STAT 6115 Final Project

Sha Yu

1. Write the dataset to a file in excel form.

```
### STAT 6115 Final Project ### -------------------------------------------
library(xlsx)
library(glmnet)
library(gam)

## write dataset to a file in excel format ## ----------------------------
data = read.table("data for project.txt", header = T)
attach(data)
write.xlsx(data, "C:/Users/yusha/Google Drive/courses/data.xlsx")
```

2. Find sample correlation between the response and each of the covariates.

```
## find sample correlation ## -------------------------------------------
corr = cor(data)[1,-1]
corr
```

```
> corr
         V2          V3          V4          V5          V6 
-0.35631353 -0.04243960 -0.04011802  0.43695150 -0.46365281 
```

- We can tell from the output that there is little correlation between V1 and V3, V1 and V4.

3. Propose an initial model to fit the dataset and check its appropriateness using regression diagnostics.

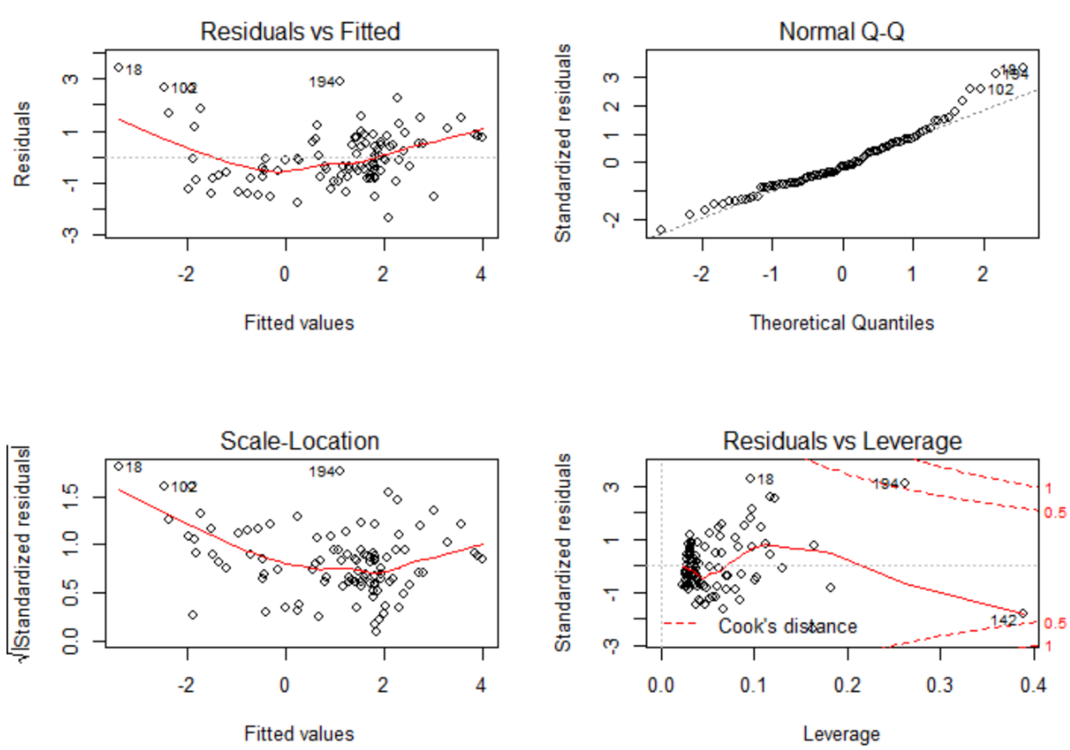
- First, I separated the data into training dataset and test dataset. And then I fit a linear model to the training dataset. From the summary(fit), p value of V3 and V4 are greater 0.05, which indicated that these two covariates are not significant.
- Next, I calculated AIC and the test error on test dataset. Also checked the performance by doing regression diagnostics. The AIC of fit1 is 309.4122 and the test error is 1.234559.
- The plot of residuals versus fitted values indicates the presence of nonlinearity in the data. The plot of standardized residuals versus leverage indicates the presence of a few outliers and a few high leverage points.

```
## separate data to training set and test set ## ------------------------
set.seed(100)
train = sample(length(V1), length(V1)/2)
test = - train
train.data = data[train,]
test.data = data[test,]

## fit a linear model to data ## ---------------------------------------
fit1 = lm(V1~., data = train.data)
summary(fit1)
```

```
> summary(fit1)

Call:
lm(formula = V1 ~ ., data = train.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.3656 -0.7552 -0.1448  0.6200  3.4111 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   0.2665     0.3791   0.703    0.484    
V2            8.5492     1.6996   5.030 2.35e-06 ***
V3            0.3526     0.2271   1.552    0.124    
V4           -0.6027     0.3835  -1.571    0.119    
V5            1.3604     0.1741   7.813 7.87e-12 ***
V6          -12.3984     1.5779  -7.857 6.35e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.093 on 94 degrees of freedom
Multiple R-squared:  0.6725,	Adjusted R-squared:  0.655 
F-statistic:  38.6 on 5 and 94 DF,  p-value: < 2.2e-16
```

```
# perform regression diagnostics for fit1 # --------------------------------
AIC(fit1)
test.error1 = mean((V1 - predict(fit1, data))[test]^2)
test.error1

par(mfrow = c(2,2))
plot(fit1)
```

```
> AIC(fit1)
[1] 309.4122
> test.error1 = mean((V1 - predict(fit1, data))[test]^2)
> test.error1
[1] 1.234559
```



