STK2100 Oblig 1

Håkon Kvernmoen

2/4/2021

Problem 1

a)

First we need to load the data. The code sample for loading did not work for me (got a 400 bad request error). Assuming the datafile "nuclear.dat" is located in the same folder as this file, we load the data and attach it for easier use.

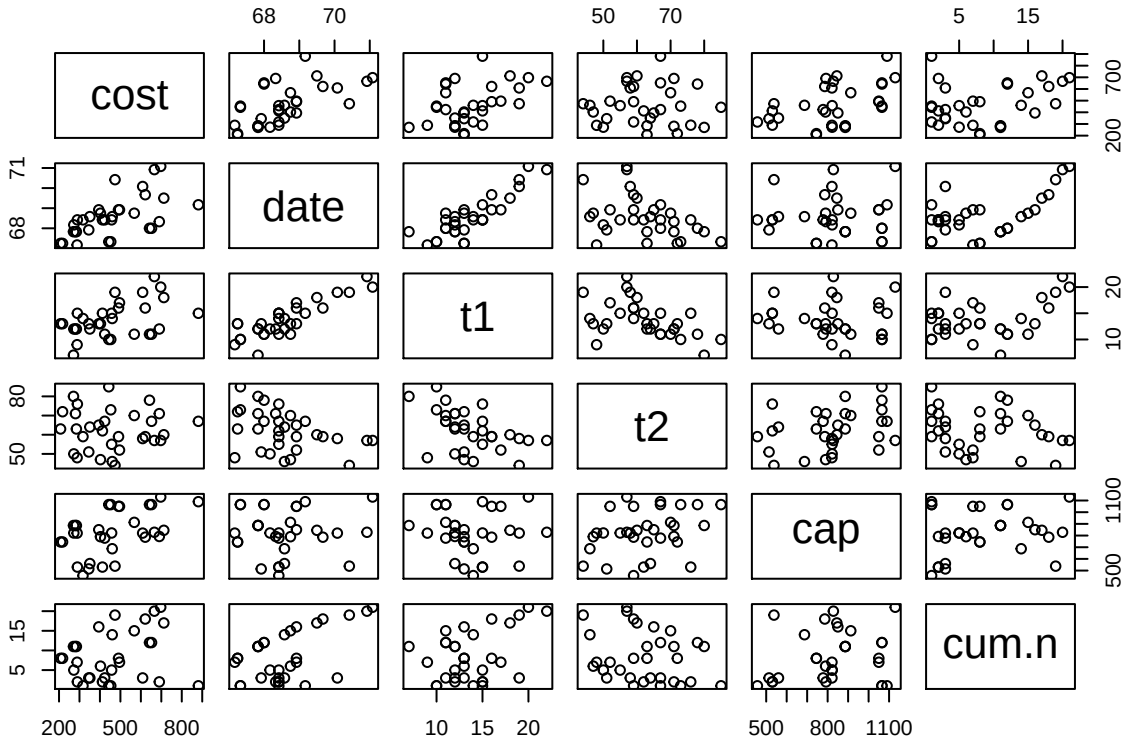
```
nuclear = read.table("nuclear.dat", sep="\t", header=T)
attach(nuclear)
```

We notice that `pr`, `ne`, `ct`, `bw` and `pt` are binary variables, so we set them as factors.

```
nuclear$pr = as.factor(nuclear$pr)
nuclear$ne = as.factor(nuclear$ne)
nuclear$ct = as.factor(nuclear$ct)
nuclear$bw = as.factor(nuclear$bw)
nuclear$pt = as.factor(nuclear$pt)
```

To investigate the data we plot the numerical features against each other. There seems to be some correlation between `date` and `t1`

```
plot(nuclear[,sapply(nuclear, is.numeric)])
```



b)

The standard assumption on the noise terms ϵ_i are.

1. The error terms are normally distributed with a mean of 0
2. The variance σ^2 of this normal distribution is constant
3. The error terms are independent, ϵ_i does not influence ϵ_j

Important?

We will now try to fit the model using all the features. As cost is always positive, we fit the log of the cost as a response variable. With y_i being the i'th observation of the cost, we will try to fit the model.

$$log(y_i) = \beta_0 + \sum_{j=1}^{p} x_{i,j} + \epsilon_i$$

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

```
##
## 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 ***
## pr1         -1.081e-01  8.351e-02  -1.295  0.20943
## ne1          2.595e-01  7.925e-02   3.274  0.00362 **
## ct1          1.155e-01  7.027e-02   1.644  0.11503
## bw1          3.680e-02  1.063e-01   0.346  0.73261
## cum.n       -1.203e-02  7.828e-03  -1.536  0.13944
## pt1         -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
```

c)

