

AdvStDaAn, Worksheet, Week 2

Micheal Lappert

05.04.2022

Contents

Exercise 1	1
Exercise 1.a)	3
Exercise 1.b)	4
Exercise 1.c)	5
Exercise 2	8
Exercise 3	22
Exercise 4	25
Exercise 4.a)	26
Exercise 4.b)	31
Exercise 4.c)	32
Question to 4.c)	33

Exercise 1

```
path <- file.path('Datasets', 'sniffer.dat')
df <- read.table(path, header=TRUE)

summary(df)
```

Dataset loading and sanity check:

```
##      Temp.Tank      Temp.Gas      Vapor.Tank      Vapor.Dispensed
##  Min.   :31.00   Min.   :35.00   Min.   :2.590   Min.   :2.590
##  1st Qu.:37.00   1st Qu.:41.00   1st Qu.:3.290   1st Qu.:3.373
##  Median :60.00   Median :60.00   Median :4.285   Median :4.090
##  Mean   :57.91   Mean   :55.91   Mean   :4.422   Mean   :4.324
##  3rd Qu.:62.00   3rd Qu.:62.00   3rd Qu.:4.630   3rd Qu.:4.540
##  Max.   :92.00   Max.   :92.00   Max.   :7.450   Max.   :7.450
##           Y
##  Min.   :16.00
```

```
## 1st Qu.:23.75
## Median :31.50
## Mean   :31.12
## 3rd Qu.:34.50
## Max.    :55.00
```

```
dim(df)
```

```
## [1] 32  5
```

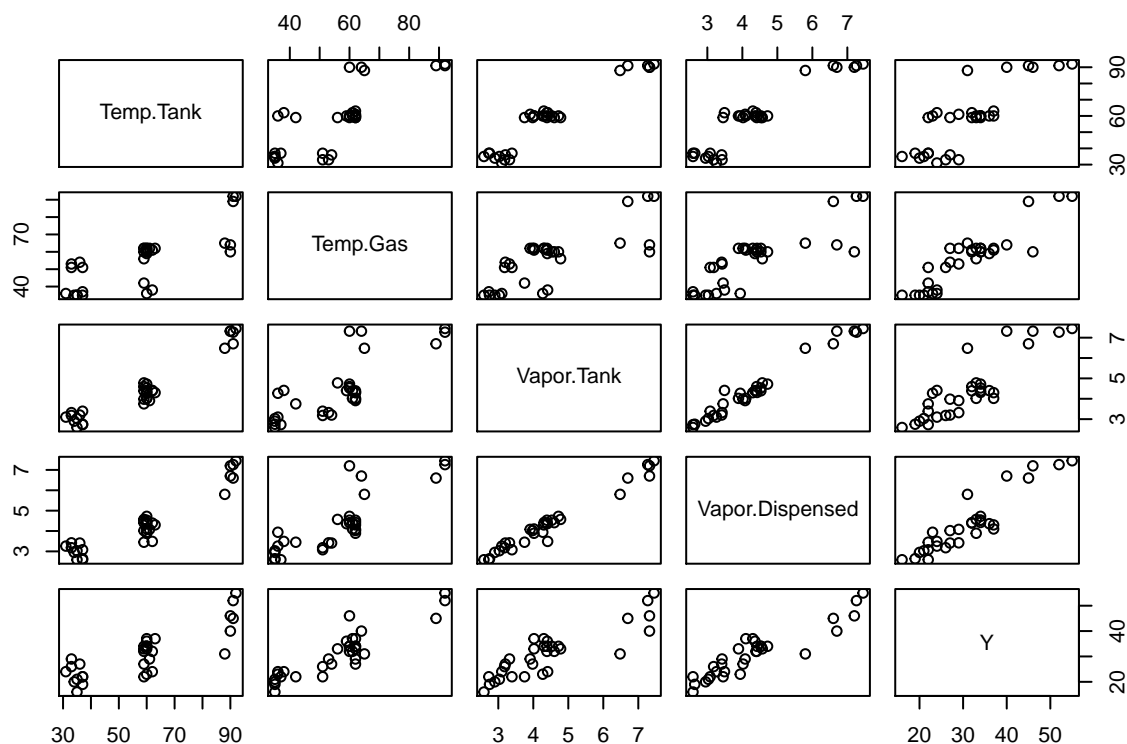
```
head(df)
```

```
##   Temp.Tank Temp.Gas Vapor.Tank Vapor.Dispensed  Y
## 1      33      53      3.32      3.42 29
## 2      31      36      3.10      3.26 24
## 3      33      51      3.18      3.18 26
## 4      37      51      3.39      3.08 22
## 5      36      54      3.20      3.41 27
## 6      35      35      3.03      3.03 21
```

```
tail(df)
```

```
##   Temp.Tank Temp.Gas Vapor.Tank Vapor.Dispensed  Y
## 27      60      62      4.02      3.89 33
## 28      59      62      3.98      4.02 27
## 29      59      62      4.39      4.53 34
## 30      37      35      2.75      2.64 19
## 31      35      35      2.59      2.59 16
## 32      37      37      2.73      2.59 22
```

```
plot(df)
```



Data looks like it is highly correlated with each other. But we keep it this way for the first exercises.

Exercise 1.a)

Fitting a first model without any transformations to the data:

```
lm1.1 <- lm(Y ~ ., data = df)
```

The model looks initially not too bad. For a proper evaluation one would need to perform a residual and sensitivity analysis to investigate the adequacy of the model. But for this exercise we keep the track of the worksheet.

E1.a)(I) Estimated coefficients

```
coef(lm1.1)
```

```
##      (Intercept)      Temp.Tank      Temp.Gas      Vapor.Tank Vapor.Dispensed
##      1.01501756     -0.02860886      0.21581693     -4.32005167      8.97488928
```

E1.a)(II) F-statistic

```
summary(lm1.1)
```

```
##
```

```
## Call:
## lm(formula = Y ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.586 -1.221 -0.118  1.320  5.106
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.01502    1.86131   0.545  0.59001
## Temp.Tank      -0.02861    0.09060  -0.316  0.75461
## Temp.Gas        0.21582    0.06772   3.187  0.00362 **
## Vapor.Tank     -4.32005    2.85097  -1.515  0.14132
## Vapor.Dispensed 8.97489    2.77263   3.237  0.00319 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.73 on 27 degrees of freedom
## Multiple R-squared:  0.9261, Adjusted R-squared:  0.9151
## F-statistic: 84.54 on 4 and 27 DF,  p-value: 7.249e-15
```

The p-value of the F-statistic is « 0.05 indicating that at least one of the variables can not be 0 and therefore are important to describe the response value. Even though, the p-values of the t-test indicate that not all of them are of the same importance. In this case are only 2 explanatory variables significantly important (Temp.Gas & Vapor.Dispensed).

E1.a)(III) Variance Inflation Factor (VIF)

Inspecting multicollinearity with the Variance Inflation Factor (VIF):

```
library(car)
```

```
## Loading required package: carData
```

```
vif(lm1.1)
```

```
##      Temp.Tank      Temp.Gas      Vapor.Tank Vapor.Dispensed
##      12.997379      4.720998      71.301491      61.932647
```

A vif above 5 to 10 indicates problems with multicollinearity. According to this guideline all variables but Temp.Gas have too high vif factors and therewith problems with multicollinearity. Vapor.Tank is affected the most.

Exercise 1.b)

Performing a variable selection using the AIC stepwise from the model fitted in Exercise 1.a):

```
step(lm1.1)
```

```
## Start:  AIC=68.84
## Y ~ Temp.Tank + Temp.Gas + Vapor.Tank + Vapor.Dispensed
##
```

```
##           Df Sum of Sq   RSS   AIC
## - Temp.Tank      1      0.743 201.97 66.956
## <none>                201.23 68.838
## - Vapor.Tank      1     17.113 218.34 69.450
## - Temp.Gas        1     75.698 276.93 77.056
## - Vapor.Dispensed 1     78.090 279.32 77.332
##
## Step:  AIC=66.96
## Y ~ Temp.Gas + Vapor.Tank + Vapor.Dispensed
##
##           Df Sum of Sq   RSS   AIC
## <none>                201.97 66.956
## - Vapor.Tank      1     36.416 238.39 70.261
## - Temp.Gas        1     78.831 280.80 75.501
## - Vapor.Dispensed 1     91.850 293.82 76.952

##
## Call:
## lm(formula = Y ~ Temp.Gas + Vapor.Tank + Vapor.Dispensed, data = df)
##
## Coefficients:
##      (Intercept)      Temp.Gas      Vapor.Tank  Vapor.Dispensed
##           1.0655           0.2091           -4.8882            9.2480
```

The best model with the stepwise variable selection from the model in Exercise 1.a) is $Y \sim \text{Temp.Gas} + \text{Vapor.Tank} + \text{Vapor.Dispensed}$. Temp.Tank gets not included. This would be due to multicollinearity with other variables.

