(In black under different title colors: questions/problems/requirements of the project)
(In red: work that are not from the original R code but are from later on insertion by typing)
(In blue: work that are from the original R code)
(In purple: coding results from R console)
(In order to clearly show my work part by part, the plots related to the certain piece of code are inserted right after the code pieces, and statements/comments/explanations are followed.)

Data description:

The data "diabetes.txt" contains 16 variables on 366 subjects who were interviewed in a study to understand the prevalence of obesity, diabetes, and other cardiovascular risk factors in central Virginia for African Americans. We will consider building regression models with glyhb as the response variable as Glycosylated Hemoglobin > 70 is often taken as a positive diagnostics of diabetes. The goal is to find the "best" model for later use.

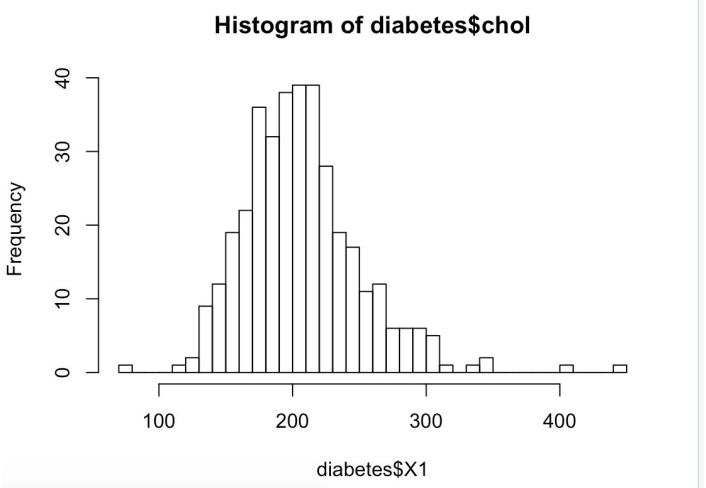
Data exploration and split data for validation later on.

#1

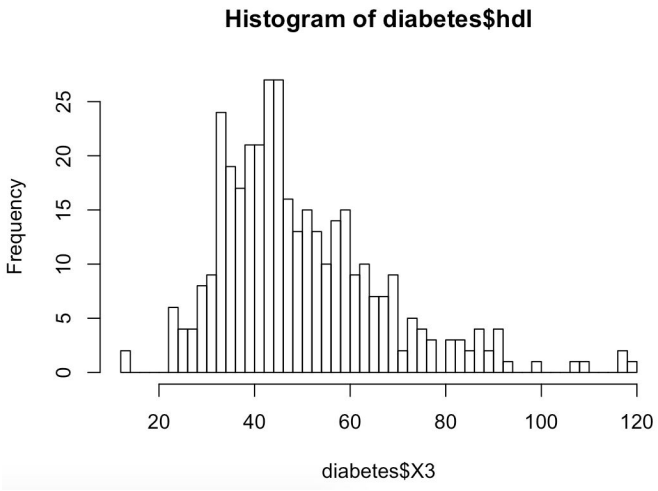
Among all the variable, which of the variables are quantitative variables?
Which are qualitative variables?
Draw histogram for each quantitative variable and comment on its distribution.
Draw pie chart for each qualitative variable and comment on how its classes are distributed.
Draw scatterplot matrix and obtain the pairwise correlation matrix for all quantitative variables in the data.
Comment on their relationships.

Ans:

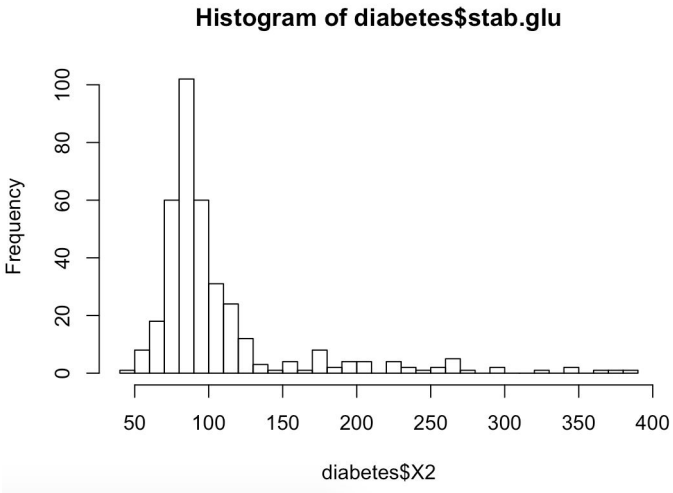
- Quantitative variables: (numerical) 12 in total
  chol, stab.glu, hdl, ratio, age, height, weight, bp.1s, bp.1d, waist, hip, time.ppn
- Qualitative variables: (categorical) 3 in total
  location, gender, frame
- Histograms, pie charts and comments.
- Scatter plot matrix, pairwise correlation matrix, and comments.



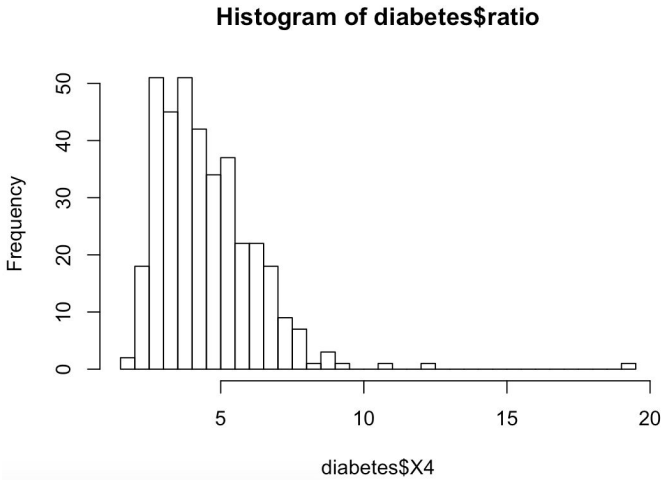
Comment:
The distribution of X1 is approximately symmetric. It has a bell shape with the peak occurs slightly to the left of the center of the graph at around values 200. There are couple outliers on the very left and the very right tail.



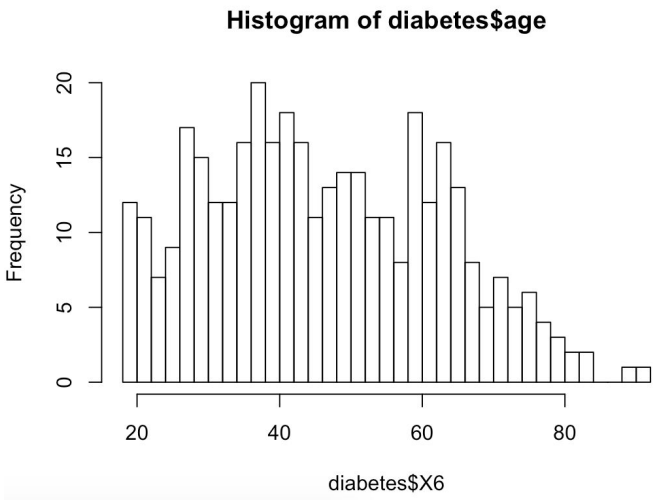
Comment:
The distribution of X3 is skewed right. The data concentrate on the left around the value 45 and to the right its tail is longer. It has two peaks, so it's bimodal.



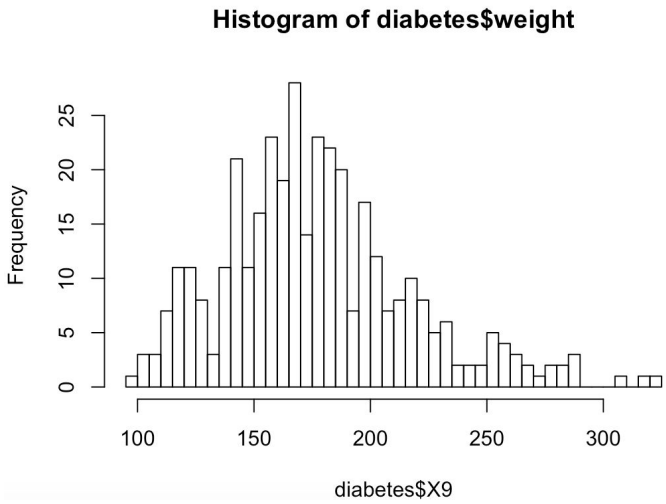
Comment:
The distribution of X2 is skewed right. The data concentrate on the left around the value 80 and to the right it has a long tail, meaning data spread further on the right. There is only one peak, so it's unimodal.



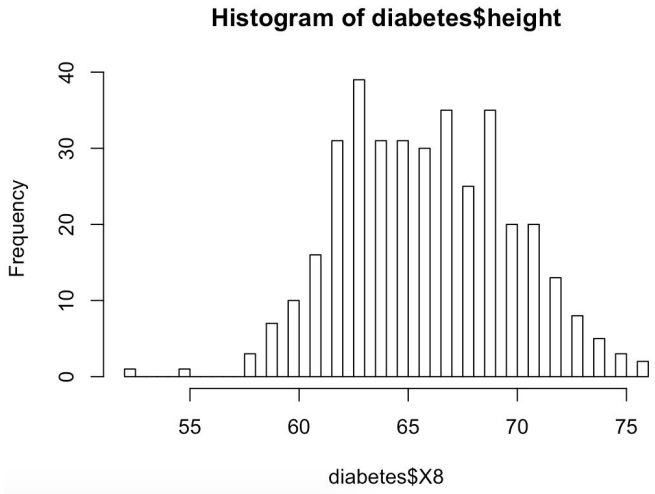
Comment:
The distribution of X4 is obviously skewed right. The data concentrate on the whole left part of the graph and there are three outliers on the right tail including one small part far away from the center of the data concentration.



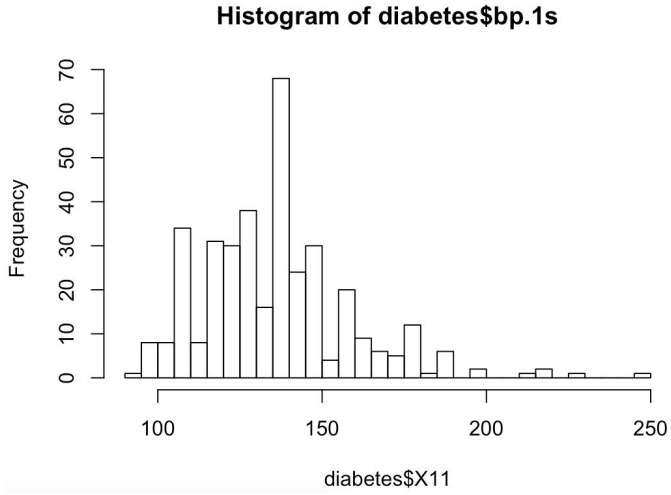
Comment:
The distribution of X6 is approximately symmetric but with couple peaks occurring at values around 20, 30, 40, 50 and 60. It's multimodal distribution.



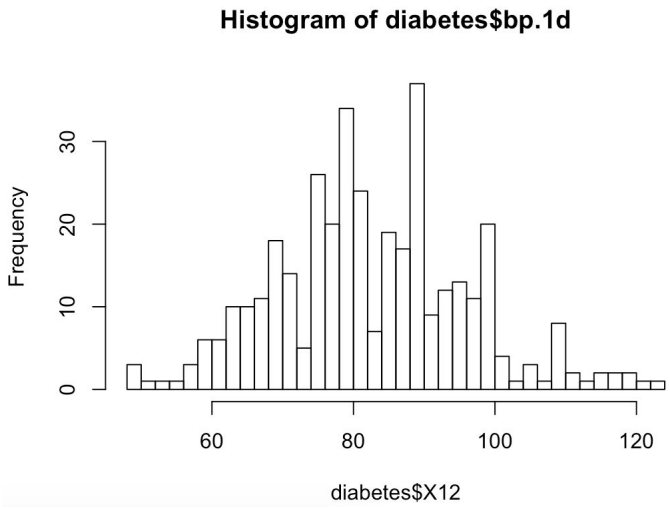
Comment:
The distribution of X9 is approximately symmetric with a bell shape. The data concentrate slightly on the left of the graph. There are two peaks at value around 140 and 170, so it's bimodal distribution. The right tail is a little bit longer, meaning data spread a little bit further on the right. And it seems there are couple outliers at the end of the right tail.



