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)
trees$type = factor(trees$type)

par(mfrow=c(1,3))
model1 = lm(volume~type, data=trees)
pval = checkAnovaNorm(model1, "model1")
```

```
## [1] "Shapiro-Wilk normality p-value for model1 residuals: 0.001"
```

```
model2 = lm(log(volume)~type, data=trees)
pval = checkAnovaNorm(model2, "model2")
```

```
## [1] "Shapiro-Wilk normality p-value for model2 residuals: 0.551"
```

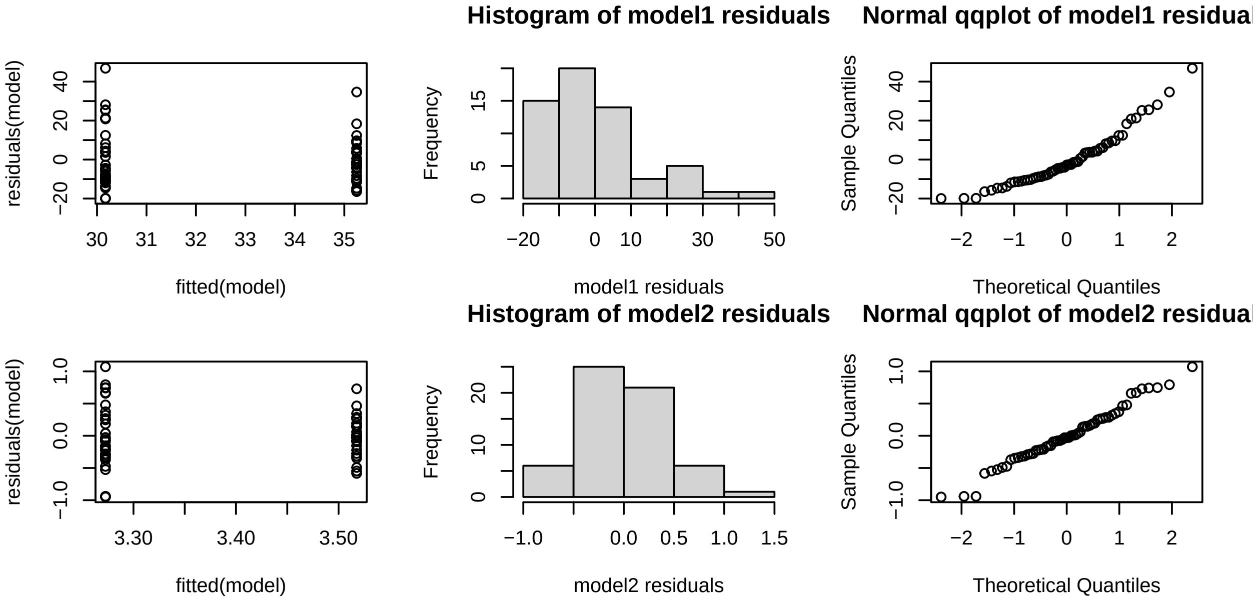
```
print("model coefficients:"); summary(model2)$coefficients
```

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

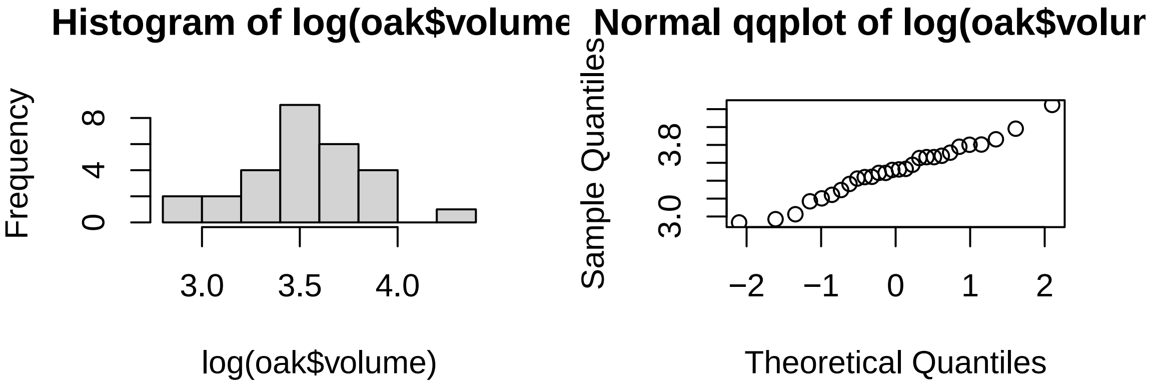
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.273     0.0782   41.84 1.75e-44
## typeoak        0.245     0.1135    2.16 3.52e-02
```

```
res = anova(model2)
sprintf("ANOVA p-value for type = %.3f", res["type", "Pr(>F)"])
```

