Lab 6

Nov. 2019

Getting Started

In this lab, we explore a data set about diabetes.
The data consist of 19 variables on 403 subjects who were interviewed in a study to understand the prevalence of obesity, diabetes, and other risk factors in central Virginia for African Americans. According to Dr John Hong, Diabetes Mellitus Type II (adult onset diabetes) is associated most strongly with obesity. The waist/hip ratio may be a predictor in diabetes and heart disease. DM II is also associated with hypertension.

More on Reading in Data

Sometimes the data comes with column names and sometimes it doesn't. For instance, the diabetes data has the column names as the first row. To tell R that the first row is just names and not data, we use

```
diabetes = read.table('diabetes.txt', header=TRUE)
```

Also, sometimes the data will be separated by commas, instead of by white space. The default for read.table() is to look for white space to distinguish between variables, so if your data has commas instead, you can use the option sep = ',' inside read.table().

A nice tool before you do any analysis is to check the types of variables you have. Factors correspond to categorical variables ("factor" means categorical variables; "integer" means integer valued variables; "numeric" means quantitative variables):

```
sapply(diabetes,class) # find out what types of variables you have
       id      chol  stab.glu       hdl     ratio     glyhb  location
"integer" "integer" "integer" "integer" "numeric" "numeric"  "factor"
...
```

Categorical Variables (factors)

Suppose we are interested in exploring a categorical variable from the diabetes data. Let's take a look at the frame variable. To check how many levels frame has, and what their names are, you can do:

```
levels(diabetes$frame)
[1] ""       "large"  "medium" "small"
```

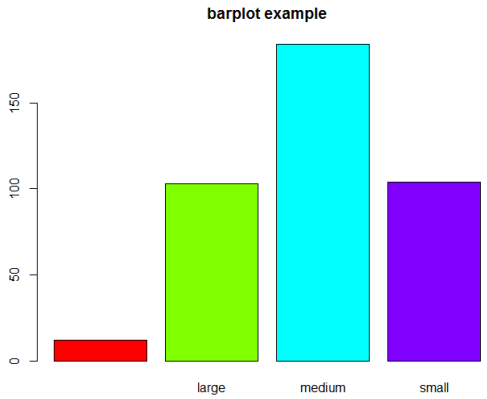
R tells us that there are 4 levels, but one of them is without a label (missing value). We can check which observations have the "" label, and then we can use the table function to get counts of each factor level:

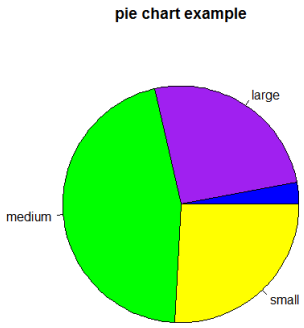
```
which(diabetes$frame==’’)
 [1]  51  64  70 109 111 153 225 283 328 333 349 381

table(diabetes$frame)

        large medium  small
    12    103    184    104
```

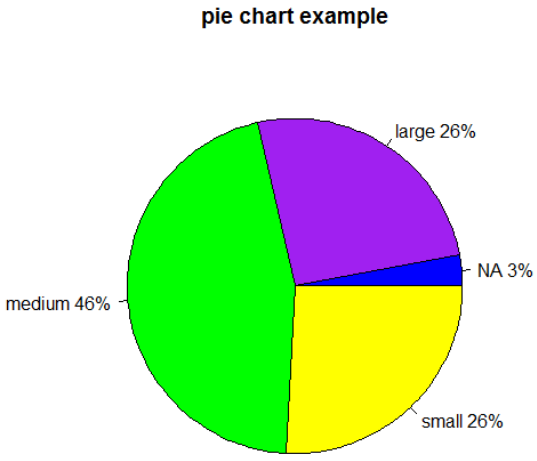
So 12 cases have a missing value for frame; 103 have the class "large," etc.

