Mandatory Assignment 1

STK2100

Håkon Berggren Olsen

Spring 2022

Problem 1.

The dataset, nuclear, contains data regarding the construction of 32 Light-Water Reactors built between 1967 and 1971. The scope of the dataset is to predict the cost of construction for further span given the dataset within this time interval.

The dataset has 32 rows and 11 columns, and has a mix of continous- and categorical variabels. Contionous data such as regarding the cost, construction and net capacity of the a given nuclear powerplant, and categorical such as cooling tower present or absent, if there exists an LWR (Light-Water Reactor) plant at the same site and such.

The description of all the variables are shown in a table below.

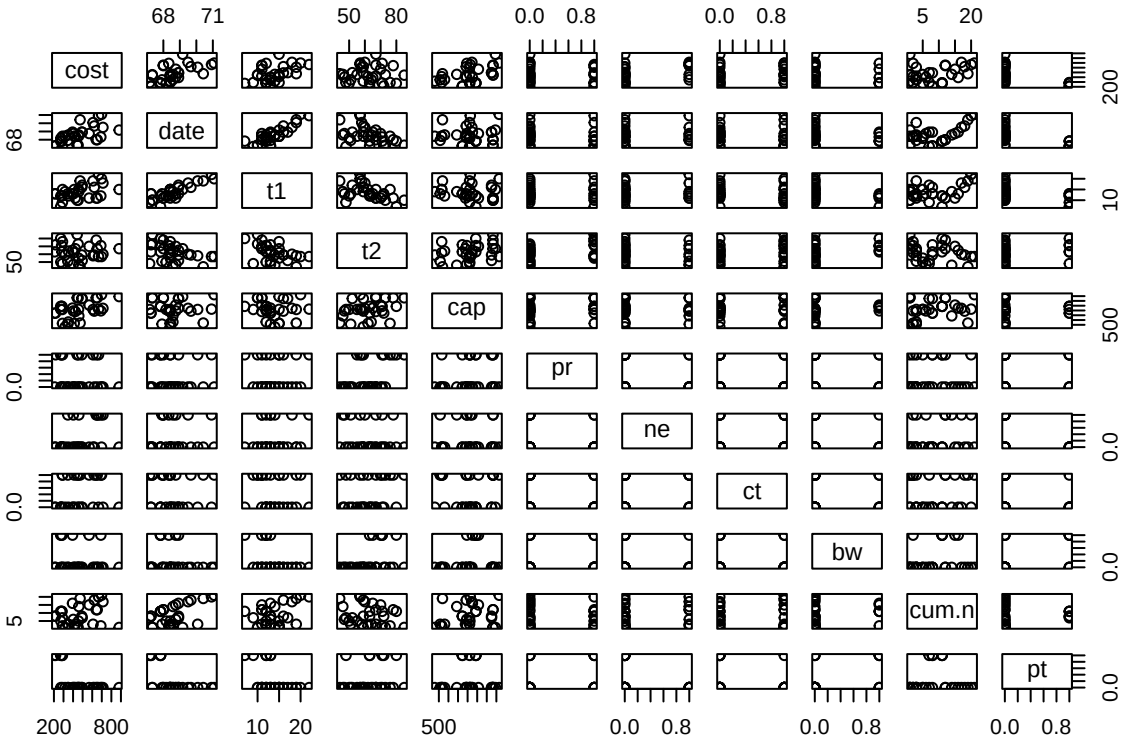
Name	**Continous Variables**
cost:	The capital cost of construction in millions of dollars adjusted to 1976 base.
date:	The date on which the construction permit was issued. The data are measured in years since January 1 1990 to the nearest month.
t1:	The time between application for and issue of the construction permit.
t2:	The time between issue of operating license and construction permit.
cap:	The net capacity of the power plant (MWe).
cum.n:	The cumulative number of power plants constructed by each architect-engineer.
Name	**Categorical Variables**
pr:	A binary variable where 1 indicates the prior existence of a LWR plant at the same site.
ne:	A binary variable where 1 indicates that the plant was constructed in the north-east region of the U.S.A.
ct:	A binary variable where 1 indicates the use of a cooling tower in the plant.
bw:	A binary variable where 1 indicates that the nuclear steam supply system was manufactured by Babcock-Wilcox.
pt:	A binary variable where 1 indicates those plants with partial turnkey guarantees.