Exercise 1.c)

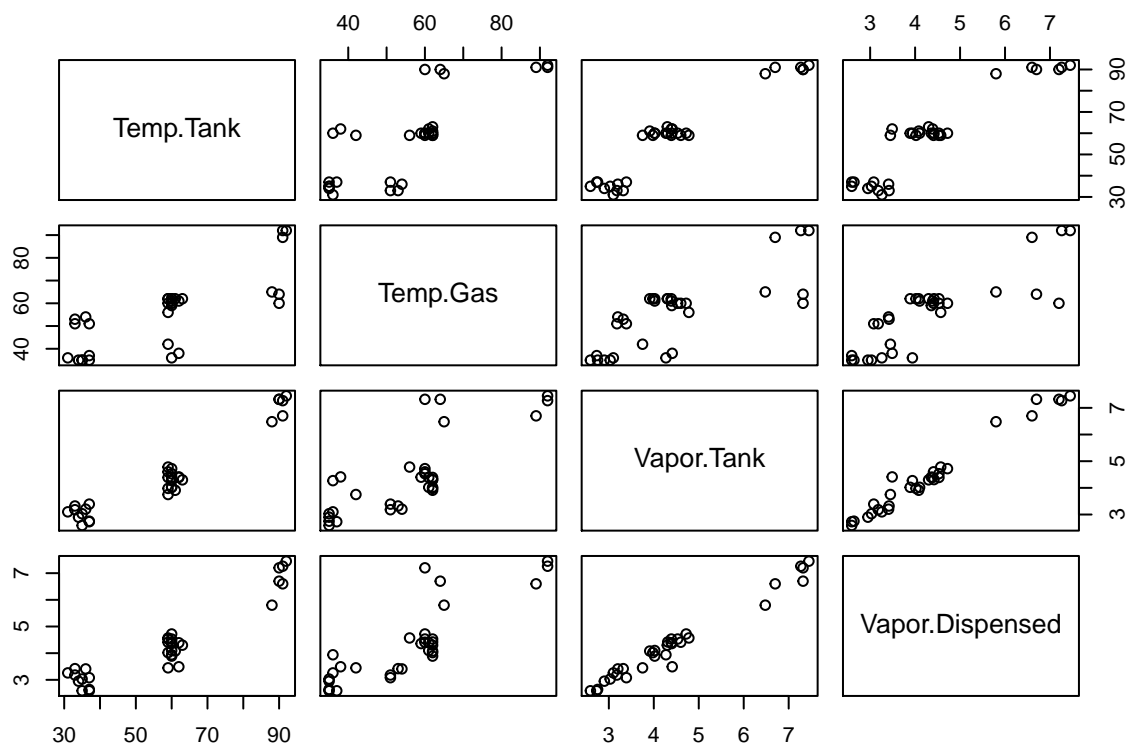
Did we already remedy the initially found multicollinearity with the stepwise variable selection? We can check by performing a vif on the newly found model.

```
lm1.2 <- lm(Y ~ Temp.Gas + Vapor.Tank + Vapor.Dispensed, data = df)
vif(lm1.2)
```

```
##      Temp.Gas      Vapor.Tank  Vapor.Dispensed
##      4.255787      42.899447      55.907555
```

No, Vapor.Tank and Vapor.Dispensed have still vif values from above 5 to 10. Which ones are correlated the most?

```
pairs(df[, -5])
```



Vapor.Tank and Vapor.Dispensed seem to be correlated the most. So we try transformations of the variables by replacing them by the mean and the difference.

```
df2 <- data.frame(diffVapor = df$Vapor.Tank - df$Vapor.Dispensed,
                  meanVapor = (df$Vapor.Tank + df$Vapor.Dispensed) / 2)

df3 <- cbind(df2, Temp.Tank = df$Temp.Tank, Temp.Gas = df$Temp.Gas, Y = df$Y)

head(df3)
```

```
##   diffVapor meanVapor Temp.Tank Temp.Gas  Y
## 1    -0.10    3.370      33      53 29
## 2    -0.16    3.180      31      36 24
## 3     0.00    3.180      33      51 26
## 4     0.31    3.235      37      51 22
## 5    -0.21    3.305      36      54 27
## 6     0.00    3.030      35      35 21
```

With the newly created data.frame with the transformed variables one can now perform another stepwise variable selection.

```
lm1.3 <- lm(Y ~ ., data = df3)
step(lm1.3)
```

```
## Start:  AIC=68.84
```

```
## Y ~ diffVapor + meanVapor + Temp.Tank + Temp.Gas
##
##           Df Sum of Sq   RSS   AIC
## - Temp.Tank  1      0.743 201.97 66.956
## <none>                201.23 68.838
## - diffVapor  1     43.585 244.81 73.112
## - Temp.Gas   1     75.698 276.93 77.056
## - meanVapor  1    114.810 316.04 81.284
##
## Step:  AIC=66.96
## Y ~ diffVapor + meanVapor + Temp.Gas
##
##           Df Sum of Sq   RSS   AIC
## <none>                201.97 66.956
## - diffVapor  1     64.398 266.37 73.813
## - Temp.Gas   1     78.831 280.80 75.501
## - meanVapor  1    265.710 467.68 91.826

##
## Call:
## lm(formula = Y ~ diffVapor + meanVapor + Temp.Gas, data = df3)
##
## Coefficients:
## (Intercept)    diffVapor    meanVapor    Temp.Gas
##      1.0655      -7.0681       4.3597       0.2091
```

This is the same model as found in Exercise 1.b) but with the transformed variables. Now one can check if the problems with multicollinearity still persists.

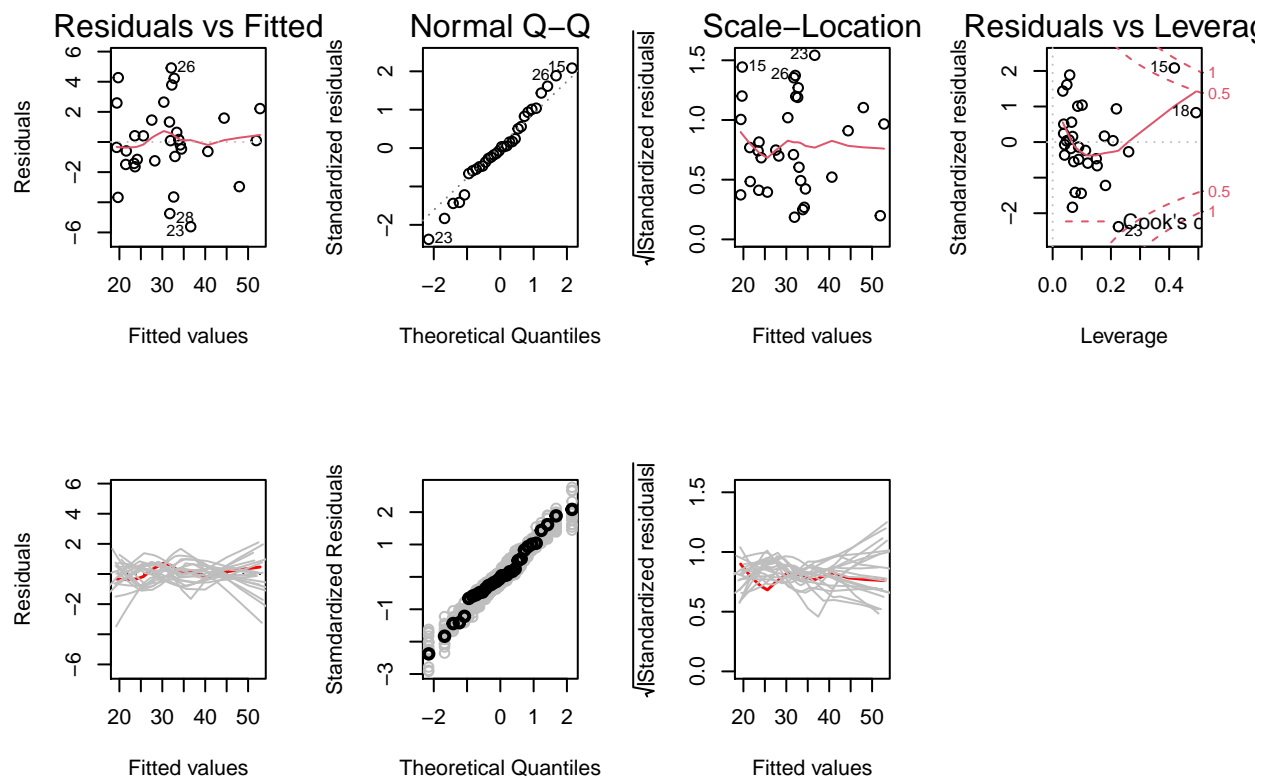
```
lm1.4 <- lm(Y ~ diffVapor + meanVapor + Temp.Gas, data = df3)
vif(lm1.4)
```

```
## diffVapor meanVapor  Temp.Gas
##  1.538981  4.450470  4.255787
```

All vif values are lower than 5 and therewith the problem with multicollinearity does not persist.

How looks the residual and sensitivity analysis?

```
par(mfrow = c(2, 4))
plot(lm1.4)
plot.lmSim(lm1.4, SEED = 1)
```



leverage points > 0.25

There is no evidence that any of the assumptions is violated.

Exercise 2

```
path <- file.path('Datasets', 'jet.dat')
df <- read.table(path, header=TRUE)

summary(df)
```

Dataset loading and sanity check:

##	Y	x1	x2	x3	x4
##	Min. :3045	Min. :1388	Min. :17780	Min. :28675	Min. :133.0
##	1st Qu.:3518	1st Qu.:1608	1st Qu.:18880	1st Qu.:29302	1st Qu.:155.2
##	Median :3977	Median :1850	Median :19765	Median :29745	Median :179.0
##	Mean :3904	Mean :1810	Mean :19495	Mean :29606	Mean :174.5
##	3rd Qu.:4332	3rd Qu.:2024	3rd Qu.:20286	3rd Qu.:29960	3rd Qu.:193.5
##	Max. :4833	Max. :2239	Max. :20740	Max. :30250	Max. :216.0
##	x5	x6			
##	Min. :1522	Min. : 85.00			
##	1st Qu.:1592	1st Qu.: 97.00			


```
## Median :1668   Median : 99.00
## Mean   :1652   Mean    : 97.42
## 3rd Qu.:1710   3rd Qu.:100.00
## Max.   :1758   Max.    :102.00
```

```
dim(df)
```

```
## [1] 40 7
```