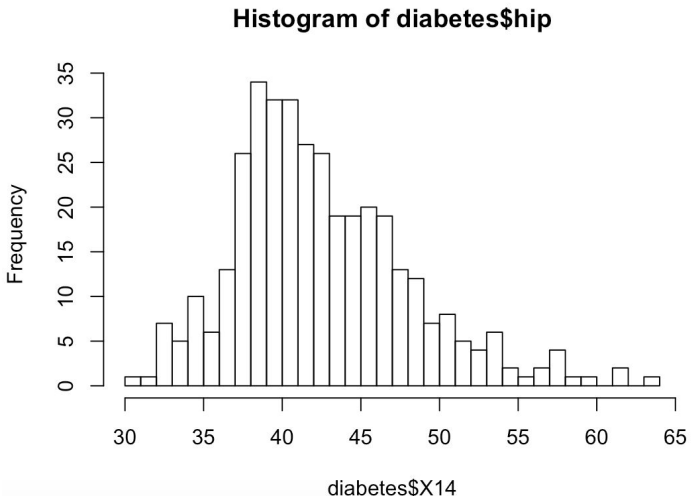
Comment:
The distribution of X8 is approximately symmetric with a bell shape. The data concentrate around the center of the graph. Two outliers occur to the left of the graph. Peaks occur around the center of the graph.



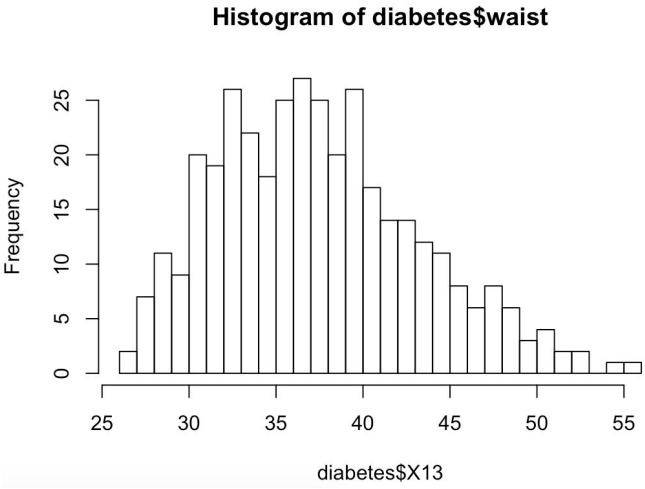
Comment:
The distribution of X11 is right skewed and unimodal. The data concentrate on the left of the graph, and the peak occurs at value around 140. The right tail is long and plat, meaning data spread further on the right. And it seems there is an outlier on the end of the right tail.



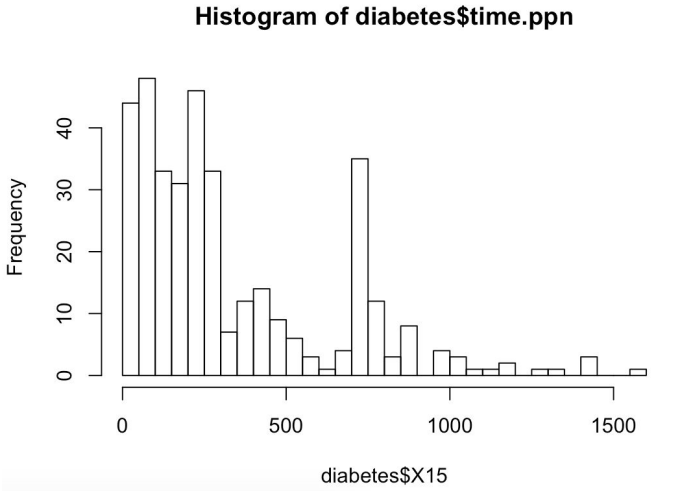
Comment:
The distribution of X12 is approximately symmetric with a bell shape. The data concentrate around the center of the graph and both tails are at the similar length, meaning data spread evenly on the tails. There are two peaks occurring, one is at the value of around 80, another is at around 100, between the peaks, the data values are lower and lower towards the middle point, so it's bimodal distribution.



Comment:
The distribution of X14 is skewed right. The data concentrate on the left of the graph where the peak occurs at around value 40, so it's unimodal. The right tail is longer that the left, meaning data spread further on the right.

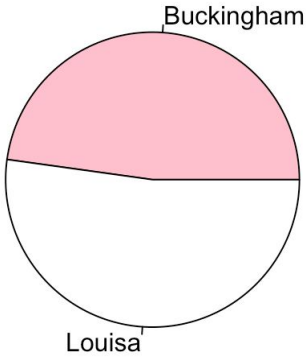


Comment:
The distribution of X13 is slightly skewed right. There are more data concentrated to the left of the graph and the right tail is longer than the left tail, meaning data spread further on the right.



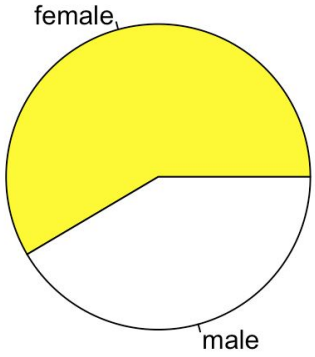
Comment:
The distribution of X15 is obviously skewed right and multimodal. The data concentrate on the left of the graph but peaks occur at around the value 0, 150, and 750. The right tail is long and flat where the left seems have no tail.

Pie Chart of diabetes$location



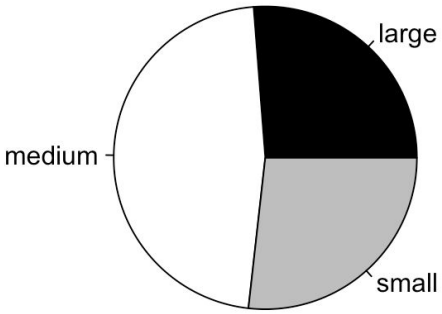
Comment:
There are two classes in this pie chart, “Buckingham” and “Louisa”.. They have similar proportions, but Louisa’s is a little bit larger than Bunkingham’s.

Pie Chart of diabetes$gender

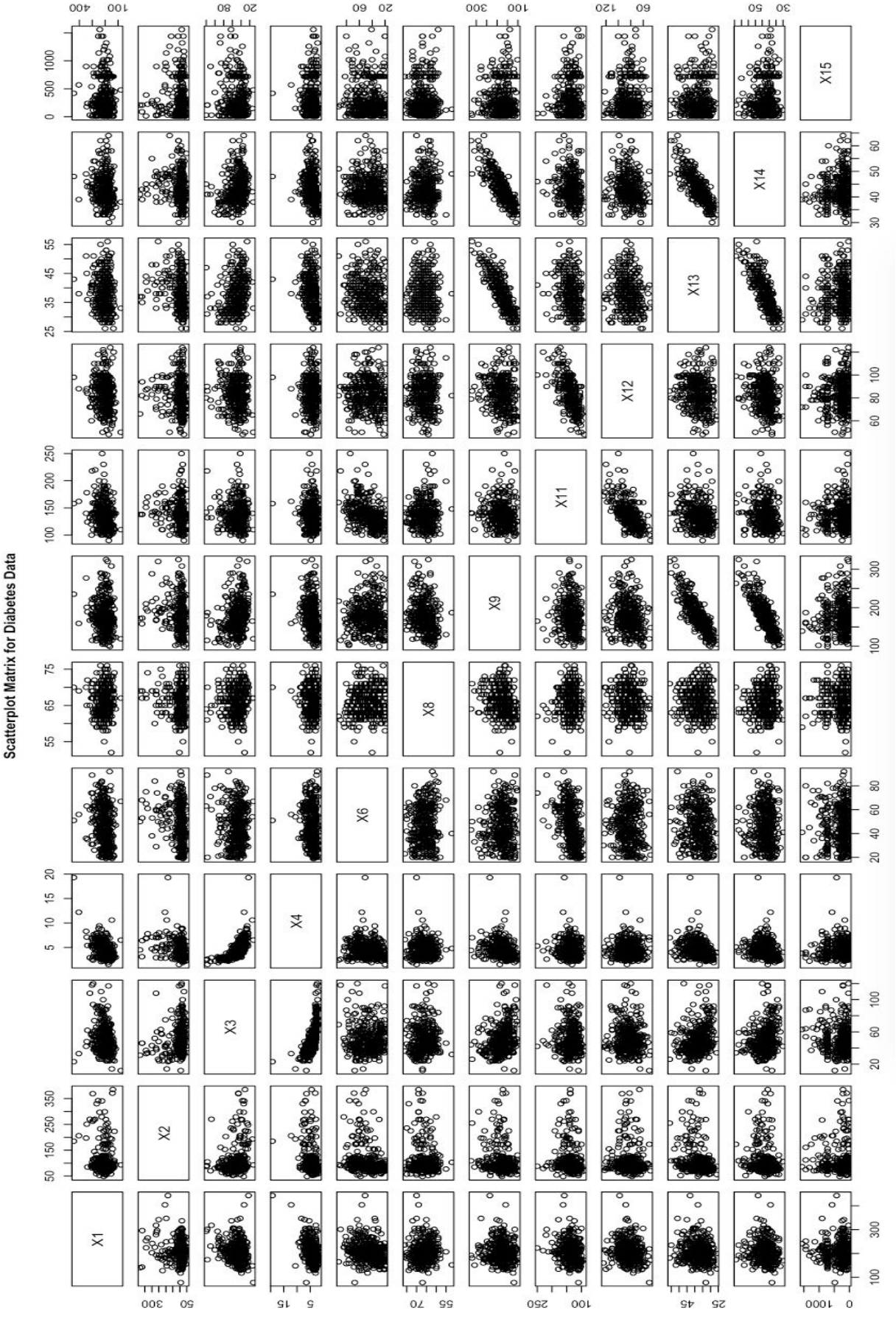


Comment:
There are two classes in this pie chart, female and male. “female” has a larger proportion than “male” with distribution on almost 7/12 of the pie.

Pie Chart of diabetes$frame



Comment:
There are three classes in this pie chart, “small”, “medium” and “large”. “medium” has the largest proportion among them, while “large” and “small” have the similar quarter proportions.


