Final Project

Shreya Agrawal

4/6/2022

Packages

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.1.2

## Loading required package: Matrix

## Loaded glmnet 4.1-3

library(ISLR2)   
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.4 v dplyr 1.0.7  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 2.1.2 v forcats 0.5.1

## Warning: package 'readr' was built under R version 4.1.3

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x tidyr::expand() masks Matrix::expand()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x tidyr::pack() masks Matrix::pack()  
## x tidyr::unpack() masks Matrix::unpack()

library(ggplot2)   
library(leaps)

## Warning: package 'leaps' was built under R version 4.1.3

library(tree)

## Warning: package 'tree' was built under R version 4.1.3

## Registered S3 method overwritten by 'tree':  
## method from  
## print.tree cli

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.1.3

## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(pls)

## Warning: package 'pls' was built under R version 4.1.3

##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

library(gam)

## Warning: package 'gam' was built under R version 4.1.3

## Loading required package: splines

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.1.2

##   
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':  
##   
## accumulate, when

## Loaded gam 1.20.1

library (splines)   
library(ggplot2)   
library(boot)   
library(bootstrap)

##   
## Attaching package: 'bootstrap'

## The following object is masked from 'package:pls':  
##   
## crossval

#knitr::opts\_knit$set(root.dir = rprojroot::find\_rstudio\_root\_file())

Loading the data in

bikeData <- read.csv("C:\\Users\\shrey\\OneDrive - University of Pittsburgh\\1361-Data Science\\Final Project\\train.csv")

Data Tidying

bikeData <- na.omit(bikeData)  
head(bikeData)

## Count Date Hour Temperature Humidity Wind Visibility Dew Solar  
## 1 254 1/12/2017 0 -5.2 37 2.2 2000 -17.6 0  
## 2 204 1/12/2017 1 -5.5 38 0.8 2000 -17.6 0  
## 3 173 1/12/2017 2 -6.0 39 1.0 2000 -17.7 0  
## 4 107 1/12/2017 3 -6.2 40 0.9 2000 -17.6 0  
## 5 78 1/12/2017 4 -6.0 36 2.3 2000 -18.6 0  
## 6 100 1/12/2017 5 -6.4 37 1.5 2000 -18.7 0  
## Rainfall Snowfall Seasons Holiday Functioning ID  
## 1 0 0 Winter No Holiday Yes 840931  
## 2 0 0 Winter No Holiday Yes 595962  
## 3 0 0 Winter No Holiday Yes 227307  
## 4 0 0 Winter No Holiday Yes 791613  
## 5 0 0 Winter No Holiday Yes 199455  
## 6 0 0 Winter No Holiday Yes 272783

#Splitting the Date variable into three variables-Month, Day, Year  
bikeData\_new <-separate(bikeData,Date,c("Day","Month","Year"),sep="/")  
bikeData\_new$Day = as.numeric(bikeData\_new$Day)  
bikeData\_new$Month = as.numeric(bikeData\_new$Month)  
bikeData\_new$Year = as.numeric(bikeData\_new$Year)

head(bikeData\_new)

## Count Day Month Year Hour Temperature Humidity Wind Visibility Dew Solar  
## 1 254 1 12 2017 0 -5.2 37 2.2 2000 -17.6 0  
## 2 204 1 12 2017 1 -5.5 38 0.8 2000 -17.6 0  
## 3 173 1 12 2017 2 -6.0 39 1.0 2000 -17.7 0  
## 4 107 1 12 2017 3 -6.2 40 0.9 2000 -17.6 0  
## 5 78 1 12 2017 4 -6.0 36 2.3 2000 -18.6 0  
## 6 100 1 12 2017 5 -6.4 37 1.5 2000 -18.7 0  
## Rainfall Snowfall Seasons Holiday Functioning ID  
## 1 0 0 Winter No Holiday Yes 840931  
## 2 0 0 Winter No Holiday Yes 595962  
## 3 0 0 Winter No Holiday Yes 227307  
## 4 0 0 Winter No Holiday Yes 791613  
## 5 0 0 Winter No Holiday Yes 199455  
## 6 0 0 Winter No Holiday Yes 272783

Splitting the data into training and testing set

set.seed(1)  
train <- bikeData\_new %>% sample\_frac(size = 0.8)   
test <- bikeData\_new %>% setdiff(train)

Removing ID variable because it is insignificant to the response-Counts.

train <- train %>% select(-ID)   
test <- test %>% select(-ID)

Finding the best linear models to use by measuring RSS, CP, and BIC.