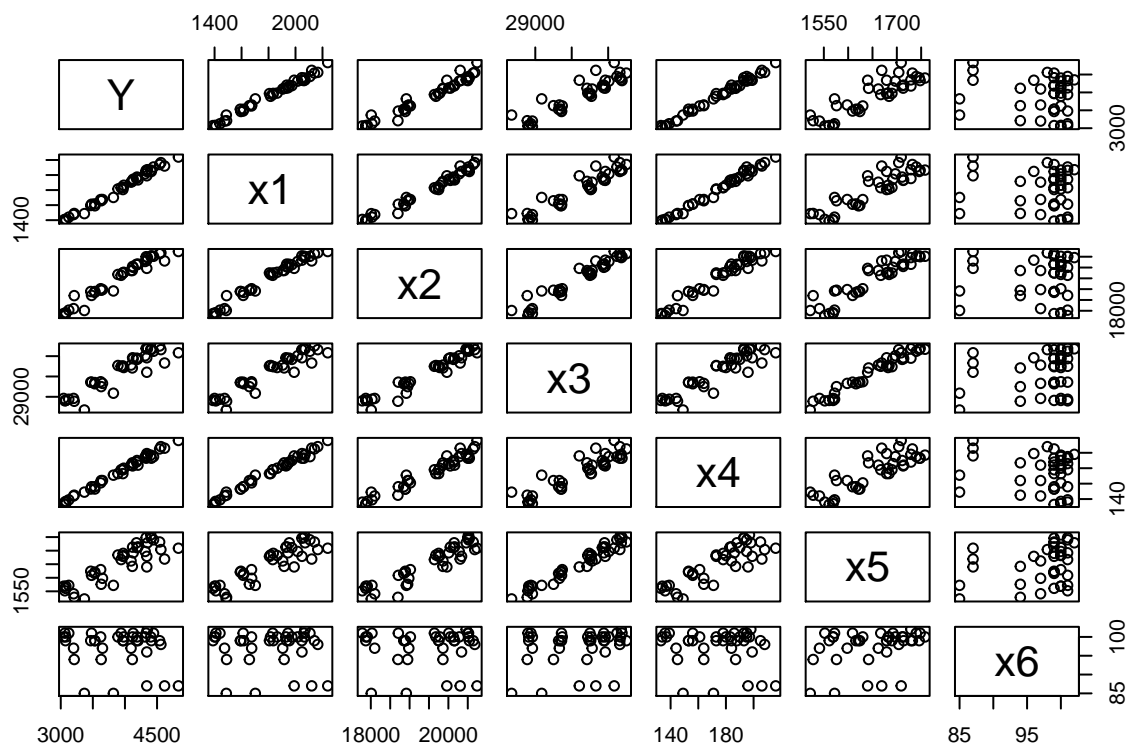
```
head(df)
```

```
##      Y    x1    x2    x3  x4    x5  x6
## 1 4540 2140 20640 30250 205 1732  99
## 2 4315 2016 20280 30010 195 1697 100
## 3 4095 1905 19860 29780 184 1662  97
## 4 3650 1675 18980 29330 164 1598  97
## 5 3200 1474 18100 28960 144 1541  97
## 6 4833 2239 20740 30083 216 1709  87
```

```
tail(df)
```

```
##      Y    x1    x2    x3  x4    x5  x6
## 35 3064 1410 17780 28900 136 1552 101
## 36 4402 2066 20520 30170 197 1758 100
## 37 4180 1954 20150 29950 188 1729  99
## 38 3973 1835 19750 29740 178 1690  99
## 39 3530 1616 18850 29320 156 1616  99
## 40 3080 1407 17910 28910 137 1569 100
```

```
plot(df)
```



There seems to be an issue with multicollinearity as can be seen in the pairsplot. But let's first transform the variables according to Tukey's first-aid transformations:

```
dft1 <- data.frame(lX1 = log(df$x1),
                   lX2 = log(df$x2),
                   lX3 = log(df$x3),
                   lX4 = log(df$x4),
                   x5 = df$x5,
                   x6 = df$x6,
                   lY = log(df$Y))
```

```
head(dft1)
```

```
##      lX1      lX2      lX3      lX4  x5  x6      lY
## 1 7.668561 9.934986 10.31725 5.323010 1732 99 8.420682
## 2 7.608871 9.917390 10.30929 5.273000 1697 100 8.369853
## 3 7.552237 9.896463 10.30159 5.214936 1662 97 8.317522
## 4 7.423568 9.851141 10.28637 5.099866 1598 97 8.202482
## 5 7.295735 9.803667 10.27367 4.969813 1541 97 8.070906
## 6 7.713785 9.939819 10.31172 5.375278 1709 87 8.483223
```

-> x5 and x6 are not transformed because temperature can be negative (do not transform variables which could be negative numbers according to Tukey's first-aid transformations).

With the transformed dataset one can now start modeling a linear model. Let's start with a full model which includes all the explanatory variables.

```
lm2.1 <- lm(lY ~ ., data = dft1)
summary(lm2.1)
```

```
##
## Call:
## lm(formula = lY ~ ., data = dft1)
##
## Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.0125101	-0.0049270	-0.0006753	0.0047059	0.0157080

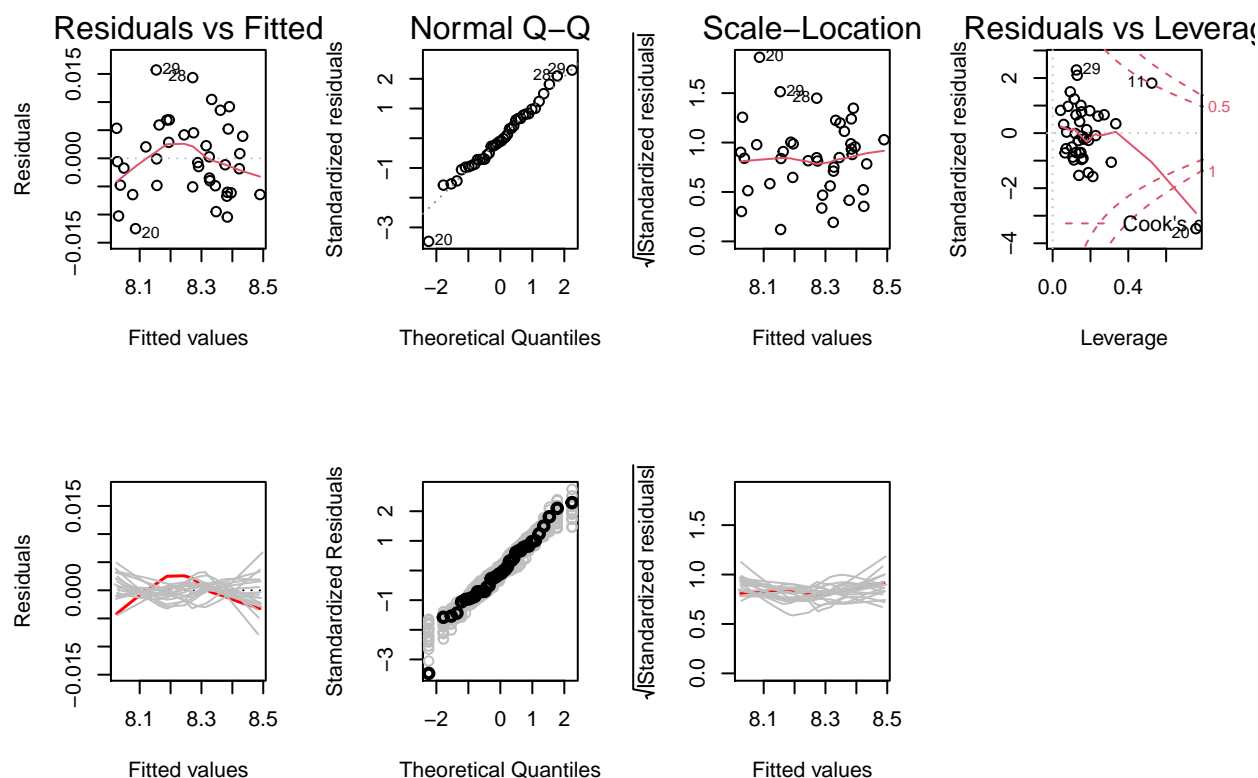
```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-9.704e+00	8.614e+00	-1.127	0.2681
lX1	4.090e-01	1.766e-01	2.316	0.0269 *
lX2	-6.751e-02	2.043e-01	-0.330	0.7431
lX3	1.364e+00	9.452e-01	1.443	0.1584
lX4	2.897e-01	1.389e-01	2.086	0.0448 *
x5	2.494e-04	9.415e-05	2.648	0.0123 *
x6	-3.925e-03	7.019e-04	-5.592	3.21e-06 ***

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007329 on 33 degrees of freedom
## Multiple R-squared:  0.9974, Adjusted R-squared:  0.997
## F-statistic: 2143 on 6 and 33 DF, p-value: < 2.2e-16
```

The R^2 looks actually pretty good. But not all the variables seem to be relevant and we have to do a residual and sensitivity analysis first.

```
par(mfrow=c(2,4))
plot(lm2.1)
plot.lmSim(lm2.1, SEED = 1)
```



Observation i=20 is an outlier. We remove it and build a new model without it and analyze it.

```
ind <- 20
lm2.2 <- lm(lY ~ ., data = dft1, subset = -ind)
summary(lm2.2)
```

```
##
## Call:
## lm(formula = lY ~ ., data = dft1, subset = -ind)
##
## Residuals:
```

	Min	1Q	Median	3Q	Max
##	-0.0104560	-0.0038227	-0.0001954	0.0043872	0.0092726

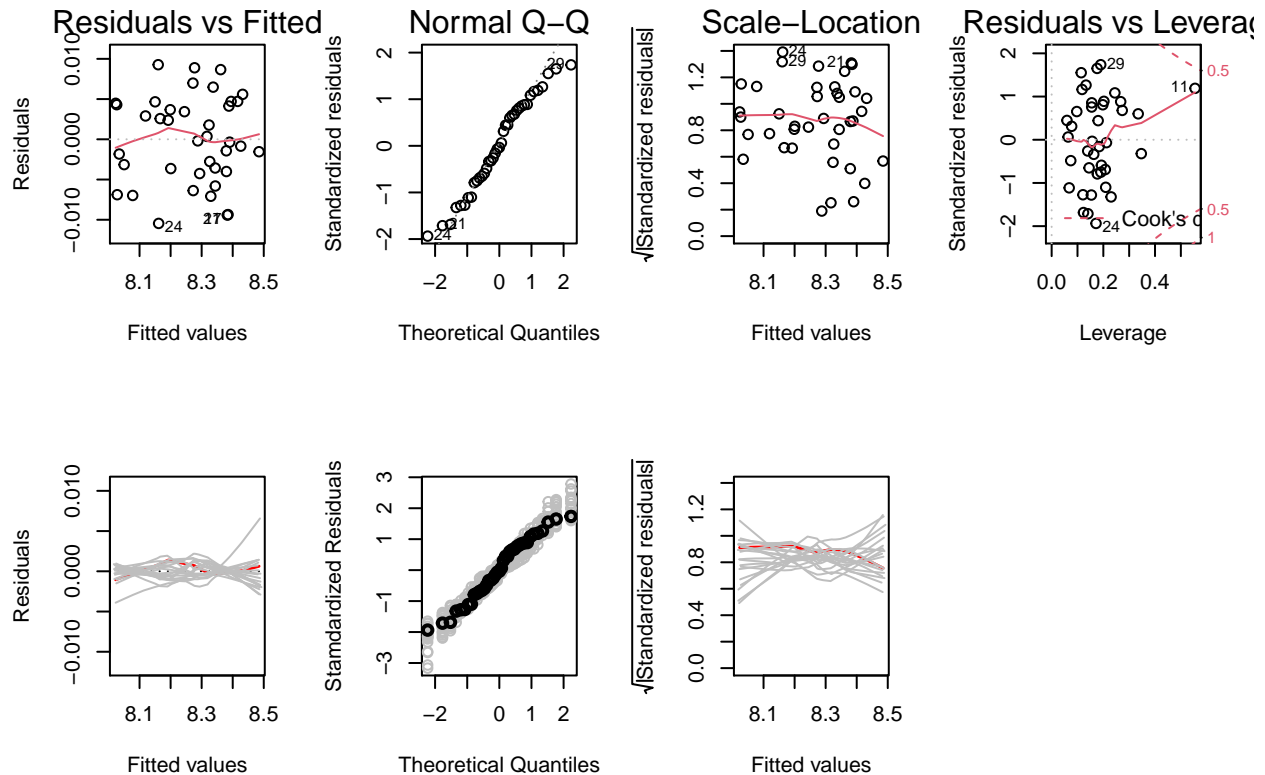
```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	-3.117e+00	7.138e+00	-0.437	0.66527
## lX1	4.957e-01	1.444e-01	3.434	0.00166 **
## lX2	1.018e+00	3.024e-01	3.367	0.00199 **
## lX3	-2.428e-01	8.519e-01	-0.285	0.77746
## lX4	5.443e-02	1.251e-01	0.435	0.66643
## x5	9.169e-05	8.461e-05	1.084	0.28658
## x6	-3.281e-03	5.876e-04	-5.585	3.62e-06 ***

