Oblig 1

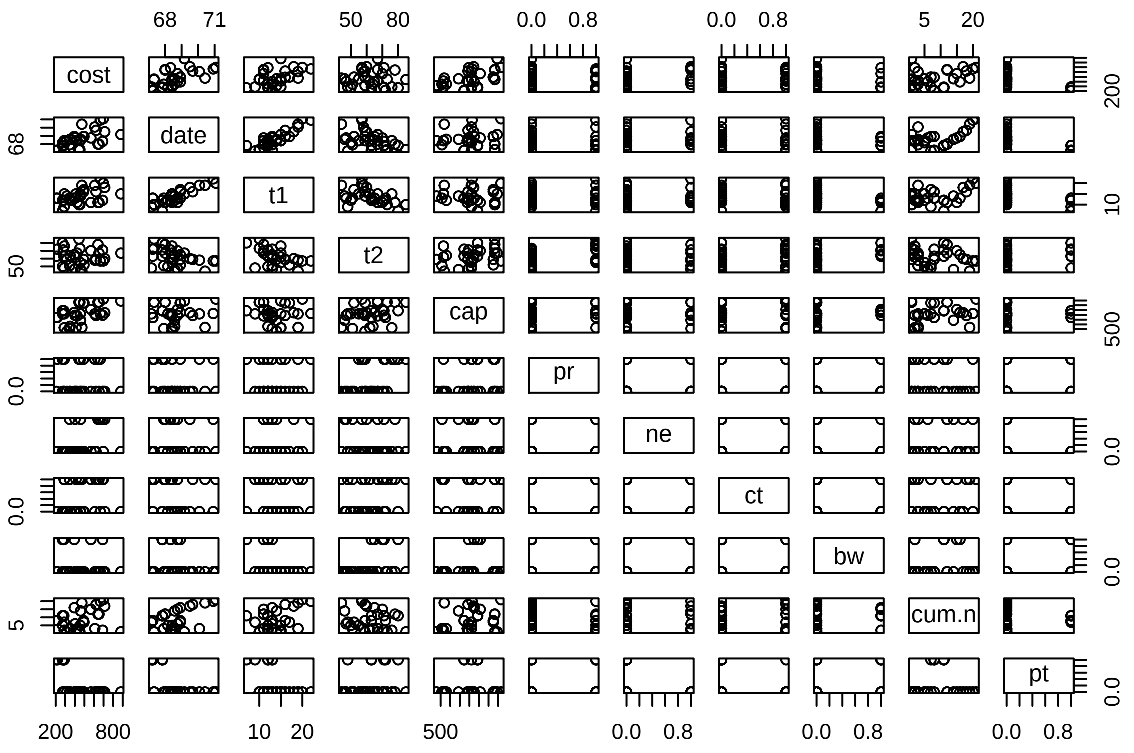
Oblig 1 Stk2100

1.a

```
datadir="https://www.uio.no/studier/emner/matnat/math/STK2100/data/"
nuclear=read.table(paste(datadir,"nuclear.dat",sep=""),header=T)
n=nrow(nuclear)
names(nuclear)
```

```
##  [1] "cost"  "date"  "t1"    "t2"    "cap"   "pr"    "ne"    "ct"    "bw"
## [10] "cum.n" "pt"
```

```
pairs(nuclear)
```



1.b

1. The Epsilons have the same variance.

2. The Epsilons has the same distribution.

3. The Epsilons are independent.

4.The mean of the epsilons are 0

The fourth condition is always satisfied when one have an estimate for β_0 in the model,because β_0 is the mean of the dependent variables.Since one those not now all the variables that are needed to explain the response(assuming that one has a linear relationship). The model will always overestimate or underestimate(beta estimates biased). Having the intercept in the model forces the mean of the residuals to be 0.

If a variable is correlated with the epsilons, condition (3) gets violated.

The variance of the epsilon should be consistent for all observation.If (1) is violated the variance of $\hat{\beta}$'s will not be the smallest one which leads to bad inference.

Assumption (2) makes it allows one to carry out hypothesis testings and get reliable confidence and prediction intervals.

All of these assumptions are important. If one had to choose some of them it would be (1) and (2).At least one would get a model which could be used,with supplementary information on how it could be upgraded.

$Y_i = \beta_0 X_i + \beta_1 X_i + \beta_2 X_i .....\beta_2 X_n$

```r
fit.model=lm(log(cost)~.,data=nuclear)
summary(fit.model)
```

```
##
## Call:
## lm(formula = log(cost) ~ ., data = nuclear)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.284032 -0.081677  0.009502  0.090890  0.266548
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.063e+01  5.710e+00  -1.862  0.07662 .
## date         2.276e-01  8.656e-02   2.629  0.01567 *
## t1           5.252e-03  2.230e-02   0.236  0.81610
## t2           5.606e-03  4.595e-03   1.220  0.23599
## cap          8.837e-04  1.811e-04   4.878 7.99e-05 ***
## pr          -1.081e-01  8.351e-02  -1.295  0.20943
## ne           2.595e-01  7.925e-02   3.274  0.00362 **
## ct           1.155e-01  7.027e-02   1.644  0.11503
## bw           3.680e-02  1.063e-01   0.346  0.73261
## cum.n       -1.203e-02  7.828e-03  -1.536  0.13944
## pt          -2.220e-01  1.304e-01  -1.702  0.10352
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1697 on 21 degrees of freedom
## Multiple R-squared:  0.8635, Adjusted R-squared:  0.7985
## F-statistic: 13.28 on 10 and 21 DF,  p-value: 5.717e-07
```

Observing from summary that there are 3 covariates that are significant. “Cap” is the most significant,then “ne” lastly date. The adjusted Rˆ2 value is 0.7985. 1 means all the points are on the model. 0.7985 is pretty good

1.c

```
#Finding the variable with the largest p-value converting it to a string
s=toString(as.name(names(which.max(summary(fit.model)$coefficients[,4]))))
#dummy vector
g=c(".",s)
outcome="log(cost)"
#updated formula where variable with largest p-value has been taken out
f <- as.formula(
  paste(outcome,
        paste(g, collapse = " - "),
        sep = " ~ "))
#creating model
fit.model2=lm(f,data=nuclear)
summary(fit.model2)
```

```
##
## Call:
## lm(formula = f, data = nuclear)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28898 -0.07856  0.01272  0.08983  0.26537
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.161e+01  3.835e+00  -3.027 0.006187 **
## date         2.431e-01  5.482e-02   4.435 0.000208 ***
## t2           5.451e-03  4.449e-03   1.225 0.233451
## cap          8.778e-04  1.755e-04   5.002 5.25e-05 ***
## pr          -1.035e-01  7.944e-02  -1.303 0.205922
## ne           2.607e-01  7.738e-02   3.368 0.002772 **
## ct           1.142e-01  6.853e-02   1.667 0.109715
## bw           2.622e-02  9.423e-02   0.278 0.783401
## cum.n       -1.220e-02  7.626e-03  -1.599 0.124034
## pt          -2.157e-01  1.249e-01  -1.727 0.098181 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.166 on 22 degrees of freedom
## Multiple R-squared:  0.8631, Adjusted R-squared:  0.8072
## F-statistic: 15.42 on 9 and 22 DF,  p-value: 1.424e-07
```

Based on the high p-value of the variable one could argue that the variable is not associated with the response. Experienced statisticians argue that a lot of times in multiple regression model a lot or some of the variables is not associated with the response. A none significant p-value indicate this. By removing these variable one can get a more understandable model. The Least -squares approach on the other hand is not likely setting any of the betas=0.

Removing one variable from the model could increase or decrease the p-value of the remaining variables because the variables could be correlated. If to variables has a low p-value,then if the remaining variables p-value increases, then they are correlated. The removed variable is the underlying variable,the underlying reason or the variable that creates the climate for the other variable to do well.