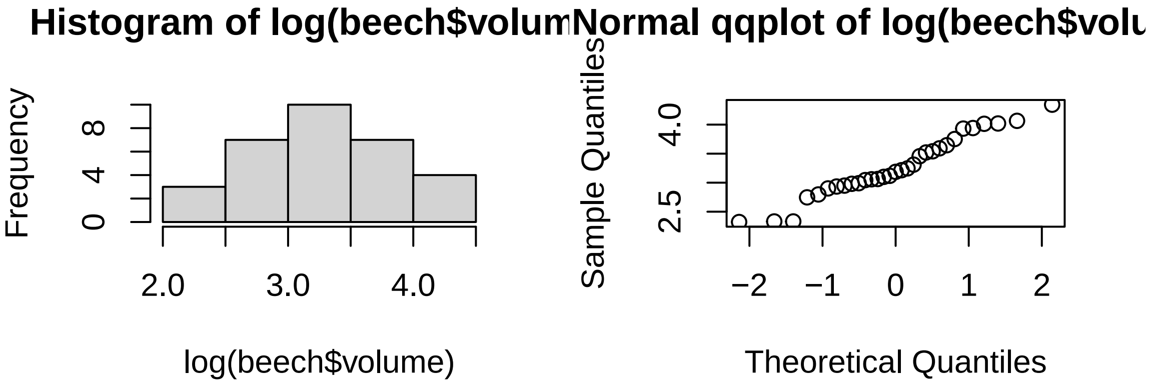
```
## [1] "ANOVA p-value for type = 0.035"
```



We tested two models, and looked at the residuals of each for normality. The results graphs suggests that using `log(volume)` in model2 is preferred as the resulting residuals are normally distributed (whereas this wasn't the case in model1). Additionally, the ANOVA analysis of model2 suggests that we can reject the H_0 that `type` is not a significant predictor of (the log of) `volume` (as the p-value $0.035 < 0.05$).



```
## [1] "Shapiro-Wilk normality p-value for log(oak$volume): 0.902"
```



```
## [1] "Shapiro-Wilk normality p-value for log(beech$volume): 0.377"
## [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. Checking normality, we see that the log transformed volumes of each tree type are normal (whereas the volumes themselves aren't as in 1a).

We can indeed compare the means of the transformed 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.

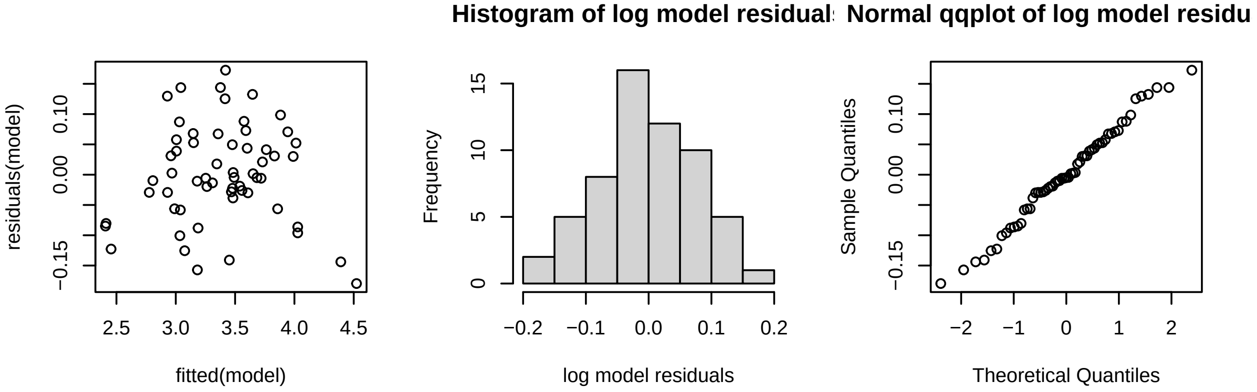
```
new_oak = data.frame(type="oak"); new_beech = data.frame(type = "beech")
pred1 = exp(predict(model2, new_oak)); pred2 = exp(predict(model2, new_beech))
sprintf("predicted volumes: oak = %.3f, beech = %.3f", pred1, pred2)
```

```
## [1] "predicted volumes: oak = 33.706, beech = 26.383"
```

Above we use `model2` to predict the log volume for each tree type, and use `exp()` to convert it to actual volume prediction.

1 b)

```
par(mfrow = c(1, 3))
model = lm(log(volume)~type*diameter+height, data=trees)
pval = checkAnovaNorm(model, "log model")
```



```
## [1] "Shapiro-Wilk normality p-value for log model residuals: 0.788"
```

```
res = drop1(model, test="F")
sprintf("drop1 p-value for type:diameter = %.3f", res["type:diameter", "Pr(>F)"])
```

```
## [1] "drop1 p-value for type:diameter = 0.212"
```

We built a linear model that added an interaction term between diameter and type, the p-value $0.212 > 0.05$ 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 = drop1(model, test="F")
sprintf("drop1 p-value for type:diameter = %.3f", res["type:height", "Pr(>F)"])
```

```
## [1] "drop1 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.

