#2. I'll prove it in two different ways.

① $t_i = \dfrac{\hat{e}_{[i]}\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}/(n-p-2)}}$, and I want to show $\dfrac{\hat{e}_{[i]}\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}/(n-p-2)}} = r_i\sqrt{\dfrac{n-p-2}{n-p-1-r_i^2}}$.

And, $r_i\sqrt{\dfrac{n-p-2}{n-p-1-r_i^2}} = \dfrac{r_i}{\sqrt{\dfrac{n-p-1-r_i^2}{n-p-2}}}$. So, I need to prove $\dfrac{\hat{e}_{[i]}\sqrt{1-h_{ii}}}{\sqrt{\dfrac{RSS_{[i]}}{n-p-2}}} = \dfrac{r_i}{\sqrt{\dfrac{n-p-1-r_i^2}{n-p-2}}}$

It implies that I need to prove $\dfrac{\hat{e}_{[i]}\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}}} = \dfrac{r_i}{\sqrt{n-p-1-r_i^2}}$

As $\hat{e}_{[i]} = \dfrac{\hat{e}_i}{1-h_{ii}}$, $\dfrac{\hat{e}_{[i]}\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}}} = \dfrac{\hat{e}_i/\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}}} = \dfrac{\hat{e}_i/\hat{\sigma}\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}}/\hat{\sigma}} = \dfrac{r_i}{\sqrt{RSS_{[i]}}/\hat{\sigma}}$.

So, I need to prove $\sqrt{n-p-1-r_i^2} = \dfrac{\sqrt{RSS_{[i]}}}{\hat{\sigma}}$.

As $\hat{\sigma} = \sqrt{\dfrac{RSS}{n-p-1}}$, $\sqrt{n-p-1-r_i^2} = \sqrt{\dfrac{RSS}{\hat{\sigma}^2} - r_i^2} = \sqrt{\dfrac{RSS}{\hat{\sigma}^2} - \dfrac{\hat{e}_i^2}{\hat{\sigma}^2(1-h_{ii})}} = \dfrac{1}{\hat{\sigma}}\sqrt{\dfrac{RSS(1-h_{ii}) - \hat{e}_i^2}{(1-h_{ii})}}$.

So, $RSS_{[i]} = \dfrac{RSS(1-h_{ii}) - \hat{e}_i^2}{(1-h_{ii})}$, and $\boxed{RSS_{[i]} = RSS - \dfrac{\hat{e}_i^2}{1-h_{ii}}}$.

<proof>

$RSS_{[i]} = Y_{[i]}^T Y_{[i]} - Y_{[i]}^T X_{[i]}(X_{[i]}^T X_{[i]})^{-1} X_{[i]}^T Y_{[i]}$ from $RSS = \sum(y_i - X\hat{\beta})^2$

$= (Y^T Y - y_i^2) - Y_{[i]}^T X_{[i]} \hat{\beta}_{[i]}$.

And, $Y_{[i]}^T X_{[i]} \hat{\beta}_{[i]} = (Y^T X - y_i x_i^T)\left\{\hat{\beta} - \dfrac{[(X^TX)^{-1} - x_i\hat{e}_i]}{1-h_{ii}}\right\} = Y^T X\hat{\beta} - y_i x_i^T\hat{\beta} - \dfrac{x_i^T\hat{\beta}\hat{e}_i}{(1-h_{ii})} + \dfrac{y_i h_{ii}\hat{e}_i}{(1-h_{ii})}$

$= Y^T H Y - y_i x_i^T\hat{\beta} + \dfrac{y_i\hat{e}_i}{(1-h_{ii})} - \dfrac{x_i^T\hat{\beta}\hat{e}_i}{(1-h_{ii})} - \dfrac{y_i\hat{e}_i}{(1-h_{ii})} + \dfrac{y_i h_{ii}\hat{e}_i}{(1-h_{ii})}$

$= Y^T H Y - y_i x_i^T\hat{\beta} + \dfrac{\hat{e}_i^2}{(1-h_{ii})} - y_i\hat{e}_i = Y^T H Y + \dfrac{\hat{e}_i^2}{(1-h_{ii})} - y_i^2$.

So, $Y^T Y - y_i^2 - Y^T H Y - \dfrac{\hat{e}_i^2}{(1-h_{ii})} + y_i^2 = Y^T(I-H)Y - \dfrac{\hat{e}_i^2}{(1-h_{ii})} = \boxed{RSS - \dfrac{\hat{e}_i^2}{(1-h_{ii})}}$

② prove in other direction.

$$t_i = r_i \sqrt{\frac{n-p-2}{n-p-1-r_i^2}}, \text{ where } r_i = \frac{\hat{e}_i}{\hat{\sigma}\sqrt{1-h_{ii}}}$$

$$= \frac{\hat{e}_i}{\hat{\sigma}\sqrt{1-h_{ii}}} \sqrt{\frac{n-p-2}{n-p-1-r_i^2}} = \frac{\hat{e}_i}{\hat{\sigma}\sqrt{1-h_{ii}}} \sqrt{\frac{n-p-2}{n-p-1-\frac{\hat{e}_i^2}{\hat{\sigma}^2(1-h_{ii})}}}$$

$$= \frac{\hat{e}_i}{\hat{\sigma}\sqrt{1-h_{ii}}} \sqrt{\frac{n-p-2}{\frac{\hat{\sigma}^2(1-h_{ii})(n-p-1)-\hat{e}_i^2}{\hat{\sigma}^2(1-h_{ii})}}} = \frac{\hat{e}_i}{\hat{\sigma}\sqrt{1-h_{ii}}} \sqrt{\frac{\hat{\sigma}^2(1-h_{ii})(n-p-2)}{\hat{\sigma}^2(1-h_{ii})(n-p-1)-\hat{e}_i^2}}$$

$$= \hat{e}_i \sqrt{\frac{n-p-2}{\hat{\sigma}^2(1-h_{ii})(n-p-1)-\hat{e}_i^2}}$$

And, I know $\hat{e}_{[i]} = \frac{\hat{e}_i}{1-h_{ii}}$.

So, $\hat{e}_{[i]}(1-h_{ii}) \sqrt{\frac{n-p-2}{\hat{\sigma}^2(1-h_{ii})(n-p-1)-\hat{e}_{[i]}^2(1-h_{ii})^2}}$ and I know $RSE = \hat{\sigma} = \sqrt{\frac{RSS}{n-p-1}}$.

So, $\hat{e}_{[i]}\sqrt{1-h_{ii}} \sqrt{\frac{n-p-2}{\hat{\sigma}^2(n-p-1)-\hat{e}_{[i]}^2(1-h_{ii})}}$ and $t_i = \frac{\hat{e}_{[i]}\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}/(n-p-2)}}$.

So, $\sqrt{\frac{n-p-2}{\hat{\sigma}^2(n-p-1)-\hat{e}_{[i]}^2(1-h_{ii})}} = \frac{1}{\sqrt{RSS_{[i]}/(n-p-2)}} = \sqrt{\frac{n-p-2}{RSS_{[i]}}}$.

So, $RSS_{[i]} = \hat{\sigma}^2(n-p-1) - \hat{e}_{[i]}^2(1-h_{ii})$.

And since $\hat{\sigma} = \sqrt{\frac{RSS}{n-p-1}}$ and $\hat{e}_{[i]} = \frac{\hat{e}_i}{1-h_{ii}}$,

$$RSS_{[i]} = \frac{RSS}{n-p-1}(n-p-1) - \frac{\hat{e}_i^2}{1-h_{ii}}.$$

Thus, $\boxed{RSS_{[i]} = RSS - \frac{\hat{e}_i^2}{1-h_{ii}}}$

Jin Kweon (3032235207) HW3

Jin Kweon

10/7/2017

Problem 1

Warning: Since this is open-ended question, and there is no strict rule of doing questions in order. I go back and forth a lot, and I appologize to the *Reader: Annie* about the confusion when she grades my homework. However, I tried to make everything clear, and wrote summary under each part.

http://www.stat.ucla.edu/projects/datasets/ant-explanation.html

```
# Will not run download again since it makes the code run slow.
#download.file("http://www.stat.ucla.edu/projects/datasets/thatch-ant.dat",
#               destfile = "data.txt")

# I will skip the data filtering codes, as this is irrelevant with our analysis.
# a <- read.table("download1199rows.txt", header = T, sep = ",")
# a <- rbind(a[(a$Colony == 1),], a[(a$Colony == 2),], a[(a$Colony == 3),],
#            a[(a$Colony == 4),], a[(a$Colony == 5),], a[(a$Colony == 6),])

ant <- read.table(file = "data.txt", header = T)
dim(ant)
```

```
## [1] 649   6
```

```
ant <- na.omit(ant)
str(ant)
```

```
## 'data.frame':    648 obs. of  6 variables:
##  $ Colony        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Distance      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Mass          : int  109 120 94 61 72 134 94 113 111 106 ...
##  $ Headwidth     : int  45 43 42 33 41 46 43 42 42 43 ...
##  $ Headwidth..mm.: num  1.9 1.81 1.77 1.39 1.73 ...
##  $ Size.class    : Factor w/ 6 levels "<30","\x80",">43",..: 3 6 6 4 6 3 6 6 6 6 ...
##  - attr(*, "na.action")=Class 'omit'  Named int 590
##   .. ..- attr(*, "names")= chr "590"
```

```
ant <- ant[,-5]
antoriginal <- ant

levels(as.factor(ant$Colony))
```

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

```
levels(as.factor(ant$Distance))
```

```
## [1] "0"  "1"  "4"  "7"  "10"
```

```
levels(as.factor(ant$Mass))
```

```
##   [1] "28"  "33"  "34"  "35"  "36"  "38"  "39"  "40"  "42"  "43"  "44"
##  [12] "45"  "46"  "47"  "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"
##  [23] "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"  "67"
##  [34] "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"  "78"
##  [45] "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"  "89"
##  [56] "90"  "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"  "100"
##  [67] "101" "102" "103" "104" "105" "106" "107" "108" "109" "110" "111"
##  [78] "112" "113" "114" "115" "116" "117" "118" "119" "120" "121" "122"
##  [89] "123" "125" "126" "127" "128" "129" "130" "131" "132" "133" "134"
## [100] "135" "136" "137" "138" "139" "140" "141" "142" "143" "144" "145"
## [111] "146" "147" "148" "149" "151" "152" "154" "156" "158" "159" "161"
## [122] "163" "165" "189"
```

```r
levels(as.factor(ant$Headwidth))
```

```
##  [1] "27" "28" "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40"
## [15] "41" "42" "43" "44" "45" "46" "47" "48" "49"
```

```r
levels(as.factor(ant$Size.class))
```

```
## [1] "<30"   "\x80"  ">43"   "30-34" "35-39" "40-43"
```

```r
ant %>% group_by(Size.class) %>% summarise(x = length(Size.class))
```

```
## # A tibble: 5 x 2
##   Size.class     x
##       <fctr> <int>
## 1        <30    13
## 2        >43   167
## 3      30-34    55
## 4      35-39   145
## 5      40-43   268
```

```r
ant$Size.class <- factor(ant$Size.class) #take out one level not used...

ant$Colony <- as.factor(ant$Colony)
ant$Distance <- as.factor(ant$Distance)
ant$Size.class <- as.factor(ant$Size.class)
str(ant)
```

```
## 'data.frame':    648 obs. of  5 variables:
##  $ Colony    : Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Distance  : Factor w/ 5 levels "0","1","4","7",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Mass      : int  109 120 94 61 72 134 94 113 111 106 ...
##  $ Headwidth : int  45 43 42 33 41 46 43 42 42 43 ...
##  $ Size.class: Factor w/ 5 levels "<30",">43","30-34",..: 2 5 5 3 5 2 5 5 5 5 ...
```

```r
levels(ant$Size.class) <- c("SS", "LL", "S", "M", "L")
```

I first removed the NA row. (590th observation) By the default, the columns colony is integer, distance is integer, mass is integer, headdwidth is integer, headwidth (mm) is numeric, and size is factor.

Also, when I observed the columns "Headwidth" and "Headwidth (mm)," they are the same but in different units. And, I found "Headwidth" column can be used more efficiently (bigger unit, so more interpretable), so I just removed "Headwidth (mm)" column.

The better way to say this is, if I consider including both of them into lm() function, they might show singularity errors. Since it causes multi-collinearity issue, as they actually mean the same thing.

Also, there are only 6 levels for "Colony," 5 levels for "Distance," and 6 levels for "Size.class" columns (and no data exist for x80 level - so I took it out...). In the future, I might switch them to categorical variables for interpretation purpose. All the measurements for these three variables have only a few integers, and I guess the scientists just approximate each observation. Thus, it is better idea for me to regard them as categorical.

Furtheremore, each colony has different number of ants, and when I interpret it, I should be aware of it.

Last but not least, I changed the names of the sizes to SS, S, M, L, and LL, for interpretation purpose. (correspond to <30, 30-34, 35-39, 40-43, >43)

Part a

From the worker conservative and energy conservative strategies, I could find some of the relationships.

Worker conservative strategy says that "distance $\propto c * mass$, where c > 0."

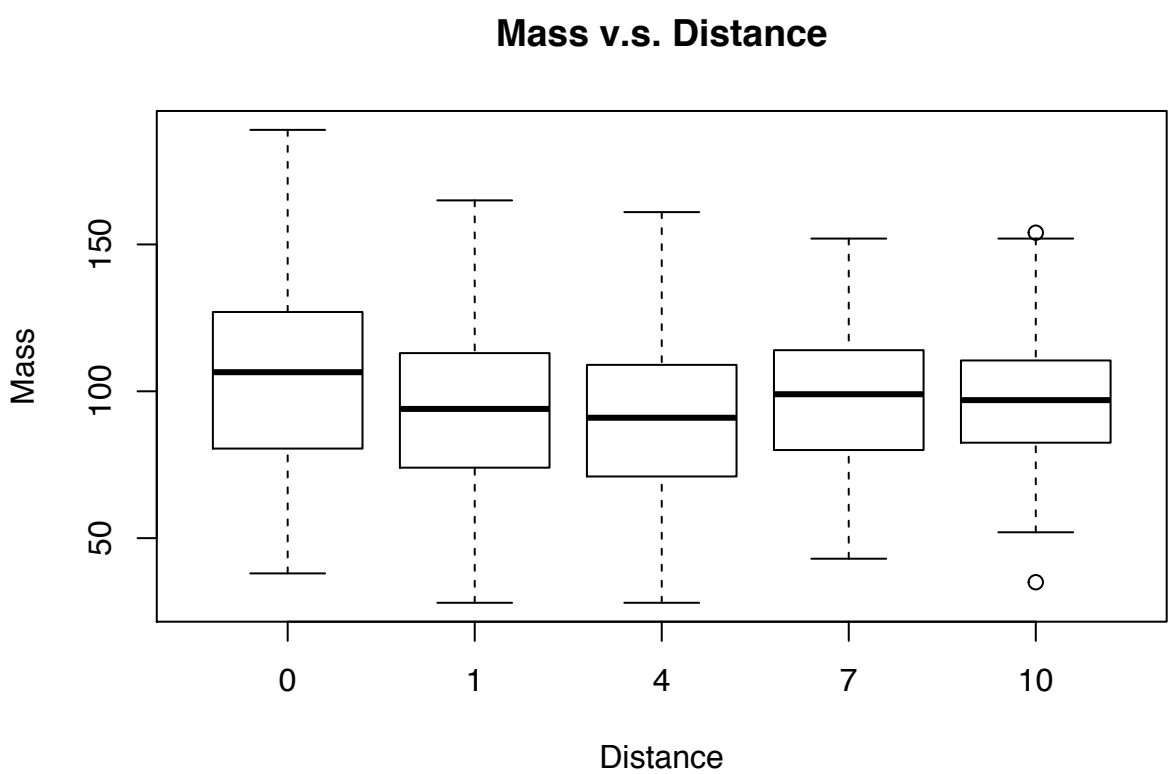
Energy conservative strategy says that "distance $\propto c * mass$, where c < 0."

Multiplot function is referred from http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/.

```
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  require(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
```

```r
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

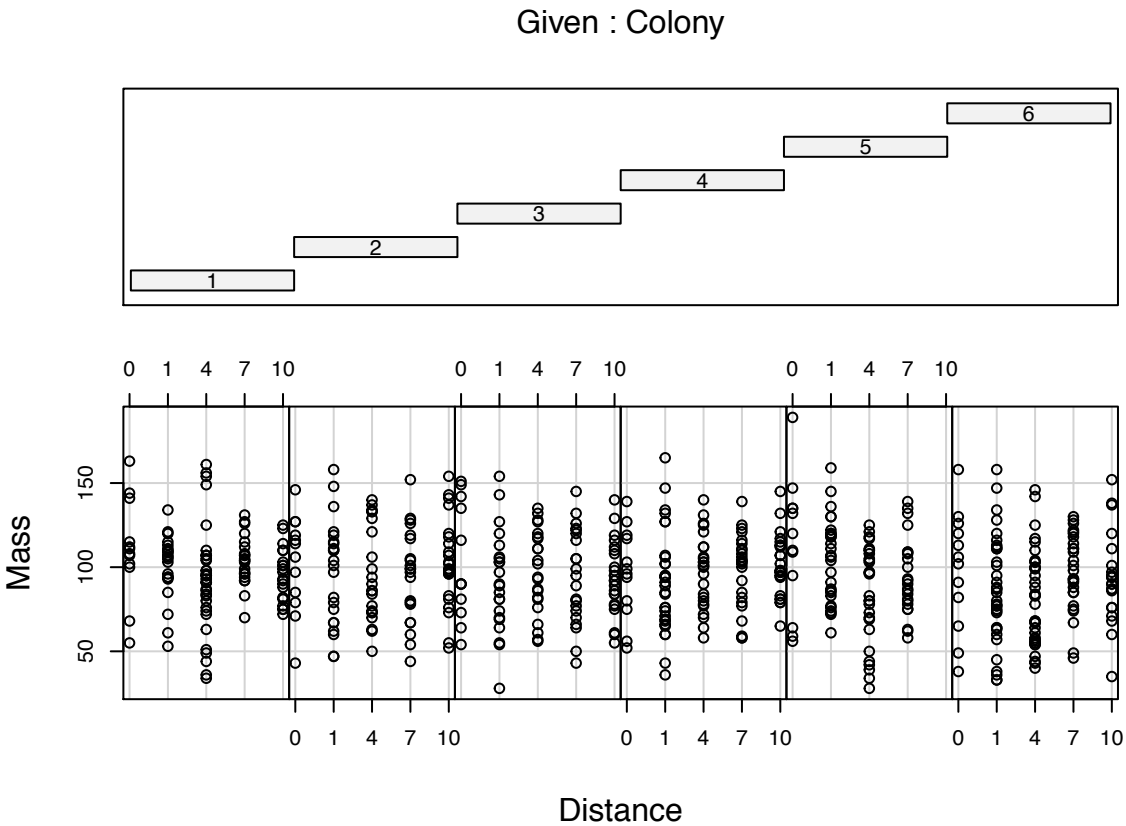
```r
plot(ant$Distance, ant$Mass, main = "Mass v.s. Distance",
     xlab = "Distance", ylab = "Mass")
```



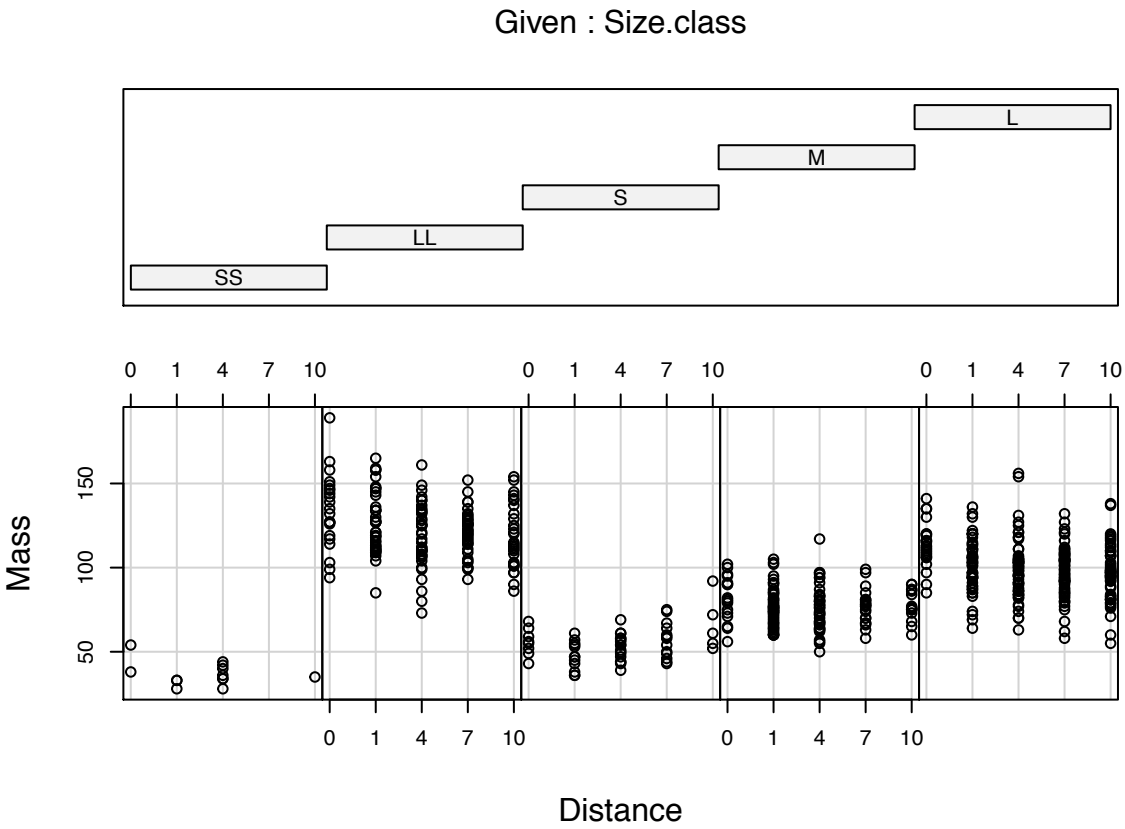
```r
print("By simply looking at the plot, it is hard to tell what foraging strategies ants are using...")
```

```
## [1] "By simply looking at the plot, it is hard to tell what foraging strategies ants are using..."
```

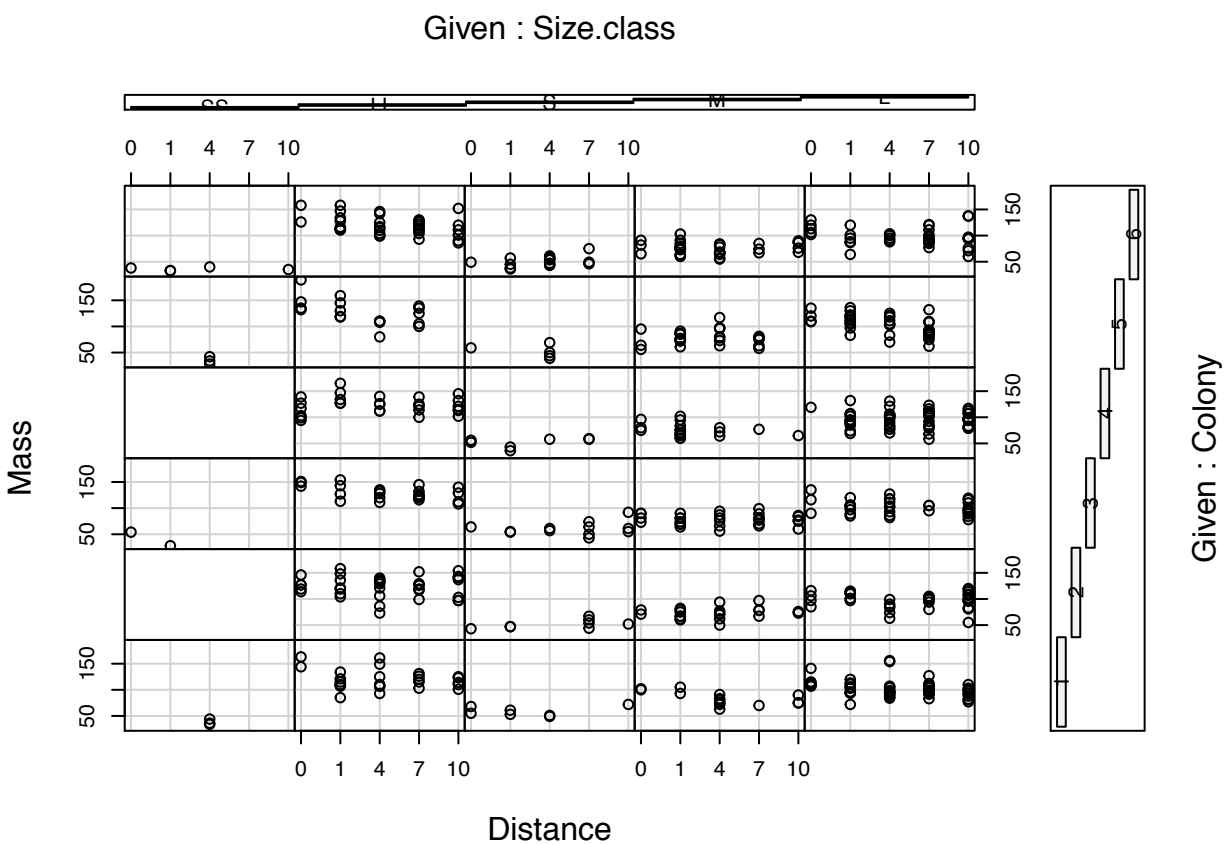
```r
coplot(Mass ~ Distance | Colony, data = ant, overlap = 0, rows = 1)
```




```r
coplot(Mass ~ Distance | Size.class, data = ant, overlap = 0, rows = 1)
```




```
coplot(Mass ~ Distance | Size.class * Colony, data = ant, overlap = 0, rows = 1)
```



```r
print("Yet, it is still hard to answer the scientific question...")
```

```
## [1] "Yet, it is still hard to answer the scientific question..."
```

```r
one <- aov(cbind(Distance, Mass) ~ as.factor(Colony), data = ant)
summary(one, test = "Wilks")
```