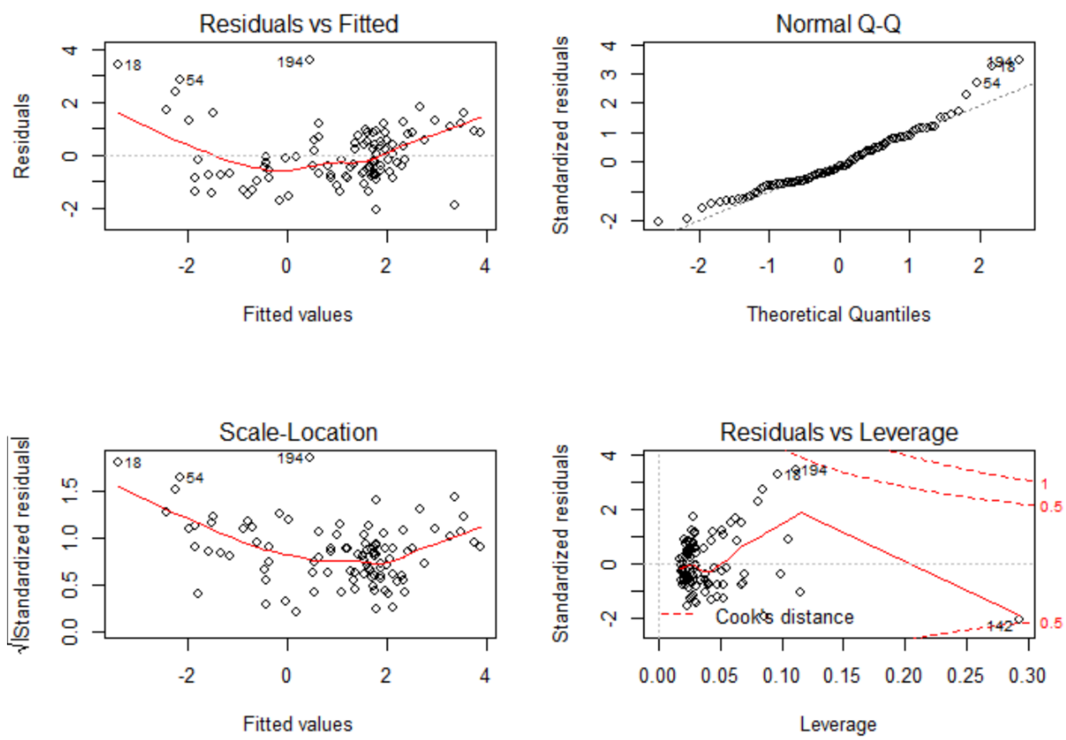
4. Refine your model based on your discovery in step 3 and check if it is appropriate.

- I refit the model by eliminating the two insignificant covariates V3 and V4. Now all variables are significant according to the summary.
- The AIC is 308.1417, which has decreased. The new test error is 1.191016 and also is smaller compared to previous test error 1.234559. Thus by eliminating the insignificant terms, the model has been refined.
- But by checking the residual plot, there is still nonlinearity. Also there are still some outliers.

```
# refit the model by elminating the insignificant terms # ------------------
fit2 = lm(V1~V2+V5+V6, data = train.data)
summary(fit2)

AIC(fit2)
test.error2 = mean((V1 - predict(fit2, data))[test]^2)
test.error2
plot(fit2)
```

```
> summary(fit2)

Call:
lm(formula = V1 ~ V2 + V5 + V6, data = train.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.0697 -0.7506 -0.1952  0.6907  3.5668 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   0.3272     0.3782   0.865    0.389    
V2            8.3547     1.6995   4.916 3.64e-06 ***
V5            1.3423     0.1740   7.715 1.14e-11 ***
V6          -12.2334     1.5790  -7.747 9.69e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.097 on 96 degrees of freedom
Multiple R-squared:  0.6634,	Adjusted R-squared:  0.6529 
F-statistic: 63.07 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
> AIC(fit2)
[1] 308.1417
> test.error2 = mean((V1 - predict(fit2, data))[test]^2)
> test.error2
[1] 1.191016
```



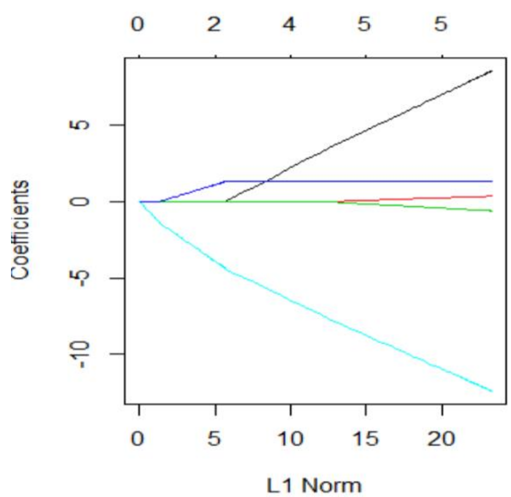
5. Can you improve the model you fit in step 4? Why?

