Predictive Analytics HW4

Jimmy G. Moore

October 19, 2018

Introduction

We will build a generalized additive model (GAM) to predict the outcome of 100 observations with unobserved response variables. Our evaluation criteria will be in minimizing MSE of prediction. The data set contains 1000 observations each with 25 features (x1-x25) and a response variable y. The response variable is observed for the first 900 observations and missing for the last 100 observations.

Analysis

Data Set up

We begin our analysis by loading the data set into our environment and removing subject id from the set.

```
#Read Data
dat = read.csv("C:/Users/James Moore/Documents/GAM_Example/files/PredictiveHW4Dataset.csv")
```

Now we will divide our data into a training and test set.

```
#create training set
obsResponse = dat[which(!is.na(dat$y)),]
#create test set
predResponse = dat[which(is.na(dat$y)),]
```

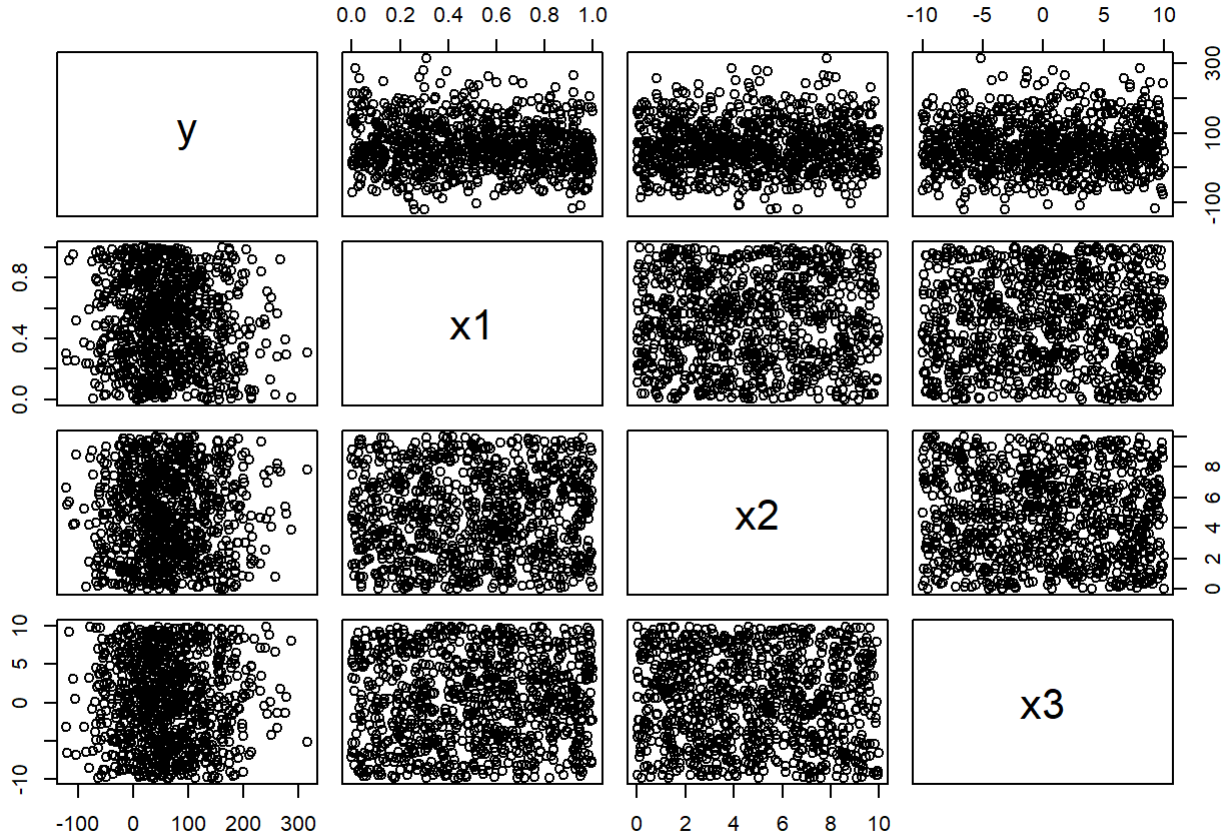
Since we will be doing a lot of cross validation it will also be adventageous to have a function that allows us to quickly shift between training and test folds without having the `fold` variable in our training and test sets. The below function inputs a dataset, and the type of cross validation you wish to conduct (i.e. 5-fold CV). The function returns a training or test set based on which folds the user would like to train and test on.

```
###
# Function: setTrainTest
setTrainTest = function(dat,option="train",foldToTest=1,kfold = 5){
  ## create folds
  for(j in 1:nrow(dat)){
    foldNum = (j-1)%%kfold + 1
    dat$fold[j] = foldNum
  }
  if(option == "train"){
    trainingSet = dat[-which(dat$fold==foldToTest),]
    trainingSet = subset(trainingSet, select = -c(fold))
    return(trainingSet)
  }else{
    testSet = dat[which(dat$fold==foldToTest),]
    testSet = subset(testSet,select = -c(fold))
    return(testSet)
  }
}
```

