Problem xy (use separate files for each problem)

10111

13 januar, 2023

```
install.packages("knitr")
install.packages("MASS")
install.packages("caret")
install.packages("pls")
install.packages("glmnet")
install.packages("gam")
install.packages("gbm")
install.packages("randomForest")
install.packages("ggfortify")
install.packages("leaps")
install.packages("pROC")
install.packages("sfsmisc")
```

```
id <- "1kGOLsnKA0Uq2lWKlMjhAF8h71sc0WcLO"  # google file ID
d.bodyfat <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id))[, -c(1)]
set.seed(1234)
training_set_size <- floor(0.8 * nrow(d.bodyfat))

samples <- sample(1:nrow(d.bodyfat), training_set_size, replace = F)
d.body.train <- d.bodyfat[samples, ]
d.body.test <- d.bodyfat[-samples, ]
```

a)

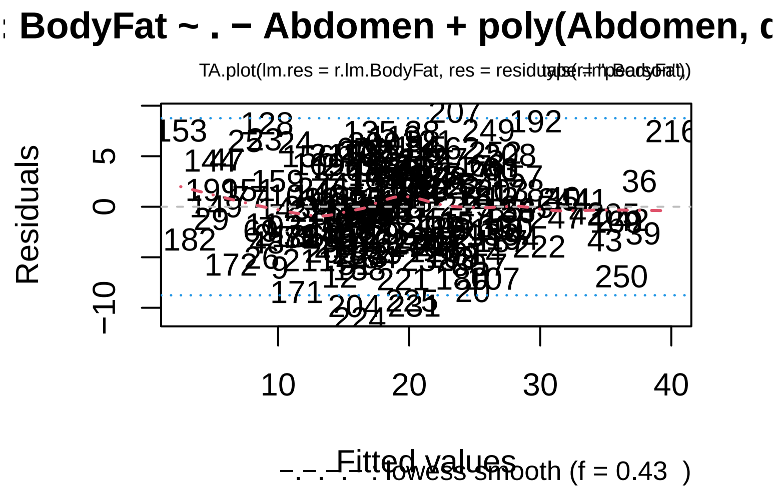
```
r.lm.BodyFat <- lm(BodyFat ~ . - Abdomen + poly(Abdomen, degree = 2), d.body.train)
summary(r.lm.BodyFat)
```

```
##
## Call:
## lm(formula = BodyFat ~ . - Abdomen + poly(Abdomen, degree = 2),
##     data = d.body.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0198  -2.9100  -0.1409   2.9595   9.3920
##
## Coefficients:
```

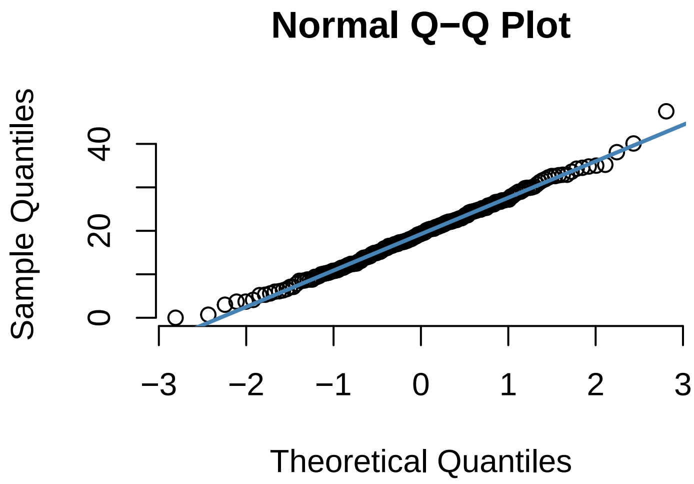
```
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  80.91992   21.31906   3.796 0.000199 ***
## Age                           0.06754    0.03566   1.894 0.059765 .
## Weight                       -0.04196    0.06164  -0.681 0.496910
## Height                       -0.06602    0.10086  -0.655 0.513560
## Neck                         -0.55179    0.25729  -2.145 0.033285 *
## Chest                        -0.08479    0.10716  -0.791 0.429830
## Hip                          -0.09304    0.16728  -0.556 0.578760
## Thigh                         0.08789    0.15638   0.562 0.574768
## Knee                         -0.10641    0.27650  -0.385 0.700799
## Ankle                         0.11223    0.23063   0.487 0.627087
## Biceps                        0.35094    0.20161   1.741 0.083391 .
## Forearm                       0.33282    0.21528   1.546 0.123807
## Wrist                        -2.08478    0.58018  -3.593 0.000418 ***
## poly(Abdomen, degree = 2)1 138.18455   15.33920   9.009 2.49e-16 ***
## poly(Abdomen, degree = 2)2 -12.58157    5.04401  -2.494 0.013490 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.234 on 186 degrees of freedom
## Multiple R-squared:  0.7559, Adjusted R-squared:  0.7375
## F-statistic: 41.15 on 14 and 186 DF,  p-value: < 2.2e-16
```