Example: relative warm temperature creates an environment for buying ice cream. If one wants to model the income of an ice cream company,where the covariates one choose is “amount of people outside” and " relative warm temperature“.”amount of people outside" will have a lower p-value when " relative warm temperature" is in the model. If one take out " relative warm temperature" variable from the model,one would expect to see an increase in the p-value of “amount of people outside”.

1.d

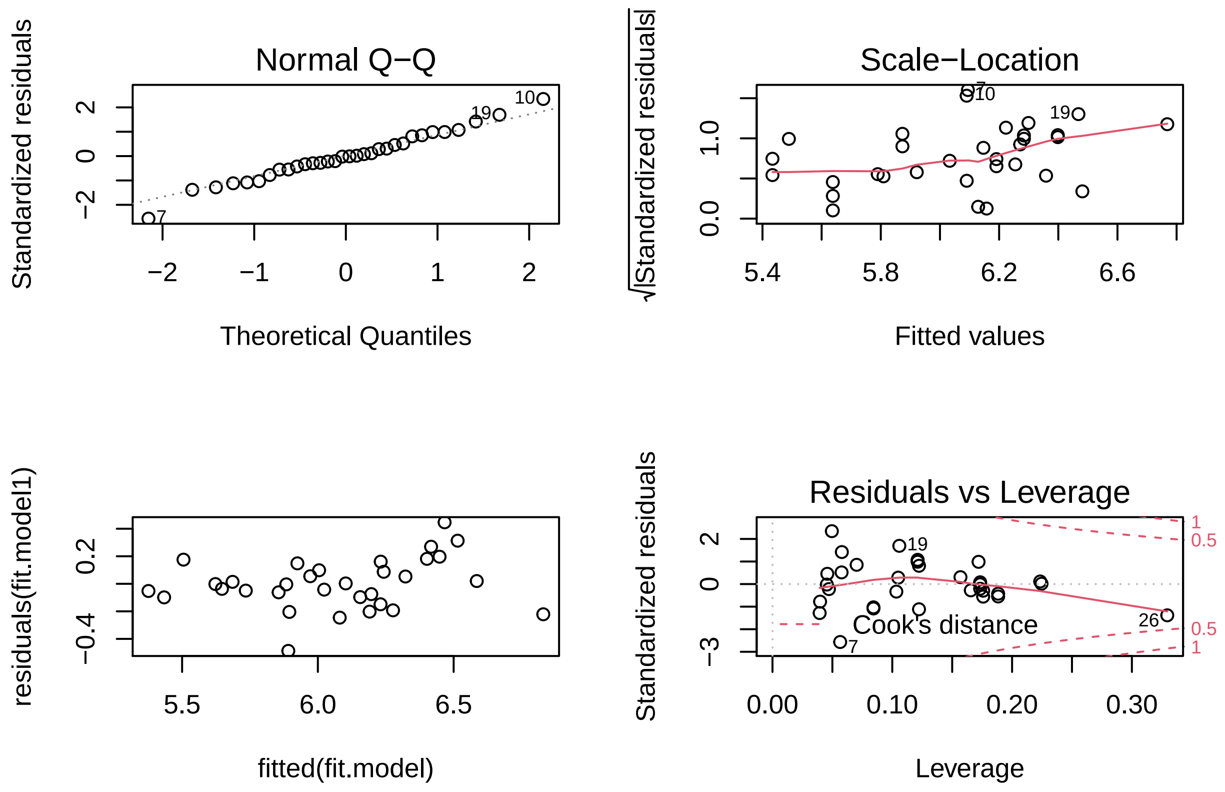
```r
fit.model1=lm(log(cost)~.,data=nuclear,x=TRUE,y=TRUE)
g=c()
for(i in 1:9){


  s=toString(as.name(names(which.max(summary(fit.model1)$coefficients[-1,4]))))
  print(s)
  g=c(g,s)
  t=c(".",g)


  outcome="log(cost)"

  f <- as.formula(
    paste(outcome,
          paste(t, collapse = " - "),
          sep = " ~ "))

  fit.model2=lm(f,data=nuclear,,x=TRUE,y=TRUE)

  w=length(summary(fit.model1)$coef[,4][summary(fit.model1)$coef[-1,4] > .05])


  print(summary(fit.model1))


  fit.model1=fit.model2
  if(w==0){
    break
  }


}
```

```
## [1] "t1"
##
## Call:
## lm(formula = log(cost) ~ ., data = nuclear, x = TRUE, y = TRUE)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.284032 -0.081677  0.009502  0.090890  0.266548
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.063e+01  5.710e+00  -1.862  0.07662 .
## date         2.276e-01  8.656e-02   2.629  0.01567 *
## t1           5.252e-03  2.230e-02   0.236  0.81610
## t2           5.606e-03  4.595e-03   1.220  0.23599
## cap          8.837e-04  1.811e-04   4.878 7.99e-05 ***
## pr          -1.081e-01  8.351e-02  -1.295  0.20943
## ne           2.595e-01  7.925e-02   3.274  0.00362 **
## ct           1.155e-01  7.027e-02   1.644  0.11503
## bw           3.680e-02  1.063e-01   0.346  0.73261
## cum.n       -1.203e-02  7.828e-03  -1.536  0.13944
## pt          -2.220e-01  1.304e-01  -1.702  0.10352
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1697 on 21 degrees of freedom
## Multiple R-squared:  0.8635, Adjusted R-squared:  0.7985
## F-statistic: 13.28 on 10 and 21 DF,  p-value: 5.717e-07
##
## [1] "bw"
##
## Call:
## lm(formula = f, data = nuclear, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28898 -0.07856  0.01272  0.08983  0.26537
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.161e+01  3.835e+00  -3.027 0.006187 **
## date         2.431e-01  5.482e-02   4.435 0.000208 ***
## t2           5.451e-03  4.449e-03   1.225 0.233451
## cap          8.778e-04  1.755e-04   5.002 5.25e-05 ***
## pr          -1.035e-01  7.944e-02  -1.303 0.205922
## ne           2.607e-01  7.738e-02   3.368 0.002772 **
## ct           1.142e-01  6.853e-02   1.667 0.109715
## bw           2.622e-02  9.423e-02   0.278 0.783401
## cum.n       -1.220e-02  7.626e-03  -1.599 0.124034
## pt          -2.157e-01  1.249e-01  -1.727 0.098181 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.166 on 22 degrees of freedom
## Multiple R-squared:  0.8631, Adjusted R-squared:  0.8072
## F-statistic: 15.42 on 9 and 22 DF,  p-value: 1.424e-07
##
## [1] "pr"
##
## Call:
## lm(formula = f, data = nuclear, x = TRUE, y = TRUE)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.290513 -0.082692  0.008663  0.098259  0.260204
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.169e+01  3.748e+00  -3.118 0.004839 **
## date         2.438e-01  5.366e-02   4.544 0.000145 ***
## t2           6.018e-03  3.874e-03   1.553 0.134024
## cap          8.739e-04  1.714e-04   5.099 3.65e-05 ***
## pr          -1.099e-01  7.458e-02  -1.473 0.154275
## ne           2.611e-01  7.580e-02   3.445 0.002206 **
## ct           1.111e-01  6.622e-02   1.677 0.106991
## cum.n       -1.176e-02  7.311e-03  -1.608 0.121414
## pt          -2.071e-01  1.186e-01  -1.747 0.094059 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1626 on 23 degrees of freedom
## Multiple R-squared:  0.8627, Adjusted R-squared:  0.8149
## F-statistic: 18.06 on 8 and 23 DF,  p-value: 3.307e-08
##
## [1] "t2"
##
## Call:
## lm(formula = f, data = nuclear, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29316 -0.06841  0.01246  0.11574  0.32795
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.016e+01  3.688e+00  -2.754 0.011057 *
## date         2.234e-01  5.309e-02   4.208 0.000311 ***
## t2           3.265e-03  3.476e-03   0.939 0.356937
## cap          8.692e-04  1.755e-04   4.953 4.69e-05 ***
## ne           2.485e-01  7.713e-02   3.222 0.003643 **
## ct           1.283e-01  6.676e-02   1.921 0.066644 .
## cum.n       -1.030e-02  7.418e-03  -1.389 0.177564
## pt          -2.464e-01  1.184e-01  -2.082 0.048223 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1665 on 24 degrees of freedom
## Multiple R-squared:  0.8497, Adjusted R-squared:  0.8059
## F-statistic: 19.38 on 7 and 24 DF,  p-value: 1.925e-08
##
## [1] "cum.n"
##
## Call:
## lm(formula = f, data = nuclear, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33603 -0.08193  0.02083  0.07418  0.33274
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.2810892  3.5603026  -2.607 0.015187 *
## date         0.2129251  0.0517795   4.112 0.000371 ***
## cap          0.0009289  0.0001632   5.693 6.28e-06 ***
## ne           0.2399222  0.0764048   3.140 0.004301 **
## ct           0.1411059  0.0651896   2.165 0.040172 *
## cum.n       -0.0108802  0.0073757  -1.475 0.152661
## pt          -0.2432014  0.1180250  -2.061 0.049892 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1661 on 25 degrees of freedom
## Multiple R-squared:  0.8442, Adjusted R-squared:  0.8068
## F-statistic: 22.57 on 6 and 25 DF,  p-value: 5.8e-09
##
## [1] "ct"
##
## Call:
## lm(formula = f, data = nuclear, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36175 -0.08063  0.00584  0.08594  0.42355
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.4058393  2.4567323  -2.200 0.036861 *
## date         0.1563992  0.0356036   4.393 0.000167 ***
## cap          0.0008674  0.0001613   5.378 1.24e-05 ***
## ne           0.1973468  0.0723259   2.729 0.011252 *
## ct           0.1154229  0.0642266   1.797 0.083943 .
## pt          -0.3477717  0.0964752  -3.605 0.001299 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1698 on 26 degrees of freedom
## Multiple R-squared:  0.8306, Adjusted R-squared:  0.798
## F-statistic:  25.5 on 5 and 26 DF,  p-value: 2.958e-09
##
## [1] "ne"
##
## Call:
## lm(formula = f, data = nuclear, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42160 -0.10554 -0.00070  0.07247  0.37328
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.5035539  2.5022087  -1.800 0.083072 .
## date         0.1439104  0.0363320   3.961 0.000491 ***
## cap          0.0008783  0.0001677   5.238 1.61e-05 ***
## ne           0.2024364  0.0751953   2.692 0.012042 *
## pt          -0.3964878  0.0963356  -4.116 0.000326 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1767 on 27 degrees of freedom
## Multiple R-squared:  0.8096, Adjusted R-squared:  0.7814
## F-statistic:  28.7 on 4 and 27 DF,  p-value: 2.255e-09
```

```
par(mfrow=c(2,2))
#normal
plot(fit.model1, which=2)
#constant variance
plot(fit.model1, which=3)
#linearity
plot(fitted(fit.model),residuals(fit.model1))
#leverage
plot(fit.model1, which=5)
```