Next, we can use this table to create a bar chart and a pie chart. If you want to annotate the percentages in the pie chart, follow the second version.

```
### bar chart
barplot(table(diabetes$frame),col=rainbow(4),main=’barplot example’)

### pie chart without percentages
pie(table(diabetes$frame),col=c(’blue’,’purple’,’green’,’yellow’),
 main=’pie chart example’)

### pie chart with percentages
lbls=c(’NA’,’large’,’medium’,’small’)
pct=round(100*table(diabetes$frame)/n)
lab=paste(lbls,pct)
lab=paste(lab,’%’,sep=’’)
lab
[1] "NA 3%"      "large 26%"  "medium 46%" "small 26%"
pie(table(diabetes$frame),labels=lab,col=c(’blue’,’purple’,’green’,’yellow’),
 main=’pie chart example’)
```

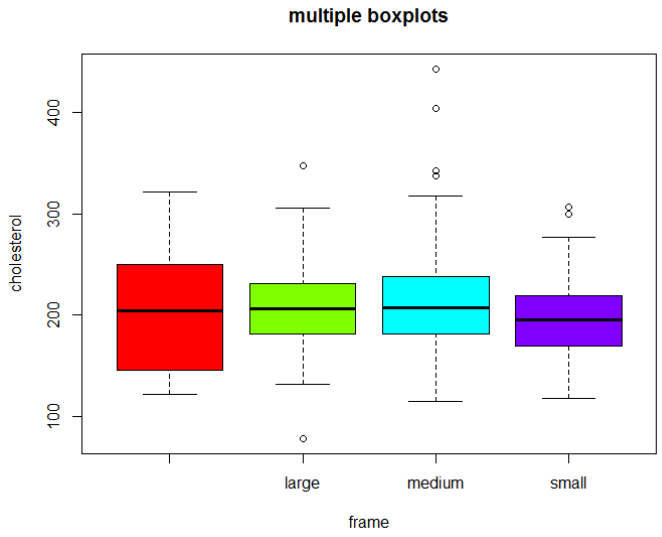






We also might be interested in, say, how the cholesterol levels are distributed for the different frame levels. For this we might use a grouped boxplot:

```
boxplot(diabetes$chol~diabetes$frame,main='multiple boxplots',
 xlab='frame',ylab='cholesterol',col=rainbow(4))
```



Dummy Variables (a.k.a. Indicator Variables)

When we quantify a categorical variable with r factor levels (classes), we use $r-1$ indicator, or dummy variables. You have seen in class that if a categorical variable only has 2 factor levels (say, male and female), then we only need one indicator variable. Here is how we can do this manually in R:

```
indicator=rep(0,n)                              # start with a vector of 0's
                                                # n is the sample size
indicator[which(diabetes$gender=='male')]=1     # replace male entries with 1
head(indicator)
[1] 0 0 0 1 1 1
head(diabetes$gender)
[1] female female female male   male   male
Levels: female male
```

Notice that the indicator variable matches with the gender variable. Now let's go back to the frame example. Here we have 3 different factor levels, so we are going to need 2 indicator variables. For the purposes of illustration, we will remove the observations where frame = "", but we will see how to deal with the missing values in the next section.

```
no.na=diabetes$frame[diabetes$frame != '']    # remove frame=''
length(no.na)                                 # get the new size
```

```
[1] 391
small=rep(0,391)                                  # start with 0's
small[which(no.na=='small')]=1                    # replace with 1's
med=rep(0,391)
med[which(no.na=='medium')]=1
head(cbind(small,med))
     small med
[1,]     0   1
[2,]     0   0
[3,]     0   0
[4,]     0   0
[5,]     0   1
[6,]     0   0
head(no.na)
[1] medium large  large  large  medium large
Levels:  large medium small
```

Missing Values (NA)

Not all data is complete. In this section we explore how we can deal with missing values using R. Here are some basic functions for dealing with NAs:

```
summary(diabetes$bp.1s)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   90.0   121.2   136.0   136.9   146.8   250.0       5

is.na(diabetes$bp.1s)[1:10]
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
```

Notice that if we take the summary of the variable bp.1s, it tells us how many missing values there are. Also, you can use the is.na function, which will return TRUE for every observation that has a missing value. To R, NA is a distinct value that it can interpret. There might arise some occasions where your data does not have NA for missing values, but rather something like '' or '.'. In these cases it would be a good idea to replace those values with the standard NA that R can recognize. In our case, this problem arises with the frame variable:

```
summary(diabetes$frame)
        large medium  small
    12    103    184    104
```

```
which(diabetes$frame==’’)                          # shows observation # of NA’s
is.na(diabetes$frame)=which(diabetes$frame==’’)  # replaces ’’ with NA
diabetes$frame=droplevels(diabetes$frame)          # takes away the old class ’’
summary(diabetes$frame)
 large medium  small   NA’s
   103    184    104     12
```