The R^2 is 0.7559. This is rather large, but we see that the Adjusted R^2 is smaller. This may indicate that the additional variables in the model is not adding value to the model.

```
TA.plot(r.lm.BodyFat, res = residuals(r.lm.BodyFat, type = "pearson"))
```



```
qqnorm(d.body.train$BodyFat, pch = 1, frame = FALSE)
qqline(d.body.train$BodyFat, col = "steelblue", lwd = 2)
```



```
# qqplot(d.body.train$BodyFat)
```

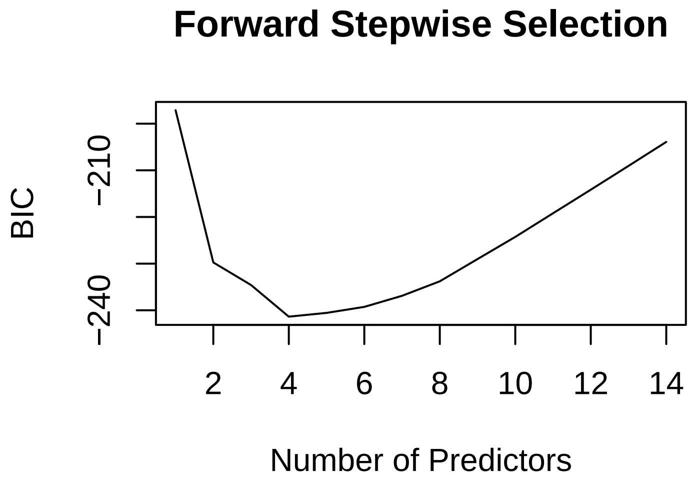
b)

Below you have to complete the code and then replace `eval=FALSE` by `eval=TRUE` in the chunk options:

```
regfit_fwd = regsubsets(BodyFat ~ . - Abdomen + poly(Abdomen, degree = 2), data = d.body.train,
    nvmax = 14, method = "forward")
regfit_fwd_summary <- summary(regfit_fwd)
regfit_fwd_summary$outmat
```

```
##           Age Weight Height Neck Chest Hip Thigh Knee Ankle Biceps Forearm
## 1  ( 1 )  " " " "    " "    " "  " "   " " " "   " "  " "   " "    " "
## 2  ( 1 )  " " "*"    " "    " "  " "   " " " "   " "  " "   " "    " "
## 3  ( 1 )  " " "*"    " "    " "  " "   " " " "   " "  " "   " "    " "
## 4  ( 1 )  " " "*"    " "    " "  " "   " " " "   " "  " "   " "    " "
## 5  ( 1 )  " " "*"    " "    " "  " "   " " " "   " "  " "   "*"    " "
## 6  ( 1 )  " " "*"    " "    "*"  " "   " " " "   " "  " "   "*"    " "
## 7  ( 1 )  "*" "*"    " "    "*"  " "   " " " "   " "  " "   "*"    " "
## 8  ( 1 )  "*" "*"    " "    "*"  " "   " " " "   " "  " "   "*"    "*"
## 9  ( 1 )  "*" "*"    " "    "*"  "*"   " " " "   " "  " "   "*"    "*"
## 10  ( 1 ) "*" "*"    "*"    "*"  "*"   " " " "   " "  " "   "*"    "*"
## 11  ( 1 ) "*" "*"    "*"    "*"  "*"   "*" " "   " "  " "   "*"    "*"
## 12  ( 1 ) "*" "*"    "*"    "*"  "*"   "*" "*"   " "  " "   "*"    "*"
## 13  ( 1 ) "*" "*"    "*"    "*"  "*"   "*" "*"   " "  "*"   "*"    "*"
## 14  ( 1 ) "*" "*"    "*"    "*"  "*"   "*" "*"   "*"  "*"   "*"    "*"
##           Wrist poly(Abdomen, degree = 2)1 poly(Abdomen, degree = 2)2
## 1  ( 1 )  " "   "*"                        " "
## 2  ( 1 )  " "   "*"                        " "
## 3  ( 1 )  "*"   "*"                        " "
## 4  ( 1 )  "*"   "*"                        "*"
## 5  ( 1 )  "*"   "*"                        "*"
## 6  ( 1 )  "*"   "*"                        "*"
## 7  ( 1 )  "*"    "*"                          "*"
## 8  ( 1 )  "*"    "*"                          "*"
## 9  ( 1 )  "*"    "*"                          "*"
## 10  ( 1 ) "*"    "*"                          "*"
## 11  ( 1 ) "*"    "*"                          "*"
## 12  ( 1 ) "*"    "*"                          "*"
## 13  ( 1 ) "*"    "*"                          "*"
## 14  ( 1 ) "*"    "*"                          "*"
```

```
plot(regfit_fwd_summary$bic, main = "Forward Stepwise Selection", xlab = "Number of Predictors",
    ylab = "BIC", type = "l")
```