The first plot shows whether the residuals are standard normal.All the residuals are placed on the line.This condition is satisfied.The second plot shows whether the residuals has constant variance. From the plot it looks like as the fitted values increase the variance increases somewhat but still it does not look that bad. The third plot shows whether their is a linear relationship between the response variable and the explanatory variables.If their is no systematic pattern in the plot, this is an indication of linear relationship.From the plot one see the same pattern as the previous plot. A little increase on average,when the value of the fitted values increases. The last plot shows the points with high leverage.

1.e

Calculating the average quadratic error:

```
avquaderr=1/n*sum((log(nuclear$cost)-fitted(fit.model1))^2)
avquaderr
```

```
## [1] 0.03342471
```

Weakness overfitting.One is using the training data to establish how good the model is.Also the more one decrease the training error the higher the test error becomes. Training error is an optimistic value for the test error.It tends to underestimate the test error.Therefore it makes sense to use cross validation on the final model,to get a better estimate of the test-MSE.

```
library(lmvar)
ten_fold_cross=cv.lm(fit.model1,k=10)$MSE$mean
ten_fold_cross
```

```
## [1] 0.03482954
```

Doing 10-fold-validation on the chosen model yield an estimate that is higher,which confirms in some sense that the previous was an underestimate. One never knows what the true test-MSE is,but based on assumptions,and experiences about the test-MSE, some of which presented in exercise 2,one can get estimates that are better then other estimates.

1.f

$$Y_i = X^T\beta_i + \epsilon_i \quad \epsilon \sim N(0,\sigma^2)$$

$$l(X,Y;\beta,\sigma^2) = -\frac{1}{2}log(2\pi\sigma^2) - \frac{1}{\sigma^2}(Y_i - \beta X_i)^2$$

$$Letting \quad \hat{\beta} \quad be(MLE), and \quad \hat{\sigma}^2 = \frac{1}{n}\sum e_i^2 \quad where \quad e_i = (Y_i - X^T\hat{\beta})$$

$$l_n = log(\prod_{i=1}^{n} f(x_1....x_n)) = \sum_{i=1}^{n} l(X,Y;\beta,\sigma^2)$$

$$= -\frac{n}{2}log(2\pi\hat{\sigma}^2) - \sum_{i=1}^{n}\underbrace{\frac{1}{\hat{\sigma}^2}(Y_i - \beta X_i)^2}_{\frac{n\hat{\sigma}^2}{\hat{\sigma}^2}}$$

$$= -\frac{n}{2}(log(2) + log(\pi) + log(\hat{\sigma}^2)) - n$$

$$-Const - \frac{n}{2}log(\hat{\sigma}^2) = -log(L(\hat{\theta}))$$

therefore:

$$\underline{\underline{AIC = -2log(L(\hat{\theta})) + 2(q+2) = Const + nlog(\hat{\sigma}^2) + 2(q+2)}}$$

and:

$$\underline{\underline{BIC = -2log(L(\hat{\theta})) + log(n)(q+2) = Const + nlog(\hat{\sigma}^2) + log(n)(q+2)}}$$

1.h

The goal is to minimize the value of AIC and BIC. Assuming that the constant term and the penalty term is positive. The only factor that can decrease the AIC and BIC value is $nlog(\hat{\sigma}^2)$ we know that $\hat{\sigma}^2 = \frac{1}{n}\sum e_i^2$. $\sum e_i^2 = RSS$. RSS i this case seem to vary between 0 and 1,which means that the smaller RSS is the more negative the value of $nlog(\hat{\sigma}^2)$. Since RSS does not depend on AIC and BIC,stays "constant". We get the same RSS values for each iteration, when we apply AIC and BIC functions. Therefore the selection-order is the same.

1.i

Selection based on p-value,could give different selection order compared to AIC and BIC, since correlated covariates creates uncertainty about what the p-value would be, once one take out one of the correlated variables.

Both AIC and BIC corresponds to model selection based on minimizing the least squares,when one has a linear model with gaussian noise. That means that for each iteration you are finding the Beta values that minimizes the least squares. These beta values are unique. What one does when reducing the numbers of covariates based on p-value, is fixing the beta corresponding to largest p-value to the constant 0. Since the values of betas attained when using least squares are the unique combinations of values of beta that minimizes the RSS,any other combinations of beta will increase RSS,this will lead to,more or less variables selected when using p-value selection methods compared to AIC and BIC. Therefore we might get a different model using p-value selection methods then using AIC or BIC. One might still get lucky,or the data one has ends up yielding the same model for all the selection-methods.