a) Loading the dataset

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

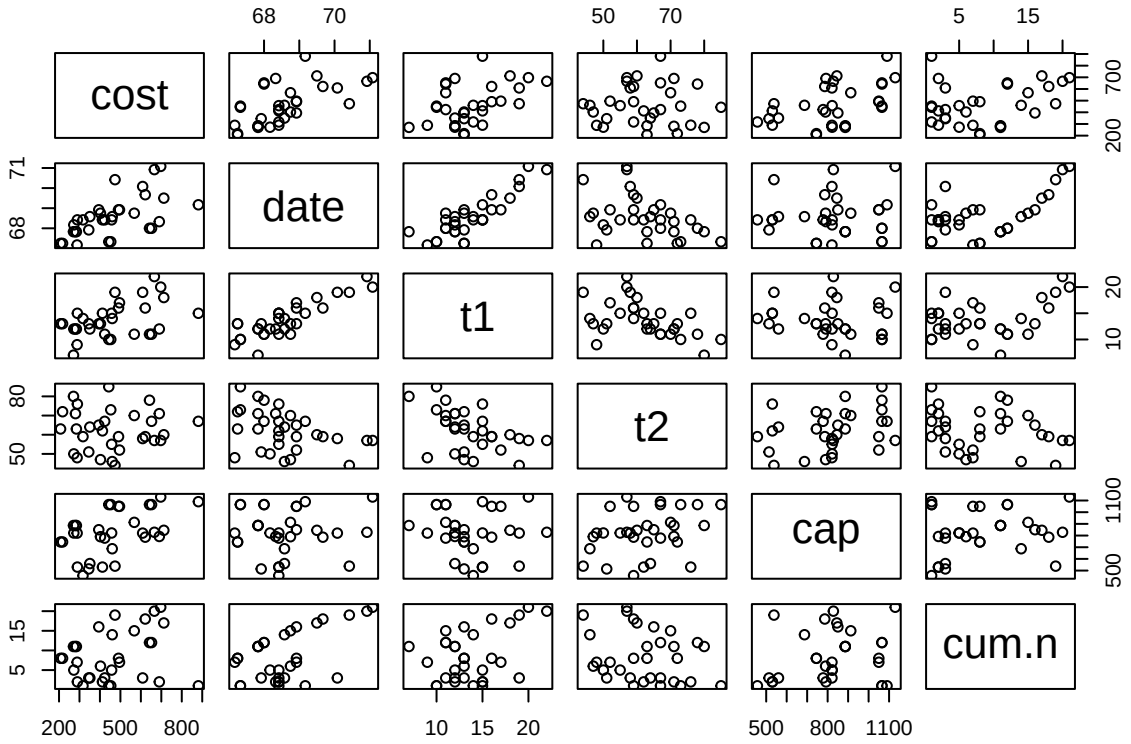
Generating some plots to gain an intuition for the dataset.

```
plot(nuclear)
```



This is very cluttered and hard to grasp, much due to categorical variables not giving any immeadiate insight. Therefore we will plot the dataset only regarding the continous variables.

```
cont_var <- c("cost", "date", "t1", "t2", "cap", "cum.n")
plot(nuclear[cont_var])
```



