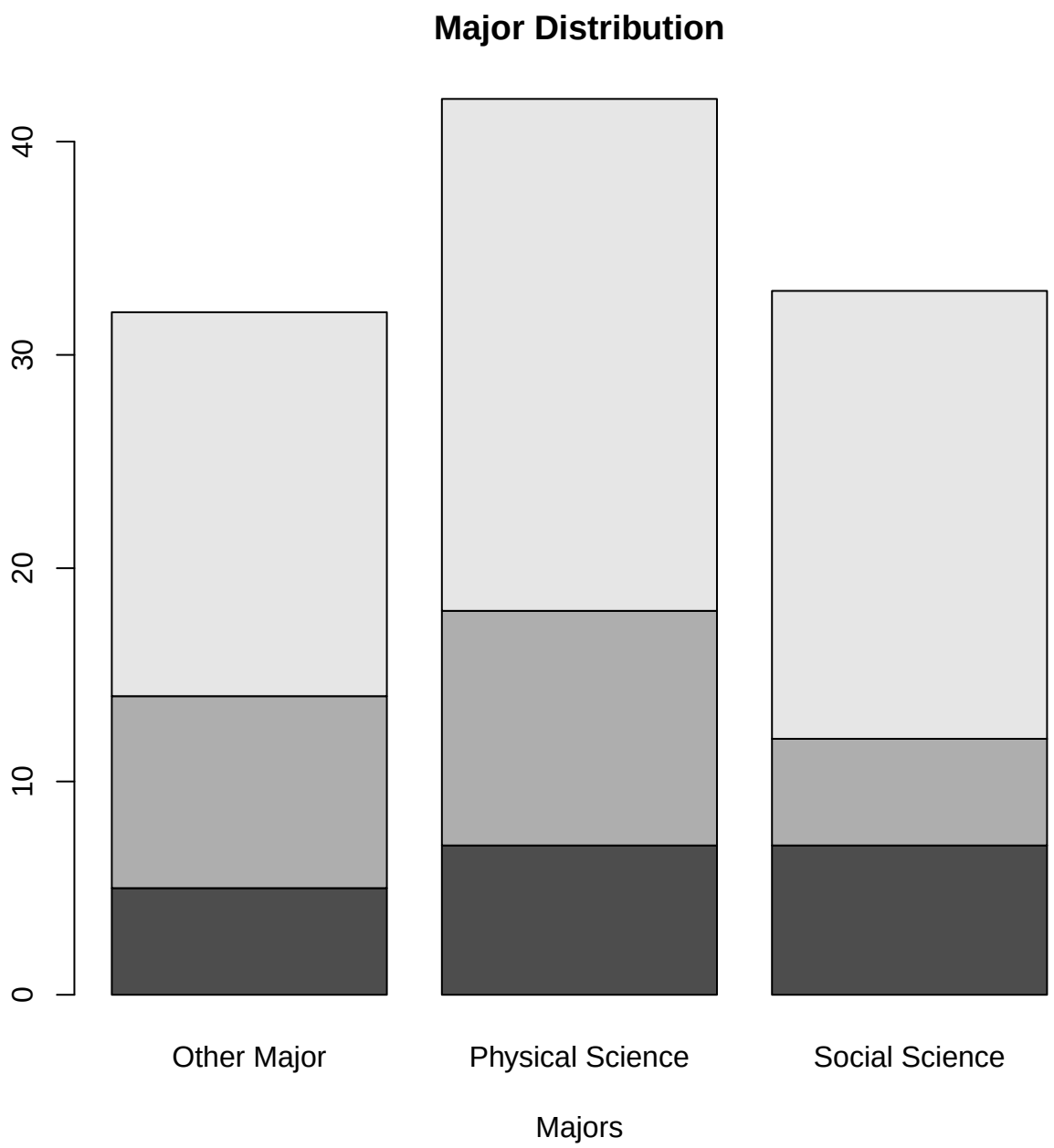

Figure 37:

$\chi^2 - value$

df/p	0.10	0.05	0.01	0.001
1	2.71	3.84	6.64	10.83
2	4.60	5.99	9.21	13.82

df/p	0.10	0.05	0.01	0.001
3	6.25	7.82	11.34	16.27
4	7.78	9.49	13.28	18.47
5	9.24	11.07	15.09	20.52
6	10.64	12.59	16.81	22.46
7	12.02	14.07	18.48	24.32
8	13.36	15.51	20.09	26.12
9	14.68	16.92	21.67	27.88
10	15.99	18.31	23.21	29.59

4.8 One-sample t-test

Research Question: Is average GPA different from the known population graduate student GPA of 3.42?

Graduate students taking the introductory level statistics courses had a GPA score ($\mu = 3.15$, $\sigma = 0.41$) that is statistically significantly different than the known polpulation mean GPA of 3.42, t(108) = -6.85, p < 0.001

```
# Boxplot
boxplot(na.omit(numeric.df$gpa),
        notch = TRUE,
        horizontal = TRUE,
        xlab = "GPA",
        col = "dodgerblue")
```

```
# one sample t-test
# Ho: mu = 3.42
t.test(numeric.df$gpa,
       alternative = "two.sided",
       mu = 3.42,
       conf.level = 0.95,
       na.action = getOption("na.action")
       )
```

```
##
##  One Sample t-test
##
## data:  numeric.df$gpa
## t = -6.8513, df = 108, p-value = 4.686e-10
## alternative hypothesis: true mean is not equal to 3.42
## 95 percent confidence interval:
##  3.076144 3.230461
## sample estimates:
## mean of x
##  3.153303
```

```
sd(na.omit(numeric.df$gpa))
```

```
## [1] 0.4064026
```