1.j

$$E[e^Z] = \int_{-\infty}^{\infty} e^z \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(z-\mu)^2}{2\sigma^2}} dz$$

Looking at the power of the exponensial we get:

$$z - \frac{(z-\mu)^2}{2\sigma^2} = -\frac{-2\sigma^2 z + z^2 + 2\mu z + \mu^2}{2\sigma^2}$$

$$= -\frac{-2\sigma^2 z + z^2 + 2\mu z + \mu^2 + (\sigma^2+\mu)^2 - (\sigma^2+\mu)^2}{2\sigma^2} = -\frac{z^2 - 2z(\mu+\sigma^2) + (\sigma^2+\mu)^2 + \mu^2 - (\sigma^2+\mu)^2}{2\sigma^2} =$$

$$= -\frac{(z-(\sigma^2+\mu)^2)^2}{2\sigma^2} - \frac{\mu^2 - (\sigma^2+\mu)^2}{2\sigma^2} = -\frac{(z-(\sigma^2+\mu)^2)^2}{2\sigma^2} + \frac{2\mu+\sigma^2}{2}$$

$$E[e^z] = \int_{-\infty}^{\infty} e^{\frac{2\mu+\sigma^2}{2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(z-(\mu+\sigma^2))^2}{2\sigma^2}} dz$$

$$\underline{\underline{E[e^Z] = e^{\frac{2\mu+\sigma^2}{2}}}}$$

I would choose the second equation since the epsilons are normally distributed that makes log(cost) normally distributed,which means that cost is log-normal. In the previous part of the exercise one calculated the expectation of a log.normal distribution.

1.k

```
anova(P_value,AIC,BIC)
```

```
## Analysis of Variance Table
##
## Model 1: log(cost) ~ date + cap + pt
## Model 2: log(cost) ~ date + t2 + cap + pr + ne + ct + cum.n + pt
## Model 3: log(cost) ~ date + cap + ne + ct + pt
##   Res.Df     RSS Df Sum of Sq      F  Pr(>F)
## 1     28 1.06959
## 2     23 0.60816  5   0.46143 3.4901 0.01712 *
## 3     26 0.75007 -3  -0.14191 1.7889 0.17739
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
d.new = data.frame(date=70.0,t1=13,t2=50,cap=800,pr=1,ne=0,ct=0,bw=1,cum.n=8,pt=1)
predict(P_value,d.new,se.fit=TRUE)
```

```
## $fit
##        1
## 5.863783
##
## $se.fit
## [1] 0.127562
##
## $df
## [1] 28
##
## $residual.scale
## [1] 0.1954474
```

```
predict(AIC,d.new,se.fit=TRUE)
```

```
## $fit
##        1
## 5.968446
##
## $se.fit
## [1] 0.1500756
##
## $df
## [1] 23
##
## $residual.scale
## [1] 0.1626095
```

```
predict(BIC,d.new,se.fit=TRUE)
```

```
## $fit
##        1
## 5.888265
##
## $se.fit
## [1] 0.1111445
##
## $df
## [1] 26
##
## $residual.scale
## [1] 0.1698493
```

One sees from the result that the new observation is predicted to be around 5.9 using all the models. The standard deviation of the new value is lowest for BIC ,0.1111445 but not much lower then AIC around 0.1500756 and P-value 0.127562. The model found by BIC seem to be the most accurate model.

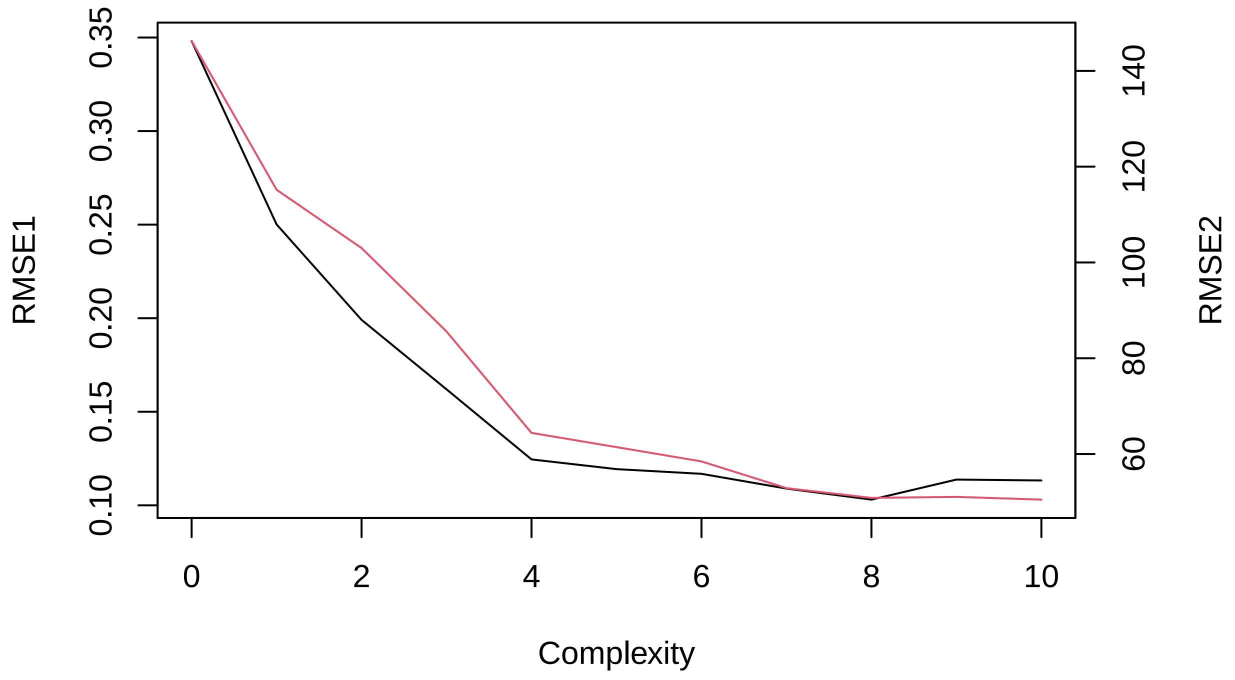
2.a

```
library(MASS)
n = nrow(nuclear)
ind = sample(1:n,n/2,replace=FALSE)
RMSE.test1 = rep(NA,11)
RMSE.test2 = rep(NA,11)
model_narrow = lm(log(cost) ~ 1, data = nuclear)
model_wide = lm(log(cost) ~ ., data = nuclear)
for(i in 0:10)
{
  fit = stepAIC(model_narrow, direction="forward", steps=i,data=nuclear[ind,],
                scope=list(lower=model_narrow, upper=model_wide), k = 0)
  pred = predict(fit,nuclear[-ind,])
  RMSE.test1[i+1] = sqrt(mean((log(nuclear$cost[-ind])-pred)^2))
  RMSE.test2[i+1] = sqrt(mean((nuclear$cost[-ind]-exp(pred))^2))
}
```

First the code-writer is making a variable "ind" generating sequence of numbers random,with length n/2.Creating two arrays named RMSE.test1 and RMSE.test2 respectively.Creating the smallest model in consideration "model_narrow" and the biggest model in consideration "model_wide" .The variable "ind" is used to create a training set and a test set.

StepAIC uses the data corresponding to the indices given by "ind". StepAIC starts with the "model_narrow".In the first for-loop iteration it finds the variable that improves the "model_narrow" the most. The rest of the data (complement of "ind") is used to test how well the model predicts these data points using the function "predict". It continues in the same fashion until it reaches the length of "model_wide". At this point it stops. While the for-loop was active the code also calculated RMSE for each model taken into consideration by StepAIC.

```
par(mar = c(5, 4, 4, 4) + 0.3)  # Leave space for z axis
plot(0:10,RMSE.test1,xlab="Complexity",ylab="RMSE1",type="l") # first plot
par(new = TRUE)
plot(0:10,RMSE.test2, type = "l", axes = FALSE, bty = "n", xlab = "", ylab = "",col=2)
axis(side=4, at = pretty(range(RMSE.test2)))
mtext("RMSE2", side=4, line=3)
```



One can observe from the plot that as the complexity increases the RMSE decreases down until the number of covariates are 6. At which point the RMSE stays almost constant. The minimum RMSE occurs when the number of covariates is 10. Forward Selection is the method that StepAIC is using to find the best model. Explained in the nest section, forward selection those not end up giving the best model,because it does not take into account how all of the covariates behave with each other. Looking at this plot, the conclusion is,one- variable model is the model that explain the test data the most,which does not have to be the case.