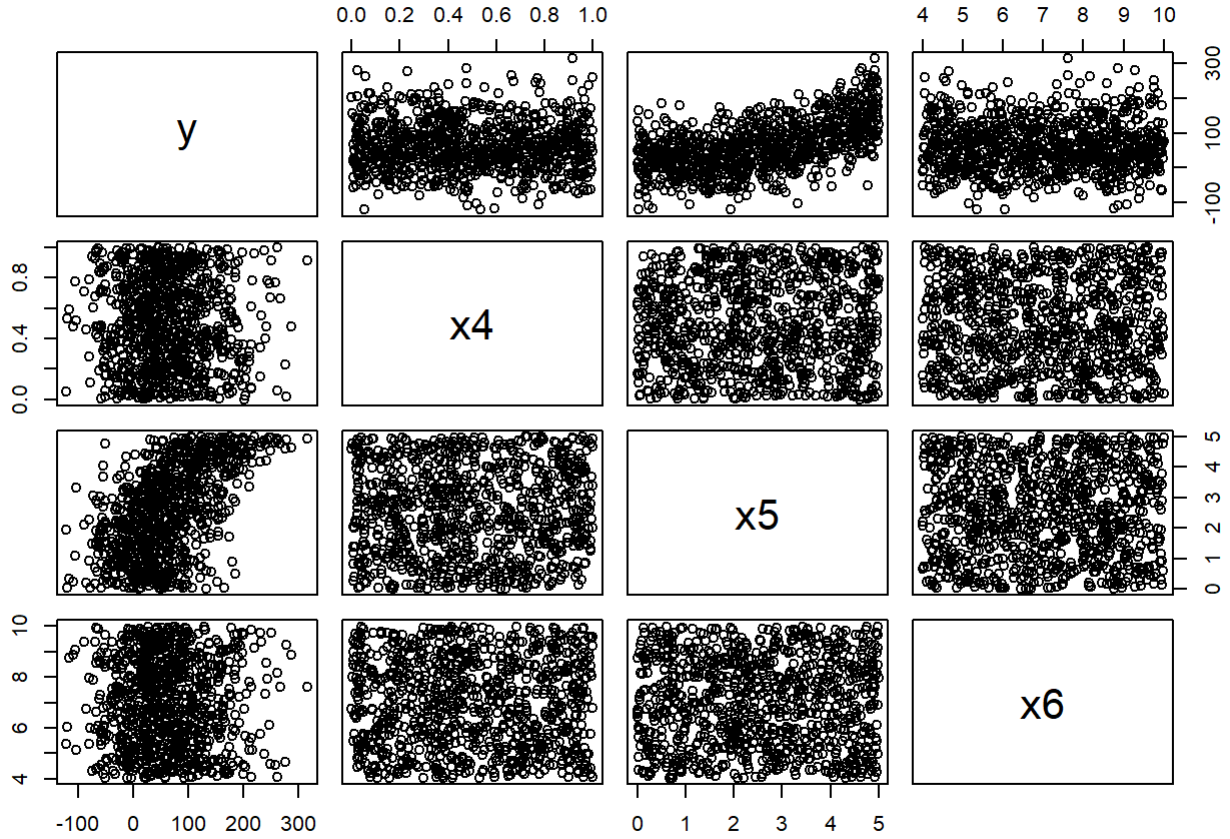
Variable Selection

Now we will look at scatter plot matrices to see which of the predictors have any type of association witht he response variable

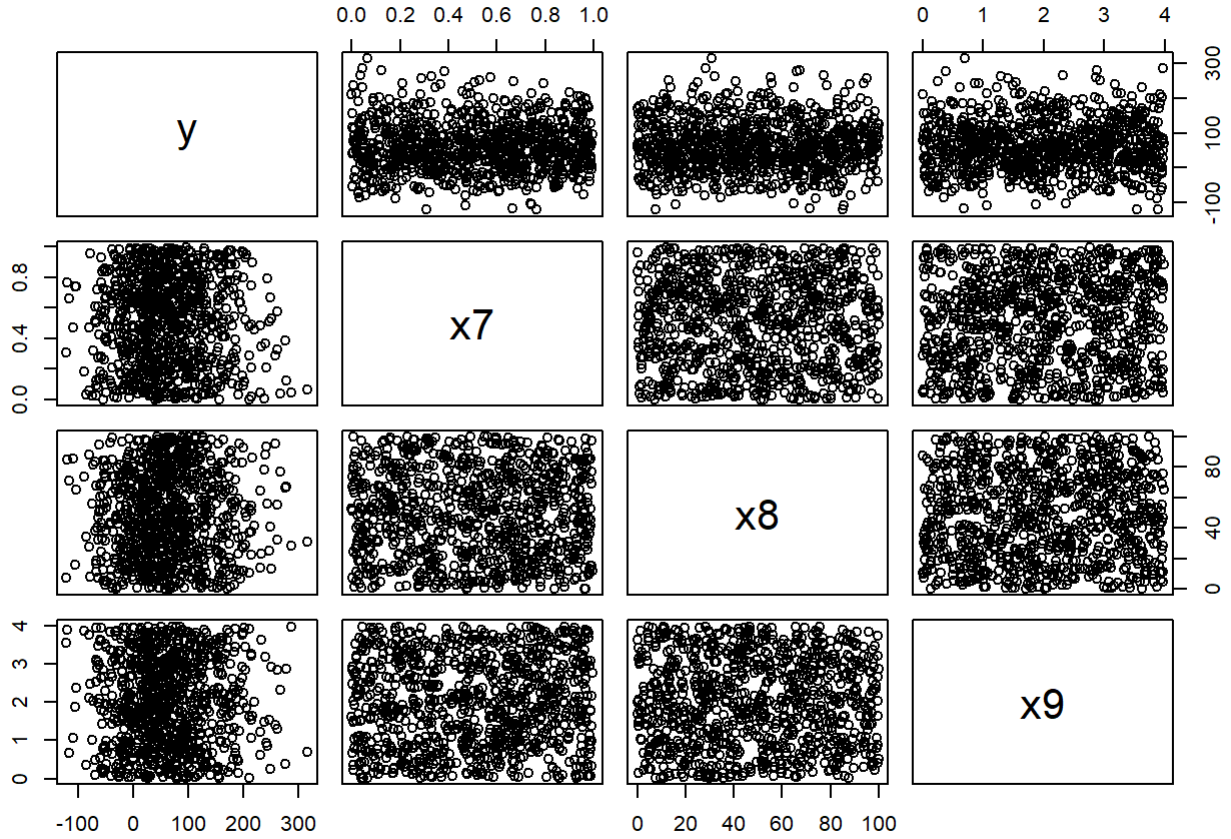
```
#x1-X5
pairs(obsResponse[,c(26,1:3)])
```