Some functions in R don't work nicely with missing values. For instance, the mean function will return NA if only 1 of the observations is missing. In this case, you can avoid the problem using the option na.rm, which removes NA's from the data:

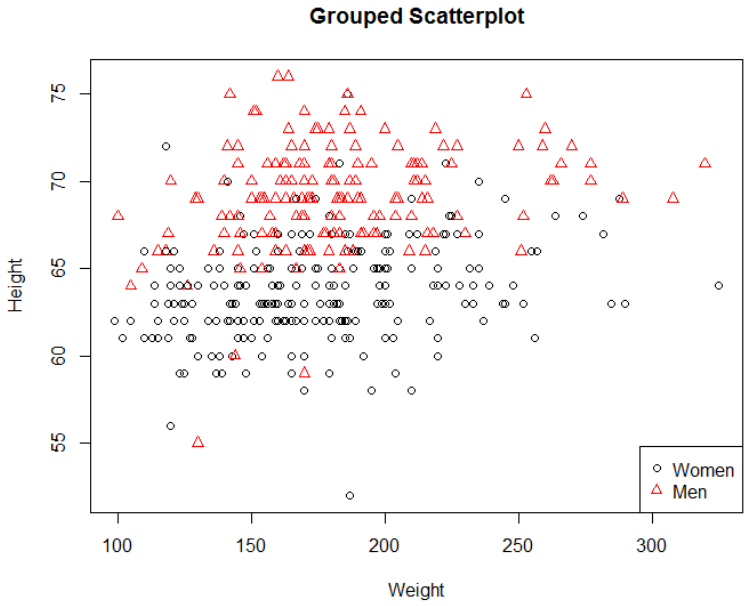
```
mean(diabetes$bp.1s)
[1] NA
mean(diabetes$bp.1s,na.rm=TRUE)
[1] 136.9045
```

Regression with Categorical Variables and NA

Note: In some data sets, a categorical variable may be coded numerically (how it is coded is usually provided in the document accompanying the data), say the three classes of "frame" might be coded as 1- small 2- medium 3-large. In such a case, we still need to treat it as a categorical variable, and explicitly use factor(frame) in the lm function, as well as with the pairs function. If we simply use frame, then it will be treated as a quantitative variable (because its values are numeric now, even though those values are only allocated codes for the three classes) and give misleading results. What would happen if it is not claimed as a factor?

Let's try a simple example using height as the response variable with predictors weight and gender. To make a scatterplot grouped by gender you can write:

```
plot(diabetes$weight,diabetes$height,pch=as.integer(diabetes$gender),
     col=as.integer(diabetes$gender),main=’Grouped Scatterplot’,
     ylab=’Height’,xlab=’Weight’)
legend(’bottomright’,legend=c(’Women’,’Men’),
     pch=c(1,2),col=c(1,2))                        # add a legend
```



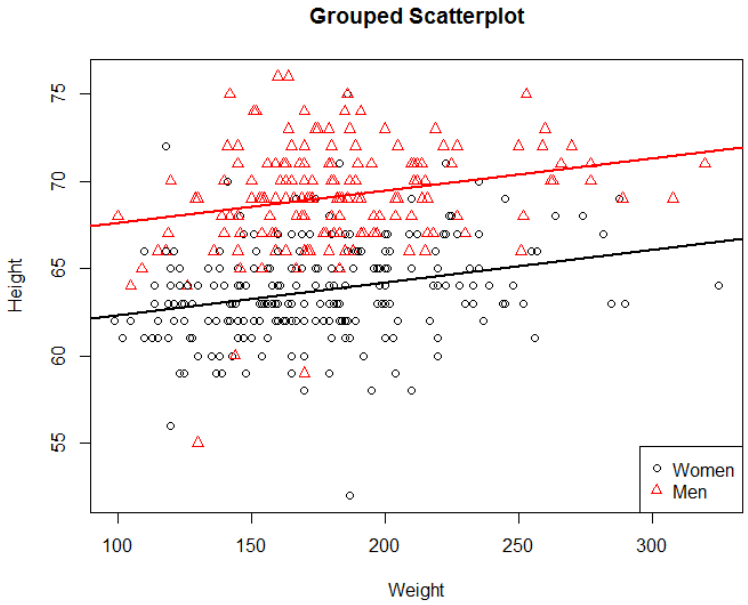
Now let's fit the model and get the coefficients so that we can draw the respective regression lines onto our plot (note: we are not performing any model diagnostics here, this is simply for a plotting demonstration).

```
weight.fit=lm(height~weight+factor(gender),data=diabetes)
summary(weight.fit)

Call:
lm(formula = height ~ weight + factor(gender), data = diabetes)

Residuals:
     Min       1Q   Median       3Q      Max
-13.1471  -1.6900  -0.1212   1.6064  11.0551

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 60.462810   0.634744  95.255  < 2e-16 ***
weight       0.018721   0.003477   5.384 1.25e-07 ***
gendermale   5.250624   0.283826  18.499  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 2.783 on 394 degrees of freedom
  (6 observations deleted due to missingness)
Multiple R-squared: 0.4986,Adjusted R-squared: 0.4961
F-statistic: 195.9 on 2 and 394 DF,  p-value: < 2.2e-16
```

