Saagar Shah and Fayha Farooqi Linear Regression Project

Saagar Shah and Fayha Farooqi

2023-04-25

```
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(leaps)
```

Downloading Data for Part 1

```
getSymbols("CFLT", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "CFLT"
```

```
getSymbols("AMD", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "AMD"
```

```
getSymbols("CS", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "CS"
```

```r
getSymbols("ATAT", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "ATAT"
```

```r
getSymbols("YMM", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "YMM"
```

```r
getSymbols("PTON", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "PTON"
```

```r
CFLT = c(CFLT$CFLT.Adjusted)
AMD = c(AMD$AMD.Adjusted)
CS =c(CS$CS.Adjusted)
ATAT = c(ATAT$ATAT.Adjusted)
YMM = c(YMM$YMM.Adjusted)
PTON = c(PTON$PTON.Adjusted)
df = cbind(AMD, CS, ATAT, YMM, PTON)
```

Function for part 1, first takes in a vector of adjusted prices for the stock we are trying to predict, then takes in a data frame with the adjusted close prices of the factors, as well as a column of ones. It then outputs the coefficients of each factor, including the intercept for the ones, and returns p values of the respective classes.

```r
getCoefficients = function(dependentStockVector, df)
{
  df = cbind(1, df)
  xMatrix = matrix(data = df, byrow = TRUE, nrow = dim(df)[2])
  xMatrix = t(xMatrix)
  y = dependentStockVector
  Coefficients = solve(t(xMatrix) %*% xMatrix, tol = NULL) %*% t(xMatrix) %*% y
  residuals = y - xMatrix %*% Coefficients
  RSS = t(residuals) %*% residuals
  df_residuals = nrow(df) - ncol(xMatrix)
  residualStandardError = sqrt(RSS/df_residuals)
  tstat = Coefficients/(residualStandardError * sqrt(diag(solve(t(xMatrix) %*% xMatrix))))
  pValues = 2 * pt(-abs(tstat), df_residuals)
  output = cbind(Coefficients, pValues)
  colNames = colnames(df)
  colNames[1] = "Intercept"
  rownames(output) = colNames
  colnames(output) = c("Coefficients", "pValues")
  return(output)
}
```

Testing the function for part 1, using the data downloaded earlier

```r
getCoefficients(CFLT, df)
```

```
##                Coefficients      pValues
## Intercept       20.39886894 8.261559e-05
## AMD.Adjusted    -0.04107007 3.444727e-01
## CS.Adjusted     -0.46381233 3.285557e-01
## ATAT.Adjusted    0.20189997 1.887375e-01
## YMM.Adjusted    -0.67339357 1.274297e-01
## PTON.Adjusted    0.67052193 7.382756e-04
```

Using the built in lm function to double check my function

```
testdf = cbind(CFLT, AMD, CS, ATAT, YMM, PTON)
model = lm(CFLT~ AMD + CS + ATAT + YMM + PTON, data = testdf)
summary(model)
```

```
##
## Call:
## lm(formula = CFLT ~ AMD + CS + ATAT + YMM + PTON, data = testdf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9347 -0.9463 -0.3183  0.7089  3.8700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 20.39887    4.80361   4.247 8.26e-05 ***
## AMD         -0.04107    0.04308  -0.953 0.344473
## CS          -0.46381    0.47058  -0.986 0.328556
## ATAT         0.20190    0.15174   1.331 0.188737
## YMM         -0.67339    0.43521  -1.547 0.127430
## PTON         0.67052    0.18775   3.571 0.000738 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.337 on 56 degrees of freedom
## Multiple R-squared:  0.4848, Adjusted R-squared:  0.4388
## F-statistic: 10.54 on 5 and 56 DF,  p-value: 3.647e-07
```

Choices for part 2: Forward Subset Selection Adjusted Rsquared Because I am using a forward subset selection, I will start with the most statistically significant factor, then moving down by significance.

