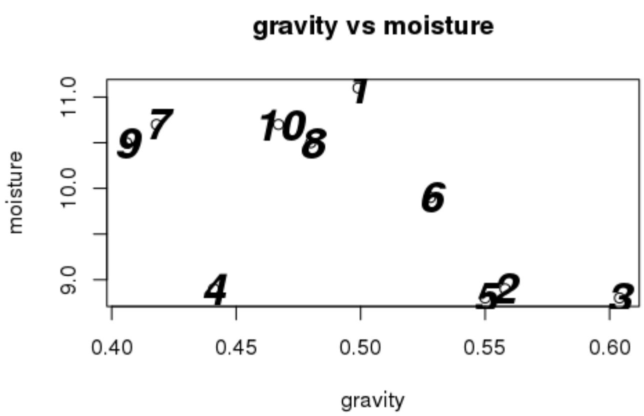
Naomi Kaduwela
Predictive Analytics HW # 4
Nov 2, 2018

Chapter 4, Exercises 4.8, 4.10, 4.11
Chapter 5, Exercises 5.4, 5.5

<u>**4.8**</u>

a.) Scatter Plot of 2 predictors (gravity and moisture) to look for influential variables:
Visually the bottom left point (#4) looks like outlier on the bottom



b.) Influential variables based on Hat Matrix: hii > 2(p+1)/n
Here we see #4 is the only influential variable as hii = 0.604 > the threshold and not the same variable that was identified by the hat matrix

```
> no <- c(1,2,3,4,5,6,7,8,9,10)
> x1gravity <- c(0.499, 0.558, 0.604, 0.441, 0.550, 0.528, 0.418, 0.480, 0.406, 0.467)
> x2moisture <- c(11.1, 8.9, 8.8, 8.9, 8.8, 9.9, 10.7, 10.5, 10.5, 10.7)
> yStrength <- c(11.14, 12.74, 13.13, 11.51, 12.38, 12.60, 11.13, 11.70, 11.02, 11.41)
> wood <- data.frame(x1gravity,x2moisture,yStrength, no)
> woodLM <- lm(yStrength ~ x1gravity + x2moisture)
> plot(x = x1gravity,y = x2moisture, xlab = "gravity", ylab = "moisture", main = "gravity vs moisture") + text(x2moisture~x
1gravity, labels=wood$no, cex = 2, font = 4)
integer(0)
> lm.influence(woodLM)$hat
        1         2         3         4         5         6         7         8         9        10 
0.4178935 0.2418666 0.4172806 0.6043904 0.2521824 0.1478688 0.2616385 0.1540321 0.3155106 0.1873364 

> # Threshold value: 2(p+1)/n
> h_threshold <- 2*(2+1)/10
> h_threshold
[1] 0.6
```

c.) Cook's distance Influencing variables
See only #1 is > than the threshold with value: 1.0692 making it the single influencer var

```
> cook_distance_threshold <- 4/(10-(2+1)) # 4/n-(p+1)
> cook_distance_threshold # = 0.571
[1] 0.5714286

> #Plot cook's distance
> cook <- cooks.distance(woodLM)
> print(cook)
           1            2            3            4            5            6            7            8            9
1.069240e+00 3.723020e-03 9.208950e-03 4.756415e-01 1.238171e-01 1.810604e-01 3.418372e-02 1.303120e-02 1.288740e-02
          10
9.905523e-05
>
```

d.) Fit equation: y = B0 + B1x1 + B2x2 + E

With the influencer variable removed - #4, we see that the Beta coefficients change:

- less weight to x1gravity (8.49 ->6.799) and more weight to x2moisture (-0.266 -> -0.39)
- but R^2 hasn't changed much (.9 -> .91) nor did any of the p values become significant

Original Model:

```
> woodLM <- lm(yStrength ~ x1gravity + x2moisture)
> summary(woodLM)

Call:
lm(formula = yStrength ~ x1gravity + x2moisture)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.44422 -0.12780  0.05365  0.10521  0.44985 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  10.3015     1.8965   5.432 0.000975 ***
x1gravity     8.4947     1.7850   4.759 0.002062 ** 
x2moisture   -0.2663     0.1237  -2.152 0.068394 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2754 on 7 degrees of freedom
Multiple R-squared:    0.9,	Adjusted R-squared:  0.8714 
F-statistic:  31.5 on 2 and 7 DF,  p-value: 0.0003163

> woodLM$coefficients
(Intercept)   x1gravity  x2moisture 
 10.3015238   8.4947108  -0.2663214 
```