From the analysis below, I have discovered that using 15 predictors(if we consider each class of each categorical predictor as an individual predictor) results in the minimum possible cP value. These 15 predictors are-“Day”,“Month”,“Year”,“Hour”,“Temperature”,“Humidity”,“Wind”,“Solar”,“Rainfall”,“Snowfall”,“SeaonsSpring”,“SeasonsSummer”,“SeasonsWinter”,“HolidayNo Holiday”,"FunctioningYes

In other words, a model built using this data will perform best using the variables-Day,Month,Year,Hour,Temperature,Humidity,Wind,Solar,Rainfall,Snowfall,Seasons,Holiday,Functioning

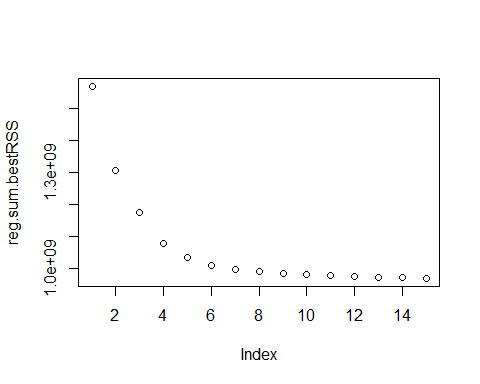
#Finding the best linear models and significant predictors   
regfit <- regsubsets (Count ~ ., data = train ,  
nvmax = 15,really.big = T)   
summary(regfit)

## Subset selection object  
## Call: regsubsets.formula(Count ~ ., data = train, nvmax = 15, really.big = T)  
## 17 Variables (and intercept)  
## Forced in Forced out  
## Day FALSE FALSE  
## Month FALSE FALSE  
## Year FALSE FALSE  
## Hour FALSE FALSE  
## Temperature FALSE FALSE  
## Humidity FALSE FALSE  
## Wind FALSE FALSE  
## Visibility FALSE FALSE  
## Dew FALSE FALSE  
## Solar FALSE FALSE  
## Rainfall FALSE FALSE  
## Snowfall FALSE FALSE  
## SeasonsSpring FALSE FALSE  
## SeasonsSummer FALSE FALSE  
## SeasonsWinter FALSE FALSE  
## HolidayNo Holiday FALSE FALSE  
## FunctioningYes FALSE FALSE  
## 1 subsets of each size up to 15  
## Selection Algorithm: exhaustive  
## Day Month Year Hour Temperature Humidity Wind Visibility Dew Solar  
## 1 ( 1 ) " " " " " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " " " "\*" "\*" " " " " " " " " " "   
## 3 ( 1 ) " " " " " " "\*" "\*" " " " " " " " " " "   
## 4 ( 1 ) " " " " " " "\*" "\*" "\*" " " " " " " " "   
## 5 ( 1 ) " " " " " " "\*" " " "\*" " " " " "\*" " "   
## 6 ( 1 ) " " " " " " "\*" "\*" "\*" " " " " " " " "   
## 7 ( 1 ) " " " " " " "\*" "\*" "\*" " " " " " " "\*"   
## 8 ( 1 ) " " " " " " "\*" "\*" "\*" " " " " " " "\*"   
## 9 ( 1 ) " " " " " " "\*" "\*" "\*" " " " " " " "\*"   
## 10 ( 1 ) " " " " " " "\*" "\*" "\*" " " " " " " "\*"   
## 11 ( 1 ) " " "\*" "\*" "\*" "\*" "\*" " " " " " " "\*"   
## 12 ( 1 ) " " "\*" "\*" "\*" "\*" "\*" " " " " " " "\*"   
## 13 ( 1 ) " " "\*" "\*" "\*" "\*" "\*" "\*" " " " " "\*"   
## 14 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " " " "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " " " "\*"   
## Rainfall Snowfall SeasonsSpring SeasonsSummer SeasonsWinter  
## 1 ( 1 ) " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " "   
## 3 ( 1 ) " " " " " " " " " "   
## 4 ( 1 ) " " " " " " " " " "   
## 5 ( 1 ) " " " " " " " " "\*"   
## 6 ( 1 ) "\*" " " " " " " "\*"   
## 7 ( 1 ) "\*" " " " " " " "\*"   
## 8 ( 1 ) "\*" " " " " "\*" "\*"   
## 9 ( 1 ) "\*" " " "\*" "\*" "\*"   
## 10 ( 1 ) "\*" " " "\*" "\*" "\*"   
## 11 ( 1 ) "\*" " " "\*" "\*" "\*"   
## 12 ( 1 ) "\*" " " "\*" "\*" "\*"   
## 13 ( 1 ) "\*" " " "\*" "\*" "\*"   
## 14 ( 1 ) "\*" " " "\*" "\*" "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" "\*"   
## HolidayNo Holiday FunctioningYes  
## 1 ( 1 ) " " " "   
## 2 ( 1 ) " " " "   
## 3 ( 1 ) " " "\*"   
## 4 ( 1 ) " " "\*"   
## 5 ( 1 ) " " "\*"   
## 6 ( 1 ) " " "\*"   
## 7 ( 1 ) " " "\*"   
## 8 ( 1 ) " " "\*"   
## 9 ( 1 ) " " "\*"   
## 10 ( 1 ) "\*" "\*"   
## 11 ( 1 ) " " "\*"   
## 12 ( 1 ) "\*" "\*"   
## 13 ( 1 ) "\*" "\*"   
## 14 ( 1 ) "\*" "\*"   
## 15 ( 1 ) "\*" "\*"