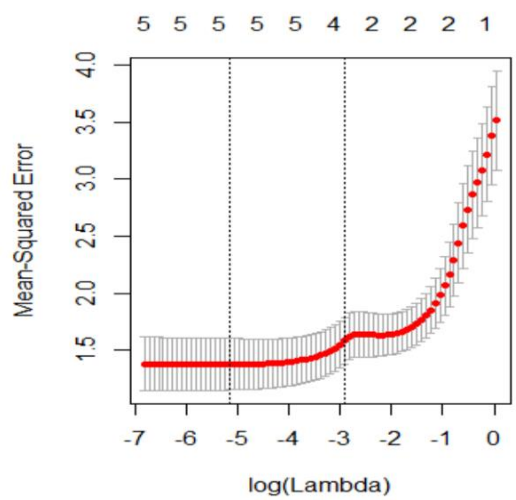
Part(1) :refine in linear models

- Although step 4 has refined the model. The model is still not very appropriate. So I calculated the correlation between all covariates. I found that the correlation between V3 and V4 is 0.88357693, the correlation between V2 and V6 is 0.97089447. So there is collinearity among covariates.
- I performed lasso to do variable selection, hoping to solve the collinearity problem. I used 10-fold cross validation to choose lambda in lasso regression.
  Lasso method chose covariates V5 and V6. It eliminates the insignificant variable and also only pick on among the two highly correlated covariates.
- I refit the linear model just using the variable selected by lasso. According to summary(fit), all covariates are significant. However, the new AIC is 328.594, which has increased. Also the test error is 1.500241, which is larger than the test error of the model with V2, V5 and V6.
- Check the plots, there are still nonlinearity and outliers.
- Thus by applying lasso method and eliminating V2, the model could not be refined anymore.
  I summarized the results of using different variables to fit linear regression as below.

```r
# detect collinearity # --------------------------------------------
cor(data)

## apply lasso to do varibale selection and estimate test error ## ------
x = model.matrix(V1~., data)[,-1]
y = V1

grid = 10^seq(-10, 10, length = 1000)
lasso.mod = glmnet(x[train,], y[train], alpha= 1, lambda = grid)
plot(lasso.mod)

cv.out = cv.glmnet(x[train,], y[train], alpha =1)
plot(cv.out)
bestlam = cv.out$lambda.1se

out = glmnet(x,y, alpha =1 , lambda = grid)
lasso.coef = predict(out, type = "coefficient", s= bestlam, newx = x[test,])
lasso.coef
```

```
> cor(data)
            V1          V2          V3          V4          V5          V6
V1  1.00000000 -0.35631353 -0.04243960 -0.04011802  0.43695150 -0.46365281
V2 -0.35631353  1.00000000 -0.05026962 -0.06689517  0.36128863  0.97089447
V3 -0.04243960 -0.05026962  1.00000000  0.88357693 -0.05573148 -0.05789913
V4 -0.04011802 -0.06689517  0.88357693  1.00000000 -0.06154981 -0.07853427
V5  0.43695150  0.36128863 -0.05573148 -0.06154981  1.00000000  0.33051479
V6 -0.46365281  0.97089447 -0.05789913 -0.07853427  0.33051479  1.00000000
```

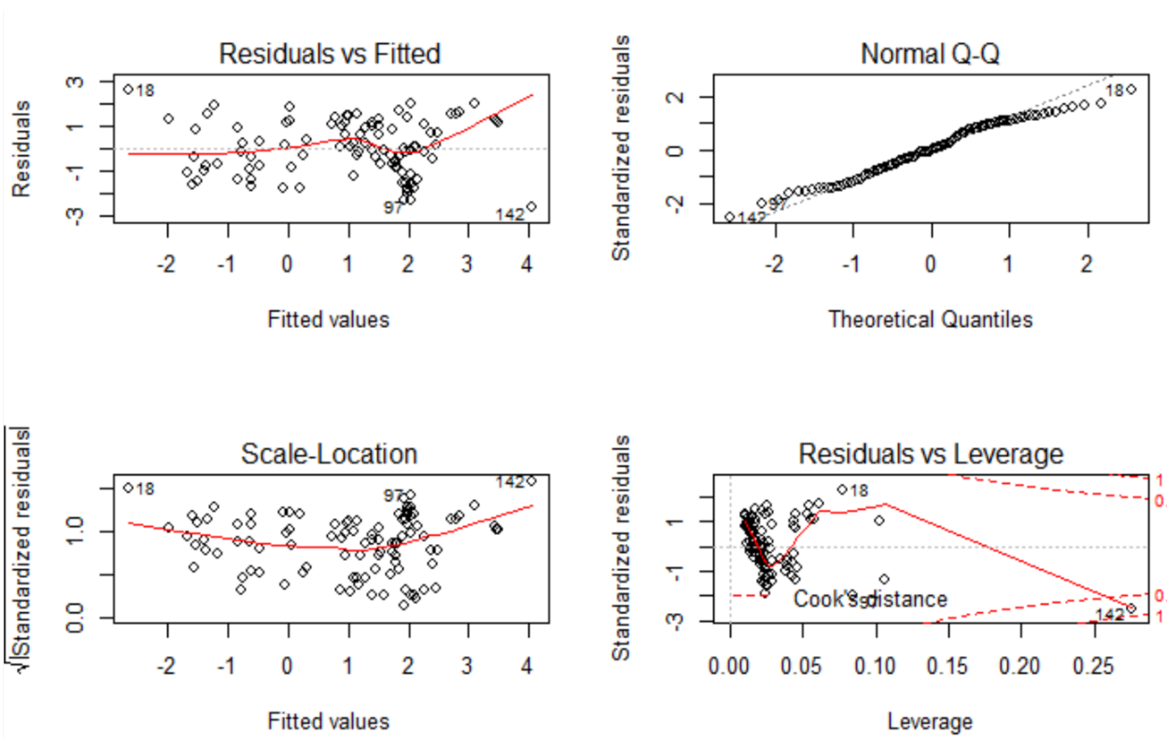





```
> lasso.coef
6 x 1 sparse Matrix of class "dgCMatrix"
                    1
(Intercept)  1.639964
V2           .
V3           .
V4           .
V5           1.476278
V6          -3.593552
```

```r
# refit the model using V5, V6 # -------------------------
fit3 = lm(V1~V5+V6, data = train.data)
summary(fit3)

# perform regression diagonistic for fit3 # -----------
AIC(fit3)
test.error3 = mean((V1 - predict(fit3, data))[test]^2)
test.error3
par(mfrow = c(2,2))
plot(fit3)
```

```
> summary(fit3)

Call:
lm(formula = V1 ~ V5 + V6, data = train.data)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.60466 -0.89124 -0.05393  1.04354  2.65299 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.9745     0.1952  10.118  < 2e-16 ***
V5            1.4596     0.1918   7.609  1.8e-11 ***
V6           -4.7106     0.4332 -10.874  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.221 on 97 degrees of freedom
Multiple R-squared:  0.5787,	Adjusted R-squared:   0.57 
F-statistic: 66.61 on 2 and 97 DF,  p-value: < 2.2e-16

> AIC(fit3)
[1] 328.594
> test.error3 = mean((V1 - predict(fit3, data))[test]^2)
> test.error3
[1] 1.500241
```