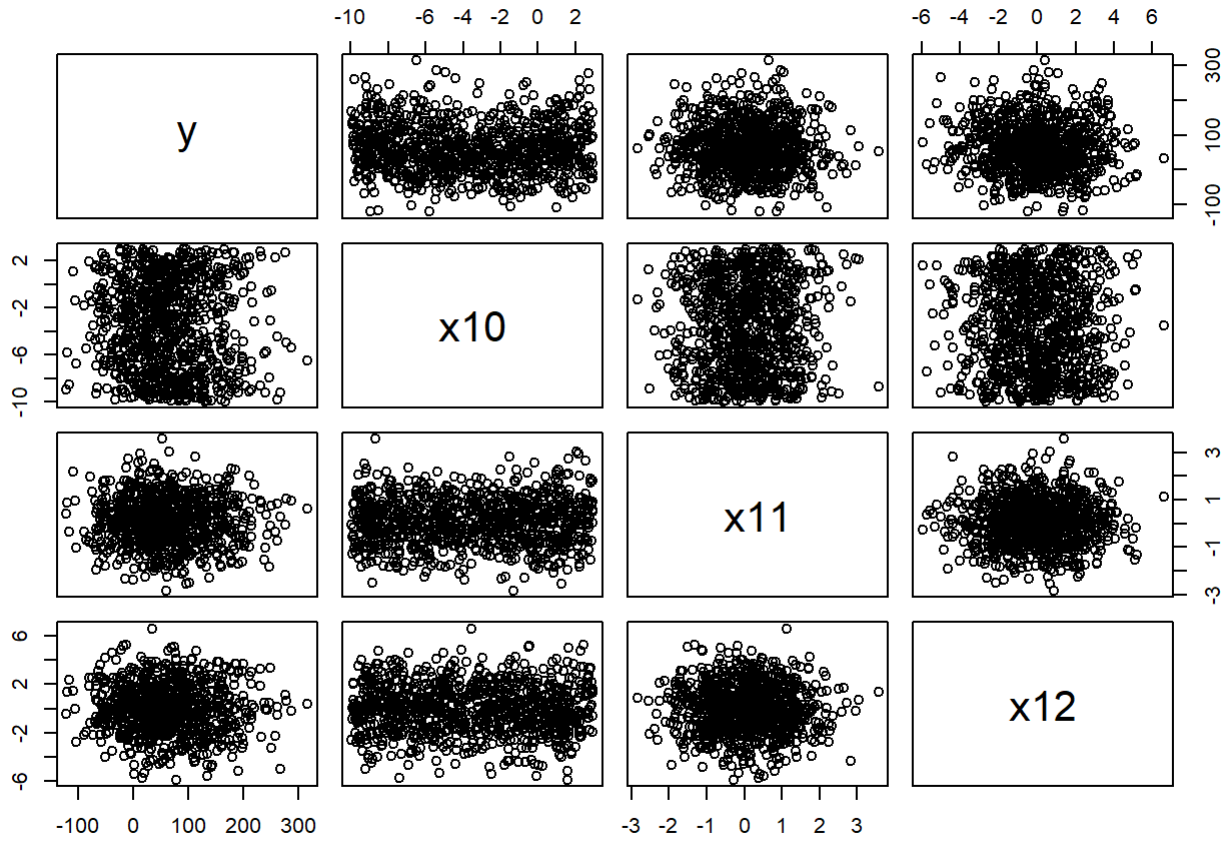

```
pairs(obsResponse[,c(26,4:6)])
```



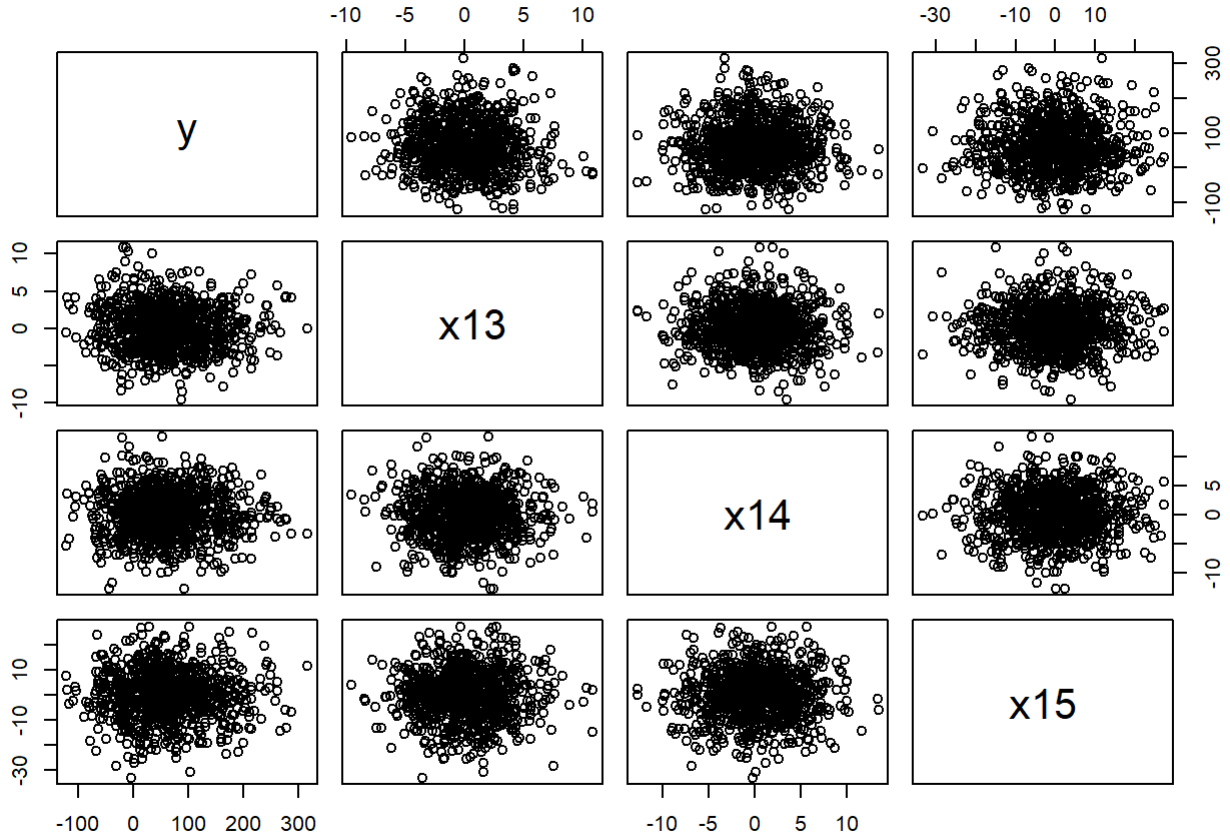

```
pairs(obsResponse[,c(26,7:9)])
```



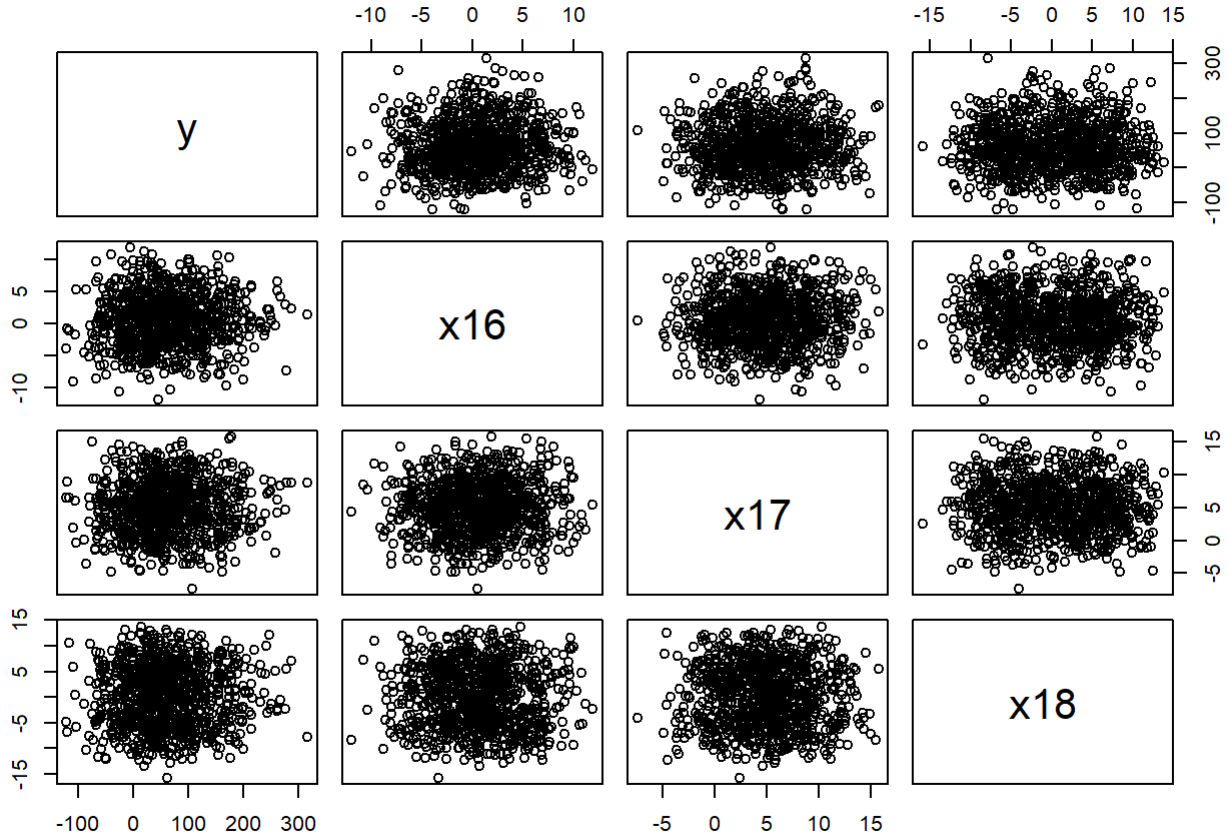

```
pairs(obsResponse[,c(26,10:12)])
```