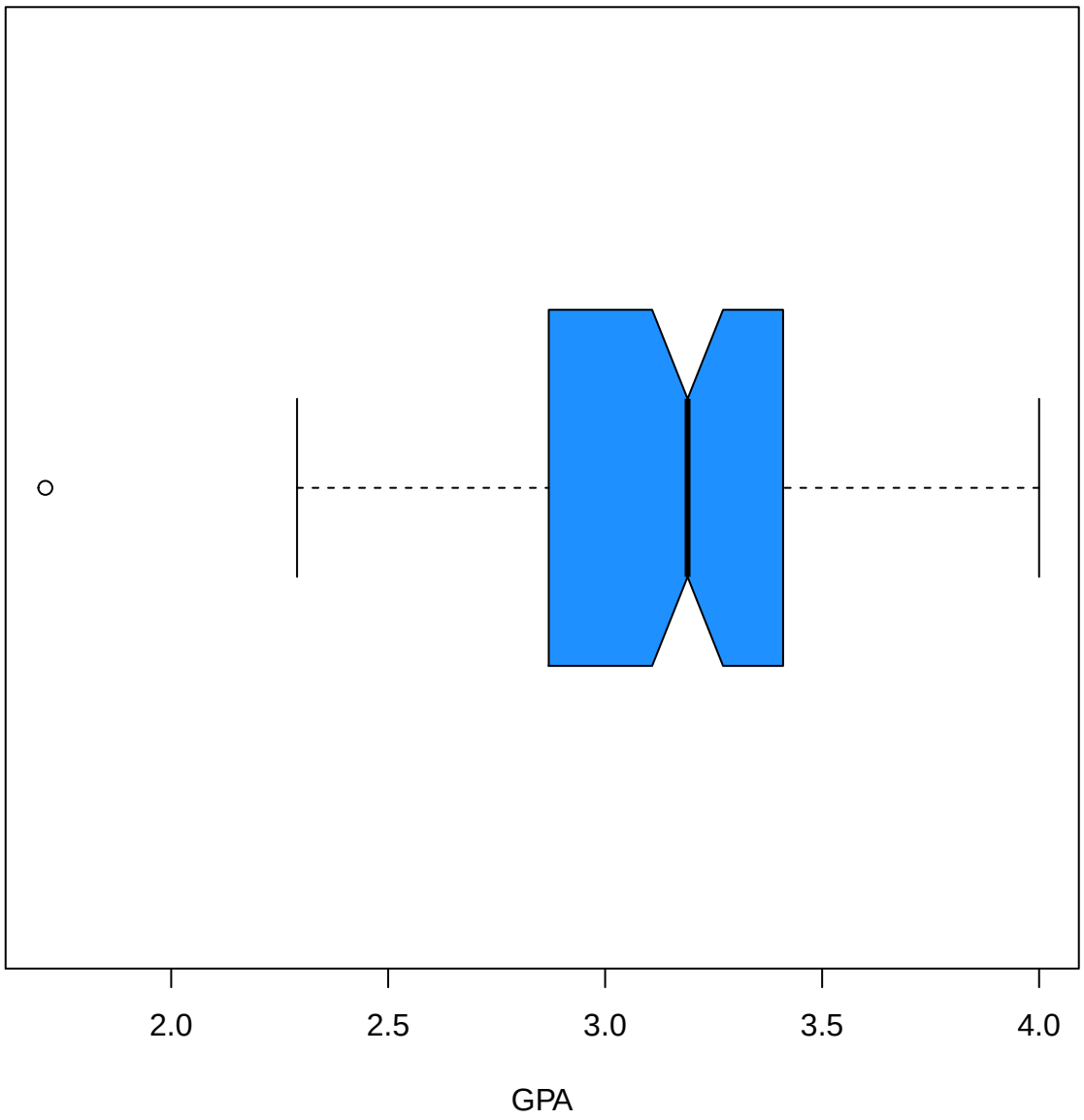

Figure 38:

4.9 Dependent (paired samples) t-test

Research Question: Is there a difference between graduate students' achievement in skills (i.e., Project Grades) and graduate students' achievement in knowledge (e.g., Final Exam scores)?

A paired samples t test was performed to assess whether mean project grades differed significantly compared with mean final grades with a group of 108 participants who took a graduate level introductory statistics course. Preliminary data screening indicated that final grades were multimodal, but the departure from normality was not judged serious enough to require the use of a nonparametric test. The mean difference of project and final grades differed significantly from zero, t (108) = 5.3341, p < 0.001}, two-tailed. Mean final grades (M = 80.18, SD = 14.95)} was about 7.28 percentage points lower than mean project grades (M = 87.46, SD = 5.59)}. The effect size, as indexed by Cohen's d, was .64; this is a very large effect. The 95% CI for the difference between sample means, $M_{project} - M_{final}$, had a lower bound of 4.24 and an upper bound of 10.33.

```
# Density plot of GPA vs a normal curve with the same parameters
x1 <- na.omit(numeric.df$final)
x2 <- na.omit(numeric.df$project)

# Kernel Density Plot
plot(density(x2),
     col = adjustcolor("dodgerblue", alpha.f = 0.8),
     lwd = 5,
     xlim = c(20, 110),
     main = "Normal Cuves (dotted) and \n Density Plots for Final and Project Grades")

# Kernel Density Plot
lines(density(x1),
     col = adjustcolor("tomato", alpha.f = 0.8),
     lwd = 5)

# means
abline(v = mean(x1),
       lty = 2,
       col = "tomato")

abline(v = mean(x2),
       lty = 2,
       col = "dodgerblue")

# Normal Curve
lines(seq(20, 100, length = 1000),
     dnorm(seq(20, 100, length = 1000), mean = mean(x1), sd = sd(x1)),
     col = adjustcolor("red", alpha.f = 0.8),
     lwd = 3,
     lty = 2)

# Normal Curve
lines(seq(20, 100, length = 1000),
     dnorm(seq(20, 100, length = 1000), mean = mean(x2), sd = sd(x2)),
     col = adjustcolor("blue", alpha.f = 0.8),
     lwd = 3,
     lty = 2)
```

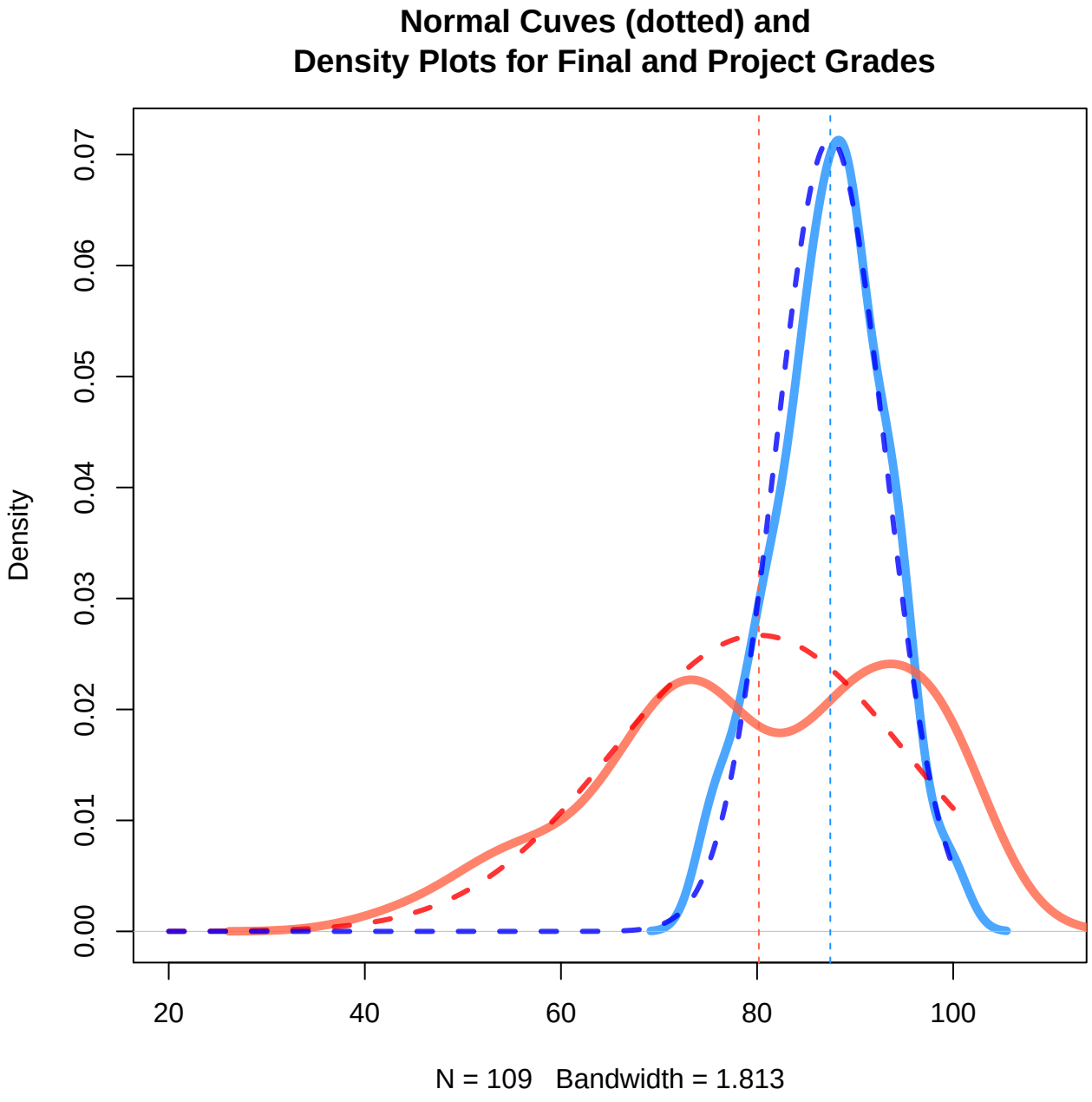
Normal Cuves (dotted) and
Density Plots for Final and Project Grades
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
20
40
60
80
100
N = 109 Bandwidth = 1.813

Figure 39:

```
# Add boxplots to a scatterplot
# scatterplot

par(fig=c(0,0.8,0,0.8), new=TRUE)
```

```
## Warning in par(fig = c(0, 0.8, 0, 0.8), new = TRUE): calling par(new=TRUE)
## with no plot
```

```
plot(numeric.df$project, numeric.df$final,
     xlab = "Project Grades",
     ylab = "Final Exam Scores"
     )

# Add fit lines
abline(lm(numeric.df$project ~ numeric.df$final),
       col="red") # regression line (y~x)
lines(lowess(numeric.df$project, numeric.df$final,
             delta = 0.01 * diff(range(numeric.df$final))
             ),
      col="blue") # lowess line (x,y)

# boxplots
par(fig=c(0,0.8,0.55,1), new = TRUE)
boxplot(numeric.df$project, horizontal = TRUE, axes = FALSE)
par(fig = c(0.65,1,0,0.8), new = TRUE)
boxplot(numeric.df$final, axes = FALSE)
mtext("Enhanced Scatterplot", side = 3, outer = TRUE, line = -3)
```

If the notches of two plots do not overlap this is strong evidence that the two medians differ (Chambers et al, 1983, p. 62).