```
model = lm(volume~diameter+height+type, data=trees)
par(mfrow = c(1, 3)); pval = checkAnovaNorm(model, "model")
```

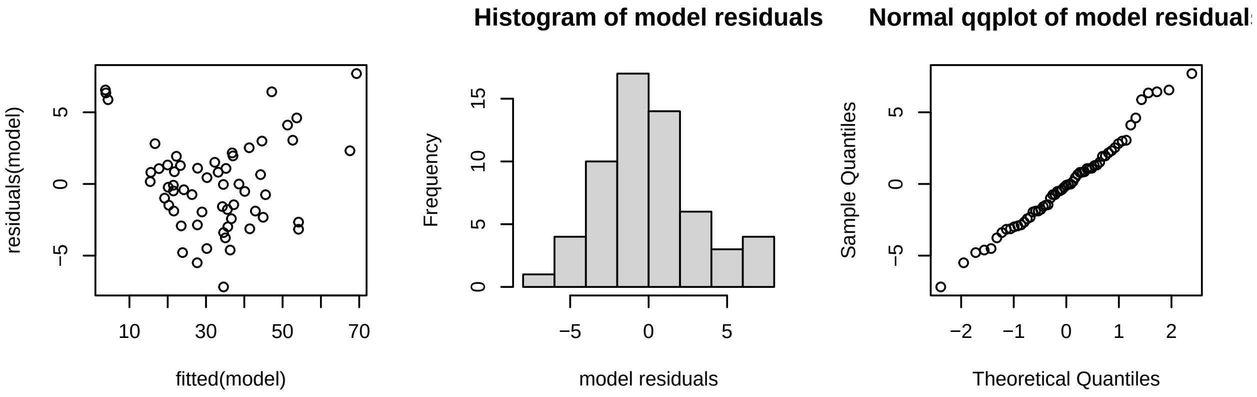
```
## [1] "Shapiro-Wilk normality p-value for model residuals: 0.524"
```

```
print("model coefficients:"); print(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
```

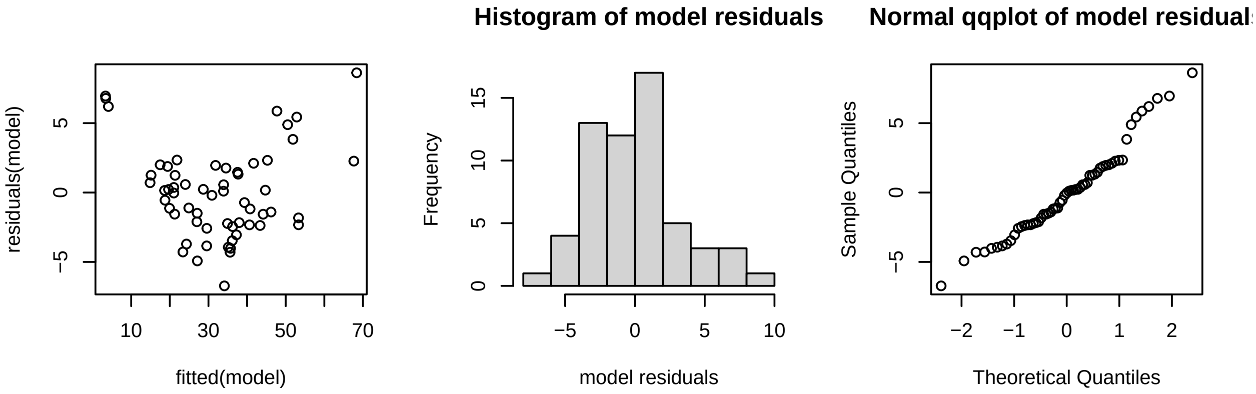
```
drop1(model, test="F")
```

```
## Single term deletions
##
## Model:
## volume ~ diameter + height + type
##          Df Sum of Sq  RSS AIC F value  Pr(>F)
## <none>                 578 143
## diameter  1      8577 9155 304  815.61 < 2e-16 ***
## height    1       324  903 167   30.82 8.4e-07 ***
## type      1        23  602 143    2.21    0.14
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



