Homework week 5

2022-06-18

Question 11.1
Question 11.1.1
Answer:

To run a stepwise regression, we need to use the built-in step() function from stats package.
First, read the data.

```
rm(list = ls())
set.seed(19)
data <- read.table("C:\\Data\\week 5 data-summer\\data 11.1\\uscrime.txt",
                   stringsAsFactors = FALSE, header = TRUE)
head(data)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

Next, we need to define two models: one includes intercept term only, the other includes all predictors.

```
#define intercept-only model
intercept_only <- lm(Crime ~ 1, data=data)

#define model with all predictors
all <- lm(Crime ~ ., data=data)
```

Then we can implement stepwise regression. The step() function will use AIC as its variable selection criteria. The info of the final model is displayed below.

```
stepwise <- step(intercept_only, direction='both',
                 scope=list(lower = intercept_only, upper = formula(all)),
                 trace=0)

#view the steps taken to get the final model
stepwise$anova
```

```
##     Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1         NA        NA        46    6880928 561.0235
## 2  + Po1 -1 3253301.8        45    3627626 532.9352
## 3 + Ineq -1  739818.6        44    2887807 524.2154
## 4   + Ed -1  587049.8        43    2300757 515.5343
## 5    + M -1  239404.6        42    2061353 512.3701
## 6 + Prob -1  258062.5        41    1803290 508.0839
## 7   + U2 -1  192233.4        40    1611057 504.7859
```

```
#View the filnal model(scaled data)
summary(stepwise)
```

```
##
## Call:
## lm(formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -470.68  -78.41  -19.68  133.12  556.23
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5040.50     899.84  -5.602 1.72e-06 ***
## Po1           115.02      13.75   8.363 2.56e-10 ***
## Ineq           67.65      13.94   4.855 1.88e-05 ***
## Ed            196.47      44.75   4.390 8.07e-05 ***
## M             105.02      33.30   3.154  0.00305 **
## Prob        -3801.84    1528.10  -2.488  0.01711 *
## U2             89.37      40.91   2.185  0.03483 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.7 on 40 degrees of freedom
## Multiple R-squared:  0.7659, Adjusted R-squared:  0.7307
## F-statistic: 21.81 on 6 and 40 DF,  p-value: 3.418e-11
```

Question 11.1.2
Answer:

To execute a LASSO regression, we need to use package glmnet. We also need to scale the first 15 columns in the uscrime dataset and use them as our predictors.
To perform LASSO regression, parameter “alpha” in the function glmnet() should be set as 1. Since glmnet() will return more than one models with different lambda values, we would directly use function cv.glmnet() to performs 5-fold cross validation and thus identify the lambda value that produces the smallest mean squared error.
The lambda value that generates smallest mean squared error will be used in our final model.

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-4
```

```
#define response variable
y <- as.matrix(data$Crime)

#define matrix of predictor variables (scaled)
x <- data.matrix(scale(data[, 1:15]))
```

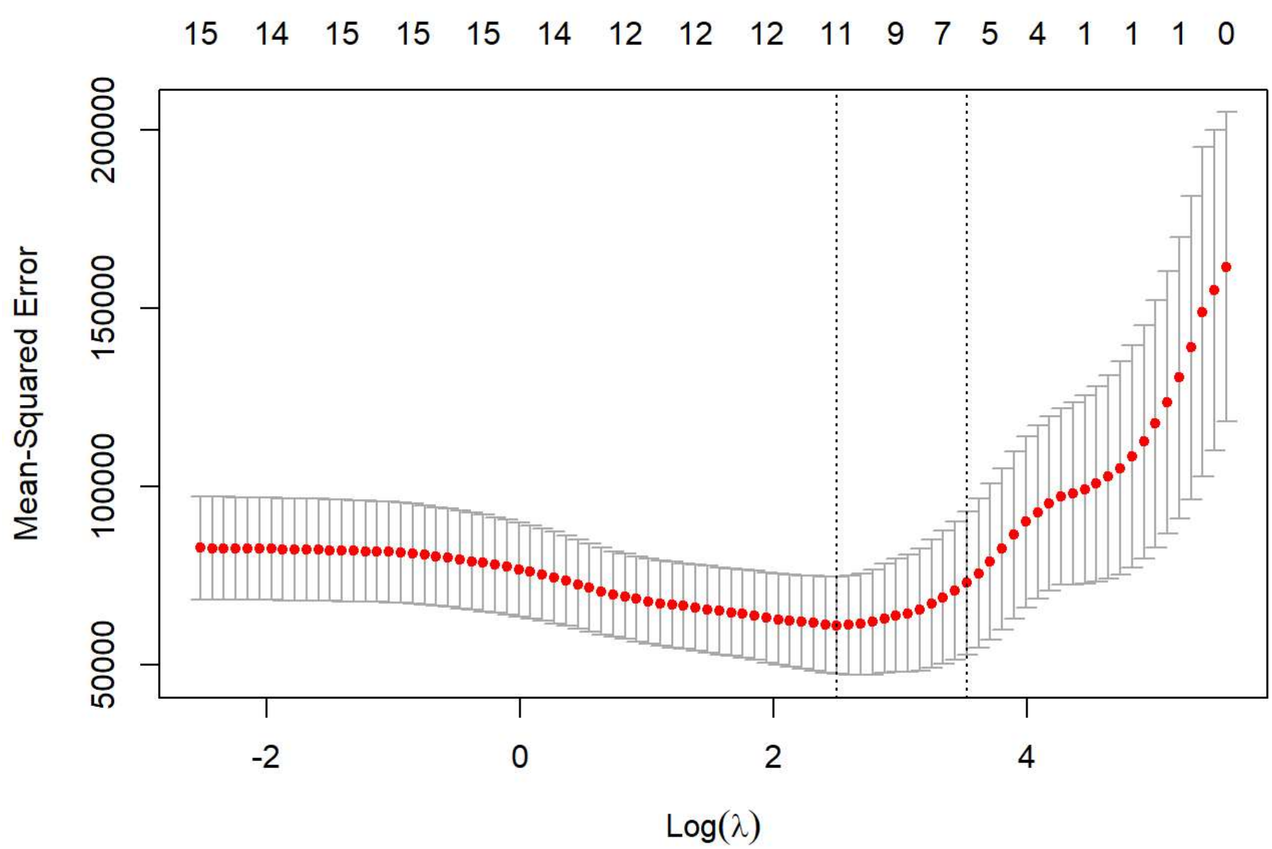
```
set.seed(19)

#3-fold cross validation
model <- cv.glmnet(x, y, alpha = 1, nfolds = 5 )

# lambda value that generate smallest MSE
best_lambda <- model$lambda.min
best_lambda
```

```
## [1] 12.21181
```

```
plot(model)
```