Scatterplot Matrix for Diabetes Data
X1
X2
X3
X4
X6
X8
X9
X11
X12
X13
X14
X15

Untitled1* | Untitled2* | Untitled3* | cor(diabetes[, -factor]) | diabetes | diabetesoriginal

Filter

	Y	X1	X2	X3	X4	X6	X8	X9	X11	X12	X13	X14	X15
Y	1.00000000	0.271652180	0.74090490	-0.1694964128	0.35465342	0.3322298936	0.052250721	0.16776851	0.19442279	0.04786459	0.24768684	0.15167273	0.037049379
X1	0.27165218	1.000000000	0.16544754	0.1709732770	0.48403807	0.2416049084	-0.063230009	0.07978999	0.20194870	0.15904230	0.14408955	0.09859715	0.006238501
X2	0.74090490	0.165447544	1.00000000	-0.1801048833	0.29889570	0.2785514141	0.082475702	0.18880052	0.15142542	0.02569721	0.23369209	0.14483314	-0.048457737
X3	-0.16949641	0.170973277	-0.18010488	1.0000000000	-0.69023141	0.0002152264	-0.068591817	-0.28298268	0.02950891	0.07224515	-0.27830010	-0.22221661	0.079938843
X4	0.35465342	0.484038069	0.29889570	-0.6902314087	1.00000000	0.1715691447	0.070898165	0.27889889	0.10534657	0.03484142	0.31549761	0.20789160	-0.053828314
X6	0.33222989	0.241604908	0.27855141	0.0002152264	0.17156914	1.0000000000	-0.097136587	-0.04621299	0.43303227	0.05891477	0.17026082	0.01829669	-0.026904947
X8	0.05225072	-0.063230009	0.08247570	-0.0685918173	0.07089817	-0.0971365873	1.000000000	0.24329556	-0.04441181	0.04345208	0.04180787	-0.11718198	-0.006180895
X9	0.16776851	0.079789987	0.18880052	-0.2829826752	0.27889889	-0.0462129859	0.243295558	1.00000000	0.09624288	0.18050511	0.85192261	0.82984527	-0.062216714
X11	0.19442279	0.201948705	0.15142542	0.0295089053	0.10534657	0.4330322675	-0.044411815	0.09624288	1.00000000	0.61984558	0.20976399	0.15142640	-0.074903689
X12	0.04786459	0.159042299	0.02569721	0.0722451474	0.03484142	0.0589147673	0.043452076	0.18050511	0.61984558	1.00000000	0.17899079	0.16282460	-0.063762636
X13	0.24768684	0.144089547	0.23369209	-0.2783001009	0.31549761	0.1702608196	0.041807866	0.85192261	0.20976399	0.17899079	1.00000000	0.83233707	-0.065861241
X14	0.15167273	0.098597154	0.14483314	-0.2222166064	0.20789160	0.0182966937	-0.117181984	0.82984527	0.15142640	0.16282460	0.83233707	1.00000000	-0.092519540
X15	0.03704938	0.006238501	-0.04845774	0.0799388429	-0.05382831	-0.0269049474	-0.006180895	-0.06221671	-0.07490369	-0.06376264	-0.06586124	-0.09251954	1.000000000

Comment:

From the scatter plot matrix, we can see that most of the graphs look random with no patterns, except some of them have linear trends. From the pairwise correlation matrix, we also see that most of the correlation value are very tiny, except some of them have around 0.85 which is near 1. Comment from both graphs, we see that despite couple pairwise quantitative variables are linearly related, most of the pairwise quantitative variables are not linearly related or not even related.

R Code:

```
# read in data
diabetesoriginal=read.table("~/Desktop/Fall Quarter 2017/STA 108/project2/diabetes.txt", header=T)
dim(diabetesoriginal)
head(diabetesoriginal)

# arrange the data set a little bit, assign variables, value the factors
diabetes=cbind(diabetesoriginal$glyhb, diabetesoriginal[,-5])
head(diabetes)

# assign variables
#   Y=diabetes$glyhb,  X1=diabetes$chol,   X2=diabetes$stab.glu, X3=diabetes$hdl,     X4=diabetes$ratio,  X5=diabetes$location
#  X6=diabetes$age,    X7=diabetes$gender, X8=diabetes$height,   X9=diabetes$weight, X10=diabetes$frame, X11=diabetes$bp.1s,
# X12=diabetes$bp.1d, X13=diabetes$waist, X14=diabetes$hip,      X15=diabetes$time.ppn

length(colnames(diabetes))
colnames(diabetes)=c("Y","X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11","X12","X13","X14","X15")
head(diabetes)
```

```
# 1
# str(filename): to see which are quantitative variables and qualitative variables
str(diabetes)

# group, view, and take out the factors, then view the data without factors
factor=c(6,8,11)                # to group: 6,8,11 are column number
head(diabetes[,factor])           # to view the factors
head(diabetes[,-factor])          # to view the data without factors

# draw histogram
#   Y=diabetes$glyhb,  X1=diabetes$chol,   X2=diabetes$stab.glu, X3=diabetes$hdl,     X4=diabetes$ratio,  X5=diabetes$location
#  X6=diabetes$age,    X7=diabetes$gender, X8=diabetes$height,   X9=diabetes$weight, X10=diabetes$frame, X11=diabetes$bp.1s,
# X12=diabetes$bp.1d, X13=diabetes$waist, X14=diabetes$hip,      X15=diabetes$time.ppn
hist(diabetes$X1, main="Histogram of diabetes$chol", breaks=40)
hist(diabetes$X2, main="Histogram of diabetes$stab.glu", breaks=40)
hist(diabetes$X3, main="Histogram of diabetes$hdl", breaks=40)
hist(diabetes$X4, main="Histogram of diabetes$ratio", breaks=40)
hist(diabetes$X6, main="Histogram of diabetes$age", breaks=40)
hist(diabetes$X8, main="Histogram of diabetes$height", breaks=40)
hist(diabetes$X9, main="Histogram of diabetes$weight", breaks=40)
hist(diabetes$X11, main="Histogram of diabetes$bp.1s", breaks=40)
hist(diabetes$X12, main="Histogram of diabetes$bp.1d", breaks=40)
hist(diabetes$X13, main="Histogram of diabetes$waist", breaks=40)
hist(diabetes$X14, main="Histogram of diabetes$hip", breaks=40)
hist(diabetes$X15, main="Histogram of diabetes$time.ppn", breaks=40)

# draw pie chart: table(variable.name)-show the weight of each frame
pie(table(diabetesoriginal$location), main="Pie Chart of diabetes$location", col=c("pink","white"))
pie(table(diabetesoriginal$gender), main="Pie Chart of diabetes$gender", col=c("yellow","white"))
pie(table(diabetesoriginal$frame), main="Pie Chart of diabetes$frame", col=c("black","white","grey"))

# draw scatterplot matrix
pairs(~X1+X2+X3+X4+X6+X8+X9+X11+X12+X13+X14+X15, data=diabetes, main="Scatterplot Matrix for Diabetes Data",
cex.main=0.8)

# obtain pairwise correlation matrix for all quantitative variables and view them
View(cor(diabetes[, -factor]))
```

R console: (contains screenshots because word format somehow messes up the columns)

```
> # read in data
> diabetesoriginal=read.table("~/Desktop/Fall Quarter 2017/STA 108/project2/diabetes.txt", header=T)
> dim(diabetesoriginal)
[1] 366  16
```

```
> head(diabetesoriginal)
  chol stab.glu hdl ratio glyhb   location age gender height weight  frame bp.1s bp.1d waist hip time.ppn
1  203       82  56   3.6  4.31 Buckingham  46 female     62    121 medium   118    59    29  38      720
2  165       97  24   6.9  4.44 Buckingham  29 female     64    218  large   112    68    46  48      360
3  228       92  37   6.2  4.64 Buckingham  58 female     61    256  large   190    92    49  57      180
4   78       93  12   6.5  4.63 Buckingham  67   male     67    119  large   110    50    33  38      480
5  249       90  28   8.9  7.72 Buckingham  64   male     68    183 medium   138    80    44  41      300
6  248       94  69   3.6  4.81 Buckingham  34   male     71    190  large   132    86    36  42      195
```

```
> # arrange the data set a little bit, assign variables, value the factors
> diabetes=cbind(diabetesoriginal$glyhb, diabetesoriginal[,-5])
```

```
> head(diabetes)
  diabetesoriginal$glyhb chol stab.glu hdl ratio   location age gender height weight  frame bp.1s bp.1d waist hip time.ppn
1                   4.31  203       82  56   3.6 Buckingham  46 female     62    121 medium   118    59    29  38      720
2                   4.44  165       97  24   6.9 Buckingham  29 female     64    218  large   112    68    46  48      360
3                   4.64  228       92  37   6.2 Buckingham  58 female     61    256  large   190    92    49  57      180
4                   4.63   78       93  12   6.5 Buckingham  67   male     67    119  large   110    50    33  38      480
5                   7.72  249       90  28   8.9 Buckingham  64   male     68    183 medium   138    80    44  41      300
6                   4.81  248       94  69   3.6 Buckingham  34   male     71    190  large   132    86    36  42      195
```

```
> length(colnames(diabetes))
[1] 16
> colnames(diabetes)=c("Y","X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11","X12","X13","X14","X15")
```

```
> head(diabetes)
     Y  X1 X2 X3  X4         X5 X6     X7 X8  X9    X10 X11 X12 X13 X14 X15
1 4.31 203 82 56 3.6 Buckingham 46 female 62 121 medium 118  59  29  38 720
2 4.44 165 97 24 6.9 Buckingham 29 female 64 218  large 112  68  46  48 360
3 4.64 228 92 37 6.2 Buckingham 58 female 61 256  large 190  92  49  57 180
4 4.63  78 93 12 6.5 Buckingham 67   male 67 119  large 110  50  33  38 480
5 7.72 249 90 28 8.9 Buckingham 64   male 68 183 medium 138  80  44  41 300
6 4.81 248 94 69 3.6 Buckingham 34   male 71 190  large 132  86  36  42 195
```

```
> # 1
```