```
##  Response Distance :
##                     Df  Sum Sq Mean Sq F value  Pr(>F)
## as.factor(Colony)   5   19.21  3.8427  2.4464 0.03285 *
## Residuals         642 1008.41  1.5707
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response Mass :
##                     Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(Colony)   5   6677 1335.46  1.8364 0.1037
## Residuals         642 466872  727.21
```

```r
one_manova <- manova(cbind(Distance, Mass) ~ as.factor(Colony), data = ant)
summary(one_manova, test = "Wilks")
```

```
##                     Df   Wilks approx F num Df den Df  Pr(>F)
## as.factor(Colony)   5 0.96741   2.1418     10   1282 0.01905 *
## Residuals         642
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
two <- aov(cbind(Distance, Mass) ~ Size.class, data = ant)
summary(two, test = "Wilks")
```

```
##  Response Distance :
##              Df Sum Sq Mean Sq F value    Pr(>F)
## Size.class    4  32.89   8.222  5.3147 0.0003244 ***
## Residuals   643 994.74   1.547
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response Mass :
##              Df Sum Sq Mean Sq F value    Pr(>F)
## Size.class    4 318904   79726  331.49 < 2.2e-16 ***
## Residuals   643 154645     241
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
two_manova <- manova(cbind(Distance, Mass) ~ Size.class, data = ant)
summary(two_manova, test = "Wilks")
```

```
##             Df   Wilks approx F num Df den Df    Pr(>F)
## Size.class   4 0.30617   129.56      8   1284 < 2.2e-16 ***
## Residuals  643
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

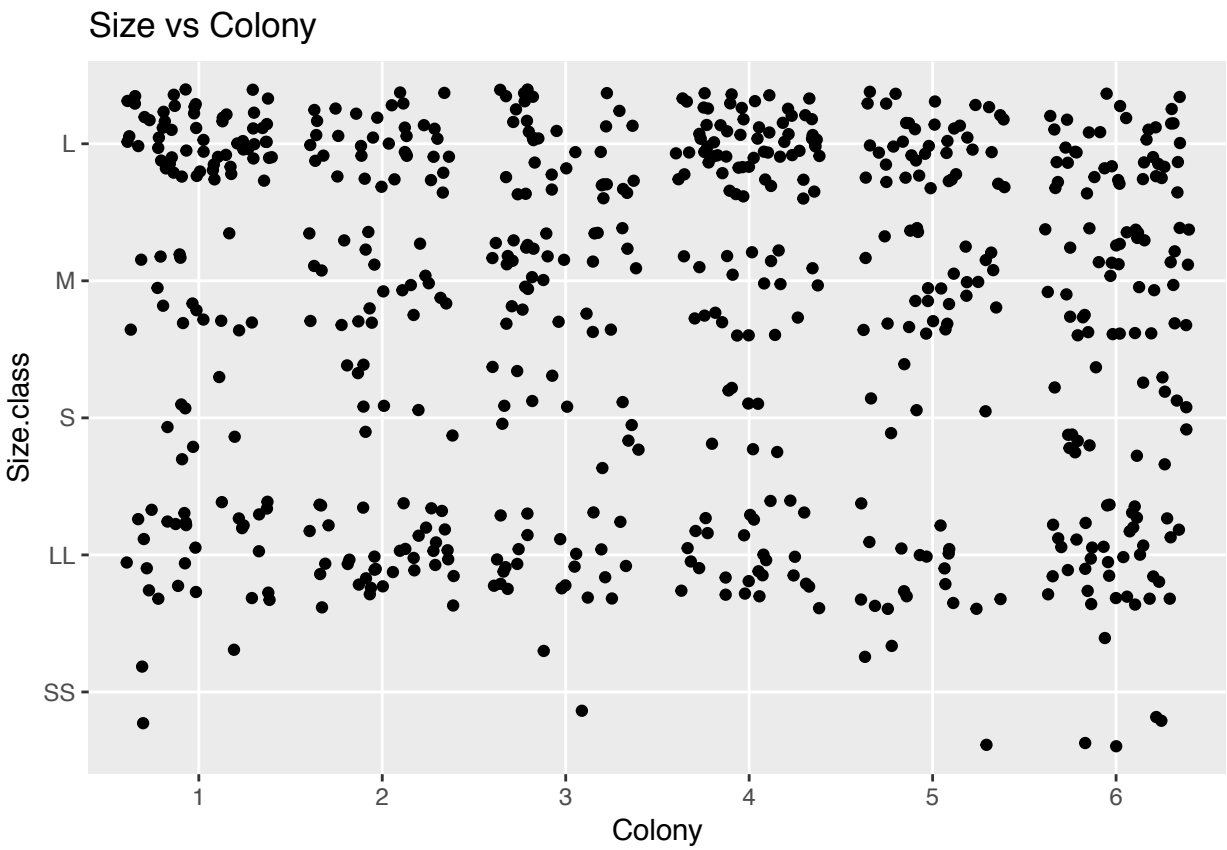
```r
three_int <- aov(cbind(Distance, Mass) ~ as.factor(Colony) * Size.class, data = ant)
summary(three_int, test = "Wilks")
```

```
##  Response Distance :
##                               Df Sum Sq Mean Sq F value    Pr(>F)
## as.factor(Colony)              5  19.21  3.8427  2.5120 0.0289509 *
## Size.class                     4  32.40  8.0998  5.2948 0.0003377 ***
## as.factor(Colony):Size.class  18  27.56  1.5309  1.0008 0.4566411
## Residuals                    620 948.46  1.5298
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response Mass :
##                               Df Sum Sq Mean Sq  F value    Pr(>F)
## as.factor(Colony)              5   6677    1335   5.5950 4.707e-05 ***
## Size.class                     4 316089   79022 331.0700 < 2.2e-16 ***
## as.factor(Colony):Size.class  18   2796     155   0.6508    0.8597
## Residuals                    620 147986     239
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

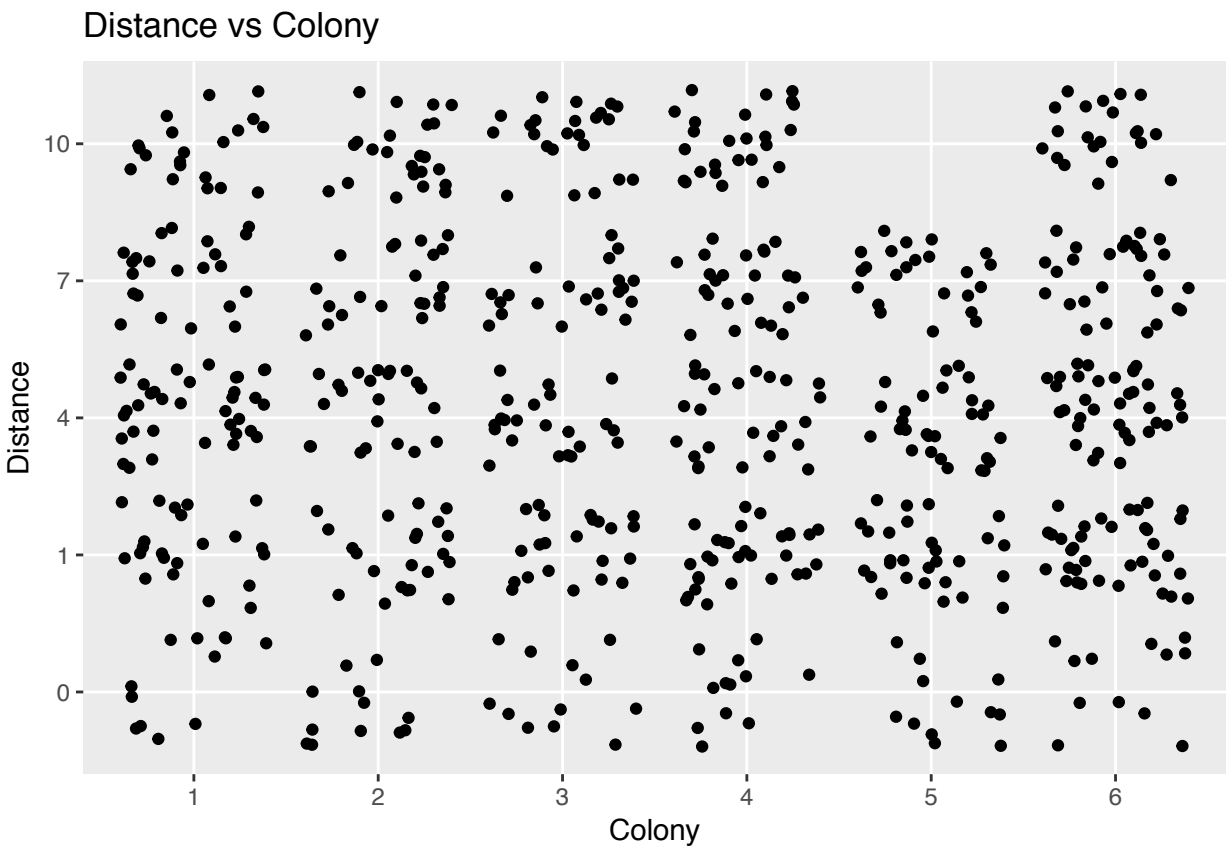
```r
three_manova <- manova(cbind(Distance, Mass) ~ as.factor(Colony) * Size.class, data = ant)
summary(three_manova, test = "Wilks")
```

```
##                               Df   Wilks approx F num Df den Df    Pr(>F)
## as.factor(Colony)              5 0.93528    4.212     10   1238 9.017e-06
## Size.class                     4 0.29919  128.168      8   1238 < 2.2e-16
## as.factor(Colony):Size.class  18 0.95396    0.820     36   1238    0.7666
## Residuals                    620
##
## as.factor(Colony)            ***
## Size.class                   ***
## as.factor(Colony):Size.class
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
ggplot(ant, aes(x = as.factor(Colony), y = Size.class)) + geom_jitter() +
  labs(title = "Size vs Colony", x = "Colony")
```



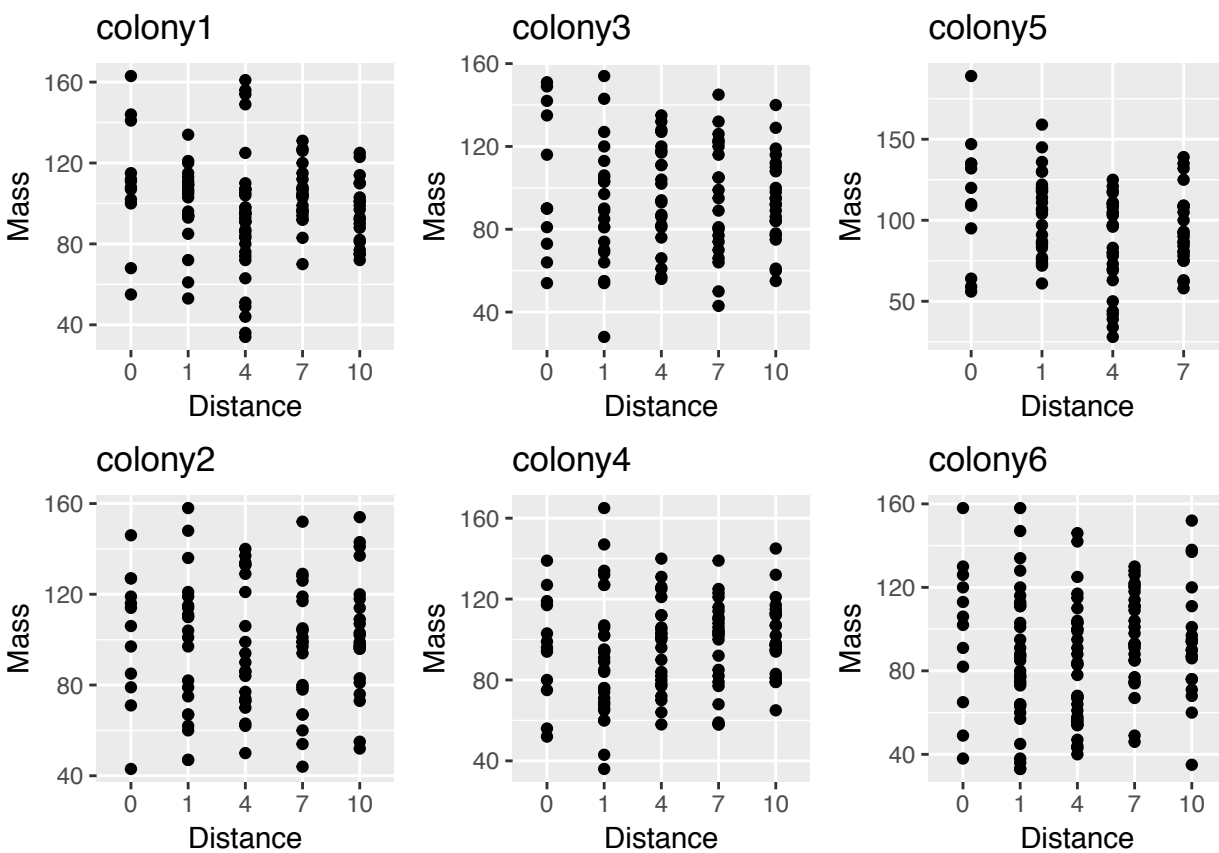

```
ggplot(ant, aes(x = as.factor(Colony), y = Distance)) + geom_jitter() +
  labs(title = "Distance vs Colony", x = "Colony")
```




```
colony1 <- ant %>% filter(Colony == 1)
colony2 <- ant %>% filter(Colony == 2)
colony3 <- ant %>% filter(Colony == 3)
colony4 <- ant %>% filter(Colony == 4)
colony5 <- ant %>% filter(Colony == 5)
colony6 <- ant %>% filter(Colony == 6)

colony1old <- antoriginal %>% filter(Colony == 1)
colony2old <- antoriginal %>% filter(Colony == 2)
colony3old <- antoriginal %>% filter(Colony == 3)
colony4old <- antoriginal %>% filter(Colony == 4)
colony5old <- antoriginal %>% filter(Colony == 5)
colony6old <- antoriginal %>% filter(Colony == 6)


sizeSS <- ant %>% filter(Size.class == "SS")
sizeS <- ant %>% filter(Size.class == "S")
sizeM <- ant %>% filter(Size.class == "M")
sizeL <- ant %>% filter(Size.class == "L")
sizeLL <- ant %>% filter(Size.class == "LL")

#Confirm whether dplyr works fine.
identical(nrow(ant), nrow(colony1) +nrow(colony2) + nrow(colony3) + nrow(colony4) +
            nrow(colony5) + nrow(colony6))
```

```
## [1] TRUE
```

```r
set <- list()
par(mfrow = c(2,3))
set[[1]] <- qplot(x = Distance, y = Mass, data = colony1, geom = c("point"), main = "colony1")
set[[2]] <- qplot(x = Distance, y = Mass, data = colony2, geom = c("point"), main = "colony2")
set[[3]] <- qplot(x = Distance, y = Mass, data = colony3, geom = c("point"), main = "colony3")
set[[4]] <- qplot(x = Distance, y = Mass, data = colony4, geom = c("point"), main = "colony4")
set[[5]] <- qplot(x = Distance, y = Mass, data = colony5, geom = c("point"), main = "colony5")
set[[6]] <- qplot(x = Distance, y = Mass, data = colony6, geom = c("point"), main = "colony6")

multiplot(plotlist = set, cols = 3)
```



Comment:

I can use co-plot(for both one factor and two factors), one-way anova (one factor), and two-way anova (two factors) to answer the first scientific question: *"Do different colonies use different foraging strategies? Is there some difference across size classes?"*

Also, I can use boxplot to answer the second scientific question: *"Are there differences across colonies in the distribution of sizes or distances of the member ants?"*

Furthermore, I can use histogram or smoothline, having mass and distance on axes, to answer the last scientific question: *"What are the strategies that are in use? Are any colonies especially similar or different"* It might be a good idea to use ggplot so I can differentiate colonies by color.

From the co-plot, I can definitely see the relationship between distance v.s. mass across size classes. As size gets bigger, the ants are carrying more foods (more mass). However, it is hard for me to make any conclusion of whether different colonies use different foraging strategies. (meaning that whether distance and mass are related) I think it is better to further investigate to conclude both of the scientific questions.

Through the anova/manova tests, I can definitely tell that distance is different over different colonies, mass is not that different over different colonies. (and, I can conclude that at least one colony should use different foraging strategies)

Also, I can find out that distance and mass are different over different size class. (so, it implies that foraging strategies will be different across size class)

From the boxplot, I can see the trend that as distance gets further, mass seems to gets smaller. I also showed how sizes or distances vary over different colonies, by boxplots. (which I personally think not good, as distance and sizes are kind-of categorical variables, and it is not good to be investigated through box plots)

From the jitter plots, I can kind-of get the idea that colony 1 and 4 have more large sizes than the other colonies, and colony 5 doesn't really go to distance 10.

In conclusion, by looking at different parts and conducting small Anova tests, I feel like it is kind-of hard to make conclusions.

It might be better for me to get mean and standard deviation of mass for for each distance for each colony (and also size.class).

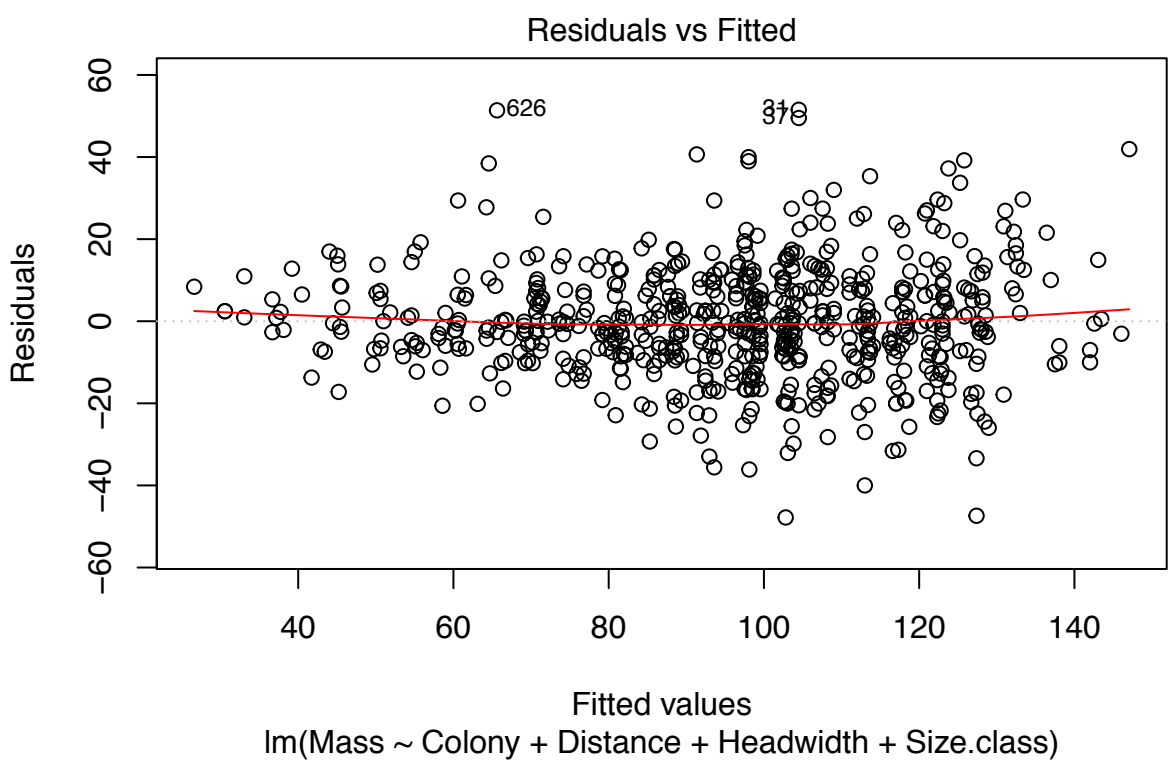
Part b

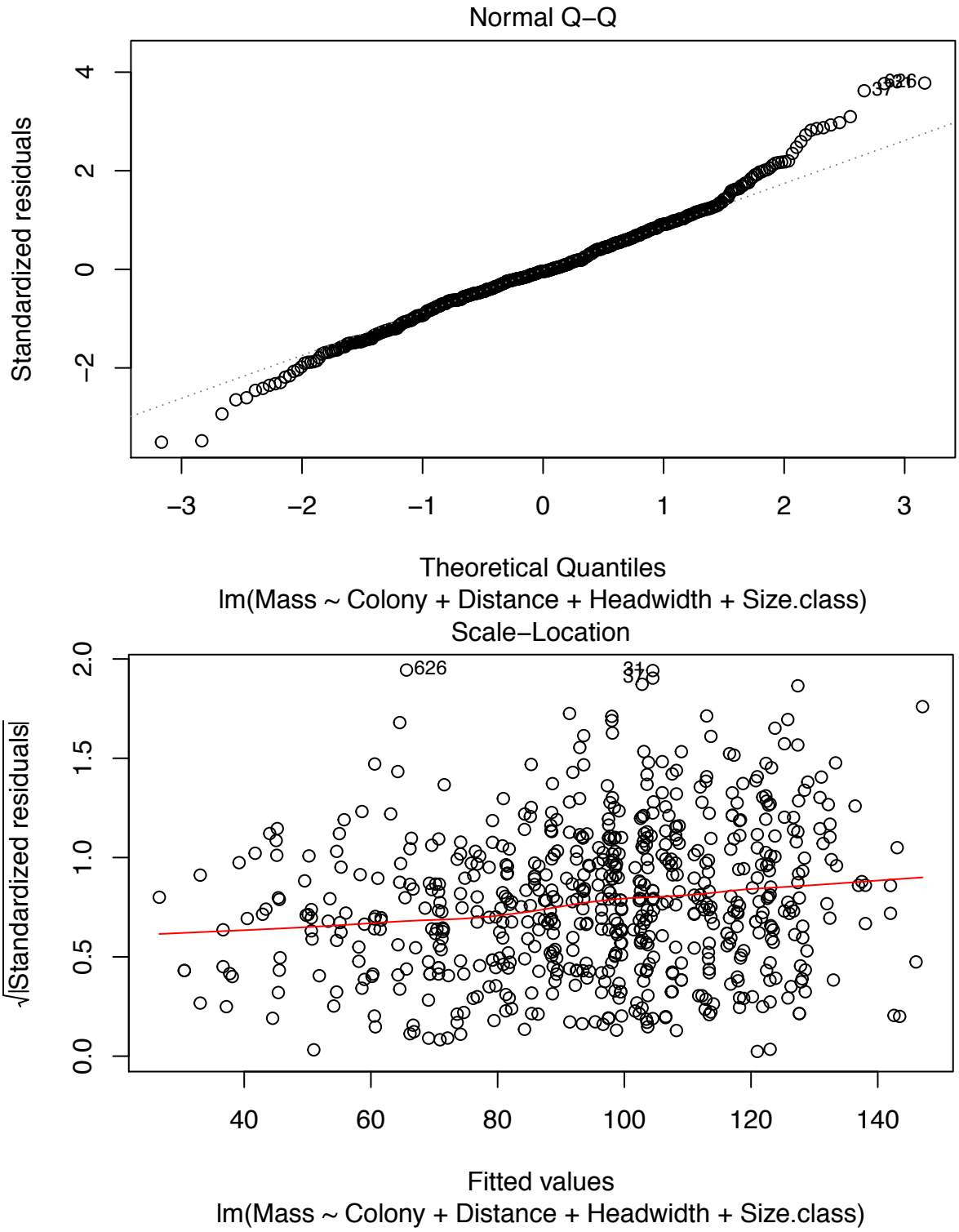
```r
lms <- lm(Mass ~ Colony + Distance + Headwidth + Size.class, data = ant)
summary(lms)
```

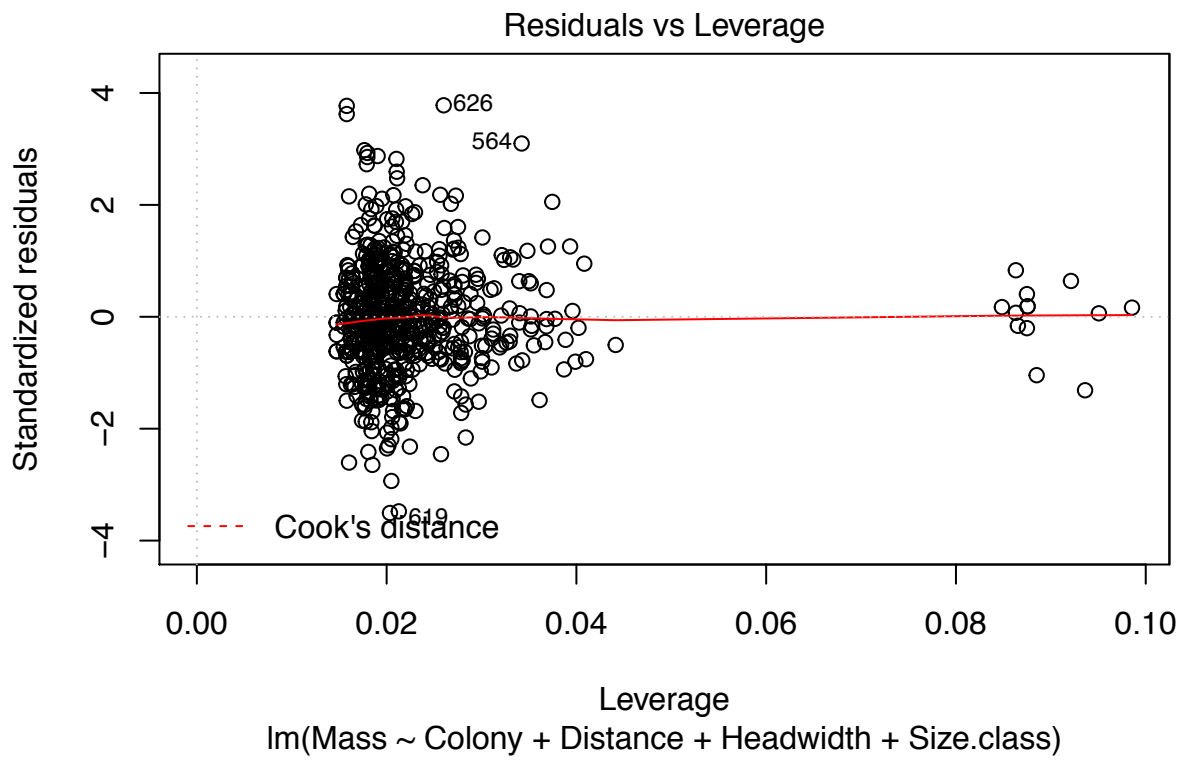
```
##
## Call:
## lm(formula = Mass ~ Colony + Distance + Headwidth + Size.class,
##     data = ant)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.804  -8.022  -0.555   8.003  51.511
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -98.8873    13.7297  -7.202 1.69e-12 ***
## Colony2       -0.6681     1.9279  -0.347 0.729058
## Colony3        4.1689     1.9311   2.159 0.031240 *
## Colony4       -0.9117     1.8497  -0.493 0.622283
## Colony5        3.6130     1.9831   1.822 0.068934 .
## Colony6       -0.3902     1.8255  -0.214 0.830829
## Distance1     -6.6332     1.9680  -3.371 0.000796 ***
## Distance4     -9.5713     1.9502  -4.908 1.17e-06 ***
## Distance7     -9.4464     2.0165  -4.685 3.44e-06 ***
## Distance10   -10.5882     2.1188  -4.997 7.54e-07 ***
## Headwidth      5.0542     0.4555  11.097  < 2e-16 ***
## Size.classLL  -0.2594     8.8522  -0.029 0.976627
## Size.classS   -2.2907     4.6612  -0.491 0.623278
## Size.classM   -6.4351     5.8475  -1.100 0.271539
## Size.classL   -4.3846     7.3629  -0.596 0.551722
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.77 on 633 degrees of freedom
## Multiple R-squared:  0.7464, Adjusted R-squared:  0.7408
## F-statistic: 133.1 on 14 and 633 DF,  p-value: < 2.2e-16
```

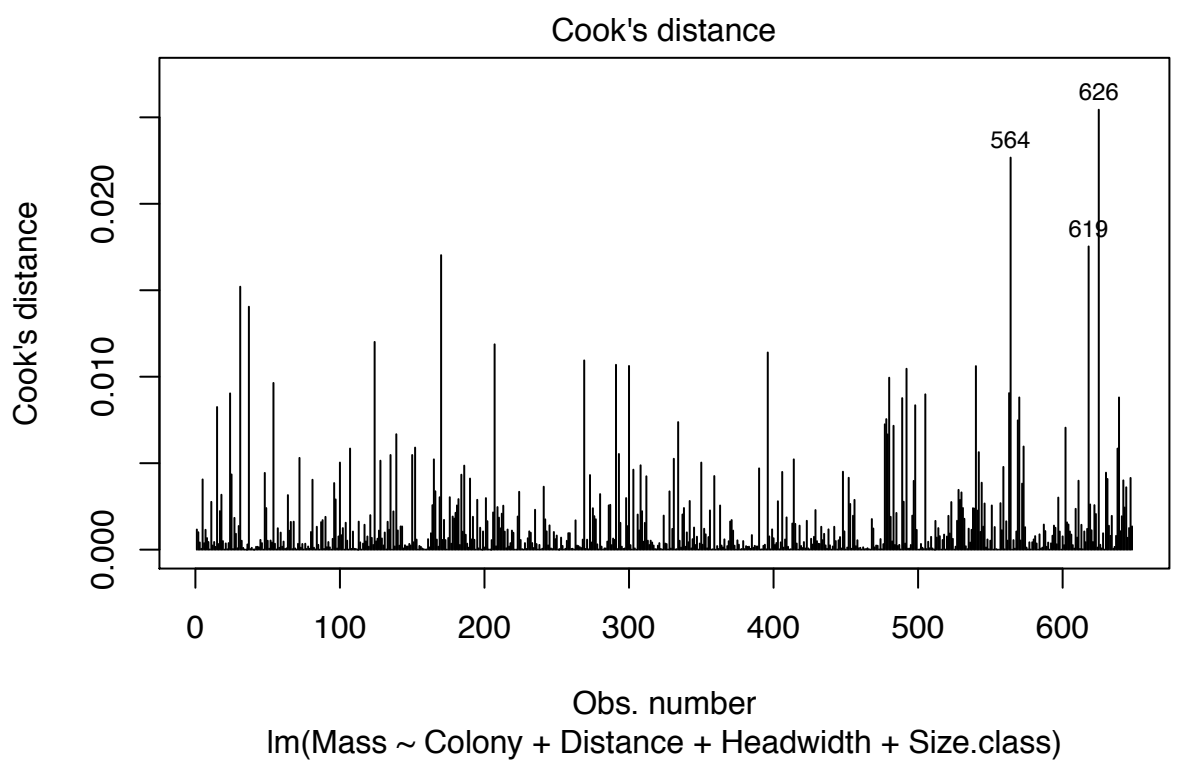
```r
plot(lms)
```



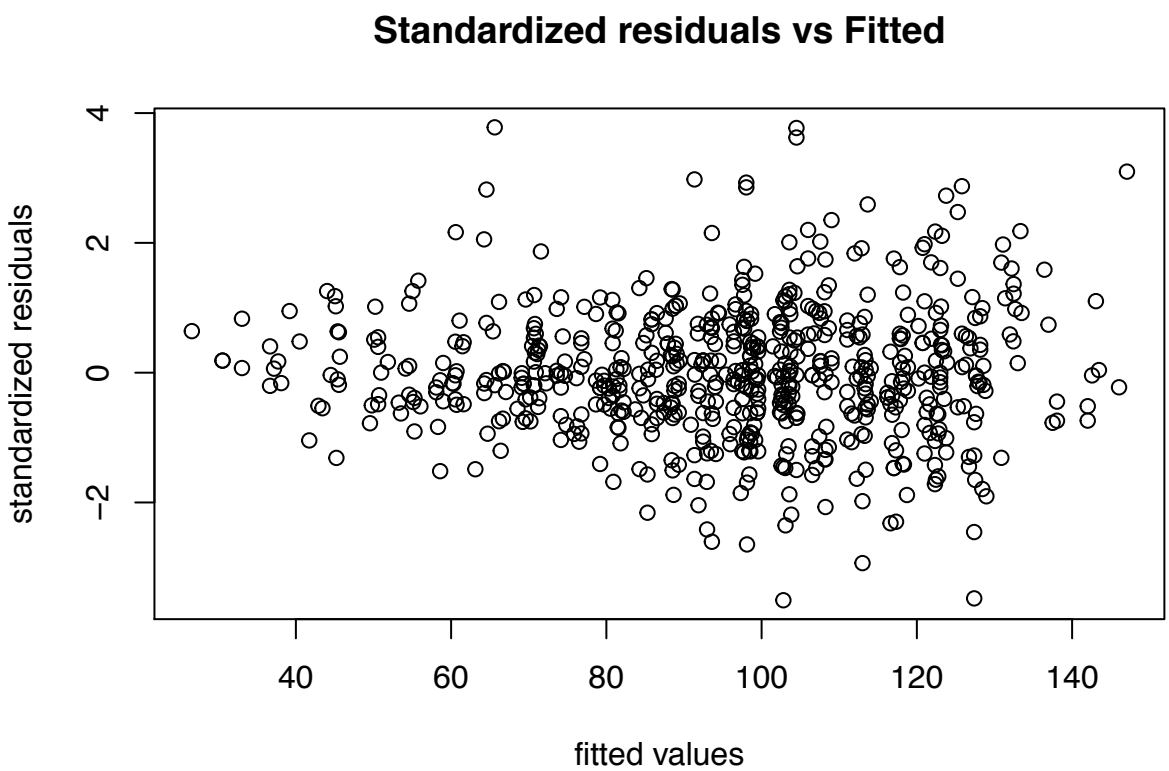

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Mass ~ Colony + Distance + Headwidth + Size.class)
Scale−Location
√|Standardized residuals|
626
Fitted values
lm(Mass ~ Colony + Distance + Headwidth + Size.class)