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.005931 on 32 degrees of freedom
## Multiple R-squared:  0.9983, Adjusted R-squared:  0.998
## F-statistic: 3104 on 6 and 32 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,4))
plot(lm2.2)
plot.lmSim(lm2.2, SEED = 1)
```



There is no evidence that any of the assumptions is violated and no outlier is visible. Lets now perform a variable selection with the step() function.

```
step(lm2.2, scope = list(upper =~ 1X1 + 1X2 + 1X3 + 1X4 + x5 + x6, lower =~ 1))
```

```
## Start: AIC=-393.66
## 1Y ~ 1X1 + 1X2 + 1X3 + 1X4 + x5 + x6
##
##      Df Sum of Sq    RSS    AIC
## - 1X3   1 0.00000286 0.0011285 -395.57
## - 1X4   1 0.00000666 0.0011323 -395.43
## - x5    1 0.00004132 0.0011670 -394.26
## <none>          0.0011257 -393.66
## - 1X2   1 0.00039868 0.0015244 -383.84
## - 1X1   1 0.00041485 0.0015405 -383.43
## - x6    1 0.00109718 0.0022229 -369.13
##
```

```

## Step: AIC=-395.57
## 1Y ~ 1X1 + 1X2 + 1X4 + x5 + x6
##
##      Df Sum of Sq      RSS      AIC
## - 1X4   1 0.00000992 0.0011385 -397.22
## - x5    1 0.00003846 0.0011670 -396.26
## <none>                0.0011285 -395.57
## + 1X3   1 0.00000286 0.0011257 -393.66
## - 1X1   1 0.00046212 0.0015907 -384.18
## - 1X2   1 0.00050646 0.0016350 -383.11
## - x6    1 0.00283395 0.0039625 -348.58
##
## Step: AIC=-397.22
## 1Y ~ 1X1 + 1X2 + x5 + x6
##
##      Df Sum of Sq      RSS      AIC
## - x5    1 0.0000286 0.0011671 -398.25
## <none>                0.0011385 -397.22
## + 1X4   1 0.0000099 0.0011285 -395.57
## + 1X3   1 0.0000061 0.0011323 -395.43
## - 1X2   1 0.0005920 0.0017305 -382.89
## - 1X1   1 0.0019052 0.0030437 -360.87
## - x6    1 0.0032661 0.0044046 -346.46
##
## Step: AIC=-398.25
## 1Y ~ 1X1 + 1X2 + x6
##
##      Df Sum of Sq      RSS      AIC
## <none>                0.0011671 -398.25
## + x5    1 0.0000286 0.0011385 -397.22
## + 1X4   1 0.0000001 0.0011670 -396.26
## + 1X3   1 0.0000000 0.0011671 -396.26
## - 1X2   1 0.0010830 0.0022501 -374.65
## - 1X1   1 0.0019053 0.0030725 -362.50
## - x6    1 0.0046090 0.0057761 -337.89
##
##
## Call:
## lm(formula = 1Y ~ 1X1 + 1X2 + x6, data = dft1, subset = -ind)
##
## Coefficients:
## (Intercept)      1X1      1X2      x6
##   -6.524816    0.521784    1.133755   -0.003275

```

```

step(lm(1Y ~ 1, data = dft1[-ind,]),
     direction = 'both',
     scope = list(upper =~ 1X1 + 1X2 + 1X3 + 1X4 + x5 + x6,
                  lower =~ 1))

```

```

## Start: AIC=-157.3
## 1Y ~ 1
##
##      Df Sum of Sq      RSS      AIC

```

```

## + 1X4    1    0.65059 0.00578 -339.85
## + 1X1    1    0.65016 0.00621 -337.08
## + 1X2    1    0.63390 0.02247 -286.91
## + 1X3    1    0.56517 0.09120 -232.27
## + x5     1    0.49540 0.16097 -210.11
## <none>                0.65637 -157.30
## + x6     1    0.01927 0.63710 -156.46
##
## Step:  AIC=-339.85
## 1Y ~ 1X4
##
##      Df Sum of Sq    RSS    AIC
## + 1X1    1    0.00166 0.00412 -351.09
## + 1X2    1    0.00146 0.00432 -349.19
## + x5     1    0.00115 0.00464 -346.46
## + 1X3    1    0.00062 0.00516 -342.27
## <none>                0.00578 -339.85
## + x6     1    0.00000 0.00578 -337.88
## - 1X4    1    0.65059 0.65637 -157.30
##
## Step:  AIC=-351.09
## 1Y ~ 1X4 + 1X1
##
##      Df Sum of Sq    RSS    AIC
## + x6     1 0.00187618 0.0022413 -372.81
## + 1X3    1 0.00054043 0.0035770 -354.57
## <none>                0.0041174 -351.09
## + 1X2    1 0.00004121 0.0040762 -349.48
## + x5     1 0.00000879 0.0041087 -349.17
## - 1X1    1 0.00166441 0.0057819 -339.85
## - 1X4    1 0.00208945 0.0062069 -337.08
##
## Step:  AIC=-372.81
## 1Y ~ 1X4 + 1X1 + x6
##
##      Df Sum of Sq    RSS    AIC
## + 1X2    1 0.0010743 0.0011670 -396.26
## + x5     1 0.0006063 0.0016350 -383.11
## + 1X3    1 0.0005090 0.0017323 -380.85
## - 1X4    1 0.0000088 0.0022501 -374.65
## <none>                0.0022413 -372.81
## - x6     1 0.0018762 0.0041174 -351.09
## - 1X1    1 0.0035363 0.0057776 -337.88
##
## Step:  AIC=-396.26
## 1Y ~ 1X4 + 1X1 + x6 + 1X2
##
##      Df Sum of Sq    RSS    AIC
## - 1X4    1 0.00000011 0.0011671 -398.25
## <none>                0.0011670 -396.26
## + x5     1 0.00003846 0.0011285 -395.57
## + 1X3    1 0.00000000 0.0011670 -394.26
## - 1X1    1 0.00063695 0.0018040 -381.27
## - 1X2    1 0.00107426 0.0022413 -372.81

```

```
## - x6      1 0.00290924 0.0040762 -349.48
##
## Step: AIC=-398.25
## 1Y ~ 1X1 + x6 + 1X2
##
##           Df Sum of Sq      RSS      AIC
## <none>             0.0011671 -398.25
## + x5      1 0.0000286 0.0011385 -397.22
## + 1X4     1 0.0000001 0.0011670 -396.26
## + 1X3     1 0.0000000 0.0011671 -396.26
## - 1X2     1 0.0010830 0.0022501 -374.65
## - 1X1     1 0.0019053 0.0030725 -362.50
## - x6      1 0.0046090 0.0057761 -337.89

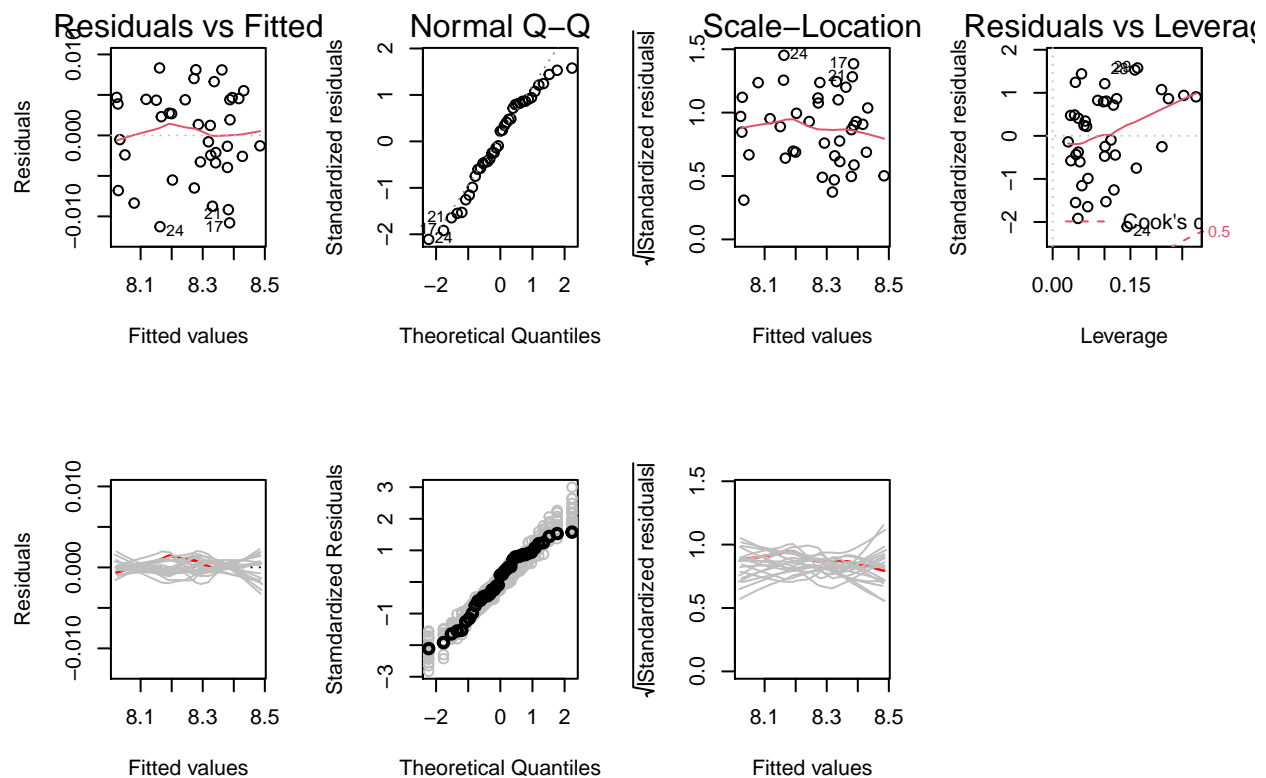
##
## Call:
## lm(formula = 1Y ~ 1X1 + x6 + 1X2, data = dft1[-ind, ])
##
## Coefficients:
## (Intercept)      1X1          x6          1X2
##   -6.524816    0.521784   -0.003275    1.133755
```

In both cases the final suggested model with the lowest AIC is $1Y = 1X1 + 1X2 + x6$ without the observation $i=20$.

So lets investigate this model with a residual and sensitivity analysis.