New Model:

```
> woodLM_withouInfluencer <- lm(woodLM_withouInfluencer$yStrength ~ woodLM_withouInfluencer$x1gravity + woodLM_withouInflue
ncer$x2moisture)
> summary(woodLM_withouInfluencer)

Call:
lm(formula = woodLM_withouInfluencer$yStrength ~ woodLM_withouInfluencer$x1gravity +
    woodLM_withouInfluencer$x2moisture)

Residuals:
     Min       1Q   Median       3Q      Max
-0.33339 -0.05037  0.01127  0.05615  0.46579

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)
(Intercept)                         12.4107     2.9071   4.269  0.00527 **
woodLM_withouInfluencer$x1gravity    6.7992     2.5166   2.702  0.03549 *
woodLM_withouInfluencer$x2moisture  -0.3905     0.1794  -2.177  0.07237 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.277 on 6 degrees of freedom
Multiple R-squared:  0.9108,	Adjusted R-squared:  0.8811
F-statistic: 30.65 on 2 and 6 DF,  p-value: 0.0007089

> woodLM_withouInfluencer$coefficients
                       (Intercept)  woodLM_withouInfluencer$x1gravity woodLM_withouInfluencer$x2moisture
                        12.4106577                          6.7992142                         -0.3905492
```

4.10

a.) None of the predictors in the correlation matrix are > 0.5, thus there are no indications of multicollinearity

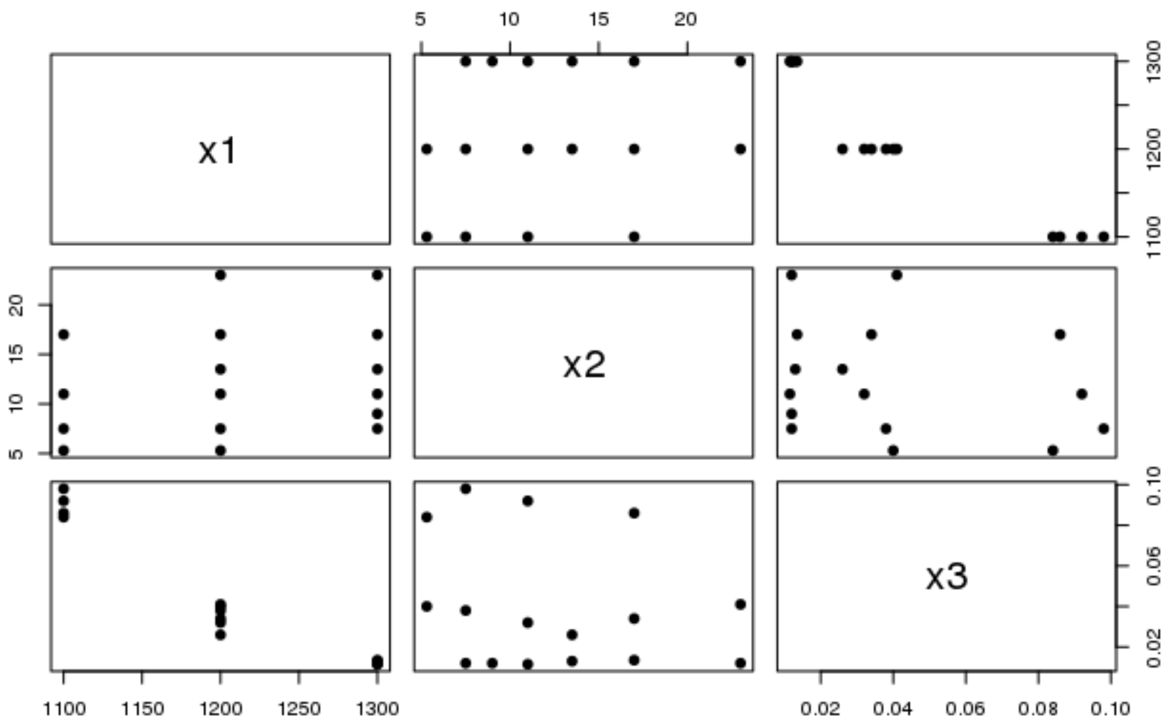
```
> no <- c(1,2,3,4,5,6,7,8,9,10,11,12)
> x1 <- c(8,8,8,0,0,0,2,2,2,0,0,0)
> x2 <- c(1,1,1,0,0,0,7,7,7,0,0,0)
> x3 <- c(1,1,1,9,9,9,0,0,0,0,0,0)
> x4 <- c(1,0,0,1,1,1,1,1,1,10,10,10)
> linearDependence <- data.frame(x1,x2,x3,x4)
> linearDependence
   x1 x2 x3 x4
1   8  1  1  1
2   8  1  1  0
3   8  1  1  0
4   0  0  9  1
5   0  0  9  1
6   0  0  9  1
7   2  7  0  1
8   2  7  0  1
9   2  7  0  1
10  0  0  0 10
11  0  0  0 10
12  0  0  0 10
> #a.) Calculate the Correlation Matrix and confirm that absolute value( correlation ) !> 0.5
> corMatrix <- cor(linearDependence)
> print(corMatrix)
            x1          x2         x3         x4
x1  1.00000000  0.05230658 -0.3433818 -0.4976109
x2  0.05230658  1.00000000 -0.4315953 -0.3706964
x3 -0.34338179 -0.43159531  1.0000000 -0.3551214
x4 -0.49761095 -0.37069641 -0.3551214  1.0000000
```