```r
plot(lms, which = 4)
```




```r
plot(lms$fitted.values, rstandard(lms), main = "Standardized residuals vs Fitted",
     xlab = "fitted values", ylab = "standardized residuals")
```




```
stand_resid <- as.data.frame(rstandard(lms))
stand_resid[,2] <- seq(1, nrow(stand_resid))
stand_resid %>% filter(rstandard(lms) < -3)
```

```
##   rstandard(lms)  V2
## 1      -3.506444 170
## 2      -3.477719 618
```

```
stand_resid %>% filter(rstandard(lms) > 3)
```

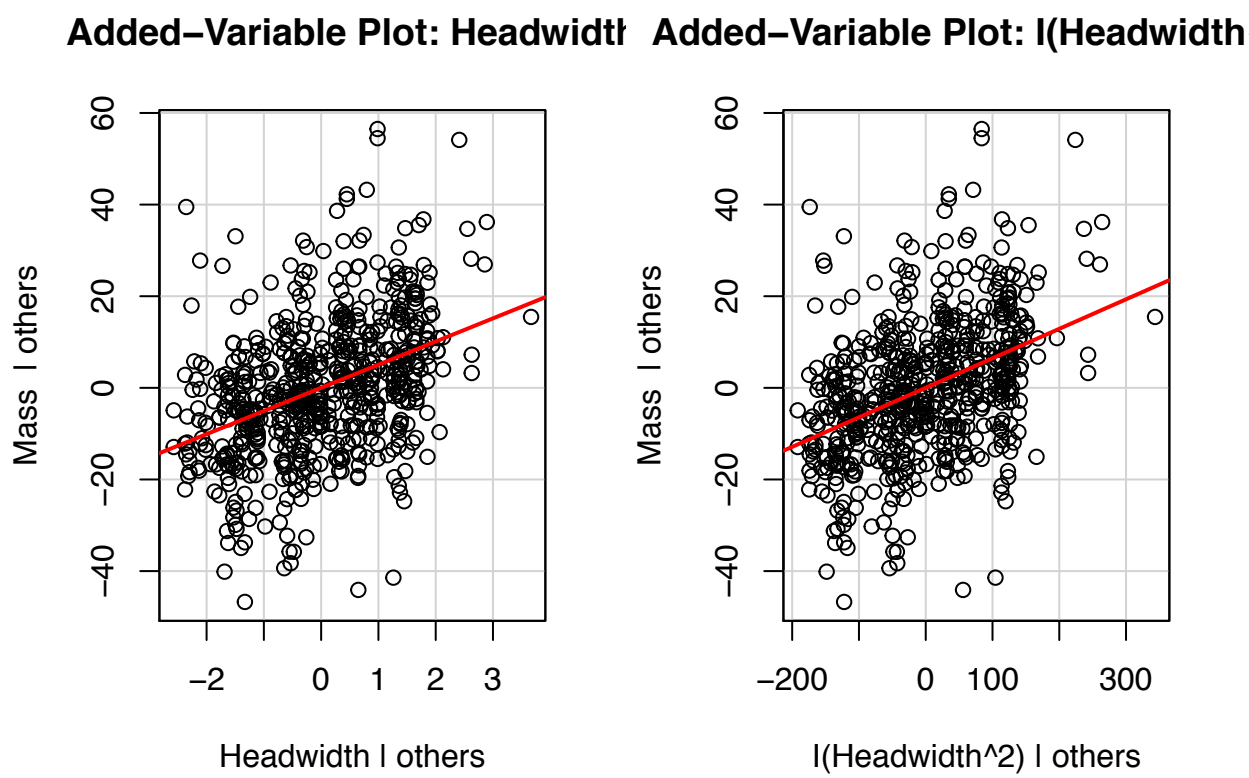
```
##   rstandard(lms)  V2
## 1       3.769612  31
## 2       3.623251  37
## 3       3.097638 564
## 4       3.779866 625
```

```
lms2 <- lm(Mass ~ Colony + Distance + I(Headwidth^2) + Size.class, data = ant)
summary(lms2)
```

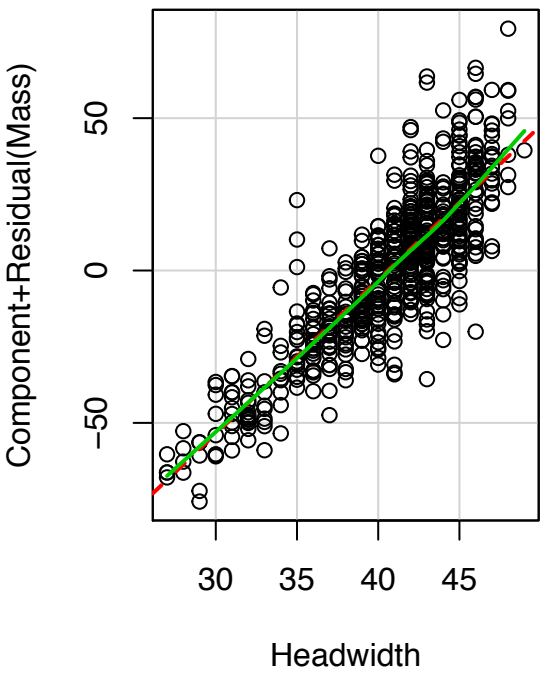
```
##
## Call:
## lm(formula = Mass ~ Colony + Distance + I(Headwidth^2) + Size.class,
##     data = ant)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.158  -7.980  -0.681   7.769  51.092
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -7.826567   6.327451  -1.237 0.216574
```

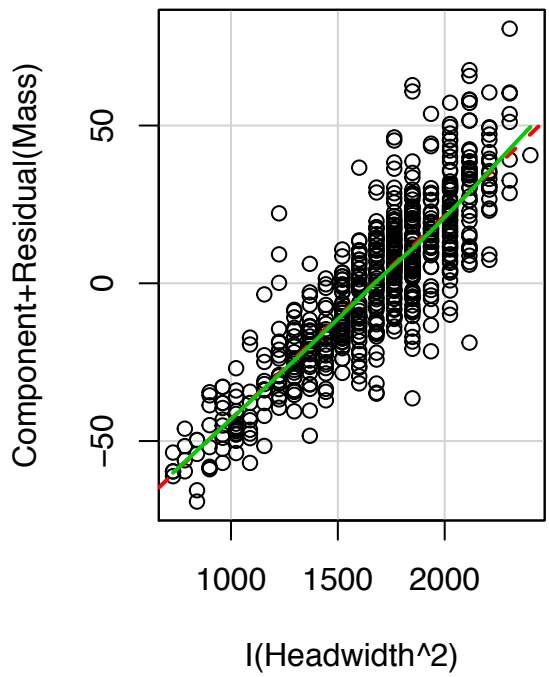
```
## Colony2          -0.746167   1.916089  -0.389 0.697095
## Colony3           4.124632   1.919691   2.149 0.032045 *
## Colony4          -0.821322   1.839640  -0.446 0.655419
## Colony5           3.515362   1.970937   1.784 0.074968 .
## Colony6          -0.443153   1.813675  -0.244 0.807047
## Distance1        -6.527255   1.957442  -3.335 0.000904 ***
## Distance4        -9.495559   1.939499  -4.896 1.24e-06 ***
## Distance7        -9.221884   2.006211  -4.597 5.18e-06 ***
## Distance10      -10.499810   2.107189  -4.983 8.10e-07 ***
## I(Headwidth^2)    0.064416   0.005615  11.473  < 2e-16 ***
## Size.classLL      5.213893   8.192362   0.636 0.524724
## Size.classS       2.319178   4.475672   0.518 0.604518
## Size.classM       1.228924   5.273339   0.233 0.815802
## Size.classL       3.124831   6.643416   0.470 0.638256
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.7 on 633 degrees of freedom
## Multiple R-squared:  0.7492, Adjusted R-squared:  0.7437
## F-statistic: 135.1 on 14 and 633 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(1,2))
avPlot(lms, "Headwidth")
avPlot(lms2, "I(Headwidth^2)")
```

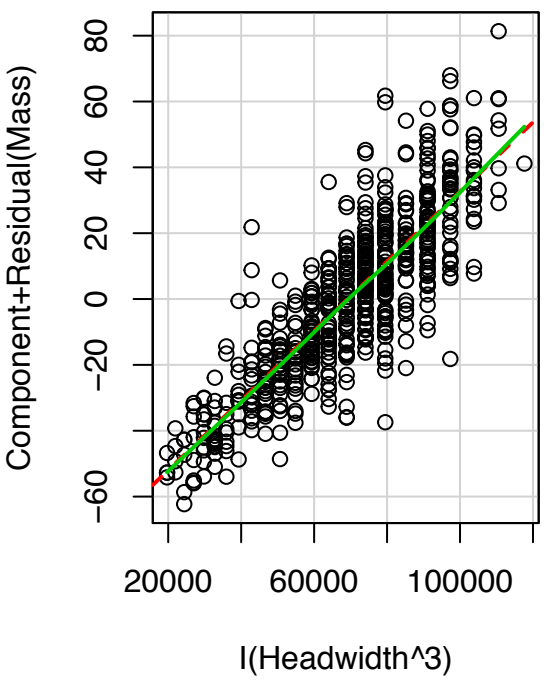


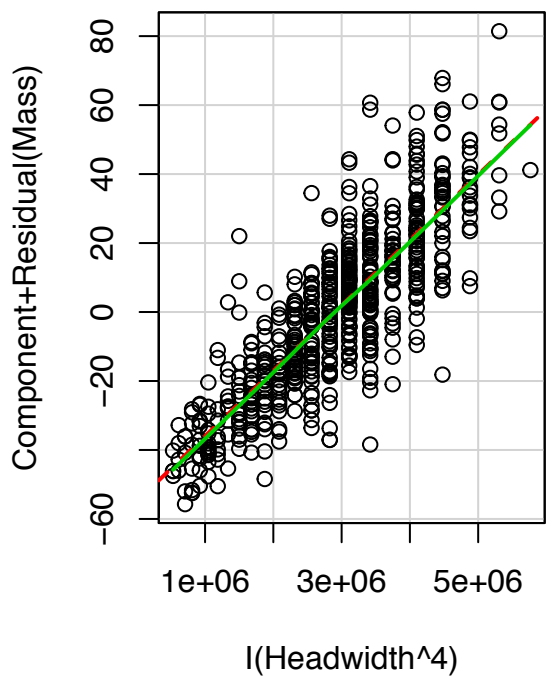

```r
crPlot(lms, "Headwidth")
crPlot(lms2, "I(Headwidth^2)")
```





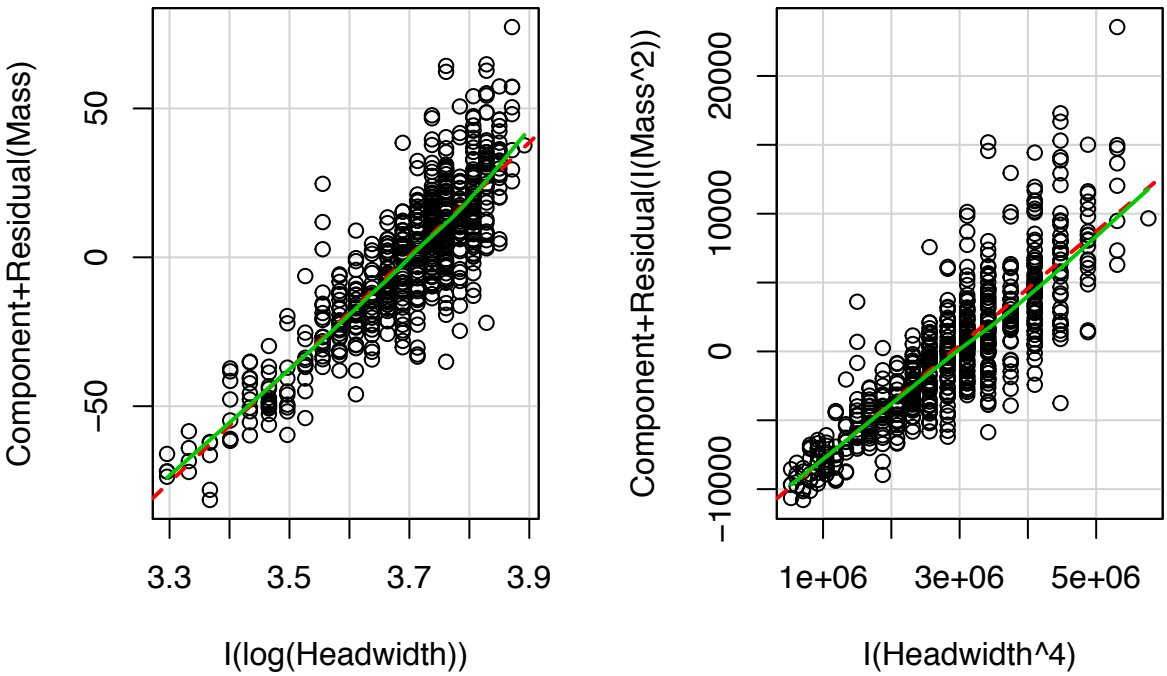

```
lms3 <- lm(Mass ~ Colony + Distance + I(Headwidth^3) + Size.class, data = ant)


lms4 <- lm(Mass ~ Colony + Distance + I(Headwidth^4) + Size.class, data = ant)

crPlot(lms3, "I(Headwidth^3)")
crPlot(lms4, "I(Headwidth^4)")
```






```
lms5 <- lm(Mass ~ Colony + Distance + I(log(Headwidth)) + Size.class, data = ant)

crPlot(lms5, "I(log(Headwidth))")
```

```r
lms5 <- lm(I(Mass^2) ~ Colony + Distance + I(Headwidth^4) + Size.class, data = ant)

crPlot(lms5, "I(Headwidth^4)")
```



Comment:

When I look at fitted values v.s. residual plot, I can find the variance of reisduals go up as the fitted values go up. It implies heteroscedasticity of errors. And, there is really no significant high leverage; however, there are some outliers (absolute magniutde of standardized residual is bigger than 3 - 31th, 37th, 170th, 564th, 618th, 625th observations) that might influence the fit.

Since it does not make sense to transform categorical variables, I would transfrom either Mass and Headwidth. And, before doing that, I used added-variable plot and component plus residual plot. There is no super significant high-leverage or influential poitns, and it seems like headwidth almost satisfies linearity (little bit of non-linearity in both tails). So, I transformed headwidth to get better linearity fit.

After squaring headwidth, I could get little bit higher R^2 and adjusted R^2, and can see better linearity in component plus residual plot. Furthermore, if I *triple power* it and *quadraple*, I could even get a better fit!!!

Taking log transformation seems not that helpful. Also, transforming response variables does not help me getting better linearity.

In conclusion, it might be better for me have fourth power of headwidth to get a better model (conclusion from component plus residual plot)

Part c

```r
print("Colony 1")
```

```
## [1] "Colony 1"
```

```r
colony1 %>% group_by(Size.class) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Size.class     n
##       <fctr> <int>
## 1         SS     3
## 2         LL    27
## 3          S     7
## 4          M    15
## 5          L    59
```

```r
colony1 %>% group_by(Distance) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Distance     n
##     <fctr> <int>
## 1        0    12
## 2        1    22
## 3        4    35
## 4        7    22
## 5       10    20
```

```r
colony1 %>% group_by(Distance, Size.class) %>% summarise(n = n())
```

```
## # A tibble: 20 x 3
## # Groups:   Distance [?]
##    Distance Size.class     n
##      <fctr>     <fctr> <int>
##  1        0         LL     2
##  2        0          S     2
##  3        0          M     2
##  4        0          L     6
##  5        1         LL     8
##  6        1          S     2
##  7        1          M     2
##  8        1          L    10
##  9        4         SS     3
## 10        4         LL     7
## 11        4          S     2
## 12        4          M     7
## 13        4          L    16
## 14        7         LL     5
## 15        7          M     1
## 16        7          L    16
```

```
## 17       10         LL     5
## 18       10          S     1
## 19       10          M     3
## 20       10          L    11
```

```r
colony1 %>% group_by(Distance, Mass) %>% summarise(n = n())
```

```
## # A tibble: 94 x 3
## # Groups:   Distance [?]
##    Distance  Mass     n
##      <fctr> <int> <int>
##  1        0    55     1
##  2        0    68     1
##  3        0   100     1
##  4        0   102     1
##  5        0   107     1
##  6        0   108     1
##  7        0   111     1
##  8        0   112     1
##  9        0   115     1
## 10        0   141     1
## # ... with 84 more rows
```

```r
by(colony1$Mass, colony1$Distance, mean)
```

```
## colony1$Distance: 0
## [1] 110.5
## ------------------------------------------------------------
## colony1$Distance: 1
## [1] 100.2273
## ------------------------------------------------------------
## colony1$Distance: 4
## [1] 92.77143
## ------------------------------------------------------------
## colony1$Distance: 7
## [1] 104
## ------------------------------------------------------------
## colony1$Distance: 10
## [1] 95.4
```

```r
by(colony1$Mass, colony1$Distance, sd)
```

```
## colony1$Distance: 0
## [1] 30.02877
## ------------------------------------------------------------
## colony1$Distance: 1
## [1] 19.27725
## ------------------------------------------------------------
## colony1$Distance: 4
## [1] 31.13262
## ------------------------------------------------------------
## colony1$Distance: 7
## [1] 14.4848
## ------------------------------------------------------------
## colony1$Distance: 10
## [1] 15.81605
```

```r
print("Colony 2")
```

```
## [1] "Colony 2"
```

```r
colony2 %>% group_by(Size.class) %>% summarise(n = n())
```

```
## # A tibble: 4 x 2
##   Size.class     n
##       <fctr> <int>
## 1         LL    36
## 2          S     8
## 3          M    21
## 4          L    35
```

```r
colony2 %>% group_by(Distance) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Distance     n
##     <fctr> <int>
## 1        0    12
## 2        1    22
## 3        4    22
## 4        7    21
## 5       10    23
```

```r
colony2 %>% group_by(Distance, Size.class) %>% summarise(n = n())
```

```
## # A tibble: 19 x 3
## # Groups:   Distance [?]
##    Distance Size.class     n
##      <fctr>     <fctr> <int>
##  1        0         LL     5
##  2        0          S     1
##  3        0          M     2
##  4        0          L     4
##  5        1         LL     7
##  6        1          S     2
##  7        1          M     7
##  8        1          L     6
##  9        4         LL    10
## 10        4          M     6
## 11        4          L     6
## 12        7         LL     7
## 13        7          S     4
## 14        7          M     4
## 15        7          L     6
## 16       10         LL     7
## 17       10          S     1
## 18       10          M     2
## 19       10          L    13
```