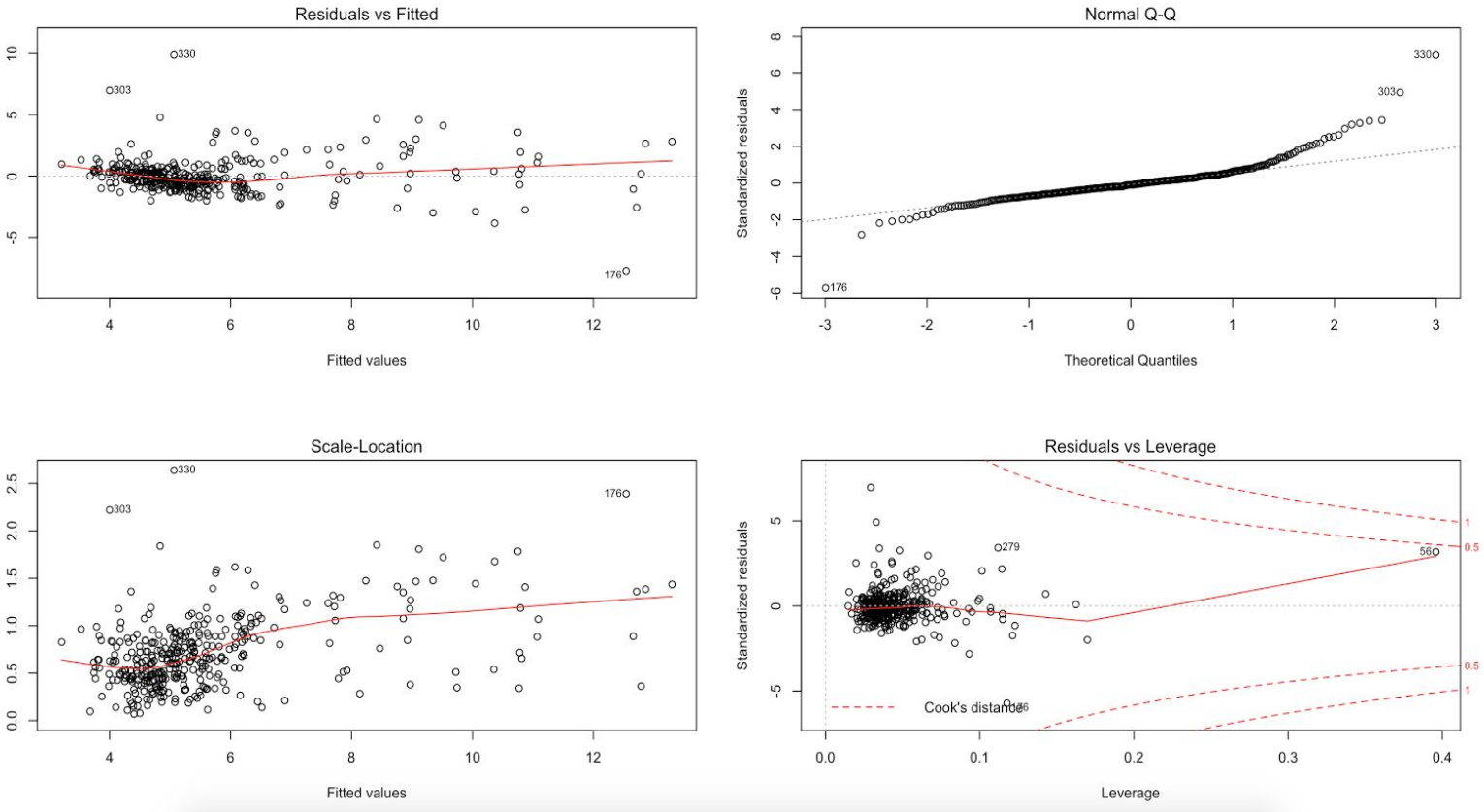
```
> # str(filename): to see which are quantitative variables and qualitative variables
> str(diabetes)
'data.frame':   366 obs. of  16 variables:
 $ Y  : num  4.31 4.44 4.64 4.63 7.72 ...
 $ X1 : int  203 165 228 78 249 248 195 177 263 242 ...
 $ X2 : int  82 97 92 93 90 94 92 87 89 82 ...
 $ X3 : int  56 24 37 12 28 69 41 49 40 54 ...
 $ X4 : num  3.6 6.9 6.2 6.5 8.9 ...
 $ X5 : Factor w/ 2 levels "Buckingham","Louisa": 1 1 1 1 1 1 1 1 1 2 ...
 $ X6 : int  46 29 58 67 64 34 30 45 55 60 ...
 $ X7 : Factor w/ 2 levels "female","male": 1 1 1 2 2 2 2 2 1 1 ...
 $ X8 : int  62 64 61 67 68 71 69 69 63 65 ...
 $ X9 : int  121 218 256 119 183 190 191 166 202 156 ...
 $ X10: Factor w/ 3 levels "large","medium",..: 2 1 1 1 2 1 2 1 3 2 ...
 $ X11: int  118 112 190 110 138 132 161 160 108 130 ...
 $ X12: int  59 68 92 50 80 86 112 80 72 90 ...
 $ X13: int  29 46 49 33 44 36 46 34 45 39 ...
 $ X14: int  38 48 57 38 41 42 49 40 50 45 ...
 $ X15: int  720 360 180 480 300 195 720 300 240 300 ...
```

```
> # group, view, and take out the factors, then view the data without factors
> factor=c(6,8,11)              # to group: 6,8,11 are column number
 > head(diabetes[,factor])            # to view the factors
          X5     X7    X10
 1 Buckingham female medium
 2 Buckingham female  large
 3 Buckingham female  large
 4 Buckingham   male  large
 5 Buckingham   male medium
 6 Buckingham   male  large
 > head(diabetes[,-factor])           # to view the data without factors
      Y  X1 X2 X3  X4 X6 X8  X9 X11 X12 X13 X14 X15
 1 4.31 203 82 56 3.6 46 62 121 118  59  29  38 720
 2 4.44 165 97 24 6.9 29 64 218 112  68  46  48 360
 3 4.64 228 92 37 6.2 58 61 256 190  92  49  57 180
 4 4.63  78 93 12 6.5 67 67 119 110  50  33  38 480
 5 7.72 249 90 28 8.9 64 68 183 138  80  44  41 300
 6 4.81 248 94 69 3.6 34 71 190 132  86  36  42 195
> # draw histogram
> #   Y=diabetes$glyhb,  X1=diabetes$chol,   X2=diabetes$stab.glu, X3=diabetes$hdl,     X4=diabetes$ratio,  X5=diabetes$location
> #  X6=diabetes$age,    X7=diabetes$gender, X8=diabetes$height,   X9=diabetes$weight, X10=diabetes$frame,
X11=diabetes$bp.1s,
> # X12=diabetes$bp.1d, X13=diabetes$waist, X14=diabetes$hip,      X15=diabetes$time.ppn
> hist(diabetes$X1, main="Histogram of diabetes$chol", breaks=40)
> hist(diabetes$X2, main="Histogram of diabetes$stab.glu", breaks=40)
> hist(diabetes$X3, main="Histogram of diabetes$hdl", breaks=40)
> hist(diabetes$X4, main="Histogram of diabetes$ratio", breaks=40)
> hist(diabetes$X6, main="Histogram of diabetes$age", breaks=40)
> hist(diabetes$X8, main="Histogram of diabetes$height", breaks=40)
> hist(diabetes$X9, main="Histogram of diabetes$weight", breaks=40)
> hist(diabetes$X11, main="Histogram of diabetes$bp.1s", breaks=40)
> hist(diabetes$X12, main="Histogram of diabetes$bp.1d", breaks=40)
> hist(diabetes$X13, main="Histogram of diabetes$waist", breaks=40)
> hist(diabetes$X14, main="Histogram of diabetes$hip", breaks=40)
> hist(diabetes$X15, main="Histogram of diabetes$time.ppn", breaks=40)
> # draw pie chart: table(variable.name)-show the weight of each frame
> pie(table(diabetesoriginal$location), main="Pie Chart of diabetes$location", col=c("pink","white"))
> pie(table(diabetesoriginal$gender), main="Pie Chart of diabetes$gender", col=c("yellow","white"))
> pie(table(diabetesoriginal$frame), main="Pie Chart of diabetes$frame", col=c("black","white","grey"))
> # draw scatterplot matrix
> pairs(~X1+X2+X3+X4+X6+X8+X9+X11+X12+X13+X14+X15, data=diabetes, main="Scatterplot Matrix for Diabetes Data",
cex.main=0.8)
> # obtain pairwise correlation matrix for all quantitative variables and view them
> View(cor(diabetes[, -factor]))
```

#2

Regress glybh on all predictor variables (Model 1).
Draw the diagnostic plots of the model and comment.

Diagnostic plots:



Comment:

From the Residuals vs Fitted plot, data points distribute not evenly but approximately along the straight red line, most of the data points concentrate on the left side of the line. The **linearity assumption** of model1 doesn't hold. In the Scale-Location plot, data points concentrate on the left side of the red curve line, which means the **equal variance** assumption also doesn't hold. In the Normal Q-Q plot, the data points fall approximately on the straight dash line, except both tails are a little bit heavier. In the Residuals vs Leverage plot, we can see data points fall inside the area under the Cook's distance lines, meaning that there is no influential outliers in this model. The **normality** assumption seems holds but with concern on the heavier tails. **Based on these observations, we may want to transform the data.**

R code:

```
# 2
model1=lm(Y~., data=diabetes)
par(mfrow = c(2,2))
plot(model1)
# when encountering error "figure margins too large": type
graphics.off() in console, and try again
```

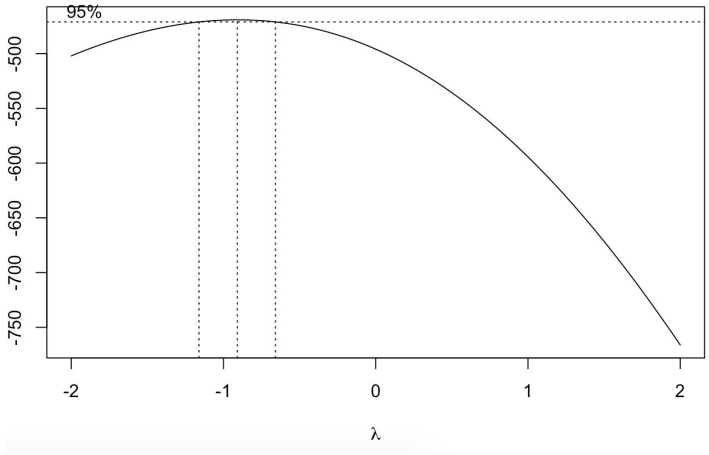
R console:

```
> # 2
> model1=lm(Y~., data=diabetes)
> par(mfrow = c(2,2))
> plot(model1)
```

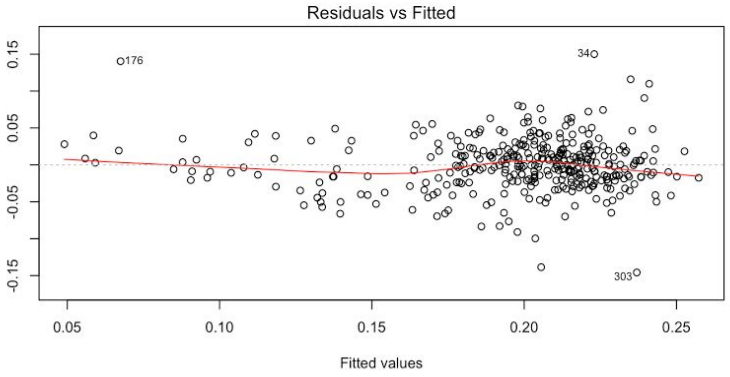
#3

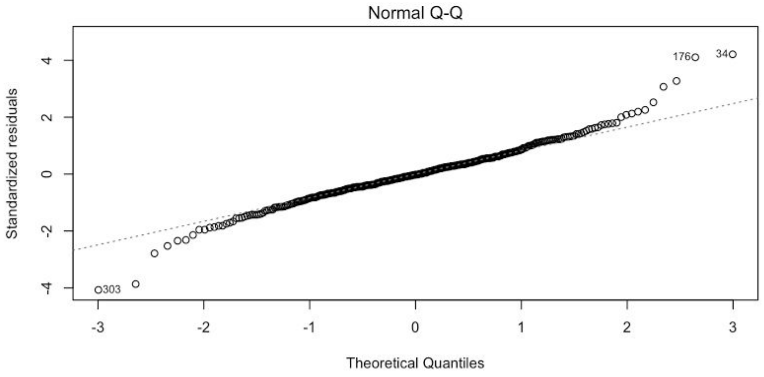
You want to check whether any transformation on the response variable is needed.
You use the function 'boxcox' to help you make the decision.
State the transformation you decide to use. In the following, we denote the transformed response variable to be glyhb∗.
Regress glyhb∗ on all predictor variables (Model 2).
Draw the diagnostic plots of this model and comment.
Apply boxcox again on Model 2; what do you find?

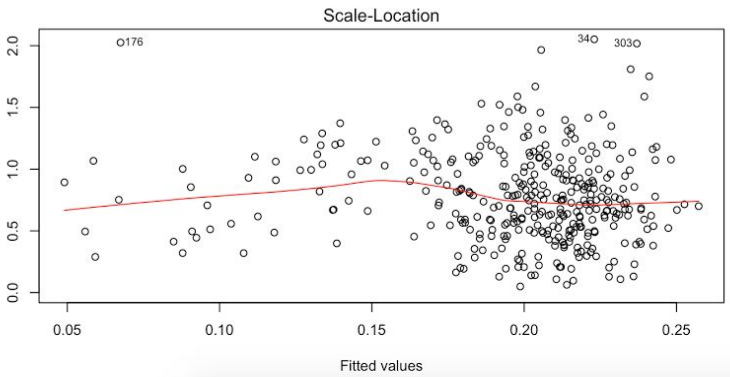
boxcox() result:
Since the λ is approximately equal to -1, I would transform y to 1/y. Therefore, glyhb*=1/glyhb.