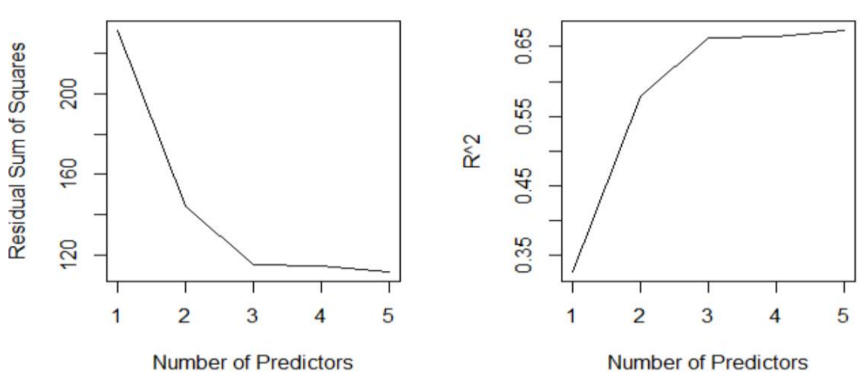


Part(2): extend to nonlinear models.

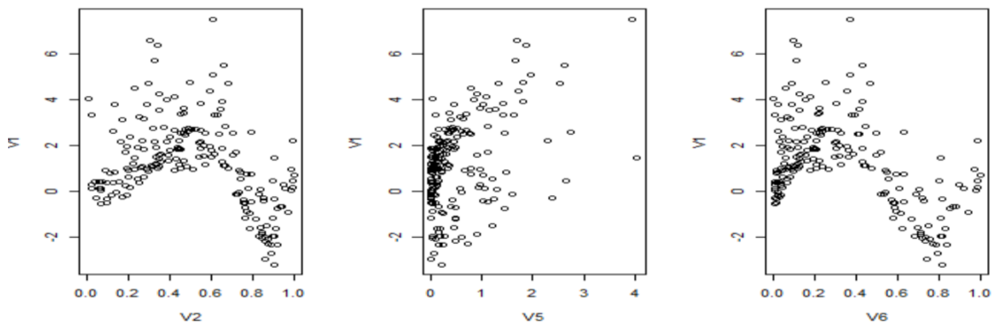
- Recall there exits nonlinear patter between response and covariates. Thus the next I tried some nonlinear models.
- Recall that V3 and V4 have little correlation with V1 and there is also collinearity among variables, so I first performed best subset selection on the training set in order to identify a satisfactory model that used just a subset of the predictors.
- According to the best subset selection output, the best 2-variable model is the model with covariates V5 and V6. The best 3-variable model is the model with covariates V2, V5 and V6.

```
# perform best subeset selection # ---------------------------------
library(leaps)
regfit.full = regsubsets(V1~., train.data)
reg.summary=summary(regfit.full)
reg.summary

par(mfrow=c(1,2))
plot(reg.summary$rss, xlab="Number of Predictors", ylab="Residual Sum of Squares", type="l")
plot(reg.summary$rsq, xlab="Number of Predictors", ylab="R^2", type="l")
coef(regfit.full, 2)
coef(regfit.full, 3)
```

```
> reg.summary
Subset selection object
Call: regsubsets.formula(V1 ~ ., train.data)
5 Variables  (and intercept)
   Forced in Forced out
V2     FALSE      FALSE
V3     FALSE      FALSE
V4     FALSE      FALSE
V5     FALSE      FALSE
V6     FALSE      FALSE
1 subsets of each size up to 5
Selection Algorithm: exhaustive
         V2  V3  V4  V5  V6
1  ( 1 ) " " " " " " " " "*"
2  ( 1 ) " " " " " " "*" "*"
3  ( 1 ) "*" " " " " "*" "*"
4  ( 1 ) "*" " " "*" "*" "*"
5  ( 1 ) "*" "*" "*" "*" "*"
>
> par(mfrow=c(1,2))
> plot(reg.summary$rss, xlab="Number of Predictors", ylab="Residual Sum of Squares", type="l")
> plot(reg.summary$rsq, xlab="Number of Predictors", ylab="R^2", type="l")
> coef(regfit.full, 2)
(Intercept)          V5          V6
   1.974459    1.459604   -4.710604
> coef(regfit.full, 3)
(Intercept)          V2          V5          V6
  0.3272144   8.3546861   1.3423166 -12.2334406
```