Here we can see that the model with 4 predictors give the lowest BIC. That is model 6 with predictors Weight, Neck, Biceps and Wrist.

```
best_model <- lm(BodyFat ~ Weight + Neck + Biceps + Wrist, d.body.train)

mse.best_model = mean((d.body.test$BodyFat - predict(best_model, newdata = d.body.test))^2)
mse.best_model
```

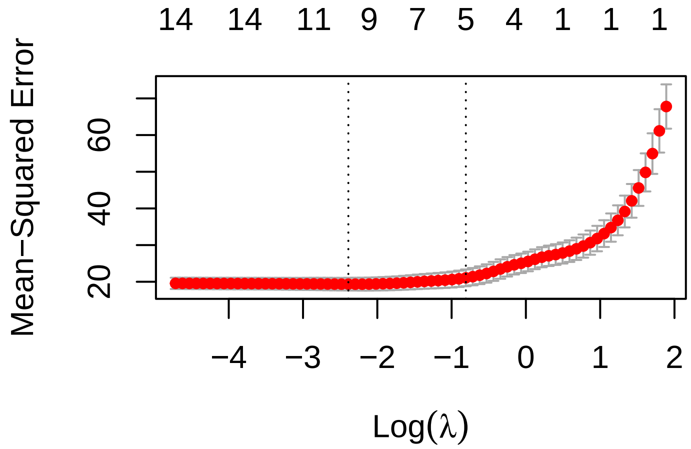
```
## [1] 40.89131
```

The MSE is 40.89131 with the reduced model

c)

```
x.train <- model.matrix(BodyFat ~ . - Abdomen + poly(Abdomen, degree = 2), data = d.body.train)[,
    -1]
y.train <- d.body.train$BodyFat
x.test = model.matrix(BodyFat ~ . - Abdomen + poly(Abdomen, degree = 2), data = d.body.test)[,
    -1]
y.test = d.body.test$BodyFat
```

```
set.seed(4268)
cv.lasso <- cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.lasso)
```



```
cv.lasso$lambda.1se
```

```
## [1] 0.4455843
```

```
bodyfat.lasso <- glmnet(x.train, y.train, alpha = 1, lambda = cv.lasso$lambda.1se)
coef(bodyfat.lasso)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                                   s0
## (Intercept)               47.09872944
## Age                        0.04167346
## Weight                     .
## Height                    -0.12282724
## Neck                       .
## Chest                      .
## Hip                        .
## Thigh                      .
## Knee                       .
## Ankle                      .
## Biceps                     .
## Forearm                    .
## Wrist                     -1.14587023
## poly(Abdomen, degree = 2)1  95.63871126
## poly(Abdomen, degree = 2)2 -11.14529896
```

```
mse.lasso = mean((y.test - predict(bodyfat.lasso, newx = x.test))^2)
mse.lasso
```

```
## [1] 50.31623
```