Chambers, J. M., Cleveland, W. S., Kleiner, B. and Tukey, P. A. (1983) Graphical Methods for Data Analysis. Wadsworth & Brooks/Cole.

```
x <- na.omit(cbind(project = numeric.df$project,
                   final = numeric.df$final))

# Boxplot
boxplot(x, notch = TRUE)
```

```
# paired t-test
t.test(numeric.df$project, numeric.df$final,
       alternative = "two.sided",
       mu = 0,
       paired = TRUE,
       var.equal = TRUE,
       conf.level = 0.95,
       na.action = getOption("na.action"))
```

```
##
##  Paired t-test
```

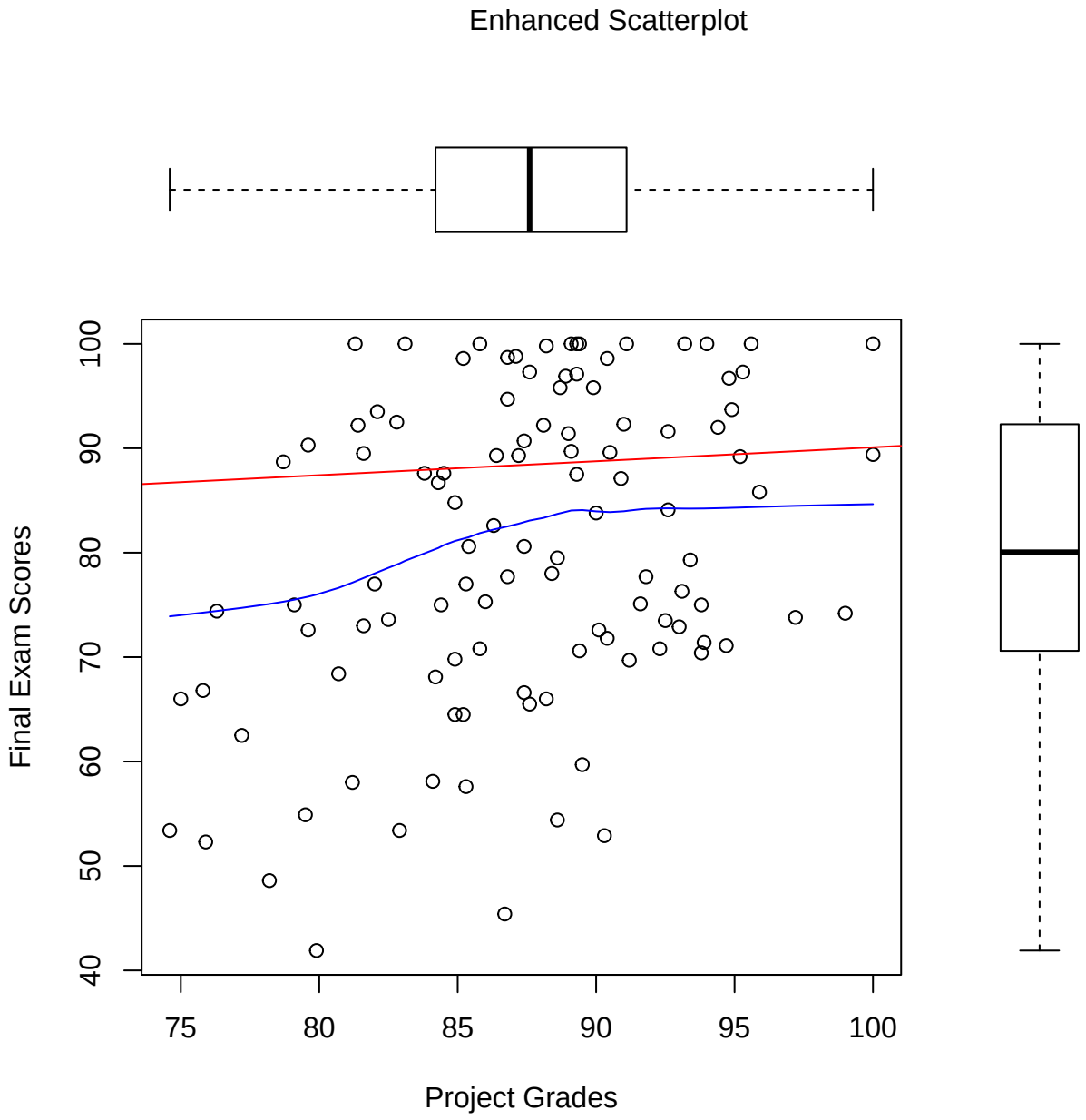
Enhanced Scatterplot
Final Exam Scores
Project Grades
40
50
60
70
80
90
100
75
80
85
90
95
100

Figure 40:

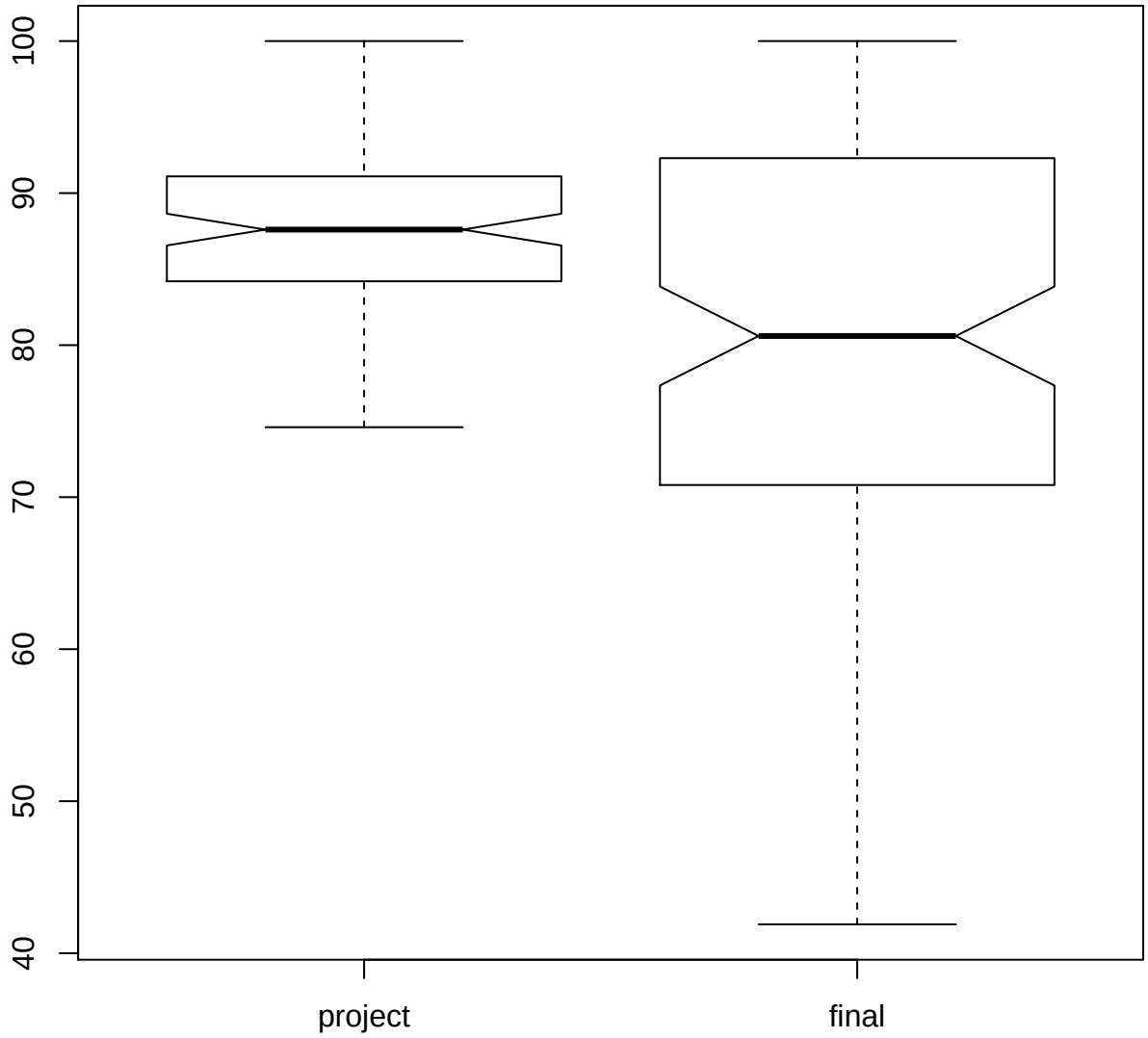

Figure 41:

```
##
## data:  numeric.df$project and numeric.df$final
## t = 5.3341, df = 108, p-value = 5.33e-07
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  4.480616 9.779934
## sample estimates:
## mean of the differences
##                7.130275
```

4.10 Cohen's d (Effect Size)

Examples of effect sizes are the correlation between two variables, the regression coefficient, and the mean difference. For each type of effect-size, a larger absolute value always indicates a stronger effect. Effect sizes complement statistical hypothesis testing, and play an important role in statistical power analyses.

Cohen's d is defined as the difference between two means divided by a standard deviation for the data. Cohen gives the following guidelines for the social sciences:

Effect size	Cohen's d
Small	$d < .20$
Medium	$0.20 < d < .79$
Large	$d > .80$

```
Jacob Cohen (1988). Statistical Power Analysis for the Behavioral Sciences (second ed.). Lawrence Erlbau
```

```r
# the denominator is length(x)+length(y)-2
cohens_d1 <- function(x, y){

    x <- na.omit(x)
    y <- na.omit(y)

    # the pooled standard deviation, as for two independent samples
    s <- sqrt(((length(x)-1)*var(x) + (length(y)-1)*var(y))/(length(x)+length(y)-2))

    # Cohen's d is defined as the difference between two means
    # divided by a standard deviation for the data
    d <- abs(mean(x) - mean(y))/s

    return(d)
}
```

```r
# Other authors choose a slightly different computation of the standard deviation
# when referring to "Cohen's d" where the denominator is length(x)+length(y)
cohens_d2 <- function(x, y){

    x <- na.omit(x)
    y <- na.omit(y)

    # the pooled standard deviation, as for two independent samples
    s <- sqrt(((length(x)-1)*var(x) + (length(y)-1)*var(y))/(length(x)+length(y)))
```

```
    # Cohen's d is defined as the difference between two means
    # divided by a standard deviation for the data
    d <- abs(mean(x) - mean(y))/s

    return(d)
}
```

```
# The standard error of the mean difference

se.m <- function(x, y){

    x <- na.omit(x)
    y <- na.omit(y)

    # the pooled variance, as for two independent samples
    s2 <- ((length(x)-1)*var(x) + (length(y)-1)*var(y))/(length(x)+length(y))

    return(sqrt(((s2)/(length(x)))+((s2)/(length(y)))))
}
```

```
# Confidence interval around the mean difference

ci <- function(x, y){

    x <- na.omit(x)
    y <- na.omit(y)

    m <- abs(mean(x)-mean(y))
    se <- se.m(x, y)

    return(list(mean.difference = m,
                standard.error.of.the.mean.difference = se,
                lower.ci = m - 2*se.m(x, y),
                upper.ci = m + 2*se.m(x, y)))
}
```

```
cohens_d1(numeric.df$final, numeric.df$project)
```

```
## [1] 0.6441565
```

```
cohens_d2(numeric.df$final, numeric.df$project)
```

```
## [1] 0.6471182
```

```
ci(numeric.df$final, numeric.df$project)
```

```
## $mean.difference
## [1] 7.28422
##
## $standard.error.of.the.mean.difference
## [1] 1.52129
```

```
##
## $lower.ci
## [1] 4.241641
##
## $upper.ci
## [1] 10.3268
```

```
## compute Cohen's d
effsize::cohen.d(na.omit(numeric.df$final), na.omit(numeric.df$project))
```

```
##
## Cohen's d
##
## d estimate: -0.6441565 (medium)
## 95 percent confidence interval:
##       inf       sup
## -0.918668 -0.369645
```

```
## compute Hedges' g
effsize::cohen.d(na.omit(numeric.df$final), na.omit(numeric.df$project), hedges.correction=TRUE)
```

```
##
## Hedges's g
##
## g estimate: -0.6419276 (medium)
## 95 percent confidence interval:
##        inf        sup
## -0.9163919 -0.3674633
```

4.11 Wilcoxon signed-rank test

The Wilcoxon signed-rank test is not the same as the Wilcoxon rank-sum test, although both are nonparametric and involve summation of ranks. The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used when comparing two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ (i.e. it is a paired difference test). It can be used as an alternative to the paired Student's t-test, t-test for matched pairs, or the t-test for dependent samples when the population cannot be assumed to be normally distributed.

4.12 Independent samples t-test, Mann–Whitney U test and Point biserial correlation

Research Question: Is there a difference between Mac users and PC users on Statistics Achievement (ACH) in the graduate population?

An independent samples t test was performed to assess whether mean statistics achievement scores differed significantly for a group of 50 participants who used Mac computers compared with a group of 56 participants who used PC computers. Preliminary data screening did not indicate that the distribution of grades either for the final or project depart from seriously from normality. The assumption of homogeneity of variance was assessed by the Levene test, F = 1.57, p = .226; this indicated no significant violation of the equal variance assumption; therefore, the pooled variances version of the t test was used. The mean statistics achievement scores of PC and Mac users did not differ significantly, (t(104) = -1.22, p = 0.2265, two-tailed).

Mean statistics achievement grades for the PC users (M = 84.90, SD = 7.90) was about 2 percentage points higher than mean scores for the Mac users (M = 82.78, SD = 10.01). The effect size, as indexed by Chen's d, was 0.24. The 95% CI for the difference between sample means, $M_{PC} - M_{Mac}$, had a lower bound of -1.33 and an upper bound of 5.57, which included 0. This study suggests that computer type has no effect on the statistics achievement scores.

```
psych::describeBy(numeric.df$achievement, numeric.df$computer)
```

```
## group: Mac
##   vars  n  mean    sd median trimmed   mad  min max range  skew kurtosis
## 1    1 50 82.78 10.01  83.85   83.29 11.12 60.9 100  39.1 -0.37    -0.84
##     se
## 1 1.42
## ------------------------------------------------------------
## group: PC
##   vars  n mean  sd median trimmed  mad   min max range  skew kurtosis   se
## 1    1 56 84.9 7.9  85.17   85.49 8.41 66.05  97 30.95 -0.54    -0.52 1.06
```

```
#boxplots
plot(numeric.df$achievement ~ numeric.df$computer, data = numeric.df)
```

```
# Density plot of GPA vs a normal curve with the same parameters
x1 <- numeric.df$achievement[which(numeric.df$computer == "PC" & !is.na(numeric.df$achievement))]
x2 <- numeric.df$achievement[which(numeric.df$computer == "Mac" & !is.na(numeric.df$achievement))]

# Kernel Density Plot
plot(density(x1),
     col = adjustcolor("dodgerblue", alpha.f = 0.8),
     lwd = 5,
     ylim = c(0, 0.05),
          main = "Normal Cuves (dotted) and \n Density Plots of Achievement Grades \n for Mac (Red) and

# Distribution lines
abline(v = c(mean(x1)-2*sd(x1),
             mean(x1)-sd(x1),
             mean(x1),
             mean(x1)+sd(x1),
             mean(x1)+2*sd(x1)),
       lty = 2,
       col = "dodgerblue")

# Kernel Density Plot
lines(density(x2),
     col = adjustcolor("tomato", alpha.f = 0.8),
     lwd = 5)

# Distribution lines
abline(v = c(mean(x2)-2*sd(x2),
             mean(x2)-sd(x2),
             mean(x2),
             mean(x2)+sd(x2),
             mean(x2)+2*sd(x2)),
```