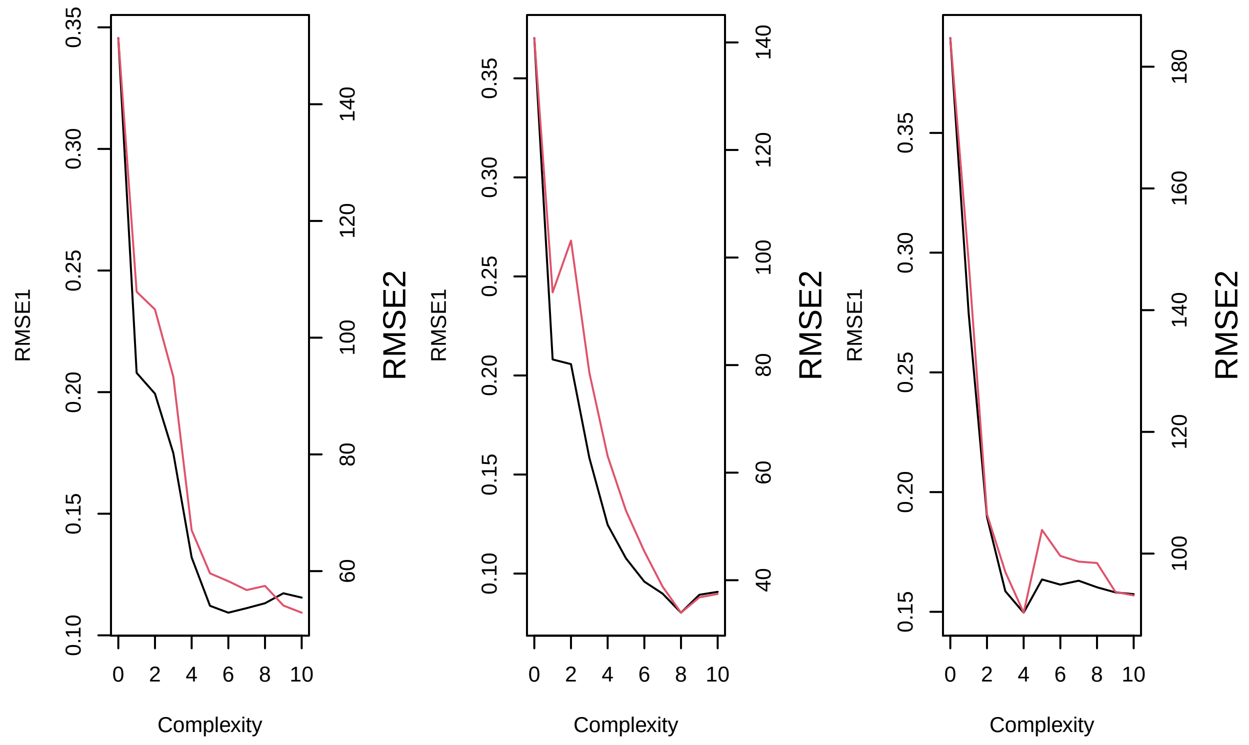
The code given for this exercise is an implementation of a forward selection procedure.When comparing this procedure with subset model selection selection,there is one issue that comes to mind. While that model selections finds the best possible model given a set of covariates, the forward selection procedure does not. It starts with no covariates and incrementally adds the covariate that gives the best improvement. The problem is,for each iteration all the variables that already is selected cant be changed,and so it does not take into account the relationship between the covariates.

Lets say one has tree covariates X_1, X_2, X_3. The best possible one variable model is X_1 i.The best possible second variable model is $X_2 \ \ and \ \ X_3$. The forward selection algorithm would have chosen X_1 in the first iteration. The final two variable model for the forward selection algorithm becomes:

$X_1 \ \ and \ \ X_2 \ \ or \ \ X_1 \ \ and \ \ X_3$,while the optimal two variable model was $X_2 \ \ and \ \ X_3$.

The advantage with forward selection is the number of models that have to be compared for each iteration. The number is mach lower compared to sub set model selection

repeating this procedure 3 times:



Looking at the the three plots,they do not get same minimum.When the complexity grows one still sees an the same behavior form RMSE. In the first plot RMSE staid almost constant after 6 covariates where added.Based on this any complexity over 6 is the optimal complexity. One often would like to get the most simple model.

Test MSE:

By dividing the data into test and training data one can avoid this to some degree. Dividing the data into two will give less data to train the model and so one expect a somewhat higher bias since one is using less observations for the training set (overestimate of the test error) . Depending on which observation are included in the training set and which are used for the validation -set the test error rate will vary highly,so the estimate would in some sense be less reliable.

Reffering to page 203"G james 2t al.Introduction to statistical learning:with Applications in R, Springer Texts statistics.

"""

Prediction: Provided that true relationship between the response and the predictors and the predictor is approximately linear,least squares estimate will have low bias. If n»p that is if n,the number of,is much larger then p,the number of variables-then the least squares estimate also have low bias,and hence will preform well in the test observations.However,if n is not much larger then p,then their could be a lot of variability in the least squares fit,resulting in overfitting and consequently poor predictions on future observations not used in the model training.

"""

Bias-variance trade of

$E[(y_0 - \hat{f}(x_0))^2] = Var[\hat{f}(x_0)] + (E[E[\hat{f}(x_0)] - y_0])^2 + Var[\epsilon_i]$

$\sqrt(E[(y_0 - \hat{f}(x_0))^2])$ defines the expected test RMSE.It refers to the average test-MSE if one would repeatedly estimate the true f ,using different training sets and tested each at x_0. $Var[\hat{f}(x_0)]$ refers to change in $\hat{f}$ using different training sets.$(E[E[\hat{f}(x_0)] - y_0])^2$ refers to the bias^2.The bias is a systematic over or under error estimate,when comparing the expected estimated value using $\hat{f}$ compared to the true value estimated by f.

The general rule of "bias-variance trade-off" is: as the flexibility increases the bias deceases and the variance increase for the test-RMSE. What one can see from the plots is as the complexity increase the RMSE decreases. The complexity increases,the bias increases,the variance decreases. It looks like the variance decreases more than the bias increases. According to the quote from the book if n is not »p one expect the variance to be higher then if n»p. The plot indicates either that n»p ,so the test-RMSE calculated cant be lower or that n is not »p. If the later is true then RMSE could be lowered even more by increasing n=32 in this case. N=32 is not a very big number compared to P=10.

If the goal is to find the right complexity this is not relevant,but if the goal is to find a more accurate estimate for the test-RMSE then the previous section is relevant.

