STK2100
Oblig 1

Egil Furnes
Studentnummer: 693784

Oppgave 1

a)

i)

The only model that can be written as a linear model as is, is model 2.

$$2. \quad Y = \beta_0 + \frac{\beta_1}{x} + \beta_2 x^2 + \epsilon$$

ii)

For model 4 we can fix β_2 to a constant, say c and get

$$4. \quad \beta_0 + \beta_1 x^c + \epsilon$$

iii)

For model 5 we can log-transform such as this:

$$Y = \beta_0 x^{\beta_1} \varepsilon$$

$$\log(Y) = \log(\beta_0 x^{\beta_1} \varepsilon)$$
$$\log(Y) = \log(\beta_0) + \log(x^{\beta_1}) + \log(\varepsilon)$$
$$\log(Y) = \log(\beta_0) + \beta_1 \log(x) + \log(\varepsilon)$$

b)

1. $X = \begin{bmatrix} \frac{1}{1+x_i} & x_i^{1/2} \end{bmatrix},\qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_2 \end{bmatrix}$
2. $X = \begin{bmatrix} 1 & \frac{1}{x_i} & x_i^2 \end{bmatrix},\qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$
3. $X = \begin{bmatrix} 1 & x_i & x_i^2 \end{bmatrix},\qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$
4. $X = \begin{bmatrix} 1 & x_i \end{bmatrix},\qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1' \end{bmatrix}$
5. $X = \begin{bmatrix} 1 & \log(x_i) \end{bmatrix},\qquad \beta = \begin{bmatrix} \beta_0' \\ \beta_1 \end{bmatrix}$

Oppgave 2

a)

```r
# set seed for replication!
set.seed(1814)

# loading packages
library(tidyverse)

# reading data
nuclear <- read_delim("nuclear.dat")

# a)

# fitting the data using linear regression
lm1 <- lm(log(cost)~., nuclear)
lm1 %>% summary()

# creating a 95 percent confidence interval
confint(lm1, level = 0.95)[c("t1","t2","bw"),]
```

A 95% confidence interval is then found to be the following.

```
          2.5 %     97.5 %
t1 -0.041123331 0.05162679
t2 -0.003949911 0.01516185
bw -0.184211843 0.25780411
```

b)

```r
# b)

# creating a new dataframe
df <- tibble(
  date = 70.0,
  t1 = 13,
  t2 = 50,
  cap = 800,
  pr = 1,
  ne = 0,
  ct = 0,
  bw = 1,
  cum.n = 8,
  pt = 1
)

```

```r
# predicting on data using linear regression
pred1 <- predict(lm1, newdata = df, interval = "prediction", level =
    0.95)

# retrieving the coefficients
yhat <- pred1[1,"fit"]
lwry <- pred1[1,"lwr"]
upry <- pred1[1,"upr"]

# transforming y to find z
zfit <- exp(yhat)
zlwr <- exp(lwry)
zupr <- exp(upry)

# saving coefficients for z
predz <- data.frame(
  fit = zfit,
  lwr = zlwr,
  upr = zupr
)

# presenting my findings
print(pred1)
print(predz)
```

The 95% prediction interval with the cost Z is then found to be the following.

```
> print(pred1)
       fit      lwr      upr
1 5.964135 5.394248 6.534022
> print(predz)
       fit      lwr      upr
1 389.2163 220.1366 688.1607
```

c)

```r
# c)

# individual t-tests
sum_lm1 <- summary(lm1)
print(sum_lm1$coefficients[c("t1", "t2", "bw"),
                           c("Estimate", "Pr(>|t|)"),
                           drop = FALSE])

# joint F-tests
lm1red <- lm(log(cost)~date+cap+pr+ne+ct+cum.n+pt, data = nuclear)
anova(lm1red, lm1)
```

Find the output for the individual t-test, where we find that the p-value is larger than 0.5 for all predictors t1, t2, and bw, such that we fail to reject the null-hypothesis $H_0 : \beta_j = 0$ that the predictors are significant.

```
      Estimate  Pr(>|t|)
t1 0.005251730 0.8160981
t2 0.005605968 0.2359862
bw 0.036796131 0.7326075
```

For the joint F-test, we find a p-value of $0.5173 > 0.5$ where we fail to reject the H_0 at a 5% confidence level.

```
Analysis of Variance Table

Model 1: log(cost) ~ date + cap + pr + ne + ct + cum.n + pt
Model 2: log(cost) ~ date + t1 + t2 + cap + pr + ne + ct + bw + cum.n +
    pt
  Res.Df     RSS Df Sum of Sq     F Pr(>F)
1     24 0.67195
2     21 0.60443  3   0.06752 0.782 0.5173
```

d)

```
# d)

library(leaps)

fwd1 <- regsubsets(log(cost) ~ ., data = nuclear, method = "forward"
    , nvmax = 10)
sum1 <- summary(fwd1)
sum1$outmat %>% as_tibble()

bic1 <- sum1$bic
bicb <- names(which(sum1$which[which.min(bic1), ]))[-1]

all1 <- names(nuclear)[-1]
aic1 <- which.max(sum1$adjr2)   # Fixed: max adj R²
aicb <- names(which(sum1$which[aic1, ]))[-1]

bicb
aicb

fitaic <- lm(log(cost) ~ ., data = nuclear[, c("cost", aicb)])
fitbic <- lm(log(cost) ~ ., data = nuclear[, c("cost", bicb)])

aic2 <- AIC(fitaic)
bic2 <- BIC(fitbic)
```

```
aic2
bic2
```