We will now remove the term with the largest P-value. Observing the summary of the linear model, we see that `t1` has the largest P-value at 0.81610. This is sensible to do since the P-value is a measure of the correctness of the null-hypothesis (H_0). A large P-value as in this case indicates that there is a very little statistical basis for `t1` to be a good predictor for log(`cost`) and is thus neglected.

```
all_no_t1.fit = lm(log(cost)~.-t1, data= nuclear)
summary(all_no_t1.fit)
```

```
##
## 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 ***
## pr1         -1.035e-01  7.944e-02  -1.303 0.205922
## ne1          2.607e-01  7.738e-02   3.368 0.002772 **
## ct1          1.142e-01  6.853e-02   1.667 0.109715
## bw1          2.622e-02  9.423e-02   0.278 0.783401
## cum.n       -1.220e-02  7.626e-03  -1.599 0.124034
```

```
## pt1         -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
```

We observe that there are some change in the P-values for a lot of the features after we excluded `t1`. This is probably due to correlation between the features. We would ideally have linearly independent explanatory variables. In example a change in `cap` should not influence any of the other explanatory variables, but this is not the case. On the other hand, the changes in P-values are not huge and the coefficients estimates seems relatively unchanged. In addition the standard error for the coefficients seems to decrease and we continue these modifications.

d)

We now want to fit our model, remove the explanatory variable with a P-value larger than 0.05 and repeat this until we have a model where all explanatory variables have P-values smaller than 0.05. We then implement a backward substitution algorithm. We note that we do not want to remove the intercept even tough its P-value can be larger than 0.05.

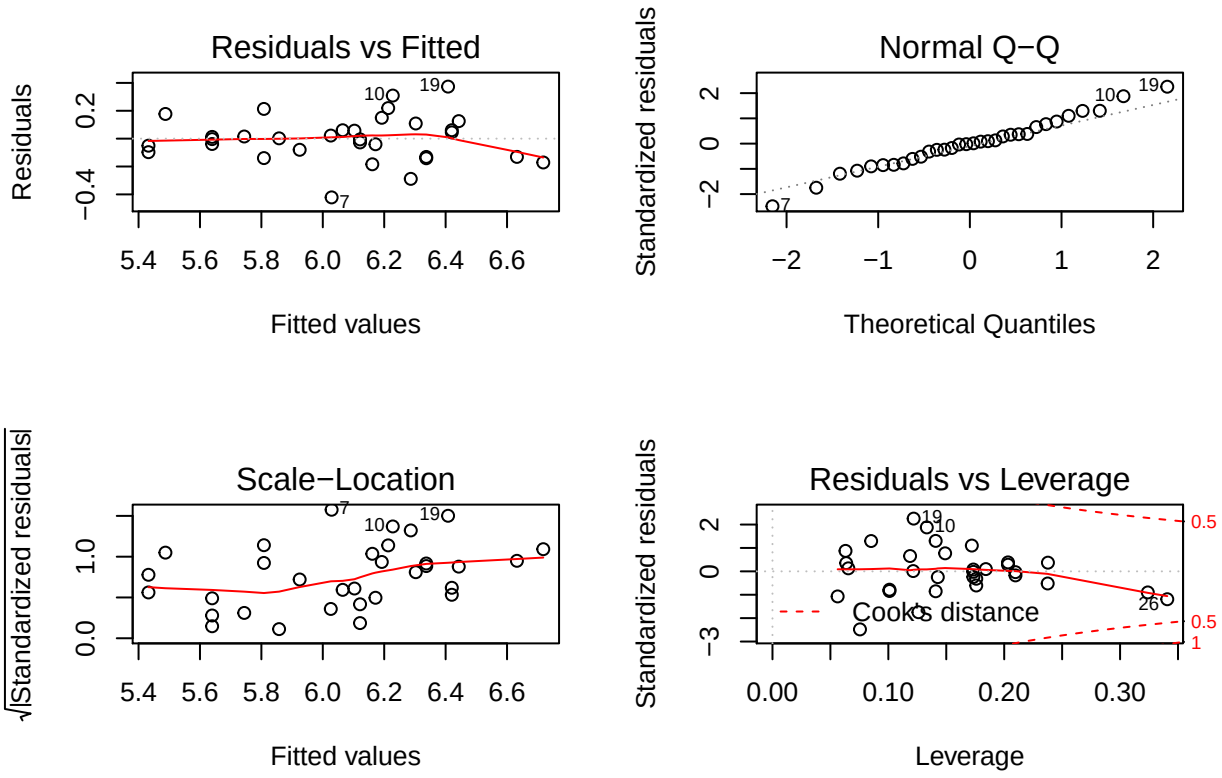
```
nuclear_backwards_sub <- data.frame(nuclear)
for (i in 1:ncol(nuclear)) {
    fit <- lm(log(cost)~., data=nuclear_backwards_sub)
    # -1 since we don't want to remove intercept
    p_vals <- summary(fit)$coefficients[-1,4]
    max_idx <- as.integer(which.max(p_vals))

    if(p_vals[max_idx] < 0.05) {
      break
    }
    else {
      # Add one since we don't want to remove targert variable (cost).
      nuclear_backwards_sub <- nuclear_backwards_sub[,-(max_idx+1)]
    }
}

summary(fit)
```

```
##
## Call:
## lm(formula = log(cost) ~ ., data = nuclear_backwards_sub)
##
## 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 ***
## ne1          0.2024364  0.0751953   2.692 0.012042 *
## pt1         -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
```