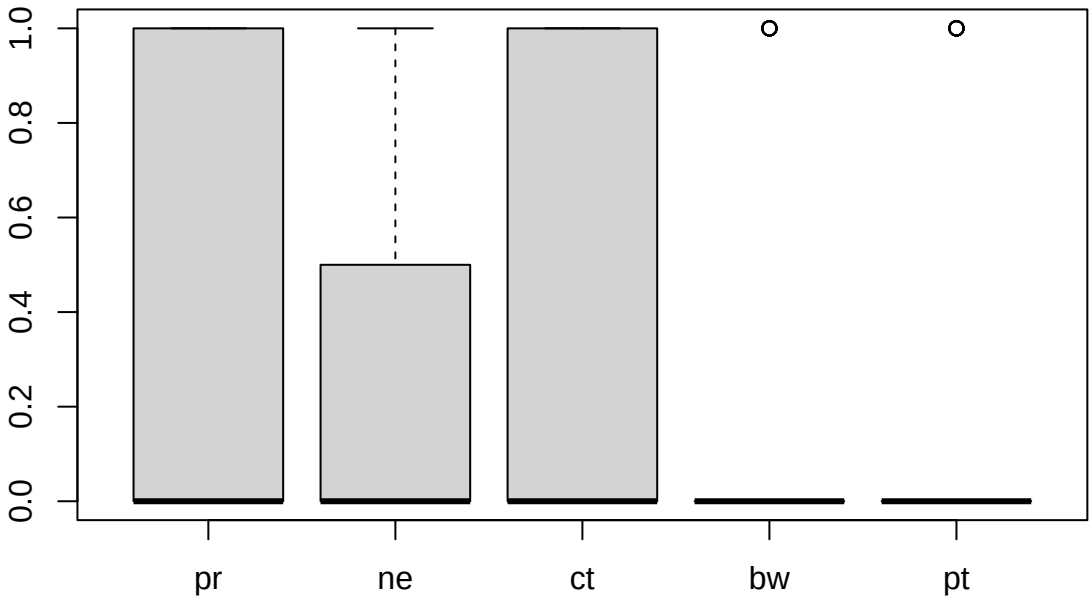
From this figure we can see some clear trends. We can see the cost of the LWRs are increasing over time, which can be attributed to inflation. A very positive correlation is also spotted in the t1 vs date indicating that the time between application and issue of the construction permits is steadily increasing and almost doubles at the end of the dataset. t2 vs date shows however a negative correlation, indicating that it takes shorter and shorter time from obtaining operating license to gaining construction permit.

The net capacity of the LWR, cap, does show a slight increase with cost. This is expected as the capital cost of the investment should match it's returns.

Box plots for the categorical variables

```
# Box plots
cat_var <- c("pr", "ne", "ct", "bw", "pt")

boxplot(nuclear[cat_var])
```



b) Constructing a model

Look at the model

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{p,i} + \epsilon_i$$

where Y_i is cost at log scale for observation i.

The standard assumptions about the noise term ϵ_i are that it is an independent, identically distributed, random variable from a Gaussian distribution $N(0, \sigma^2)$. It also assumed that the variance of the noise is constant $\text{Var}[\epsilon_i] = \sigma^2$ and that the expectance is $\mathbb{E}[\epsilon_i] = 0$.

```
#fit with all covariates
model2 = lm(log(cost)~., data=nuclear)
summary(model2)
```

```
##
## 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
```

The model on the log-transformed cost shows an R^2 value of 0.799. Although there are alot of covariates which have high p-values and therefore does not help the model with more accurate predictions. As seen from the plots in problem a), the net capacity has a strong, linear relationship with the capital cost of the power plant.

c) Removing covariates with high P-value

Removing the highest corresponding P-value, i.e. t1 gives a better result as a high P-value means that the outcomes in $log(cost)$ cannot be sufficiently explained by the changes in $t1$. When doing this we get

```
#Fit with t1 taken away.
model2 = lm(log(cost)~.-t1,data=nuclear)
summary(model2)
```

```
##
## Call:
## lm(formula = log(cost) ~ . - t1, 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
```

An increase from 13.28 to 15.42 in the F-statistic (explain F-statistic), increase of the multiple R-squared from 0.8635 to 0.8631 and likewise 0.7985 to 0.8072 for the adjusted R-squared.