```
lm2.3 <- lm(1Y ~ 1X1 + 1X2 + x6, data = dft1[-20,])

par(mfrow = c(2,4))
plot(lm2.3)
plot.lmSim(lm2.3, SEED = 1)
```

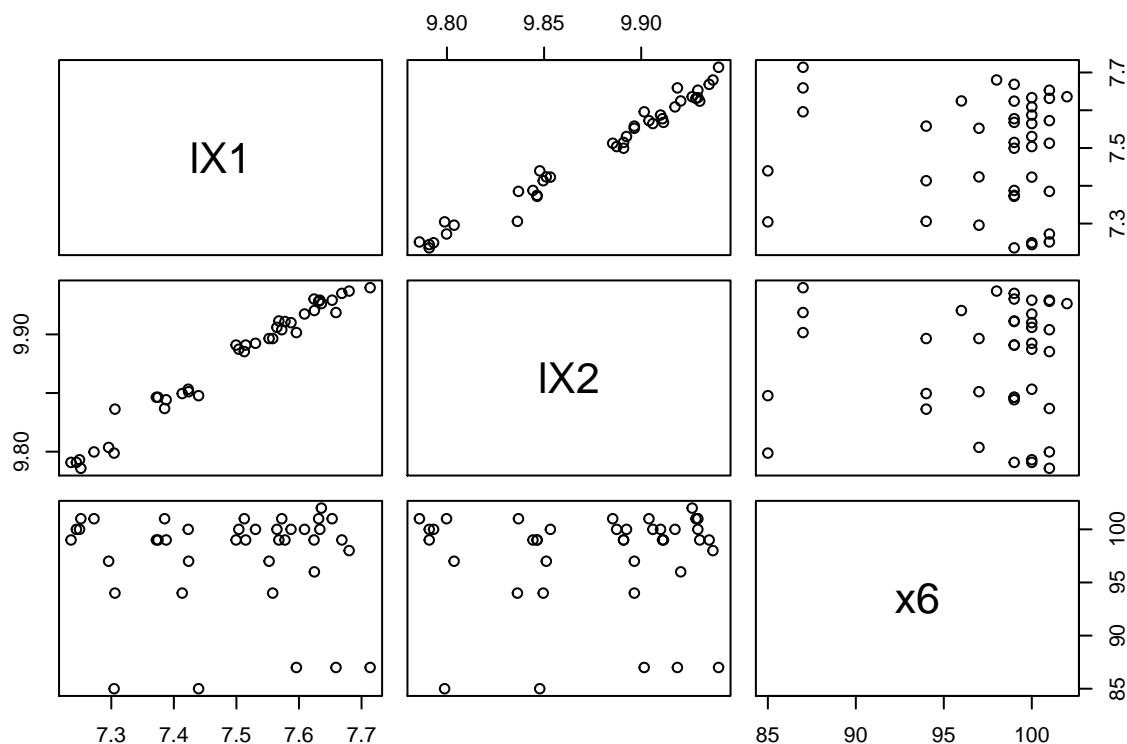
There is no evidence of any violation of the model assumptions. So let's now investigate the multicollinearity with the Variance Inflation Factor (vif)

```
library(car)
vif(lm2.3)
```

```
##          lX1          lX2          x6
## 109.721932 108.742539   1.991382
```

There seems to be a problem with multicollinearity for the variables lX1 and lX2. Let's look at it:

```
plot(dft1[, c('lX1', 'lX2', 'x6')])
```



Indeed, IX1 and IX2 are highly correlated. Let's transform them to the mean and their difference and check if the plot looks better:

```
dft1$dlRoSp <- dft1$IX2 - dft1$IX1
dft1$mlRoSp <- (dft1$IX1 + dft1$IX2)/2
head(dft1)
```

```
##      lX1      lX2      lX3      lX4  x5  x6      lY  dlRoSp  mlRoSp
## 1 7.668561 9.934986 10.31725 5.323010 1732 99 8.420682 2.266425 8.801774
## 2 7.608871 9.917390 10.30929 5.273000 1697 100 8.369853 2.308520 8.763131
## 3 7.552237 9.896463 10.30159 5.214936 1662 97 8.317522 2.344226 8.724350
## 4 7.423568 9.851141 10.28637 5.099866 1598 97 8.202482 2.427573 8.637355
## 5 7.295735 9.803667 10.27367 4.969813 1541 97 8.070906 2.507932 8.549701
## 6 7.713785 9.939819 10.31172 5.375278 1709 87 8.483223 2.226035 8.826802
```

```
lm2.4 <- lm(lY ~ dlRoSp + mlRoSp + x6, data = dft1[-20,])
summary(lm2.4)
```

```
##
## Call:
## lm(formula = lY ~ dlRoSp + mlRoSp + x6, data = dft1[-20, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.011274 -0.003329  0.001228  0.004430  0.008328
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.5248156  1.4320310  -4.556 6.08e-05 ***
## dlRoSp       0.3059856  0.1338667   2.286  0.0284 *
## mlRoSp       1.6555392  0.1304000  12.696 1.17e-14 ***
## x6           -0.0032750  0.0002786 -11.757 1.04e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005775 on 35 degrees of freedom
## Multiple R-squared:  0.9982, Adjusted R-squared:  0.9981
## F-statistic: 6550 on 3 and 35 DF,  p-value: < 2.2e-16
```

```
vif(lm2.4)
```

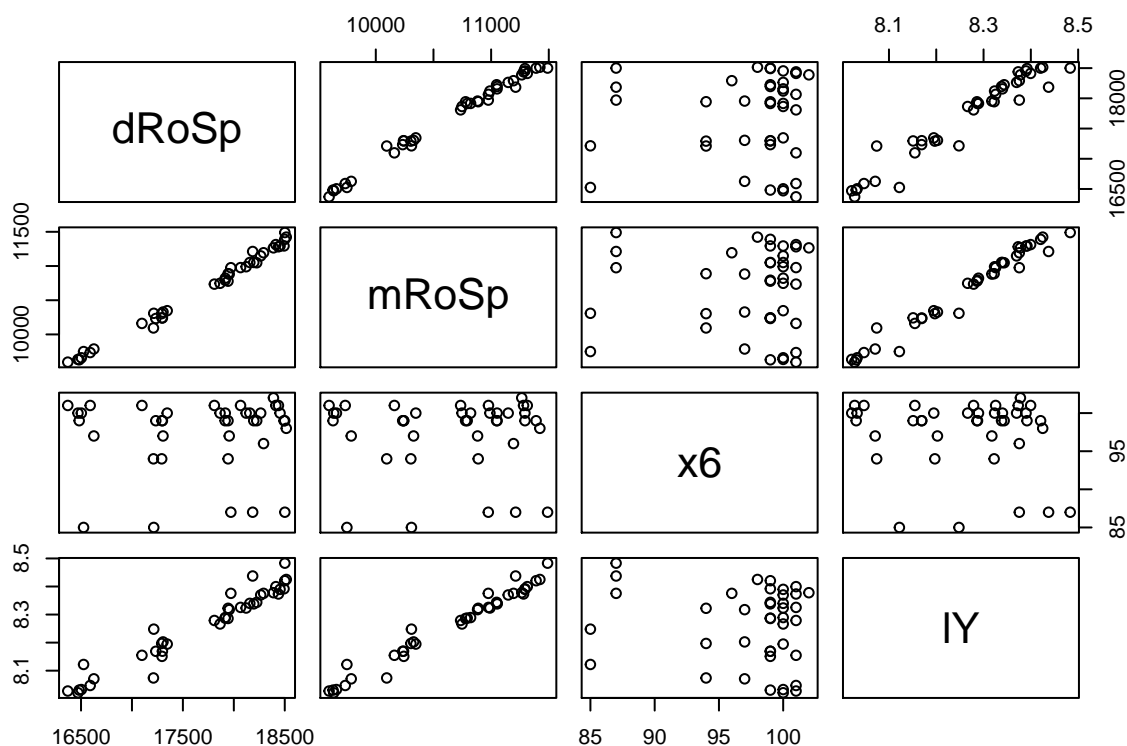
```
##      dlRoSp      mlRoSp      x6
## 179.409858 176.580766  1.991382
```

This does not get any better: Still looks like a very strong correlation. So one could try the transformation with the untransformed variables.