We are then left with 4 explanatory variables. Two of them continues (`date, cap`) and two binary (`ne, pt`).

```
par(mfrow=c(2,2))
plot(fit)
```



The model seems to fit reasonably well, but the simple linear model does not seem to capture the complexity of the data. From the residuals against fitted values plot we see that the residual mean seems to be around 0 for the lower fitted values, but for higher values $\log(\texttt{cost}) > 6.1$ the mean seems to not be centered around 0. This is probably due to a small data set. The QQ-plot confirms this displaying that data point 7, 10 and 19 deviates a lot from the theoretical quantiles. From the Standardized residuals against leverage plot point 10 and 19 again shows a high deviance and should probably be considered outliers. Lastly point 26 deviates from the mean of zero as well as having a high leverage (influence on the model) and should also be considerd as an outlier.

e)

Finally we calculate the MSE

```
MSE = (cost-exp(fitted.values(fit)))^2
MSE = mean(MSE)
MSE
```

```
## [1] 7443.737
```

This MSE is high and most of the contribution comes from the data points discussed in d). There is also a major drawback to this approach, as we have tested the correctness of our model (using the MSE) on the same data that we used to fit the model. The MSE resulted in a large value either way, but this should generally be avoided as more flexible models can follow the data used too closely (overfitting) and not represent the whole population. A better approach would be to split the data into a training and testing set, using only the training data to fit the model and then evaluate the correctness using the training data. Other techniques such as cross-validation can also be used.

g)

We first use backward selection with the AIC.

```
library("MASS")
stepAIC(all.fit, direction = "backward")
```

```
## Start:  AIC=-105.01
## log(cost) ~ date + t1 + t2 + cap + pr + ne + ct + bw + cum.n +
##     pt
##
##         Df Sum of Sq     RSS      AIC
## - t1     1   0.00160 0.60603 -106.930
## - bw     1   0.00345 0.60788 -106.832
## <none>               0.60443 -105.014
## - t2     1   0.04284 0.64727 -104.823
## - pr     1   0.04826 0.65269 -104.556
## - cum.n  1   0.06792 0.67235 -103.607
## - ct     1   0.07781 0.68224 -103.140
## - pt     1   0.08337 0.68781 -102.879
## - date   1   0.19899 0.80343  -97.907
## - ne     1   0.30859 0.91302  -93.815
## - cap    1   0.68497 1.28940  -82.770
##
## Step:  AIC=-106.93
## log(cost) ~ date + t2 + cap + pr + ne + ct + bw + cum.n + pt
##
##         Df Sum of Sq     RSS      AIC
## - bw     1   0.00213 0.60816 -108.818
## <none>               0.60603 -106.930
## - t2     1   0.04135 0.64738 -106.818
## - pr     1   0.04680 0.65283 -106.550
## - cum.n  1   0.07045 0.67648 -105.411
## - ct     1   0.07654 0.68257 -105.124
```

```
## - pt     1   0.08216 0.68819 -104.862
## - ne     1   0.31255 0.91858  -95.621
## - date   1   0.54190 1.14793  -88.489
## - cap    1   0.68916 1.29518  -84.627
##
## Step:  AIC=-108.82
## log(cost) ~ date + t2 + cap + pr + ne + ct + cum.n + pt
##
##         Df Sum of Sq     RSS      AIC
## <none>               0.60816 -108.818
## - pr     1   0.05738 0.66554 -107.932
## - t2     1   0.06379 0.67195 -107.626
## - cum.n  1   0.06839 0.67656 -107.407
## - ct     1   0.07440 0.68257 -107.124
## - pt     1   0.08066 0.68882 -106.832
## - ne     1   0.31375 0.92192  -97.505
## - date   1   0.54592 1.15408  -90.318
## - cap    1   0.68739 1.29556  -86.617
```

```
##
## Call:
## lm(formula = log(cost) ~ date + t2 + cap + pr + ne + ct + cum.n +
##     pt, data = nuclear)
##
## Coefficients:
## (Intercept)         date           t2          cap          pr1          ne1
##  -1.169e+01    2.438e-01    6.018e-03    8.739e-04   -1.099e-01    2.611e-01
##         ct1        cum.n          pt1
##   1.111e-01   -1.176e-02   -2.071e-01
```