reg.sum <- summary(regfit)   
reg.sum.bestRSS <- reg.sum$rss   
which.min(reg.sum.bestRSS)

## [1] 15

plot(reg.sum.bestRSS)



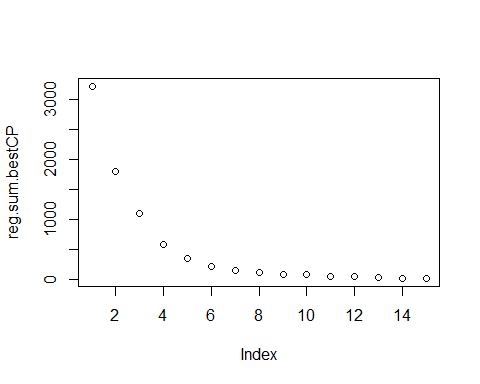
coef(regfit,which.min(reg.sum.bestRSS))

## (Intercept) Day Month Year   
## 1.075570e+06 -2.504501e+00 -4.350354e+01 -5.329163e+02   
## Hour Temperature Humidity Wind   
## 2.621828e+01 2.885960e+01 -8.552644e+00 2.124487e+01   
## Solar Rainfall Snowfall SeasonsSpring   
## -8.783692e+01 -5.548568e+01 4.012462e+01 -3.651982e+02   
## SeasonsSummer SeasonsWinter HolidayNo Holiday FunctioningYes   
## -3.114043e+02 -7.237397e+02 1.158753e+02 9.588869e+02

reg.sum <- summary(regfit)   
reg.sum.bestCP <- reg.sum$cp   
which.min(reg.sum.bestCP)

## [1] 15

plot(reg.sum.bestCP)



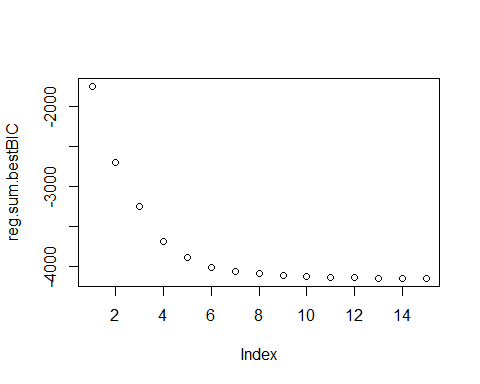
coef(regfit,which.min(reg.sum.bestCP))

## (Intercept) Day Month Year   
## 1.075570e+06 -2.504501e+00 -4.350354e+01 -5.329163e+02   
## Hour Temperature Humidity Wind   
## 2.621828e+01 2.885960e+01 -8.552644e+00 2.124487e+01   
## Solar Rainfall Snowfall SeasonsSpring   
## -8.783692e+01 -5.548568e+01 4.012462e+01 -3.651982e+02   
## SeasonsSummer SeasonsWinter HolidayNo Holiday FunctioningYes   
## -3.114043e+02 -7.237397e+02 1.158753e+02 9.588869e+02

reg.sum <- summary(regfit)   
reg.sum.bestBIC <- reg.sum$bic   
which.min(reg.sum.bestBIC)

## [1] 14

plot(reg.sum.bestBIC)



coef(regfit,which.min(reg.sum.bestBIC))