Next we will plug in lambda = 12.21181 to glmnet() to obtain our final model.
Here are the coefficients of the final model (based on scaled data):

```
final_model <- glmnet(x,y, alpha=1, lambda = best_lambda)

#show coefficients of the final model
coef(final_model)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) 905.085106
## M            82.758328
## So           22.929369
## Ed          115.745467
## Po1         310.430186
## Po2            .
## LF            2.352681
## M.F          49.722388
## Pop            .
## NW            3.571258
## U1          -13.474475
## U2           46.340117
## Wealth         .
## Ineq        175.639893
## Prob        -80.369954
## Time           .
```

Question 11.1.3
Answer:

To run the elastic net regression, we are going to use the train() function in caret package. The glmnet() function will be invoked by train() function.
We use train() to automatically select the best tuning parameters, alpha and lambda . The train() function will tests a range of possible alpha and lambda values, and then select the best values for lambda and alpha that will minimize the model’s mean squared error. A 5-fold cross validation approach will be applied in the tuning process.

The best alpha and lambda values and the final model will be displayed as follows:

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
# Scale the data frame
data_scaled <- as.data.frame(cbind(scale(data[,1:15]), data[,16]))
names(data_scaled)[16] <- "Crime"
head(data_scaled)
```

```
##            M         So         Ed        Po1        Po2         LF         M.F
## 1  0.9886930  1.3770536 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.12060499
## 2  0.3521372 -0.7107373  0.6580587  0.6056737  0.5280852  0.5396568  0.98341752
## 3  0.2725678  1.3770536 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.47582390
## 4 -0.2048491 -0.7107373  1.3731746  2.1535064  2.1732150  0.3911854  0.37257228
## 5  0.1929983 -0.7107373  1.3731746  0.8075649  0.7426673  0.7376187  0.06714965
## 6 -1.3983912 -0.7107373  0.3898903  1.1104017  1.2433590 -0.3511718 -0.64550313
##           Pop           NW          U1         U2     Wealth       Ineq
## 1 -0.09500679  1.943738564  0.69510600  0.8313680 -1.3616094  1.6793638
## 2 -0.62033844  0.008483424  0.02950365  0.2393332  0.3276683  0.0000000
## 3 -0.48900552  1.146296747 -0.08143007 -0.1158877 -2.1492481  1.4036474
## 4  3.16204944 -0.205464381  0.36230482  0.5945541  1.5298536 -0.6767585
## 5 -0.48900552 -0.691709391 -0.24783066 -1.6551781  0.5453053 -0.5013026
## 6 -0.30513945 -0.555560788 -0.63609870 -0.5895155  1.6956723 -1.7044289
##         Prob        Time Crime
## 1  1.6497631 -0.05599367   791
## 2 -0.7693365 -0.18315796  1635
## 3  1.5969416 -0.32416470   578
## 4 -1.3761895  0.46611085  1969
## 5 -0.2503580 -0.74759413  1234
## 6 -0.5669349 -0.78996812   682
```

```
# Build the model
set.seed(19)
model <- train(Crime ~.,
               data = data_scaled,
               method = "glmnet",
               trControl = trainControl("cv", number = 5),
)
# Different models tried in the tuning process
model
```