2.b

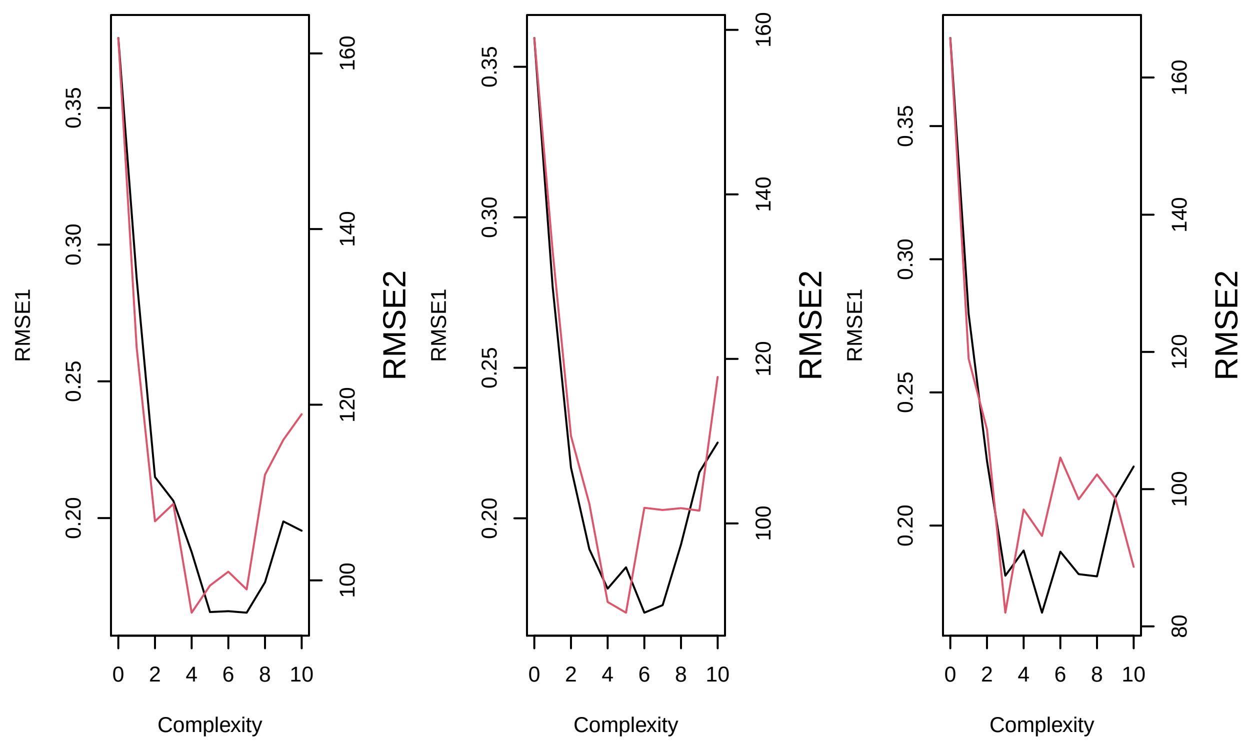
```r
library(lmvar)
RMSE.cv1 = rep(0,10)
RMSE.cv2 = rep(0,10)
for(i in 0:10)
{
  fit = stepAIC(model_narrow, direction="forward", steps=i,k=0,
                scope=list(lower=model_narrow, upper=model_wide),trace=0)
  fit = lm(formula(fit),data=nuclear,x=TRUE,y=TRUE)
  #Note: the k in the command below has a different meaning than k above!!!
  RMSE.cv1[i+1] = cv.lm(fit,k=10)$MSE_sqrt$mean
  RMSE.cv2[i+1] = cv.lm(fit,k=10,log=TRUE)$MSE_sqrt$mean
}
```

This is an implementation of cross- validation.Creating two zero vectors "RMSE.cv1" and "RMSE.cv2". Using stepAIC in the same way as is 2.a.At the end using 10-fold cross validation for each model in consideration and extracting RMSE from the 10-fold cross validation,both for cost and log(cost),putting them into "RMSE.cv1" and "RMSE.cv2" respectively.

When using cross-validation techniques one is trying to pinpoint what the test error is.Using all the data for training gives the problem of overfitting.The training error goes down,while the test error does not have to.Using this estimate for the test error is an underestimate of the test error.

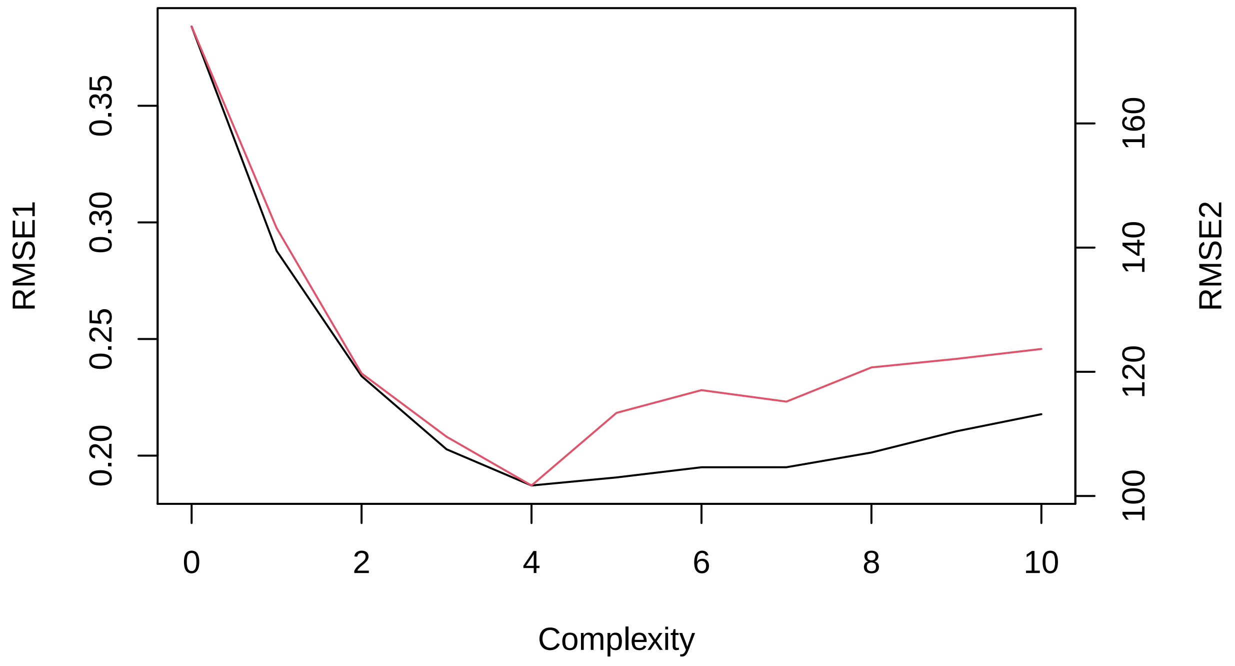
The algorithm presented uses 10-fold cross validation which has the advantage to lower test error,because it decreases the bias compared to just dividing the data into a training set and test set. 10-fold cross validation uses more of the data for training for each iteration. For each iteration it divides the data into 10,uses 9 of the sets as one set for training(fitting) and the last one for testing (predicting).Using the predicted values one can calculate test-MSE and take the average over all the 10 test-MSE.

```r
par(mar = c(5, 4, 4, 4) + 0.3)  # Leave space for z axis
plot(0:10,RMSE.cv1,xlab="Complexity",ylab="RMSE1",type="l") # first plot
par(new = TRUE)
plot(0:10,RMSE.cv2, type = "l", axes = FALSE, bty = "n", xlab = "", ylab = "",col=2)
axis(side=4, at = pretty(range(RMSE.test2)))
mtext("RMSE2", side=4, line=3)
```



2.c

```
library(lmvar)
RMSE.cv1 = rep(0,10)
RMSE.cv2 = rep(0,10)
for(i in 0:10)
{
  fit = stepAIC(model_narrow, direction="forward", steps=i,k=0,
                scope=list(lower=model_narrow, upper=model_wide),trace=0)
  print(fit)
  fit = lm(formula(fit),data=nuclear,x=TRUE,y=TRUE)
  #Note: the k in the command below has a different meaning than k above!!!
  RMSE.cv1[i+1]=sqrt(1/n*sum(((log(nuclear$cost)-fitted(fit))/(1-hatvalues(fit)))^2))
  RMSE.cv2[i+1]=sqrt(1/n*sum((((nuclear$cost)-exp(fitted(fit)))/(1-hatvalues(fit)))^2))
}
par(mar = c(5, 4, 4, 4) + 0.3)  # Leave space for z axis
plot(0:10,RMSE.cv1,xlab="Complexity",ylab="RMSE1",type="l") # first plot
par(new = TRUE)
plot(0:10,RMSE.cv2, type = "l", axes = FALSE, bty = "n", xlab = "", ylab = "",col=2)
axis(side=4, at = pretty(range(RMSE.test2)))
mtext("RMSE2", side=4, line=3)
```



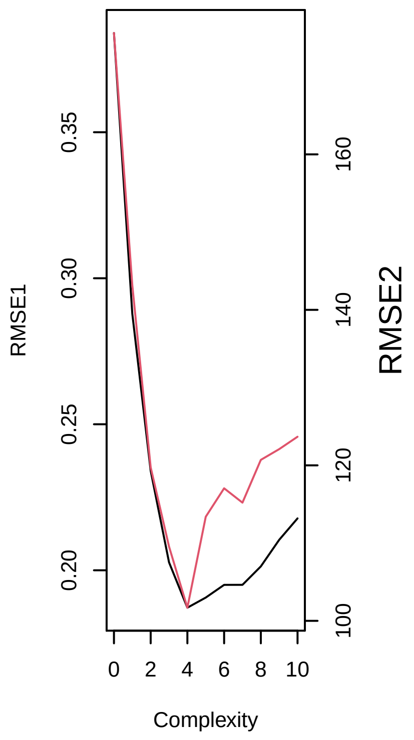
Using the formula on page 180 in the book "Introduction to statistical learning".Using a single fit for each iteration,using all of the fitted values,and putting them into the $\frac{1}{n}\sum_{i=1}^{n}(\frac{y_i-\hat{y}}{1-h_i})^2$.