The output of the order matrix is below, which gives the following order of variables included: `cap` →`date` →`ne` →`ct` →`bw` →`pr` →`t2` →`t1`.

```
# A tibble: 10 × 10
   date  t1    t2    cap   pr    ne    ct    bw
   <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 " "   " "   " "   " "   " "   " "   " "   " "
 2 " "   " "   " "   "*"   " "   " "   " "   " "
 3 "*"   " "   " "   "*"   " "   " "   " "   " "
 4 "*"   " "   " "   "*"   " "   "*"   " "   " "
 5 "*"   " "   " "   "*"   " "   "*"   "*"   " "
 6 "*"   " "   " "   "*"   " "   "*"   "*"   " "
 7 "*"   " "   " "   "*"   " "   "*"   "*"   "*"
 8 "*"   " "   " "   "*"   "*"   "*"   "*"   "*"
 9 "*"   " "   "*"   "*"   "*"   "*"   "*"   "*"
10 "*"   "*"   "*"   "*"   "*"   "*"   "*"   "*"
#  2 more variables: cum.n <chr>, pt <chr>
```

Using forward selection I end up with variables `[1] "date" "cap" "ne" "ct" "pt"` selected on BIC, and with `"date" "t2" "cap" "pr" "ne" "ct" "bw" "cum.n" "pt"` selected on AIC. This yields respectively AIC and BIC:

```
> aic2
[1] -14.11792
> bic2
[1] -5.034235
```

e)

```
# e)

bwd2 <- regsubsets(log(cost) ~ ., data = nuclear, method = "backward
  ", nvmax = 10)
sum2 <- summary(bwd2)
sum2$outmat %>% as_tibble()

bic2 <- sum2$bic
bicc <- names(which(sum2$which[which.min(bic2), ]))[-1]

all2 <- names(nuclear)[-1]
aic2 <- which.max(sum2$adjr2)  # Fixed: max adj R²
aicc <- names(which(sum2$which[aic2, ]))[-1]
```

```

bicc
aicc

fitaic3 <- lm(log(cost) ~ ., data = nuclear[, c("cost", aicc)])
fitbic3 <- lm(log(cost) ~ ., data = nuclear[, c("cost", bicc)])

aic3 <- AIC(fitaic3)
bic3 <- BIC(fitbic3)

aic3
bic3
```

For backward selection we find the order `cap` →`date` →`ct` →`t2` →`pr` →`bw` →`t1`.

```
# A tibble: 10 × 10
   date  t1    t2    cap   pr    ne    ct    bw   
   <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 " "   " "   " "   " "   " "   " "   " "   " "  
 2 " "   " "   " "   "*"   " "   " "   " "   " "  
 3 "*"   " "   " "   "*"   " "   " "   " "   " "  
 4 "*"   " "   " "   "*"   " "   "*"   " "   " "  
 5 "*"   " "   " "   "*"   " "   "*"   "*"   " "  
 6 "*"   " "   " "   "*"   " "   "*"   "*"   " "  
 7 "*"   " "   "*"   "*"   " "   "*"   "*"   " "  
 8 "*"   " "   "*"   "*"   "*"   "*"   "*"   " "  
 9 "*"   " "   "*"   "*"   "*"   "*"   "*"   "*"  
10 "*"   "*"   "*"   "*"   "*"   "*"   "*"   "*"  
#   2 more variables: cum.n <chr>, pt <chr>
```

Using BIC we find the selected variables `"date" "cap" "ne" "ct" "pt"` and for AIC we find `"date" "t2" "cap" "pr" "ne" "ct" "cum.n" "pt"` , and their respective AIC and BIC:

```
> aic3
[1] -16.00549
> bic3
[1] -5.034235
```

f)

```
# f)

faic <- aicb
fbic <- bicb
baic <- aicc
bbic <- bicc
```

```r

cv <- function(vars, data, k=10){
  n = nrow(data)
  folds = sample(rep(1:k, length.out = n))
  errors = numeric(k)
  for (i in 1:k){
    train = data[folds != i, ]
    test = data[folds == i, ]
    fit = lm(log(cost)~., data = train[, c("cost", vars)])
    pred = predict(fit, newdata = test)
    errors[i] = mean((log(test$cost) - pred)^2)
  }
  mean(errors)
}

cv_faic <- cv(faic, nuclear)
cv_fbic <- cv(fbic, nuclear)
cv_baic <- cv(baic, nuclear)
cv_bbic <- cv(bbic, nuclear)
cvs <- c(cv_faic, cv_fbic, cv_baic, cv_bbic)
names(cvs) <- c("fwd-aic", "fwd-bic", "bwd-aic", "bwd-bic")
bestcv <- names(cvs)[which.min(cvs)]
bestcv
min(cvs)
```