d) Filtering on p-values

Iterating over all the covariates to remove those that do not have a p-value over 0.05.

```
# Iteratively removing the highest covariate with p-value over 0.05
variables <- names(nuclear)[-1]
target <- "log(cost)"

removed_variables = c("-t1")
for (i in 1:(length(variables)-1)){
  # Defining the formula
  f <- as.formula(
    paste(target, paste(removed_variables,
                        collapse= " - "),
                        sep = " ~."))

  # Computing the model
  filtermodel <- lm(f, data=nuclear)

  # Saving R2 coefficients

  # Finding the covariate with the highest p value
  p_values <- summary(filtermodel)$coefficients[2:(length(variables)-length(removed_variables)),4]

  max_p_ind <- which.max(p_values)
  max_p_name <- names(p_values[max_p_ind])

  # Checking if the highest p value is higher than 0.05
  if (p_values[max_p_ind]>0.05){
    removed_variables <- append(removed_variables, max_p_name)

  } else {
    break
  }
}

summary(filtermodel)
```

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

```
## lm(formula = f, data = nuclear)
##
## 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
```

```
dev.new(width=5, height=5, unit="cm")

plot(filtermodel)
```

Normal Q-Q plot indicates some datapoints as outliers for a normally distributed assumption.

Scale Location plot shows that the variance of the residuals are somewhat constant, however with an increase towards higher fitted values.

Residuals vs Leverage plot shows very horizontal trend, although with two datapoints exhibiting high leverage.

e) Model based on Mean Squared Error

Evaluating the model with Mean Squared Error ($\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$):

```
y.hat <- predict(filtermodel, nuclear)
y <- log(nuclear$cost)

MSE <- sum((y-y.hat)^2)/length(y)
MSE
```

```
## [1] 0.02635124
```

Evaluating the model with the average quadratic error we get on average a squared difference of 0.026. But this does not tell us everything about the model. Additionally, computing the R^2 metric gives us a indication of how the variance in the data is transferred to the model.

```
R2 <- 1- sum((y-y.hat)^2)/sum((y-mean(y))^2)
R2
```

```
## [1] 0.8095693
```

```r
# Or:
var(y.hat)/var(y)
```

```
## [1] 0.8095693
```

With a R^2 of 0.809 which tells us that the model explains 80.9 of the variance in data.

f) Differences between confidence and predict commands

Given a new datapoint $\mathbf{x}*$, we want to find out the $\theta = \mathbb{E}[Y|\mathbf{x}^*]$ as well as $\eta = \mathbb{E}[\exp(Y)|\mathbf{x}^*]$.

Predicting the datapoint from the problem with two different intervals: confidence and predict

```r
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)

interval.conf <- predict(filtermodel, d.new, interval="confidence")
interval.pred <-predict(filtermodel, d.new, interval="predict")

interval.conf
```

```
##        fit      lwr      upr
## 1 5.876308 5.639454 6.113161
```

```r
interval.pred
```

```
##        fit      lwr      upr
## 1 5.876308 5.443199 6.309416
```

For both the interval choices, the fit is the same. However the confidence intervals upper and lower bounds are narrower than the prediction.

The confidence interval portrays the uncertainty regarding the mean of the prediction values, and as $\mathbb{E}[\epsilon] = 0$, the random, gaussian noise gets omitted from the calculations. The prediction interval however portrays the uncertainty of a single, predicted point. Therefore it takes into account the random noise ϵ.