We built a linear model (where the normality assumptions hold). Based on the drop1 p-values, type is not a significant predictor for volume (p-value 0.14 > 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.

```
# build better model where type isn't considered
modelC = lm(volume~height+diameter, data=trees)
par(mfrow = c(1, 3)); pval = checkAnovaNorm(modelC, "model")
```



```
## [1] "Shapiro-Wilk normality p-value for model residuals: 0.089"
```

```
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"
```

```
#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"
```

We omitted type from our model as it was shown above to be insignificant, we also observe that the normality assumptions hold. 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$cylinder = pi * (trees$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)
```

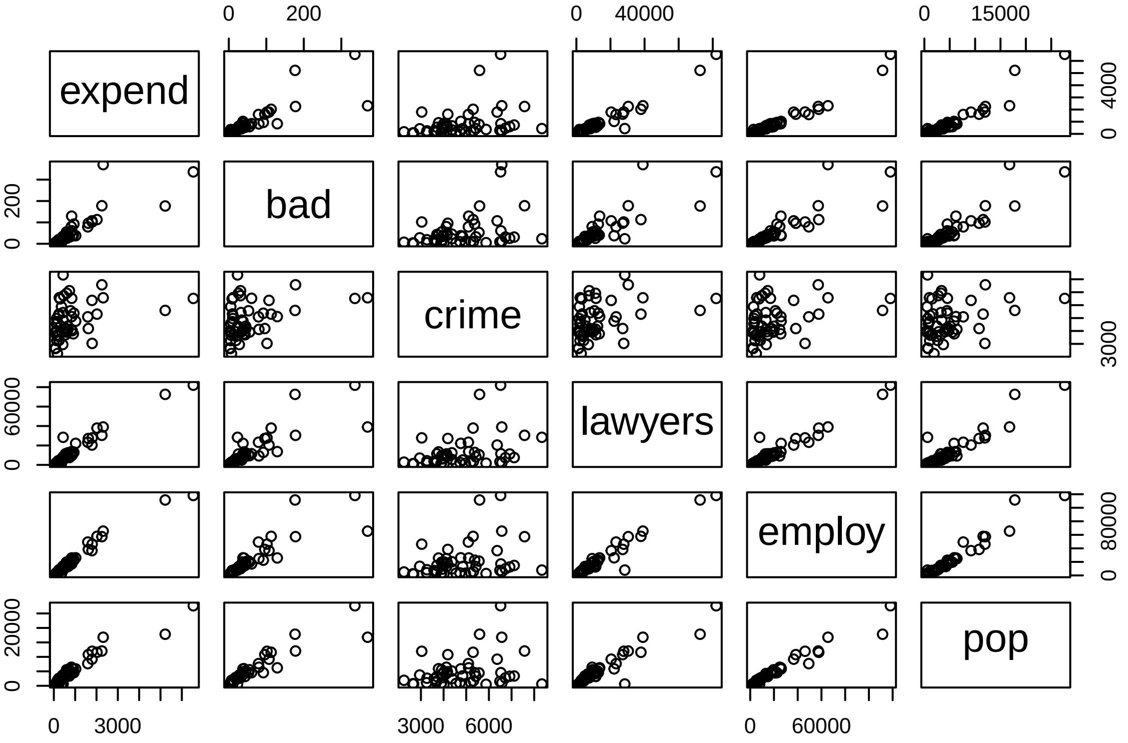
```
## [1] "model: R^2 = 0.975, Adj. R^2 = 0.974"
```

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])
```