This function outputs the RSS when given the y vector and model vector

```
getRSS = function(y, model){
  output = sum((y - model)^2)
  return(output)
}
```

This function outputs the TSS when given the y vector

```
getTSS = function(y){
  ybar = mean(y)
  output = sum((y - ybar)^2)
  return(output)
}
```

This function outputs the AdjRsq when given the y vector, model vector, and d value

```
getAdjRsq = function(y, model, d){
  n = length(y)
  numerator = getRSS(y, model)/(n - d - 1)
  denominator = getTSS(y)/(n-1)
  return(numerator/denominator)
}
```

This function returns the Rsq value when given the y vector and model vector

```
getRsq = function(y, model){
  return(1 - (getRSS(y, model)/getTSS(y)))
}
```

```
getMSE = function(y, prediction){
  return(mean((y - prediction)^2))
}
```

```
testdf = cbind(CFLT, AMD, CS, ATAT, YMM, PTON)
model = lm(CFLT~ AMD + CS + ATAT + YMM + PTON, data = testdf)
summary(model)
```

```
##
## Call:
## lm(formula = CFLT ~ AMD + CS + ATAT + YMM + PTON, data = testdf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9347 -0.9463 -0.3183  0.7089  3.8700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 20.39887    4.80361   4.247 8.26e-05 ***
## AMD         -0.04107    0.04308  -0.953 0.344473
## CS          -0.46381    0.47058  -0.986 0.328556
## ATAT         0.20190    0.15174   1.331 0.188737
## YMM         -0.67339    0.43521  -1.547 0.127430
## PTON         0.67052    0.18775   3.571 0.000738 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.337 on 56 degrees of freedom
## Multiple R-squared:  0.4848, Adjusted R-squared:  0.4388
## F-statistic: 10.54 on 5 and 56 DF,  p-value: 3.647e-07
```

```
subsets = regsubsets(CFLT ~ AMD + CS + ATAT + YMM + PTON, data = testdf, method = "forward")
summary(subsets)
```

```
## Subset selection object
## Call: regsubsets.formula(CFLT ~ AMD + CS + ATAT + YMM + PTON, data = testdf,
##     method = "forward")
```

```
## 5 Variables  (and intercept)
##      Forced in Forced out
## AMD      FALSE      FALSE
## CS       FALSE      FALSE
## ATAT     FALSE      FALSE
## YMM      FALSE      FALSE
## PTON     FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: forward
##          AMD CS  ATAT YMM PTON
## 1  ( 1 ) " " " " " "  " " "*"
## 2  ( 1 ) " " " " " "  "*" "*"
## 3  ( 1 ) " " " " "*"  "*" "*"
## 4  ( 1 ) " " "*" "*"  "*" "*"
## 5  ( 1 ) "*" "*" "*"  "*" "*"
```

This function takes in an input that is the dependent factor, and a data frame of the independent factors and it returns the order in which the factors should be added for models with each number of factors from 1 to k

```r
forward_subset_selection <- function(dependent_factor, independent_factors) {
  num_factors <- ncol(independent_factors)
  current_factors <- vector(mode = "numeric", length = 0)
  best_factors <- NULL
  min_rss <- Inf

  for (i in seq_len(num_factors)) {
    remaining_factors <- setdiff(seq_len(num_factors), current_factors)
    model_rss <- numeric(length = length(remaining_factors))

    for (j in seq_along(remaining_factors)) {
      test_factors <- c(current_factors, remaining_factors[j])
      model_rss[j] <- sum(resid(lm(dependent_factor ~ ., data = independent_factors[, test_factors]))^2
    }

    best_index <- which.min(model_rss)
    if (model_rss[best_index] < min_rss) {
      min_rss <- model_rss[best_index]
      best_factors <- c(current_factors, remaining_factors[best_index])
      current_factors <- best_factors
    } else {
      break
    }
  }

  return(list(best_factors = best_factors))
}
```

```r
setA = CFLT[1:46]
setB = AMD[1:46]
setC = CS[1:46]
setD = ATAT[1:46]
setE = YMM[1:46]
setF = PTON[1:46]
```

```
df = cbind(setB, setC, setD, setE, setF)

model1 = getCoefficients(setA, setF)[1] + getCoefficients(setA, setF)[2]*setF

model2 = getCoefficients(setA, cbind(setD, setF))[1] + getCoefficients(setA, cbind(setD, setF))[2]*setD

model3 = (getCoefficients(setA, cbind(setD, setE, setF))[1] + getCoefficients(setA, cbind(setD, setE, se

model4 = (getCoefficients(setA, cbind(setB, setD, setE, setF))[1] + getCoefficients(setA, cbind(setB, se

model5 = (getCoefficients(setA, df)[1] + getCoefficients(setA, df)[2]*setB + getCoefficients(setA, df)[3

colnames(model1) = c("Model 1")
colnames(model2) = c("Model 2")
colnames(model3) = c("Model 3")
colnames(model4) = c("Model 4")
colnames(model5) = c("Model 5")

adjrsq = rep(0, 5)
adjrsq[1] = getAdjRsq(setA, model1, 1)
adjrsq[2] = getAdjRsq(setA, model2, 2)
adjrsq[3] = getAdjRsq(setA, model3, 3)
adjrsq[4] = getAdjRsq(setA, model4, 4)
adjrsq[5] = getAdjRsq(setA, model5, 5)
bestModel = which.max(adjrsq)
#Based on this, the best model is model1
adjrsq[bestModel]
```

```
## [1] 0.3773474
```

```
setAtest = CFLT[47:length(CFLT)]
setFtest = PTON[47:length(CFLT)]
model1test = getCoefficients(setAtest, setFtest)[1] + getCoefficients(setAtest, setFtest)[2]*setFtest
yTest = CFLT[47:length(CFLT)]
getMSE(yTest, model1test)
```

```
## [1] 1.119763
```