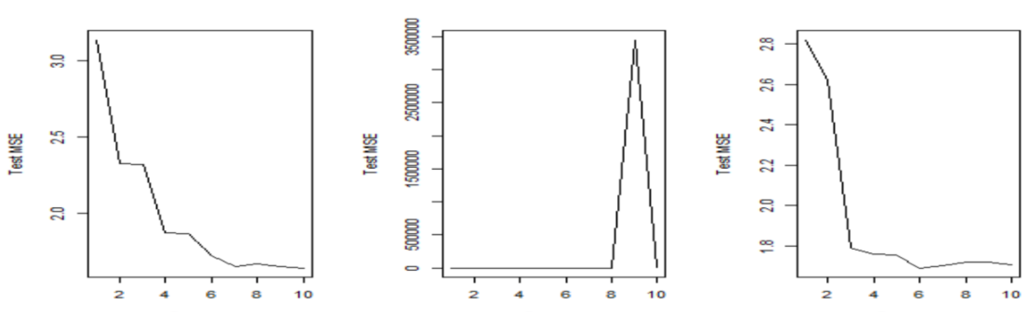


- To explore more information between V1 and V2, V5 and V6, I first did scatter plots. We can tell from the plots, there is nonlinear patter between V1 and V2, also between V1 and V6.



- First, I tried to fit univariate polynomial model between response and each covariate V2, V5, V6. I performed the 10-fold cross validation to choose the best degree for each covariate. Then I fit the generalized additive model to the training data by setting each basis function as the best polynomial. I fit a model with best 2-variable along with the model with best 3-variable.
- From the output, the best degree for V2 is 10, best degree for V5 is 1 and best degree for V6 is 6.

```
> # fit polynomial to V2 # --------------------------------------------
> library(boot)
> cv.error= rep(NA, 10)
> for (i in 1:10) {
+   fit = glm(V1 ~ poly(V2, i), data = data)
+   cv.error[i] = cv.glm(data, fit, K = 10)$delta[1]
+ }
> plot(1:10, cv.error, xlab = "Degree", ylab = "Test MSE", type = "l")
> which.min(cv.error)
[1] 10
>
> # fit polynomial to V5 # --------------------------------------------
> cv.error= rep(NA, 10)
> for (i in 1:10) {
+   fit = glm(V1 ~ poly(V5, i), data = data)
+   cv.error[i] = cv.glm(data, fit, K = 10)$delta[1]
+ }
> plot(1:10, cv.error, xlab = "Degree", ylab = "Test MSE", type = "l")
> which.min(cv.error)
[1] 1
>
>
> # fit polynomial to V6 # --------------------------------------------
> cv.error = rep(NA, 10)
> for (i in 1:10) {
+   fit = glm(V1 ~ poly(V6, i), data = data)
+   cv.error[i] = cv.glm(data, fit, K = 10)$delta[1]
+ }
> plot(1:10, cv.error, xlab = "Degree", ylab = "Test MSE", type = "l")
> which.min(cv.error)
[1] 6
```



- For best 3-variable model, fit a GAM to V2, V5 and V6.
  The test error is 0.7564776, which is much smaller compared to any of the test error by fitting linear model.
- Thus this new model significantly improves the performance.

```
> # fit GAM to V2, V5, V6 # --------------------------------------------
> fit4 = gam(V1 ~poly(V2, df = 10) + poly(V5, df = 1) + poly(V6, df = 6), data=train.data)
> summary(fit4)

Call: gam(formula = V1 ~ poly(V2, df = 10) + poly(V5, df = 1) + poly(V6, 
    df = 6), data = train.data)
Deviance Residuals:
     Min       1Q   Median       3Q      Max 
-1.30595 -0.48567 -0.02668  0.42184  1.87294 

(Dispersion Parameter for gaussian family taken to be 0.3847)

    Null Deviance: 342.9341 on 99 degrees of freedom
Residual Deviance: 33.4708 on 87 degrees of freedom
AIC: 202.3382 

Number of Local Scoring Iterations: 2 

Anova for Parametric Effects
                  Df  Sum Sq Mean Sq  F value    Pr(>F)    
poly(V2, df = 10) 10 242.386  24.239  63.0029 < 2.2e-16 ***
poly(V5, df = 1)   1  64.246  64.246 166.9920 < 2.2e-16 ***
poly(V6, df = 6)   1   2.832   2.832   7.3602  0.008037 ** 
Residuals         87  33.471   0.385                       
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> 
> preds.4 = predict(fit4, test.data)
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(type ==  :
  prediction from a rank-deficient fit may be misleading
> test.error4 = mean((test.data$V1 - preds.4)^2)
> test.error4
[1] 0.7564776
```