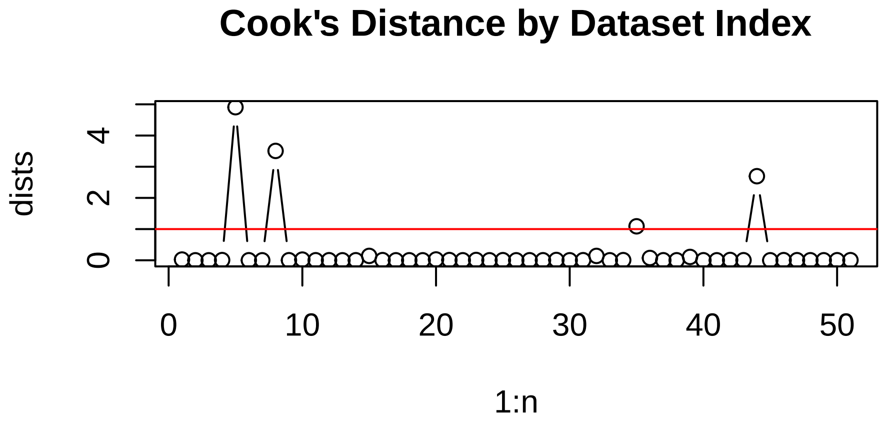


```
crimes$state = factor(crimes$state)
model = lm(expend~crime+bad+lawyers+employ+pop, data=crimes)
summary(model)$coefficients; anova(model)
```

```
##                 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
```

```
## 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
```

```
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
```



```
# 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
```

```
# 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
```

```
# 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("trying var '%s'\t\tPr(>|t|) =  %.3f, model R^2 = %.3f\n", name, pVal, r2))
}

doStepUp = function() {
  cat("\n****round1 (building model with 1 var)****\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 (building on model with employ)****\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")

  cat("\n****round3 (building on model with employ+laywers)****\n")
  evalModel(lm(expend~employ+lawyers+bad, data=crimes), name="bad")
  evalModel(lm(expend~employ+lawyers+crime, data=crimes), name="crime")
  evalModel(lm(expend~employ+lawyers+pop, data=crimes), name="pop")
}
doStepUp()
```

```
##
## ****round1 (building model with 1 var)****
## trying var 'bad'     Pr(>|t|) =  0.000, model R^2 = 0.696
## trying var 'crime'       Pr(>|t|) =  0.016, model R^2 = 0.112
## trying var 'lawyers'     Pr(>|t|) =  0.000, model R^2 = 0.937
## trying var 'employ'      Pr(>|t|) =  0.000, model R^2 = 0.954
## trying var 'pop'     Pr(>|t|) =  0.000, model R^2 = 0.907
##
## ****round2 (building on model with employ)****
## trying var 'bad'     Pr(>|t|) =  0.279, model R^2 = 0.955
## trying var 'crime'       Pr(>|t|) =  0.289, model R^2 = 0.955
## trying var 'lawyers'     Pr(>|t|) =  0.001, model R^2 = 0.963
## trying var 'pop'     Pr(>|t|) =  0.555, model R^2 = 0.954
##
## ****round3 (building on model with employ+laywers)****
## trying var 'bad'     Pr(>|t|) =  0.345, model R^2 = 0.964
## trying var 'crime'       Pr(>|t|) =  0.896, model R^2 = 0.963
## trying var 'pop'     Pr(>|t|) =  0.399, model R^2 = 0.964
```

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 (building up from the model with "employ"), "lawyers" was found to lead to the largest increase in R^2 while still being statistically significant ($p < 0.05$), so we add it to the model. The 3rd round of the "step up" method showed that no further variables could be added while being statistically significant. So our final model is `lm(expend~employ+lawyers, data=crimes)`.

2 c)

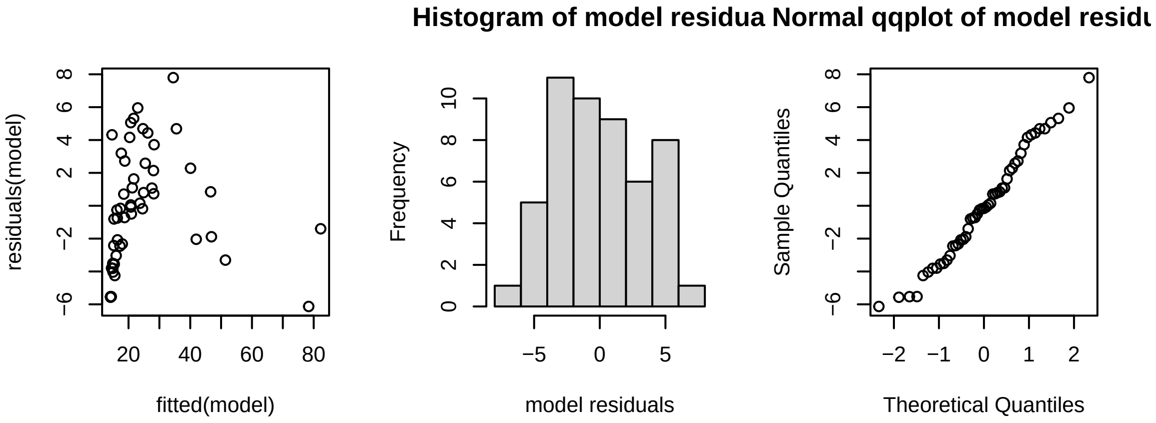
Taking the resulting model from part 2b, we found the normality assumptions weren't met, so we adjusted the model by taking the square root of `expend`. The residuals then indicated normality assumptions were met, however the p-value for `lawyers` was no longer significant.