From the summary, we see that the regression line for the female group has intercept 60.463, with a slope of .0187 while the regression line for the male group has intercept $60.463 + 5.251 = 65.714$ with the same slope as the female regression line. We can add lines to our plot using abline:

```
abline(60.46281,.018721,col=1,lwd=2)
abline(60.46281+5.250624,.018721,col='red',lwd=2)
```



Now suppose we'd like to make male instead of female as our reference level. We need to specify the reference level by function "relevel":

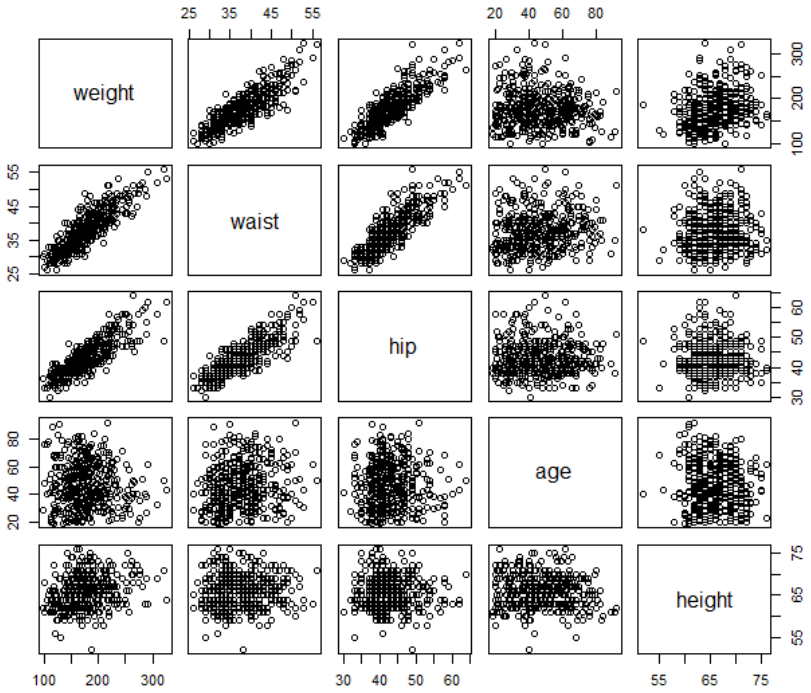
```
weight.fit2 = lm(height~weight+relevel(factor(gender), ref="male"), data=diabetes)
summary(weight.fit2)

Call:
lm(formula = height ~ weight + relevel(factor(gender), ref = "male"),
```

```
data = diabetes)

Residuals:
Min       1Q   Median       3Q      Max
-13.1471  -1.6900  -0.1212   1.6064  11.0551

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)                                  65.713434   0.668004  98.373  < 2e-16 ***
weight                                        0.018721   0.003477   5.384 1.25e-07 ***
relevel(factor(gender), ref = "male")female -5.250624   0.283826 -18.499  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.783 on 394 degrees of freedom
(6 observations deleted due to missingness)
Multiple R-squared:  0.4986,Adjusted R-squared:  0.4961
F-statistic: 195.9 on 2 and 394 DF,  p-value: < 2.2e-16
```

From the summary, we see that the regression line for the male group has intercept 65.713, with a slope of .0187 while the regression line for the female group has intercept $65.713 - 5.250 = 60.463$ with the same slope as the male regression line. We can see that the slope and intercepts are exactly the same as in our previously fitted model.