```
dft2 <- data.frame(dRoSp = df$x2 - df$x1,
                  mRoSp = (df$x1 + df$x2)/2,
                  x6 = df$x6,
                  lY = dft1$lY)
head(dft2)
```

```
##   dRoSp  mRoSp  x6      lY
## 1 18500 11390.0  99 8.420682
## 2 18264 11148.0 100 8.369853
## 3 17955 10882.5  97 8.317522
## 4 17305 10327.5  97 8.202482
## 5 16626  9787.0  97 8.070906
## 6 18501 11489.5  87 8.483223
```

```
plot(dft2)
```



```
lm2.5 <- lm(lY ~ ., data = dft2[-20,])
summary(lm2.5)
```

```
##
## Call:
## lm(formula = lY ~ ., data = dft2[-20, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.014169 -0.004725  0.000420  0.005255  0.012592
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.499e+00  1.088e-01  59.737  < 2e-16 ***
## dRoSp       -3.152e-05  2.365e-05  -1.333   0.191
## mRoSp        2.539e-04  2.773e-05   9.158 8.04e-11 ***
## x6          -3.931e-03  3.137e-04 -12.532 1.70e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00709 on 35 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9971
## F-statistic: 4340 on 3 and 35 DF, p-value: < 2.2e-16
```

```
vif(lm2.5)
```

```
##          dRoSp          mRoSp          x6
## 206.481767 206.281338   1.674926
```

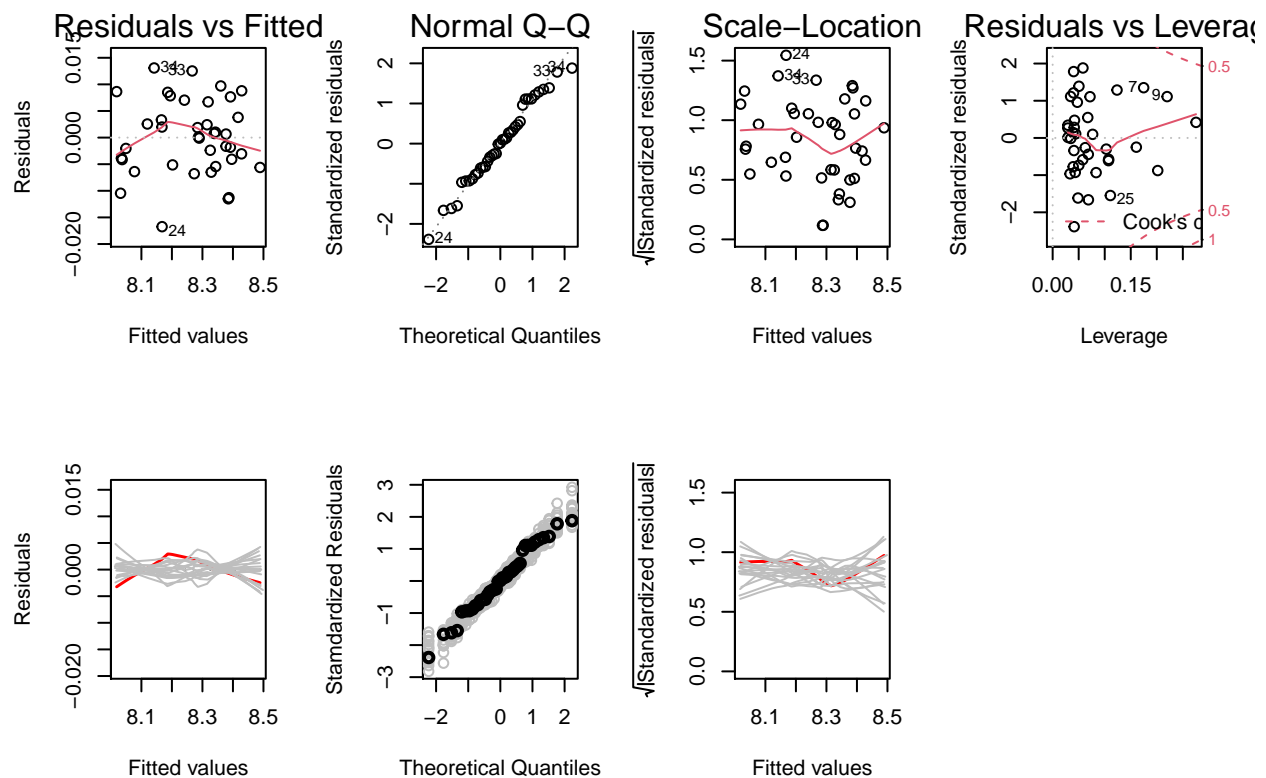
Still problems with the multicollinearity. Because in this model dRoSp is not significant, one can drop this variable.

```
lm2.6 <- lm(lY ~ mRoSp + x6, data = dft2[-20,])
summary(lm2.6)
```

```
##
## Call:
## lm(formula = lY ~ mRoSp + x6, data = dft2[-20, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.0167433 -0.0046339  0.0000973  0.0052560  0.0130837
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.360e+00  3.203e-02  198.58  <2e-16 ***
## mRoSp        2.171e-04  1.952e-06  111.22  <2e-16 ***
## x6          -4.196e-03  2.450e-04  -17.13  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007166 on 36 degrees of freedom
## Multiple R-squared:  0.9972, Adjusted R-squared:  0.997
## F-statistic: 6372 on 2 and 36 DF, p-value: < 2.2e-16
```

Like that all the variables are significant and the R^2 is still 0.9972 the model performance and suitability looks still very good. How about the residual and sensitivity analysis?

```
par(mfrow=c(2,4))
plot(lm2.6)
plot.lmSim(lm2.6, SEED = 1)
```



No model assumptions are violated. What about the multicollinearity problem?

```
vif(lm2.6)
```

```
##      mRoSp      x6
## 1.000405 1.000405
```

Multicollinearity seems also not to be a problem anymore. The model fits the data well like that.

Exercise 3

```
path <- file.path('Datasets', 'windmill.dat')
df <- read.table(path, header=TRUE)

summary(df)
```

Dataset loading and sanity check:

```
##      velocity      DC.output
## Min.   : 5.482    Min.   :0.123
## 1st Qu.: 8.838    1st Qu.:1.144
## Median :13.424    Median :1.800
```

```
## Mean :13.720 Mean :1.610
## 3rd Qu.:18.235 3rd Qu.:2.166
## Max. :22.822 Max. :2.386
```

```
dim(df)
```

```
## [1] 25 2
```

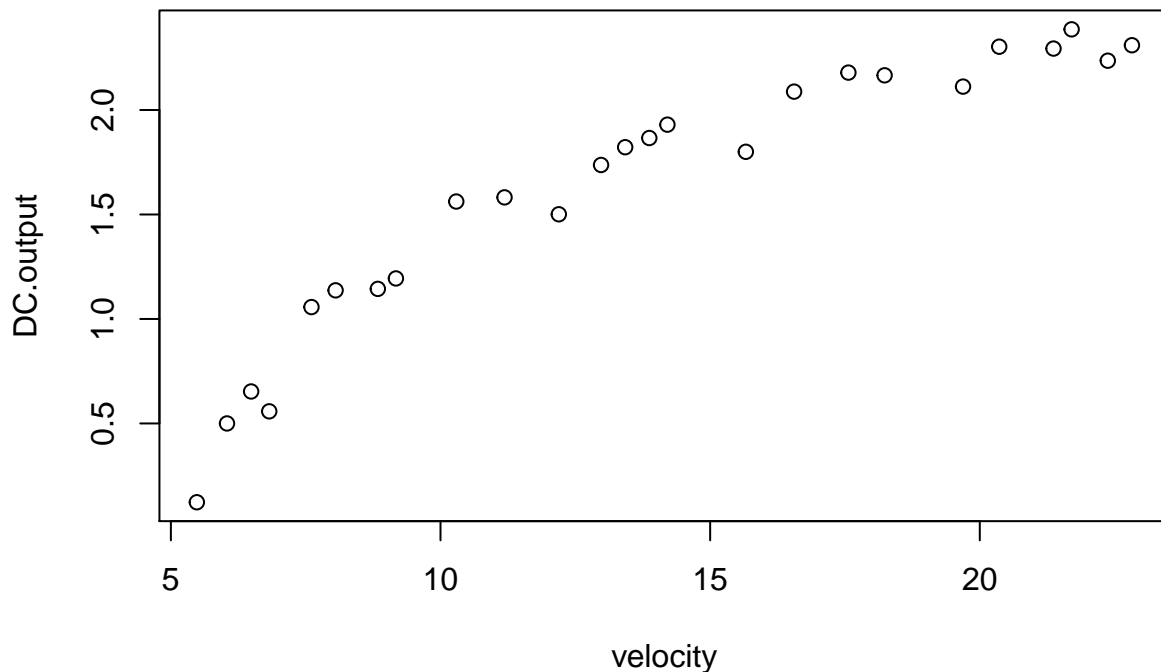
```
head(df)
```

```
## velocity DC.output
## 1 11.187073 1.582
## 2 13.424487 1.822
## 3 7.607209 1.057
## 4 6.041019 0.500
## 5 22.374145 2.236
## 6 21.702921 2.386
```

```
tail(df)
```

```
## velocity DC.output
## 20 12.193909 1.501
## 21 20.360472 2.303
## 22 22.821628 2.310
## 23 9.173400 1.194
## 24 8.837787 1.144
## 25 5.481666 0.123
```

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



```
df$tVel <- (1 / df$velocity)
head(df)
```

```
##   velocity DC.output      tVel
## 1 11.187073    1.582 0.08938889
## 2 13.424487    1.822 0.07449074
## 3  7.607209    1.057 0.13145426
## 4  6.041019    0.500 0.16553499
## 5 22.374145    2.236 0.04469445
## 6 21.702921    2.386 0.04607675
```

The model was already used in the worksheet of week 1, that is why it is not investigated here but instead used for predictions of DC.output for wind velocities of one and ten meter per second. So we fit first the known model and used it then for prediction.