So we tried the best model from round 1 of step up which was `lm(expend~employ, data=crimes)` (noting that this model was comporable as adding `laywers` had only increased the R^2 by $0.963 - 0.954 = 0.009$). Finally, after checking the normality assumptions of this model, we again saw they weren't met, and so settled on the final model of `lm(sqrt(expend)~employ, data=crimes)` for which the assumptions are now met (the qqplot of residuals below is approximately linear now).

```
# this model meets normality assumptions, but p-value for lawyers is 0.44 > 0.05
#model = lm(sqrt(expend)~employ+lawyers, data=crimes)

model = lm(sqrt(expend)~employ, data=crimes)
summary(model)$coefficients
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.30e+01   6.27e-01    20.8 6.27e-26
## employ      5.86e-04   1.96e-05    29.9 3.73e-33
```

```
par(mfrow = c(1, 3)); pval = checkAnovaNorm(model, "model")
```



```
## [1] "Shapiro-Wilk normality p-value for model residuals: 0.485"
```

```
state = data.frame(bad=50, crime=5000, lawyers=5000, employ=5000, pop=5000)
pred = predict(model, state, interval="prediction")^2

sprintf("predicted expenditure CI ="); pred
```

```
## [1] "predicted expenditure CI ="
```

```
##   fit  lwr upr
## 1 254 80.3 526
```

The predicted interval $[80.3, 526]$ can't be further improved in this case, as both numbers are > 0 and the process of finding a solid model already improved the accuracy of this CI. Arguably, you could consider removing some influence points to further "improve" this CI (but this is risky).

2 d)

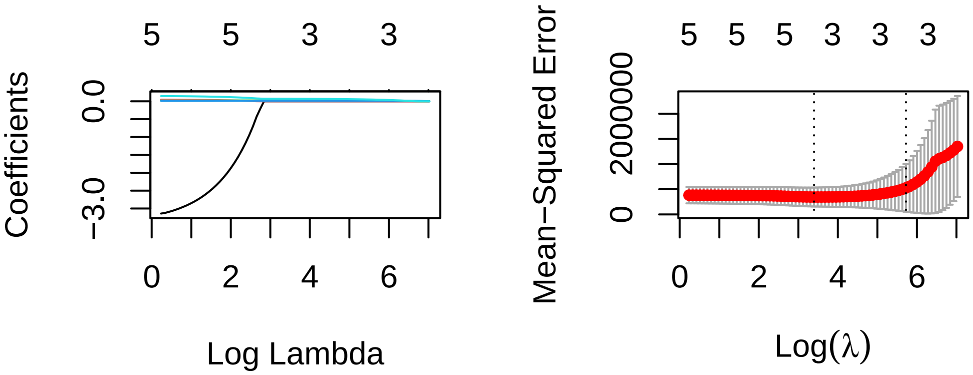
```
library(glmnet)
```

```
par(mfrow=c(1,2))
set.seed(42) # 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 = 54935.728"
```

```
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 = 29.891, lambda.1se = 305.942"
```

```
# lasso min (minimizing R^2)
coef(lasso.model,s=lasso.cv$lambda.min)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) -132.2017
## bad               .
## crime             .
## lawyers        0.0261
## employ         0.0132
## pop            0.0703
```

```
# 1 SE (minimizing number of vars, while remaining within 1 std. err.)
coef(lasso.model,s=lasso.cv$lambda.1se)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 130.4846
## bad              .
## crime            .
## lawyers       0.0171
## employ        0.0132
## pop           0.0449
```

```
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 = 45429.278, mse2 = 169706.874"
```

As we can see from the lambdas calculated by the model above, for the minimum error, the relevant variables are `laywers`, `employ`, and `crime`. To obtain a reduced model that is within one standard error of the minimum, we can take into account only these relevant variables.

Considering the scale differences of the relevant variables, the coefficient of `pop` is not only the largest, but it also has the largest contribution for predictions (as the `pop` variable will always be larger than `employ` and `lawyer` for any given state by a considerable margin).

As a disclaimer, the results from the lasso method will always vary with random chance, however this intrerpretation holds for our resulting model.

The coefficients of the other two relevant variables (`lawyers` and `pop`) are much smaller due to their collinearity with `employ`.