Now let's say we'd like to run a more complex regression using weight as the response variable. First we can take a look at the scatterplot matrix with some of the variables: weight, waist, hip, age, frame, gender and height (note: for simplicity of illustration, we are not considering all variables, and we will not put frame and gender in the scatterplot matrix because they are not quantitative):

```
pairs(~weight+waist+hip+age+height,data=diabetes)
```



We can see that these relationships are fairly linear, so a linear regression model might be a good fit. Now we can fit the model, but before we do, we have to think about the categorical variables and how we want to deal with missing values.

For categorical variables, we must make sure that R treats them the right way, so we use the factor() function to tell R when a variable is to be treated as categorical. For missing values, we have two options using na.action. If we choose na.exclude, R doesn't use the missing values, but keeps their position for residuals and fitted values. If we choose na.omit, R removes the missing value observations completely from the analysis. These can be shown by example:

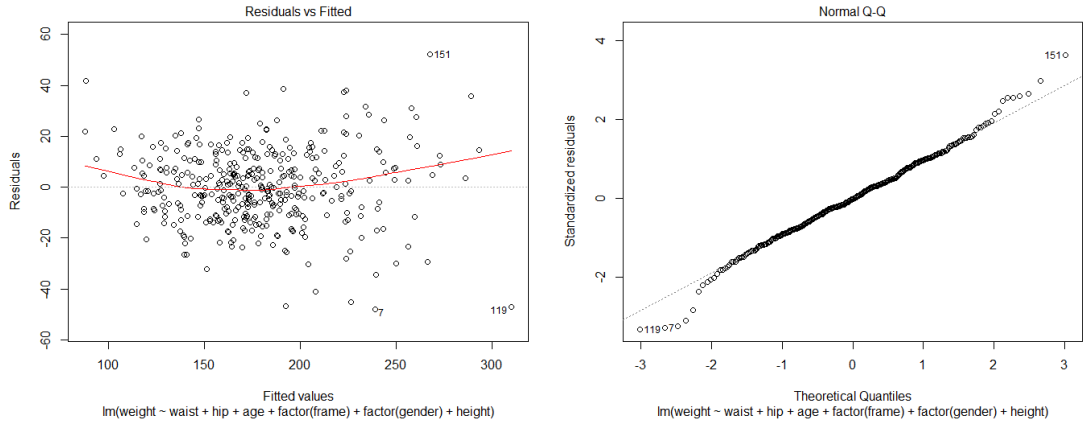
```
fit1 = lm(weight ~ waist + hip + age + factor(frame) + factor(gender) +
 height, na.action=na.omit, data=diabetes)
fit2 = lm(weight ~ waist + hip + age + factor(frame) + factor(gender) +
 height, na.action=na.exclude, data=diabetes)
length(residuals(fit1))   # less than the total number of
                          # subjects in the data, which is 403
[1] 384
```

```
length(residuals(fit2))   # residual for every case in the data;
                          # for those with missing values, NA is returned
[1] 403
summary(residuals(fit1))  # no NA's
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-47.910  -9.360  -0.295   0.000   9.400  52.150
summary(residuals(fit2))  # NA's
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-47.910  -9.360  -0.295   0.000   9.400  52.150      19
```

We might want to check some diagnostics plots and summary:

```
plot(fit1,which=1)
plot(fit1,which=2)
summary(fit1)

Call:
lm(formula = weight ~ waist + hip + age + factor(frame) + factor(gender) +
    height, data = diabetes, na.action = na.omit)

Residuals:
    Min      1Q  Median      3Q     Max
-47.915  -9.360  -0.295   9.400  52.150

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         -198.99606   18.52113 -10.744  < 2e-16 ***
waist                  2.66094    0.27201   9.782  < 2e-16 ***
hip                    3.83546    0.27783  13.805  < 2e-16 ***
age                   -0.33165    0.05034  -6.588 1.51e-10 ***
factor(frame)medium   -5.09454    2.01218  -2.532 0.011752 *
factor(frame)small    -9.52146    2.54012  -3.748 0.000206 ***
factor(gender)male     9.27262    2.35885   3.931 0.000101 ***
height                 1.93249    0.27635   6.993 1.24e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.75 on 376 degrees of freedom
  (19 observations deleted due to missingness)
```

```
Multiple R-squared: 0.8701,Adjusted R-squared: 0.8677
F-statistic: 359.8 on 7 and 376 DF,  p-value: < 2.2e-16
```