b.) All VIF values > 150 and the max = 289.3750, indicating multicollinearity

```
> #b.) VIF calculation shows all X's > 150 and max = 289.3750 showing multicolinearity
> library(usdm)
> vif(linearDependence)
  Variables      VIF
1        x1 178.2874
2        x2 158.0460
3        x3 257.9074
4        x4 289.3750
>
```

4.11

a.) x1 & x3 strongly negatively correlated (-0.958)




```
> #a.) Plot 3 predictor variables
> #model
> acetyleneLM <- lm(y ~ x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + I(x1^2) + I(x2^2) + I(x3^2), data = a
cetylene)
> pairs(acetylene[,c("x1", "x2", "x3")], pch=19)
> cor(acetylene[,c("x1", "x2", "x3")])
           x1         x2         x3
x1  1.0000000  0.2236278 -0.9582041
x2  0.2236278  1.0000000 -0.2402310
x3 -0.9582041 -0.2402310  1.0000000
>
```

b.) VIF values are large, indicating multicollinearity

```
> vif(acetyleneLM)
          x1           x2           x3      I(x1^2)      I(x2^2)      I(x3^2)        x1:x2
2.856749e+06 1.095614e+04 2.017163e+06 2.501945e+06 6.573359e+01 1.266710e+04 9.802903e+03
       x1:x3        x2:x3
1.428092e+06 2.403594e+02
```

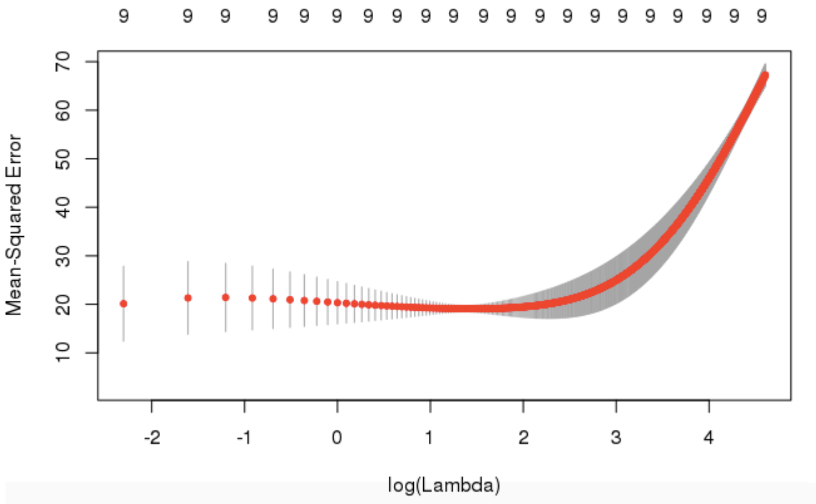
c.) Looking at the VIF values after centering the predictor variables (x's - mean), we see that centering has made the multicollinearity problem less severe.

```
> # vif numbers are still large
> acetyleneLM_mean <- lm(y ~ x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + I(x1^2) + I(x2^2) + I(x3^2), dat
a = acetylene_mean)
> vif(acetyleneLM_mean)
         x1          x2          x3     I(x1^2)     I(x2^2)     I(x3^2)       x1:x2
 375.247759    1.740631  680.280039 1762.575365    3.164318 1156.766284   31.037059
      x1:x3       x2:x3
6563.345193   35.611286
>
```

5.4

Optimum lambda for ridge regression = 0

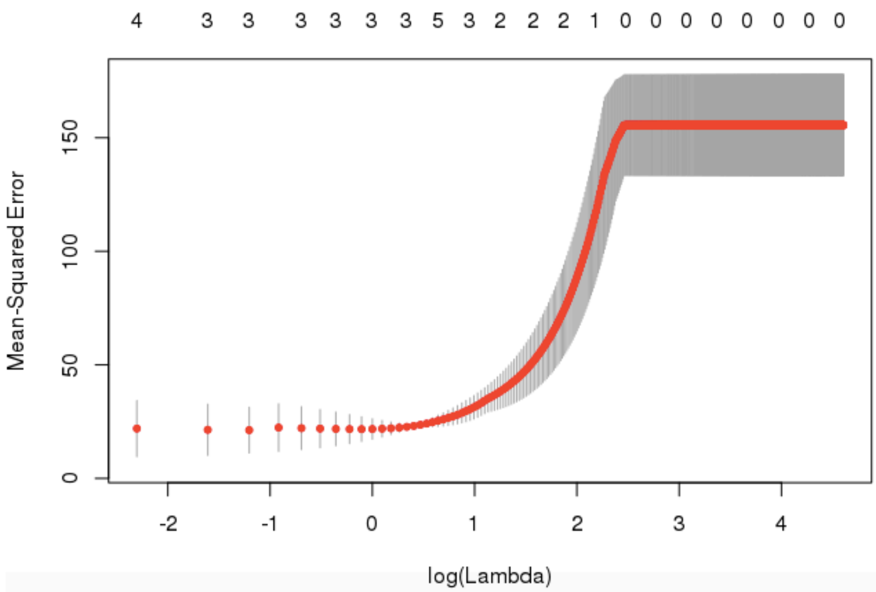
```
> #ridge regression model -- use this one or use the standardized one where we subtract mean?
> x1 <- c(1300,1300,1300,1300,1300,1300,1200,1200,1200,1200,1200,1200,1100,1100,1100,1100)
> x2 <- c(7.5, 9.0, 11.0, 13.5, 17.0, 23.0, 5.3, 7.5, 11.0, 13.5, 17.0, 23.0, 5.3, 7.5, 11.0, 17.0)
> x3 <- c(0.0120,0.0120,0.0115, 0.0130, 0.0135, 0.0120,0.0400, 0.0380, 0.0320, 0.0260, 0.0340, 0.041
0, 0.0840, 0.0980, 0.0920, 0.0860)
> y <- c(49.0, 50.2, 50.5, 48.5, 47.5, 44.5, 28.0, 31.5, 34.5, 35, 38, 38.5, 15.0, 17.0, 20.5, 29.5)
> acetylene <- data.frame(x1,x2,x3,y)
>
> acetyleneRidge <- data.frame(y, x1, x2, x3, x1*x2, x1*x2, x1*x3, x1*x1, x2*x2, x3*x3)
> x=model.matrix(y~., acetyleneRidge)
> ridgefit = glmnet(x, y, alpha = 0, lambda = seq(0,100,.1))
> ridgecv = cv.glmnet(x,y, alpha = 0, nfold = 3, lambda = seq(0,100,.1))
> plot(ridgecv)
>
> #get lambda
> lambdaridge = ridgecv$lambda.min
> print(lambdaridge)
[1] 0
>
```



5.5

Lasso regression does have 1 coefficient set to 0 that should be dropped from the model

```
> lassoRegression <- data.frame(y, x1, x2, x3, x1*x2, x1*x2, x1*x3, x1*x1, x2*x2, x3*x3)
>
> x=model.matrix(y~., lassoRegression)
>
> lassoFit = glmnet(x, y, alpha = 1, lambda = seq(0,100,.1))
> lassoCV = cv.glmnet(x,y, alpha = 1, nfold = 3, lambda = seq(0,100,.1))
> plot(lassoCV)
> #calculate lambda
> lambdalasso <- lassoCV$lambda.min
> print(lambdalasso)
[1] 0
> small.lambda.index <- which(lassoCV$lambda == lassoCV$lambda.min)
> #Plot
> plot(lassoFit, xvar = "lambda", label = TRUE, main="coeffs of Lasso regression", type="l", xlab=ex
pression("log_lambda"), ylab="Coeff")
> abline(h=0); abline(v=log(lassoCV$lambda.min))
> grid()
> print(small.lambda.betas)
  (Intercept)   (Intercept)            x1            x2            x3       x1...x2
 2.505102e+01  0.000000e+00 -2.030683e-01  9.932362e+00  4.667868e+01 -7.377790e-03
     x1...x2.1       x1...x3       x1...x1       x2...x2       x3...x3
 1.197737e-15 -9.484522e-02  1.703632e-04 -2.075063e-02  2.251869e+02
```



4 3 3 3 3 3 3 5 3 2 2 2 1 0 0 0 0 0 0 0 0
Mean-Squared Error
0
50
100
150
-2
-1
0
1
2
3
4
log(Lambda)