Step 4:

```
setA = CFLT[1:46]
setB = AMD[1:46]
setBsq = setB^2
setC = CS[1:46]
setCsq = setC^2
setD = ATAT[1:46]
setDsq = setD^2
setE = YMM[1:46]
setEsq = setE^2
setF = PTON[1:46]
setFsq = setF^2
df = cbind(setB, setBsq, setC, setCsq, setD, setDsq, setE, setEsq, setF, setFsq)
```

```
df1 = cbind(setF)
df2 = cbind(setF, setFsq)
df3 = cbind(setF, setFsq, setB)
df4 = cbind(setF, setFsq, setB, setDsq)
df5 = cbind(setF, setFsq, setB, setDsq, setE)
df6 = cbind(setF, setFsq, setB, setDsq, setE, setEsq)
df7 = cbind(setF, setFsq, setB, setDsq, setE, setEsq, setD)
df8 = cbind(setF, setFsq, setB, setDsq, setE, setEsq, setD, setBsq)
df9 = cbind(setF, setFsq, setB, setDsq, setE, setEsq, setD, setBsq, setCsq)
df10 = cbind(setF, setFsq, setB, setDsq, setE, setEsq, setD, setBsq, setCsq, setC)


model1 = getCoefficients(setA, df1)[1] + getCoefficients(setA, df1)[2]*setF

model2 = getCoefficients(setA, df2)[1] + getCoefficients(setA, df2)[2]*setF + getCoefficients(setA, df2

model3 = getCoefficients(setA, df3)[1] + getCoefficients(setA, df3)[2]*setF + getCoefficients(setA, df3

model4 = getCoefficients(setA, df4)[1] + getCoefficients(setA, df4)[2]*setF + getCoefficients(setA, df4

model5 = getCoefficients(setA, df5)[1] + getCoefficients(setA, df5)[2]*setF + getCoefficients(setA, df5

model6 = getCoefficients(setA, df6)[1] + getCoefficients(setA, df6)[2]*setF + getCoefficients(setA, df6

model7 = getCoefficients(setA, df7)[1] + getCoefficients(setA, df7)[2]*setF + getCoefficients(setA, df7

model8 = getCoefficients(setA, df8)[1] + getCoefficients(setA, df8)[2]*setF + getCoefficients(setA, df8

model9 = getCoefficients(setA, df9)[1] + getCoefficients(setA, df9)[2]*setF + getCoefficients(setA, df9

model10 = getCoefficients(setA, df10)[1] + getCoefficients(setA, df10)[2]*setF + getCoefficients(setA, 


colnames(model1) = c("Model 1")
colnames(model2) = c("Model 2")
colnames(model3) = c("Model 3")
colnames(model4) = c("Model 4")
colnames(model5) = c("Model 5")
colnames(model6) = c("Model 6")
colnames(model7) = c("Model 7")
colnames(model8) = c("Model 8")
colnames(model9) = c("Model 9")
colnames(model10) = c("Model 10")

adjrsq = rep(0, 10)
adjrsq[1] = getAdjRsq(setA, model1, 1)
adjrsq[2] = getAdjRsq(setA, model2, 2)
adjrsq[3] = getAdjRsq(setA, model3, 3)
adjrsq[4] = getAdjRsq(setA, model4, 4)
adjrsq[5] = getAdjRsq(setA, model5, 5)
adjrsq[6] = getAdjRsq(setA, model1, 6)
adjrsq[7] = getAdjRsq(setA, model2, 7)
adjrsq[8] = getAdjRsq(setA, model3, 8)
```

```r
adjrsq[9] = getAdjRsq(setA, model4, 9)
adjrsq[10] = getAdjRsq(setA, model5, 10)
bestModel = which.max(adjrsq)
#Based on this, the best model is model6
adjrsq[bestModel]
```

```
## [1] 0.4257252
```

```r
setAtest = CFLT[47:length(CFLT)]
setBtest = AMD[47:length(AMD)]
setBsqtest = setBtest^2
setCtest = CS[47:length(CS)]
setCsqtest = setCtest^2
setDtest = ATAT[47:length(ATAT)]
setDsqtest = setDtest^2
setEtest = YMM[47:length(YMM)]
setEsqtest = setEtest^2
setFtest = PTON[47:length(PTON)]
setFsqtest = setFtest^2
df6test = cbind(setFtest, setFsqtest, setBtest, setDsqtest, setEtest, setEsqtest)
prediction = getCoefficients(setAtest, df6test)[1] + getCoefficients(setAtest, df6test)[2]*setFtest + ge
yTest = CFLT[47:length(CFLT)]
getMSE(yTest, prediction)
```

```
## [1] 0.2231844
```

```r
getSymbols("DAL", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "DAL"
```

```r
getSymbols("NET", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "NET"
```

```r
getSymbols("MTCH", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "MTCH"
```

```r
getSymbols("JPM", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "JPM"
```

```r
getSymbols("MDB", from = "2023-01-26", to = "2023-04-26", warnings = FALSE, auto.assign = TRUE)
```

```
## [1] "MDB"
```

```
DAL = c(DAL$DAL.Adjusted)
NET = c(NET$NET.Adjusted)
MTCH =c(MTCH$MTCH.Adjusted)
JPM = c(JPM$JPM.Adjusted)
MDB = c(MDB$MDB.Adjusted)
```