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
```

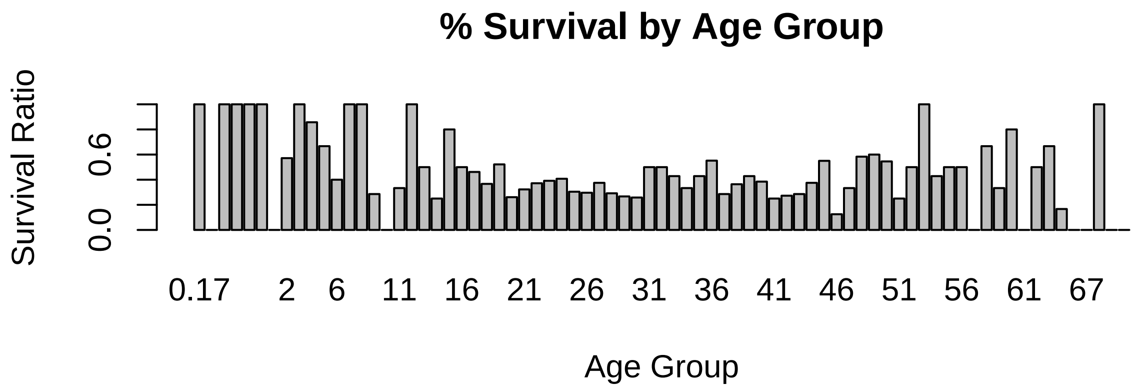
```
print("Survival rate by class and gender combinations:")
```

```
## [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
```

```
# 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")
```



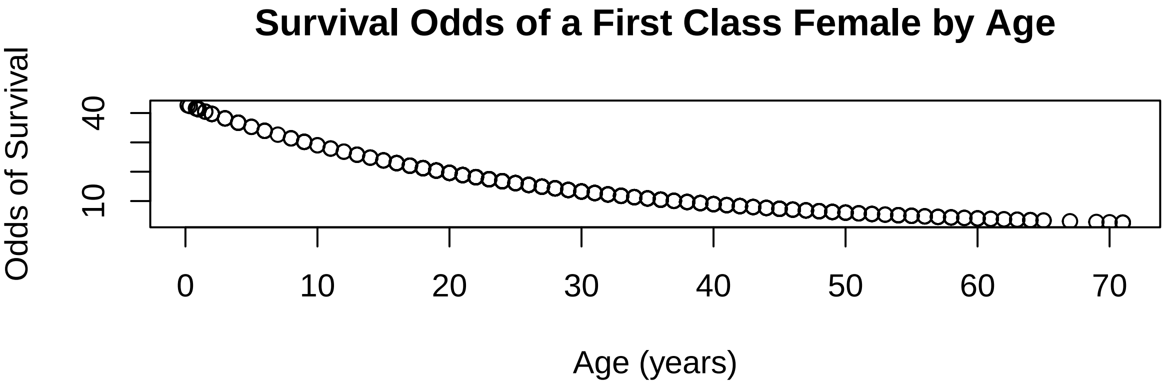
```
# build logistic regression model
model = glm(Survived ~ Age + Sex + PClass, data = titanic, family = binomial)
drop1(model, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## Survived ~ Age + Sex + PClass
##        Df Deviance AIC   LRT Pr(>Chi)
## <none>         695 705
## Age     1      724 732  28.5  9.6e-08 ***
## Sex     1      910 918 214.8  < 2e-16 ***
## PClass  2      796 802 100.4  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model)$coefficients
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.7597    0.39757    9.46 3.18e-21
## Age          -0.0392    0.00762   -5.14 2.69e-07
## Sexmale      -2.6314    0.20151  -13.06 5.68e-39
## PClass2nd    -1.2920    0.26008   -4.97 6.78e-07
## PClass3rd    -2.5214    0.27666   -9.11 7.95e-20
```

```
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 summary 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)
drop1(ageclass_model, test="Chisq")
```

```
## Single term deletions
##
## Model:
## Survived ~ Age * PClass + Sex
##            Df Deviance AIC   LRT Pr(>Chi)
## <none>             690 704
## Sex         1      909 921 219.1   <2e-16 ***
## Age:PClass  2      695 705   5.5    0.064 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
agesex_model = glm(Survived~Age*Sex+PClass,data=titanic, family=binomial)
drop1(agesex_model, test="Chisq")
```

```
## Single term deletions
##
```

```
## Model:
## Survived ~ Age * Sex + PClass
##         Df Deviance AIC   LRT Pr(>Chi)
## <none>          667 679
## PClass   2      771 779 103.5  < 2e-16 ***
## Age:Sex  1      695 705  28.1  1.2e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
predict(model, data.frame(Age=55, Sex = "female", PClass = "1st"), type="response")[[1]]
```

```
## [1] 0.833
```

```
predict(model, data.frame(Age=55, Sex = "female", PClass = "2nd"), type="response")[[1]]
```

```
## [1] 0.578
```

```
predict(model, data.frame(Age=55, Sex = "female", PClass = "3rd"), type="response")[[1]]
```

```
## [1] 0.286
```

```
predict(model, data.frame(Age=55, Sex = "male", PClass = "1st"), type="response")[[1]]
```

```
## [1] 0.264
```

```
predict(model, data.frame(Age=55, Sex = "male", PClass = "2nd"), type="response")[[1]]
```

```
## [1] 0.0896
```

```
predict(model, data.frame(Age=55, Sex = "male", PClass = "3rd"), type="response")[[1]]
```

```
## [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 (`agesex_model`) as this model found that the interaction was significant (and 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.