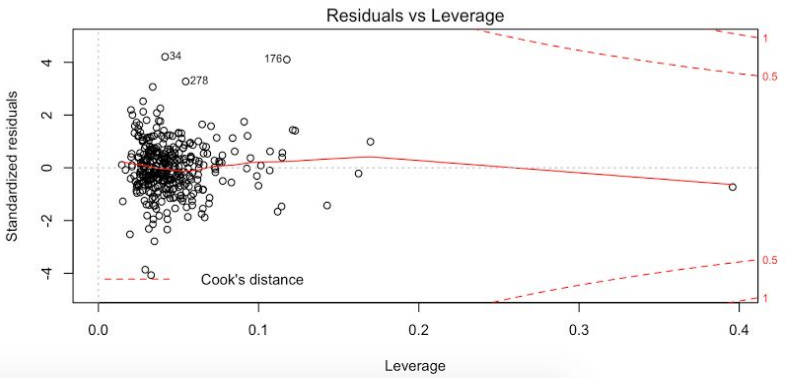


Diagnostic plot:







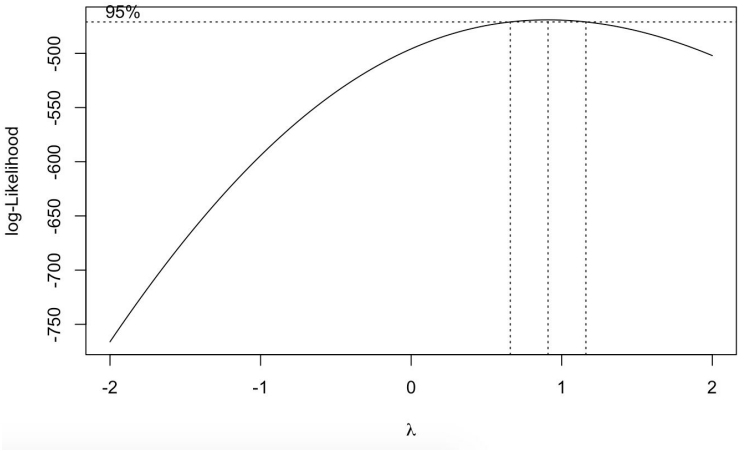


Comment:

After taking the y transformation, from the Residuals vs Fitted plot, data points distribute more evenly along the straight red line, although now most of the data points concentrate on the right side of the line. The **linearity assumption** of model1 seems holding. In the Scale-Location plot, data points spread out more on the left side of the red line, and most of the data spread evenly along the read line, which means the **equal variance** assumption also seems holding. In the Normal Q-Q plot, the data points fall approximately on the straight dash line, except both tails are slightly heavier. In the Residuals vs Leverage plot, we can see data points fall inside the area under the Cook's distance lines, meaning that there is no influential outliers in this model. The **normality** assumption seems holding but the concern on the heavier tails also still maintains**.**

Apply boxcox() again. Its result:

The λ is approximately equal to 1 now, which approximately means the transformation on y above is good enough.



R code:

```
# 3
library(MASS)                    # for boxcox()
boxcox(model1)
diabetes$Ytrans=1/diabetes$Y       # transform Y to 1/Y and attach it to the data
diabetes=diabetes[, names(diabetes)!="Y"]    # extract all variables not named Y
View(diabetes)
head(diabetes)

model2=lm(Ytrans~., data=diabetes)
par(mfrow = c(2,2))
plot(model2)
boxcox(model2)
```

R console: (contains screenshots because word format somehow messes up the columns)

```
> # 3
> library(MASS)                    # for boxcox()
> boxcox(model1)
> diabetes$Ytrans=1/diabetes$Y       # transform Y to 1/Y and attach it to the data
> diabetes=diabetes[, names(diabetes)!="Y"]    # extract all variables not named Y
```

```
> View(diabetes)
> head(diabetes)
   X1 X2 X3  X4         X5 X6     X7 X8  X9    X10 X11 X12 X13 X14 X15    Ytrans
1 203 82 56 3.6 Buckingham 46 female 62 121 medium 118  59  29  38 720 0.2320186
2 165 97 24 6.9 Buckingham 29 female 64 218  large 112  68  46  48 360 0.2252252
3 228 92 37 6.2 Buckingham 58 female 61 256  large 190  92  49  57 180 0.2155172
4  78 93 12 6.5 Buckingham 67   male 67 119  large 110  50  33  38 480 0.2159827
5 249 90 28 8.9 Buckingham 64   male 68 183 medium 138  80  44  41 300 0.1295337
6 248 94 69 3.6 Buckingham 34   male 71 190  large 132  86  36  42 195 0.2079002
> model2=lm(Ytrans~., data=diabetes)
> par(mfrow = c(2,2))
> plot(model2)
> boxcox(model2)
```

#4

Randomly split data into two equal halves: a training data set and a validation data set.

R code:

```
# 4
set.seed(10)                  # set seed for random variable generator so everyone gets the same split of the data
N=nrow(diabetes)              # number of cases in diabetes
N
# randomly sample N/2 cases to form the training data
index=sample(1:N, size=N/2, replace=FALSE)
data.t=diabetes[index,]       # get the training data set
data.v=diabetes[-index,]      # use the remaining cases to form the validation set
```

R console:

```
> # 4
> set.seed(10)                # set seed for random variable generator so everyone gets the same split of the data
> N=nrow(diabetes)            # number of cases in diabetes
> N
[1] 366
> # randomly sample N/2 cases to form the training data
> index=sample(1:N, size=N/2, replace=FALSE)
> data.t=diabetes[index,]       # get the training data set
> data.v=diabetes[-index,]      # use the remaining cases to form the validation set
```

Selection of first-order effects.

We now consider subsets selection from the pool of all first-order effects of the 15 predictors. glyhb* is used as the response variable for the following problems.

#5

Fit a model with all first-order effects (Model 3). How many regression coefficients are there in this model? What is its MSE?

Ans: there are 17 regression coefficients in Model3. The MSE from this model is: 0.001383855.

R code:

```
# 5
model3=lm(Ytrans~., data=data.t)
summary(model3)
countbeta=length(model3$coefficients)
countbeta
MSEmodel3=summary(model3)$sigma^2
MSEmodel3
```

R console: (contains screenshots because word format somehow messes up the columns)

```
> # 5
> model3=lm(Ytrans~., data=data.t)
> summary(model3)

Call:
lm(formula = Ytrans ~ ., data = data.t)

Residuals:
      Min        1Q    Median        3Q       Max
-0.097813 -0.022472 -0.002034  0.021097  0.134611

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.819e-01  8.499e-02   5.670 6.19e-08 ***
X1          -6.857e-05  1.695e-04  -0.405   0.6863
X2          -5.314e-04  5.418e-05  -9.807  < 2e-16 ***
X3           1.211e-04  5.492e-04   0.220   0.8258
X4          -2.414e-03  6.588e-03  -0.366   0.7145
X5Louisa    -1.808e-03  5.969e-03  -0.303   0.7623
X6          -5.487e-04  2.199e-04  -2.495   0.0136 *
X7male      -7.422e-04  1.018e-02  -0.073   0.9420
X8          -1.212e-03  1.123e-03  -1.079   0.2820
X9           2.210e-04  2.034e-04   1.087   0.2788
X10medium    1.417e-03  7.861e-03   0.180   0.8572
X10small    -1.062e-02  9.596e-03  -1.107   0.2699
X11         -1.214e-04  1.708e-04  -0.711   0.4782
X12          3.198e-05  2.505e-04   0.128   0.8986
X13         -1.893e-03  1.148e-03  -1.649   0.1010
X14         -1.177e-03  1.352e-03  -0.870   0.3854
X15         -1.444e-05  9.881e-06  -1.461   0.1459
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0372 on 166 degrees of freedom
Multiple R-squared:  0.5547,    Adjusted R-squared:  0.5118
F-statistic: 12.92 on 16 and 166 DF,  p-value: < 2.2e-16

> countbeta=length(model3$coefficients)
> countbeta
[1] 17
> MSEmodel3=summary(model3)$sigma^2
> MSEmodel3
[1] 0.001383855
```

#6

Consider best subsets selection using the R function regsubsets() from the leaps library with Model 3 as the full model.
Return the top 1 best subset of all subset sizes (i.e., number of X variables) up to 16 (because frame has 3 levels).
Get SSE_p,R_{p2}, $R_{a2,p}$, C_p, AIC_p, BIC_p for each of these models, as well as the none-model (the model with only an intercept).
Identify the best model according to each criterion.
For the best model according to C_p criterion, what do you observe about its C_p value?
Do you have a possible explanation for it?
Denote the best models according to AIC, BIC, and adjusted R_2 be Model 3.1, Model 3.2, Model 3.3, respectively. (It is possible that some of the three models are the same.)

Ans:

- Shown in R console: best subsets, criteria values of each of them, chosen best models, denote model 3.1, 3.2 and 3.3.
- The best model according to Cp criterion has Cp=0.05337754 and p=5. Cp is much smaller than p.
- Since we don't have a good estimator of sigma square in here for the target model, we use the MSE of the full model to estimate sigma hat. Cp=SSEp/(sigmahat^2)-(n-p) will be equal to p if we have a good estimator of sigma square. But the smallest Cp here is way too small than p, which means that the sigmahat of the full model is not a good estimator. We need to look for another better estimator.

R code:

```
# 6
install.packages("leaps")
library(leaps)
# nbest-number of best model to be selected, nvmax-number of maximum models in pool
best=regsubsets(Ytrans~., data=data.t, nbest=1, nvmax=16)
summary.subset=summary(best)
summary.subset                  # return the top models in their subset
summary.subset$which            # clearly see which X to be included in top models

# SSEp, R2p, Ra2p, Cp, AICp, BICp for top models and none model
n=nrow(data.t)                  # number of cases in data.t
n
p.m=as.integer(as.numeric(rownames(summary.subset$which))+1)
p.m                             # coding "p.m=2:17" to get p.m is fine too
sse=summary.subset$rss
sse
r2=summary.subset$rsq
r2
ra2=summary.subset$adjr2
ra2
c=summary.subset$cp
c
aic=n*log(sse)+2*p.m-n*log(n)
aic
bic=n*log(sse)+log(n)*p.m-n*log(n)
bic
```

```
list.result=cbind(summary.subset$which,sse,r2,ra2,c,aic,bic)
list.result

modelnone=lm(Ytrans~1,data=data.t) # fit the none model
ssenone=sum(modelnone$residuals^2)
ssenone
p=1
r2none=0
ra2none=0
cnone=ssenone/MSEmodel3-(n-2*p)
cnone
aicnone=n*log(ssenone)+2*p-n*log(n)
aicnone
bicnone=n*log(ssenone)+log(n)*p-n*log(n)
bicnone

list.resultnone=c(1,rep(0,16),ssenone,r2none, ra2none,cnone,aicnone,bicnone)
list.resultnone

# combine list.resultnone to list.result, set columm names
list.result=rbind(list.resultnone,list.result)
colnames(list.result)=c(colnames(summary.subset$which), "SSE", "R2", "Ra2", "C","AIC","BIC")
colnames(list.result)
list.result
Frametype=model.matrix(~data.t$X10-1)

# base on the result, best models:
model3.1=lm(Ytrans~X2+X4+X6+Frametype[,3]+X13, data=data.t)          # in terms of AIC
model3.2=lm(Ytrans~X2+X6+X13, data=data.t)                          # in terms of BIC
model3.3=lm(Ytrans~X2+X4+X6+Frametype[,3]+X13+X15, data=data.t)       # in terms of Ra2
```