```
lm3.1 <- lm(DC.output ~ tVel, data = df)

dfPreds <- data.frame(tVel = c(1/1,
                               1/10))
predict(lm3.1, newdata = dfPreds, interval = 'prediction', level = 0.95)
```

```
##           fit          lwr          upr
## 1 -12.536597 -13.430108 -11.643086
## 2  1.427314  1.228331  1.626298
```


The prediction of the first line (1 meter per second wind velocity) is not usable. One has always to investigate the prediction(-range) and make sure that they are plausible!

Exercise 4

```
path <- file.path('Datasets', 'NPSCosts.dat')
df <- read.table(path, header=TRUE)

summary(df)
```

Dataset loading and sanity check:

```
##      cost      date      t1      t2
## Min.   :207.5   Min.   :67.17   Min.    : 7.00   Min.    :44.00
## 1st Qu.:310.3   1st Qu.:67.90   1st Qu.:11.75   1st Qu.:56.50
## Median :448.1   Median :68.42   Median :13.00   Median :62.50
## Mean   :461.6   Mean   :68.58   Mean    :13.75   Mean    :62.38
## 3rd Qu.:612.0   3rd Qu.:68.92   3rd Qu.:15.25   3rd Qu.:70.25
## Max.   :881.2   Max.    :71.08   Max.    :22.00   Max.    :85.00
##      cap      pr      ne      ct
## Min.    : 457.0   Min.    :0.0000   Min.    :0.00   Min.    :0.0000
## 1st Qu.: 745.0   1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:0.0000
## Median : 822.0   Median :0.0000   Median :0.00   Median :0.0000
## Mean    : 825.4   Mean    :0.3125   Mean    :0.25   Mean    :0.4062
## 3rd Qu.: 947.2   3rd Qu.:1.0000   3rd Qu.:0.25   3rd Qu.:1.0000
## Max.    :1130.0   Max.    :1.0000   Max.    :1.00   Max.    :1.0000
##      bw      cum.n      pt
## Min.    :0.0000   Min.    : 1.000   Min.    :0.0000
## 1st Qu.:0.0000   1st Qu.: 3.000   1st Qu.:0.0000
## Median :0.0000   Median : 7.500   Median :0.0000
## Mean    :0.1875   Mean    : 8.531   Mean    :0.1875
## 3rd Qu.:0.0000   3rd Qu.:12.500   3rd Qu.:0.0000
## Max.    :1.0000   Max.    :21.000   Max.    :1.0000
```

```
dim(df)
```

```
## [1] 32 11
```

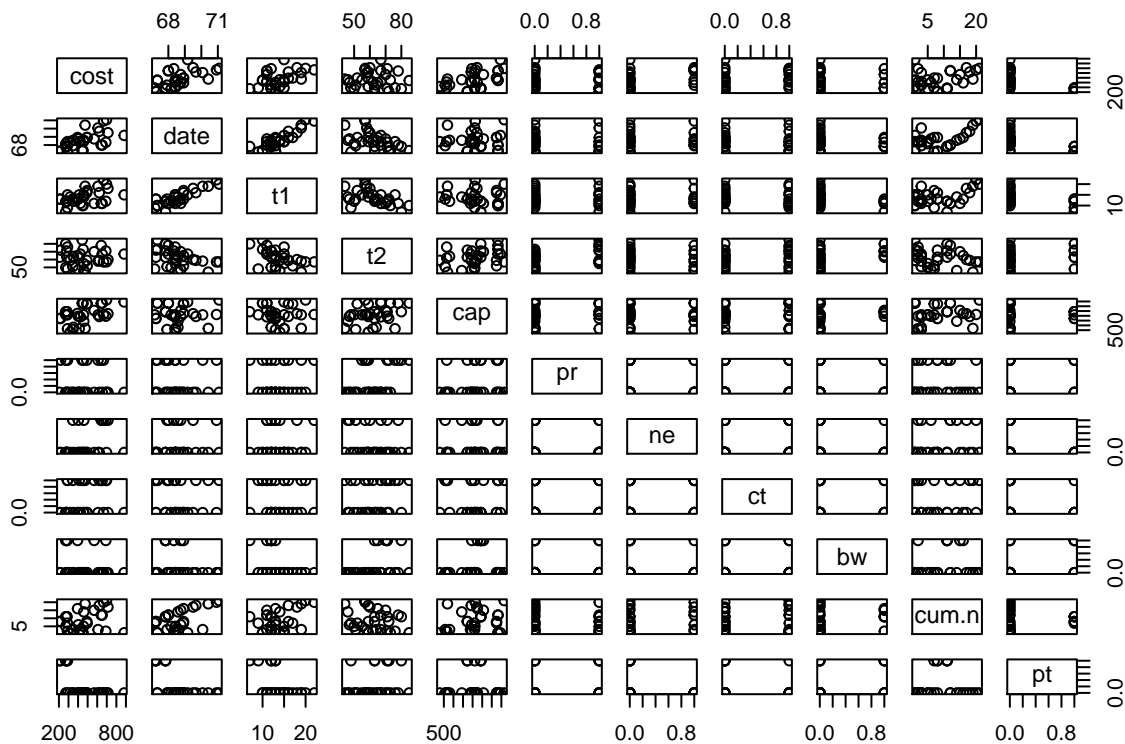
```
head(df)
```

```
##      cost date t1 t2 cap pr ne ct bw cum.n pt
## 1 460.05 68.58 14 46 687 0 1 0 0 14 0
## 2 452.99 67.33 10 73 1065 0 0 1 0 1 0
## 3 443.22 67.33 10 85 1065 1 0 1 0 1 0
## 4 652.32 68.00 11 67 1065 0 1 1 0 12 0
## 5 642.23 68.00 11 78 1065 1 1 1 0 12 0
## 6 345.39 67.92 13 51 514 0 1 1 0 3 0
```

```
tail(df)
```

```
##      cost  date t1 t2 cap pr ne ct bw cum.n pt
## 27 207.51 67.25 13 63 745 0 0 0 0      8 1
## 28 288.48 67.17  9 48 821 0 0 1 0      7 1
## 29 284.88 67.83 12 63 886 0 0 0 1     11 1
## 30 280.36 67.83 12 71 886 1 0 0 1     11 1
## 31 217.38 67.25 13 72 745 1 0 0 0      8 1
## 32 270.71 67.83  7 80 886 1 0 0 1     11 1
```

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



The dataset and the model were already partially investigated in Exercise 3 from week 1. We take off where we ended there.

Exercise 4.a)

First a variable selection gets performed on the model built in Exercise 3 week 1 (including the performed variable transformations).

```
df$lCost <- log(df$cost)
df$lCap <- log(df$cap)
df$sCum.n <- sqrt(df$cum.n)
```

```
mod3.1 <- lm(lCost ~ date + t1 + t2 + lCap + pr + ne + ct + bw + sCum.n + pt,
             data = df)
summary(mod3.1)
```

```
##
## Call:
## lm(formula = lCost ~ date + t1 + t2 + lCap + pr + ne + ct + bw +
##     sCum.n + pt, data = df)
##
## Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.29896	-0.10332	0.02118	0.09019	0.26731

```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-13.875045	5.404826	-2.567	0.01796 *
date	0.219318	0.082426	2.661	0.01463 *
t1	0.006067	0.021990	0.276	0.78531
t2	0.005273	0.004564	1.155	0.26092
lCap	0.692542	0.137131	5.050	5.32e-05 ***
pr	-0.105307	0.082004	-1.284	0.21307
ne	0.254326	0.078075	3.257	0.00377 **
ct	0.122969	0.068386	1.798	0.08654 .
bw	0.029418	0.104469	0.282	0.78101
sCum.n	-0.069016	0.040985	-1.684	0.10700
pt	-0.229133	0.128059	-1.789	0.08800 .

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1666 on 21 degrees of freedom
## Multiple R-squared:  0.8684, Adjusted R-squared:  0.8057
## F-statistic: 13.85 on 10 and 21 DF, p-value: 3.983e-07
```

```
step(mod3.1, scope = list(upper =~ date + t1 + t2 + lCap + pr + ne + ct + bw + sCum.n + pt,
                           lower =~ 1))
```

```
## Start: AIC=-106.18
## lCost ~ date + t1 + t2 + lCap + pr + ne + ct + bw + sCum.n +
##     pt
##
```

	Df	Sum of Sq	RSS	AIC
- t1	1	0.00211	0.58498	-108.061
- bw	1	0.00220	0.58507	-108.057
- t2	1	0.03705	0.61991	-106.205
<none>			0.58286	-106.177
- pr	1	0.04577	0.62864	-105.758
- sCum.n	1	0.07870	0.66157	-104.124
- pt	1	0.08886	0.67172	-103.637
- ct	1	0.08974	0.67261	-103.594
- date	1	0.19650	0.77937	-98.880
- ne	1	0.29452	0.87738	-95.090
- lCap	1	0.70790	1.29076	-82.736

```
##
## Step: AIC=-108.06
## lCost ~ date + t2 + lCap + pr + ne + ct + bw + sCum.n + pt
##
##      Df Sum of Sq    RSS    AIC
## - bw      1   0.00094 0.58592 -110.010
## - t2      1   0.03519 0.62017 -108.192
## <none>                0.58498 -108.061
## - pr      1   0.04370 0.62867 -107.756
## + t1      1   0.00211 0.58286 -106.177
## - sCum.n  1   0.08248 0.66746 -105.840
## - pt      1   0.08712 0.67210 -105.619
## - ct      1   0.08804 0.67302 -105.575
## - ne      1   0.29978 0.88476  -96.822
## - date    1   0.60779 1.19276  -87.263
## - lCap    1   0.71127 1.29625  -84.600
##
## Step: AIC=-110.01
## lCost ~ date + t2 + lCap + pr + ne + ct + sCum.n + pt
##
##      Df Sum of Sq    RSS    AIC
## <none>                0.58592 -110.010
## - pr      1   0.05231 0.63823 -109.273
## - t2      1   0.05233 0.63825 -109.272
## + bw      1   0.00094 0.58498 -108.061
## + t1      1   0.00085 0.58507 -108.057
## - sCum.n  1   0.08294 0.66886 -107.773
## - ct      1   0.08740 0.67332 -107.561
## - pt      1   0.08764 0.67356 -107.549
## - ne      1   0.30004 0.88596  -98.778
## - date    1   0.61189 1.19781  -89.128
## - lCap    1   0.71083 1.29675  -86.588
##
##
## Call:
## lm(formula = lCost ~ date + t2 + lCap + pr + ne + ct + sCum.n +
##      pt, data = df)
##
## Coefficients:
## (Intercept)      date          t2          lCap          pr          ne
## -15.017735    0.237731    0.005466    0.685606   -0.104189    0.256006
##          ct      sCum.n          pt
##   0.119313   -0.068484   -0.216158
```

Starting from the full model the stepwise variable selection suggests the following model: $lCost = date + t2 + lCap + pr + ne + ct + sCum.n + pt$ and therewith drops just 2 variables ($t1$ & bw) and results in an AIC of -110.01.