```r
colony2 %>% group_by(Distance, Mass) %>% summarise(n = n())
```

```
## # A tibble: 89 x 3
## # Groups:   Distance [?]
##    Distance  Mass     n
##      <fctr> <int> <int>
```

```
## 1        0    43     1
## 2        0    71     1
## 3        0    79     1
## 4        0    85     1
## 5        0    97     1
## 6        0   106     1
## 7        0   114     1
## 8        0   116     1
## 9        0   119     1
## 10       0   127     2
## # ... with 79 more rows
```

```
by(colony2$Mass, colony2$Distance, mean)
```

```
## colony2$Distance: 0
## [1] 102.5
## ------------------------------------------------------------
## colony2$Distance: 1
## [1] 96.81818
## ------------------------------------------------------------
## colony2$Distance: 4
## [1] 96.09091
## ------------------------------------------------------------
## colony2$Distance: 7
## [1] 95.19048
## ------------------------------------------------------------
## colony2$Distance: 10
## [1] 102.2174
```

```
by(colony2$Mass, colony2$Distance, sd)
```

```
## colony2$Distance: 0
## [1] 28.78289
## ------------------------------------------------------------
## colony2$Distance: 1
## [1] 31.27548
## ------------------------------------------------------------
## colony2$Distance: 4
## [1] 28.40675
## ------------------------------------------------------------
## colony2$Distance: 7
## [1] 27.94033
## ------------------------------------------------------------
## colony2$Distance: 10
## [1] 26.20194
```

```
print("Colony 3")
```

```
## [1] "Colony 3"
```

```
colony3 %>% group_by(Size.class) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Size.class     n
##       <fctr> <int>
## 1         SS     2
## 2         LL    23
```

```
## 3          S    12
## 4          M    29
## 5          L    34
```

```
colony3 %>% group_by(Distance) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Distance     n
##     <fctr> <int>
## 1        0    12
## 2        1    22
## 3        4    22
## 4        7    21
## 5       10    23
```

```
colony3 %>% group_by(Distance, Size.class) %>% summarise(n = n())
```

```
## # A tibble: 22 x 3
## # Groups:   Distance [?]
##    Distance Size.class     n
##      <fctr>     <fctr> <int>
##  1        0         SS     1
##  2        0         LL     3
##  3        0          S     1
##  4        0          M     4
##  5        0          L     3
##  6        1         SS     1
##  7        1         LL     4
##  8        1          S     2
##  9        1          M     7
## 10        1          L     8
## # ... with 12 more rows
```

```
colony3 %>% group_by(Distance, Mass) %>% summarise(n = n())
```

```
## # A tibble: 91 x 3
## # Groups:   Distance [?]
##    Distance  Mass     n
##      <fctr> <int> <int>
##  1        0    54     1
##  2        0    64     1
##  3        0    73     1
##  4        0    81     1
##  5        0    90     3
##  6        0   116     1
##  7        0   135     1
##  8        0   142     1
##  9        0   149     1
## 10        0   151     1
## # ... with 81 more rows
```

```
by(colony3$Mass, colony3$Distance, mean)
```

```
## colony3$Distance: 0
## [1] 102.9167
## ------------------------------------------------------------
## colony3$Distance: 1
```

```
## [1] 91.40909
## ------------------------------------------------------------
## colony3$Distance: 4
## [1] 97.45455
## ------------------------------------------------------------
## colony3$Distance: 7
## [1] 94.38095
## ------------------------------------------------------------
## colony3$Distance: 10
## [1] 93.69565
```

```
by(colony3$Mass, colony3$Distance, sd)
```

```
## colony3$Distance: 0
## [1] 34.30732
## ------------------------------------------------------------
## colony3$Distance: 1
## [1] 29.96293
## ------------------------------------------------------------
## colony3$Distance: 4
## [1] 24.94513
## ------------------------------------------------------------
## colony3$Distance: 7
## [1] 28.27981
## ------------------------------------------------------------
## colony3$Distance: 10
## [1] 21.58163
```

```
print("Colony 4")
```

```
## [1] "Colony 4"
```

```
colony4 %>% group_by(Size.class) %>% summarise(n = n())
```

```
## # A tibble: 4 x 2
##   Size.class     n
##       <fctr> <int>
## 1         LL    27
## 2          S     7
## 3          M    19
## 4          L    63
```

```
colony4 %>% group_by(Distance) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Distance     n
##     <fctr> <int>
## 1        0    12
## 2        1    30
## 3        4    25
## 4        7    24
## 5       10    25
```

```
colony4 %>% group_by(Distance, Size.class) %>% summarise(n = n())
```

```
## # A tibble: 19 x 3
## # Groups:   Distance [?]
##    Distance Size.class     n
```

```
##      <fctr>     <fctr> <int>
##  1        0         LL     6
##  2        0          S     2
##  3        0          M     3
##  4        0          L     1
##  5        1         LL     5
##  6        1          S     2
##  7        1          M    11
##  8        1          L    12
##  9        4         LL     5
## 10        4          S     1
## 11        4          M     3
## 12        4          L    16
## 13        7         LL     5
## 14        7          S     2
## 15        7          M     1
## 16        7          L    16
## 17       10         LL     6
## 18       10          M     1
## 19       10          L    18
```

```
colony4 %>% group_by(Distance, Mass) %>% summarise(n = n())
```

```
## # A tibble: 97 x 3
## # Groups:   Distance [?]
##    Distance  Mass     n
##      <fctr> <int> <int>
##  1        0    52     1
##  2        0    56     1
##  3        0    75     1
##  4        0    80     1
##  5        0    94     1
##  6        0    96     1
##  7        0    99     1
##  8        0   103     1
##  9        0   117     1
## 10        0   119     1
## # ... with 87 more rows
```

```
by(colony4$Mass, colony4$Distance, mean)
```

```
## colony4$Distance: 0
## [1] 96.41667
## ------------------------------------------------------------
## colony4$Distance: 1
## [1] 91.33333
## ------------------------------------------------------------
## colony4$Distance: 4
## [1] 97.48
## ------------------------------------------------------------
## colony4$Distance: 7
## [1] 97.66667
## ------------------------------------------------------------
## colony4$Distance: 10
## [1] 100.92
```

```r
by(colony4$Mass, colony4$Distance, sd)
```

```
## colony4$Distance: 0
## [1] 27.09062
## ------------------------------------------------------------
## colony4$Distance: 1
## [1] 30.11281
## ------------------------------------------------------------
## colony4$Distance: 4
## [1] 21.7622
## ------------------------------------------------------------
## colony4$Distance: 7
## [1] 22.4764
## ------------------------------------------------------------
## colony4$Distance: 10
## [1] 18.0345
```

```r
print("Colony 5")
```

```
## [1] "Colony 5"
```

```r
colony5 %>% group_by(Size.class) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Size.class     n
##       <fctr> <int>
## 1         SS     3
## 2         LL    18
## 3          S     5
## 4          M    26
## 5          L    39
```

```r
colony5 %>% group_by(Distance) %>% summarise(n = n())
```

```
## # A tibble: 4 x 2
##   Distance     n
##     <fctr> <int>
## 1        0    12
## 2        1    28
## 3        4    28
## 4        7    23
```

```r
colony5 %>% group_by(Distance, Size.class) %>% summarise(n = n())
```

```
## # A tibble: 15 x 3
## # Groups:   Distance [?]
##    Distance Size.class     n
##      <fctr>     <fctr> <int>
##  1        0         LL     4
##  2        0          S     1
##  3        0          M     3
##  4        0          L     4
##  5        1         LL     5
##  6        1          M    10
##  7        1          L    13
##  8        4         SS     3
##  9        4         LL     4
```

```
## 10        4          S     4
## 11        4          M     8
## 12        4          L     9
## 13        7         LL     5
## 14        7          M     5
## 15        7          L    13
```

```
colony5 %>% group_by(Distance, Mass) %>% summarise(n = n())
```

```
## # A tibble: 82 x 3
## # Groups:   Distance [?]
##    Distance  Mass     n
##      <fctr> <int> <int>
##  1        0    56     1
##  2        0    59     1
##  3        0    64     1
##  4        0    95     1
##  5        0   109     1
##  6        0   110     1
##  7        0   120     1
##  8        0   132     1
##  9        0   135     2
## 10        0   147     1
## # ... with 72 more rows
```

```
by(colony5$Mass, colony5$Distance, mean)
```

```
## colony5$Distance: 0
## [1] 112.5833
## ------------------------------------------------------------
## colony5$Distance: 1
## [1] 104.3214
## ------------------------------------------------------------
## colony5$Distance: 4
## [1] 84.10714
## ------------------------------------------------------------
## colony5$Distance: 7
## [1] 94.17391
## ------------------------------------------------------------
## colony5$Distance: 10
## [1] NA
```

```
by(colony5$Mass, colony5$Distance, sd)
```

```
## colony5$Distance: 0
## [1] 39.56918
## ------------------------------------------------------------
## colony5$Distance: 1
## [1] 24.77611
## ------------------------------------------------------------
## colony5$Distance: 4
## [1] 29.10306
## ------------------------------------------------------------
## colony5$Distance: 7
## [1] 23.13135
## ------------------------------------------------------------
```

```
## colony5$Distance: 10
## [1] NA
```

```r
print("Colony 6")
```

```
## [1] "Colony 6"
```

```r
colony6 %>% group_by(Size.class) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Size.class     n
##       <fctr> <int>
## 1         SS     5
## 2         LL    36
## 3          S    16
## 4          M    35
## 5          L    38
```

```r
colony6 %>% group_by(Distance) %>% summarise(n = n())
```

```
## # A tibble: 5 x 2
##   Distance     n
##     <fctr> <int>
## 1        0    12
## 2        1    35
## 3        4    35
## 4        7    27
## 5       10    21
```

```r
colony6 %>% group_by(Distance, Size.class) %>% summarise(n = n())
```

```
## # A tibble: 23 x 3
## # Groups:   Distance [?]
##    Distance Size.class     n
##      <fctr>     <fctr> <int>
##  1        0         SS     1
##  2        0         LL     2
##  3        0          S     1
##  4        0          M     3
##  5        0          L     5
##  6        1         SS     2
##  7        1         LL     9
##  8        1          S     4
##  9        1          M    13
## 10        1          L     7
## # ... with 13 more rows
```

```r
colony6 %>% group_by(Distance, Mass) %>% summarise(n = n())
```

```
## # A tibble: 112 x 3
## # Groups:   Distance [?]
##    Distance  Mass     n
##      <fctr> <int> <int>
##  1        0    38     1
##  2        0    49     1
##  3        0    65     1
##  4        0    82     1
##  5        0    91     1
```

```
## 6        0   102     1
## 7        0   106     1
## 8        0   113     1
## 9        0   120     1
## 10       0   126     1
## # ... with 102 more rows
```

```r
by(colony6$Mass, colony6$Distance, mean)
```

```
## colony6$Distance: 0
## [1] 98.33333
## ------------------------------------------------------
## colony6$Distance: 1
## [1] 86.11429
## ------------------------------------------------------
## colony6$Distance: 4
## [1] 81.97143
## ------------------------------------------------------
## colony6$Distance: 7
## [1] 97.33333
## ------------------------------------------------------
## colony6$Distance: 10
## [1] 93.57143
```

```r
by(colony6$Mass, colony6$Distance, sd)
```

```
## colony6$Distance: 0
## [1] 35.1499
## ------------------------------------------------------
## colony6$Distance: 1
## [1] 31.68018
## ------------------------------------------------------
## colony6$Distance: 4
## [1] 28.17434
## ------------------------------------------------------
## colony6$Distance: 7
## [1] 22.89609
## ------------------------------------------------------
## colony6$Distance: 10
## [1] 27.25174
```

```r
print("Size SS")
```

```
## [1] "Size SS"
```

```r
sizeSS %>% group_by(Colony) %>% summarise(n = n())
```

```
## # A tibble: 4 x 2
##   Colony     n
##   <fctr> <int>
## 1      1     3
## 2      3     2
## 3      5     3
## 4      6     5
```

```r
by(sizeSS$Mass, sizeSS$Distance, mean)
```

```
## sizeSS$Distance: 0
```

```
## [1] 46
## ------------------------------------------------------------
## sizeSS$Distance: 1
## [1] 31.33333
## ------------------------------------------------------------
## sizeSS$Distance: 4
## [1] 36.85714
## ------------------------------------------------------------
## sizeSS$Distance: 7
## [1] NA
## ------------------------------------------------------------
## sizeSS$Distance: 10
## [1] 35
```

```r
by(sizeSS$Mass, sizeSS$Distance, sd)
```

```
## sizeSS$Distance: 0
## [1] 11.31371
## ------------------------------------------------------------
## sizeSS$Distance: 1
## [1] 2.886751
## ------------------------------------------------------------
## sizeSS$Distance: 4
## [1] 5.520524
## ------------------------------------------------------------
## sizeSS$Distance: 7
## [1] NA
## ------------------------------------------------------------
## sizeSS$Distance: 10
## [1] NA
```

```r
print("Size S")
```

```
## [1] "Size S"
```

```r
sizeS %>% group_by(Colony) %>% summarise(n = n())
```

```
## # A tibble: 6 x 2
##   Colony     n
##   <fctr> <int>
## 1      1     7
## 2      2     8
## 3      3    12
## 4      4     7
## 5      5     5
## 6      6    16
```

```r
by(sizeS$Mass, sizeS$Distance, mean)
```

```
## sizeS$Distance: 0
## [1] 55.75
## ------------------------------------------------------------
## sizeS$Distance: 1
## [1] 47.66667
## ------------------------------------------------------------
## sizeS$Distance: 4
## [1] 52.64706
```

```
## ------------------------------------------------------------
## sizeS$Distance: 7
## [1] 57.15385
## ------------------------------------------------------------
## sizeS$Distance: 10
## [1] 66.4
```

```r
by(sizeS$Mass, sizeS$Distance, sd)
```

```
## sizeS$Distance: 0
## [1] 8.031189
## ------------------------------------------------------------
## sizeS$Distance: 1
## [1] 8.435136
## ------------------------------------------------------------
## sizeS$Distance: 4
## [1] 7.833751
## ------------------------------------------------------------
## sizeS$Distance: 7
## [1] 10.73814
## ------------------------------------------------------------
## sizeS$Distance: 10
## [1] 16.22652
```

```r
print("Size M")
```

```
## [1] "Size M"
```

```r
sizeM %>% group_by(Colony) %>% summarise(n = n())
```

```
## # A tibble: 6 x 2
##   Colony     n
##   <fctr> <int>
## 1      1    15
## 2      2    21
## 3      3    29
## 4      4    19
## 5      5    26
## 6      6    35
```

```r
by(sizeM$Mass, sizeM$Distance, mean)
```

```
## sizeM$Distance: 0
## [1] 81.76471
## ------------------------------------------------------------
## sizeM$Distance: 1
## [1] 76.48
## ------------------------------------------------------------
## sizeM$Distance: 4
## [1] 76
## ------------------------------------------------------------
## sizeM$Distance: 7
## [1] 76.71429
## ------------------------------------------------------------
## sizeM$Distance: 10
## [1] 77.75
```

```r
by(sizeM$Mass, sizeM$Distance, sd)
```

```
## sizeM$Distance: 0
## [1] 13.34883
## ------------------------------------------------------------
## sizeM$Distance: 1
## [1] 11.60795
## ------------------------------------------------------------
## sizeM$Distance: 4
## [1] 13.62901
## ------------------------------------------------------------
## sizeM$Distance: 7
## [1] 10.31088
## ------------------------------------------------------------
## sizeM$Distance: 10
## [1] 8.955445
```

```r
print("Size L")
```

```
## [1] "Size L"
```

```r
sizeL %>% group_by(Colony) %>% summarise(n = n())
```

```
## # A tibble: 6 x 2
##   Colony     n
##   <fctr> <int>
## 1      1    59
## 2      2    35
## 3      3    34
## 4      4    63
## 5      5    39
## 6      6    38
```

```r
by(sizeL$Mass, sizeL$Distance, mean)
```

```
## sizeL$Distance: 0
## [1] 113.1739
## ------------------------------------------------------------
## sizeL$Distance: 1
## [1] 101.9107
## ------------------------------------------------------------
## sizeL$Distance: 4
## [1] 99.12903
## ------------------------------------------------------------
## sizeL$Distance: 7
## [1] 97.63077
## ------------------------------------------------------------
## sizeL$Distance: 10
## [1] 96.27419
```

```r
by(sizeL$Mass, sizeL$Distance, sd)
```

```
## sizeL$Distance: 0
## [1] 13.61033
## ------------------------------------------------------------
## sizeL$Distance: 1
## [1] 15.32826
```

```
## ------------------------------------------------------------
## sizeL$Distance: 4
## [1] 17.52608
## ------------------------------------------------------------
## sizeL$Distance: 7
## [1] 14.45961
## ------------------------------------------------------------
## sizeL$Distance: 10
## [1] 15.59862
```

```r
print("Size LL")
```

```
## [1] "Size LL"
```

```r
sizeLL %>% group_by(Colony) %>% summarise(n = n())
```

```
## # A tibble: 6 x 2
##   Colony     n
##   <fctr> <int>
## 1      1    27
## 2      2    36
## 3      3    23
## 4      4    27
## 5      5    18
## 6      6    36
```

```r
by(sizeLL$Mass, sizeLL$Distance, mean)
```

```
## sizeLL$Distance: 0
## [1] 134
## ------------------------------------------------------------
## sizeLL$Distance: 1
## [1] 126.9474
## ------------------------------------------------------------
## sizeLL$Distance: 4
## [1] 118.775
## ------------------------------------------------------------
## sizeLL$Distance: 7
## [1] 120.6923
## ------------------------------------------------------------
## sizeLL$Distance: 10
## [1] 118.6429
```

```r
by(sizeLL$Mass, sizeLL$Distance, sd)
```

```
## sizeLL$Distance: 0
## [1] 22.22611
## ------------------------------------------------------------
## sizeLL$Distance: 1
## [1] 18.57919
## ------------------------------------------------------------
## sizeLL$Distance: 4
## [1] 18.93509
## ------------------------------------------------------------
## sizeLL$Distance: 7
## [1] 13.01043
## ------------------------------------------------------------
```

```
## sizeLL$Distance: 10
## [1] 18.87623
```

```r
lms
```

```
##
## Call:
## lm(formula = Mass ~ Colony + Distance + Headwidth + Size.class,
##     data = ant)
##
## Coefficients:
##  (Intercept)       Colony2       Colony3       Colony4       Colony5
##     -98.8873       -0.6681        4.1689       -0.9117        3.6130
##      Colony6     Distance1     Distance4     Distance7    Distance10
##      -0.3902       -6.6332       -9.5713       -9.4464      -10.5882
##    Headwidth  Size.classLL   Size.classS   Size.classM   Size.classL
##       5.0542       -0.2594       -2.2907       -6.4351       -4.3846
```

```r
summary(lms)
```

```
##
## Call:
## lm(formula = Mass ~ Colony + Distance + Headwidth + Size.class,
##     data = ant)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.804  -8.022  -0.555   8.003  51.511
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -98.8873    13.7297  -7.202 1.69e-12 ***
## Colony2       -0.6681     1.9279  -0.347 0.729058
## Colony3        4.1689     1.9311   2.159 0.031240 *
## Colony4       -0.9117     1.8497  -0.493 0.622283
## Colony5        3.6130     1.9831   1.822 0.068934 .
## Colony6       -0.3902     1.8255  -0.214 0.830829
## Distance1     -6.6332     1.9680  -3.371 0.000796 ***
## Distance4     -9.5713     1.9502  -4.908 1.17e-06 ***
## Distance7     -9.4464     2.0165  -4.685 3.44e-06 ***
## Distance10   -10.5882     2.1188  -4.997 7.54e-07 ***
## Headwidth      5.0542     0.4555  11.097  < 2e-16 ***
## Size.classLL  -0.2594     8.8522  -0.029 0.976627
## Size.classS   -2.2907     4.6612  -0.491 0.623278
## Size.classM   -6.4351     5.8475  -1.100 0.271539
## Size.classL   -4.3846     7.3629  -0.596 0.551722
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.77 on 633 degrees of freedom
## Multiple R-squared:  0.7464, Adjusted R-squared:  0.7408
## F-statistic: 133.1 on 14 and 633 DF,  p-value: < 2.2e-16
```

```r
lms2 <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Size.class, data = colony1)
lms3 <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
```

```r
             Size.class, data = colony2)
lms4 <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Size.class, data = colony3)
lms5 <- lm(Mass ~ Distance + + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Size.class, data = colony4)
lms6 <- lm(Mass ~ Distance + + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Size.class, data = colony5)
lms7 <- lm(Mass ~ Distance + + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Size.class, data = colony6)

summary(lms2)
```

```
##
## Call:
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Size.class, data = colony1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.360  -8.435  -1.932   5.609  50.565
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.739e+03  9.263e+03   0.512 0.610039
## Distance1      -1.619e+01  5.329e+00  -3.039 0.003045 **
## Distance4      -1.446e+01  5.010e+00  -2.886 0.004793 **
## Distance7      -1.411e+01  5.369e+00  -2.628 0.009966 **
## Distance10     -1.935e+01  5.341e+00  -3.624 0.000463 ***
## Headwidth      -4.929e+02  9.744e+02  -0.506 0.614067
## I(Headwidth^2)  1.887e+01  3.801e+01   0.496 0.620720
## I(Headwidth^3) -3.118e-01  6.528e-01  -0.478 0.633998
## I(Headwidth^4)  1.897e-03  4.167e-03   0.455 0.649892
## Size.classLL   -4.784e+00  5.684e+01  -0.084 0.933101
## Size.classS     3.621e+00  4.241e+01   0.085 0.932129
## Size.classM    -7.819e+00  5.505e+01  -0.142 0.887347
## Size.classL    -1.092e+01  5.677e+01  -0.192 0.847820
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.18 on 98 degrees of freedom
## Multiple R-squared:  0.6904, Adjusted R-squared:  0.6524
## F-statistic: 18.21 on 12 and 98 DF,  p-value: < 2.2e-16
```

```r
summary(lms3)
```

```
##
## Call:
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Size.class, data = colony2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.349  -6.349  -1.654   8.707  27.085
##
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.315e+04  7.960e+03   1.652  0.10212
## Distance1      -2.059e+00  4.849e+00  -0.425  0.67215
## Distance4      -1.522e+01  4.974e+00  -3.060  0.00293 **
## Distance7      -3.516e+00  4.949e+00  -0.711  0.47925
## Distance10     -6.811e+00  4.859e+00  -1.402  0.16450
## Headwidth      -1.428e+03  8.373e+02  -1.706  0.09154 .
## I(Headwidth^2)  5.782e+01  3.275e+01   1.766  0.08092 .
## I(Headwidth^3) -1.030e+00  5.644e-01  -1.825  0.07139 .
## I(Headwidth^4)  6.835e-03  3.617e-03   1.889  0.06214 .
## Size.classS    -1.236e+01  1.645e+01  -0.752  0.45424
## Size.classM    -1.219e+01  1.169e+01  -1.043  0.29985
## Size.classL    -4.545e+00  6.935e+00  -0.655  0.51392
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.15 on 88 degrees of freedom
## Multiple R-squared:  0.8055, Adjusted R-squared:  0.7812
## F-statistic: 33.13 on 11 and 88 DF,  p-value: < 2.2e-16
```

```r
summary(lms4)
```

```
##
## Call:
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Size.class, data = colony3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.7338  -6.9398  -0.5055   6.0714  28.5470
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -4.170e+03  5.531e+03  -0.754  0.45294
## Distance1      -1.229e+01  3.954e+00  -3.107  0.00255 **
## Distance4      -1.174e+01  4.099e+00  -2.864  0.00525 **
## Distance7      -1.157e+01  4.063e+00  -2.846  0.00552 **
## Distance10     -1.216e+01  4.018e+00  -3.026  0.00326 **
## Headwidth       4.634e+02  5.798e+02   0.799  0.42632
## I(Headwidth^2) -1.896e+01  2.256e+01  -0.840  0.40293
## I(Headwidth^3)  3.413e-01  3.865e-01   0.883  0.37962
## I(Headwidth^4) -2.262e-03  2.460e-03  -0.920  0.36024
## Size.classLL    2.522e+01  2.091e+01   1.206  0.23110
## Size.classS     1.673e+01  1.460e+01   1.146  0.25504
## Size.classM     1.874e+01  1.807e+01   1.037  0.30263
## Size.classL     2.273e+01  1.903e+01   1.195  0.23553
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.88 on 87 degrees of freedom
## Multiple R-squared:  0.8577, Adjusted R-squared:  0.8381
## F-statistic: 43.69 on 12 and 87 DF,  p-value: < 2.2e-16
```

```r
summary(lms5)
```

```
##
## Call:
## lm(formula = Mass ~ Distance + +Headwidth + I(Headwidth^2) +
##     I(Headwidth^3) + I(Headwidth^4) + Size.class, data = colony4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.678  -8.940  -0.100   7.268  42.867
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -2.453e+02  1.123e+04  -0.022    0.983
## Distance1       2.333e+00  4.931e+00   0.473    0.637
## Distance4       1.166e+00  5.200e+00   0.224    0.823
## Distance7       3.262e+00  5.238e+00   0.623    0.535
## Distance10     -3.695e+00  5.304e+00  -0.697    0.488
## Headwidth       3.089e+01  1.179e+03   0.026    0.979
## I(Headwidth^2) -1.185e+00  4.599e+01  -0.026    0.979
## I(Headwidth^3)  1.876e-02  7.901e-01   0.024    0.981
## I(Headwidth^4) -8.139e-05  5.051e-03  -0.016    0.987
## Size.classS    -1.054e+01  1.951e+01  -0.540    0.590
## Size.classM    -1.597e+00  1.083e+01  -0.147    0.883
## Size.classL    -1.015e-01  6.958e+00  -0.015    0.988
##
## Residual standard error: 13.93 on 104 degrees of freedom
## Multiple R-squared:  0.6965, Adjusted R-squared:  0.6645
## F-statistic:  21.7 on 11 and 104 DF,  p-value: < 2.2e-16
```