Finally, there is another commonly used method of dealing with missing values for quantitative variables. Here we replace the missing value by the mean of all the non-missing observations:

```
summary(diabetes$bp.1s)    # NA's
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   90.0   121.2   136.0   136.9   146.8   250.0       5

diabetes$bp.1s[is.na(diabetes$bp.1s)]=mean(diabetes$bp.1s,na.rm=TRUE)

summary(diabetes$bp.1s)   # no NA's
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   90.0   122.0   136.0   136.9   146.0   250.0
```

Note that this does not change the mean of the variable. **The above may only be done for predictor variables and we do not do this for response variable since it is what we are trying to model. Filling in missing values can also create biases, so be cautious.**

Polynomial Regression

The residual against fitted value plot above for fit1 shows a quadratic trend which suggests that we may include polynomial terms in the model. It is reasonable since

weight is proportional to the volume which is a 3-dimensional measure, while waist, hip and height are all 1-dimensional measures. Now we fit a second order model. First we delete rows with missing values.

```
diabetes1=diabetes[!is.na(diabetes$weight)  & !is.na(diabetes$waist) & !is.na(diabete
& !is.na(diabetes$age) & !is.na(diabetes$height) & !is.na(diabetes$gender)
& !is.na(diabetes$frame),]
```

In order to reduce the correlation between the linear and quadratic terms, we center the data.

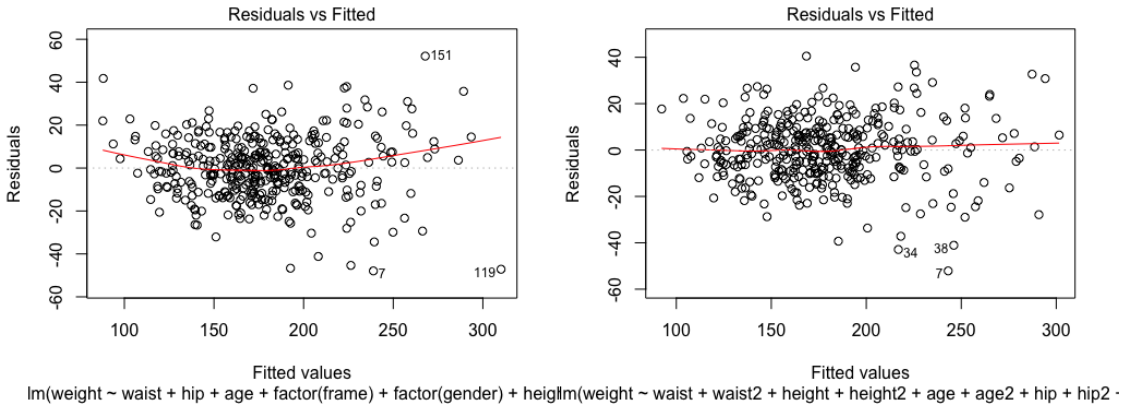
```
center=function(x) x-mean(x)

waist=center(diabetes1$waist)
hip=center(diabetes1$hip)
age=center(diabetes1$age)
height=center(diabetes1$height)
waist2=center(diabetes1$waist^2)
hip2=center(diabetes1$hip^2)
age2=center(diabetes1$age^2)
height2=center(diabetes1$height^2)
```

Now we fit a second order full model.

```
weight=diabetes1$weight
frame=diabetes1$frame
gender=diabetes1$gender

fit3=lm(weight~waist+waist2+height+height2 + age+ age2+hip
+hip2 +waist*height*hip*age+ factor(frame)+factor(gender))
Or
fit3 = lm(weight~waist+I(waist^2)+height+I(height^2)+age
+I(age^2)+hip+I(hip^2)+waist*height*hip*age+ factor(frame)+factor(gender))
```

To do diagnostic again and compare it to fit1.

```
plot(fit1,which=1)
plot(fit3,which=1)
```



No systematic pattern is observed for the second order model.