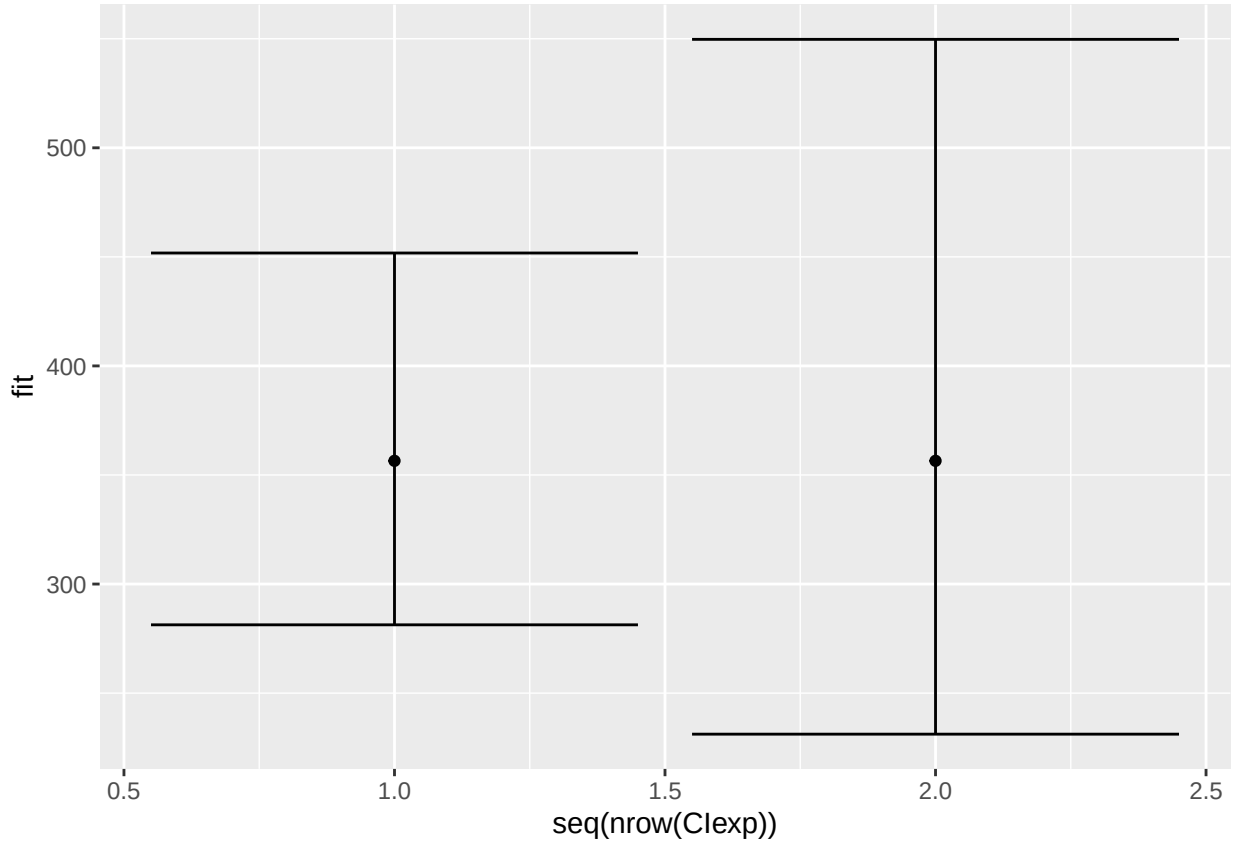
g) Constructing log intervals on non-logarithmic scale

Plotting the intervals from the previous problem onto non-logscale:

```r
CI <- data.frame(rbind(interval.conf, interval.pred))
CIexp <- exp(CI)
CIexp
```

```
##           fit      lwr      upr
## X1   356.4905 281.3092 451.7644
## X1.1 356.4905 231.1806 549.7237
```

```r
ggplot(data=CIexp, aes(x=seq(nrow(CIexp)), y=fit))+
  geom_point() +
  geom_errorbar(aes(ymin = lwr, ymax = upr))
```



h) Lasso regression

Lasso regression on the data set is shown below.