```r
summary(lms6)
```

```
##
## Call:
## lm(formula = Mass ~ Distance + +Headwidth + I(Headwidth^2) +
##     I(Headwidth^3) + I(Headwidth^4) + Size.class, data = colony5)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.466 -10.484  -0.746  10.003  46.332
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.791e+03  8.822e+03   0.430   0.6686
## Distance1      -1.338e+00  6.633e+00  -0.202   0.8407
## Distance4      -7.615e+00  6.645e+00  -1.146   0.2553
## Distance7      -1.652e+01  6.606e+00  -2.501   0.0145 *
## Headwidth      -4.049e+02  9.395e+02  -0.431   0.6676
## I(Headwidth^2)  1.624e+01  3.717e+01   0.437   0.6634
## I(Headwidth^3) -2.880e-01  6.483e-01  -0.444   0.6581
## I(Headwidth^4)  1.923e-03  4.207e-03   0.457   0.6489
## Size.classLL    3.796e+01  4.509e+01   0.842   0.4024
## Size.classS     1.838e+01  3.100e+01   0.593   0.5548
## Size.classM     3.100e+01  4.234e+01   0.732   0.4663
```

```
## Size.classL     3.993e+01  4.355e+01   0.917   0.3620
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.52 on 79 degrees of freedom
## Multiple R-squared:  0.7224, Adjusted R-squared:  0.6837
## F-statistic: 18.69 on 11 and 79 DF,  p-value: < 2.2e-16
```

```r
summary(lms7)
```

```
##
## Call:
## lm(formula = Mass ~ Distance + +Headwidth + I(Headwidth^2) +
##     I(Headwidth^3) + I(Headwidth^4) + Size.class, data = colony6)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.660  -6.196   0.562   5.933  38.603
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -6.563e+02  3.928e+03  -0.167   0.8676
## Distance1      -8.583e+00  4.361e+00  -1.968   0.0514 .
## Distance4      -1.062e+01  4.442e+00  -2.391   0.0184 *
## Distance7      -9.339e+00  4.544e+00  -2.055   0.0421 *
## Distance10     -9.299e+00  4.707e+00  -1.976   0.0506 .
## Headwidth       4.754e+01  4.244e+02   0.112   0.9110
## I(Headwidth^2) -7.620e-01  1.699e+01  -0.045   0.9643
## I(Headwidth^3) -6.850e-03  2.993e-01  -0.023   0.9818
## I(Headwidth^4)  1.953e-04  1.956e-03   0.100   0.9206
## Size.classLL    7.869e+00  2.355e+01   0.334   0.7388
## Size.classS    -4.648e+00  1.802e+01  -0.258   0.7969
## Size.classM     4.886e+00  2.140e+01   0.228   0.8198
## Size.classL     1.219e+01  2.233e+01   0.546   0.5862
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.78 on 117 degrees of freedom
## Multiple R-squared:  0.824,  Adjusted R-squared:  0.8059
## F-statistic: 45.63 on 12 and 117 DF,  p-value: < 2.2e-16
```

```r
lmSS <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Colony, data = sizeSS)
lmS <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Colony, data = sizeS)
lmM <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Colony, data = sizeM)
lmL <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Colony, data = sizeL)
lmLL <- lm(Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) + I(Headwidth^4) +
             Colony, data = sizeLL)

lmSS
```

```
##
## Call:
```

```
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Colony, data = sizeSS)
##
## Coefficients:
##    (Intercept)       Distance1       Distance4      Distance10
##      -9236.833         -14.000         -14.333          -9.000
##      Headwidth  I(Headwidth^2)  I(Headwidth^3)  I(Headwidth^4)
##        658.500         -11.667              NA              NA
##        Colony3         Colony5         Colony6
##             NA          -3.333           6.333
```

```
lmS
```

```
##
## Call:
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Colony, data = sizeS)
##
## Coefficients:
##    (Intercept)       Distance1       Distance4       Distance7
##     -1.029e+06      -7.057e+00      -1.015e+00       1.243e+00
##     Distance10       Headwidth  I(Headwidth^2)  I(Headwidth^3)
##      9.614e+00       1.288e+05      -6.040e+03       1.258e+02
## I(Headwidth^4)         Colony2         Colony3         Colony4
##     -9.814e-01      -4.571e+00       3.670e+00      -5.413e+00
##        Colony5         Colony6
##     -1.140e+00      -3.056e+00
```

```
lmM
```

```
##
## Call:
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Colony, data = sizeM)
##
## Coefficients:
##    (Intercept)       Distance1       Distance4       Distance7
##      1.206e+06      -3.281e+00      -5.573e+00      -5.808e+00
##     Distance10       Headwidth  I(Headwidth^2)  I(Headwidth^3)
##     -4.502e+00      -1.306e+05       5.301e+03      -9.562e+01
## I(Headwidth^4)         Colony2         Colony3         Colony4
##      6.466e-01      -7.378e+00      -3.105e+00      -5.482e+00
##        Colony5         Colony6
##     -3.106e-01      -4.451e+00
```

```
lmL
```

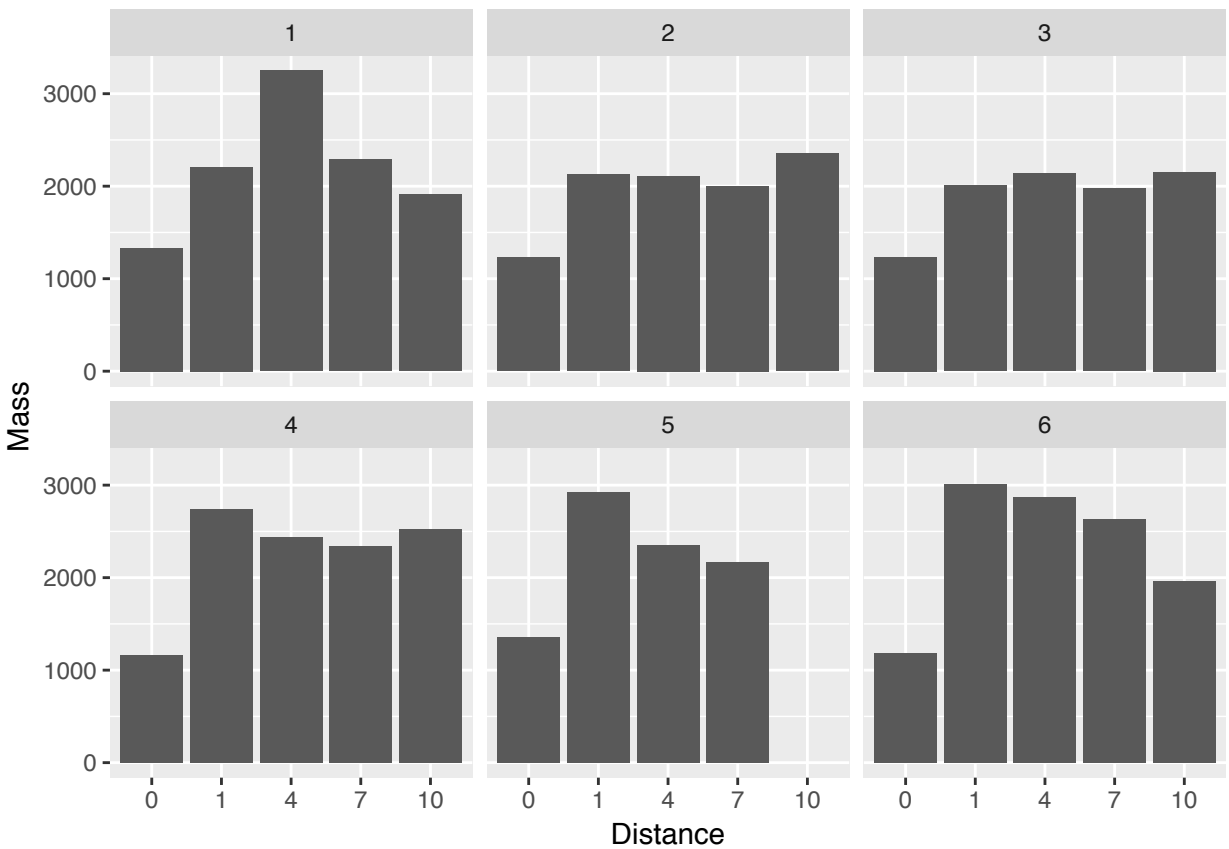
```
##
## Call:
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Colony, data = sizeL)
##
## Coefficients:
##    (Intercept)       Distance1       Distance4       Distance7
##      1.405e+05      -1.094e+01      -1.345e+01      -1.423e+01
##     Distance10       Headwidth  I(Headwidth^2)  I(Headwidth^3)
##     -1.627e+01      -1.019e+04       2.464e+02      -1.984e+00
```

```
## I(Headwidth^4)         Colony2         Colony3         Colony4
##             NA      -2.580e+00       4.047e+00      -4.186e-01
##        Colony5         Colony6
##      3.768e+00       1.103e-01
```

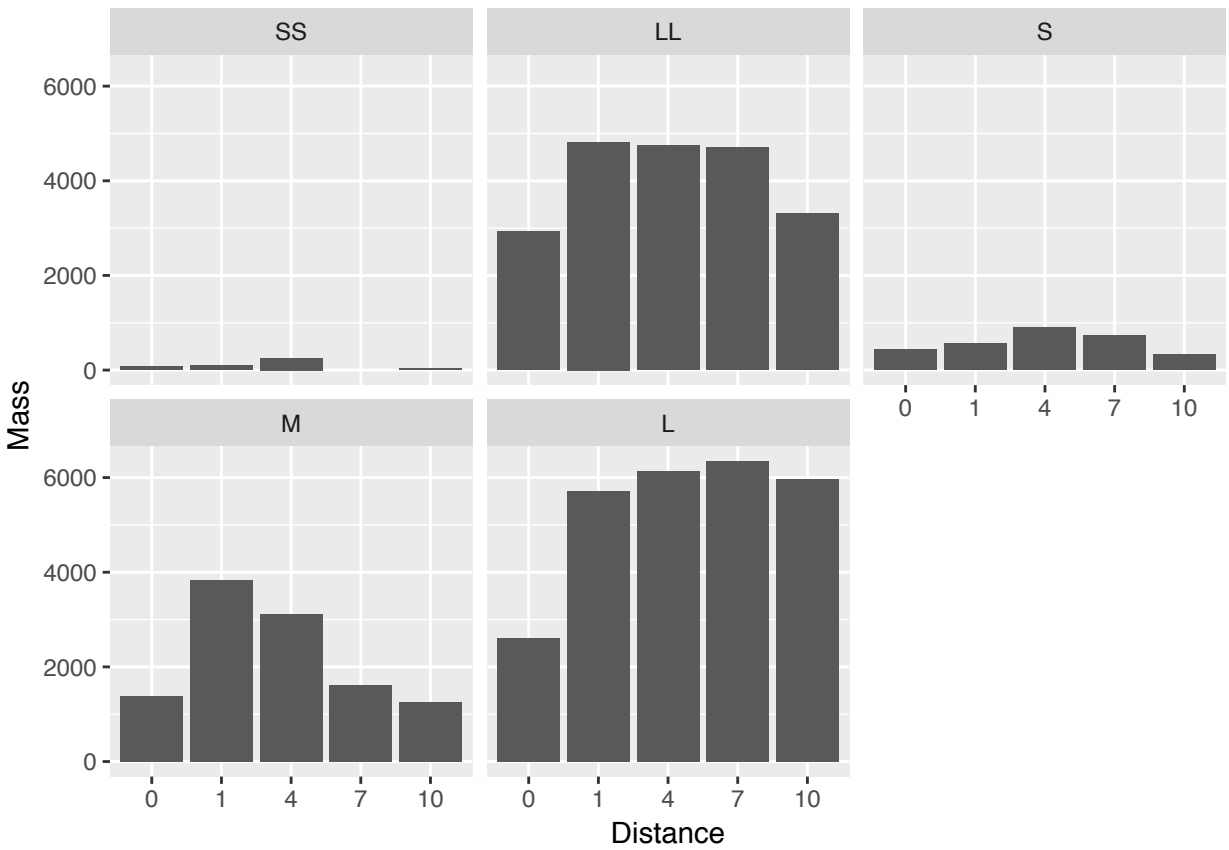
```
lmLL
```

```
##
## Call:
## lm(formula = Mass ~ Distance + Headwidth + I(Headwidth^2) + I(Headwidth^3) +
##     I(Headwidth^4) + Colony, data = sizeLL)
##
## Coefficients:
##    (Intercept)       Distance1       Distance4       Distance7
##     -1.485e+06      -9.655e-01      -1.128e+01      -6.896e+00
##     Distance10       Headwidth  I(Headwidth^2)  I(Headwidth^3)
##     -9.861e+00       1.296e+05      -4.241e+03       6.165e+01
## I(Headwidth^4)         Colony2         Colony3         Colony4
##     -3.359e-01       5.784e+00       9.539e+00       6.482e+00
##        Colony5         Colony6
##      6.281e+00       1.785e+00
```

```
ggplot(ant, aes(x = Distance, y = Mass)) + geom_bar(stat = "identity") + facet_wrap(~Colony)
```




```
ggplot(ant, aes(x = Distance, y = Mass)) + geom_bar(stat = "identity") + facet_wrap(~Size.class)
```



Comment:

I found it makes more sense to quadraple headwidth, so from now on, I would include upto quadrapled heawidth when I regress on headwidth.

When I just look at the p-values of the coefficients, most of them are not significant (meaning I can drop most of them). For example, when I look at the sizes, they all have high p-values. This is not the only suspicious thing... When I look at the coefficients of all the sizes, they have negative values. They are telling me that when the number of ant's sizes goes up, mass is going down, which does not make sense logically. So, this is telling me that I might need to make some transformations! —> Go back to b).

I printed out all the mean and standard deviation of mass for for each distance for each colony (and also size.class). So, by looking at the dplyr tables, the colony 1 has a lot of L and LL sizes, colony 2 has no SS size, and colony 4 has a lot of L sizes. Size has to do with headwidth, and when ants show the larger headwidth and sizes, it implies that there might be more worker conservative strategy.

Before I go explain more detail of what I got from lm() functions, here is what I got from these (cumulative mass) bar plots (it should be in part a), but I just included in here for interpretation purpose) So, here from the bar plots and dplyr tables, I can intuitively get some idea that the colony 1 is using mixed strategy of energy and worker conservative strategy, colony 2, 3, 4 are using worker conservative strategy, and colony 5 and 6 are using mixed strategy but closed to worker conservative strategy.

From the summary of these bunch of lm() functions, I found that I cannot fully count on the coefficients, as the p-value is pretty high for most of them and it might not be significant (or have multi-collinearity issues). However, I can kind-of see that the colony 1 and 4 are kind-of using energy conservative, and the others are using more worker conservative strategies.

Conclustion:

Here is the summary of what I got from the above.

So, I concluded that the colony 1 and 4 are closed to energy conservative, and the others are closed to worker conservative strategies.

And, I concluded that the colony SS size is closed to worker conservative, S size is using worker conservative, M size is kind-of mixed strategy (closer to energy conservative), L size is using energy conserviative, LL is closerto energy conservative strategy.

However, I cannot 100% say that the colonies are just using only one strategy. By looking at the graphs and test-statistics, I can definitely see that most of them are not monotone, meaning that most of them are using mixed strategies. (there might be other variables not in this data significant. Or, the strategies proposed here might not be correct)

I found that the colony 2 and 3 are similar, and colony 5 and 6 are simliar. (in the sense that distance how each colony's distance differs. Also, the sum of the mass for each distance are quite similar. Again, our scientific questions are focusing on foraging strategies, and I am more closely investigating mass and distance relationship. So, when I say some colonies are similar, I will take double attention onto the mass and distance, especially.)

And, as i found in part a) graph and part c) dplyr tables, colonies ahve different sizes or distances of the ants.

Problem 3

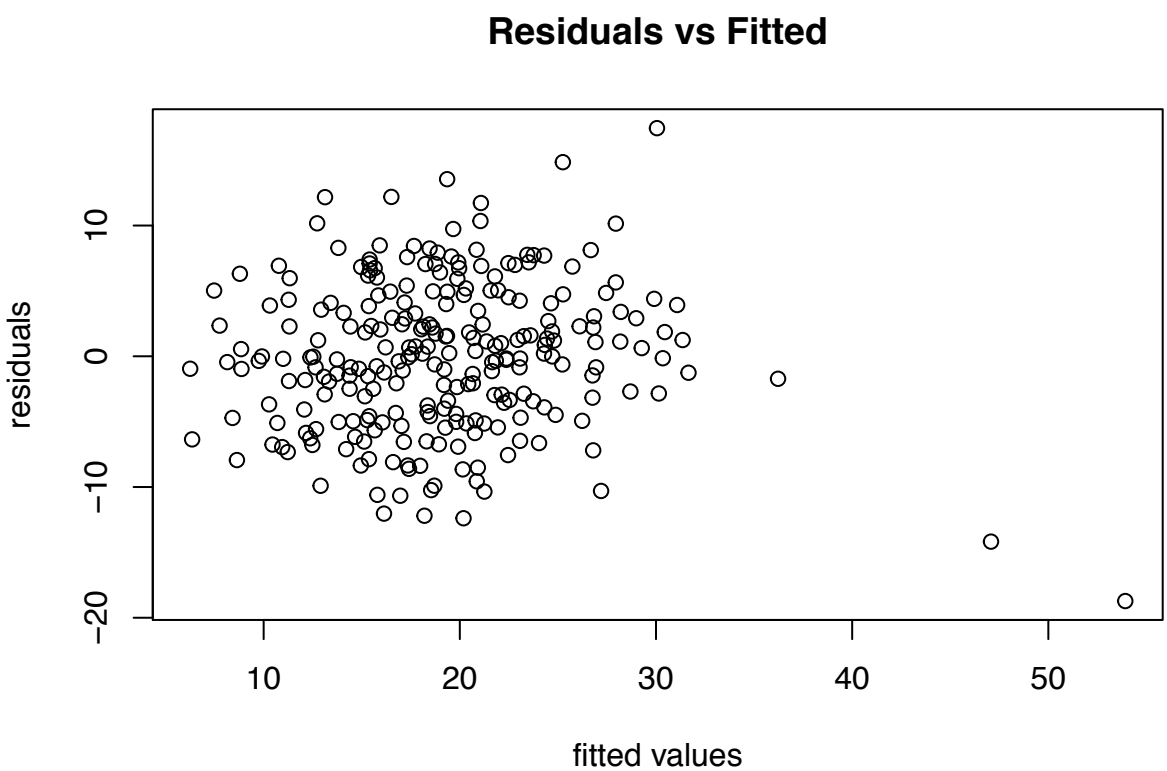
```
bodyfat <- read.csv("bodyfat.csv", header = T, stringsAsFactors = F)

dim(bodyfat)
```

```
## [1] 252  15
```

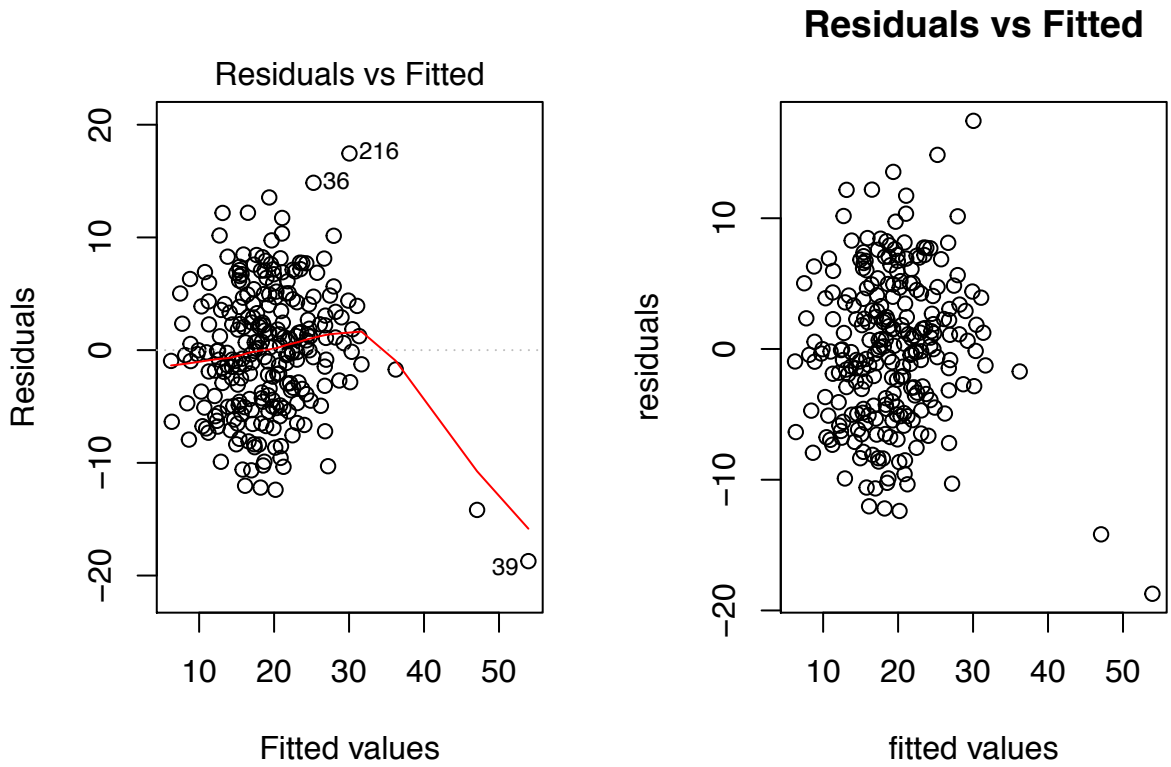
Part a

Again, the $\hat{\beta}$ is $(X^T X)^{-1} X^T y$.

```
linmod <- lm(bodyfat ~ Age + Weight + Height + Thigh , data = bodyfat)

#Create a design matrix.
y <- bodyfat$bodyfat
X <- as.matrix(cbind(1, bodyfat[,3:5], bodyfat[,10]))
colnames(X)[1] <- "intercept"
colnames(X)[5] <- "thigh"
H <- X %*% solve(t(X) %*% X) %*% t(X) #Hat matrix.
yhat <- H %*% y
residual <- y - yhat

par(mfrow = c(1,1))
plot(yhat, residual, main = "Residuals vs Fitted",
     xlab = "fitted values", ylab = "residuals")
```



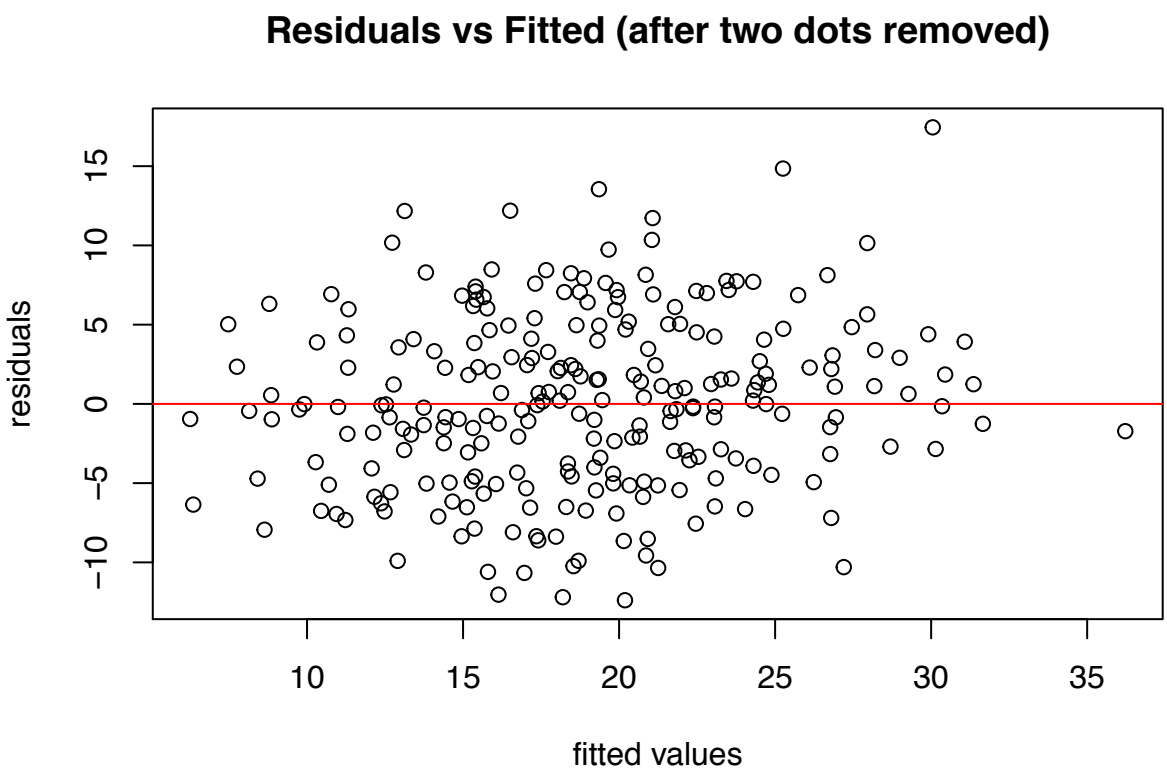
```r
#Check answers
par(mfrow = c(1,2))
plot(linmod, which = 1)
plot(linmod$fitted.values, linmod$residuals, main = "Residuals vs Fitted",
     xlab = "fitted values", ylab = "residuals")
```