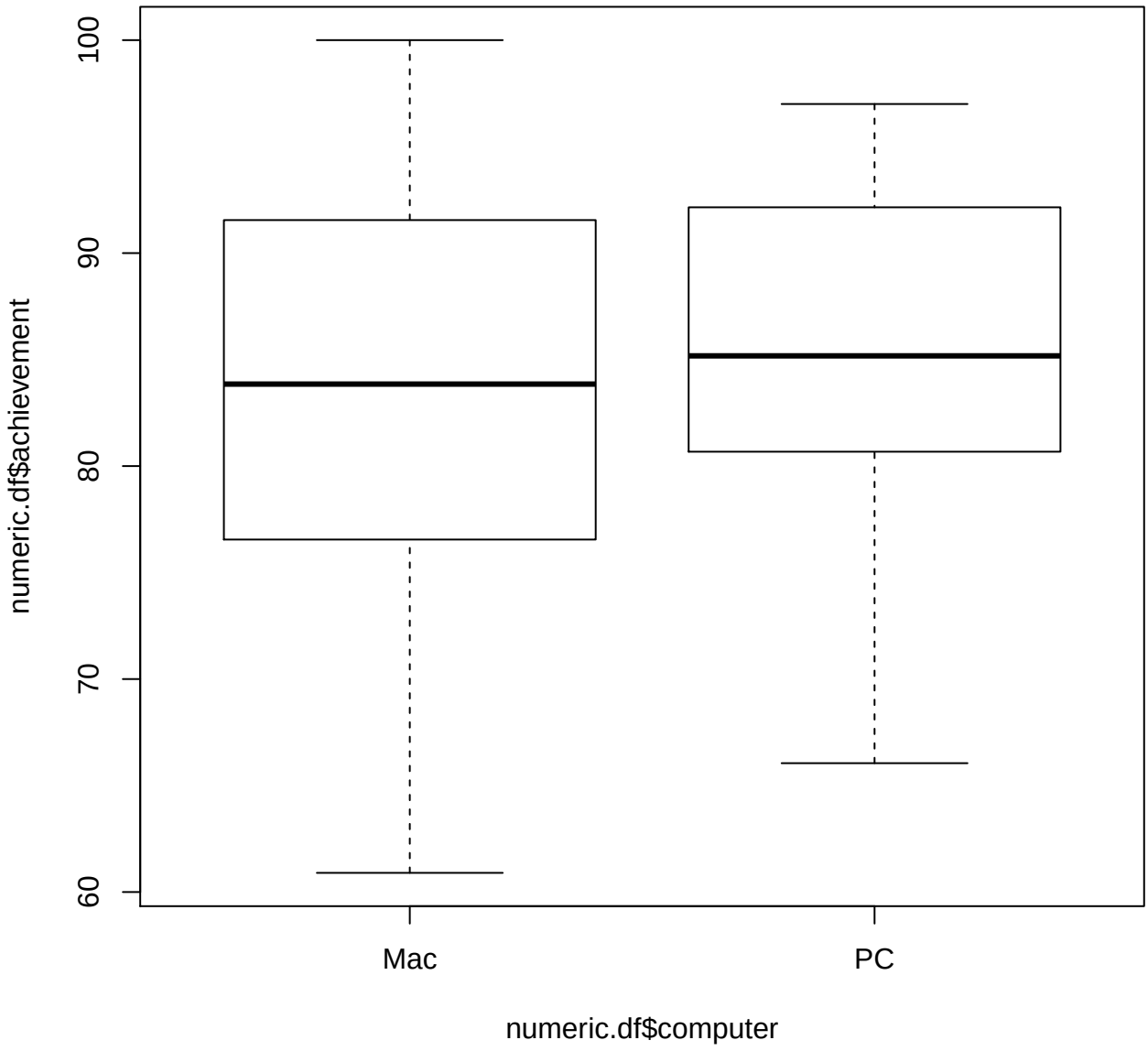
100
90
80
70
60
numeric.df$achievement
Mac
PC
numeric.df$computer

Figure 42:

```
       lty = 2,
       col = "tomato")

# Normal Curve
lines(seq(20, 100, length = 1000),
     dnorm(seq(20, 100, length = 1000), mean = mean(x1), sd = sd(x1)),
     col = adjustcolor("blue", alpha.f = 0.8),
     lwd = 3,
     lty = 2)

# Normal Curve
lines(seq(20, 100, length = 1000),
     dnorm(seq(20, 100, length = 1000), mean = mean(x2), sd = sd(x2)),
     col = adjustcolor("red", alpha.f = 0.8),
     lwd = 3,
     lty = 2)
```

```
t.test(numeric.df$achievement ~ numeric.df$computer,
       alternative = "two.sided",
       mu = 0,
       paired = FALSE,
       var.equal = TRUE,
       conf.level = 0.95,
       na.action = getOption("na.action"))
```

```
##
##  Two Sample t-test
##
## data:  numeric.df$achievement by numeric.df$computer
## t = -1.2166, df = 104, p-value = 0.2265
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.574841  1.335484
## sample estimates:
## mean in group Mac  mean in group PC
##          82.78300          84.90268
```

4.12.1 Mann–Whitney U test

Mann–Whitney U test (also called the Mann–Whitney–Wilcoxon (MWW), Wilcoxon rank-sum test (WRS), or Wilcoxon–Mann–Whitney test) is a nonparametric test of the null hypothesis that two samples come from the same population against an alternative hypothesis, especially that a particular population tends to have larger values than the other. It has greater efficiency than the t-test on non-normal distributions, such as a mixture of normal distributions, and it is nearly as efficient as the t-test on normal distributions.

```
# independent 2-group Mann-Whitney U Test
wilcox.test(numeric.df$achievement ~ numeric.df$computer,
            alternative = "two.sided",
            mu = 0,
            paired = FALSE,
            exact = NULL,
```