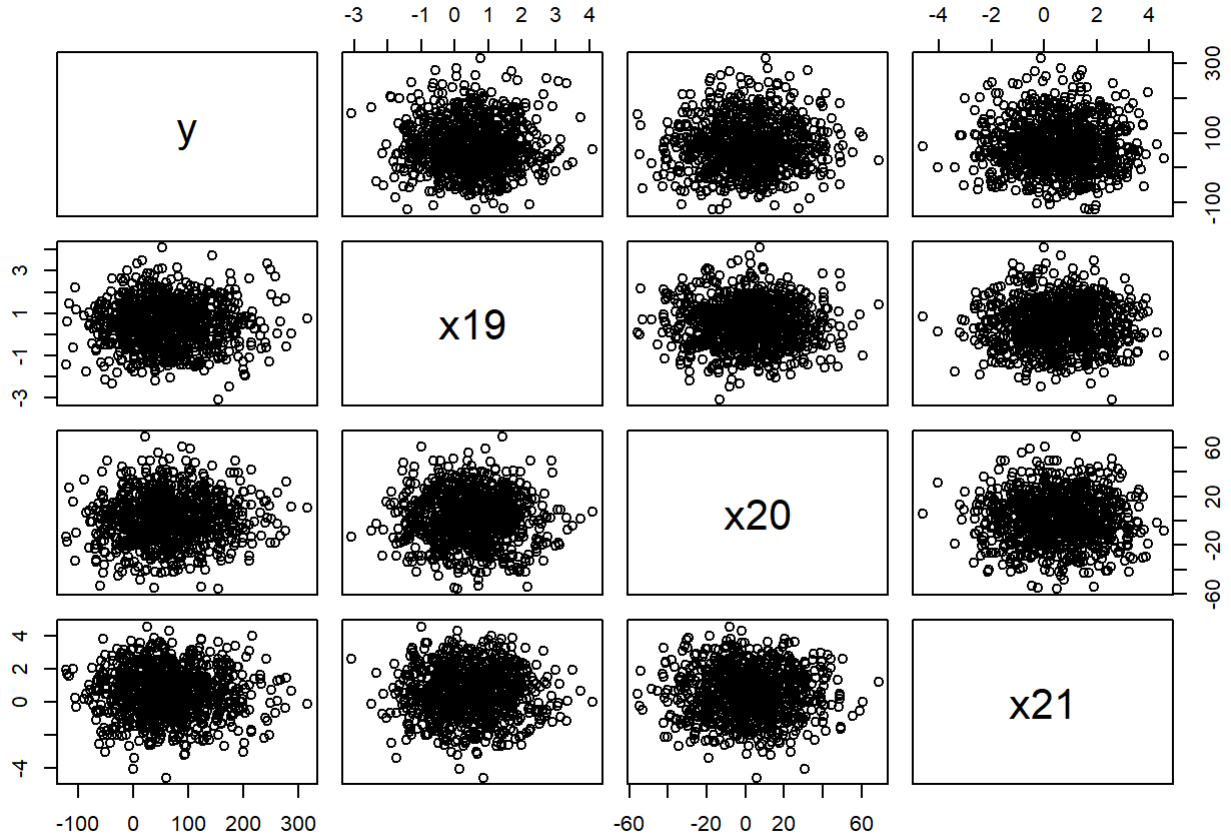

```
pairs(obsResponse[,c(26,13:15)])
```



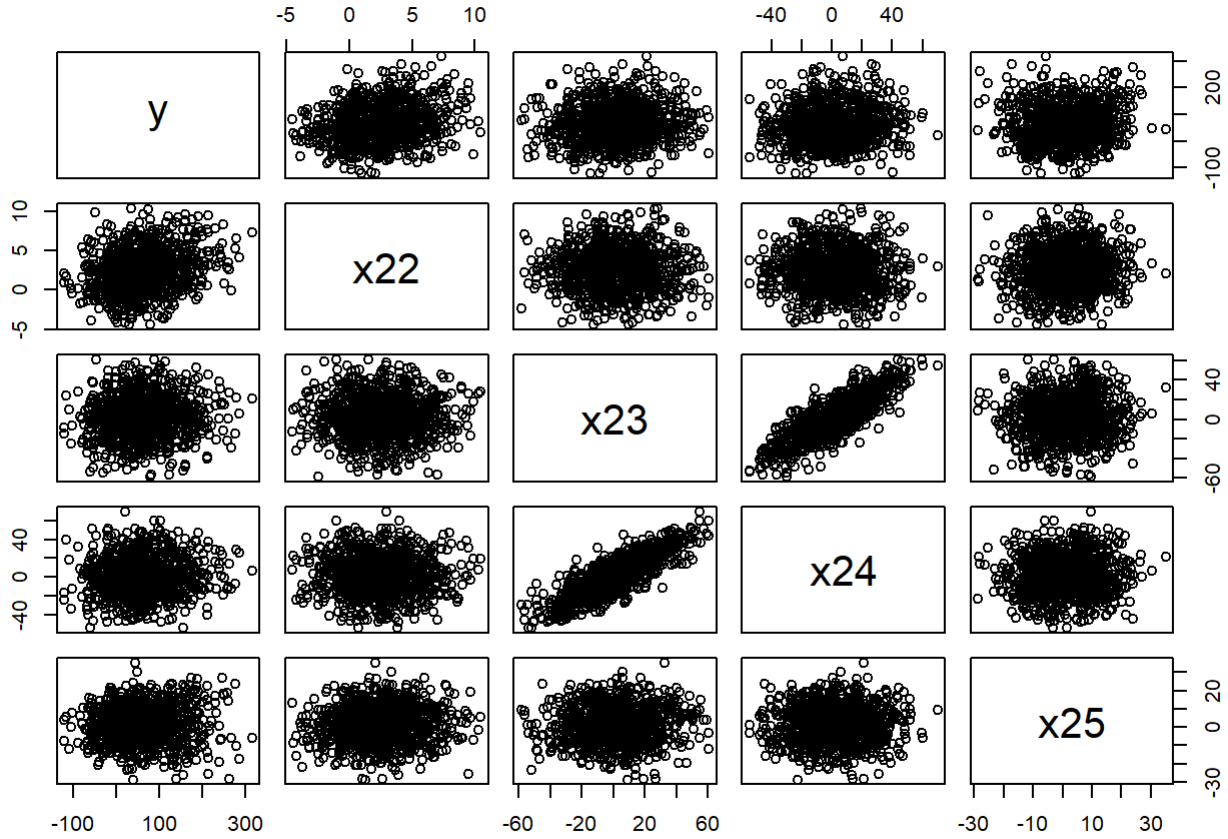

```
pairs(obsResponse[,c(26,16:18)])
```




```
pairs(obsResponse[,c(26,19:21)])
```




```
pairs(obsResponse[,c(26,22:25)])
```



Of

the above plots 3 variables in particular stand out with strong associations with the response y. These predictors are `x5` , `x10` , and `x22` . These 3 will get special attention when tuning our model. Also, there were several variables that mildly indicate an association with `y` . These are x7-x9 as well as x4 and x6. These 5 will be fit with a loess curve since the relationship is difficult to identify.

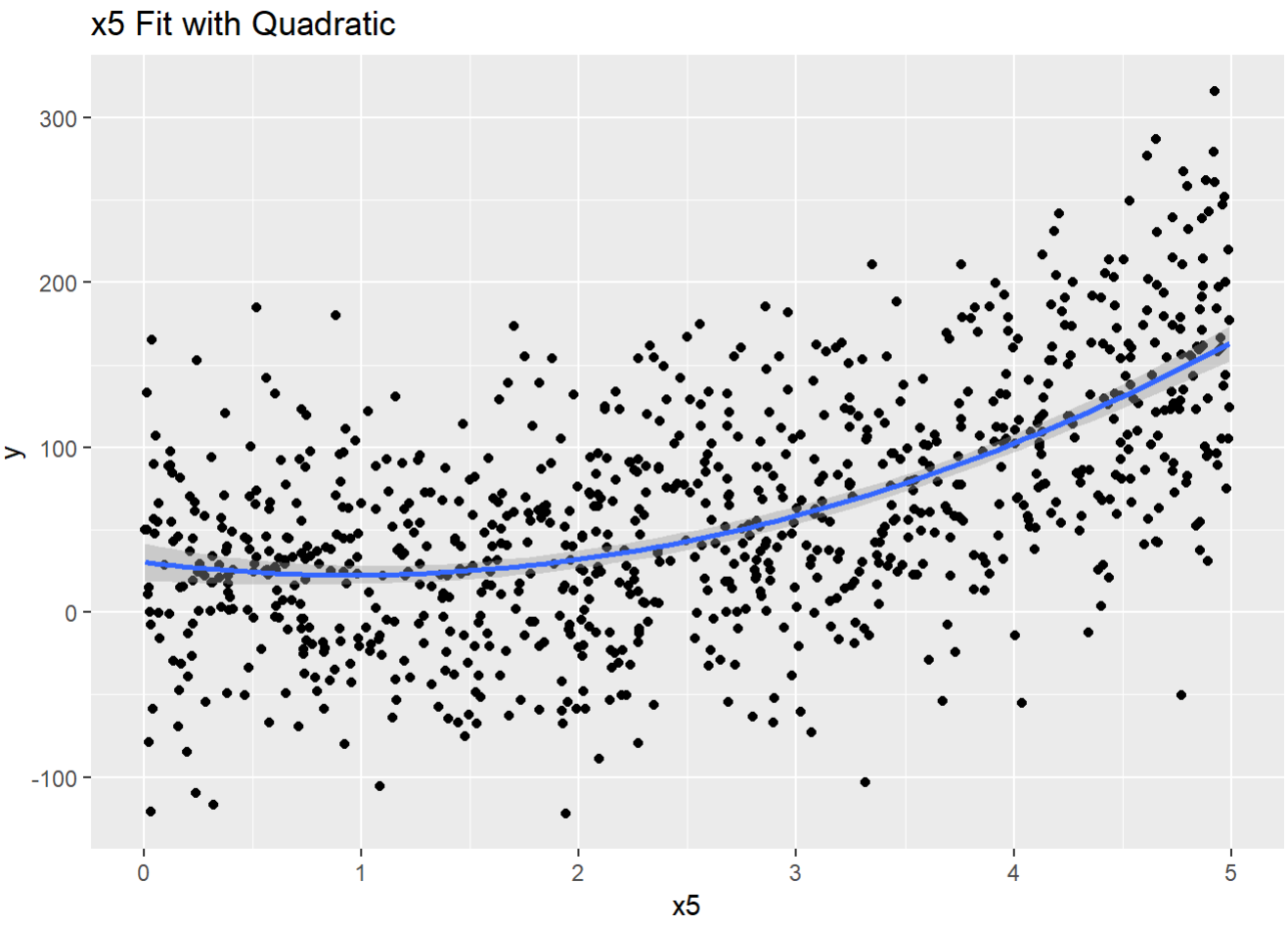
Fitted Functions

For `x5` we fit the scatter plot with a quadratic linear model. This appears to be a very good fit and for that reason we will use a quadratic linear model for x5 in our final GAM.

```
library(ggplot2)
```

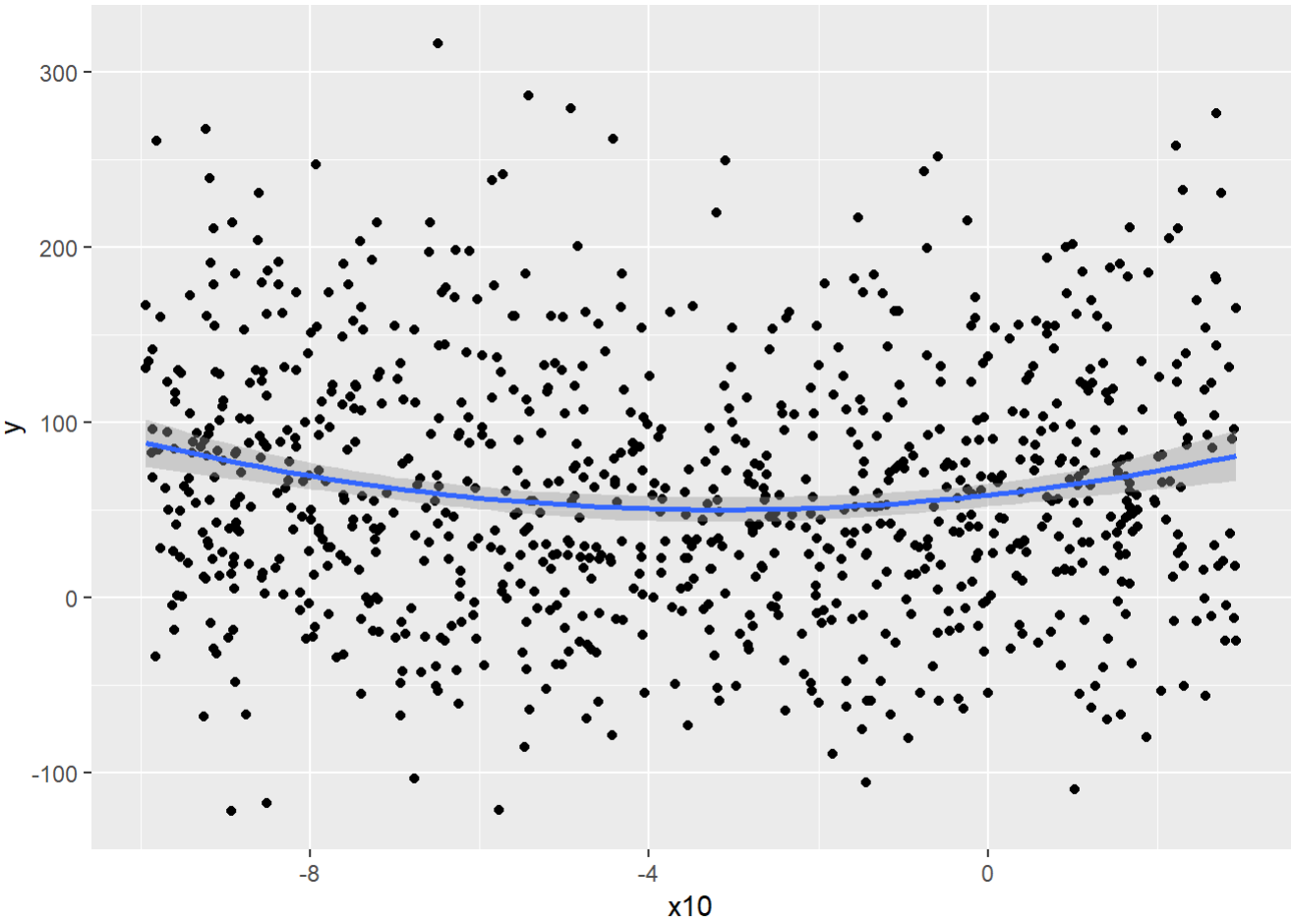
```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
ggplot(data = obsResponse,aes(x = x5,y=y))+
  geom_point()+
  stat_smooth(method = "lm", formula = y ~ x + I(x^2), size = 1)+
  ggtitle("x5 Fit with Quadratic")
```



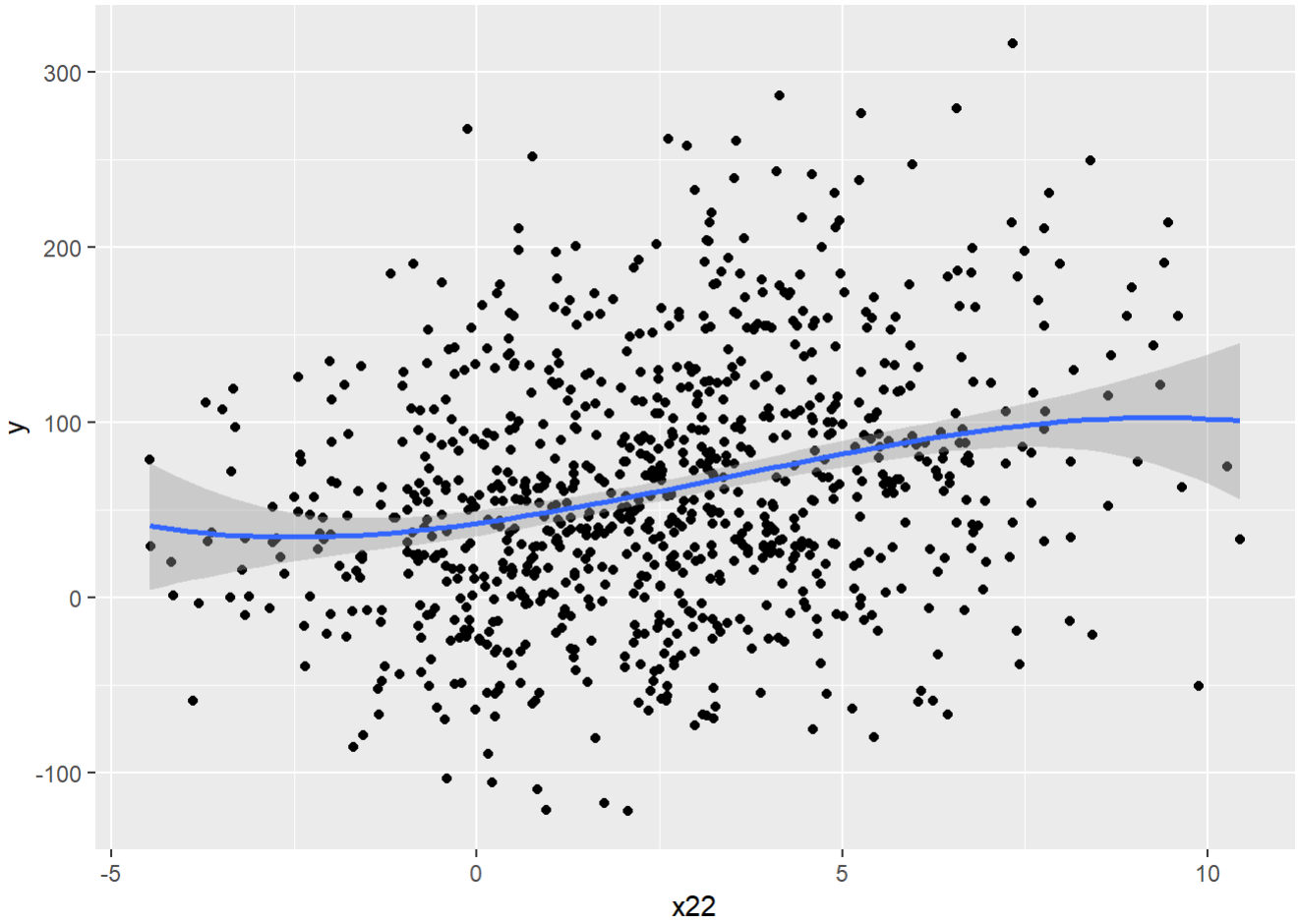
For `x10` we can see that it also follows roughly a 2nd order polynomial so we wll use that for our GAM function as well.

```
ggplot(data = obsResponse, aes(x = x10, y = y))+
  geom_point()+
  stat_smooth(method = "lm", formula = y~poly(x,2))
```



For `x22` we fit with a 3rd order polynomial so we will use that at x22's fitted function in our GAM model.

```
ggplot(data = obsResponse, aes(x=x22,y=y))+
  geom_point()+
  stat_smooth(method = "lm",formula = y~poly(x,3))
```



the other variables (x4,and x6-x9) have rather confusing associations with our response variable so we will use non parametric, local techniques such as loess and natural splines, as the ffitted functions.

Build GAM

In this example we will build 3 different GAM Models and compare their MSE of prediction to determine which model is the best. The first model `polyGAM` will be fit with only the polynomial functions from x5, x10, and x22. The second model `completeGAM` will be fit with the polynomial functions and loess models for the other 5 variables we identified as potentially significant. The last model `rawGAM` will be fit entirely from loess functions, and represents the most “flexible model.”

```
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.16
```

```
MSE = matrix(0,nrow = 3,ncol = 5)
for (i in 1:5) {
  #Set Training and test set
  trainingSet = setTrainTest(obsResponse,"train",i,5)
  testSet = setTrainTest(obsResponse,"test",i,5)

  #create 3 models on training fold
  polyGAM=gam(y~poly(x5,2)+poly(x10,2)+poly(x22,3),data = trainingSet)
  completeGAM = gam(y~poly(x5,2)+poly(x10,2)+poly(x22,3)+
                      lo(x4,span=0.5)+lo(x6,span=0.5)+lo(x7,span =0.5)+lo(x8,span=0.5)+lo(x9,spa
n=0.5),
                    data = trainingSet)
  rawGAM = gam(y~lo(x5,span = 0.5)+lo(x10,span = 0.5)+lo(x22,span = 0.5)+
                 lo(x4,span = 0.5)+lo(x6,span = 0.5)+lo(x7,span = 0.5)+
                 lo(x8,span = 0.5)+lo(x9,span = 0.5),data = trainingSet)

  #Make predictions
  polyHat = predict(polyGAM,testSet)
  completeHat = predict(completeGAM,testSet)
  rawHat = predict(rawGAM,testSet)

  ##Compute MSE
  MSE[1,i] = mean((polyHat-testSet$y)^2)
  MSE[2,i] = mean((completeHat-testSet$y)^2)
  MSE[3,i] = mean((rawHat-testSet$y)^2)
}

#MSE for poly
mean(MSE[1,])
```

```
## [1] 3215.243
```

```
#MSE for complete
mean(MSE[2,])
```

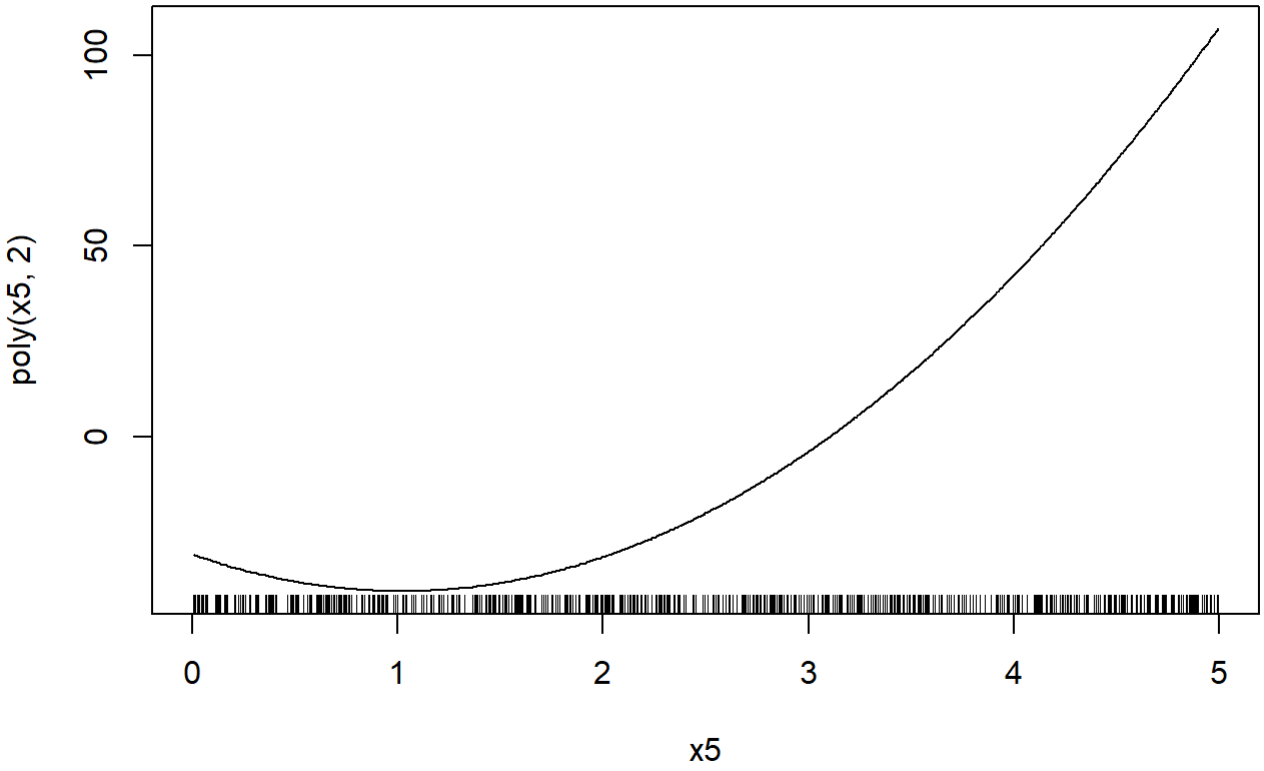
```
## [1] 3285.375
```

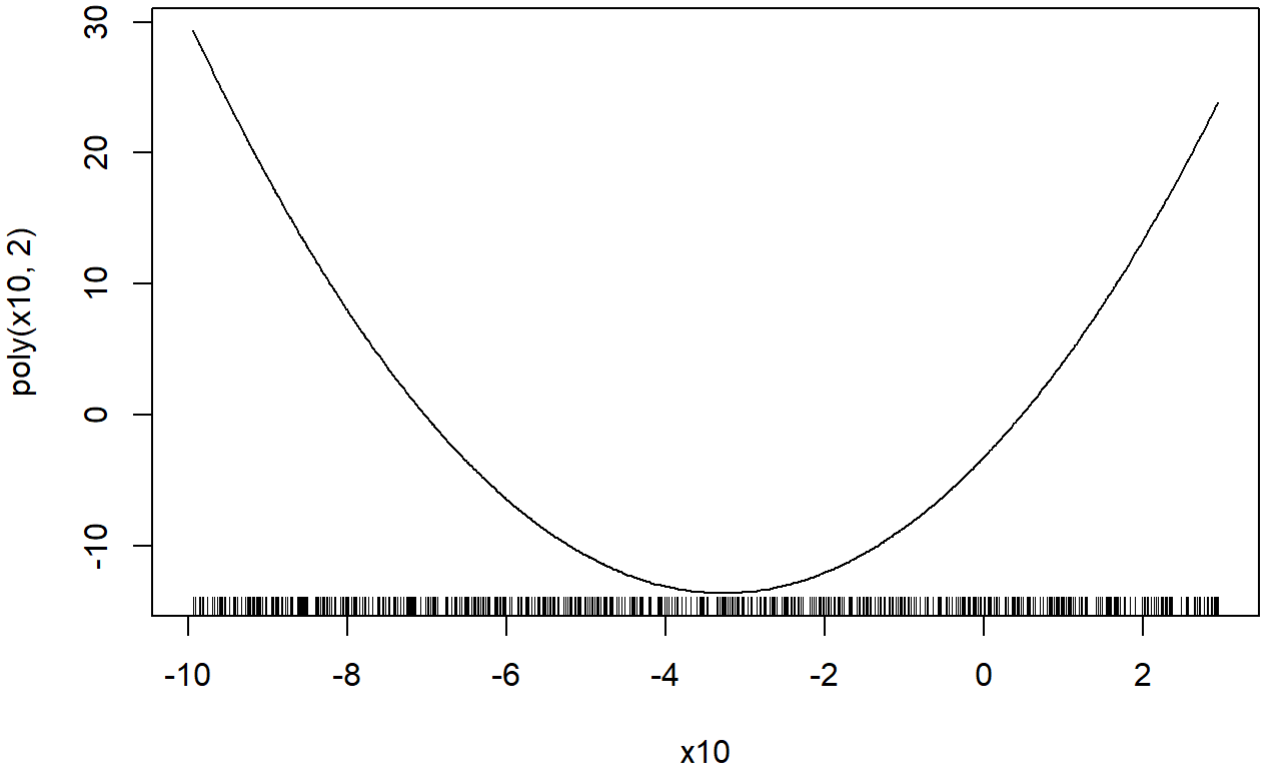
```
#MSE for raw
mean(MSE[3,])
```

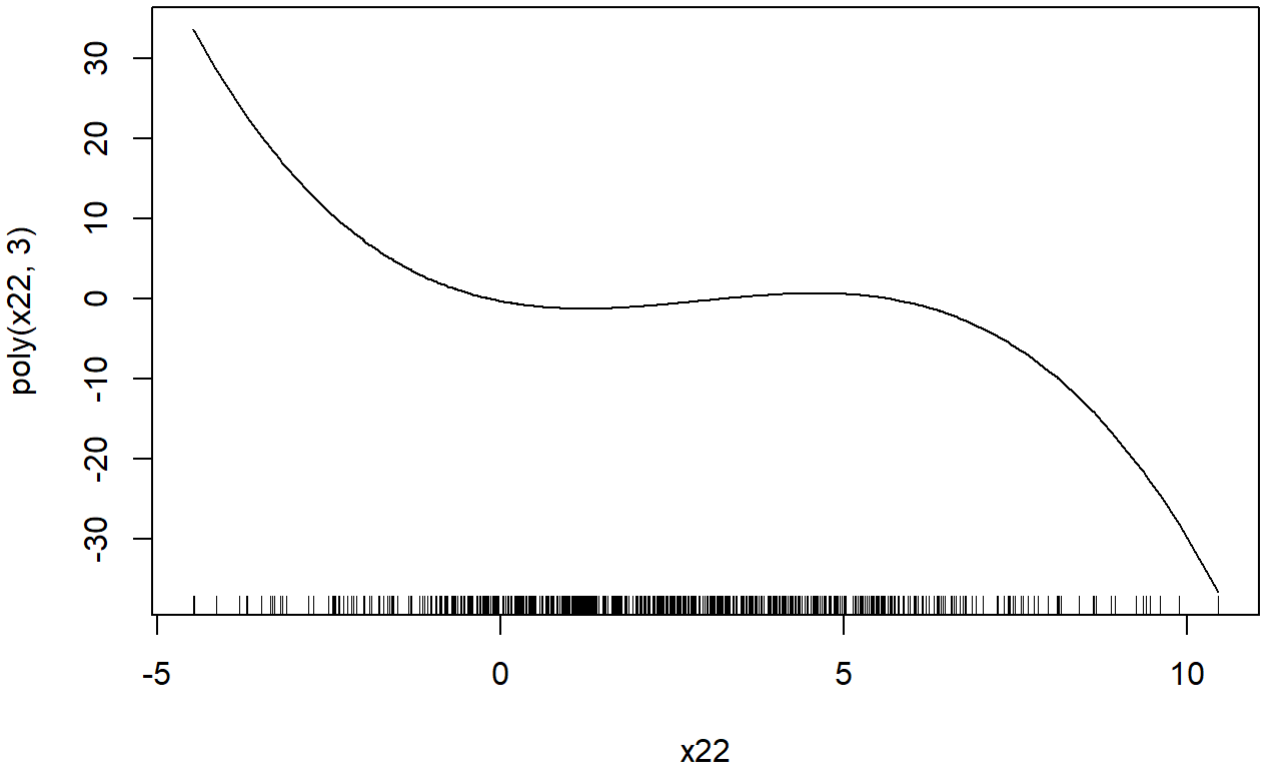
```
## [1] 3290.698
```

We can see that our polynomial model gives of the lowest MSE of prediction indicating that this is ths strongest model. In addition, it has the fewest predictors which makes it the simplest. Simple models that predict well are really powerful. Let’f visualize this model below.

```
plot.Gam(polyGAM)
```


poly(x5, 2)
100
50
0
0
1
2
3
4
5
x5


poly(x10, 2)
30
20
10
0
-10
-10
-8
-6
-4
-2
0
2
x10



Final Model

We will now format our final model and create a .csv file with our predictions.

```
finalGAM = gam(y~poly(x5,2)+poly(x10,2)+poly(x22,3),data = obsResponse)
predictions = predict(finalGAM,predResponse)
finalData = data.frame(id = predResponse$id,
                       predictedValue = predictions)
write.csv(finalData,"C:/2018_Fall/STAT_488/homework/hw4/predValues.csv",row.names = FALSE)
```