- For best 2-variable model, fit a GAM to V5 and V6.
  The test error is 0.746642, which is much smaller compared to any of the test error by fitting linear model. And it is also smaller than the test error of GAM with variable V2, V5 and V6.
- So far, GAM models fit the data much better than multiple linear models. And it is the best model so far.

```
> # fit GAM to V5, V6 # --------------------------------------------------
> fit5 = gam(V1 ~ poly(V5, df = 1) + poly(V6, df = 6), data=train.data)
> summary(fit5)

Call: gam(formula = V1 ~ poly(V5, df = 1) + poly(V6, df = 6), data = train.data)
Deviance Residuals:
     Min       1Q   Median       3Q      Max
-1.25254 -0.48158 -0.09244  0.36664  3.30195

(Dispersion Parameter for gaussian family taken to be 0.5137)

    Null Deviance: 342.9341 on 99 degrees of freedom
Residual Deviance: 47.2635 on 92 degrees of freedom
AIC: 226.8445

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                 Df  Sum Sq Mean Sq F value    Pr(>F)
poly(V5, df = 1)  1  22.314  22.314  43.434 2.699e-09 ***
poly(V6, df = 6)  6 273.357  45.560  88.683 < 2.2e-16 ***
Residuals        92  47.263   0.514
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

>
> preds.5 = predict(fit5, test.data)
> test.error5 = mean((test.data$V1 - preds.5)^2)
> test.error5
[1] 0.746642
```

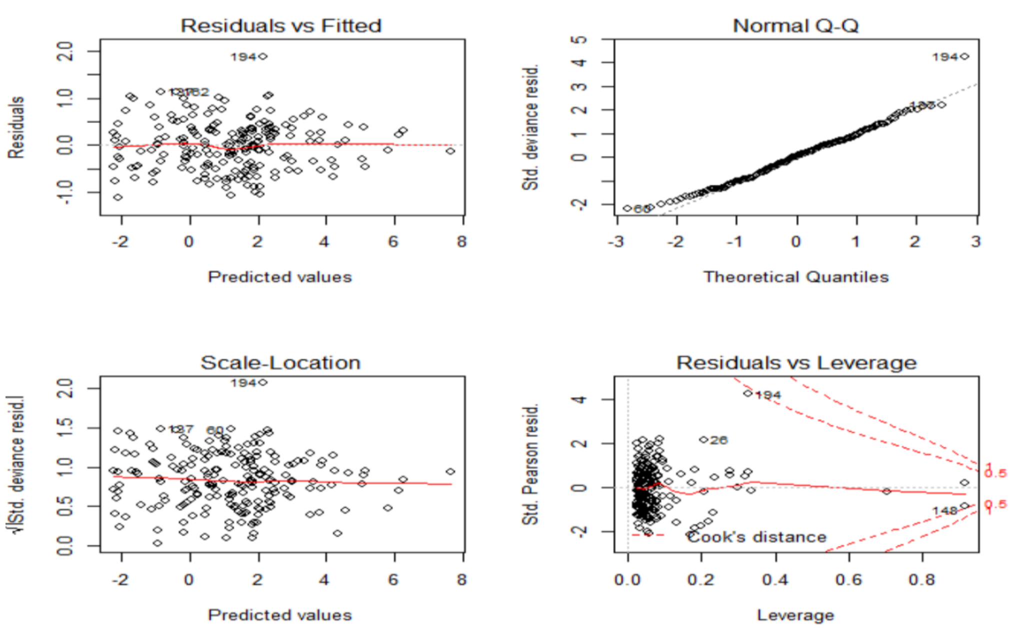
- I want to see whether I could refine the model more. Considering that GAM models do not consider the interaction between covariates. So in order to include the possible interaction between covariates, I fit polynomial to covariates by using 10-fold cross validation to choose the best degree.

- For best 3- variable model (V2, V5 and V6), the best degree is 3. According to summary, some interactions are significant. Test error is 0.3245652, both significantly reduced further. But some coefficients are not defined due to singularities. Consider together with the high correlation between V2 and V6, it is better to only include one of them in the model.
  The diagnostic plots show that the model is adequate. There is almost no pattern in residual plot and also the QQ plot looks good.