R console: (contains screenshots because word format somehow messes up the columns)

```
> # 6
> install.packages("leaps")
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.4/leaps_3.0.tgz'
Content type 'application/x-gzip' length 69196 bytes (67 KB)
==================================================
downloaded 67 KB
The downloaded binary packages are in
        /var/folders/9n/7nbzmhbd1nv_6flztdt6yylh0000gn/T//Rtmp6lTNWy/downloaded_packages
> library(leaps)
> # nbest-number of best model to be selected, nvmax-number of maximum models in pool
> best=regsubsets(Ytrans~., data=data.t, nbest=1, nvmax=16)
> summary.subset=summary(best)
```

```
> summary.subset                    # return the top models in their subset
Subset selection object
Call: regsubsets.formula(Ytrans ~ ., data = data.t, nbest = 1, nvmax = 16)
16 Variables  (and intercept)
          Forced in Forced out
X1            FALSE      FALSE
X2            FALSE      FALSE
X3            FALSE      FALSE
X4            FALSE      FALSE
X5Louisa      FALSE      FALSE
X6            FALSE      FALSE
X7male        FALSE      FALSE
X8            FALSE      FALSE
X9            FALSE      FALSE
X10medium     FALSE      FALSE
X10small      FALSE      FALSE
X11           FALSE      FALSE
X12           FALSE      FALSE
X13           FALSE      FALSE
X14           FALSE      FALSE
X15           FALSE      FALSE
1 subsets of each size up to 16
Selection Algorithm: exhaustive
          X1  X2  X3  X4  X5Louisa X6  X7male X8  X9  X10medium X10small X11 X12 X13 X14 X15
1  ( 1 )  " " "*" " " " " " " " "       " " " " " " " "        " "       " " " " " " " " " "
2  ( 1 )  " " "*" " " " " " " " "       "*" " "     " " " " " " " "        " "       " " " " " " " " " "
3  ( 1 )  " " "*" " " " " " " " "       "*" " "     " " " " " " " "        " "       " " " " "*" " " " "
4  ( 1 )  " " "*" " " "*" " " "       "*" " "     " " " " " " " "        " "       " " " " "*" " " " "
5  ( 1 )  " " "*" " " "*" " " "       "*" " "     " " " " " " " "        "*"       " " " " "*" " " " "
6  ( 1 )  " " "*" " " "*" " " "       "*" " "     " " " " " " " "        "*"       " " " " "*" " " "*"
7  ( 1 )  " " "*" " " "*" " " "       "*" " "     " " " " " " " "        "*"       "*" " " "*" " " "*"
8  ( 1 )  " " "*" " " "*" " " "       "*" " "     "*" " " " " " "        "*"       "*" " " "*" " " "*"
9  ( 1 )  " " "*" " " "*" " " "       "*" " "     "*" "*" " " " "        "*"       " " " " "*" "*" "*"
10  ( 1 ) " " "*" " " "*" " " "       "*" " "     "*" "*" " " " "        "*"       "*" " " "*" "*" "*"
11  ( 1 ) "*" "*" " " "*" " " "       "*" " "     "*" "*" " " " "        "*"       "*" " " "*" "*" "*"
12  ( 1 ) "*" "*" " " "*" "*" "       "*" " "     "*" "*" " " " "        "*"       "*" " " "*" "*" "*"
13  ( 1 ) "*" "*" "*" "*" "*" "       "*" " "     "*" "*" " " " "        "*"       "*" " " "*" "*" "*"
14  ( 1 ) "*" "*" "*" "*" "*" "       "*" " "     "*" "*" "*" " "        "*"       "*" " " "*" "*" "*"
15  ( 1 ) "*" "*" "*" "*" "*" "       "*" " "     "*" "*" "*" " "        "*"       "*" "*" "*" "*" "*"
16  ( 1 ) "*" "*" "*" "*" "*" "       "*" "*"     "*" "*" "*" " "        "*"       "*" "*" "*" "*" "*"
> summary.subset$which               # clearly see which X to be included in top models
   (Intercept)    X1   X2    X3    X4 X5Louisa    X6 X7male    X8    X9 X10medium X10small   X11   X12   X13   X14   X15
1         TRUE FALSE TRUE FALSE FALSE    FALSE FALSE  FALSE FALSE FALSE     FALSE    FALSE FALSE FALSE FALSE FALSE FALSE
2         TRUE FALSE TRUE FALSE FALSE    FALSE  TRUE  FALSE FALSE FALSE     FALSE    FALSE FALSE FALSE FALSE FALSE FALSE
3         TRUE FALSE TRUE FALSE FALSE    FALSE  TRUE  FALSE FALSE FALSE     FALSE    FALSE FALSE FALSE  TRUE FALSE FALSE
4         TRUE FALSE TRUE FALSE  TRUE    FALSE  TRUE  FALSE FALSE FALSE     FALSE    FALSE FALSE FALSE  TRUE FALSE FALSE
5         TRUE FALSE TRUE FALSE  TRUE    FALSE  TRUE  FALSE FALSE FALSE     FALSE     TRUE FALSE FALSE  TRUE FALSE FALSE
6         TRUE FALSE TRUE FALSE  TRUE    FALSE  TRUE  FALSE FALSE FALSE     FALSE     TRUE FALSE FALSE  TRUE FALSE  TRUE
7         TRUE FALSE TRUE FALSE  TRUE    FALSE  TRUE  FALSE FALSE FALSE     FALSE     TRUE  TRUE FALSE  TRUE FALSE  TRUE
8         TRUE FALSE TRUE FALSE  TRUE    FALSE  TRUE  FALSE  TRUE FALSE     FALSE     TRUE  TRUE FALSE  TRUE FALSE  TRUE
9         TRUE FALSE TRUE FALSE  TRUE    FALSE  TRUE  FALSE  TRUE  TRUE     FALSE     TRUE FALSE FALSE  TRUE  TRUE  TRUE
10        TRUE FALSE TRUE FALSE  TRUE    FALSE  TRUE  FALSE  TRUE  TRUE     FALSE     TRUE  TRUE FALSE  TRUE  TRUE  TRUE
11        TRUE  TRUE TRUE FALSE  TRUE    FALSE  TRUE  FALSE  TRUE  TRUE     FALSE     TRUE  TRUE FALSE  TRUE  TRUE  TRUE
12        TRUE  TRUE TRUE FALSE  TRUE     TRUE  TRUE  FALSE  TRUE  TRUE     FALSE     TRUE  TRUE FALSE  TRUE  TRUE  TRUE
13        TRUE  TRUE TRUE  TRUE  TRUE     TRUE  TRUE  FALSE  TRUE  TRUE     FALSE     TRUE  TRUE FALSE  TRUE  TRUE  TRUE
14        TRUE  TRUE TRUE  TRUE  TRUE     TRUE  TRUE  FALSE  TRUE  TRUE      TRUE     TRUE  TRUE FALSE  TRUE  TRUE  TRUE
15        TRUE  TRUE TRUE  TRUE  TRUE     TRUE  TRUE  FALSE  TRUE  TRUE      TRUE     TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
16        TRUE  TRUE TRUE  TRUE  TRUE     TRUE  TRUE   TRUE  TRUE  TRUE      TRUE     TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
> # SSEp, R2p, Ra2p, Cp, AICp, BICp for top models and none model
> n=nrow(data.t)                    # number of cases in data.t
> n
[1] 183
> p.m=as.integer(as.numeric(rownames(summary.subset$which))+1)
> p.m                               # coding "p.m=2:17" to get p.m is fine too
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
> sse=summary.subset$rss
> sse
 [1] 0.2864076 0.2574112 0.2428890 0.2401432 0.2367131 0.2343460 0.2331725 0.2326634 0.2314193 0.2303187 0.2300477 0.2299216 0.2298166 0.2297510
[15] 0.2297274 0.2297200
> r2=summary.subset$rsq
> r2
 [1] 0.4448009 0.5010102 0.5291612 0.5344840 0.5411332 0.5457220 0.5479966 0.5489836 0.5513952 0.5535287 0.5540541 0.5542986 0.5545020 0.5546292
[15] 0.5546751 0.5546893
> ra2=summary.subset$adjr2
> ra2
 [1] 0.4417335 0.4954659 0.5212701 0.5240230 0.5281708 0.5302352 0.5299165 0.5282473 0.5280574 0.5275711 0.5253676 0.5228374 0.5202329 0.5175150
[15] 0.5146758 0.5117678
> c=summary.subset$cp
> c
 [1] 27.96351331  9.01014928  0.51619889  0.53201659  0.05337754  0.34280455  1.49487219  3.12693590  4.22797088  5.43265348  7.23678869
[12]  9.14564365 11.06983181 13.02241521 15.00531267 17.00000000
> aic=n*log(sse)+2*p.m-n*log(n)
> aic
 [1] -1178.148 -1195.682 -1204.309 -1204.389 -1205.022 -1204.861 -1203.780 -1202.180 -1201.161 -1200.033 -1198.249 -1196.349 -1194.433 -1192.485
[15] -1190.504 -1188.510
> bic=n*log(sse)+log(n)*p.m-n*log(n)
> bic
 [1] -1171.729 -1186.053 -1191.471 -1188.342 -1185.765 -1182.395 -1178.104 -1173.294 -1169.066 -1164.729 -1159.735 -1154.626 -1149.500 -1144.343
[15] -1139.152 -1133.948
```

```
> list.result=cbind(summary.subset$which,sse,r2,ra2,c,aic,bic)
```

```
> list.result
   (Intercept) X1 X2 X3 X4 X5Louisa X6 X7male X8 X9 X10medium X10small X11 X12 X13 X14 X15       sse        r2       ra2           c       aic
1            1  0  1  0  0        0  0       0  0  0         0        0   0   0   0   0   0 0.2864076 0.4448009 0.4417335 27.96351331 -1178.148
2            1  0  1  0  0        0  1       0  0  0         0        0   0   0   0   0   0 0.2574112 0.5010102 0.4954659  9.01014928 -1195.682
3            1  0  1  0  0        0  1       0  0  0         0        0   0   0   1   0   0 0.2428890 0.5291612 0.5212701  0.51619889 -1204.309
4            1  0  1  0  1        0  1       0  0  0         0        0   0   0   1   0   0 0.2401432 0.5344840 0.5240230  0.53201659 -1204.389
5            1  0  1  0  1        0  1       0  0  0         0        1   0   0   1   0   0 0.2367131 0.5411332 0.5281708  0.05337754 -1205.022
6            1  0  1  0  1        0  1       0  0  0         0        1   0   0   1   0   1 0.2343460 0.5457220 0.5302352  0.34280455 -1204.861
7            1  0  1  0  1        0  1       0  0  0         0        1   1   0   1   0   1 0.2331725 0.5479966 0.5299165  1.49487219 -1203.780
8            1  0  1  0  1        0  1       0  1  0         0        1   1   0   1   0   1 0.2326634 0.5489836 0.5282473  3.12693590 -1202.180
9            1  0  1  0  1        0  1       0  1  1         0        1   0   0   1   1   1 0.2314193 0.5513952 0.5280574  4.22797088 -1201.161
10           1  0  1  0  1        0  1       0  1  1         0        1   1   0   1   1   1 0.2303187 0.5535287 0.5275711  5.43265348 -1200.033
11           1  1  1  0  1        0  1       0  1  1         0        1   1   0   1   1   1 0.2300477 0.5540541 0.5253676  7.23678869 -1198.249
12           1  1  1  0  1        1  1       0  1  1         0        1   1   0   1   1   1 0.2299216 0.5542986 0.5228374  9.14564365 -1196.349
13           1  1  1  1  1        1  1       0  1  1         0        1   1   0   1   1   1 0.2298166 0.5545020 0.5202329 11.06983181 -1194.433
14           1  1  1  1  1        1  1       0  1  1         1        1   1   0   1   1   1 0.2297510 0.5546292 0.5175150 13.02241521 -1192.485
15           1  1  1  1  1        1  1       0  1  1         1        1   1   1   1   1   1 0.2297274 0.5546751 0.5146758 15.00531267 -1190.504
16           1  1  1  1  1        1  1       1  1  1         1        1   1   1   1   1   1 0.2297200 0.5546893 0.5117678 17.00000000 -1188.510
         bic
1  -1171.729
2  -1186.053
3  -1191.471
4  -1188.342
5  -1185.765
6  -1182.395
7  -1178.104
8  -1173.294
9  -1169.066
10 -1164.729
11 -1159.735
12 -1154.626
13 -1149.500
14 -1144.343
15 -1139.152
16 -1133.948
```

```
> modelnone=lm(Ytrans~1,data=data.t) # fit the none model
> ssenone=sum(modelnone$residuals^2)
> ssenone
[1] 0.5158646
> p=1
> r2none=0
> ra2none=0
> cnone=ssenone/MSEmodel3-(n-2*p)
> cnone
[1] 191.7735
> aicnone=n*log(ssenone)+2*p-n*log(n)
> aicnone
[1] -1072.466
> bicnone=n*log(ssenone)+log(n)*p-n*log(n)
> bicnone
[1] -1069.256
> list.resultnone=c(1,rep(0,16),ssenone,r2none, ra2none,cnone,aicnone,bicnone)
```

```
> list.resultnone
 [1]     1.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000
[11]     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.0000000     0.5158646     0.0000000     0.0000000
[21]   191.7734708 -1072.4656729 -1069.2561867
> # combine list.resultnone to list.result, set columm names
> list.result=rbind(list.resultnone,list.result)
> colnames(list.result)=c(colnames(summary.subset$which), "SSE", "R2", "Ra2", "C","AIC","BIC")
> colnames(list.result)
 [1] "(Intercept)" "X1"          "X2"          "X3"          "X4"          "X5Louisa"    "X6"          "X7male"      "X8"          "X9"
[11] "X10medium"   "X10small"    "X11"         "X12"         "X13"         "X14"         "X15"         "SSE"         "R2"          "Ra2"
[21] "C"           "AIC"         "BIC"
> list.result
               (Intercept) X1 X2 X3 X4 X5Louisa X6 X7male X8 X9 X10medium X10small X11 X12 X13 X14 X15       SSE        R2       Ra2           C
list.resultnone           1  0  0  0  0        0  0      0  0  0         0        0   0   0   0   0   0 0.5158646 0.0000000 0.0000000 191.77347084
1                         1  0  1  0  0        0  0      0  0  0         0        0   0   0   0   0   0 0.2864076 0.4448009 0.4417335  27.96351331
2                         1  0  1  0  0        0  1      0  0  0         0        0   0   0   0   0   0 0.2574112 0.5010102 0.4954659   9.01014928
3                         1  0  1  0  0        0  1      0  0  0         0        0   0   0   1   0   0 0.2428890 0.5291612 0.5212701   0.51619889
4                         1  0  1  0  1        0  1      0  0  0         0        0   0   0   1   0   0 0.2401432 0.5344840 0.5240230   0.53201659
5                         1  0  1  0  1        0  1      0  0  0         0        1   0   0   1   0   0 0.2367131 0.5411332 0.5281708   0.05337754
6                         1  0  1  0  1        0  1      0  0  0         0        1   0   0   1   0   1 0.2343460 0.5457220 0.5302352   0.34280455
7                         1  0  1  0  1        0  1      0  0  0         0        1   1   0   1   0   1 0.2331725 0.5479966 0.5299165   1.49487219
8                         1  0  1  0  1        0  1      0  1  0         0        1   1   0   1   0   1 0.2326634 0.5489836 0.5282473   3.12693590
9                         1  0  1  0  1        0  1      0  1  1         0        1   0   0   1   1   1 0.2314193 0.5513952 0.5280574   4.22797088
10                        1  0  1  0  1        0  1      0  1  1         0        1   1   0   1   1   1 0.2303187 0.5535287 0.5275711   5.43265348
11                        1  1  1  0  1        0  1      0  1  1         0        1   1   0   1   1   1 0.2300477 0.5540541 0.5253676   7.23678869
12                        1  1  1  0  1        1  1      0  1  1         0        1   1   0   1   1   1 0.2299216 0.5542986 0.5228374   9.14564365
13                        1  1  1  1  1        1  1      0  1  1         0        1   1   0   1   1   1 0.2298166 0.5545020 0.5202329  11.06983181
14                        1  1  1  1  1        1  1      0  1  1         1        1   1   0   1   1   1 0.2297510 0.5546292 0.5175150  13.02241521
15                        1  1  1  1  1        1  1      0  1  1         1        1   1   1   1   1   1 0.2297274 0.5546751 0.5146758  15.00531267
16                        1  1  1  1  1        1  1      1  1  1         1        1   1   1   1   1   1 0.2297200 0.5546893 0.5117678  17.00000000
                     AIC       BIC
list.resultnone -1072.466 -1069.256
1               -1178.148 -1171.729
2               -1195.682 -1186.053
3               -1204.309 -1191.471
4               -1204.389 -1188.342
5               -1205.022 -1185.765
6               -1204.861 -1182.395
7               -1203.780 -1178.104
8               -1202.180 -1173.294
9               -1201.161 -1169.066
10              -1200.033 -1164.729
11              -1198.249 -1159.735
12              -1196.349 -1154.626
13              -1194.433 -1149.500
14              -1192.485 -1144.343
15              -1190.504 -1139.152
16              -1188.510 -1133.948
```

```
> Frametype=model.matrix(~data.t$X10-1)
> # base on the result, best models:
> model3.1=lm(Ytrans~X2+X4+X6+Frametype[,3]+X13, data=data.t)          # in terms of AIC
> model3.2=lm(Ytrans~X2+X6+X13, data=data.t)                          # in terms of BIC
> model3.3=lm(Ytrans~X2+X4+X6+Frametype[,3]+X13+X15, data=data.t)     # in terms of Ra2
```