Then the same with BIC

```
n = nrow(nuclear)
stepAIC(all.fit, direction="backward", k=log(n))
```

```
## Start:  AIC=-88.89
## log(cost) ~ date + t1 + t2 + cap + pr + ne + ct + bw + cum.n +
##     pt
##
##         Df Sum of Sq     RSS     AIC
## - t1     1   0.00160 0.60603 -92.273
## - bw     1   0.00345 0.60788 -92.175
## - t2     1   0.04284 0.64727 -90.166
## - pr     1   0.04826 0.65269 -89.899
## - cum.n  1   0.06792 0.67235 -88.949
## <none>               0.60443 -88.891
## - ct     1   0.07781 0.68224 -88.482
## - pt     1   0.08337 0.68781 -88.222
## - date   1   0.19899 0.80343 -83.250
## - ne     1   0.30859 0.91302 -79.158
## - cap    1   0.68497 1.28940 -68.113
##
## Step:  AIC=-92.27
```

```
## log(cost) ~ date + t2 + cap + pr + ne + ct + bw + cum.n + pt
##
##         Df Sum of Sq     RSS     AIC
## - bw     1   0.00213 0.60816 -95.626
## - t2     1   0.04135 0.64738 -93.626
## - pr     1   0.04680 0.65283 -93.358
## <none>               0.60603 -92.273
## - cum.n  1   0.07045 0.67648 -92.219
## - ct     1   0.07654 0.68257 -91.933
## - pt     1   0.08216 0.68819 -91.670
## - ne     1   0.31255 0.91858 -82.430
## - date   1   0.54190 1.14793 -75.297
## - cap    1   0.68916 1.29518 -71.435
##
## Step:  AIC=-95.63
## log(cost) ~ date + t2 + cap + pr + ne + ct + cum.n + pt
##
##         Df Sum of Sq     RSS     AIC
## - pr     1   0.05738 0.66554 -96.207
## - t2     1   0.06379 0.67195 -95.900
## - cum.n  1   0.06839 0.67656 -95.681
## <none>               0.60816 -95.626
## - ct     1   0.07440 0.68257 -95.398
## - pt     1   0.08066 0.68882 -95.106
## - ne     1   0.31375 0.92192 -85.779
## - date   1   0.54592 1.15408 -78.592
## - cap    1   0.68739 1.29556 -74.892
##
## Step:  AIC=-96.21
## log(cost) ~ date + t2 + cap + ne + ct + cum.n + pt
##
##         Df Sum of Sq     RSS     AIC
## - t2     1   0.02447 0.69001 -98.517
## - cum.n  1   0.05351 0.71905 -97.198
## <none>               0.66554 -96.207
## - ct     1   0.10237 0.76791 -95.094
## - pt     1   0.12015 0.78570 -94.361
## - ne     1   0.28784 0.95339 -88.171
## - date   1   0.49109 1.15664 -81.987
## - cap    1   0.68019 1.34573 -77.141
##
## Step:  AIC=-98.52
## log(cost) ~ date + cap + ne + ct + cum.n + pt
##
##         Df Sum of Sq     RSS     AIC
## - cum.n  1   0.06006 0.75007 -99.312
## <none>               0.69001 -98.517
## - pt     1   0.11719 0.80720 -96.963
## - ct     1   0.12931 0.81932 -96.486
## - ne     1   0.27215 0.96216 -91.343
## - date   1   0.46672 1.15673 -85.450
## - cap    1   0.89456 1.58457 -75.379
##
## Step:  AIC=-99.31
```

```
## log(cost) ~ date + cap + ne + ct + pt
##
##        Df Sum of Sq     RSS     AIC
## <none>              0.75007 -99.312
## - ct    1   0.09317 0.84324 -99.031
## - ne    1   0.21478 0.96485 -94.720
## - pt    1   0.37487 1.12494 -89.807
## - date  1   0.55668 1.30675 -85.013
## - cap   1   0.83451 1.58458 -78.845

##
## Call:
## lm(formula = log(cost) ~ date + cap + ne + ct + pt, data = nuclear)
##
## Coefficients:
## (Intercept)         date          cap          ne1          ct1          pt1
##  -5.4058393    0.1563992    0.0008674    0.1973468    0.1154229   -0.3477717
```

Both AIC and BIC starts with the full model and tries to remove one. The model with the highest likelihood is chosen and the cycle continues until there is no improvement in AIC-value. The two models we ended up with was

AIC: `date,t2, cap, pr, ne, ct, cum.n, pt`

BIC: `date, cap, ne, ct, pt`

h)

g)

P-values does not take complexity into account.