```r
#Let's try to remove the two bottom-right dots.
par(mfrow = c(1,1))
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]

paste(points[1], "and", points[2], "elements might be outliers.")
```

```
## [1] "39 and 42 elements might be outliers."
```

```r
plot(linmod$fitted.values[-points], linmod$residuals[-points],
     main = "Residuals vs Fitted (after two dots removed)",
     xlab = "fitted values", ylab = "residuals")
abline(a = 0, b = 0, col = "red")
```



Comment:

The only problem in the diagram is the two dots on the right bottom. The model will show better display (and we could make better analysis) if we remove these two subjects. The graph is distorted due to those two dots. (this can be easily seen from the red loess fitted line, from the first picture) Based on what we learned in the class, the cloud of points should be clustered near around y = 0 to say the data fits the linear model (when we fit $\hat{y}$ into the data, if it fits the linear model, the model should have residuals around 0) However, as the two dots drag the trend, it might be better to just remove them.

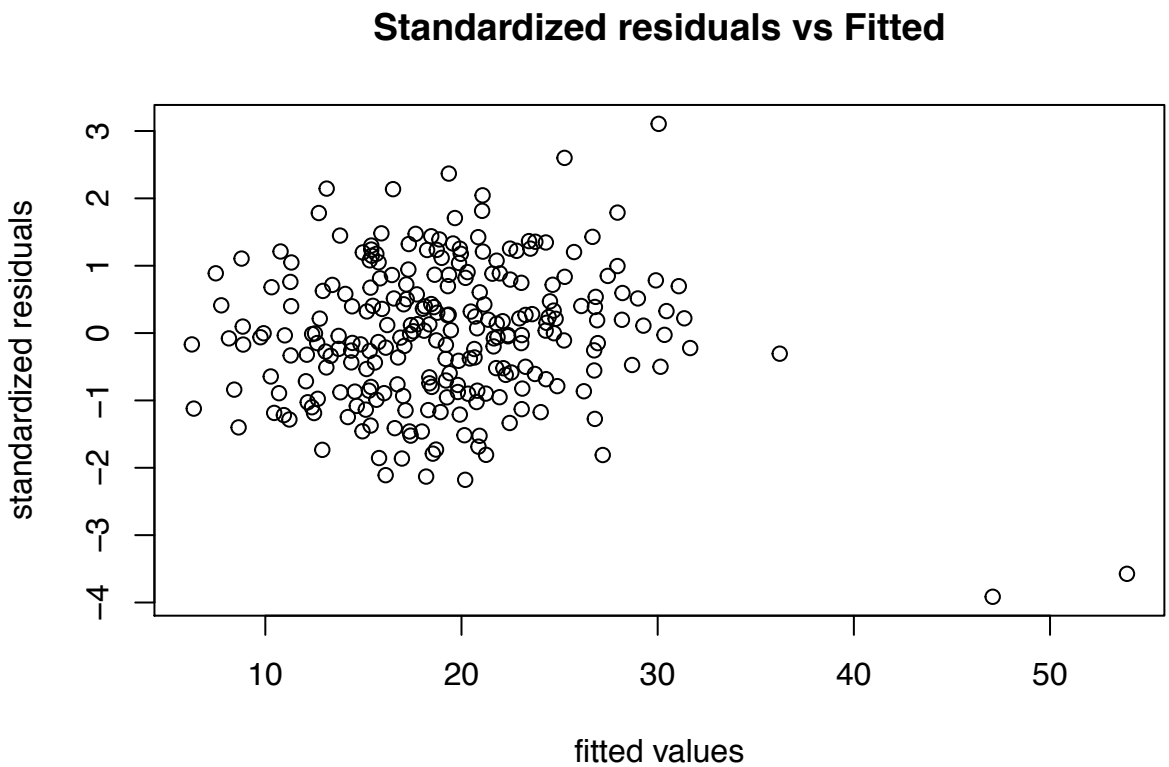
So, we can make good guess that those two dots in down right side can be **outliers** in the dataset. (this will be verified in the further investigation later. Because we do not know how big they should be to be considered as outliers)

After I removed these two dots (third picture), as you can see, I cannot really find any pattern in the diagram. And, the reisduals are randomly placed near zero, which means thtat the data can be regarded linear. Also, since the residuals roughly have horizontal band and this implies that errors are homoscedasticitic.

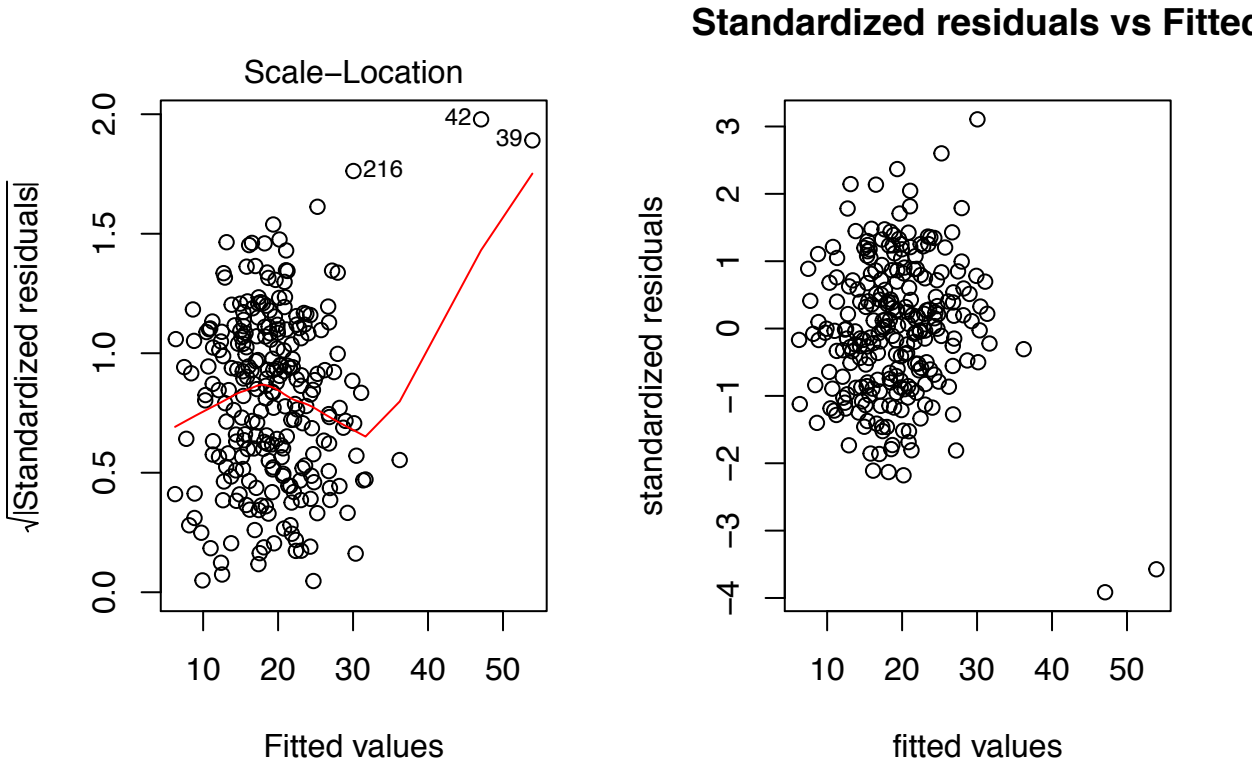
Part b

I included the square root of standard residuals v.s. fitted values with fitted line, just for visual purpose. Standardized residuals are defined as $\frac{\hat{e}_i}{\hat{\sigma}\sqrt{1-h_{ii}}}$ as residuals $\hat{e} \sim N(0,\ \hat{\sigma}^2(I-H))$. h_{ii} are the diagonal entries in Hat matrix I got from part a). Also, $\hat{\sigma}$ should be $\frac{RSS}{n-p-1}$.

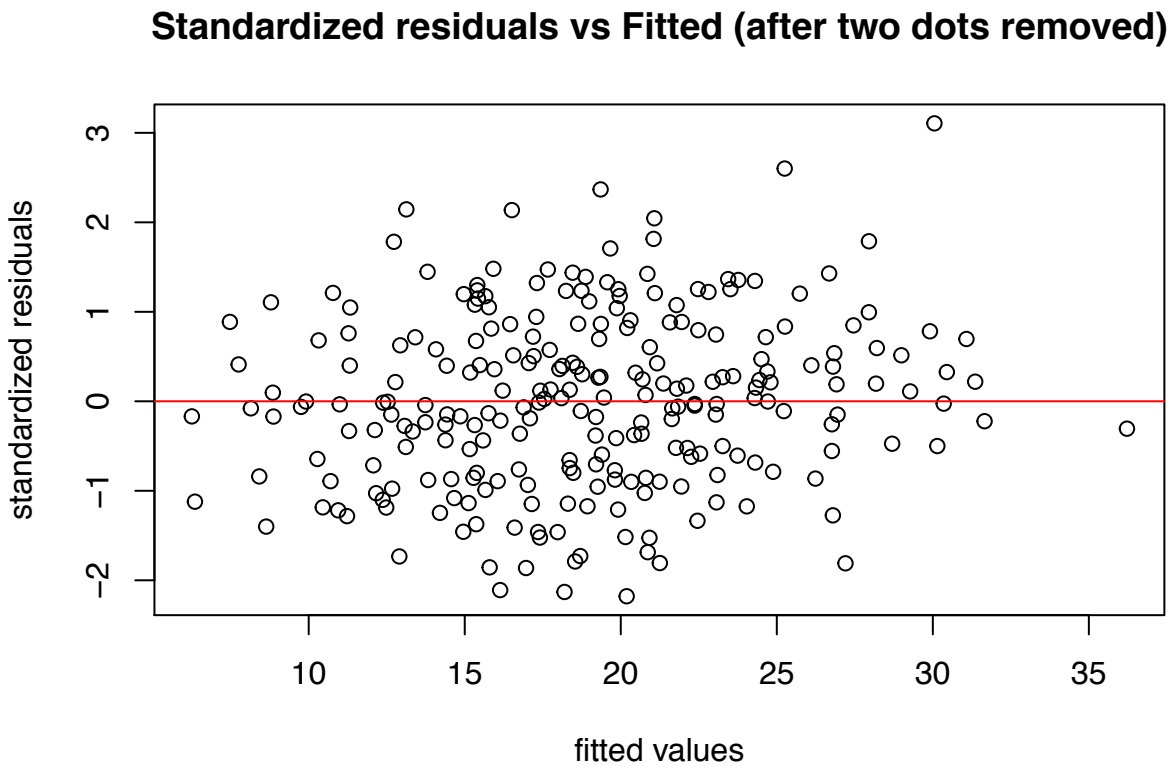
```
summ <- summary(linmod)
rse <- sqrt(sum((y - yhat)^2) / (nrow(bodyfat) - 4 - 1))
lev <- diag(H)

stdres <- residual / (rse * sqrt(1 - lev))

par(mfrow = c(1,1))
plot(yhat, stdres, main = "Standardized residuals vs Fitted",
     xlab = "fitted values", ylab = "standardized residuals")
```




```
#Check answers.
par(mfrow = c(1,2))
plot(linmod, which = 3)
plot(linmod$fitted.values, rstandard(linmod), main = "Standardized residuals vs Fitted",
     xlab = "fitted values", ylab = "standardized residuals")
```




```
#Different way to graph it. - need library MASS
#plot(linmod$fitted.values, stdres(linmod))


#Let's try to remove the two dots on bottom-right.
par(mfrow = c(1,1))
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]
plot(linmod$fitted.values[-points], rstandard(linmod)[-points],
     main = "Standardized residuals vs Fitted (after two dots removed)",
     xlab = "fitted values", ylab = "standardized residuals")
abline(a = 0, b = 0, col = "red")
```



Comment:

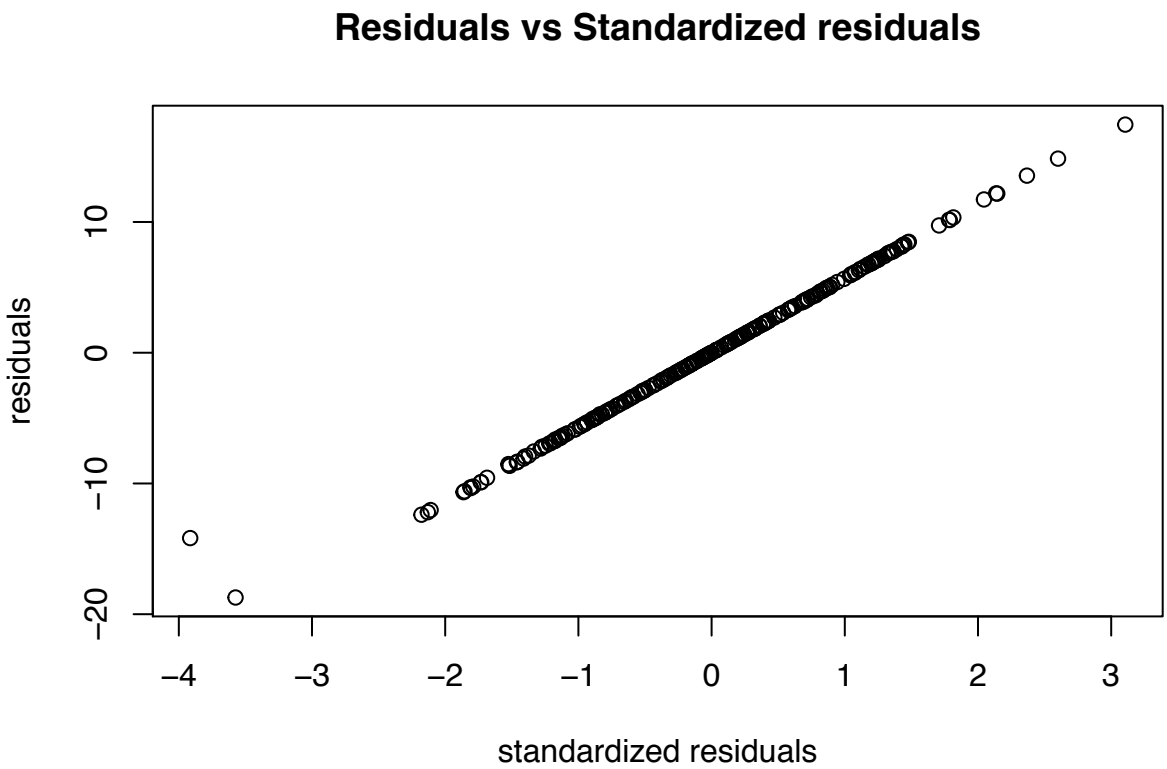
Again, as the diagram shwos similiar pattern as in part a. Similiar analysis as I stated in part a). After I removed these two dots (third picture), as you can see, I cannot really find any pattern in the diagram. And, the reisduals are randomly placed near zero, which means thtat the data can be regarded linear. Also, since the residuals roughly have horizontal band and this implies that errors are homoscedasticitic.

The only difference between the picture in part a) and part b) is the two points at the margin at the bottom-right. If we think residual of i is $\hat{e_i}$, then, standardized residual of i is $\frac{\hat{e_i}}{\sigma(1-h_i)} \approx \frac{\hat{e_i}}{\hat{\sigma}(1-h_i)}$. And, by looking at the difference bettween these two pictures, I can make a good guess that those two points might have high leverages. (if I divide residual with standardized residual, I will get $\hat{\sigma}(1-h_i)$, and since $\hat{\sigma}$ is fixed, $1-h_i$ is the one that makes any change)

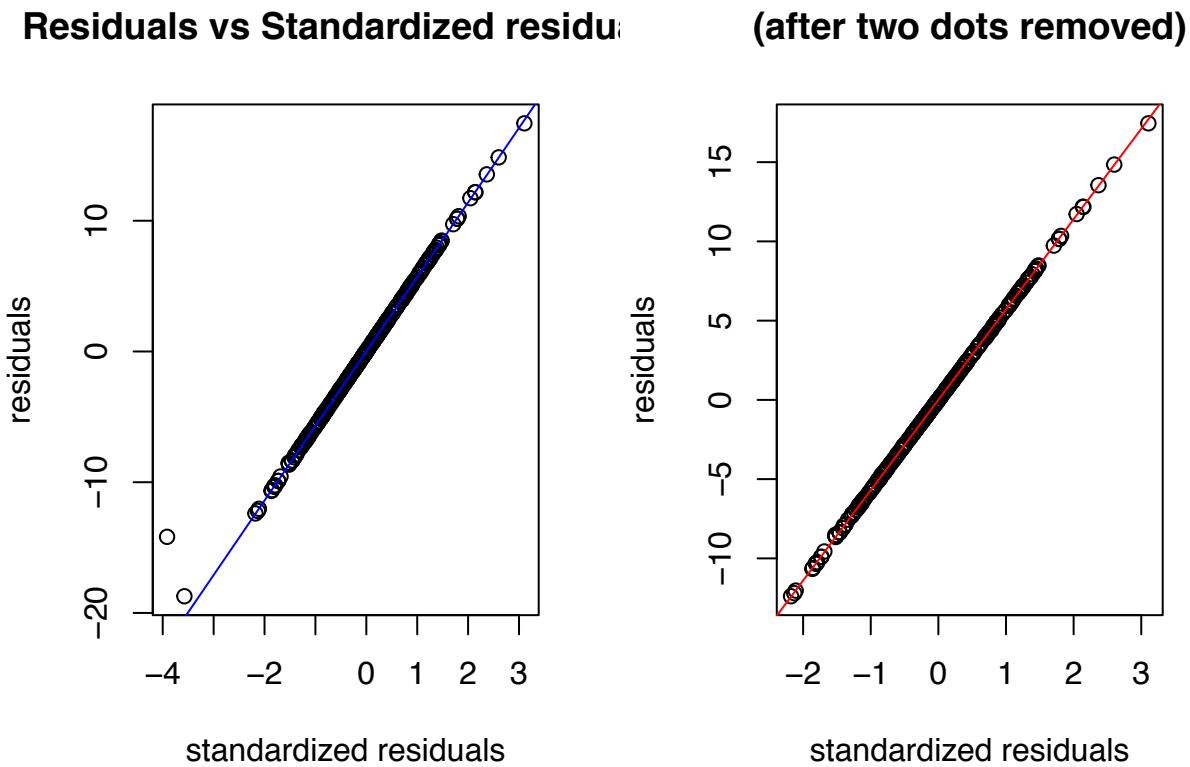
Also, as we could find out from the diagram, we can kind-of tell that 216th, 39th, and 42th observatiosn are outliers. (as absolute numbers of standardized residual are bigger than 3 as we learned in the class)

Part c

```
par(mfrow = c(1,1))
plot(stdres, residual, main = "Residuals vs Standardized residuals",
     xlab = "standardized residuals", ylab = "residuals")
```




```
#Check answers.
par(mfrow = c(1,2))

plot(rstandard(linmod), linmod$residuals, main = "Residuals vs Standardized residuals",
     xlab = "standardized residuals", ylab = "residuals")
abline(0, 5.7, col = "blue")


#Let's try to remove the two dots on bottom-left.
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]
plot(rstandard(linmod)[-points], linmod$residuals[-points],
     main = "(after two dots removed)",
     xlab = "standardized residuals", ylab = "residuals")
abline(a = 0, b = 5.7, col = "red")
```



Comment:

As it is shown, 39 and 42 elements are not on the straight line, and after removing these two, everything is on the straight line. As I mentioned in part b), I can say that these two points might have high leverages. (Also, as professor mentioned in the class, whenever standardized residual absolute magnitude is larger than 3, it can be considered as large. So, 216th observation might be an outlier)

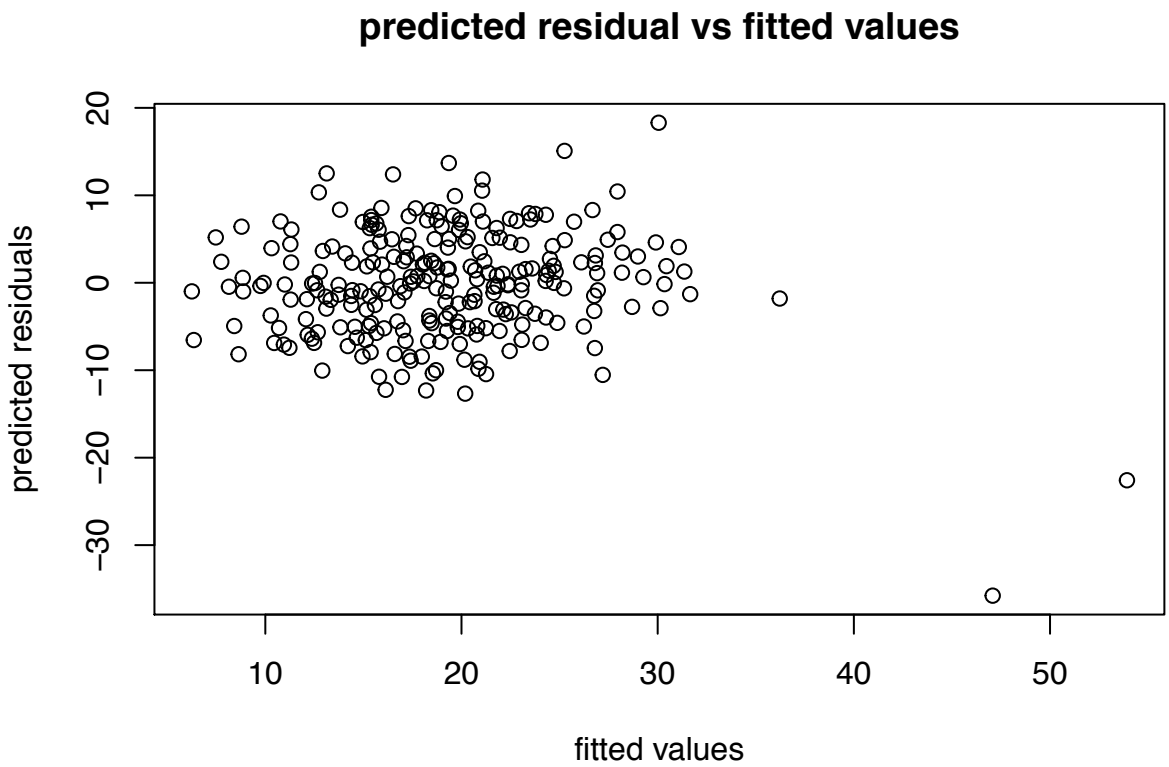
Standardized residuals are defined as $\frac{\hat{e_i}}{\hat{\sigma}\sqrt{1-h_{ii}}}$ as residuals $\hat{e} \sim N(0,\ \hat{\sigma}^2(I-H))$.

Part d

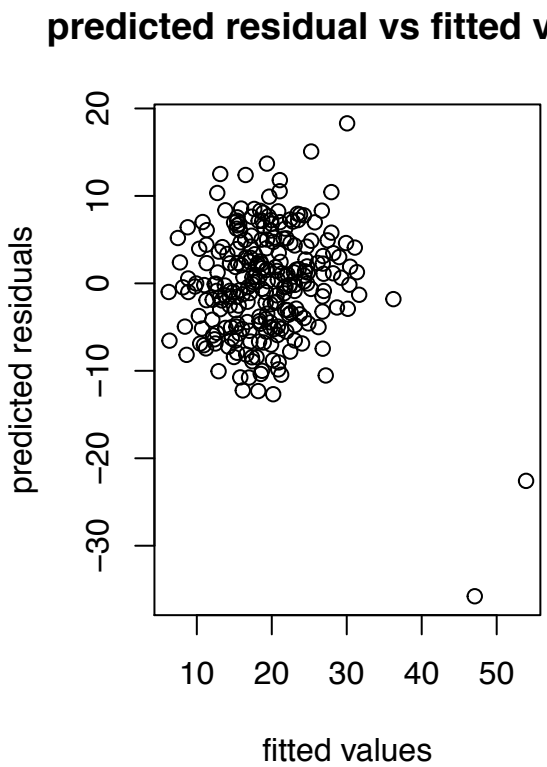
The i-th predicted residual is $\hat{e_{[i]}} = y_i - x_i^T\hat{\beta_{[i]}} = \frac{\hat{e_i}}{1-h_{ii}}$. Using Woodbury matrix identity, we could say they are equal, and that way, it is way easier for us to compute predicted residual (since we do not actually need to computer n times predicted residual).

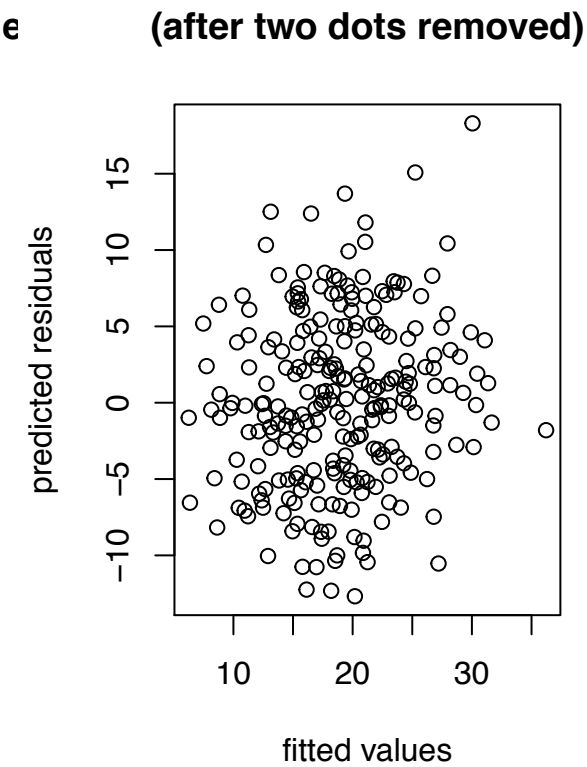
```
par(mfrow = c(1,1))

pre_res <- residual / (1 - lev)
```

```r
plot(yhat, pre_res, main = "predicted residual vs fitted values",
     xlab = "fitted values", ylab = "predicted residuals")
```




```r
#Check answers.
pre_res <- linmod$residuals / (1 - lev)

par(mfrow = c(1,2))

plot(linmod$fitted.values, pre_res, main = "predicted residual vs fitted values",
     xlab = "fitted values", ylab = "predicted residuals")

#Let's try to remove the two dots on bottom-left.
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]
plot(linmod$fitted.values[-points], pre_res[-points],
     main = "(after two dots removed)",
     xlab = "fitted values", ylab = "predicted residuals")
```