Selection of first- and second- order effects.

We now consider subsets selection from the pool of first-order effects as well as 2-way interaction effects of the 15 predictors.

#7

Fit a model with all first-order and 2-way interaction effects (Model 4).
How many regression coefficients are there in this model?
What is the MSE from this model?
Do you have any concern about the fitting of this model and why?

Ans:

- There are 136 regression coefficients in this model.
- The MSE from this model is: 0.001036088.
- My concern on the fitting of this model is that it may contain too many predicted variable terms because adding more and more predicted variables will increase the r square of this model. We don't want to include so many predicted variables, especially where among them there might be some not so useful terms. We don't want to add these not so useful coefficients to the model.

R code:

```
# 7
model4=lm(Ytrans~.^2, data=data.t)
countcof=length(model4$coefficients)
countcof
MSEmodel4=summary(model4)$sigma^2
MSEmodel4
```

R console:

```
> # 7
> model4=lm(Ytrans~.^2, data=data.t)
> countcof=length(model4$coefficients)
> countcof
[1] 136
> MSEmodel4=summary(model4)$sigma^2
> MSEmodel4
[1] 0.001036088
```

#8

Apply the forward stepwise procedure using R function step() (or stepAIC()), starting from the none-model and using the AIC_p criterion.

What is the model being selected? Denote this model by Model fs1.

Compare its AIC value with that of Model3.1. What do you find?

Ans:

- Shown in R console: fsp1 model summary
- fsp1 AIC= -1205.14
- model3.1 AIC= -1205.022
- Compare these two AICs, I found that fsp1 has a smaller AIC value.

R code:

```
fsp1=stepAIC(modelnone, scope=list(lower=modelnone,upper=model4), direction="both", k=2, data=data.t)
summary(fsp1)   # In the results, "X2:X4" means X2*X4.
sse.fsp1=sum(fsp1$residuals^2)
sse.fsp1
p.fsp1=length(fsp1$coefficients)
p.fsp1
```

R console: (contains screenshots because word format somehow messes up the columns; include only the last part, where there shows the model and the final AIC, of the result on the stepAIC function)

……

```
Step:  AIC=-1205.14
Ytrans ~ X2 + X6 + X13 + X4 + X2:X4 + X6:X4

          Df Sum of Sq     RSS     AIC
<none>                 0.23398 -1205.1
- X6:X4    1 0.0026083 0.23659 -1205.1
+ X15      1 0.0019693 0.23201 -1204.7
+ X2:X6    1 0.0011229 0.23286 -1204.0
+ X8       1 0.0010043 0.23298 -1203.9
+ X11      1 0.0005834 0.23340 -1203.6
+ X3       1 0.0004351 0.23355 -1203.5
+ X2:X13   1 0.0004169 0.23357 -1203.5
+ X1       1 0.0002772 0.23371 -1203.4
+ X9       1 0.0002713 0.23371 -1203.4
+ X10      2 0.0027231 0.23126 -1203.3
+ X7       1 0.0001272 0.23386 -1203.2
+ X12      1 0.0000804 0.23390 -1203.2
+ X6:X13   1 0.0000781 0.23391 -1203.2
+ X5       1 0.0000033 0.23398 -1203.2
+ X14      1 0.0000016 0.23398 -1203.1
+ X4:X13   1 0.0000012 0.23398 -1203.1
- X2:X4    1 0.0052565 0.23924 -1203.1
- X13      1 0.0087815 0.24277 -1200.4
```

……

```
> summary(fsp1)   # In the results, "X2:X4" means X2*X4.

Call:
lm(formula = Ytrans ~ X2 + X6 + X13 + X4 + X2:X4 + X6:X4, data = data.t)

Residuals:
      Min        1Q    Median        3Q       Max
-0.089202 -0.022258 -0.003599  0.021182  0.145324

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.527e-01  3.162e-02  11.152  < 2e-16 ***
X2          -9.522e-04  2.186e-04  -4.355 2.25e-05 ***
X6           7.247e-05  5.277e-04   0.137   0.8909
X13         -1.305e-03  5.079e-04  -2.570   0.0110 *
X4          -2.158e-03  6.565e-03  -0.329   0.7427
X2:X4        7.507e-05  3.775e-05   1.988   0.0483 *
X6:X4       -1.724e-04  1.231e-04  -1.401   0.1631
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03646 on 176 degrees of freedom
Multiple R-squared:  0.5464,    Adjusted R-squared:  0.531
F-statistic: 35.34 on 6 and 176 DF,  p-value: < 2.2e-16
```

```
> sse.fsp1=sum(fsp1$residuals^2)
> sse.fsp1
[1] 0.2339843
> p.fsp1=length(fsp1$coefficients)
> p.fsp1
[1] 7
```

#9

Apply the forward stepwise procedure using R function step() (or stepAIC()), starting from the full model (Model 3) and using the AIC_p criterion.
What is the model being selected? Denote this model by Model fs2.
Compare its AIC value with that of Model fs1. What do you find?

Ans:

- Shown in R console: fsp2 model summary.
- fsp2 AIC= -1230.61
- fsp1 AIC= -1205.14
- Compare these two AICs, I found that fsp2 has the smaller AIC.
- Shown in R code: model denoting.