What if we start with an empty model and perform the variable selection?

```
step(lm(lCost ~ 1, data = df),
     direction = 'both',
     scope = list(upper =~ date + t1 + t2 + lCap + pr + ne + ct + bw + sCum.n + pt,
                  lower =~ 1))
```

```

## Start:  AIC=-61.29
## lCost ~ 1
##
##          Df Sum of Sq  RSS    AIC
## + pt      1   2.01272 2.4153 -78.685
## + date     1   1.75252 2.6755 -75.411
## + t1       1   0.91394 3.5141 -66.686
## + lCap     1   0.76606 3.6620 -65.367
## + ne       1   0.65915 3.7689 -64.446
## + ct       1   0.29142 4.1366 -61.467
## <none>          4.4281 -61.289
## + sCum.n   1   0.15052 4.2775 -60.395
## + bw       1   0.08878 4.3393 -59.937
## + pr       1   0.05087 4.3772 -59.658
## + t2       1   0.00581 4.4223 -59.331
##
## Step:  AIC=-78.68
## lCost ~ pt
##
##          Df Sum of Sq  RSS    AIC
## + lCap     1   0.91498 1.5004 -91.921
## + date     1   0.49197 1.9234 -83.973
## + sCum.n   1   0.36628 2.0491 -81.947
## + ne       1   0.18965 2.2257 -79.301
## + t1       1   0.18163 2.2337 -79.186
## <none>          2.4153 -78.685
## + bw       1   0.07200 2.3433 -77.653
## + ct       1   0.04550 2.3698 -77.293
## + t2       1   0.03212 2.3832 -77.113
## + pr       1   0.00261 2.4127 -76.719
## - pt       1   2.01272 4.4281 -61.289
##
## Step:  AIC=-91.92
## lCost ~ pt + lCap
##
##          Df Sum of Sq  RSS    AIC
## + date     1   0.43560 1.0648 -100.896
## + t1       1   0.26714 1.2332 -96.195
## + ne       1   0.17146 1.3289 -93.804
## + sCum.n   1   0.15713 1.3432 -93.461
## <none>          1.5004 -91.921
## + ct       1   0.04889 1.4515 -90.981
## + bw       1   0.01747 1.4829 -90.296
## + pr       1   0.01151 1.4888 -90.167
## + t2       1   0.01069 1.4897 -90.150
## - lCap     1   0.91498 2.4153 -78.685
## - pt       1   2.16165 3.6620 -65.367
##
## Step:  AIC=-100.9
## lCost ~ pt + lCap + date
##
##          Df Sum of Sq  RSS    AIC
## + ne       1   0.20229 0.86247 -105.638
## + ct       1   0.12776 0.93700 -102.986

```

```

## <none>          1.06476 -100.896
## + t2          1  0.03080 1.03397 -99.835
## + pr          1  0.01883 1.04593 -99.466
## + bw          1  0.01148 1.05328 -99.243
## + t1          1  0.00442 1.06034 -99.029
## + sCum.n      1  0.00120 1.06356 -98.932
## - date        1  0.43560 1.50036 -91.921
## - lCap        1  0.85861 1.92337 -83.973
## - pt          1  0.86610 1.93086 -83.849
##
## Step:  AIC=-105.64
## lCost ~ pt + lCap + date + ne
##
##           Df Sum of Sq    RSS    AIC
## + ct       1  0.11570 0.74677 -108.248
## + t2       1  0.06248 0.79999 -106.045
## + sCum.n   1  0.05322 0.80925 -105.676
## <none>          0.86247 -105.638
## + pr       1  0.01563 0.84684 -104.223
## + bw       1  0.01009 0.85239 -104.014
## + t1       1  0.00688 0.85559 -103.894
## - ne       1  0.20229 1.06476 -100.896
## - date     1  0.46643 1.32890 -93.804
## - pt       1  0.60368 1.46615 -90.659
## - lCap     1  0.83751 1.69998 -85.924
##
## Step:  AIC=-108.25
## lCost ~ pt + lCap + date + ne + ct
##
##           Df Sum of Sq    RSS    AIC
## + sCum.n   1  0.08994 0.65683 -110.354
## <none>          0.74677 -108.248
## + t2       1  0.03416 0.71261 -107.746
## + pr       1  0.00852 0.73825 -106.615
## + bw       1  0.00816 0.73861 -106.599
## + t1       1  0.00074 0.74603 -106.279
## - ct       1  0.11570 0.86247 -105.638
## - ne       1  0.19023 0.93700 -102.986
## - pt       1  0.42372 1.17049 -95.866
## - date     1  0.54164 1.28841 -92.795
## - lCap     1  0.83781 1.58458 -86.173
##
## Step:  AIC=-110.35
## lCost ~ pt + lCap + date + ne + ct + sCum.n
##
##           Df Sum of Sq    RSS    AIC
## <none>          0.65683 -110.354
## + bw       1  0.02032 0.63651 -109.360
## + t2       1  0.01859 0.63823 -109.273
## + pr       1  0.01857 0.63825 -109.272
## + t1       1  0.00639 0.65044 -108.667
## - sCum.n   1  0.08994 0.74677 -108.248
## - pt       1  0.11283 0.76965 -107.282
## - ct       1  0.15243 0.80925 -105.676

```

```
## - ne      1    0.26944 0.92626 -101.355
## - date    1    0.54419 1.20101  -93.042
## - lCap    1    0.92655 1.58338  -84.198

##
## Call:
## lm(formula = lCost ~ pt + lCap + date + ne + ct + sCum.n, data = df)
##
## Coefficients:
## (Intercept)          pt          lCap          date          ne          ct
##   -13.42364    -0.24042     0.72559     0.21509     0.24059     0.15020
##      sCum.n
##   -0.07013
```

This puts out $lCost = pt + lCap + date + ne + ct + sCum.n$ and drops 4 variables (t1, t2, bw & pr) with an AIC of -110.35. Therewith one can conclude that the second model is more parsimonious than the first because its lower AIC.

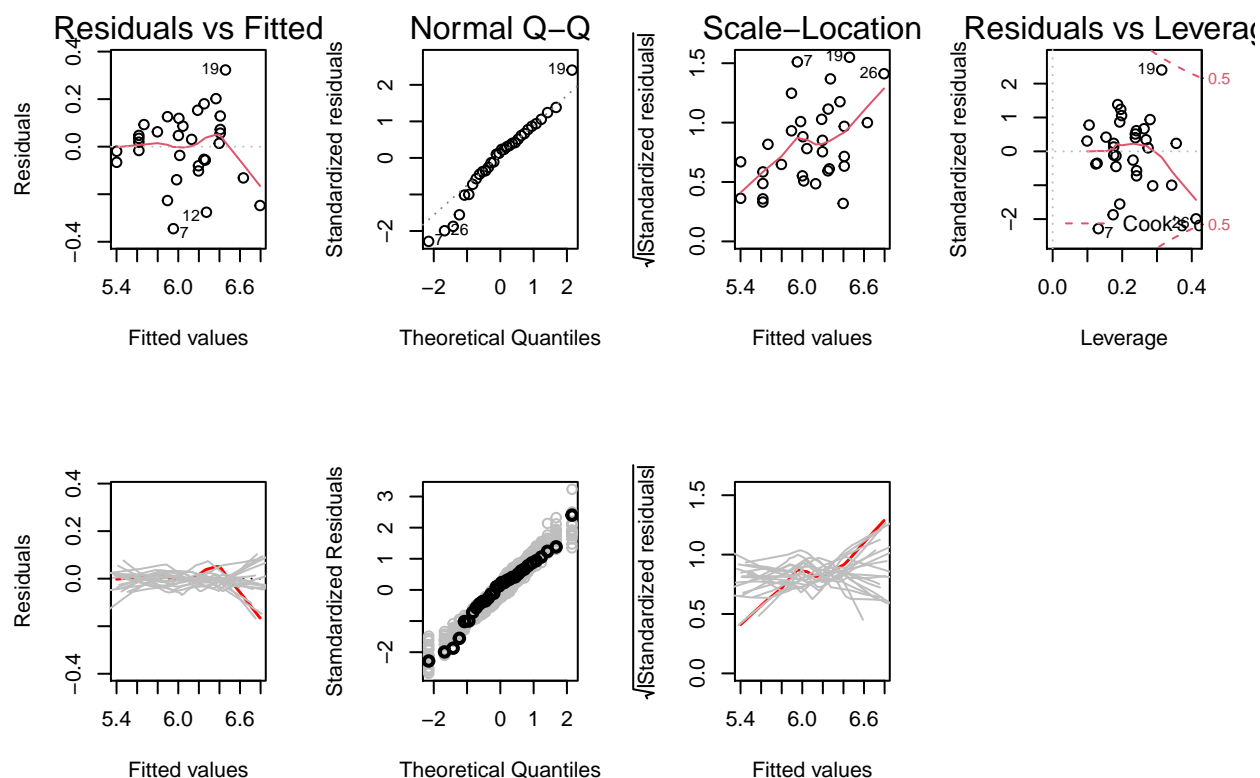
Exercise 4.b)

What about the residual and sensitivity analysis?

```
lm3.2 <- lm(lCost ~ pt + lCap + date + ne + ct + sCum.n, data = df)
summary(lm3.2)
```

```
##
## Call:
## lm(formula = lCost ~ pt + lCap + date + ne + ct + sCum.n, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34497 -0.06940  0.02474  0.08788  0.32267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -13.42364     3.42411  -3.920 0.000608 ***
## pt          -0.24042     0.11602  -2.072 0.048700 *
## lCap         0.72559     0.12218   5.939 3.37e-06 ***
## date         0.21509     0.04726   4.551 0.000119 ***
## ne           0.24059     0.07513   3.202 0.003694 **
## ct           0.15020     0.06236   2.409 0.023707 *
## sCum.n       -0.07013     0.03790  -1.850 0.076133 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1621 on 25 degrees of freedom
## Multiple R-squared:  0.8517, Adjusted R-squared:  0.8161
## F-statistic: 23.92 on 6 and 25 DF,  p-value: 3.183e-09
```

```
par(mfrow=c(2,4))
plot(lm3.2)
plot.lmSim(lm3.2, SEED = 1)
```



There is no evidence that any of the model assumptions is violated. What about multicollinearity?

```
vif(lm3.2)
```

```
##      pt      lCap      date      ne      ct      sCum.n
## 2.497443 1.089288 2.716501 1.289015 1.142437 2.248181
```

There is also no problem with multicollinearity. One can conclude therewith that the model fits the data well.

Exercise 4.c)

```
confint(lm3.2)
```

```
##              2.5 %      97.5 %
## (Intercept) -20.47573093 -6.371552102
## pt          -0.47935760 -0.001481439
## lCap         0.47395060 0.977234875
## date        0.11775366 0.312423592
## ne          0.08586099 0.395323607
## ct          0.02177208 0.278632684
## sCum.n      -0.14819161 0.007933141
```


Question to 4.c)

How do we see with the output above, that p_t affects the price significantly?