step5:

```
setA = CFLT[1:46]
setB = AMD[1:46]
setBsq = setB^2
setC = CS[1:46]
setCsq = setC^2
setD = ATAT[1:46]
setDsq = setD^2
setE = YMM[1:46]
setEsq = setE^2
setF = PTON[1:46]
setFsq = setF^2
setG = DAL[1:46]
setH = NET[1:46]
setI = MTCH[1:46]
setJ = JPM[1:46]
setK = MDB[1:46]

df = cbind(setB, setBsq, setC, setCsq, setD, setDsq, setE, setEsq, setF, setFsq, setG, setH, setI, setJ

forward_subset_selection(setA, df)
```

```
## $best_factors
##  [1]  9 12 11 15 14  3  1  4 13  2  8  6  5  7 10
```

```
df1 = cbind(setK)
df2 = cbind(setK, setF)
df3 = cbind(setK, setF, setH)
df4 = cbind(setK, setF, setH, setD)
df5 = cbind(setK, setF, setH, setD, setJ)
df6 = cbind(setK, setF, setH, setD, setJ, setC)
df7 = cbind(setK, setF, setH, setD, setJ, setC, setI)
df8 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG)
df9 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG, setE)
df10 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG, setE, setBsq)
df11 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG, setE, setBsq, setB)
df12 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG, setE, setBsq, setB, setCsq)
df13 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG, setE, setBsq, setB, setCsq, setDsq)
df14 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG, setE, setBsq, setB, setCsq, setDsq, setFsq)
df15 = cbind(setK, setF, setH, setD, setJ, setC, setI, setG, setE, setBsq, setB, setCsq, setDsq, setFsq

coefficients1 = getCoefficients(setA, df1)
model1 = coefficients1[1] + coefficients1[2]*setK

coefficients2 = getCoefficients(setA, df2)
```

```
model2 = coefficients2[1] + coefficients2[2]*setK + coefficients2[3]*setF

coefficients3 = getCoefficients(setA, df3)
model3 = coefficients3[1] + coefficients3[2]*setK + coefficients3[3]*setF + coefficients3[4]*setH

coefficients4 = getCoefficients(setA, df4)
model4 = coefficients4[1] + coefficients4[2]*setK + coefficients4[3]*setF + coefficients4[4]*setH + coe

coefficients5 = getCoefficients(setA, df5)
model5 = coefficients5[1] + coefficients5[2]*setK + coefficients5[3]*setF + coefficients5[4]*setH + coe

coefficients6 = getCoefficients(setA, df6)
model6 = coefficients6[1] + coefficients6[2]*setK + coefficients6[3]*setF + coefficients6[4]*setH + coe

coefficients7 = getCoefficients(setA, df7)
model7 = coefficients7[1] + coefficients7[2]*setK + coefficients7[3]*setF + coefficients7[4]*setH + coe

coefficients8 = getCoefficients(setA, df8)
model8 = coefficients8[1] + coefficients8[2]*setK + coefficients8[3]*setF + coefficients8[4]*setH + coe

coefficients9 = getCoefficients(setA, df9)
model9 = coefficients9[1] + coefficients9[2]*setK + coefficients9[3]*setF + coefficients9[4]*setH + coe

coefficients10 = getCoefficients(setA, df10)
model10 = coefficients10[1] + coefficients10[2]*setK + coefficients10[3]*setF + coefficients10[4]*setH

coefficients11 = getCoefficients(setA, df11)
model11 = coefficients11[1] + coefficients11[2]*setK + coefficients11[3]*setF + coefficients11[4]*setH

coefficients12 = getCoefficients(setA, df12)
model12 = coefficients12[1] + coefficients12[2]*setK + coefficients12[3]*setF + coefficients12[4]*setH

coefficients13 = getCoefficients(setA, df13)
model13 = coefficients13[1] + coefficients13[2]*setK + coefficients13[3]*setF + coefficients13[4]*setH

coefficients14 = getCoefficients(setA, df14)
model14 = coefficients14[1] + coefficients14[2]*setK + coefficients14[3]*setF + coefficients14[4]*setH

coefficients15 = getCoefficients(setA, df15)
model15 = coefficients15[1] + coefficients15[2]*setK + coefficients15[3]*setF + coefficients15[4]*setH

colnames(model1) = c("Model 1")
colnames(model2) = c("Model 2")
colnames(model3) = c("Model 3")
colnames(model4) = c("Model 4")
colnames(model5) = c("Model 5")
colnames(model6) = c("Model 6")
colnames(model7) = c("Model 7")
colnames(model8) = c("Model 8")
colnames(model9) = c("Model 9")
colnames(model10) = c("Model 10")
colnames(model11) = c("Model 11")
colnames(model12) = c("Model 12")
```

```r
colnames(model13) = c("Model 13")
colnames(model14) = c("Model 14")
colnames(model15) = c("Model 15")

adjrsq = rep(0, 15)
adjrsq[1] = getAdjRsq(setA, model1, 1)
adjrsq[2] = getAdjRsq(setA, model2, 2)
adjrsq[3] = getAdjRsq(setA, model3, 3)
adjrsq[4] = getAdjRsq(setA, model4, 4)
adjrsq[5] = getAdjRsq(setA, model5, 5)
adjrsq[6] = getAdjRsq(setA, model6, 6)
adjrsq[7] = getAdjRsq(setA, model7, 7)
adjrsq[8] = getAdjRsq(setA, model8, 8)
adjrsq[9] = getAdjRsq(setA, model9, 9)
adjrsq[10] = getAdjRsq(setA, model10, 10)
adjrsq[11] = getAdjRsq(setA, model11, 11)
adjrsq[12] = getAdjRsq(setA, model12, 12)
adjrsq[13] = getAdjRsq(setA, model13, 13)
adjrsq[14] = getAdjRsq(setA, model14, 14)
adjrsq[15] = getAdjRsq(setA, model15, 15)
bestModel = which.max(adjrsq)
#Based on this, the best model is model1
adjrsq[bestModel]
```

```
## [1] 0.7110442
```

```r
setKtest = MDB[47:length(MDB)]
coefficientstest = getCoefficients(setAtest, setKtest)
prediction = coefficientstest[1] + coefficientstest[2]*setKtest
yTest = CFLT[47:length(CFLT)]
getMSE(yTest, prediction)
```

```
## [1] 0.6945658
```

The model with the lowest MSE was step 4. This is good because it indicates that the model is accurately predicting the target variable, meaning that it is making fewer errors. A low MSE can also be a sign that the model is not overfitting the data, meaning it starts to memorize the training data instead of learning the underlying patterns which can lead to poor performance on new data. When comparing multiple models, the one with the lowest MSE is generally considered to be the best performing. The model with the highest adjusted R^2 is step 5. This is good because it indicates that the independent variables in the model are good predictors of the dependent variable. It can also help to identify which independent variables are most important in predicting the dependent variable, by looking at the coefficients of the independent variables in the model. When comparing multiple models, the one with the highest adjusted R-squared is generally considered to be the best performing. The optimal model we would rely on would be step 4 because it has the lowest MSE, however the model with the largest adjusted R squared is the model from step 5, meaning it was a better fit for the training data, however it wasn't as good of a predictor as the model from step 4.

The model that we found in step 4 that we could use for predicting CFLT was the following:

y = -46.285+16.390$*PTON-$0.845$*PTON^2+$0.169$*AMD+$0.0033$*ATAT^2-$8.648$*YMM+$0.711$*YMM^2$