The estimates for the probability of survival of each of the combinations of factors is show above.

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`, which was calculated in part 3a by rounding ages to the nearest 10 years).

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 individual 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 tests (Chi-squared and Fisher's respectively on the variables) which test the significance of each factor.

```r
# build contingency table
cont1 = as.matrix(xtabs(~PClass+Survived, data=titanic)); cont1
```

```
##       Survived
## PClass   0   1
##    1st 129 193
##    2nd 161 119
##    3rd 573 138
```

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

```
## 
##  Pearson's Chi-squared test
## 
## data:  cont1
## X-squared = 172, df = 2, p-value <2e-16
```

```r
# this is 2x2 so fisher's test is more appropriate
cont2 = as.matrix(xtabs(~Sex+Survived, data=titanic)); cont2
```

```
##         Survived
## Sex        0   1
##   female 154 308
##   male   709 142
```

```r
fisher.test(cont2)
```

```
## 
##  Fisher's Exact Test for Count Data
## 
## data:  cont2
## p-value <2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.0762 0.1316
## sample estimates:
## odds ratio
##        0.1
```

Both the Chi-squared test and Fisher's exact test 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 logistic regression approach (part 3c) vs the Chi-squared approach (part 3d), the logistic regression approach has the advantage of being able to make predictions on a given individual's chance of survival, but the disadvantage of not testing the validity of the underlying assumptions of the logistic regression model.

The Chi-squared approach on the other hand has the disadvantage of requiring you to test each factor individually (as including multiple factors in the contigency table, doesn't allow you to identify which were impactful in the final significance result). Additionally, it can't test the significance of numerical explanatory variables (which is an advantage of the glm approach).

Both methods can also give insights into the significance of categorical explanatory variables by providing p-values for them (e.g. by calling `drop1(agesex_model, test="Chisq")`)

Exercise 4: Military Coups

4 a)

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

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

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.233427   0.997611  -0.234  0.81500
## oligarchy    0.072566   0.035346   2.053  0.04007
## pollib1     -1.103244   0.655811  -1.682  0.09252
## pollib2     -1.690306   0.676650  -2.498  0.01249
## parties      0.031221   0.011166   2.796  0.00517
## pctvote      0.015441   0.010103   1.528  0.12641
## popn         0.010959   0.007149   1.533  0.12531
## size        -0.000265   0.000269  -0.985  0.32444
## numelec     -0.029619   0.069625  -0.425  0.67054
## numregim     0.210943   0.233933   0.902  0.36720
```

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
VERBOSE = F
# prints just the top 3 least significant coefficients of a
# model:
evalModel = function(model, name, verbose = VERBOSE) {
    if (verbose) {
        cat(sprintf("\n%s:\n", name))
        res = drop1(model, test = "Chisq")
        # res = summary(model)$coefficients
        print(head(res[order(res[, 5], decreasing = T), ], n = 3))
    }
}

coupsglm1 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn + size + numelec + numregim, family = poisson, data = coups)
evalModel(coupsglm1, "coupsglm1")
# numelec has highest p-value (0.67) so we remove it

coupsglm2 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn + size + numregim, family = poisson, data = coups)
evalModel(coupsglm2, "coupsglm2")
# numregim has highest p-value (0.42) so we remove it

coupsglm3 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn + size, family = poisson, data = coups)
evalModel(coupsglm3, "coupsglm3")
# size has highest p-value (0.33) so we remove it

coupsglm4 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote +
    popn, family = poisson, data = coups)
evalModel(coupsglm4, "coupsglm4")
# popn has highest p-value (0.31) so we remove it

coupsglm5 = glm(miltcoup ~ oligarchy + pollib + parties + pctvote,
    family = poisson, data = coups)
evalModel(coupsglm5, "coupsglm5")
# pctvote has highest p-value (0.19) so we remove it

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

```
## [1] "coupsglm6 (final model):"
```

```r
drop1(coupsglm6, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## miltcoup ~ oligarchy + pollib + parties
##           Df Deviance AIC   LRT Pr(>Chi)
## <none>           32.8 108
## oligarchy  1     49.5 122 16.64  4.5e-05 ***
## pollib     2     40.0 111  7.20    0.027 *
## parties    1     38.2 111  5.34    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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 (`coupsglm6`) 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.