```
## glmnet
##
## 47 samples
## 15 predictors
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 38, 38, 37, 39, 36
## Resampling results across tuning parameters:
##
##   alpha  lambda      RMSE      Rsquared   MAE
##   0.10    0.5261908  275.4363  0.5699113  208.5414
##   0.10    5.2619079  270.8485  0.5427620  207.6630
##   0.10   52.6190793  272.3376  0.5155554  217.4572
##   0.55    0.5261908  275.3085  0.5683242  208.2275
##   0.55    5.2619079  268.9321  0.5451441  207.7069
##   0.55   52.6190793  283.7196  0.5028243  224.2369
##   1.00    0.5261908  275.3899  0.5658953  207.9369
##   1.00    5.2619079  269.4731  0.5439143  208.4011
##   1.00   52.6190793  299.4991  0.4914756  234.4098
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.55 and lambda = 5.261908.
```

```
    #best parameters
    model$bestTune
```

```
##   alpha   lambda
## 5  0.55 5.261908
```

```
    best_alpha <- model$bestTune$alpha
    best_lambda <- model$bestTune$lambda

    # display the final model with best alpha and lambda (scaled data)
    final_model <- glmnet(x,y, alpha=best_alpha, lambda = best_lambda)
    coef(final_model)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) 905.08511
## M           101.81656
## So           17.53785
## Ed          170.49775
## Po1         288.76389
## Po2           .
## LF            .
## M.F          56.03589
## Pop         -16.69157
## NW           18.51147
## U1          -71.97887
## U2          115.30076
## Wealth       54.77997
## Ineq        240.29885
## Prob        -90.46793
## Time          .
```

Question 12.1
Answer:

I am working at a management consulting firm. Recently one of our clients is making advertisements on the internet, and their management what to know which platform (twitter or facebook) is the best channel for online advertising.
I believe this situation is a very good scenario for A/B testing. Our client can broadcast exactly the same advertisements on both two websites. They can determine which website is better channel by viewing the user click rate.

Question 12.2
Answer:
To find a factorial design of this experiment, we simply need to run function FrF2() with following parameters:

nruns = 16 and nfactors = 10

Here are the results:

```
library(FrF2)
```

```
## Loading required package: DoE.base
```

```
## Loading required package: grid
```

```
## Loading required package: conf.design
```

```
## Registered S3 method overwritten by 'partitions':
##   method            from
##   print.equivalence lava
```

```
## Registered S3 method overwritten by 'DoE.base':
##   method           from
##   factorize.factor conf.design
```

```
##
## Attaching package: 'DoE.base'
```

```
## The following objects are masked from 'package:stats':
##
##     aov, lm
```

```
## The following object is masked from 'package:graphics':
##
##     plot.design
```

```
## The following object is masked from 'package:base':
##
##     lengths
```

```
    FrF2(nruns = 16, nfactors = 10)
```

```
##     A  B  C  D  E  F  G  H  J  K
## 1  -1 -1  1 -1  1 -1 -1  1  1 -1
## 2   1  1  1 -1  1  1  1 -1 -1 -1
## 3  -1  1 -1  1 -1  1 -1 -1 -1  1
## 4   1  1 -1 -1  1 -1 -1 -1  1  1
## 5   1 -1 -1  1 -1 -1  1  1  1  1
## 6  -1  1 -1 -1 -1  1 -1  1  1 -1
## 7   1  1  1  1  1  1  1  1  1  1
## 8   1  1 -1  1  1 -1 -1  1 -1 -1
## 9  -1  1  1 -1 -1 -1  1  1 -1  1
## 10 -1 -1 -1 -1  1  1  1  1 -1  1
## 11  1 -1 -1 -1 -1 -1  1 -1 -1 -1
## 12  1 -1  1  1 -1  1 -1  1 -1 -1
## 13 -1  1  1  1 -1 -1  1 -1  1 -1
## 14 -1 -1  1  1  1 -1 -1 -1 -1  1
## 15  1 -1  1 -1 -1  1 -1 -1  1  1
## 16 -1 -1 -1  1  1  1  1 -1  1 -1
## class=design, type= FrF2
```

Question 13.1

Answer:

1. Binomial: in a plant of manufacturing company, there is a detective rate of finished goods coming down from the assembly line. Assuming the defective rate is constant all the time and independent for each piece of finished goods produced, the number of defective products out of total products produced will follow a binomial distribution.

2. Geometric: In the same example as binomial distribution, the number of non-defective products produced before each one defective product is produced will follow a geometric distribution.

3. Poisson: The number of calls received per hour at a call center will follow a poisson distribution.

4. Exponential: In the same example as poisson distribution, the time (minutes) between each call received will follow a exponential distribution.

5. Weibull: The mileage that a newly produced car can run before any failure will follow a Weibull distribution. Since the parts of the car can wear out as it runs longer, the likelihood of failure may increases as mileage increases. Therefore, we would expect $k>1$.