```
#Three other ways to get leverages.
# 1. x <- cbind(1, bodyfat$Age, bodyfat$Weight, bodyfat$Height, bodyfat$Thigh)
# x <- unname(as.matrix(x))
# lev <- hat(x)
#
# 2. influence(linmod)$hat
#
# 3. diag(x %*% solve(t(x) %*% x) %*% t(x))
#
# 4. hatvalues((linmod))
```

Comment:

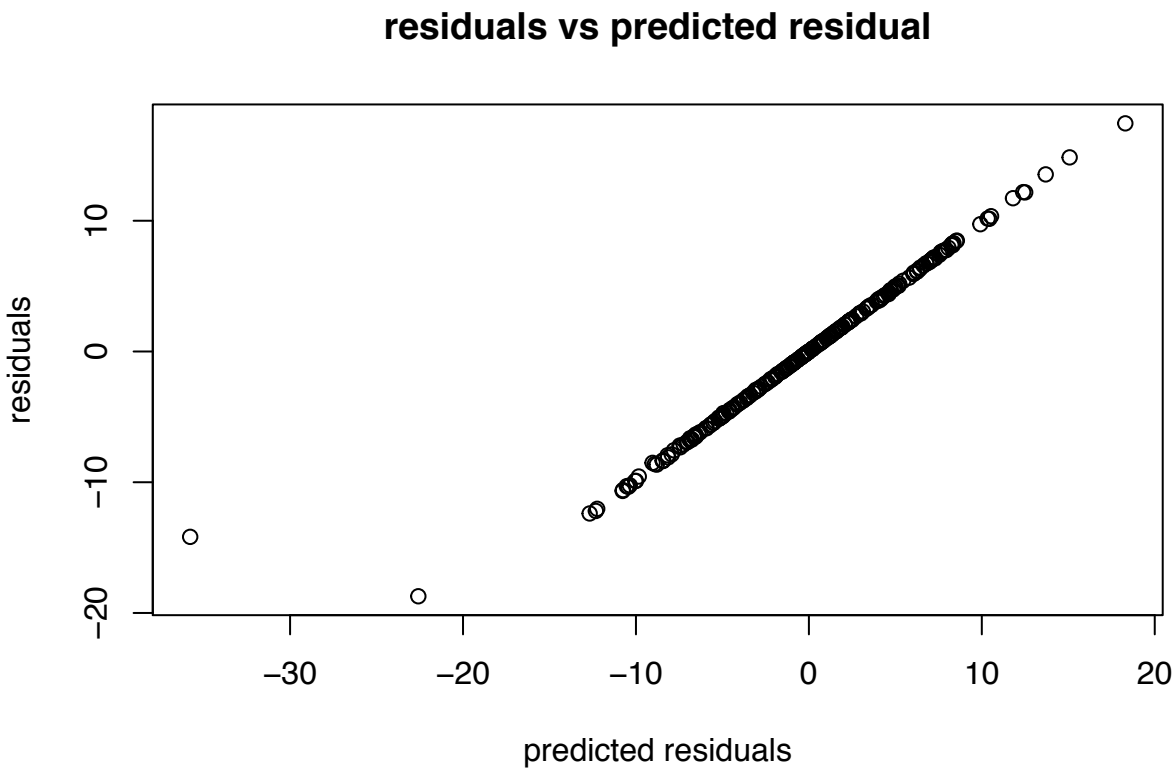
If the leverage is large, the absolute magnitude of predicted residual might be large. Again, I included the diagram after removing those two points, so I can visually compare how those two points (39th and 42th observations) are impacting the entire picture.

However, it is hard to conclude that they are outliers just because their magnitudes are big. To conclude it (decide if they are big enough to be considered as outliers), I need to find standardized predicted residual (since it follows t_{n-p-2}, I can get p-value).

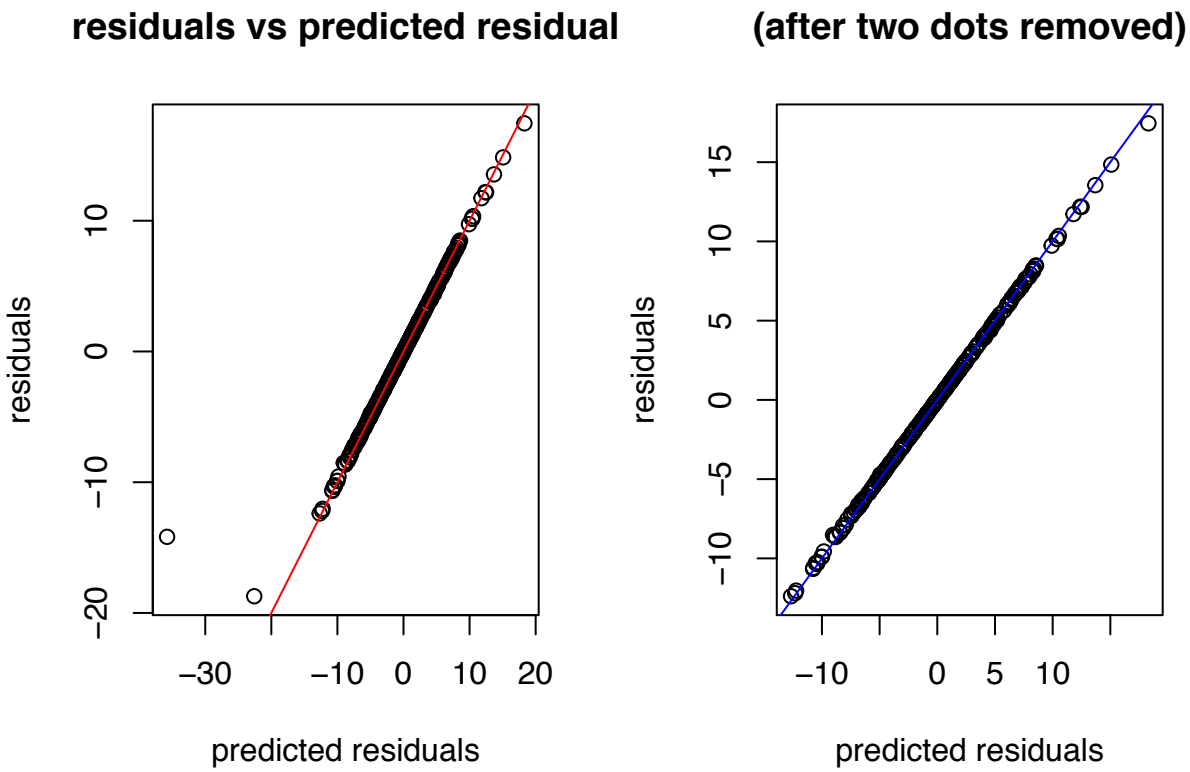
Part e

The i-th predicted residual is $\hat{e_{[i]}} = y_i - x_i^T \hat{\beta_{[i]}} = \frac{\hat{e_i}}{1-h_{ii}}$ and residual is $\hat{e_i}$.

```
par(mfrow = c(1,1))
plot(pre_res, residual, main = "residuals vs predicted residual",
     xlab = "predicted residuals", ylab = "residuals")
```




```
#Check answers.
par(mfrow = c(1,2))

plot(pre_res, linmod$residuals, main = "residuals vs predicted residual",
     xlab = "predicted residuals", ylab = "residuals")
abline(0, 1, col = "red")

#Let's try to remove the two dots on bottom-left.
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]
plot(pre_res[-points], linmod$residuals[-points],
     main = "(after two dots removed)",
     xlab = "predicted residuals", ylab = "residuals")
abline(0, 1, col = "blue")
```



Comment:

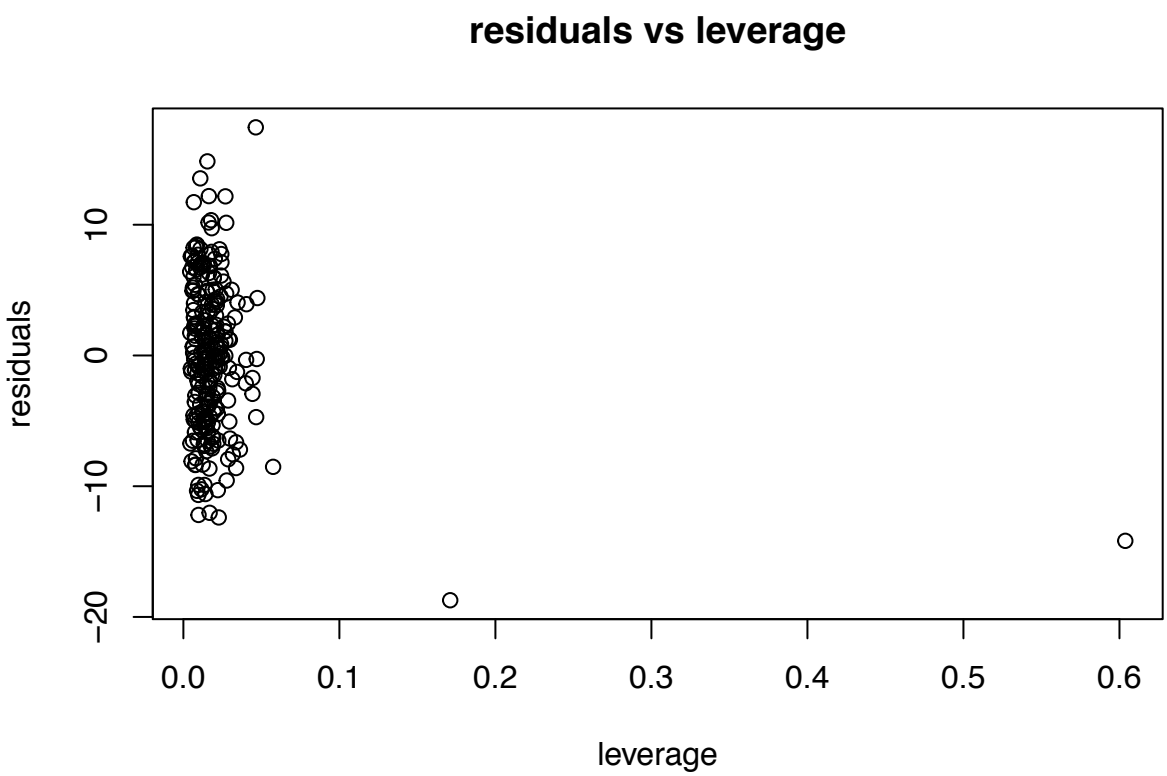
Again, predicted residual is just a residual divided by (1 - leverage). So, these two points (39th and 42th) that are not on the straight line imply that they might have unusual/high leverages.

Also, their absolute magnitudes of predicted residuals are defintely high.

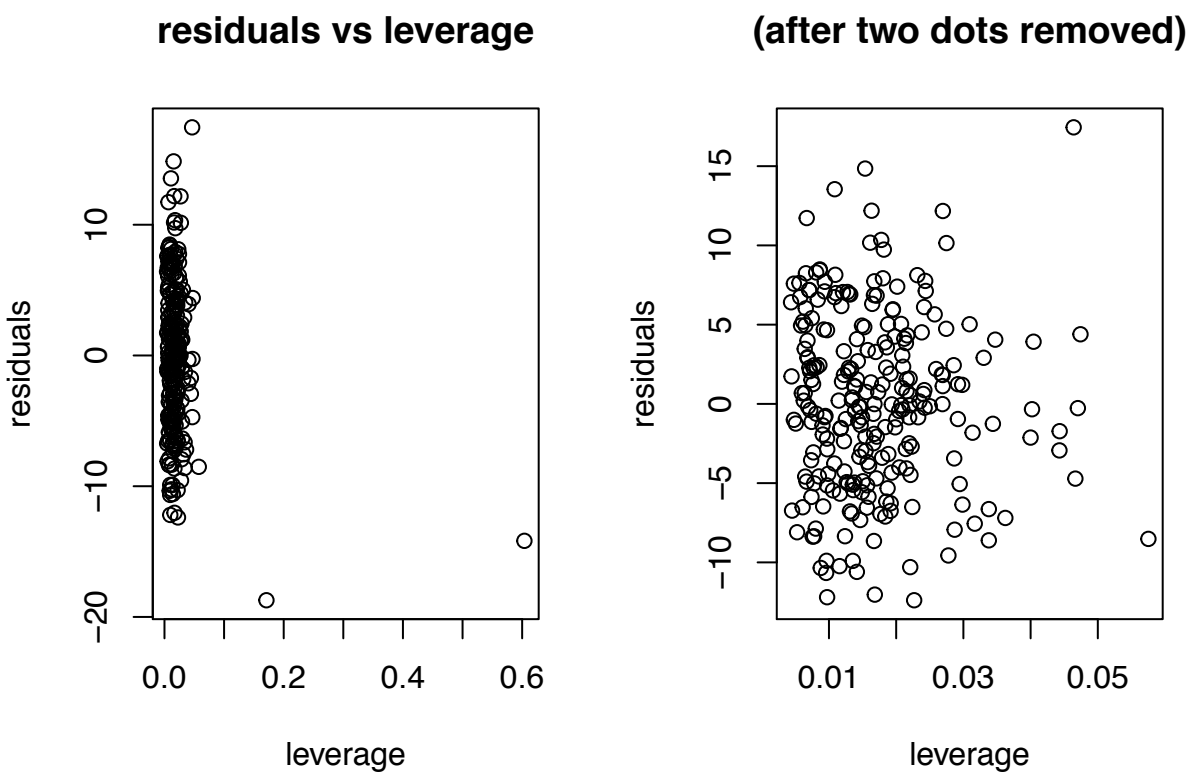
Part f

I will use the embedded plot(lm) function to show stadnardized residual v.s. leverage and Cook's distance v.s. leverage, since I could get general ideas after I plotted it. Also, this shows Cook's distance with the red dotted-contours.

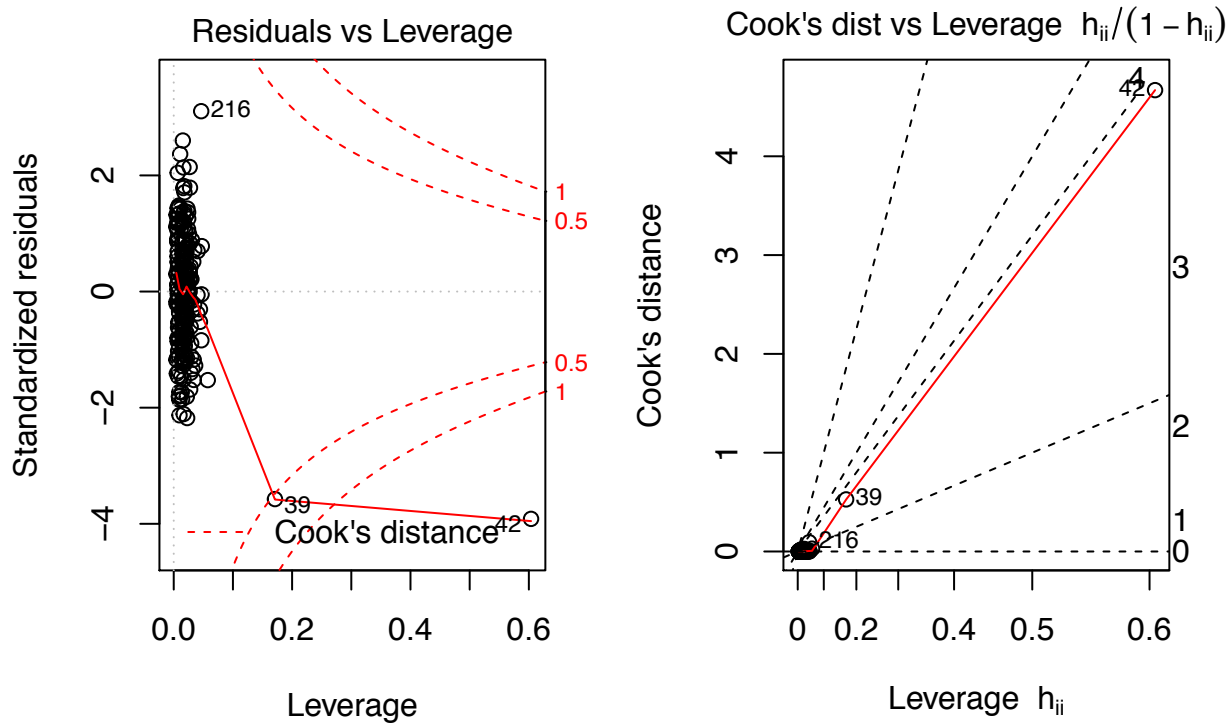
```
plot(lev, residual, main = "residuals vs leverage",
     xlab = "leverage", ylab = "residuals")
```




```
#Check answers.
par(mfrow = c(1,2))
x <- cbind(1, bodyfat$Age, bodyfat$Weight, bodyfat$Height, bodyfat$Thigh)
x <- unname(as.matrix(x))
lev <- hat(x)

plot(lev, linmod$residuals, main = "residuals vs leverage",
     xlab = "leverage", ylab = "residuals")

#Let's try to remove the two dots on bottom-left.
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]
plot(lev[-points], linmod$residuals[-points],
     main = "(after two dots removed)",
     xlab = "leverage", ylab = "residuals")
```




```r
#Extra
plot(linmod, which = 5)
plot(linmod, which = 6)
```




```r
points2 <- order(linmod$residuals, decreasing = TRUE)[1]
paste0(points2, "th observation might also be an outlier.")
```

```
## [1] "216th observation might also be an outlier."
```

Comment:

So, from the diagrams, as I have kept said from the previous parts, 39th and 42th have high leverages. Also, I found another another point (216th observation) that might be possible to be an outlier (but not high leverage).

In here, 42th observation have both highest leverage (over 0.6) and absolute magnitude of residual; thus, it has the highest Cook's distance.

Although, 216th observation has pretty high residual, its Cook's distance is not that high because the leverage is not high at all. Remeber that Cook's distance is determined by both residual and leverage. (*Be aware that by just looking at Cook's distance, we really cannot conclude the points will have high leverage or residual. So, that is why we need to make separate diagram of leverage only or standardized residual only, to see how the observation is like)

From the diagram, we can definitely find 39 (weak - at the boundary of first contour of Cook's distance) and 42 (strong - beyond the second contour) are having high Cook's distance and standardized residuals (magnitude-wise). So, this implies that those two points might be **influential observations**. (by the definition, Cook's distance is used to estimate the influence of observations - large residuals (outliers) and high leverage) So, Cook's distance is larger as having larger residuals and leverage.

Part g

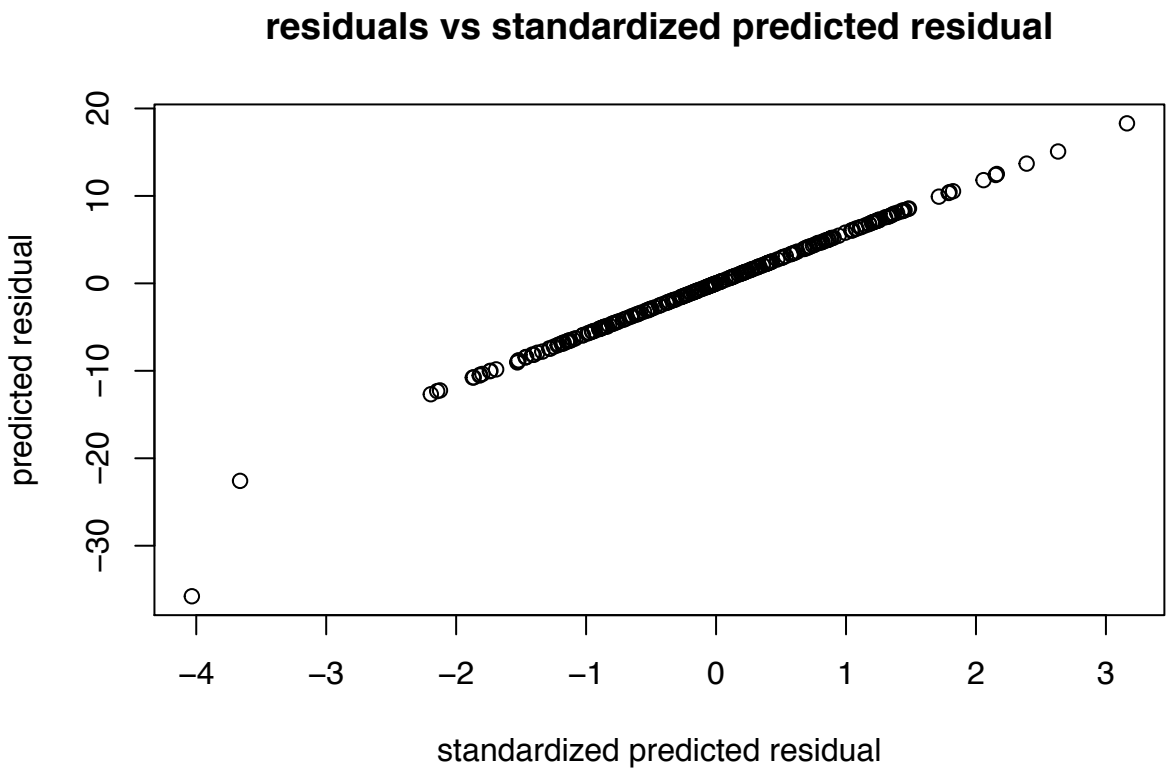
First, $\text{var}(\hat{e_{[i]}}) = \frac{\sigma^2}{1-h_{ii}} \approx \frac{\hat{\sigma}^2}{1-h_{ii}}$.

Standardized predicted reisdual is $\frac{\hat{e_{[i]}}\sqrt{1-h_{ii}}}{\sigma} \approx t_i = \frac{\hat{e_{[i]}}\sqrt{1-h_{ii}}}{\sqrt{RSS_{[i]}/n-p-2}} \sim t(n-p-2)$, where $\hat{e_{[i]}} = \frac{\hat{e_i}}{1-h_{ii}}$.

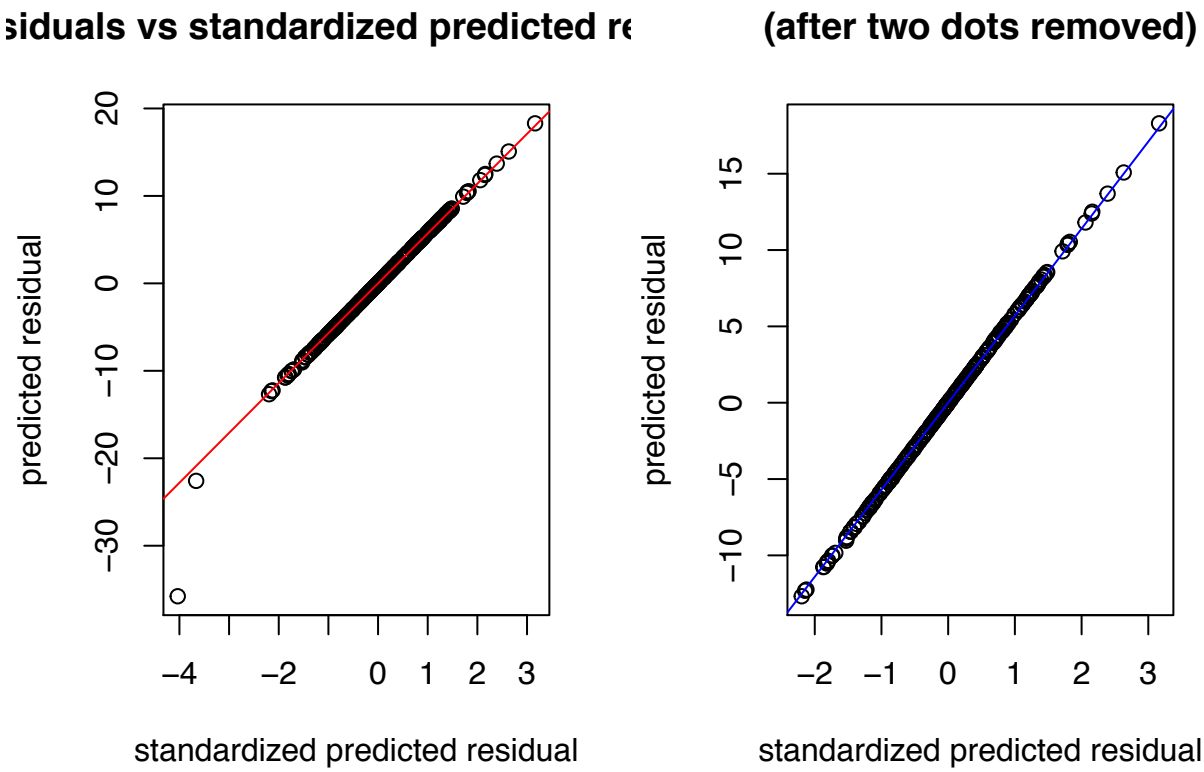
Alternatively, $t_i = r_i\sqrt{\frac{n-p-2}{n-p-1-r_i^2}}$, where r_i is the standardized residual.

And, $RSS_{[i]} = \sum_{i=1}^{n} \hat{e_{[i]}}^T \hat{e_{[i]}}$.