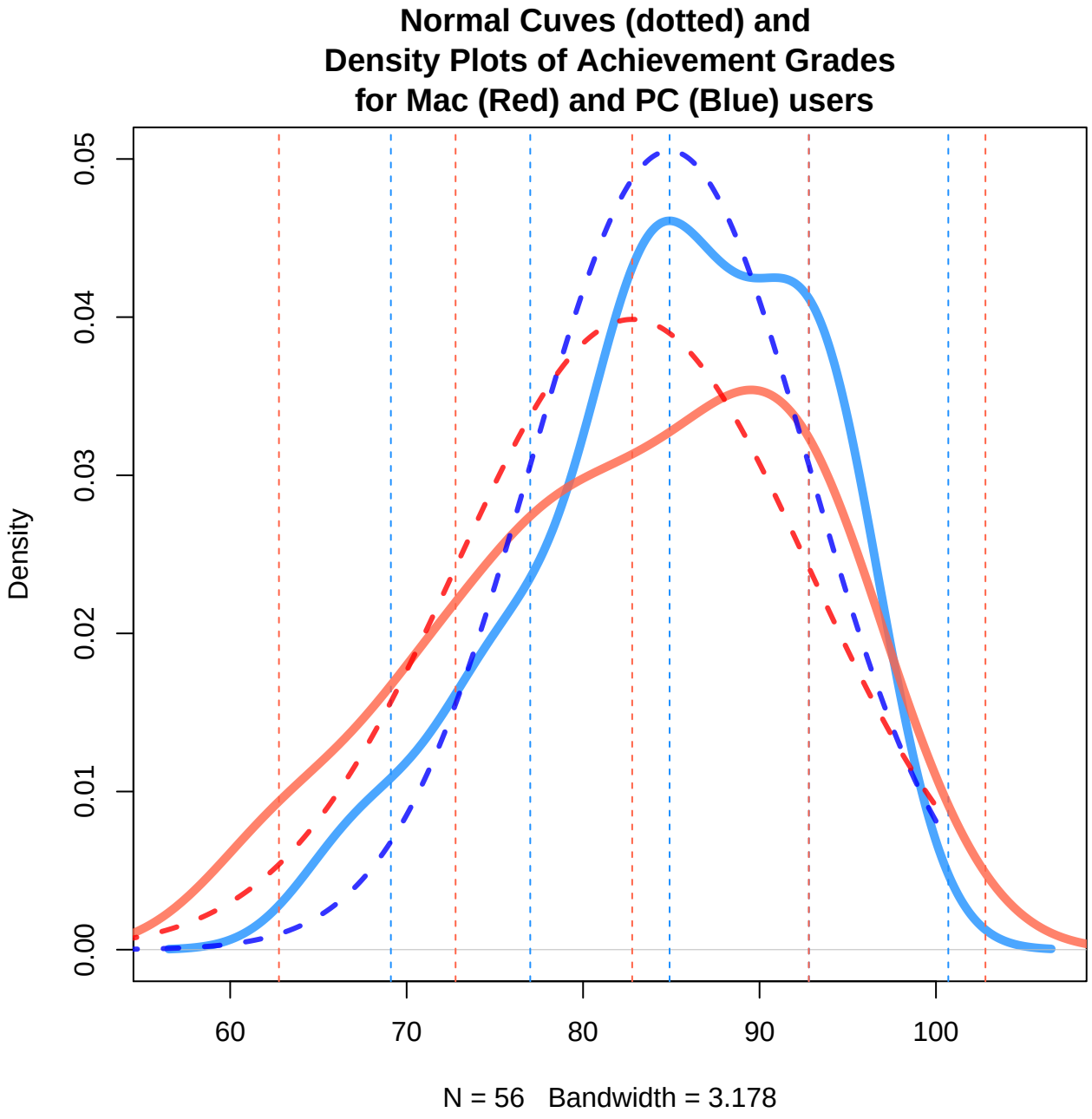
Normal Cuves (dotted) and
Density Plots of Achievement Grades
for Mac (Red) and PC (Blue) users
Density
0.00
0.01
0.02
0.03
0.04
0.05
60
70
80
90
100
N = 56 Bandwidth = 3.178

Figure 43:

```
            correct = TRUE,
            conf.int = TRUE,
            conf.level = 0.95)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  numeric.df$achievement by numeric.df$computer
## W = 1237.5, p-value = 0.3052
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -5.350011  1.750069
## sample estimates:
## difference in location
##              -1.762802
```

```
ci(x1, x2)
```

```
## $mean.difference
## [1] 2.119679
##
## $standard.error.of.the.mean.difference
## [1] 1.725843
##
## $lower.ci
## [1] -1.332007
##
## $upper.ci
## [1] 5.571365
```

```
effsize::cohen.d(x1, x2)
```

```
##
## Cohen's d
##
## d estimate: 0.2367044 (small)
## 95 percent confidence interval:
##        inf        sup
## -0.1542091  0.6276180
```

4.12.2 Point-biserial correlation coefficient

The point biserial correlation coefficient (r_{pb}) is a correlation coefficient used when one variable (e.g. Y) is dichotomous; Y can either be "naturally" dichotomous, like gender, or an artificially dichotomized variable. In most situations it is not advisable to dichotomize variables artificially. When you artificially dichotomize a variable the new dichotomous variable may be conceptualized as having an underlying continuity. If this is the case, a biserial correlation would be the more appropriate calculation.

The point-biserial correlation is mathematically equivalent to the Pearson (product moment) correlation, that is, if we have one continuously measured variable X and a dichotomous variable Y, $r_{XY} = r_{pb}$. This can be shown by assigning two distinct numerical values to the dichotomous variable.

4.13 One-way ANOVA and Kruskal-Wallis

One Way Anova (Completely Randomized Design): In statistics, one-way analysis of variance (abbreviated one-way ANOVA) is a technique used to compare means of two or more samples (using the F distribution). This technique can be used only for numerical data. The ANOVA tests the null hypothesis that samples in two or more groups are drawn from populations with the same mean values. To do this, two estimates are made of the population variance.

Research Question: Is there a mean difference between graduate students from the different majors on Statistics Anxiety (SA)?

A one-way between-subjects ANOVA was done to compare the mean scores on an anxiety scale (0 = not at all anxious, 25 = extremely anxious) for participants who were assigned to one of three groups according to their academic major: Social Science, Physical Science and Other majors. Examination of a histogram of anxiety scores indicated that the scores were approximately normally distributed with no extreme outliers. Prior to the analysis, the Levene test for homogeneity of variance was used to examine whether there were serious violations of the homogeneity of variance assumption across groups, but no significant violation was found: F(2, 104) = 0.157, p = 0.8549. The overall F for the one-way ANOVA was not statistically significant, F(1, 105) = 1.8419, p = 0.1776. This corresponded to an effect size of $\eta^2 = 0.017$; that is, about 2% of the variance in anxiety scores was predictable from the type of academic major. This is a very small effect. The means and standard deviations for the three groups are shown in the plot above. In addition, all possible pairwise comparisons were made using the Tukey HSD test. Based on this test (using $\alpha = .05$), it was found that none of the academic majors had a mean anxiety score that was statistically significantly different. Figure above shows a 95% CI around each group mean.

4.14 Omnibus Tests:

4.14.1 Bartlett test of homogeneity of variances

```
stats::bartlett.test(anxiety ~ major, data = numeric.df)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  anxiety by major
## Bartlett's K-squared = 0.36197, df = 2, p-value = 0.8344
```

4.14.2 Levene's Test for Homogeneity of Variance

```
car::leveneTest(numeric.df$anxiety, numeric.df$major)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   2  0.1018 0.9033
##       104
```

```
fit <- aov(anxiety ~ major, data = numeric.df)
summary(fit)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## major         2   22.5   11.24   0.928  0.399
## Residuals   104 1259.4   12.11
## 3 observations deleted due to missingness
```

```
#report the means and the number of subjects/cell
print(model.tables(fit ,"means"), digits = 4)
```

```
## Tables of means
## Grand mean
##
## 15.8972
##
##  major
##     Other Major Physical Science Social Science
##           16.44            15.98          15.27
## rep       32.00            42.00          33.00
```

```
qqplot(as.factor(numeric.df$major), numeric.df$anxiety,
      data = numeric.df,
      geom = c("boxplot", "jitter"),
      main ="Anxiety by Major",
      ylab="Statistics Anxiety",
      xlab = "Major")
```

```
# display Type II (hierarchical or partially sequential) Sums of Squares ANOVA table
# using the 'car' package

car::Anova(lm(anxiety ~ major,
         data = numeric.df,
         type = "II",
         test.statistic = c("LR", "Wald", "F"),
         error.estimate = c("pearson", "dispersion", "deviance")))
```

```
## Warning in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...):
## extra arguments c("'type'", "'test.statistic'", "'error.estimate'") are
## disregarded.

## Anova Table (Type II tests)
##
## Response: anxiety
##            Sum Sq  Df F value Pr(>F)
## major       22.47   2  0.9279 0.3986
## Residuals 1259.40 104
```

```
# One Way Anova (Completely Randomized Design)
fit <- aov(anxiety ~ major, data = numeric.df)
summary(fit)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## major         2   22.5   11.24   0.928  0.399
## Residuals   104 1259.4   12.11
## 3 observations deleted due to missingness
```