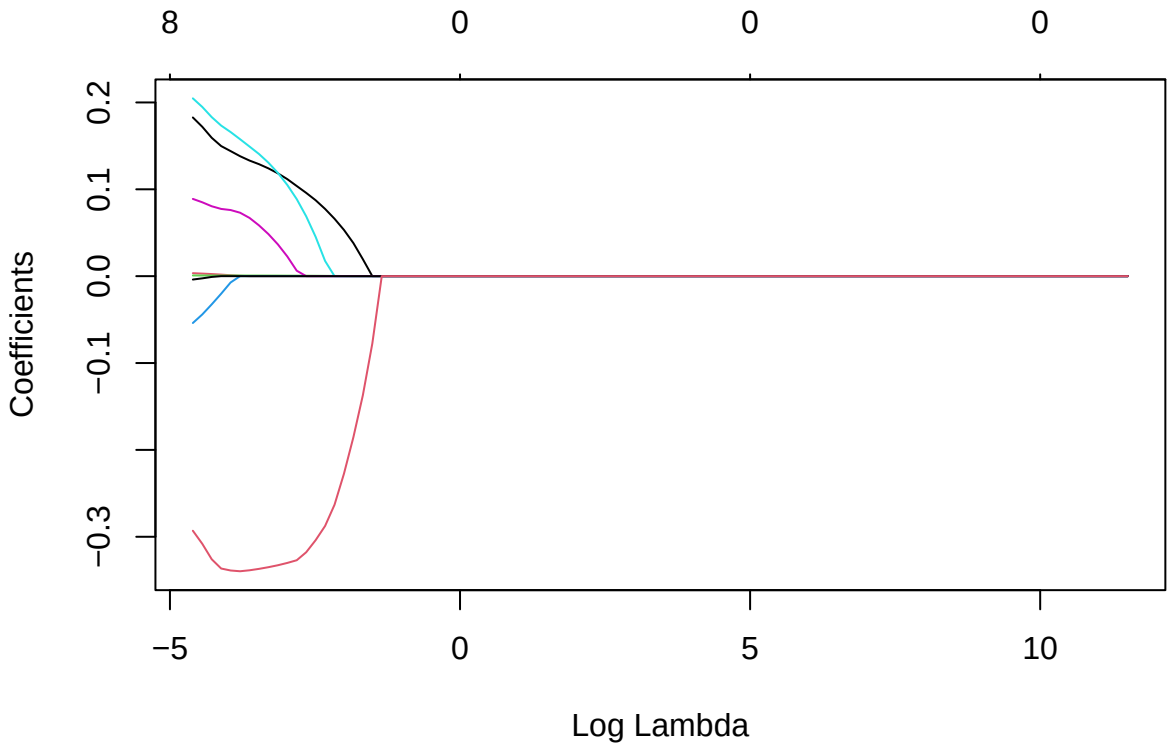
```
# Lasso regression
library(glmnet)

grid <- 10^seq(5, -2, length=100)

x <- nuclear[ ,-1]
y <- log(nuclear[ , 1])

# Fitting with Lasso Regression : alpha=1 equals the l1 norm
lasso.model <- glmnet(x,y, alpha=1, lambda=grid)

#Plotting estimates for different values of lambda
plot(lasso.model, xvar="lambda")
```



