Assignment 2: Group 45

Daniel Engbert, Rik Timmer, Koen van der Pool

03 March 2023

Note: we made a function `checkNorm()` which prints a histogram, qqplot, and p-value from the shapiro-wilk normality test. And we made a function `printPval()` which simply prints a given p-value to 3 significant figures. We utilize both functions throughout this assignment.

Exercise 1: Trees

1 a)

```
trees = read.table("treeVolume.txt", header=T)
model = lm(volume~type, data=trees)
print("model coefficients:"); summary(model)$coefficients
```

```
## [1] "model coefficients:"
```

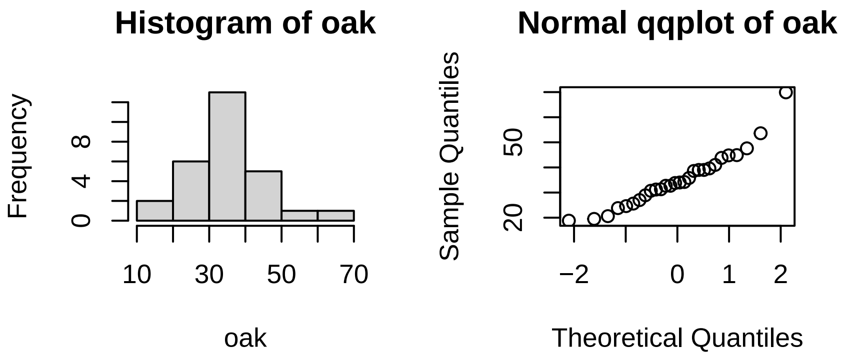
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    30.17       2.54   11.88 4.68e-17
## typeoak         5.08       3.69    1.38 1.74e-01
```

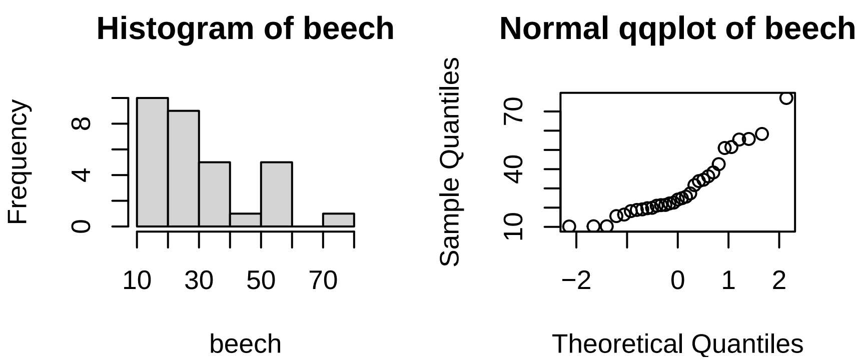
```
res = anova(model)
sprintf("ANOVA p-value for type = %.3f", res["type", "Pr(>F)"])
```

```
## [1] "ANOVA p-value for type = 0.174"
```

The p-value $0.174 > 0.05$ for the type in the ANOVA analysis of the linear model, suggests there's insufficient evidence to reject the H_0 (that tree type influences volume).

```
## [1] "Shapiro-Wilk normality p-value for oak: 0.082"
```






```
## [1] "Shapiro-Wilk normality p-value for beech: 0.004"
```

```
## [1] "oak mean volume = 35.250, beech mean volume = 30.171"
```

We can split the data into two samples of tree volume based on the tree types, and compare the means of the samples using a t-test to determine whether, based on this data, there is a significant difference in mean volume between the two tree types. As can be seen in the output of the t-test $0.166 > 0.05$, signifying once again that there is not enough evidence to reject the null hypothesis that the means of the samples are the same. This concurs with the results of the ANOVA.

```
new_oak = data.frame(type="oak"); new_beech = data.frame(type = "beech")
pred1 = predict(model, new_oak); pred2 = predict(model, new_beech)
sprintf("predicted volumes: oak = %.3f, beech = %.3f", pred1, pred2)
```

```
## [1] "predicted volumes: oak = 35.250, beech = 30.171"
```

1 b)

```
model = lm(volume~type*diameter + height, data=trees)
res = anova(model)
sprintf("ANOVA p-value for type:diameter = %.3f", res["type:diameter", "Pr(>F)"])
```

```
## [1] "ANOVA p-value for type:diameter = 0.474"
```

We built a linear model that added an interaction term between diameter and type, the p-value $0.474 > 0.05$ for this term suggests there's insufficient evidence to reject the H_0 (that the influence of diameter on volume is the same for both tree types).

```r
model = lm(volume~type*height + diameter, data=trees)
res = anova(model)
sprintf("ANOVA p-value for type:diameter = %.3f", res["type:height", "Pr(>F)"])
```

```
## [1] "ANOVA p-value for type:diameter = 0.176"
```

Now running another linear model that includes an interaction term between height and type instead, the p-value $0.176 > 0.05$ for this term suggests there's insufficient evidence to reject the H_0 (that the influence of height on volume is the same for both tree types).

So based on the results from our two models above, there's insufficient evidence to suggest that the influences of diameter and height aren't similar for both tree types.

1 c)

We construct a linear model to investigate how diameter, height and type influence volume.

```r
model = lm(volume~type+height+diameter, data=trees)
print("model coefficients:"); summary(model)$coefficients
```

```
## [1] "model coefficients:"

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -63.781     5.5129  -11.57 2.33e-16
## typeoak       -1.305     0.8779   -1.49 1.43e-01
## height         0.417     0.0752    5.55 8.42e-07
## diameter       4.698     0.1645   28.56 1.14e-34
```

```r
print("anova:"); res = anova(model); res
```

```
## [1] "anova:"

## Analysis of Variance Table
##
## Response: volume
##           Df Sum Sq Mean Sq F value  Pr(>F)
## type       1    380     380    36.1 1.6e-07 ***
## height     1   2239    2239   212.9 < 2e-16 ***
## diameter   1   8577    8577   815.6 < 2e-16 ***
## Residuals 55    578      11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the ANOVA p-values, type is not a significant predictor for volume (p-value $0.143 > 0.05$), while height and diameter are significant (p-values less than 0.05). Diameter and height are both positively correlated with the volume, with diameter having the largest contribution (coefficient) of the two.

```r
# build better model where type isn't considered
modelC = lm(volume~height+diameter, data=trees)

avgTree = data.frame(height=mean(trees$height), diameter=mean(trees$diameter))
```

```r
pred = predict(modelC, avgTree)
sprintf("predicted volume of average tree = %.3f", pred)
```

```
## [1] "predicted volume of average tree = 32.581"
```

```r
# mean(trees$volume) # this also gives the same result as expected

r2 = summary(modelC)$r.squared; ar2 = summary(modelC)$adj.r.squared
sprintf("modelC: R^2 = %.3f, Adj. R^2 = %.3f", r2, ar2)
```

```
## [1] "modelC: R^2 = 0.949, Adj. R^2 = 0.947"
```

Using the resulting model, the volume of a tree with the average height and diameter is predicted to be 32.581 .

1 d)

We propose to transform the data to create a new column that contains the volume of a (theoretical) cylinder based on the tree's diameter and height. (Note we omit tree type from the model as we found it to not be a significant predictor above).

```r
# create predictor as cylinderical volume
trees$cylinder = trees$diameter * pi * trees$height

modelD = lm(volume~cylinder, data=trees)
print("model coefficients:"); summary(modelD)$coefficients
```

```
## [1] "model coefficients:"

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -26.8923   2.058137   -13.1 8.67e-19
## cylinder      0.0179   0.000603    29.7 2.47e-36
```

```r
r2 = summary(modelD)$r.squared; ar2 = summary(modelD)$adj.r.squared
sprintf("model: R^2 = %.3f, Adj. R^2 = %.3f", r2, ar2)
```

```
## [1] "model: R^2 = 0.939, Adj. R^2 = 0.938"
```

```r
print("ANOVA:"); anova(model)
```

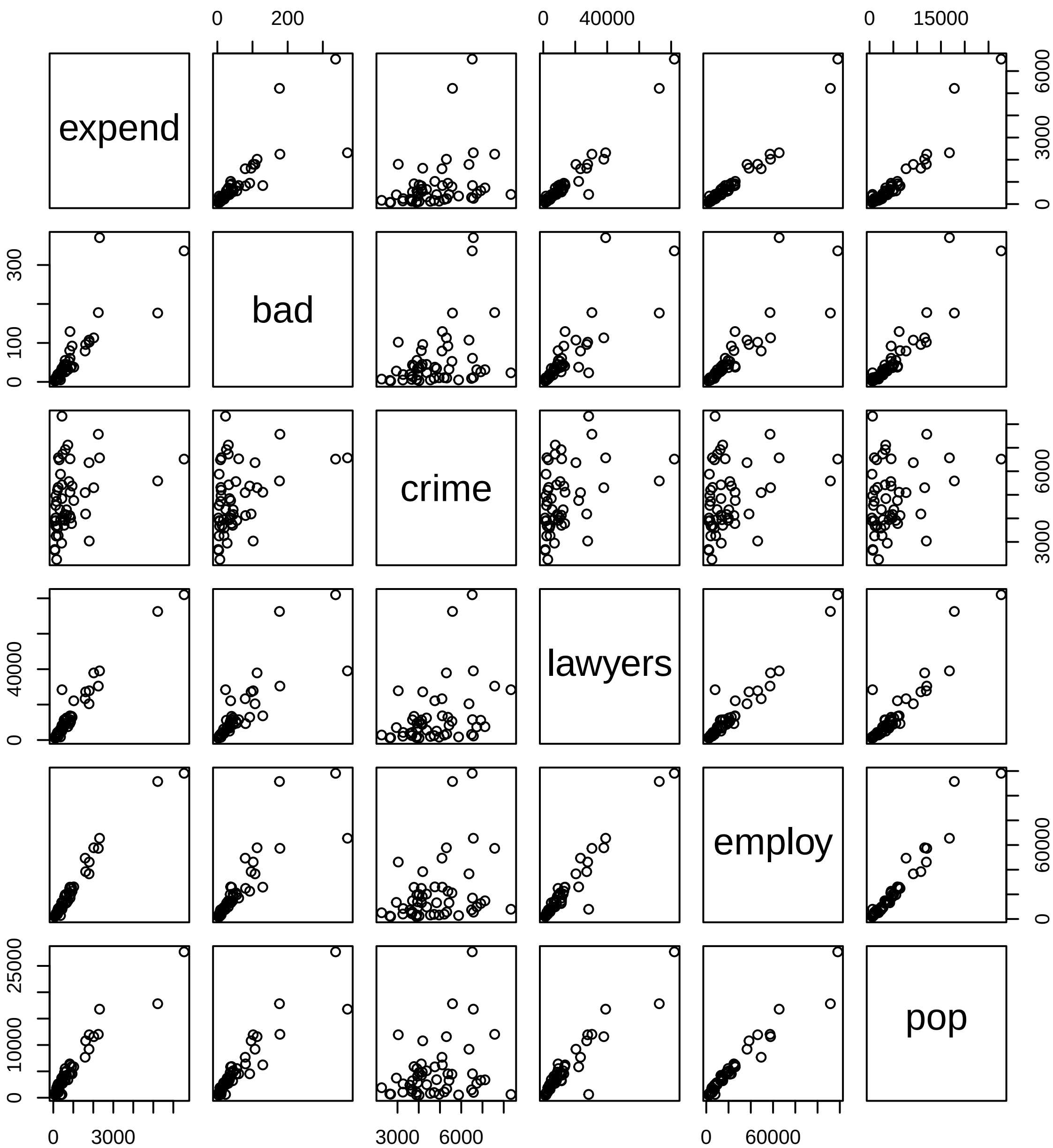
```
## [1] "ANOVA:"

## Analysis of Variance Table
##
## Response: volume
##           Df Sum Sq Mean Sq F value  Pr(>F)
## type       1    380     380    36.1 1.6e-07 ***
## height     1   2239    2239   212.9 < 2e-16 ***
## diameter   1   8577    8577   815.6 < 2e-16 ***
## Residuals 55    578      11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After constructing a linear model for predicting the actual tree volume from our proposed cylindrical estimator, we see that the cylinder variable is a significant predictor of volume ($p < 0.05$). However the adjusted R^2 values (and the regular R^2 values) for this model are less than that of the model in part c), so while cylinder is a useful predictor, it's still inferior to using just the provided height and diameter variables in the model.

Exercise 2: Expenditure on criminal activities

2 a)

```
crimes = read.table("expensescrime.txt", header=T)
pairs(crimes[,-1])
```



```r
crimes$state = factor(crimes$state)


model = lm(expend~bad+crime+lawyers+employ+pop, data=crimes)
summary(model)$coefficients
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -299.1341   1.40e+02   -2.14  0.03817
## bad           -2.8319   1.24e+00   -2.28  0.02719
## crime          0.0324   2.81e-02    1.15  0.25534
## lawyers        0.0232   8.04e-03    2.89  0.00592
## employ         0.0230   7.46e-03    3.08  0.00354
## pop            0.0779   3.51e-02    2.22  0.03184
```

```r
anova(model)
```

```
## Analysis of Variance Table
##
## Response: expend
##           Df   Sum Sq  Mean Sq F value  Pr(>F)
## bad        1 49109638 49109638  965.16 < 2e-16 ***
## crime      1    44115    44115    0.87   0.357
## lawyers    1 17237521 17237521  338.77 < 2e-16 ***
## employ     1  1590235  1590235   31.25 1.3e-06 ***
## pop        1   249704   249704    4.91   0.032 *
## Residuals 45  2289716    50883
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
print('model 2:')
```

```
## [1] "model 2:"
```

```r
model = lm(expend~crime+bad+lawyers+employ+pop, data=crimes)
summary(model)$coefficients
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -299.1341   1.40e+02   -2.14  0.03817
## crime          0.0324   2.81e-02    1.15  0.25534
## bad           -2.8319   1.24e+00   -2.28  0.02719
## lawyers        0.0232   8.04e-03    2.89  0.00592
## employ         0.0230   7.46e-03    3.08  0.00354
## pop            0.0779   3.51e-02    2.22  0.03184
```

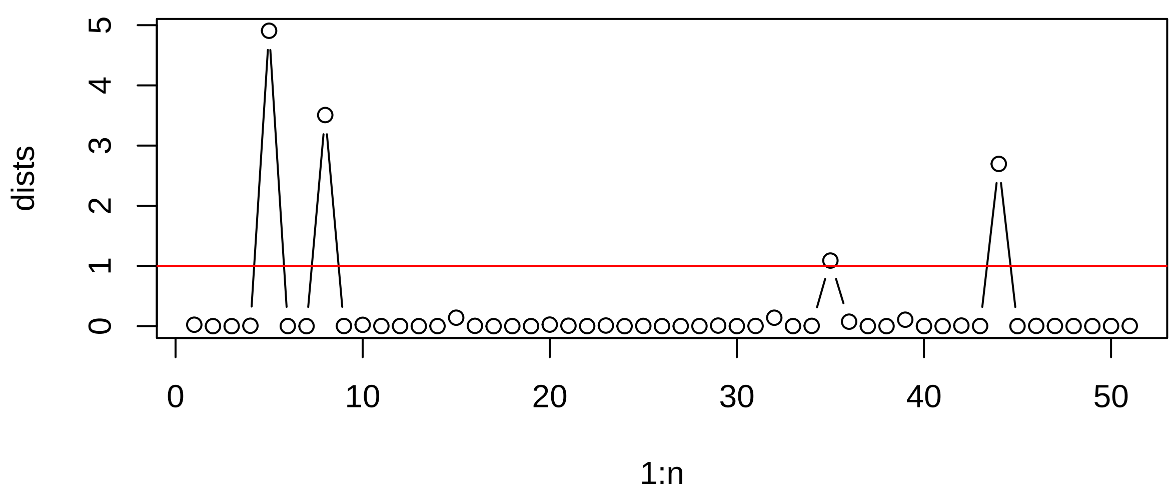
```r
anova(model)
```

```
## Analysis of Variance Table
##
## Response: expend
##           Df   Sum Sq  Mean Sq F value  Pr(>F)
## crime      1  7888219  7888219  155.03 3.5e-16 ***
```

```
## bad         1 41265535 41265535  811.00 < 2e-16 ***
## lawyers     1 17237521 17237521  338.77 < 2e-16 ***
## employ      1  1590235  1590235   31.25 1.3e-06 ***
## pop         1   249704   249704    4.91   0.032 *
## Residuals 45  2289716    50883
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# sorting by population
#crimes[order(crimes$pop, decreasing=TRUE),]




#crimes
```

```
n = length(crimes[,1])
dists = cooks.distance(model)
plot(1:n, dists, type="b")
abline(1, 0, col = 'red') # plot y=1 for reference
```




```
# these are the indices into crimes that are cook's points
dists[dists > 1]
```

```
##    5    8   35   44
## 4.91 3.51 1.09 2.70
```

```
# TODO: print state names
#cooked = crimes[dists[dists > 1],]
#cooked

# investigating collinearity
cor(crimes[,-1])
```

```
##           expend   bad crime lawyers employ   pop
## expend     1.000 0.834 0.334   0.968  0.977 0.953
## bad        0.834 1.000 0.373   0.832  0.871 0.920
## crime      0.334 0.373 1.000   0.375  0.311 0.275
```

```
## lawyers  0.968 0.832 0.375   1.000  0.966 0.934
## employ   0.977 0.871 0.311   0.966  1.000 0.971
## pop      0.953 0.920 0.275   0.934  0.971 1.000
```

```r
res = cor(crimes[,-1])
# using 0.8 as a threshold to help with visiblility
res[res >= 0.8] = T; res[res <= 0.8] = F;
res
```

```
##         expend bad crime lawyers employ pop
## expend       1   1     0       1      1   1
## bad          1   1     0       1      1   1
## crime        0   0     1       0      0   0
## lawyers      1   1     0       1      1   1
## employ       1   1     0       1      1   1
## pop          1   1     0       1      1   1
```

Based on the correlation coefficients, it appears that all the explanatory variables are correlated, except for crime which has no correlation with any of the other variables (its highest correlation coefficient is 0.375). The other variables all have a correlation coefficient of at least 0.832 between each other.

2 b)

```r
evalModel = function(model, name) {
  print(sprintf("adding var '%s':", name))
  print(summary(model)$coefficients)
  r2 = summary(model)$r.squared; ar2 = summary(model)$adj.r.squared
  sprintf("model: R^2 = %.3f", r2)
}

evalModel(lm(expend~bad, data=crimes), name="bad")
```

```
## [1] "adding var 'bad':"
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    126.7     114.86     1.1 2.75e-01
## bad             13.3       1.26    10.6 2.80e-14
```

```
## [1] "model: R^2 = 0.696"
```

```r
evalModel(lm(expend~crime, data=crimes), name="crime")
```

```
## [1] "adding var 'crime':"
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -531.039    577.166   -0.92   0.3620
## crime          0.287      0.116    2.48   0.0165
```

```
## [1] "model: R^2 = 0.112"
```

```r
evalModel(lm(expend~lawyers, data=crimes), name="lawyers")
```

```
## [1] "adding var 'lawyers':"
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -59.6120    53.7994   -1.11 2.73e-01
## lawyers       0.0704     0.0026   27.06 4.02e-31
```

```
## [1] "model: R^2 = 0.937"
```

```r
evalModel(lm(expend~employ, data=crimes), name="employ")
```

```
## [1] "adding var 'employ':"
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -116.7052   47.06076   -2.48 1.66e-02
## employ         0.0468    0.00147   31.87 2.03e-34
```

```
## [1] "model: R^2 = 0.954"
```

```r
evalModel(lm(expend~pop, data=crimes), name="pop")
```

```
## [1] "adding var 'pop':"
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -188.767   69.67628   -2.71 9.27e-03
## pop            0.217    0.00992   21.90 5.83e-27
```

```
## [1] "model: R^2 = 0.907"
```

```r
# employ has highest adj. R^2 (0.955) and is significant
print("****round2****")
```

```
## [1] "****round2****"
```

```r
evalModel(lm(expend~employ+bad, data=crimes), name="bad")
```

```
## [1] "adding var 'bad':"
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -116.4498   46.96559   -2.48 1.67e-02
## employ         0.0497    0.00299   16.63 1.48e-21
## bad           -1.0898    0.99481   -1.10 2.79e-01
```

```
## [1] "model: R^2 = 0.955"
```

```r
evalModel(lm(expend~employ+crime, data=crimes), name="crime")
```

```
## [1] "adding var 'crime':"
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -248.3631   1.32e+02   -1.89 6.50e-02
## employ         0.0463   1.54e-03   30.01 9.37e-33
## crime          0.0296   2.76e-02    1.07 2.89e-01
```

```
## [1] "model: R^2 = 0.955"
```

```r
evalModel(lm(expend~employ+lawyers, data=crimes), name="lawyers")
```

```
## [1] "adding var 'lawyers':"
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -110.6588   42.56735   -2.60 1.24e-02
## employ         0.0297    0.00511    5.81 4.89e-07
```

```
## lawyers        0.0269    0.00776    3.46 1.13e-03
```

```
## [1] "model: R^2 = 0.963"
```

```
evalModel(lm(expend~employ+pop, data=crimes), name="pop")
```

```
## [1] "adding var 'pop':"
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -126.5921   50.21637  -2.521 1.51e-02
## employ         0.0433    0.00616   7.026 6.72e-09
## pop            0.0174    0.02930   0.594 5.55e-01
```

```
## [1] "model: R^2 = 0.954"
```

In the 1st round of the "step up" method we found "employ" to lead to the largest R^2 model, while still being statistically significant.

In the 2nd round of the "step up" method, "lawyers" was found to lead to the largest increase in R^2 while still being statistically significant, however the increase in R^2 was only $0.963 - 0.954 = 0.009$, which is quite low, so we don't deem it worth adding to the model.

The result of the "step up" method suggesting the model should only have one explanatory variable ("employ") is not surprising as we showed further above that all the variables (except for "crime") are collinear.

2 c)

```
model = lm(expend~employ, data=crimes) # result of part 2b
state = data.frame(bad=50, crime=5000, lawyers=5000, employ=5000, pop=5000)
predict(model, state, interval="prediction")
```

```
##   fit  lwr upr
## 1 117 -407 642
```

The predicted interval $[-407, 642]$ can be improved by adjusting it to $[0, 642]$ as we know the expenditure must be a positive number. So we're 95% confident that the expenditure by this hypothetical state would be between \$0 and \$642,000.

2 d)

```
mtcars # dataset mtcars: mpg is the response
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.62 16.5  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.88 17.0  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.32 18.6  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.21 19.4  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.44 17.0  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.46 20.2  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.57 15.8  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.19 20.0  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.15 22.9  1  0    4    2
```

```
## Merc 280            19.2   6 167.6 123 3.92 3.44 18.3  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.44 18.9  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.07 17.4  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.73 17.6  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.78 18.0  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.25 18.0  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.42 17.8  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.34 17.4  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.20 19.5  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.61 18.5  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.83 19.9  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.46 20.0  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.52 16.9  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.44 17.3  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.84 15.4  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.85 17.1  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.94 18.9  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.14 16.7  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.51 16.9  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.17 14.5  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.77 15.5  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.57 14.6  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.78 18.6  1  1    4    2
```

```r
x=as.matrix(mtcars[,-1])
y=mtcars[,1]

train=sample(1:nrow(x),0.67*nrow(x)) # train by using 2/3 of the x rows
x.train=x[train,]; y.train=y[train]  # data to train
x.test=x[-train,]; y.test = y[-train] # data to test the prediction quality

# Prediction by using the linear model
# first fit linear model on the train data
lm.model=lm(mpg~cyl+disp+hp+drat+wt+qsec+vs+am+gear+carb,data=mtcars,subset=train)
y.predict.lm=predict(lm.model,newdata=mtcars[-train,]) # predict for the test rows
mse.lm=mean((y.test-y.predict.lm)^2); mse.lm # prediction quality by the linear model
```

```
## [1] 9.03
```

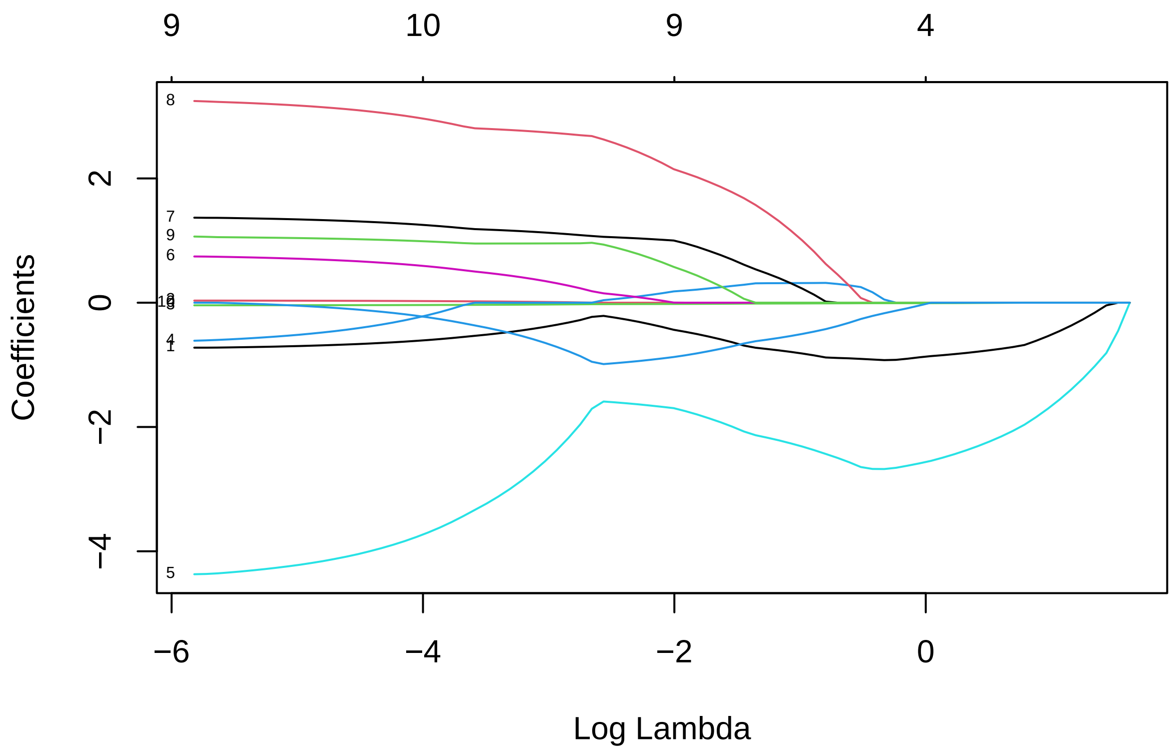
```r
# Now apply lasso for selecting the variables and prediction
library(glmnet)
```

```
## Loading required package: Matrix
```

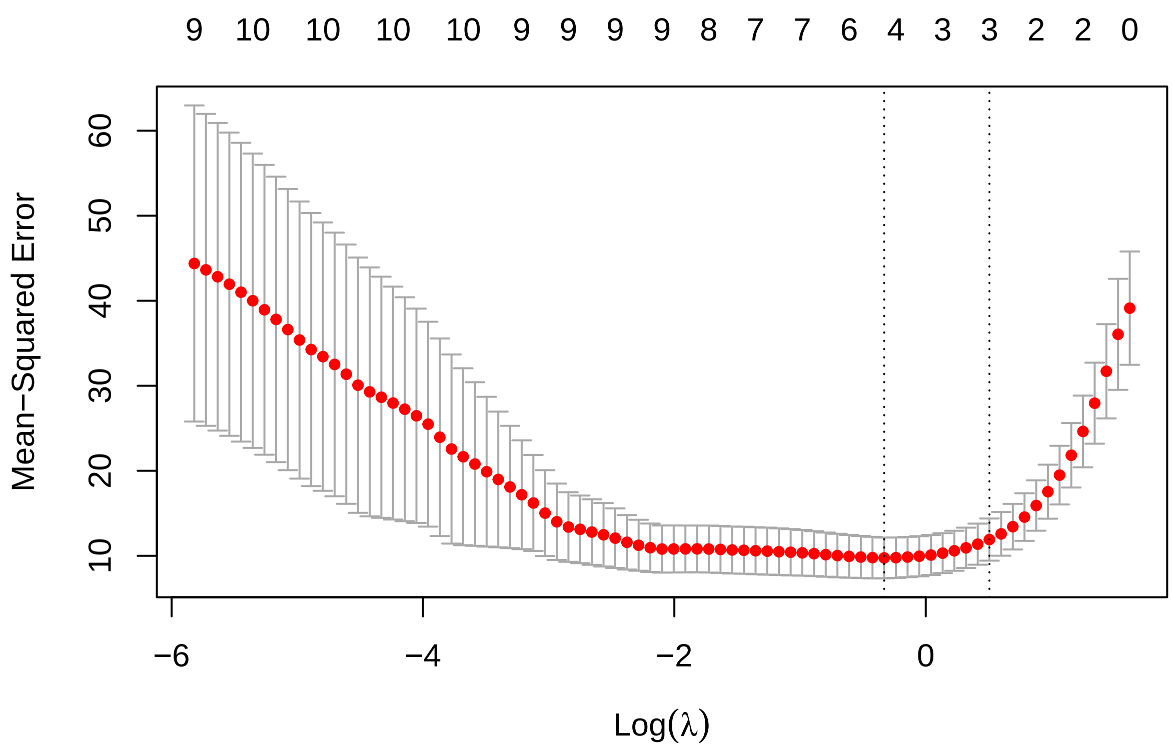
```
## Loaded glmnet 4.1-6
```

```r
lasso.model=glmnet(x.train,y.train,alpha=1) # alpha=1 for lasso
#more options: standardize=TRUE, intercept=FALSE,nlambda=1000
lasso.cv=cv.glmnet(x.train,y.train,alpha=1,type.measure="mse",nfolds=5)
# option nfolds=5 means 5-fold cross validation. By default, the method
```

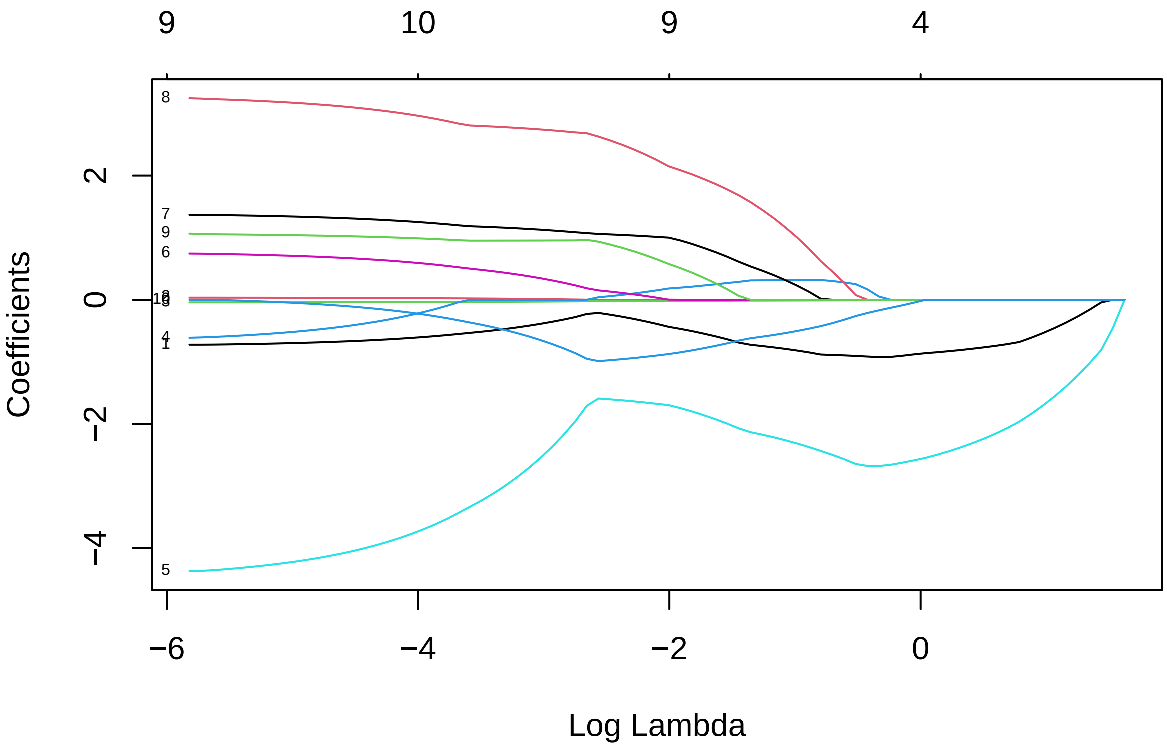
```
# performs 10-fold cross validation to choose the best lambda.
# plots
plot(lasso.model,label=T,xvar="lambda") #standardize=T,type.coef="2norm",xvar="norm") "coef"
```




```
#plot(lasso.cv$glmnet.fit,xvar="lambda",label=T) # the same plot
plot(lasso.cv)
```



```r
plot(lasso.cv$glmnet.fit,xvar="lambda",label=T)
```




```r
# With label="T" in plot commando you see which curve corresponds
# to which coefficients. The glmnet plot above shows the shrinkage of
# the lasso coefficients as you move from the right to the left,
# but unfortunately, it is not clearly labelled.
# Lasso contrasts with ridge regression, which flattens out
# everything, but does not zero out any of the regression coefficients.

lambda.min=lasso.cv$lambda.min; lambda.1se=lasso.cv$lambda.1se;
lambda.min; lambda.1se # best lambda by cross validation
```

```
## [1] 0.718
```

```
## [1] 1.66
```

```r
coef(lasso.model,s=lasso.cv$lambda.min) # cyl,hp,wt,am and carb are relevant
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 36.0042
## cyl         -0.9239
## disp          .
## hp          -0.0079
## drat         0.0513
## wt          -2.6769
## qsec          .
## vs            .
```

```
## am            .
## gear          .
## carb          -0.1695
```

```
coef(lasso.model,s=lasso.cv$lambda.1se) # only cyl,hp and wt are releveant
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                   s1
## (Intercept) 32.88598
## cyl         -0.76546
## disp         .
## hp          -0.00362
## drat         .
## wt          -2.23447
## qsec         .
## vs           .
## am           .
## gear         .
## carb         .
```

```
# lambda.min is the value of lambda that gives minimum mean cross-validated
# error. The other lambda saved is lambda.1se, which gives the most regularized
# (reduced) model such that error is within one standard error of the minimum.

lasso.pred1=predict(lasso.model,s=lambda.min,newx=x.test)
lasso.pred2=predict(lasso.model,s=lambda.1se,newx=as.matrix(x.test))
mse1.lasso=mean((y.test-lasso.pred1)^2); mse1.lasso
```

```
## [1] 6.26
```

```
mse2.lasso=mean((y.test-lasso.pred2)^2); mse2.lasso
```

```
## [1] 10.5
```

```
library(glmnet)
```

Exercise 3: Titanic

3 a)

```
titanic = read.table("titanic.txt", header=T)
#titanic
```

Exercise 4: Military Coups

4 a)

```
coups = read.table("coups.txt", header=T)
#coups
```