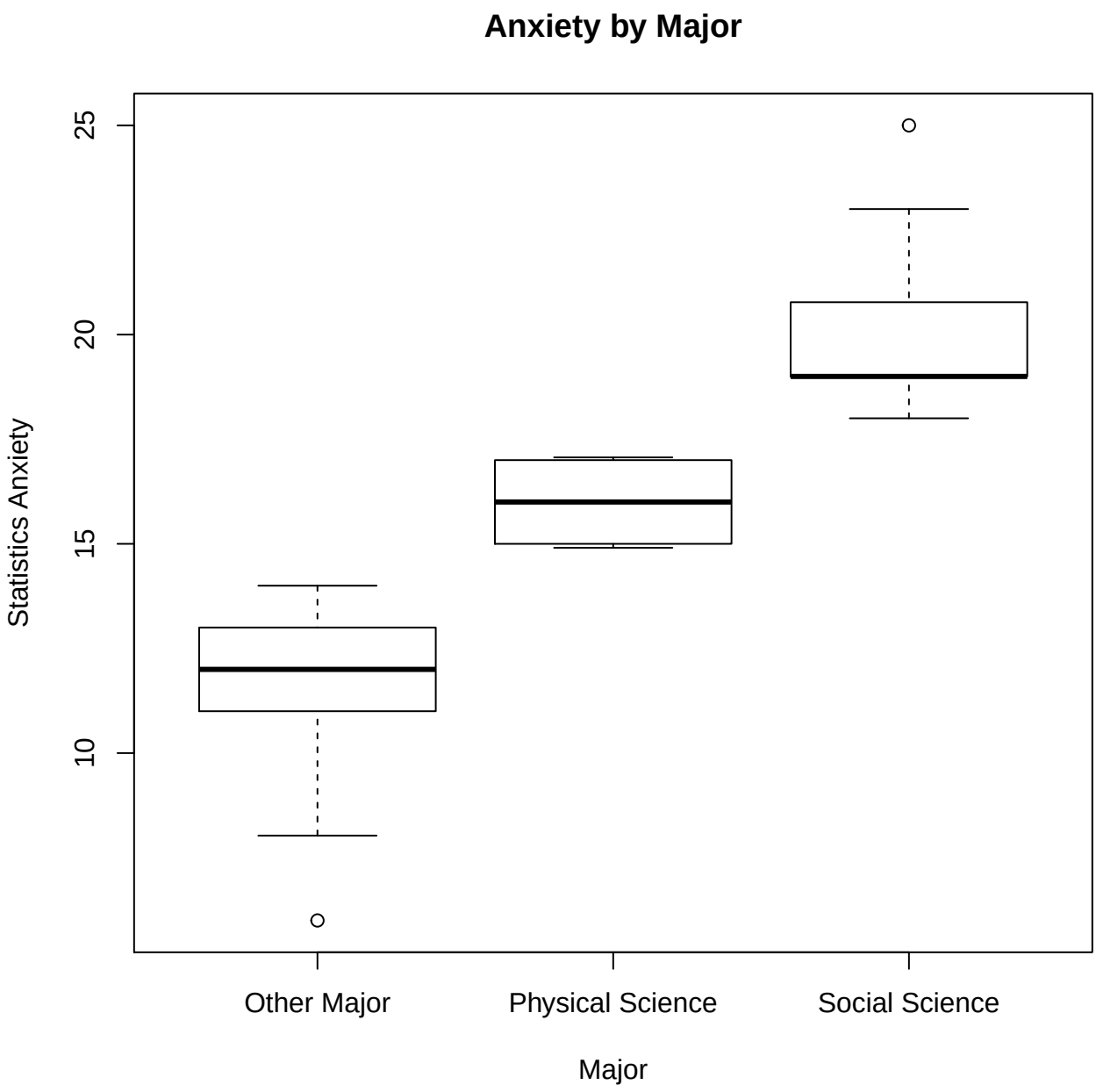
Anxiety by Major
Statistics Anxiety
10
15
20
25
Other Major
Physical Science
Social Science
Major

Figure 44:

```
layout(matrix(c(1,2,3,4),2,2)) # optional layout
plot(fit) # diagnostic plots
```