R code:

```
# 9
fsp2=stepAIC(model3, scope=list(lower=modelnone,upper=model4), direction="both", k=2, data=data.t)
summary(fsp2)
sse.fsp2=sum(fsp2$residuals^2)
sse.fsp2
p.fsp2=length(fsp2$coefficients)
p.fsp2
```

R console: (contains screenshots because word format somehow messes up the columns; include only the last part, where there shows the model and the final AIC, of the result on the stepAIC function)

……

```
 - X7:X8    1 0.0072488 0.16564 -1224.4
 - X2:X11   1 0.0102983 0.16869 -1221.0
 - X6:X14   1 0.0122841 0.17068 -1218.9
 - X2:X7    1 0.0129067 0.17130 -1218.2
 - X3:X4    1 0.0160045 0.17440 -1214.9
 - X6:X12   1 0.0160819 0.17448 -1214.8

 Step:  AIC=-1230.61
 Ytrans ~ X1 + X2 + X3 + X4 + X6 + X7 + X8 + X9 + X11 + X12 +
     X13 + X14 + X15 + X2:X7 + X3:X4 + X6:X12 + X9:X11 + X6:X14 +
     X14:X15 + X7:X8 + X2:X11 + X2:X15 + X2:X13 + X6:X13 + X1:X15 +
     X3:X9 + X12:X13 + X9:X14

            Df Sum of Sq     RSS     AIC
 <none>                   0.16008 -1230.6
 + X13:X15  1 0.0016817 0.15839 -1230.5
 + X4:X13   1 0.0015327 0.15854 -1230.4
 + X11:X14  1 0.0015135 0.15856 -1230.3
 + X8:X15   1 0.0014705 0.15860 -1230.3
 + X6:X7    1 0.0013284 0.15875 -1230.1
 + X4:X9    1 0.0013265 0.15875 -1230.1
```

……

```
> summary(fsp2)

Call:
lm(formula = Ytrans ~ X1 + X2 + X3 + X4 + X6 + X7 + X8 + X9 +
    X11 + X12 + X13 + X14 + X15 + X2:X7 + X3:X4 + X6:X12 + X9:X11 +
    X6:X14 + X14:X15 + X7:X8 + X2:X11 + X2:X15 + X2:X13 + X6:X13 +
    X1:X15 + X3:X9 + X12:X13 + X9:X14, data = data.t)

Residuals:
     Min       1Q   Median       3Q      Max
-0.07810 -0.01738 -0.00148  0.01728  0.10831

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.454e-01  1.585e-01   1.548 0.123555
X1          -2.644e-03  7.046e-04  -3.753 0.000247 ***
X2          -1.074e-03  4.662e-04  -2.304 0.022538 *
X3           8.082e-04  7.721e-04   1.047 0.296834
X4          -1.765e-02  6.686e-03  -2.640 0.009146 **
X6          -1.019e-03  1.432e-03  -0.712 0.477529
X7male       3.321e-01  1.305e-01   2.545 0.011918 *
X8           6.412e-04  1.291e-03   0.497 0.620248
X9           3.080e-03  7.854e-04   3.921 0.000132 ***
X11          7.464e-04  7.323e-04   1.019 0.309683
X12         -8.385e-04  1.584e-03  -0.529 0.597288
X13         -3.229e-03  4.674e-03  -0.691 0.490772
X14         -4.314e-03  3.540e-03  -1.219 0.224789
X15          1.220e-04  8.843e-05   1.380 0.169704
X2:X7male    3.144e-04  9.242e-05   3.402 0.000853 ***
X3:X4        2.723e-03  7.085e-04   3.843 0.000177 ***
X6:X12      -4.576e-05  1.183e-05  -3.868 0.000162 ***
X9:X11      -1.017e-05  4.205e-06  -2.417 0.016801 *
X6:X14       1.651e-04  5.090e-05   3.243 0.001450 **
X14:X15     -4.726e-06  1.782e-06  -2.652 0.008836 **
X7male:X8   -5.439e-03  1.933e-03  -2.814 0.005535 **
X2:X11       8.366e-06  2.740e-06   3.053 0.002667 **
X2:X15      -4.768e-07  1.991e-07  -2.395 0.017812 *
X2:X13      -1.857e-05  9.828e-06  -1.889 0.060726 .
X6:X13      -7.270e-05  4.541e-05  -1.601 0.111414
X1:X15       5.308e-07  2.681e-07   1.980 0.049481 *
X3:X9       -1.137e-05  4.971e-06  -2.287 0.023538 *
X12:X13      8.134e-05  4.476e-05   1.817 0.071122 .
X9:X14      -1.917e-05  1.118e-05  -1.714 0.088454 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03224 on 154 degrees of freedom
Multiple R-squared:  0.6897,    Adjusted R-squared:  0.6333
F-statistic: 12.22 on 28 and 154 DF,  p-value: < 2.2e-16

> sse.fsp2=sum(fsp2$residuals^2)
> sse.fsp2
[1] 0.160076
> p.fsp2=length(fsp2$coefficients)
> p.fsp2
[1] 29
```

#10

Compare the BIC values of Model fs1 and Model fs2. What do you find?
Do AIC and BIC choose the same model among these two models or not?
Denote the model selected by AIC among the two models by Model 4.1 and that selected by BIC be Model 4.2. (It is possible that Model 4.1 and Model 4.2 are the same model.)

Ans:

- fs1 BIC= -229.3413
- fs2 BIC= -184.2004
- Compare these two BICs, I found that fs1 has the smaller BIC.
- Among these two models, AIC chooses fsp2, BIC chooses fsp1.
- Shown in R code: model denoting.

R code:

```
# 10
bic.fsp1=n*log(sse.fsp1)+log(n)*p.fsp1
bic.fsp1
bic.fsp2=n*log(sse.fsp2)+log(n)*p.fsp2
bic.fsp2

model4.1=fsp2       # in terms of AIC
model4.2=fsp1       # in terms of BIC
```

R console:

```
> # 10
> bic.fsp1=n*log(sse.fsp1)+log(n)*p.fsp1
> bic.fsp1
[1] -229.3413
> bic.fsp2=n*log(sse.fsp2)+log(n)*p.fsp2
> bic.fsp2
[1] -184.2004
> model4.1=fsp2       # in terms of AIC
> model4.2=fsp1       # in terms of BIC
```

Model validation.

We now consider validation of the models (Model 3.1, Model 3.2, Model 3.3, Model 4.1, Model4.2) you selected in the previous studies.

#11

Internal validation. We use PRESS for this purpose. Calculate PRESS for each of these models. Comment.

Ans:

- Shown in R console: press results.
- Comment: model4.1 has the smallest PRESS value while the others have the similar PRESS values.

R code:

```
# 11
# PRESS=sum( ( ei/(1-hii) )^2) where e=(yi-yiminusihat)
press.model3.1=sum(model3.1$residuals^2/(1-lm.influence(model3.1)$hat)^2)
press.model3.1
press.model3.2=sum(model3.2$residuals^2/(1-lm.influence(model3.2)$hat)^2)
press.model3.2
press.model3.3=sum(model3.3$residuals^2/(1-lm.influence(model3.3)$hat)^2)
press.model3.3
press.model4.1=sum(model4.1$residuals^2/(1-lm.influence(model4.1)$hat)^2)
press.model4.1
press.model4.2=sum(model4.2$residuals^2/(1-lm.influence(model4.2)$hat)^2)
press.model4.2
# Since model4.1 has the smallest PRESS, we choose it as the best model among the five.
```

R console:

```
> press.model3.1=sum(model3.1$residuals^2/(1-lm.influence(model3.1)$hat)^2)
> press.model3.1
[1] 0.252777
> press.model3.2=sum(model3.2$residuals^2/(1-lm.influence(model3.2)$hat)^2)
> press.model3.2
[1] 0.2539834
> press.model3.3=sum(model3.3$residuals^2/(1-lm.influence(model3.3)$hat)^2)
> press.model3.3
[1] 0.252575
> press.model4.1=sum(model4.1$residuals^2/(1-lm.influence(model4.1)$hat)^2)
> press.model4.1
[1] 0.2171946
> press.model4.2=sum(model4.2$residuals^2/(1-lm.influence(model4.2)$hat)^2)
> press.model4.2
[1] 0.2534834
```

#12

External validation using the validation set.

For each of these models (Model 3.1, Model 3.2, Model 3.3, Model 4.1, Model4.2), calculate the mean squared prediction error (MSPR), i.e., you use the model to predict 183 observations in the validation set and calculate the averaged squared prediction error. How do these MSPRs compare with the respective PRSSE/n (here n is the sample size of the training data, i.e., 183). Which model has the smallest MSPR?

Ans:

- Shown in R console: MSPR for these 5 models.
- Compare these MSPRs with the respective PRESS/n: for model3.1, model3.2 and model3.3, their mspr values are slightly smaller than their respective press/n values; for model4.1 and model4.2, their press/n values are slightly smaller than the respective msprs.
- model3.3 has the smallest MSPR, 0.00134099.

R code:

```
# 12
size=nrow(data.t)
size

mspr=function(model){
 yhat=predict(model, data.v)
 mspr = sum((data.v$Ytrans-yhat)^2) / size
 print(mspr)
 return(mspr)
}

mspr.model3.1=mspr(model3.1)
mspr.model3.2=mspr(model3.2)
mspr.model3.3=mspr(model3.3)
mspr.model4.1=mspr(model4.1)
mspr.model4.2=mspr(model4.2)
# choose model3.3 because it has the smallest mspr.

press.model3.1/size
press.model3.2/size
press.model3.3/size
press.model4.1/size
press.model4.2/size
```

R console: (contains screenshots because word format somehow messes up the columns)

```
> # 12
> size=nrow(data.t)
> size
[1] 183
> mspr=function(model){
+   yhat=predict(model, data.v)
+   mspr = sum((data.v$Ytrans-yhat)^2) / size
+   print(mspr)
+   return(mspr)
+ }
> mspr.model3.1=mspr(model3.1)
[1] 0.001368448
> mspr.model3.2=mspr(model3.2)
[1] 0.001377312
> mspr.model3.3=mspr(model3.3)
[1] 0.00134099
> mspr.model4.1=mspr(model4.1)
[1] 0.001797609
> mspr.model4.2=mspr(model4.2)
[1] 0.00152642
> press.model3.1/size
[1] 0.001381295
> press.model3.2/size
[1] 0.001387887
> press.model3.3/size
[1] 0.001380191
> press.model4.1/size
[1] 0.001186856
> press.model4.2/size
[1] 0.001385155
```

#13

Based on both internal and external validation, which model you would choose as the final model?
Fit the final model using the entire data set (training and validation combined) (Model 5).
Write down the fitted regression function and report the R summary() and anova() output.

Ans:

- Based on both internal and external validation, I choose model3.3 as the final model, since it has the smallest mspr.
- Shown in R console: fitted regression function, model5 summary, anova output.

R code:

```
# 13
model3.3
Frametypenew=model.matrix(~diabetes$X10-1)
model5=lm(Ytrans ~ X2 + X4 + X6 + Frametypenew[, 3] + X13 + X15, data=diabetes)
summary(model5)
anova(model5)
```

R console: (contains screenshots because word format somehow messes up the columns)

```
> # 13
> model3.3

Call:
lm(formula = Ytrans ~ X2 + X4 + X6 + Frametype[, 3] + X13 + X15,
    data = data.t)

Coefficients:
   (Intercept)             X2             X4             X6  Frametype[, 3]            X13            X15
     0.3713388      -0.0005352     -0.0036137     -0.0006709     -0.0119334     -0.0016735     -0.0000125

> Frametypenew=model.matrix(~diabetes$X10-1)
> model5=lm(Ytrans ~ X2 + X4 + X6 + Frametypenew[, 3] + X13 + X15, data=diabetes)
```

```
> summary(model5)

Call:
lm(formula = Ytrans ~ X2 + X4 + X6 + Frametypenew[, 3] + X13 + 
    X15, data = diabetes)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.154503 -0.020705 -0.001382  0.019680  0.150207 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)       3.414e-01  1.536e-02  22.221  < 2e-16 ***
X2               -4.947e-04  3.824e-05 -12.937  < 2e-16 ***
X4               -3.665e-03  1.187e-03  -3.088  0.00217 ** 
X6               -6.525e-04  1.230e-04  -5.306 1.97e-07 ***
Frametypenew[, 3]  2.008e-03  4.774e-03   0.421  0.67422    
X13              -1.061e-03  3.737e-04  -2.839  0.00479 ** 
X15              -1.328e-05  6.176e-06  -2.150  0.03223 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03628 on 359 degrees of freedom
Multiple R-squared:  0.5075,    Adjusted R-squared:  0.4993 
F-statistic: 61.66 on 6 and 359 DF,  p-value: < 2.2e-16

> anova(model5)
Analysis of Variance Table

Response: Ytrans
                   Df  Sum Sq Mean Sq  F value    Pr(>F)    
X2                  1 0.39753 0.39753 302.0648 < 2.2e-16 ***
X4                  1 0.02794 0.02794  21.2300 5.667e-06 ***
X6                  1 0.04221 0.04221  32.0769 3.041e-08 ***
Frametypenew[, 3]   1 0.00377 0.00377   2.8640  0.091448 .  
X13                 1 0.00936 0.00936   7.1122  0.008003 ** 
X15                 1 0.00608 0.00608   4.6223  0.032227 *  
Residuals         359 0.47245 0.00132                        
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```