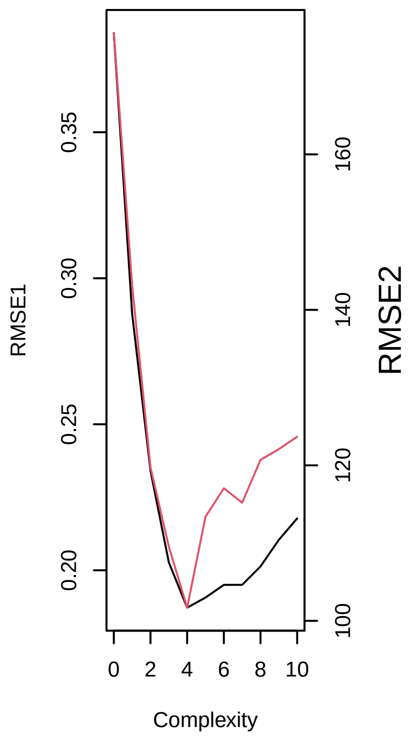
This approach is much less computationally expensive,instead of fitting the model n times, the cost of doing what is done in the code the cost is equal to fitting the model one time.This formula could be used for linear model fitting by least squares

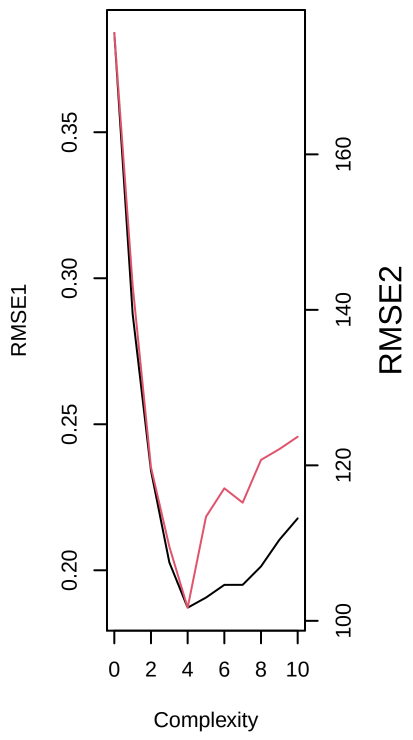
LOOCV has the advantage over the algorithm in 2.a, in that one is using almost all of the data for fitting the model. The LOOCV therefor does not usually overestimate the test error.In contrast to algorithm in 2.a. LOOCV is an unbiased estimate of the test-MSE.Comparing it to the algorithm in 2.b,which has an intermediate level of bias, since each training-set contains (k-1)n/k fewer observations observations then in LOOCV. LOOCV would have a higher variance since the the training set are highly positively correlated.Compared to the algorithm in 2.b where the k<n folds are less correlated. LOOCV is a special case of k-fold validation.

Looking at the plot from both LOOCV and 10-fold cross validation one sees almost no difference.Might be because the extra bias of the 10-fold validation compared to LOOCV cancels the extra variance of the LOOCV compared to 10-fold validation.

Looking at the plot from both LOOCV, and 2-fold cross validation one there is a huge difference. This might be because 2-fold cross validation is more a biased estimate for test-MSE then LOOCV.The variability between the fitted models is also higher,although the variability is lower in the true test-MSE.







3.a

	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	$x_{i,4}$	……..	……..	……..	$x_{i,K-1}$	$x_{i,K}$
$c_i = 1$	1	0	0	0	……..	……..	……..	0	0
$c_i = 2$	0	1	0	0	……..	……..	……..	0	0
$c_i = 3$	0	0	1	0	……..	……..	……..	0	0
$c_i = 4$	0	0	0	1	……..	……..	……..	0	0
……	……..	……..	……..	……..	……..	……..	……..	……..	……..
…….	……..	……..	……..	……..	……..	……..	……..	……..	……..
…….	……..	……..	……..	……..	……..	……..	……..	……..	……..
…….	……..	……..	……..	………	………	……..	……..	……..	……..
$c_i = n - 1$	0	0	0	0	………	………	………	1	0
$c_i = n$	0	0	0	0	………	………	………	0	1

$c_i = 1$	$\beta_0 + \epsilon_i$	$\alpha_1 + \epsilon_i$
$c_i = 2$	$\beta_0 + \beta_2 + \epsilon_i$	$\alpha_2 + \epsilon_i$
$c_i = 3$	$\beta_0 + \beta_3 + \epsilon_i$	$\alpha_3 + \epsilon_i$
$c_i = 4$	$\beta_0 + \beta_4 + \epsilon_i$	$\alpha_4 + \epsilon_i$
....	....	....
....	....	....
....	....	....
....	....	....
....	....	....
$c_i = n - 1$	$\beta_0 + \beta_{K-1} + \epsilon_i$	$\alpha_{K-1} + \epsilon_i$
$c_i = n$	$\beta_0 + \beta_K + \epsilon_i$	$\alpha_K + \epsilon_i$

3.b

$$X = \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,K} \\ x_{2,1} & \dots & x_{2,j} & \dots & x_{2,K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n,j} & \dots & x_{n,j} & \dots & x_{n,K} \end{bmatrix}$$

$$X^T = \begin{bmatrix} x_{1,1} & x_{2,1} & \dots & x_{i,1} & \dots & x_{n,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_{1,j} & x_{2,j} & \dots & x_{i,j} & \dots & x_{n,j} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_{1,K} & x_{2,K} & \dots & x_{i,K} & \dots & x_{n,k} \end{bmatrix}$$

$$X^T X = \begin{bmatrix} \sum_{i=1}^K x_{l,1}x_{l,1} & \dots & \sum_{i=1}^K x_{l,1}x_{l,j} & \dots & \sum_{i=1}^K x_{l,1}x_{l,K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^K x_{l,j}x_{l,1} & \dots & \sum_{i=1}^K x_{l,j}x_{l,j} & \dots & \sum_{i=1}^K x_{l,j}x_{l,K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^K x_{l,K}x_{l,1} & \dots & \sum_{i=1}^K x_{l,K}x_{l,j} & \dots & \sum_{i=1}^K x_{l,K}x_{l,K} \end{bmatrix}$$

$$= \begin{bmatrix} n_1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & n_j & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \dots & n_K \end{bmatrix}$$

$$X^T y = \begin{bmatrix} x_{1,1} & x_{2,1} & \dots & x_{i,1} & \dots & x_{n,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_{1,j} & x_{2,j} & \dots & x_{i,j} & \dots & x_{n,j} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_{1,K} & x_{2,K} & \dots & x_{i,K} & \dots & x_{n,k} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \sum x_{i,1}y_l \\ \sum x_{i,2}y_l \\ \vdots \\ \vdots \\ \vdots \\ \sum x_{i,K}y_l \end{bmatrix} = \begin{bmatrix} \sum_{i:c_i=1} y_l \\ \sum_{i:c_i=2} y_l \\ \vdots \\ \sum_{i:c_i=j} y_l \\ \vdots \\ \sum_{i:c_i=K} y_l \end{bmatrix}$$

an estimate for α is $(X^TX)^{-1}X^Ty$.

$$\hat{\alpha} = \frac{1}{n_j}\begin{bmatrix} \sum_{i:c_i=1} y_i \\ \sum_{i:c_i=2} y_i \\ \vdots \\ \sum_{i:c_i=j} y_i \\ \vdots \\ \sum_{i:c_i=K} y_i \end{bmatrix}$$

This result shows that the value a new observation gets using a model based on this least squares is $\hat{\alpha}_j$ when the person belongs to group j for $j \in 1....K$. Each $\hat{\alpha}_j$ is the mean of all the observations belonging to group j. That sounds reasonable. The only problem with this estimate is if there are no individuals that belong to group j.