```r
par(mfrow = c(1,1))

stdpre_res <- stdres * sqrt((nrow(bodyfat) - 4 - 2)/(nrow(bodyfat) - 4 - 1 - (stdres)^2))

# RSS_i <- crossprod(pre_res, pre_res)
#(pre_res * sqrt(1 - lev)) / as.numeric(sqrt(RSS_i / (nrow(bodyfat) - 4 - 2))) -- different...???
# ==> Mistake!!! RSS_[i] has nth to do with e_[i]!!!

plot(stdpre_res, pre_res, main = "residuals vs standardized predicted residual",
     xlab = "standardized predicted residual", ylab = "predicted residual")
```




```r
#Check answers.
par(mfrow = c(1,2))

plot(rstudent(linmod), pre_res, main = "residuals vs standardized predicted residual",
     xlab = "standardized predicted residual", ylab = "predicted residual")
abline(0, 5.7, col = "red")

#Let's try to remove the two dots on bottom-left.
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]
plot(rstudent(linmod)[-points], pre_res[-points],
     main = "(after two dots removed)",
     xlab = "standardized predicted residual", ylab = "predicted residual")
abline(0, 5.7, col = "blue")
```

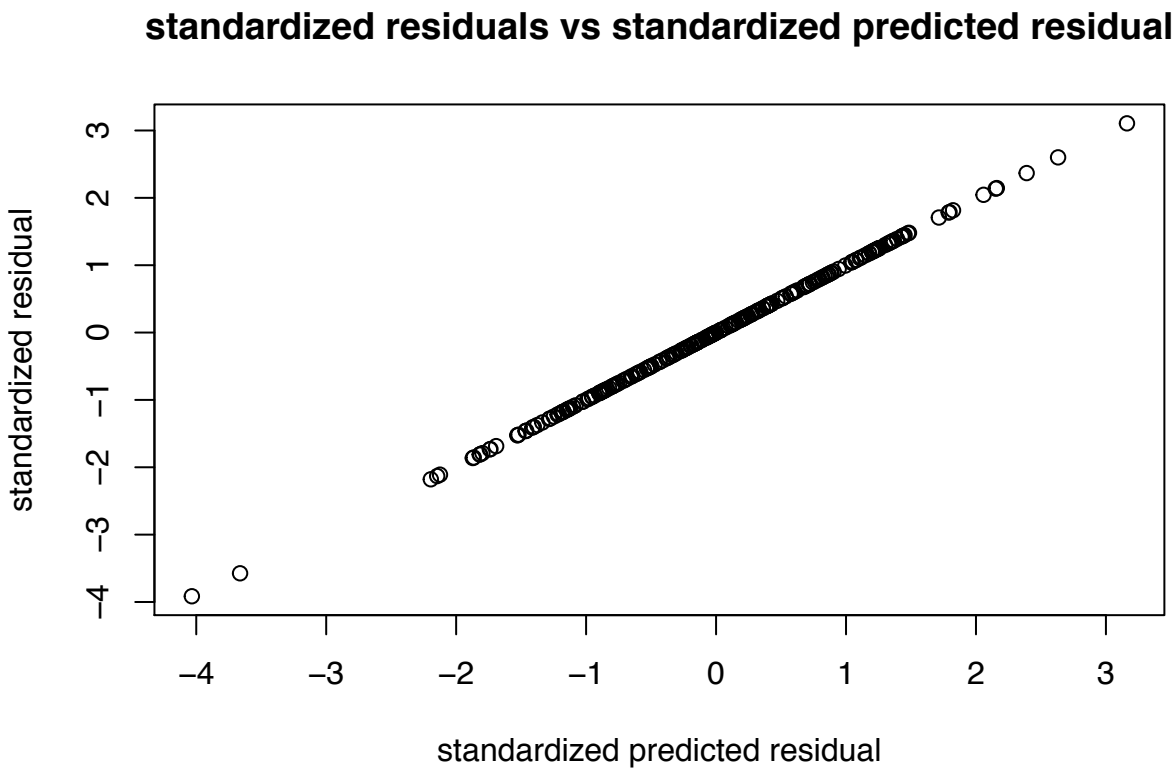


Comment:

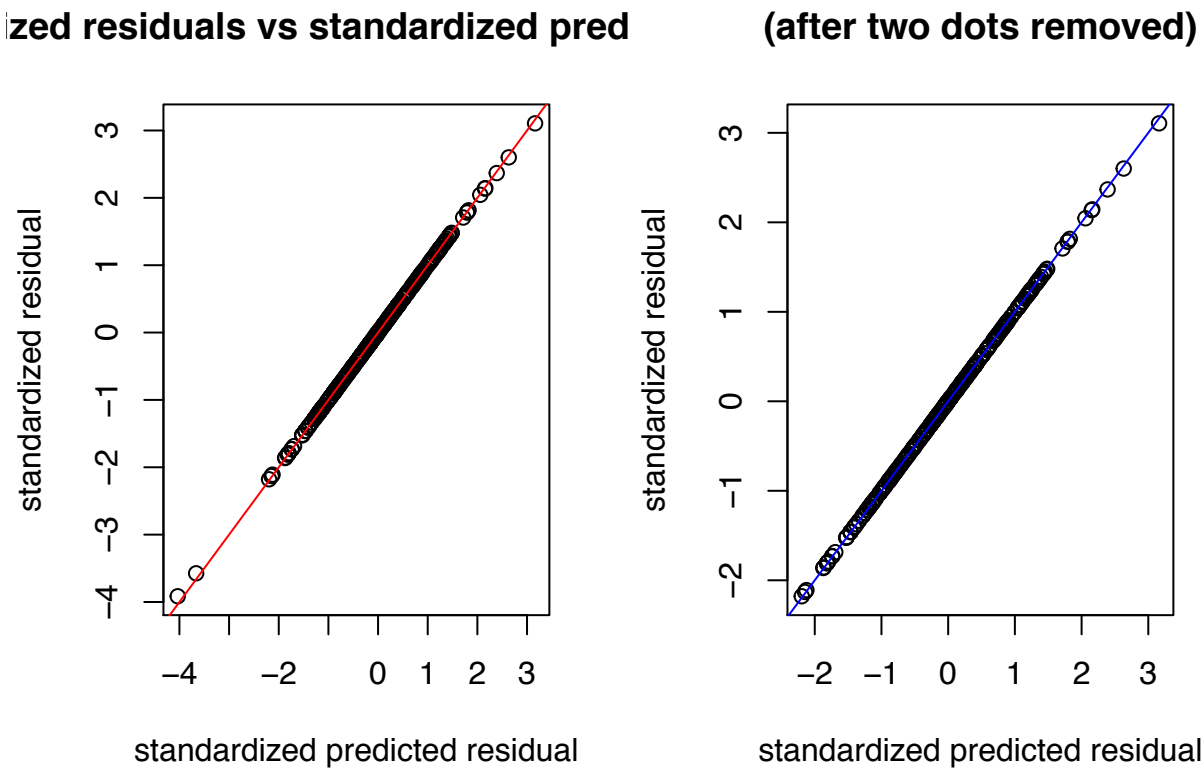
Again, 39th and 42th are not on the straight line. (outliers) Later in part k), I would get p-value using standardized predicted residual to have better idea.

Part h

```
par(mfrow = c(1,1))
plot(stdpre_res, stdres,
     main = "standardized residuals vs standardized predicted residual",
     xlab = "standardized predicted residual", ylab = "standardized residual")
```




```
#Check answers.
par(mfrow = c(1,2))

plot(rstudent(linmod), rstandard(linmod),
     main = "standardized residuals vs standardized predicted residual",
     xlab = "standardized predicted residual", ylab = "standardized residual")
abline(0, 1, col = "red")

#Let's try to remove the two dots on bottom-left.
points <- order(linmod$fitted.values, decreasing = TRUE)[c(1,2)]
plot(rstudent(linmod)[-points], rstandard(linmod)[-points],
     main = "(after two dots removed)",
     xlab = "standardized predicted residual", ylab = "standardized residual")
abline(0, 1, col = "blue")
```



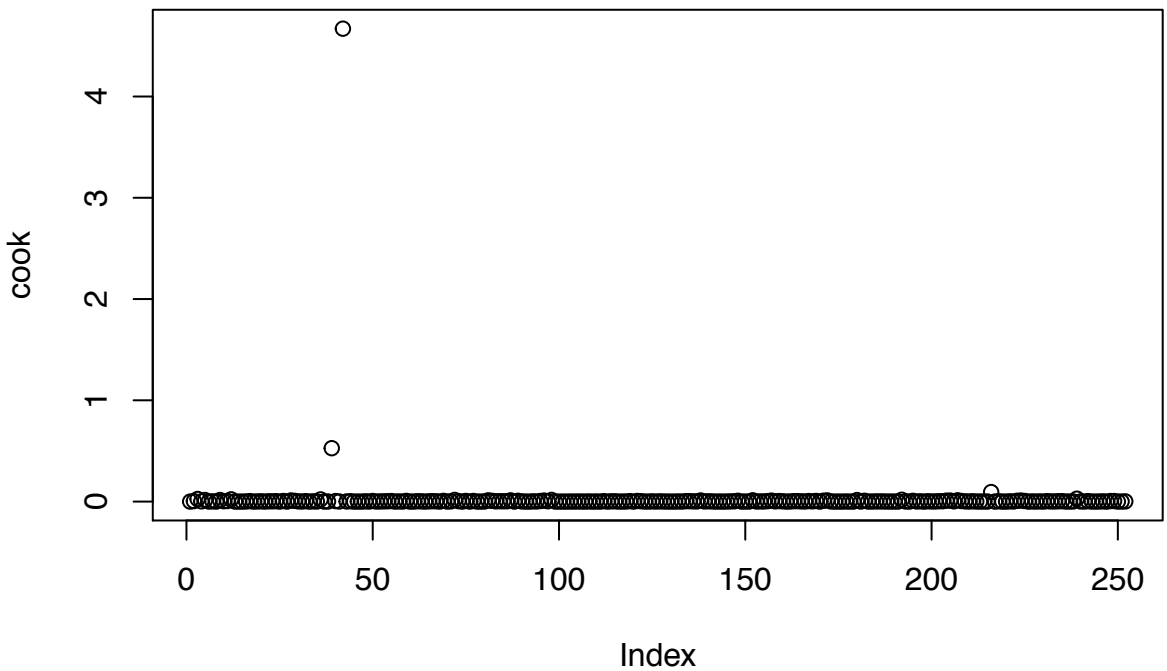
Comment:

39th and 42th elements are slightly off from the straight line. However, this diagram looks pretty straight line even with these two plots. From here, I can guess that although those two observations have unusal residuals, we divide by n - p - 2, and then square root it. So, these difference get less effective. Also, if I look at the two observations more carefully. I can tell that 42th observation (the one has the lowest standardized predicted residual) is bit more off from the straight line than 39th observation. And, this makes sense as 42th observation has more unusal residual.

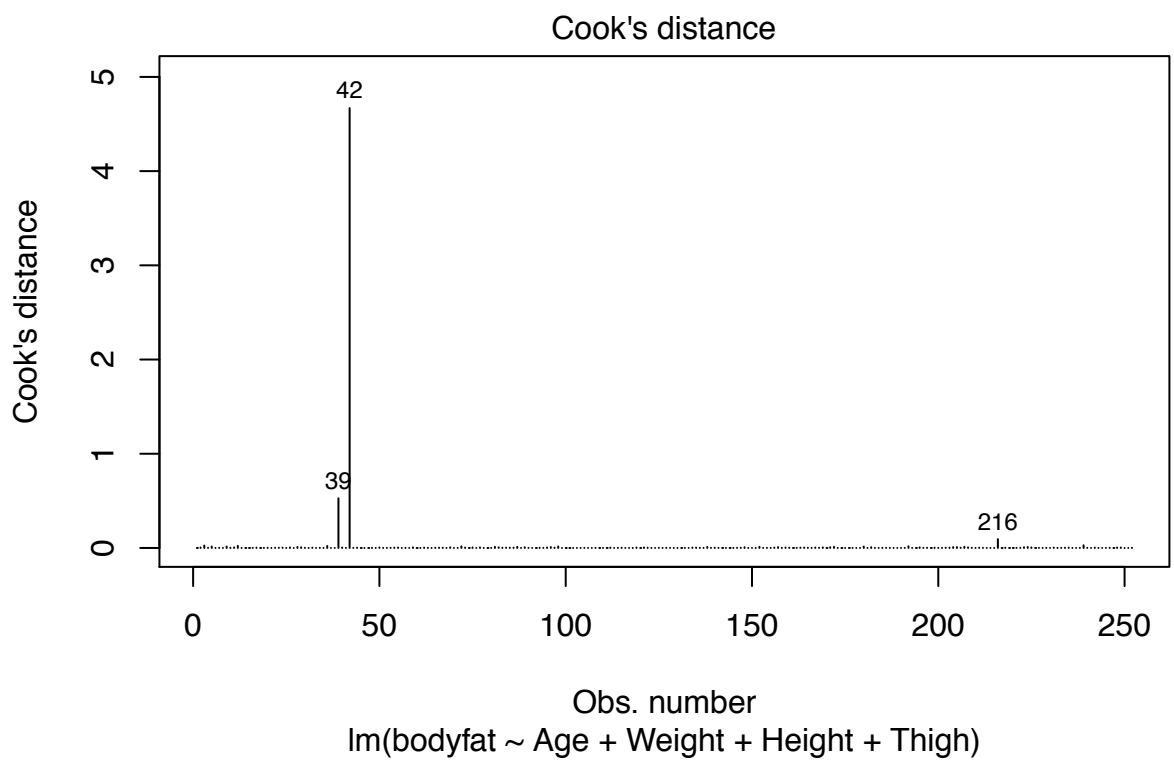
Part i

Cook's distance can be expressed as $r_i^2 \frac{h_{ii}}{(1-h_{ii})(p+1))} = \frac{r_i^2}{(p+1)} \frac{h_{ii}}{(1-h_{ii})}$, where r_i is the standardized residual.

```
cook <- stdres^2 * (lev / ((1 - lev) * (4+1)))
plot(cook)
```




```r
#Check answers.
plot(linmod, which = 4)
```




```r
#Get cook's distance
#cooks.distance()
```

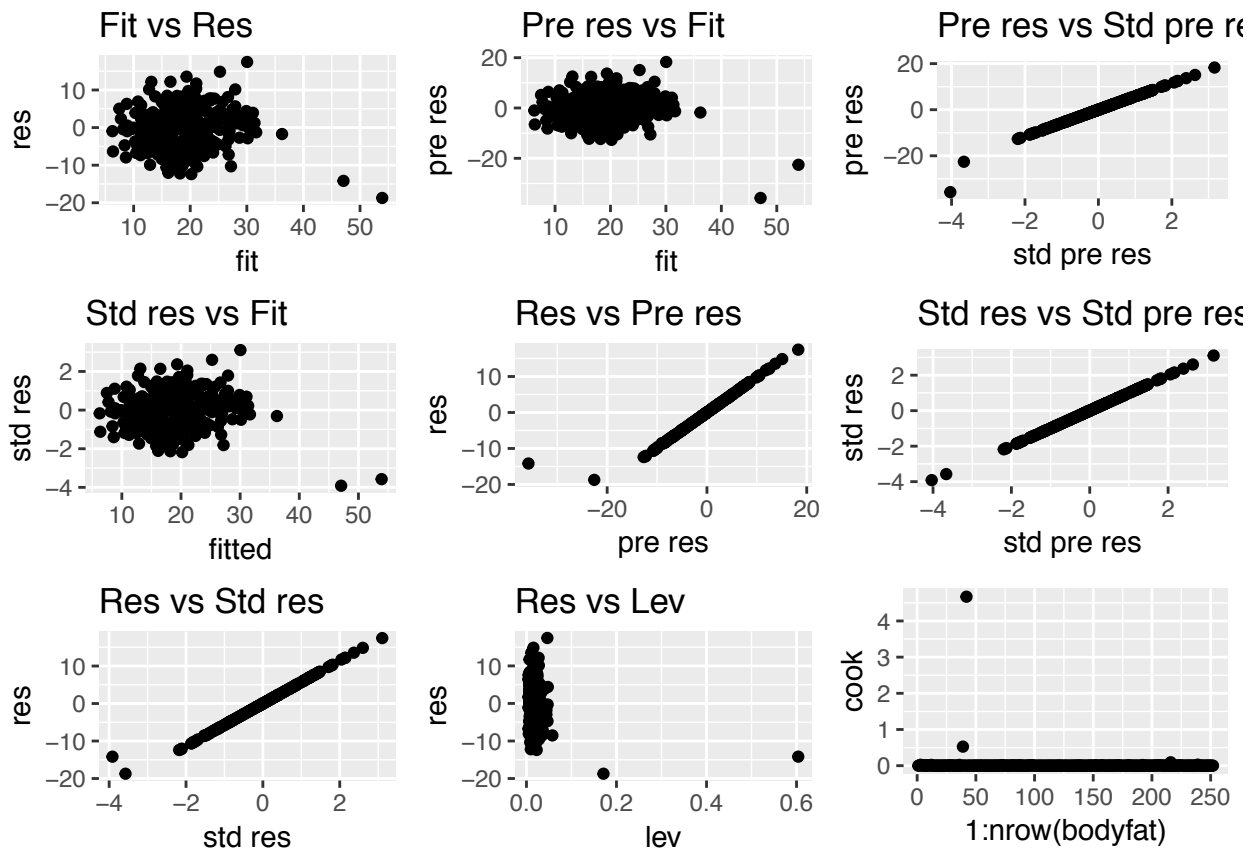
Comment:

As I explained in part f), Cook's distance shows **influential** points, and as I haved shown in part a) ~ h), 39th and 42th are obvious influential points. 216th observation also mentioned once in part f), but it is not

that influential compared to other two points.

I will show all 9 diagrams in one paint.

```
pic <- list()
pic[[1]] <- qplot(linmod$fitted.values, linmod$residuals, main = "Fit vs Res",
     xlab = "fit", ylab = "res")
pic[[2]] <- qplot(linmod$fitted.values, rstandard(linmod), main = "Std res vs Fit",
     xlab = "fitted", ylab = "std res")
pic[[3]] <- qplot(rstandard(linmod), linmod$residuals, main = "Res vs Std res",
     xlab = "std res", ylab = "res")
pic[[4]] <- qplot(linmod$fitted.values, pre_res, main = "Pre res vs Fit",
     xlab = "fit", ylab = "pre res")
pic[[5]] <- qplot(pre_res, linmod$residuals, main = "Res vs Pre res",
     xlab = "pre res", ylab = "res")
pic[[6]] <- qplot(lev, linmod$residuals, main = "Res vs Lev",
     xlab = "lev", ylab = "res")
pic[[7]] <- qplot(rstudent(linmod), pre_res, main = "Pre res vs Std pre res",
     xlab = "std pre res", ylab = "pre res")
pic[[8]] <- qplot(rstudent(linmod), rstandard(linmod),
     main = "Std res vs Std pre res",
     xlab = "std pre res", ylab = "std res")
pic[[9]] <- qplot(1:nrow(bodyfat), cook)

multiplot(plotlist = pic, cols = 3)
```



Part j

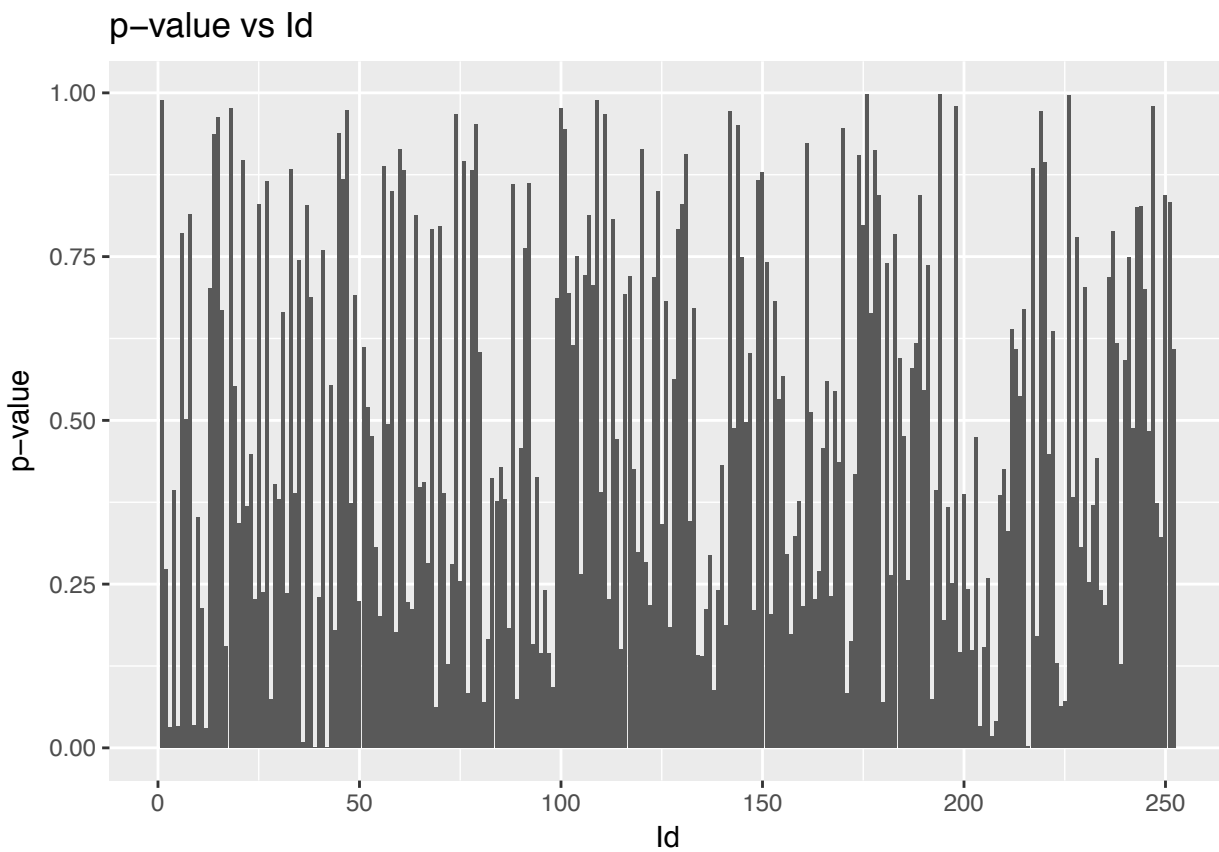
If I summarize all the plots, I can definitely say 39th and 42th observations are outliers and influential. And, 36th and 216th observations might be outliers, as we can see in part a). 216 might be an influential observation, based on what I got from part i).

Part k

Again, standardized predicted residual follows *t distribution* with n-p-2 degrees of freedom. n is 252 and p is 4, so n-p-2 should be 246.

Standardized predicted residual is good at finding outliers. As it says on the instruction, I will calculate p-value on the standardized predicted residual: $rstudent(linmod)$.

```
# Below is not saving the p-values, but saving quantile probability and give the same answer
# eventually!!!
# pval <- c()
# for (i in 1:nrow(bodyfat)){
#   pval[i] <- pt(stdpre_res[i], 246, lower.tail = F)
# }
# pval <- as.data.frame(pval)
# colnames(pval) <- "p"
# pval[,2] <- seq(1, nrow(pval))
#
#
# pval_plot <- ggplot(aes(x = V2, y = p), data = pval) +
#   geom_bar(stat = "identity") + labs(title = "p-value vs Id", x = "Id", y = "p-value")
#
# pval_plot
#
# pval %>% filter(p < 0.025 | p > 0.975)
#
# pval %>% filter(p < 0.0001 | p > 0.999)

pvals <- c()
for (i in 1:nrow(bodyfat)){
  pvals[i] <- 2 * pt(abs(stdpre_res[i]), 246, lower.tail = F)
}
pvals <- as.data.frame(pvals)
colnames(pvals) <- "p"
pvals[,2] <- seq(1, nrow(pvals))

pval_plot <- ggplot(aes(x = V2, y = p), data = pvals) +
  geom_bar(stat = "identity") + labs(title = "p-value vs Id", x = "Id", y = "p-value")
pval_plot
```




```
pvals %>% filter(p < 0.05)
```

```
##               p  V2
## 1  3.169823e-02   3
## 2  3.237542e-02   5
## 3  3.461962e-02   9
## 4  2.909586e-02  12
## 5  9.029060e-03  36
## 6  3.054138e-04  39
## 7  7.325343e-05  42
## 8  3.285382e-02 204
## 9  1.760332e-02 207
## 10 4.062107e-02 208
## 11 1.764147e-03 216
```

```
pvals %>% filter(p < 0.0001)
```

```
##              p V2
## 1 7.325343e-05 42
```

Comment:

3rd, 5th, 9th, 12th, 36th, 39th, 42th, 204th, 207th, 208th, and 216th observations have p-values less than 0.05. Under the linearity and normality of errors assumptions, since the standardized predicted residual follows t distribution with degree of freedom n - p - 2, we can test an outlier.

I would *NOT* rule only 39th and 42th based on what we learned in the lecture. If we say 252 observations are independent (not neccessarily true, but let's assume that), then I would expect around 252 * 0.05 $\approx$ 13 to be larger in absolute value and place inside of critical regions. (two-tailed) So, that is why we are getting a

conservative value $\frac{0.05}{252} \approx 0.0002$ as our critical regions. (Bonferroni correction) - then, I would get only two outliers.

Part l

```
newbody <- bodyfat[-c(39, 42), ]
linmod2 <- lm(bodyfat ~ Age + Weight + Height + Thigh , data = newbody)
summary(linmod2)
```

```
##
## Call:
## lm(formula = bodyfat ~ Age + Weight + Height + Thigh, data = newbody)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.4982  -3.7381  -0.0034   3.7581  12.0943
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.82844   13.74245   3.117  0.00205 **
## Age          0.16101    0.03164   5.089 7.18e-07 ***
## Weight       0.21150    0.03020   7.003 2.39e-11 ***
## Height      -1.18281    0.16753  -7.060 1.70e-11 ***
## Thigh        0.24418    0.15252   1.601  0.11068
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.365 on 245 degrees of freedom
## Multiple R-squared:  0.5883, Adjusted R-squared:  0.5816
## F-statistic: 87.54 on 4 and 245 DF,  p-value: < 2.2e-16
```

```
linmod <- lm(bodyfat ~ Age + Weight + Height + Thigh , data = bodyfat)
summary(linmod)
```

```
##
## Call:
## lm(formula = bodyfat ~ Age + Weight + Height + Thigh, data = bodyfat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.722  -4.283  -0.055   4.061  17.449
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.27488   11.12642  -0.204   0.8382
## Age          0.20517    0.03274   6.267 1.63e-09 ***
## Weight       0.13417    0.02952   4.545 8.59e-06 ***
## Height      -0.49810    0.11313  -4.403 1.59e-05 ***
## Thigh        0.38970    0.16142   2.414   0.0165 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.753 on 247 degrees of freedom
## Multiple R-squared:  0.5349, Adjusted R-squared:  0.5274
## F-statistic: 71.03 on 4 and 247 DF,  p-value: < 2.2e-16
```

Comment:

Although I found that 42th is the only possible outlier, I regarded 39th as an outlier as well from what I learned from the previous parts. So, I also excluded 39th observation to get better conclusion!!!

The most influential points are 39th and 42th observations. As I mentioned above, there are some other possible influential, outlying, high-leverage observations, but they are not that significant.

Although I removed those two observations, R^2 does not go up that high. By looking at F-statistic, it seems makes sense to fit the linear model; however, I think there is a better model to be fitted for this data. However, R^2 is not a good measurement here, so, I looked at p-values for intercept and each explanatory variable.

After removing two outliers, I found that intercept, weight, and height are more significant (do not reject); however, less significant in age and thigh. Especially, I might remove thigh from our model.