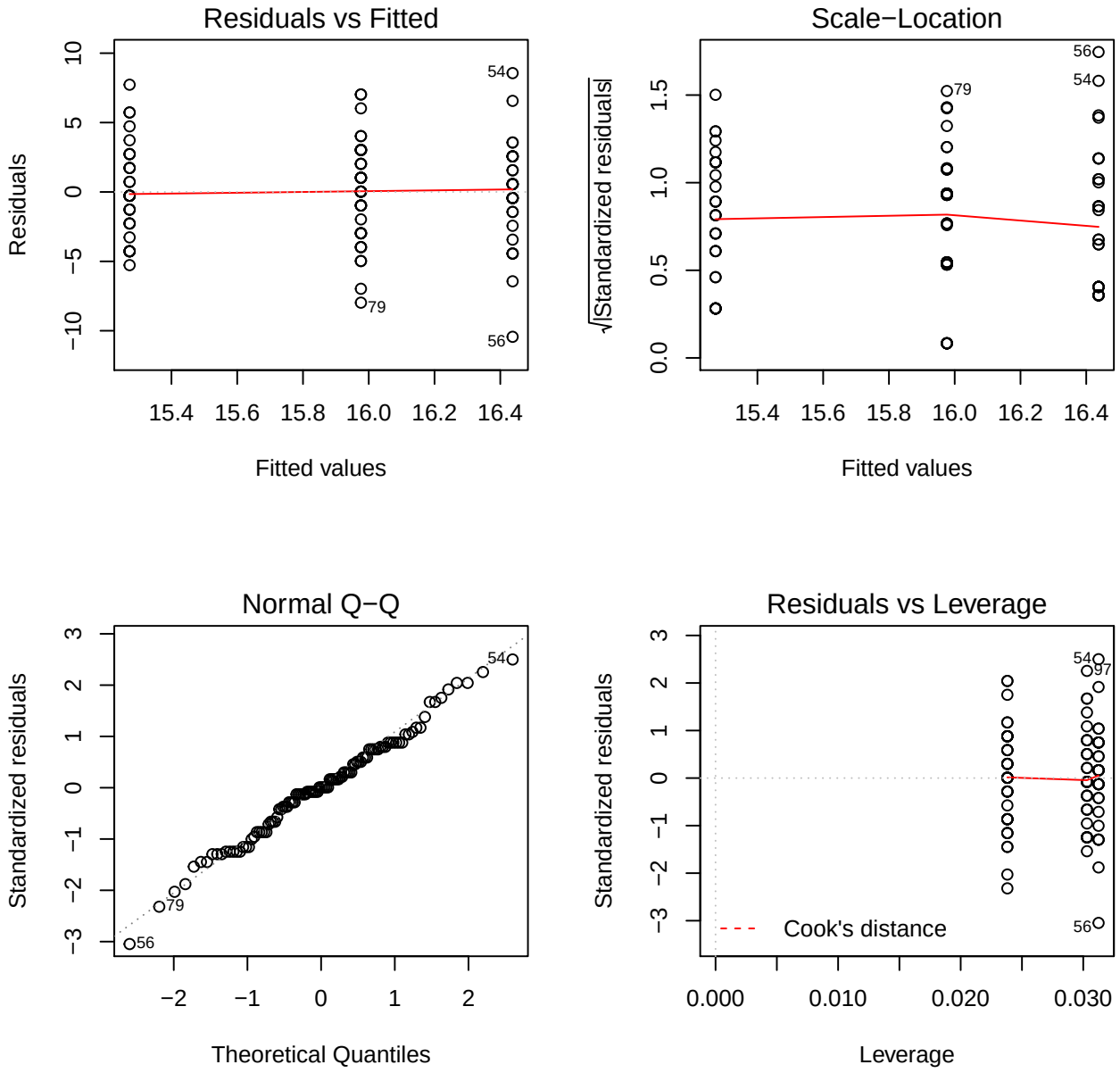

Figure 45:

```
drop1(fit,~., test="F") # type III SS and F Tests
```

```
## Single term deletions
##
## Model:
## anxiety ~ major
##         Df Sum of Sq    RSS    AIC F value Pr(>F)
## <none>               1259.4 269.81
## major    2    22.473 1281.9 267.71  0.9279 0.3986
```

```r
# P-values larger than 0.05, so none of the group statistical anxiety means are
# statistically significantly different from eachother
pairwise.t.test(numeric.df$anxiety, numeric.df$major, p.adjust = "bonferroni", alternative = c("two.sid
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  numeric.df$anxiety and numeric.df$major
##
##                  Other Major Physical Science
## Physical Science 1.00        -
## Social Science   0.54        1.00
##
## P value adjustment method: bonferroni
```

```r
summary.lm(fit)
```

```
##
## Call:
## aov(formula = anxiety ~ major, data = numeric.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4375  -2.2727   0.0238   2.5625   8.5625
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            16.4375     0.6152  26.721   <2e-16 ***
## majorPhysical Science  -0.4613     0.8165  -0.565    0.573
## majorSocial Science    -1.1648     0.8634  -1.349    0.180
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.48 on 104 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.01753,    Adjusted R-squared:  -0.001363
## F-statistic: 0.9279 on 2 and 104 DF,  p-value: 0.3986
```

4.15 Post-Hoc Tests:

4.15.1 Tukey's HSD (honest significant difference)

```r
TukeyHSD(fit, ordered = TRUE, conf.level = 0.95)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##     factor levels have been ordered
##
## Fit: aov(formula = anxiety ~ major, data = numeric.df)
##
```

```
## $major
##                                       diff        lwr      upr     p adj
## Physical Science-Social Science 0.7034632 -1.2212976 2.628224 0.6609750
## Other Major-Social Science      1.1647727 -0.8880548 3.217600 0.3714896
## Other Major-Physical Science    0.4613095 -1.4802199 2.402839 0.8389943
```

4.15.2 Means Plot

```
#  Means Plot with Error Bars

#png(filename="MeanPlot.png")
gplots::plotmeans(anxiety ~ major,
          data = numeric.df,
          xlab = "Major",
          ylab = "Anxiety",
          ylim = c(14, 18),
          mean.labels = TRUE,
          main = "Mean Plot\nwith 95% CI",
          barwidth = 10,
          col = adjustcolor("red", alpha.f = 0.5),
          ccol = adjustcolor("tomato", alpha.f = 0.5),
          lwd = 10,
          barcol = adjustcolor("dodgerblue", alpha.f = 0.5),
          na.action = na.omit,
          bars = TRUE,
          p = 0.95,
          minsd = 0,
          ci.label = TRUE,
          n.label = TRUE,
          legends = c("Social Science", "Phsical Science", "Other"),
          digits = 3)
```

```
#dev.off()
```

4.16 Kruskal-Wallis one-way analysis of variance

The Kruskal-Wallis one-way analysis of variance by ranks (named after William Kruskal and W. Allen Wallis) is a non-parametric method for testing whether samples originate from the same distribution. It is used for comparing two or more samples that are independent, and that may have different sample sizes, and extends the Mann-Whitney U test to more than two groups. The parametric equivalent of the Kruskal-Wallis test is the one-way analysis of variance (ANOVA).

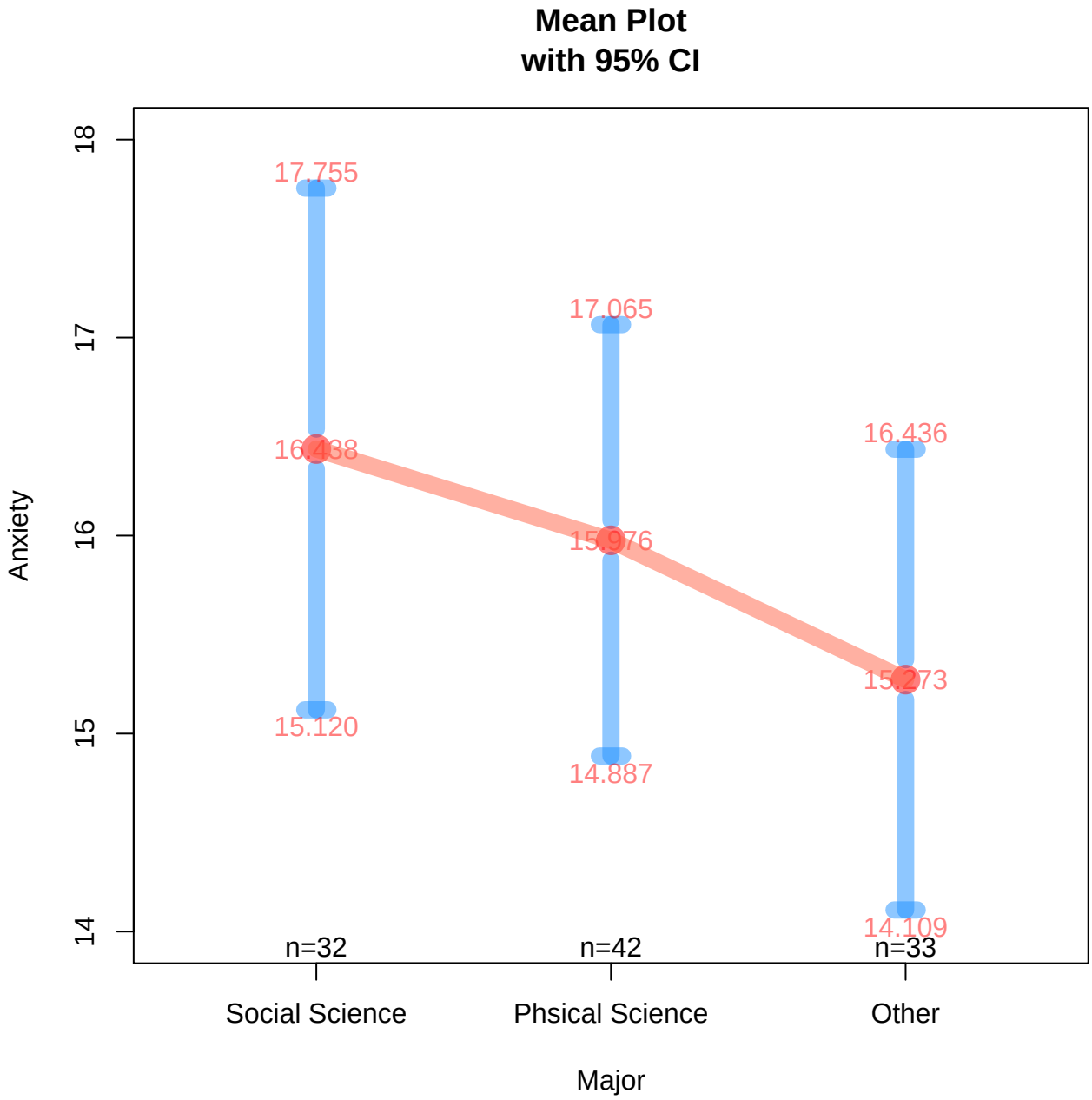
Mean Plot
with 95% CI
Anxiety
18
17
16
15
14
17.755
17.065
16.436
16.438
15.976
15.273
15.120
14.887
14.109
n=32
n=42
n=33
Social Science
Phsical Science
Other
Major

Figure 46: