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)

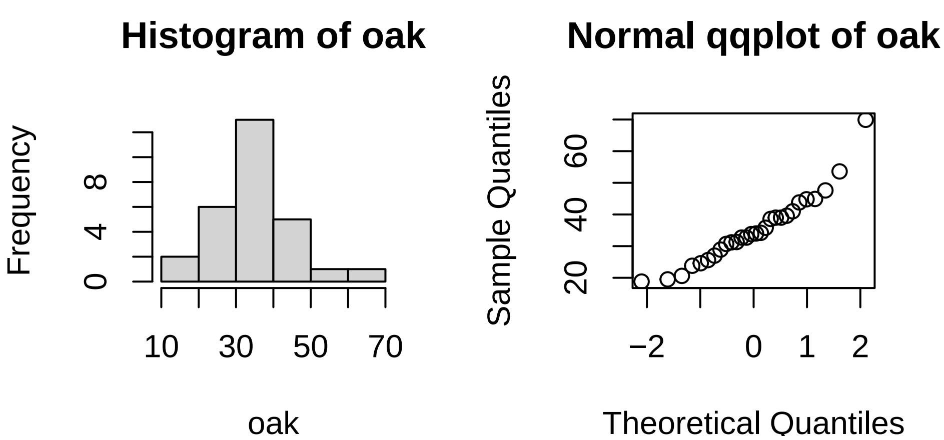
```
trees = read.table("treeVolume.txt", header=T)
model = lm(volume~type, data=trees)
print("model coefficients:"); summary(model)$coefficients
```

```
## [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
```

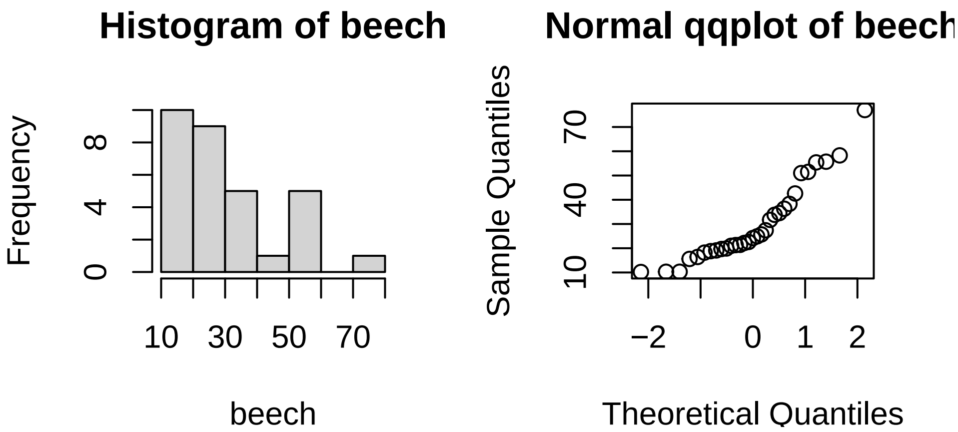
```
res = anova(model)
sprintf("ANOVA p-value for type = %.3f", res["type", "Pr(>F)"])
```

```
## [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).

```
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)
model = lm(volume~diameter+height+type, 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
## diameter       4.698     0.1645   28.56 1.14e-34
## height         0.417     0.0752    5.55 8.42e-07
## typeoak       -1.305     0.8779   -1.49 1.43e-01
```

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

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

## Analysis of Variance Table
##
## Response: volume
##           Df Sum Sq Mean Sq F value  Pr(>F)
## diameter   1  10827   10827 1029.51 < 2e-16 ***
## height     1    346     346   32.92 4.3e-07 ***
## type       1     23      23    2.21    0.14
## 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))
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).

```
# create predictor as cylinderical volume
#trees$cylinder = trees$diameter * pi * trees$height
trees$cylinder = pi * (trees$diameter / 2)^2 * trees$height

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

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

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.37942   7.63e-01  -0.497 6.21e-01
## cylinder     0.00273   5.82e-05  46.913 3.09e-47
```

```
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 = 0.975, Adj. R^2 = 0.974"
```

```
print("ANOVA:"); anova(modelD)
```

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

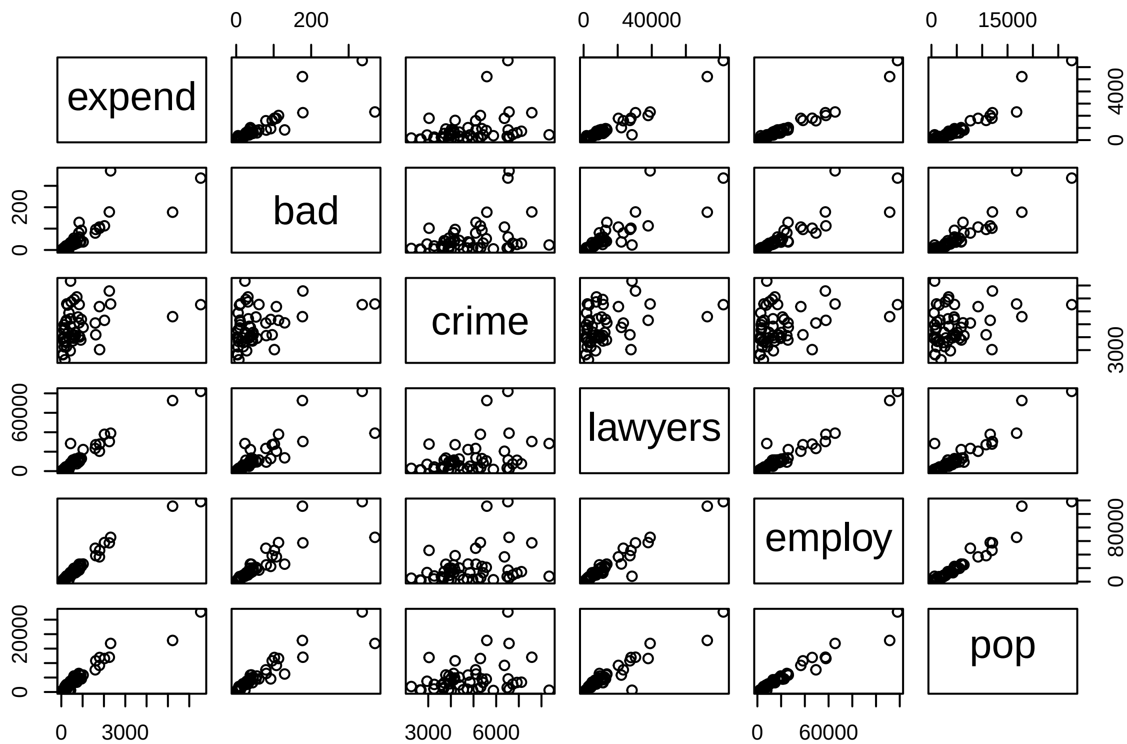
```
## Analysis of Variance Table
##
## Response: volume
##           Df Sum Sq Mean Sq F value Pr(>F)
## cylinder   1  11477   11477    2201 <2e-16 ***
## Residuals 57    297       5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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 ($p < 0.05$). The adjusted R^2 values (and the regular R^2 values) for this model are both greater than that of the model in part c), so this model appears to be superior 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
```

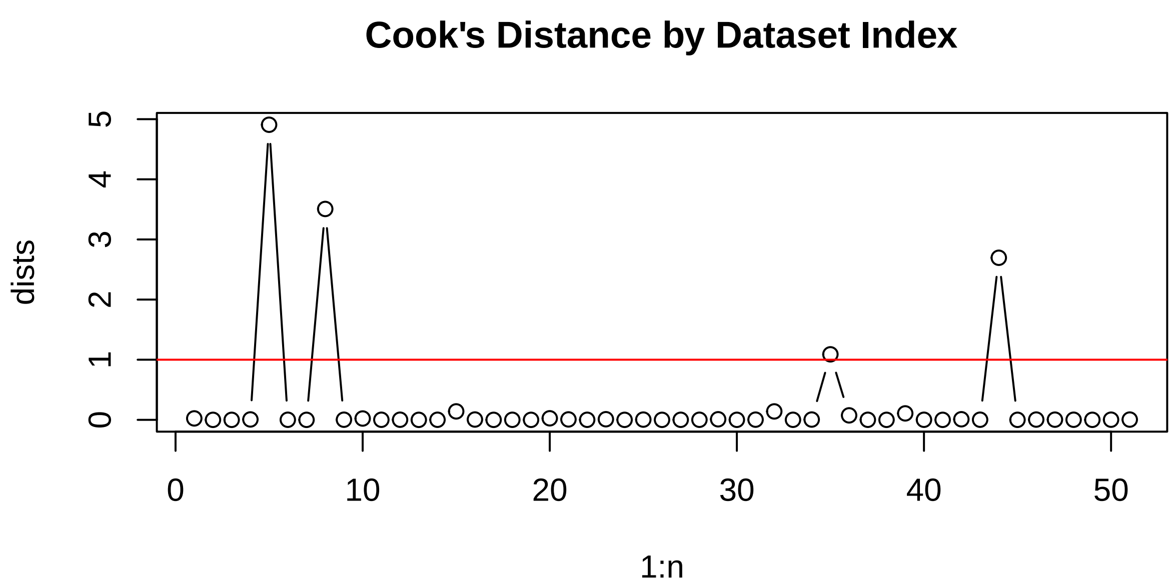
```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -299.1341   1.40e+02   -2.14  0.03817
## crime          0.0324   2.81e-02    1.15  0.25534
## bad           -2.8319   1.24e+00   -2.28  0.02719
## lawyers        0.0232   8.04e-03    2.89  0.00592
## employ         0.0230   7.46e-03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

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

```r
n = length(crimes[,1])
dists = cooks.distance(model)
plot(1:n, dists, type="b", main="Cook's Distance by Dataset Index")
abline(1, 0, col = 'red') # plot y=1 for reference
```




```r
# these are the indices into crimes that are cook's points
print("Influence points:"); crimes[dists > 1,]
```

```
## [1] "Influence points:"
```

```
##    state expend   bad crime lawyers employ   pop
## 5     CA   6539 336.2  6518   82001 118149 27663
## 8     DC    435  23.3  8339   28399   7925   622
## 35    NY   5220 176.7  5589   72575 111518 17825
## 44    TX   2313 370.1  6569   39028  65488 16789
```

```r
# 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
# using 0.8 as a threshold to help with visiblility:
res = cor(crimes[,-1])
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 with each other, 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)

```
evalModel = function(model, name) {
  r2 = summary(model)$r.squared; ar2 = summary(model)$adj.r.squared
  pVal = summary(model)$coefficients[name, "Pr(>|t|)"]
  cat(sprintf("adding var '%s'\t\tPr(>|t|) =  %.3f, model R^2 = %.3f\n", name, pVal, r2))
}

doStepUp = function() {
  cat("\n****round1****\n")
  evalModel(lm(expend~bad, data=crimes), name="bad")
  evalModel(lm(expend~crime, data=crimes), name="crime")
  evalModel(lm(expend~lawyers, data=crimes), name="lawyers")
  evalModel(lm(expend~employ, data=crimes), name="employ")
  evalModel(lm(expend~pop, data=crimes), name="pop")

  # employ has highest adj. R^2 (0.954) and is significant
  cat("\n****round2****\n")
  evalModel(lm(expend~employ+bad, data=crimes), name="bad")
  evalModel(lm(expend~employ+crime, data=crimes), name="crime")
  evalModel(lm(expend~employ+lawyers, data=crimes), name="lawyers")
  evalModel(lm(expend~employ+pop, data=crimes), name="pop")
}
doStepUp()
```

```
##
## ****round1****
## adding var 'bad'     Pr(>|t|) =  0.000, model R^2 = 0.696
## adding var 'crime'       Pr(>|t|) =  0.016, model R^2 = 0.112
## adding var 'lawyers'     Pr(>|t|) =  0.000, model R^2 = 0.937
## adding var 'employ'      Pr(>|t|) =  0.000, model R^2 = 0.954
## adding var 'pop'     Pr(>|t|) =  0.000, model R^2 = 0.907
##
## ****round2****
## adding var 'bad'     Pr(>|t|) =  0.279, model R^2 = 0.955
## adding var 'crime'       Pr(>|t|) =  0.289, model R^2 = 0.955
## adding var 'lawyers'     Pr(>|t|) =  0.001, model R^2 = 0.963
## adding var 'pop'     Pr(>|t|) =  0.555, model R^2 = 0.954
```

In the 1st round of the "step up" method we found "employ" to lead to the model with the largest R^2, while still being statistically significant. In the 2nd round, "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. (And so our final model is `lm(expend~employ, data=crimes)`).

The result of the "step up" method suggesting that 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.

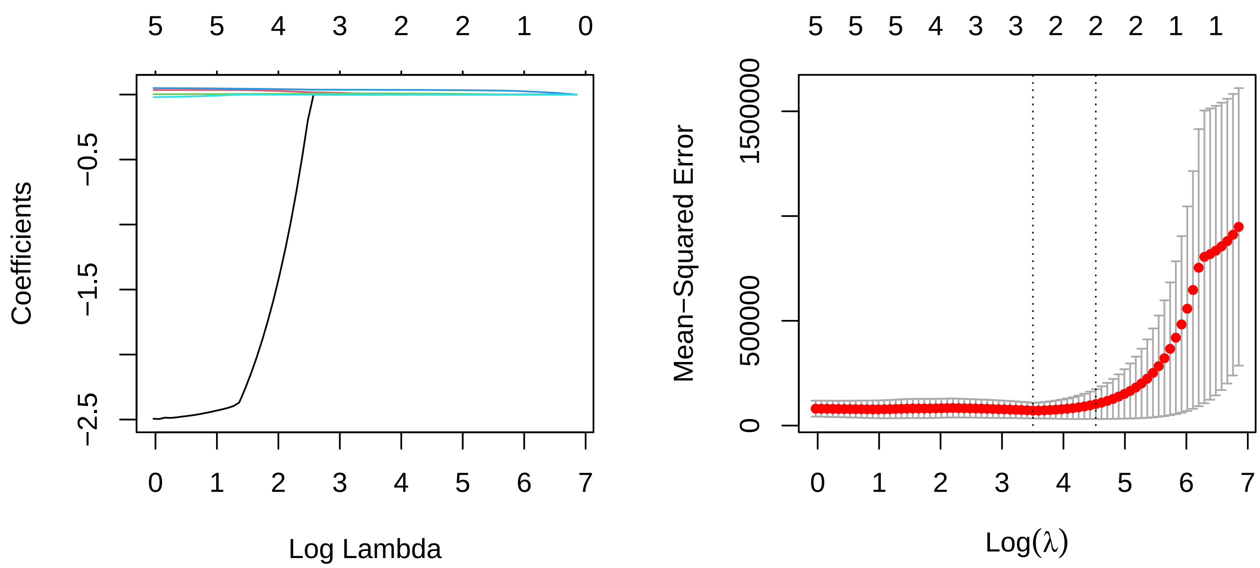
2 d)

```
library(glmnet)
```

```
par(mfrow=c(1,2))
#set.seed(444) # ensuring results don't change each time its run
x = as.matrix(crimes[,-1]) # remove states column
x = x[,-1] # remove expenditure
y = crimes[,2]

train = sample(1:nrow(x),0.67*nrow(x))
x.train = x[train,]; y.train = y[train]
x.test = x[-train,]; y.test = y[-train]

lm.model = lm(expend~bad+crime+lawyers+employ+pop,data=crimes, subset=train)
y.predict.lm = predict(lm.model,newdata=crimes[-train,])
mse.lm = mean((y.test-y.predict.lm)^2)
sprintf("linear model mse = %.3f", mse.lm)
```

```
## [1] "linear model mse = 219347.002"
```

```
lasso.model = glmnet(x.train,y.train,alpha=1)
lasso.cv = cv.glmnet(x.train,y.train,alpha=1,type.measure="mse",nfolds=5)

plot(lasso.model,label=T,xvar="lambda")
plot(lasso.cv)
```




```
lambda.min = lasso.cv$lambda.min; lambda.1se=lasso.cv$lambda.1se;
sprintf("lambda.min = %.3f, lambda.1se = %.3f", lambda.min, lambda.1se)
```

```
## [1] "lambda.min = 33.209, lambda.1se = 92.407"
```

```
coef(lasso.model,s=lasso.cv$lambda.min)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) -46.35094
## bad               .
## crime          0.00495
## lawyers        0.00859
## employ         0.03655
## pop               .
```

```
coef(lasso.model,s=lasso.cv$lambda.1se)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                   s1
## (Intercept) 31.50130
## bad              .
## crime            .
## lawyers      0.00699
## employ       0.03488
## pop              .
```

```
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)
mse2.lasso = mean((y.test-lasso.pred2)^2)
sprintf("mse1 = %.3f, mse2 = %.3f", mse1.lasso, mse2.lasso)
```

```
## [1] "mse1 = 159994.010, mse2 = 209281.192"
```

As we can see from the lambdas calculated by the model above, for the minimum error, the relevant variables are bad, crime, lawyers, and employ. To obtain a reduced model that is within one standard error of the minimum, we can take into account only bad, lawyers, and employ. In general the model is very similar to the one in b). While in b), the variables that were collinear could have been added to the model to improve the R^2, we chose not to do so because the improvement was marginal. The lasso model does something similar, in that it selects one of the aforementioned variables as the primary factor (which has a large coefficient), and the other (believed to be collinear) variables that are shown as relevant by the lasso model have very small coefficients, because as we saw, they had very little effect on the R^2 in b).

Exercise 3: Titanic

3 a)

```
titanic = read.table("titanic.txt", header = T)
titanic$PClass = factor(titanic$PClass)
titanic$Sex = factor(titanic$Sex)
# titanic$Survived = factor(titanic$Survived)

library(plyr)
# round age to nearest 10 years
# (https://stackoverflow.com/a/6466894)
titanic$Ager = round_any(titanic$Age, 20)
titanic$Ager = factor(titanic$Ager)
titanic$Age2 = titanic$Age^2

print("total number of individuals for each combination of class and gender:")
```

```
## [1] "total number of individuals for each combination of class and gender:"
```

```
tot = xtabs(~PClass + Sex, data = titanic)
tot
```

```
##       Sex
## PClass female male
##    1st    143  179
##    2nd    107  173
##    3rd    212  499
```

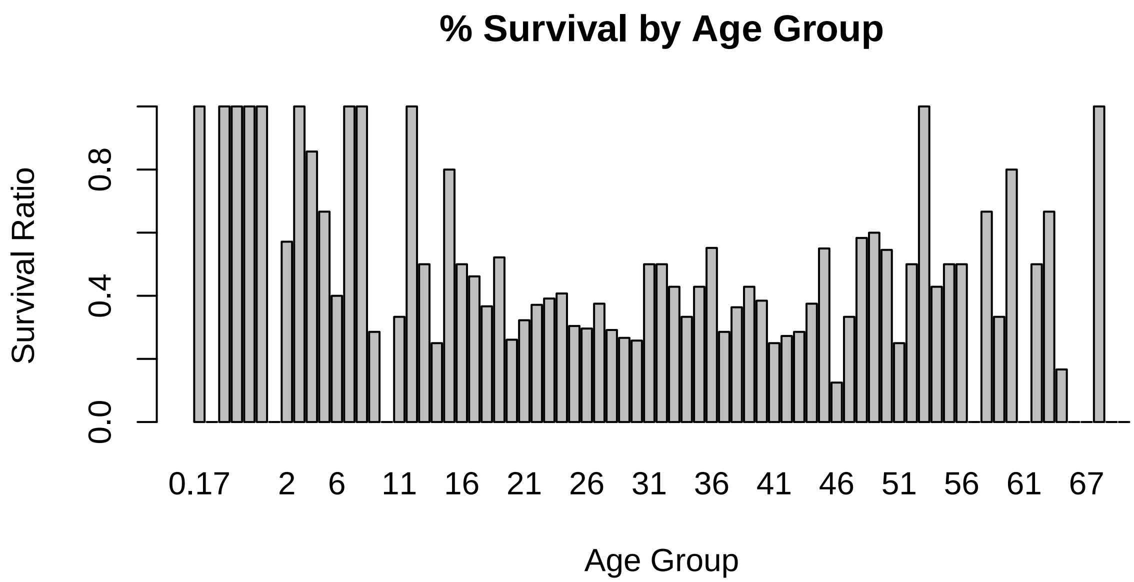
```
print("Survival rate by class and gender combinations:")
```

```
## [1] "Survival rate by class and gender combinations:"
```

```
totc = xtabs(Survived ~ PClass + Sex, data = titanic)
round(totc/tot, 2)
```

```
##       Sex
## PClass female male
##    1st   0.94 0.33
##    2nd   0.88 0.14
##    3rd   0.38 0.12
```

```r
# plot(titanic$Age, y=titanic$Survived, xlim=c(0, 100))
totage = xtabs(~Age, data = titanic)
barplot(xtabs(Survived ~ Age, data = titanic)/totage, main = "% Survival by Age Group",
    xlab = "Age Group", ylab = "Survival Ratio")
```

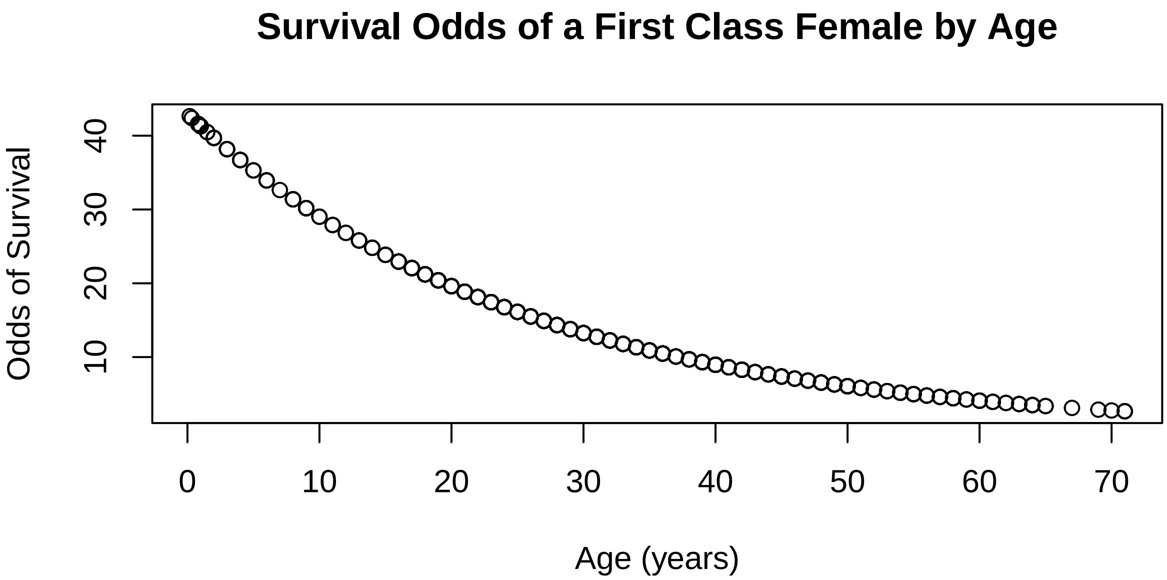



```r
# build logistic regression model
model = glm(Survived ~ Age + Sex + PClass, data = titanic, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = Survived ~ Age + Sex + PClass, family = binomial,
##     data = titanic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.723   -0.707   -0.392    0.649    2.529
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.75966    0.39757    9.46  < 2e-16 ***
## Age         -0.03918    0.00762   -5.14  2.7e-07 ***
## Sexmale     -2.63136    0.20151  -13.06  < 2e-16 ***
## PClass2nd   -1.29196    0.26008   -4.97  6.8e-07 ***
## PClass3rd   -2.52142    0.27666   -9.11  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025.57  on 755  degrees of freedom
```

```
## Residual deviance:  695.14  on 751  degrees of freedom
##   (557 observations deleted due to missingness)
## AIC: 705.1
##
## Number of Fisher Scoring iterations: 5
```

```
plot(titanic$Age, exp(3.759662 + titanic$Age * -0.039177), ylab = "Odds of Survival",
    xlab = "Age (years)", main = "Survival Odds of a First Class Female by Age")
```



In the model above, the intercept can be interpreted as a female who traveled in 1st class. We can see that other classes (namely 2nd and 3rd) lower the odds of survival by the negative coefficients, where 3rd class has a larger impact than 2nd. Similarly, an increase in age has a negative impact on survival odds, however the impact is low because of the parabolic shape of the graph shown above. How we interpret this is that the fact that very young and very old people have the highest survival rates are balancing each other out to form a coefficient that is close to 0, which inaccurately represents the actual relationship between age and survival, because it is clear that it does have an effect. Finally the odds of survival of a male are also significantly lower than that of a female, as shown by the negative coefficient.

3 b)

```
ageclass_model = glm(Survived~Age*PClass+Sex,data=titanic, family=binomial)
summary(ageclass_model)
```

```
##
## Call:
## glm(formula = Survived ~ Age * PClass + Sex, family = binomial,
##     data = titanic)
##
## Deviance Residuals:
##    Min       1Q   Median       3Q      Max
## -2.721  -0.695  -0.399   0.630   2.356
```

```
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)       3.7534     0.5405    6.94  3.8e-12 ***
## Age              -0.0380     0.0117   -3.23   0.0012 **
## PClass2nd        -0.5762     0.6637   -0.87   0.3853
## PClass3rd        -3.0180     0.6351   -4.75  2.0e-06 ***
## Sexmale          -2.6893     0.2058  -13.07  < 2e-16 ***
## Age:PClass2nd    -0.0258     0.0187   -1.38   0.1663
## Age:PClass3rd     0.0211     0.0182    1.16   0.2450
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025.57  on 755  degrees of freedom
## Residual deviance:  689.64  on 749  degrees of freedom
##   (557 observations deleted due to missingness)
## AIC: 703.6
##
## Number of Fisher Scoring iterations: 5
```

```r
agesex_model = glm(Survived~Age*Sex+PClass,data=titanic, family=binomial)
summary(agesex_model)
```

```
##
## Call:
## glm(formula = Survived ~ Age * Sex + PClass, family = binomial,
##     data = titanic)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.435  -0.656  -0.353   0.696   2.728
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.75656    0.43764    6.30  3.0e-10 ***
## Age          0.00244    0.01141    0.21     0.83
## Sexmale     -0.50819    0.44251   -1.15     0.25
## PClass2nd   -1.54337    0.28736   -5.37  7.8e-08 ***
## PClass3rd   -2.65398    0.29142   -9.11  < 2e-16 ***
## Age:Sexmale -0.07559    0.01501   -5.04  4.7e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025.57  on 755  degrees of freedom
```

```
## Residual deviance:  667.08  on 750  degrees of freedom
##   (557 observations deleted due to missingness)
## AIC: 679.1
##
## Number of Fisher Scoring iterations: 5
```

```
male1 = data.frame(Age=55, Sex = "male", PClass = "1st")
male2 = data.frame(Age=55, Sex = "male", PClass = "2nd")
male3 = data.frame(Age=55, Sex = "male", PClass = "3rd")
female1 = data.frame(Age=55, Sex = "female", PClass = "1st")
female2 = data.frame(Age=55, Sex = "female", PClass = "2nd")
female3 = data.frame(Age=55, Sex = "female", PClass = "3rd")

predict(model,female1, type="response")
```

```
##     1
## 0.833
```

```
predict(model,female2, type="response")
```

```
##     1
## 0.578
```

```
predict(model,female3, type="response")
```

```
##     1
## 0.286
```

```
predict(model,male1, type="response")
```

```
##     1
## 0.264
```

```
predict(model,male2, type="response")
```

```
##      1
## 0.0896
```

```
predict(model,male3, type="response")
```

```
##     1
## 0.028
```

Given the results of the two models produced here, as well as the one from 3a), we will select the model that considers the interaction between sex and age, given that the model found that the interaction was significant (the model in 3a) found that sex and age by themselves were also significant). Furthermore, the model that considered the interaction between age and class found that the interaction was not significant ($P > 0.05$ for both 2nd class and 3rd class).

The estimates for the probability of survival of each of the combinations of factors is as shown below:

3 c)

We propose to do k-fold cross-validation, using 10% of the data in each fold, and calculating the average prediction accuracy per combination of factor levels. We would use a slightly modified version of the model from 3b), namely `glm(Survived~Ager*Sex+PClass,data=titanic, family=binomial)` , where the difference would be that the age would be split into several groups (represented as the variable `Ager`.

To be able to predict for the odds resulting from the model, we would use a threshold of 1, where if the odds were above 1, the subject would be predicted as survived and not survived otherwise. (Because when the odds are ≥ 1 , there's a $\geq 50\%$ chance of survival).

3 d)

Based on the (3x2) table in part 3a, every unique combination of classes in this dataset has > 5 individuals, so a chi-squared test is applicable.

To investigate the effect of the factors `PClass` and `Sex` separately, we'll perform two separate chisquare tests which each test the significance of a single factor.

```
# build contigency table
cont1 = as.matrix(xtabs(~PClass, data=titanic)); cont1
```

```
##     [,1]
## 1st  322
## 2nd  280
## 3rd  711
```

```
chisq.test(cont1)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  cont1
## X-squared = 258, df = 2, p-value <2e-16
```

```
cont2 = as.matrix(xtabs(~Sex, data=titanic)); cont2
```

```
##        [,1]
## female  462
## male    851
```

```
chisq.test(cont2)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  cont2
## X-squared = 115, df = 1, p-value <2e-16
```

Both chisquare tests yield p-values < 0.05 so we reject the null hypotheses that Sex and PClass aren't significantly explanatory variables for predicting survival.

3 e)

Comparing the linear regression approach vs the chisquare approach, the linear regression approach has the advantage of being able to make predictions on a given individual's chance of survival (which chisquare has no ability to do), and the linear regression approach can give p-values for each variable.

Both methods can also give insights into the significance of explanatory variables.

Exercise 4: Military Coups

4 a)

```
coups = read.table("coups.txt", header=T)
coups$pollib=factor(coups$pollib)

coupsglm = glm(miltcoup~oligarchy+pollib+parties+pctvote+popn+size+numelec+numregim, family=poi
summary(coupsglm)
```

```
##
## Call:
## glm(formula = miltcoup ~ oligarchy + pollib + parties + pctvote +
##     popn + size + numelec + numregim, family = poisson, data = coups)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.508  -0.953  -0.310   0.486   1.646
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.233427   0.997611   -0.23   0.8150
## oligarchy    0.072566   0.035346    2.05   0.0401 *
## pollib1     -1.103244   0.655811   -1.68   0.0925 .
## pollib2     -1.690306   0.676650   -2.50   0.0125 *
## parties      0.031221   0.011166    2.80   0.0052 **
## pctvote      0.015441   0.010103    1.53   0.1264
## popn         0.010959   0.007149    1.53   0.1253
## size        -0.000265   0.000269   -0.99   0.3244
## numelec     -0.029619   0.069625   -0.43   0.6705
## numregim     0.210943   0.233933    0.90   0.3672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 65.945  on 35  degrees of freedom
## Residual deviance: 28.249  on 26  degrees of freedom
## AIC: 113.1
##
```

```
## Number of Fisher Scoring iterations: 5
```

As can be seen by the results of the Poisson model above, it is seen that the variables *oligarchy*, *pollib*, and *parties* are seen as significant in predicting the number of successful military coups from independence to 1989.

4 b)

```r
# prints just the top 3 least significant coefficients of a
# model:
evalModel = function(model, name) {
    cat(sprintf("\n%s:\n", name))
    res = summary(model)$coefficients
    print(head(res[order(res[, 4], decreasing = T), ], n = 3))
}

coupsglm1 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn + size + numelec + numregim, family = poisson, data = coups)
evalModel(coupsglm1, "coupsglm1")
```

```
##
## coupsglm1:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.2334     0.9976  -0.234    0.815
## numelec      -0.0296     0.0696  -0.425    0.671
## numregim      0.2109     0.2339   0.902    0.367
```

```r
# numelec has highest p-value (0.6705) so we remove it

coupsglm2 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn + size + numregim, family = poisson, data = coups)
evalModel(coupsglm2, "coupsglm2")
```

```
##
## coupsglm2:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.457746   0.860234  -0.532    0.595
## numregim     0.180442   0.224117   0.805    0.421
## size        -0.000269   0.000269  -1.000    0.317
```

```r
# numregim has highest p-value so we remove it

coupsglm3 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn + size, family = poisson, data = coups)
evalModel(coupsglm3, "coupsglm3")
```

```
##
## coupsglm3:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.041976   0.577410  0.0727    0.942
```

```
## size        -0.000258   0.000266 -0.9688    0.333
## popn         0.007165   0.005684  1.2604    0.208
```

```
# size has highest p-value so we remove it

coupsglm4 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn, family = poisson, data = coups)
evalModel(coupsglm4, "coupsglm4")
```

```
##
## coupsglm4:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.23143    0.52889  -0.438    0.662
## popn         0.00566    0.00548   1.032    0.302
## pollib1     -0.68359    0.49582  -1.379    0.168
```

```
# popn has highest p-value so we remove it

coupsglm5 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote,
    family = poisson, data = coups)
evalModel(coupsglm5, "coupsglm5")
```

```
##
## coupsglm5:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.1165    0.51375  -0.227    0.821
## pollib1      -0.6208    0.48753  -1.273    0.203
## pctvote       0.0121    0.00907   1.329    0.184
```

```
# pctvote has highest p-value so we remove it

coupsglm6 = glm(miltcoup ~ oligarchy + pollib + parties, family = poisson,
    data = coups)
evalModel(coupsglm6, "coupsglm6")
```

```
##
## coupsglm6:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.208      0.446   0.467   0.6407
## pollib1       -0.495      0.476  -1.042   0.2976
## pollib2       -1.112      0.459  -2.420   0.0155
```

```
# print('coupsglm6:'); summary(coupsglm6)$coefficients
```

The number of explanatory variables was reduced following the step-down approach, and in doing so the variables were removed in the following order: *numelec*, *numregim*, *size popn*, *pctvote*. This left the model with only relevant explanatory variables (significant as $P < 0.05$). These variables were the same as those seen as relevant in the previous model in 4a), namely *oligarchy*, *pollib*, and *parties*.

4 c)

```
country1 = data.frame(oligarchy = mean(coups$oligarchy),
  pollib = factor(0), parties = mean(coups$parties))
country2 = data.frame(oligarchy = mean(coups$oligarchy),
  pollib = factor(1), parties = mean(coups$parties))
country3 = data.frame(oligarchy = mean(coups$oligarchy),
  pollib = factor(2), parties = mean(coups$parties))

printResults = function() {
  print(sprintf("country1 prediction = %.3f", predict(coupsglm6, country1, type="response")))
  print(sprintf("country2 prediction = %.3f", predict(coupsglm6, country2, type="response")))
  print(sprintf("country3 prediction = %.3f", predict(coupsglm6, country3, type="response")))
}
printResults()
```

```
## [1] "country1 prediction = 2.908"
## [1] "country2 prediction = 1.772"
## [1] "country3 prediction = 0.956"
```

Using the model from 4b), we found that the coefficient of *pollib* was significant and negative. This entails that with an increase in the value of the political liberalization (i.e. with an increase in civil rights), there would be a decrease in the predicted number of successful military coups. The predictions above confirm this.