The MSE for Lasso is 50.31623.

```r
cv.lasso$lambda.min
```

```
## [1] 0.09163496
```

```r
bodyfat.lasso.min <- glmnet(x.train, y.train, alpha = 1, lambda = cv.lasso$lambda.min)
coef(bodyfat.lasso.min)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                                 s0
## (Intercept)            69.29797647
## Age                     0.07085015
## Weight                  .
## Height                 -0.11547703
## Neck                   -0.42359496
## Chest                   .
## Hip                     .
## Thigh                   .
## Knee                   -0.08296349
## Ankle                   .
## Biceps                  0.17064055
## Forearm                 0.20158403
## Wrist                  -2.02171921
## poly(Abdomen, degree = 2)1 112.32782320
## poly(Abdomen, degree = 2)2 -14.66393600
```

```r
mse.lasso.min = mean((y.test - predict(bodyfat.lasso.min, newx = x.test))^2)
mse.lasso.min
```

```
## [1] 74.07258
```

The MSE when λ_{min} is used is much larger than when λ_{1se} is used, thus the model with λ_{min} is not useful.

d)

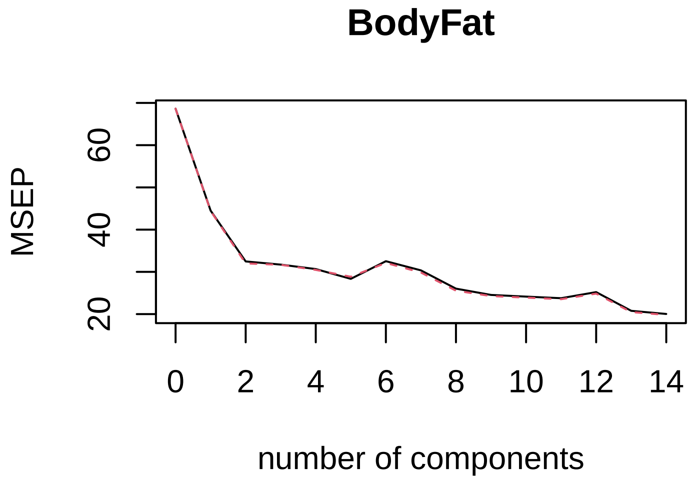
```r
# Had problems running this in markdown but it worked in rscript pca.train <-
# prcomp(d.body.train[, c(2:6,8:-1)] + poly(d.body.train[, c(7)], degree = 2),
# scale = TRUE) var_explained = pca.train$sdev^2 / sum(pca.train$sdev^2)

# screeplot(pca.train, npcs = min(13, length(pca.train$sdev)), type =
# c('lines'))
```

See that the first PC explain over 60% of variability, and PC 2-4 explain approximately 10%, and the rest go near 0. So a useful number would be 5 PCs.

```r
pcr_fit = pcr(BodyFat ~ . - Abdomen + poly(Abdomen, degree = 2), data = d.body.train,
    scale = TRUE, validation = "CV")
summary(pcr_fit)
```

```
## Data:    X dimension: 201 14
##  Y dimension: 201 1
## Fit method: svdpc
## Number of components considered: 14
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           8.285    6.671    5.698    5.632    5.538    5.324    5.704
## adjCV        8.285    6.665    5.656    5.628    5.519    5.371    5.667
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       5.509    5.102    4.954     4.914     4.875     5.021     4.558
## adjCV    5.463    5.054    4.926     4.889     4.851     4.985     4.525
##        14 comps
## CV        4.476
## adjCV     4.451
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X          56.69    66.41    75.17    81.79    86.37    90.61    92.83    94.91
## BodyFat    36.12    54.98    55.53    58.39    58.40    62.92    66.50    69.40
##          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X          96.67     98.00     98.96     99.50     99.84    100.00
## BodyFat    69.87     70.18     70.70     71.38     75.53     75.59
```

```
validationplot(pcr_fit, val.type = "MSEP")
```


BodyFat
MSEP
number of components