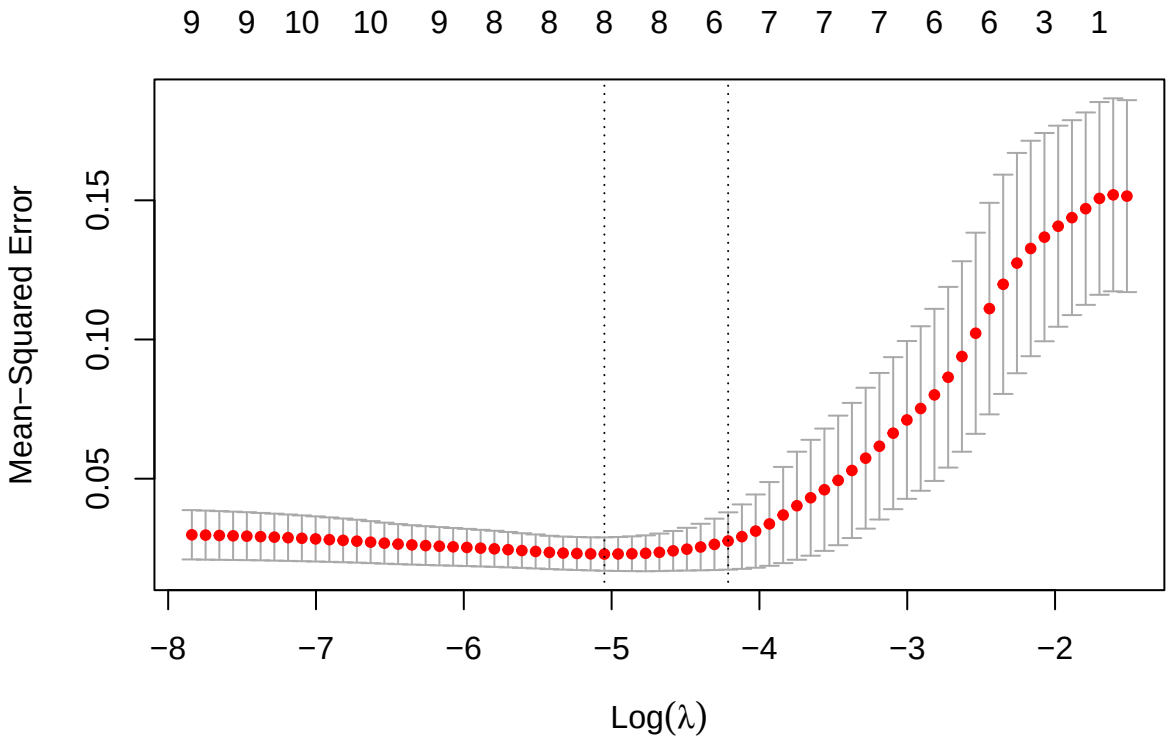

```
#Dividing into training/test set
set.seed(1)
train=sample(1:nrow(x),nrow(x)/2)
test=(-train)

x.train = as.matrix(x[train,])
x.test = as.matrix(x[test,])
y.train = y[train]
y.test=y[test]

#Cross-validation
set.seed(1)
cv.out <- cv.glmnet(x.train, y.train, alpha=1)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
plot(cv.out)
```



After cross-validation for the selection of the penalty parameter, the variables included in the final model are everyone expect ct and bw, as shown below. Comparing with the previous model, the lasso regression has a much lower intercept but has not omitted as many covariates.

```
coef(cv.out, cv.out$lambda.min)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) -20.967584780
## date          0.366020027
## t1            0.007059889
## t2            0.016420106
## cap           0.001026460
## pr           -0.239125848
## ne            0.389224827
## ct            .
## bw            .
## cum.n        -0.001830142
## pt           -0.190614011
```

```
filtermodel$coefficients
```

```
##   (Intercept)          date           cap            ne            pt
## -4.5035539482  0.1439104285  0.0008782741  0.2024363762 -0.3964877791
```

Problem 2.

a) Reading data

Getting the data

```
datadir = "http://www.uio.no/studier/emner/matnat/math/STK2100/data/"
Fe <- read.table(paste(datadir, "fe.dat", sep=""), header=T, sep=",")
```

Running the code from the problem with contrasts:

```
options(contrasts=c("contr.treatment", "contr.treatment"))

fit1 <- lm(Fe ~ form, data=Fe)
summary(fit1)
```

```
##
## 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 model behaves poorly with an R^2 of around 0.364. The model only assumes there is one covariate, which is form, and that it ranges between 1,2,3 and 4.

Transforming the "form" column into factors. This lets the model know there are four different covariates, denoted by the dummy variables "form"

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