Using K-folds cross validation with $K = 10$ I find that the backward selection using BIC gives the best model, and now prefer the model using `date, cap, ne, ct, pt`.

```
> bestcv <- names(cvs)[which.min(cvs)]
> bestcv
[1] "bwd-bic"
> min(cvs)
[1] 0.03565251
```

g)

```r
# g)

library(boot)

boot632 <- function(vars, data, B = 1000) {
  set.seed(1814)
  fit <- lm(log(cost) ~ ., data = data[, c("cost", vars)])
  err_app <- mean(resid(fit)^2)
  boot_fn <- function(data, indices) {
    d <- data[indices, ]
    fit_boot <- lm(log(cost) ~ ., data = d[, c("cost", vars)])
    pred <- predict(fit_boot, newdata = data)
```

```
    mean((log(data$cost) - pred)^2)
  }
  boot_res <- boot(data, boot_fn, R = B)
  err_boot <- mean(boot_res$t)
  0.368 * err_app + 0.632 * err_boot
}

boot_faic <- boot632(faic, nuclear)
boot_fbic <- boot632(fbic, nuclear)
boot_baic <- boot632(baic, nuclear)
boot_bbic <- boot632(bbic, nuclear)

boot_faic
boot_fbic
boot_baic
boot_bbic
```

Find the following AIC and BIC, with the best being the backward model selected based on AIC.

```
> boot_faic
[1] 0.02699085
> boot_fbic
[1] 0.02738823
> boot_baic
[1] 0.02579897
> boot_bbic
[1] 0.02738823
```

h)

```
# h)
library(glmnet)  # For ridge regression

x <- as.matrix(nuclear[, -1])  # All predictors, no cost
y <- log(nuclear$cost)         # Response

ridge_cv <- cv.glmnet(x, y, alpha = 0, nfolds = 10)
best_lambda <- ridge_cv$lambda.min

ridge_fit <- glmnet(x, y, alpha = 0, lambda = best_lambda)

cv <- function(vars, data, k = 10) {
  set.seed(1814)
  n <- nrow(data)
  folds <- sample(rep(1:k, length.out = n))
  errors <- numeric(k)
  for (i in 1:k) {
    train <- data[folds != i, ]
```

```r
    test <- data[folds == i, ]
    fit <- lm(log(cost) ~ ., data = train[, c("cost", vars)])
    pred <- predict(fit, newdata = test)
    errors[i] <- mean((log(test$cost) - pred)^2)
  }
  mean(errors)
}

cv_ridge <- cv(all1, nuclear)  # all1 from d) is all predictors

best_lambda
cv_ridge

cv_faic
cv_fbic
cv_baic
cv_bbic
```

We can now assess how the ridge regression compares to the cross-validated subset models. The best lambda λ we found was $\lambda \approx 0.0697$ which gave a cross-validated ridge-regression of 0.0513, this is though higher than all of the subset models shown below, having values lower than 0.05.

```
> best_lambda
[1] 0.06978489
> cv_ridge
[1] 0.05138956
```

```
> cv_faic
[1] 0.04475947
> cv_fbic
[1] 0.03703112
> cv_baic
[1] 0.04238047
> cv_bbic
[1] 0.03565251
```

i)

```r
# i)

library(glmnet)

x <- as.matrix(nuclear[, -1])
y <- log(nuclear$cost)

lasso_cv <- cv.glmnet(x, y, alpha = 1, nfolds = 10)
best_lambda_lasso <- lasso_cv$lambda.min
```

```
lasso_fit <- glmnet(x, y, alpha = 1, lambda = best_lambda_lasso)

# True Lasso CV error
cv_lasso <- lasso_cv$cvm[which(lasso_cv$lambda == best_lambda_lasso)
   ]

best_lambda_lasso
cv_lasso
cv_faic
cv_fbic
cv_baic
cv_bbic
cv_ridge
```

Here again we find a best lambda of $\lambda \approx 0.0245$, which gives a prediction error of 0.0387, which in this case is better than that of ridge regression. This is also better than the subset models selected on AIC.

```
> best_lambda_lasso
[1] 0.02450281
> cv_lasso
[1] 0.03874789


> cv_faic
[1] 0.04837609
> cv_fbic
[1] 0.03693269
> cv_baic
[1] 0.04351076
> cv_bbic
[1] 0.03808641
> cv_ridge
[1] 0.05138956
```