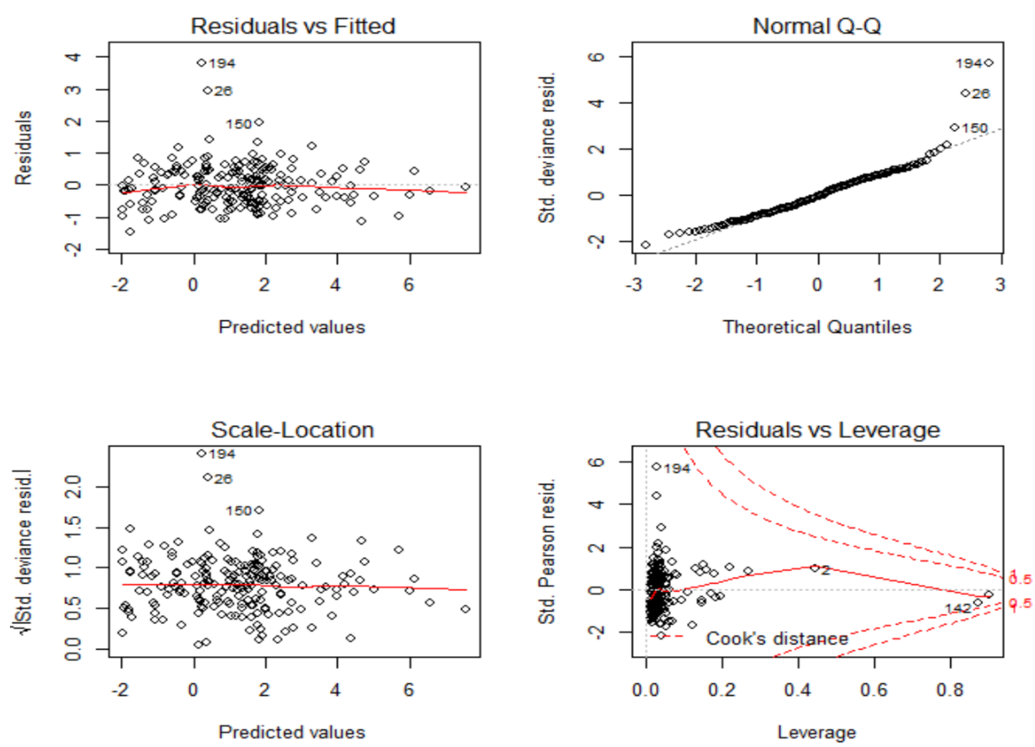
- For best 2-variable model (V5 and V6), there are also significant interaction terms. Also, the test error is 0.4686955.
  The diagnostic plots show that the model is adequate. There is almost no pattern in residual plot and also the QQ plot looks good except some outliers.

```r
# fit poly to V2, V5, V6 # ------------------------------
cv.error.6 = rep(NA, 10)
for (i in 1:10) {
  fit = glm(V1 ~ poly(V2, V5, V6, degree = i), data =data)
  cv.error.6[i] = cv.glm(data, fit, K = 10)$delta[1]
}
plot(1:10, cv.error.6, xlab = "Degree", ylab = "Test MSE", type = "l")
d.min.1 = which.min(cv.error.6)
test.error.6 = min(cv.error.6)
test.error.6

fit6 = glm(V1 ~ poly(V2, V5, V6, degree = d.min.1), data = data)
summary(fit6)

par(mfrow=c(2,2))
plot(fit6)

# fit poly to V5, V6 # ------------------------------------
cv.error.7 = rep(NA, 10)
for (i in 1:10) {
  fit = glm(V1 ~ poly(V5, V6, degree = i), data = data)
  cv.error.7[i] = cv.glm(data, fit, K = 10)$delta[1]
}
plot(1:10, cv.error.7, xlab = "Degree", ylab = "Test MSE", type = "l")
d.min.2 = which.min(cv.error.7)
test.error.7 = min(cv.error.7)
test.error.7

fit7 = glm(V1~poly(V5, V6, degree = d.min.2), data = data)
summary(fit7)

par(mfrow=c(2,2))
plot(fit7)
```

```
> test.error.6
[1] 0.3659772
>
> fit6 = glm(V1 ~ poly(V2, V5, V6, degree = d.min.1), data = data)
> summary(fit6)

Call:
glm(formula = V1 ~ poly(V2, V5, V6, degree = d.min.1), data = data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.11267  -0.40728   0.01345   0.32667   1.88565

Coefficients: (4 not defined because of singularities)
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                             -9.623e+00  4.011e+00  -2.399 0.017443 *
poly(V2, V5, V6, degree = d.min.1)1.0.0 -1.342e+02  3.332e+01  -4.028 8.22e-05 ***
poly(V2, V5, V6, degree = d.min.1)2.0.0  7.680e+01  2.504e+01   3.067 0.002486 **
poly(V2, V5, V6, degree = d.min.1)3.0.0 -1.399e+02  4.010e+01  -3.489 0.000607 ***
poly(V2, V5, V6, degree = d.min.1)0.1.0  2.080e+02  5.936e+01   3.504 0.000575 ***
poly(V2, V5, V6, degree = d.min.1)1.1.0  4.777e+02  2.236e+02   2.136 0.033965 *
poly(V2, V5, V6, degree = d.min.1)2.1.0  1.797e+03  5.316e+02   3.380 0.000886 ***
poly(V2, V5, V6, degree = d.min.1)0.2.0 -2.769e-01  2.683e+00  -0.103 0.917910
poly(V2, V5, V6, degree = d.min.1)1.2.0 -1.878e+01  1.456e+02  -0.129 0.897521
poly(V2, V5, V6, degree = d.min.1)0.3.0 -2.404e-01  6.060e-01  -0.397 0.692117
poly(V2, V5, V6, degree = d.min.1)0.0.1         NA         NA      NA       NA
poly(V2, V5, V6, degree = d.min.1)1.0.1         NA         NA      NA       NA
poly(V2, V5, V6, degree = d.min.1)2.0.1  5.609e+03  1.904e+03   2.946 0.003640 **
poly(V2, V5, V6, degree = d.min.1)0.1.1         NA         NA      NA       NA
poly(V2, V5, V6, degree = d.min.1)1.1.1 -3.607e+04  1.155e+04  -3.123 0.002080 **
poly(V2, V5, V6, degree = d.min.1)0.2.1  1.957e+01  1.070e+02   0.183 0.855045
poly(V2, V5, V6, degree = d.min.1)0.0.2         NA         NA      NA       NA
poly(V2, V5, V6, degree = d.min.1)1.0.2 -4.709e+03  1.675e+03  -2.811 0.005468 **
poly(V2, V5, V6, degree = d.min.1)0.1.2  8.180e+02  2.924e+02   2.798 0.005693 **
poly(V2, V5, V6, degree = d.min.1)0.0.3  1.201e+02  3.815e+01   3.148 0.001921 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.2913669)

    Null deviance: 704.392  on 199  degrees of freedom
Residual deviance:  53.612  on 184  degrees of freedom
AIC: 338.26

Number of Fisher Scoring iterations: 2
```



```
> test.error.7
[1] 0.4686955
>
> fit7 = glm(V1~poly(V5, V6, degree = d.min.2), data = data)
> summary(fit7)

Call:
glm(formula = V1 ~ poly(V5, V6, degree = d.min.2), data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4443  -0.4529  -0.0437   0.4153   3.8047

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                          1.39466    0.05748  24.262  < 2e-16 ***
poly(V5, V6, degree = d.min.2)1.0   20.81718    1.26252  16.489  < 2e-16 ***
poly(V5, V6, degree = d.min.2)2.0   -1.30966    1.04655  -1.251  0.21232
poly(V5, V6, degree = d.min.2)3.0   -0.15861    0.71139  -0.223  0.82380
poly(V5, V6, degree = d.min.2)0.1  -19.08055    0.81436 -23.430  < 2e-16 ***
poly(V5, V6, degree = d.min.2)1.1 -104.24259   16.33368  -6.382  1.3e-09 ***
poly(V5, V6, degree = d.min.2)2.1    4.87665   13.07450   0.373  0.70957
poly(V5, V6, degree = d.min.2)0.2   -2.03542    0.73911  -2.754  0.00646 **
poly(V5, V6, degree = d.min.2)1.2   18.70482   13.34901   1.401  0.16278
poly(V5, V6, degree = d.min.2)0.3   10.33903    0.74946  13.795  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4547515)

    Null deviance: 704.392  on 199  degrees of freedom
Residual deviance:  86.403  on 190  degrees of freedom
AIC: 421.72

Number of Fisher Scoring iterations: 2
```



6. Write down the final model which is the best for fitting the dataset and provide the 95% confidence intervals/band s to your estimators of the parameters/curves.

- Summary of all the models fitted

	model	Test Error
best 3-variable model: V2, V5, V6	Linear	1.191016
	GAM	0.7390142
	Poly	0.3659772
Best 2-variable model: V5, V6	Linear	1.500241
	GAM	0.7466420
	Poly	0..4686955

- Although fit6 has smaller test error than fit7 but it has singularities. So the best model I chose fit6. Check the summary(fit7), the significant terms are V5, V6, V6^2, V6^3 and V5*V6. So I fit the model by eliminating the insignificant terms. From the summary, this time all variables are significant. Also the diagnostics plots show that the fit is adequate although there still exits outliers.

- The final model is:

$$V1 = 0.29064 - 13.15292 * V6 - 2.13691 * V6^2 + 10.89504 * V6^3 + 2.8990 * V5 - 2.45894 * V5 * V6$$

- The 95% confidence interval for parameters are:

```
                             2.5 %      97.5 %
(Intercept)              0.1686865   0.4125922
poly(V6, degree = 3)1 -14.8908291 -11.4150029
poly(V6, degree = 3)2  -3.5873768  -0.6864524
poly(V6, degree = 3)3   9.5500860  12.2399851
V5                      2.5791365   3.2206562
V5:V6                  -3.0168648  -1.9010223
```

```r
# best model # ------------------------------------------
best.model = glm(V1~poly(V6, degree =3)+ V5+V5:V6, data= data)
summary(best.model)

par(mfrow=c(2,2))
plot(best.model)

confint(best.model)
```

```
> summary(best.model)

Call:
glm(formula = V1 ~ poly(V6, degree = 3) + V5 + V5:V6, data = data)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.4998  -0.4738  -0.0537   0.3821   3.7515  

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)             0.29064    0.06222   4.671 5.59e-06 ***
poly(V6, degree = 3)1 -13.15292    0.88671 -14.833  < 2e-16 ***
poly(V6, degree = 3)2  -2.13691    0.74005  -2.888  0.00432 ** 
poly(V6, degree = 3)3  10.89504    0.68621  15.877  < 2e-16 ***
V5                      2.89990    0.16366  17.719  < 2e-16 ***
V5:V6                  -2.45894    0.28466  -8.638 2.09e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.4615098)

    Null deviance: 704.392  on 199  degrees of freedom
Residual deviance:  89.533  on 194  degrees of freedom
AIC: 420.83

Number of Fisher Scoring iterations: 2
```