```
##
## 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
```

The resulting R^2 has now improved up to 0.431.

b) Constraints

As explained in the text, the "options(contrasts=c("contr.treatment", "contr.treatment"))" code will set the constraint that $\beta_1 = \hat{\beta}_1 = 0$.

This is needed as to portray the relative differences of iron content between the different formations, without discarding the intercept. The interpretation of the other β_j parameters is therefore how much more (or less, if negative) iron are in the formations relative to formation 1 (carbonates).

c) Alternative Constraint

An alternative constraint is to put $\beta_0 = 0$, i.e. removing the intercept. This can be obtained by

```
fit2 <- lm(Fe~form+0, data=Fe)

summary(fit2)
```

```
##
## Call:
## lm(formula = Fe ~ form + 0, 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
```

By setting the intercept, $\beta_0 = 0$, the β_j is then interpreted as the mean of the iron content in their respective formations.

d) Sum of coefficients equals zero

Using the "contrast=c("contr.sum", "contr.sum")" imposes the constraint $\sum_{j=1}^{K} \beta_j = 0$. However, as the summary of the fit only displays three of the β_j, the last one can be found by taking the negative of the sum of all the coeffiecients expect the intercept.

```
options(contrasts=c("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
```

```
beta4 <- -sum(fit3$coefficients[2:4])
```

e) Differences between formations

We now have three different models which all are based on constraints on some or more of the β_j. Printing out the coefficients for all the different models gives us the following

```
print("Model 1 - First Coeff Zero")
```

```
## [1] "Model 1 - First Coeff Zero"
```

```
model1 <- fit1$coefficients[2:4]
intercept <- fit1$coefficients[1]
print(setNames(c(intercept, 0,model1), c("(intercept)","form1", names(model1))))
```

```
## (intercept)       form1       form2       form3       form4
##       26.08        0.00       -1.39        3.87        7.76
```

```r
print("Model 2 - no intercept")
```

```
## [1] "Model 2 - no intercept"
```

```r
model2 <- fit2$coefficients
print(setNames(c(0,model2), c("(intercept)", names(model2))))
```

```
## (intercept)       form1       form2       form3       form4
##        0.00       26.08       24.69       29.95       33.84
```

```r
print("Model 3 - sum of coeffs zero ")
```

```
## [1] "Model 3 - sum of coeffs zero "
```

```r
model3 <- fit3$coefficients
print(setNames(c(model3, beta4),c(names(model3), "form4")))
```

```
## (Intercept)       form1       form2       form3       form4
##       28.64       -2.56       -3.95        1.31        5.20
```

The different models are different combinations of the β_j. The first model shows the intercept as the iron content of carbonate (form1), while the other coefficients point to how much mure iron there are in the differing formations.

The second model just shows the average iron content across all the formation.

The third model shows the average difference in iron in the formation relative to the average iron in the entire dataset.

As to which model is better to convey that there exists iron differences, it is somewhat subjective, however I would prefer the third model.

f) Predicting on four datapoitns

Creating four new datapoints and predicting upon them

```r
newdata = data.frame(form=as.factor(c(1,2,3,4)))

newdata
```

```
##   form
## 1    1
## 2    2
## 3    3
## 4    4
```

```r
pred1 = predict(fit1, newdata)
pred2 = predict(fit2, newdata)
pred3 = predict(fit3, newdata)

pred1
```

```
##     1     2     3     4
## 26.08 24.69 29.95 33.84
```

```
pred2
```

```
##     1     2     3     4
## 26.08 24.69 29.95 33.84
```

```
pred3
```

```
##     1     2     3     4
## 26.08 24.69 29.95 33.84
```

All these predictions are the same, as the models are essentially similiar as discussed in the previous section, and only differs by how large the intercept is.

g) Summary ouputs of the model

The summary outputs of the different models are shown below. As the second model which omits an intercept gives the best R^2 score, it should suffice to simplify the model to this extent i.e. basing the model only on the means of the iron content for the respecitve formations.

```
summary(fit1)
```

```
##
## 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
```

```
summary(fit2)
```

```
##
## Call:
## lm(formula = Fe ~ form + 0, 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
```

```
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
```