3.c

$$\beta_j = \begin{cases} \alpha_1 & if \quad j = 0 \\ \alpha_j - \alpha_1 & if \quad j \; != 0 \end{cases} \quad for \quad j \in 1.......K$$

$$\hat{\beta} = \begin{cases} \frac{1}{n_1}\sum_{i:c_1=1} y_i & if \quad j = 0 \\ \frac{1}{n_j}\sum_{i:c_j=j} y_i - \frac{1}{n_1}\sum_{i:c_1=1} y_i & if \quad j \; != 0 \end{cases} \quad for \quad j \in 1.......K$$

Getting β from α is as one to one mapping so $\hat{\beta}$ is also a least squares estimate.

3.d

$\gamma_i + \gamma_0 = \alpha_i$

$\gamma_0 + \gamma_0 + ...\gamma_0 + \gamma_1 + \gamma_3....\gamma_k = \alpha_1 + \alpha_2 + \alpha_3... + \alpha_K$

$K\gamma_0 + \sum_{i=1}^K \gamma_i = \sum_{i=1}^K \alpha_i$

$\gamma_0 = \frac{1}{K}\sum_{i=1}^K \alpha_i$

$\gamma_i = \alpha_i - \frac{1}{K}\sum_{i=1}^K \alpha_i$

3.e

```r
datadir = "http://www.uio.no/studier/emner/matnat/math/STK2100/data/"
Fe <- read.table(paste(datadir,"fe.txt",sep=""),
                 header=T,sep=",")
fit <- lm(Fe~form,data=Fe)
summary(fit)
```

```
##
## Call:
## lm(formula = Fe ~ form, data = Fe)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.113  -2.580  -0.290   2.901  11.279
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  21.5050     1.6202  13.273 7.59e-16 ***
## form          2.8540     0.5916   4.824 2.30e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.183 on 38 degrees of freedom
## Multiple R-squared:  0.3798, Adjusted R-squared:  0.3635
## F-statistic: 23.27 on 1 and 38 DF,  p-value: 2.296e-05
```

This goes wrong because the lm function does not understand that form is a categorical variable.

```
Fe$form <- as.factor(Fe$form)
fit1 <- lm(Fe~form-1,data=Fe)
summary(fit1)
```

```
##
## Call:
## lm(formula = Fe ~ form - 1, data = Fe)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.340 -1.255 -0.250  1.770 10.360
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## form1   26.080      1.251   20.85   <2e-16 ***
## form2   24.690      1.251   19.74   <2e-16 ***
## form3   29.950      1.251   23.95   <2e-16 ***
## form4   33.840      1.251   27.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.955 on 36 degrees of freedom
## Multiple R-squared:  0.9834, Adjusted R-squared:  0.9815
## F-statistic: 532.5 on 4 and 36 DF,  p-value: < 2.2e-16
```

this model belongs to $Y_i = \alpha_1 x_{i,1} + \alpha_2 x_{i,2} + \alpha_3 x_{i,3} .... \alpha_K x_{i,K} + \epsilon_{i,j}$ the constraint is $\alpha_1 = 0$

3.f

```
options()$contrasts
```

```
##         unordered           ordered
## "contr.treatment"      "contr.poly"
```

```
options(contrasts=c("contr.treatment","contr.treatment"))
fit2 <- lm(Fe~form,data=Fe)
summary(fit2)
```

```
##
## Call:
## lm(formula = Fe ~ form, data = Fe)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.340 -1.255 -0.250  1.770 10.360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   26.080      1.251  20.852  < 2e-16 ***
## form2         -1.390      1.769  -0.786   0.4371
## form3          3.870      1.769   2.188   0.0352 *
## form4          7.760      1.769   4.387  9.6e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.955 on 36 degrees of freedom
## Multiple R-squared:  0.4748, Adjusted R-squared:  0.431
## F-statistic: 10.85 on 3 and 36 DF,  p-value: 3.199e-05
```

```
options(contrasts=c("contr.sum","contr.sum"))
options()$contrasts
```

```
## [1] "contr.sum" "contr.sum"
```

```
fit3 <- lm(Fe~form,data=Fe)
summary(fit3)
```

```
##
## Call:
## lm(formula = Fe ~ form, data = Fe)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.340 -1.255 -0.250  1.770 10.360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  28.6400     0.6254  45.798  < 2e-16 ***
## form1        -2.5600     1.0831  -2.363 0.023622 *
## form2        -3.9500     1.0831  -3.647 0.000833 ***
## form3         1.3100     1.0831   1.209 0.234375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.955 on 36 degrees of freedom
## Multiple R-squared:  0.4748, Adjusted R-squared:  0.431
## F-statistic: 10.85 on 3 and 36 DF,  p-value: 3.199e-05
```

fit2 corresponds to $Y_i = \beta_0 + \beta_2 x_{i,2} + \beta_3 x_{i,3} + \beta_4 x_{i,4} .... \beta_K x_{i,K} + \epsilon_{i,j}$,while fit3 corresponds to $Y_i = \gamma_0 + \gamma_1 x_{i,1} + \gamma_2 x_{i,2} + \gamma_3 x_{i,3} .... \gamma_K x_{i,K} + \epsilon_{i,j}$ with the constraint $\sum_{i=1}^{K} \gamma_i = 0$

	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\gamma}$
$j = 0$	0	26.08	28.64
$j = 1$	26.08	0	-2.56
$j = 2$	24.69	-1.39	-3.95
$j = 3$	29.95	3.87	1.31
$j = 4$	33.84	7.76	

One sees from the table that $\hat{\beta}_0 = \hat{\alpha}_1$, $\hat{\beta}_2 = \hat{\alpha}_2 - \hat{\alpha}_1 = 24.69 - 26.08 = -1.39$ and so on.

$\hat{\gamma}_0 = \frac{1}{4}\sum_{j=1}^{4}\hat{\alpha}_j = \frac{1}{4}(26.08 + 24.69 + 29.95 + 33.84) = 28.64$

$\hat{\gamma}_1 = \hat{\alpha}_1 - \frac{1}{4}\sum_{j=1}^{4}\hat{\alpha}_j = 26.08 - \frac{1}{4}(26.08 + 24.69 + 29.95 + 33.84) = -2.56$

and so on. The result match with the formulas derived in the earlier exercises.

3.g

testing whether all of γ_j are equal to 0 or not.

$H_0 : \gamma_j = 0 \ H_a : \gamma_j \ \ ! = 0$

one can directly see from the summary what the F-statistic is.In this case it is 10.85>1 and use it to reject the null hypothesis with p-value 3.1999e-05,so at 0.0001 significance.

one could also derive the F- statistic using the summary table.

$F = \frac{MSR}{MSE} = \frac{\frac{SSR}{K}}{\frac{SSE}{n-(k+1)}} \ \ where \ \ SSR = R^2 SST \ \ and \ \ SSE = (1 - R^2)SST \ \ which \ \ gives$

$\frac{\frac{SSR}{K}}{\frac{SSE}{n-(k+1)}} = \frac{\frac{R^2 SST}{K}}{\frac{(1-R^2)SST}{n-(k+1)}} = \frac{\frac{R^2}{K}}{(1-R^2)(n-(k+1))}$

in our case $\frac{\frac{R^2}{K}}{(1-R^2)(n-(k+1))} = \frac{\frac{0.4748^2}{3}}{(1-0.4748^2)36} = 10.85$

3.h

Based on exercise one the natural thing that comes to mind is to remove none-significant covariates.