## (Intercept) Day Month Year   
## 1.066449e+06 -2.179148e+00 -4.256379e+01 -5.284114e+02   
## Hour Temperature Humidity Wind   
## 2.639209e+01 2.847764e+01 -8.258815e+00 2.139011e+01   
## Solar Rainfall SeasonsSpring SeasonsSummer   
## -8.392654e+01 -5.625664e+01 -3.636484e+02 -3.094072e+02   
## SeasonsWinter HolidayNo Holiday FunctioningYes   
## -7.124092e+02 1.180296e+02 9.596499e+02

Getting rid of insignificant variables based on results of best subset selection

tr <- train %>% select(-Visibility,-Dew)   
te <- test %>% select(-Visibility,-Dew)

Multinomial Linear Regression Model The test MSE of this model of all the variables is 682003. The test MSE of this model when only the significant predictors are used is 682062. The test MSE of this model when only the most significant predictor-Temperature-is used, is 562903.4

#Using all of the predictor terms  
linear.mod <- lm(Count ~., data = train)   
linear.pred <- predict(linear.mod,data=test)   
linear.mse <- mean ((test$Count - linear.pred)^2)

## Warning in test$Count - linear.pred: longer object length is not a multiple of  
## shorter object length

linear.mse

## [1] 682003

#Using significant predictor terms based on best subset selection  
linear.mod <- lm(Count ~., data = tr)   
linear.pred <- predict(linear.mod,data=te)   
linear.mse <- mean ((te$Count - linear.pred)^2)

## Warning in te$Count - linear.pred: longer object length is not a multiple of  
## shorter object length

linear.mse

## [1] 682062

#Multinomial Linear Regression with only the most significant predictor  
linear.mod <- lm(Count ~ Temperature, data = train)   
linear.pred <- predict(linear.mod,data=test)   
linear.mse <- mean ((test$Count - linear.pred)^2)

## Warning in test$Count - linear.pred: longer object length is not a multiple of  
## shorter object length

linear.mse

## [1] 562903.4

Ridge Regression Model  
The MSE with of this model using all predictors is 189448.4, the MSE when only the significant variables are used is 190706.7, and the MSE when only the most significant variable- Temperature variable is used, is 306187.2

set.seed(1)   
  
#Converting test and training data into matrices so that the glmnet function can be used  
train\_x <- model.matrix(Count ~.,train)  
train\_y <- train$Count   
  
test\_x <- model.matrix(Count ~.,test)  
test\_y <- test$Count  
  
#Doing cross validation to find the best lambda to use for ridge  
cv\_ridge <- cv.glmnet(train\_x, train\_y, alpha=0)  
best\_lambda\_ridge <- cv\_ridge$lambda.min   
  
#Fitting ridge model with best lambda value   
ridge.mod <- glmnet(train\_x,train\_y,alpha=0,lambda = best\_lambda\_ridge)  
  
#Calculating test error(test MSE)   
ridge.pred <- predict(ridge.mod, s=best\_lambda\_ridge,newx = test\_x)   
ridge.mse <- mean ((test$Count - ridge.pred)^2)   
ridge.mse

## [1] 189448.4

set.seed(1)  
train\_x <- model.matrix(Count ~.,tr)  
train\_y <- tr$Count   
  
test\_x <- model.matrix(Count ~.,te)  
test\_y <- te$Count  
  
#Doing cross validation to find the best lambda to use for ridge  
cv\_ridge <- cv.glmnet(train\_x, train\_y, alpha=0)  
best\_lambda\_ridge <- cv\_ridge$lambda.min   
  
#Fitting ridge model with best lambda value   
ridge.mod <- glmnet(train\_x,train\_y,alpha=0,lambda = best\_lambda\_ridge)  
  
#Calculating test error(test MSE)   
ridge.pred <- predict(ridge.mod, s=best\_lambda\_ridge,newx = test\_x)   
ridge.mse <- mean ((te$Count - ridge.pred)^2)   
ridge.mse

## [1] 190706.7

set.seed(1)  
train\_x <- model.matrix(Count ~ Temperature,train)  
train\_y <- train$Count   
  
test\_x <- model.matrix(Count ~ Temperature,test)  
test\_y <- test$Count  
  
#Doing cross validation to find the best lambda to use for ridge  
cv\_ridge <- cv.glmnet(train\_x, train\_y, alpha=0)  
best\_lambda\_ridge <- cv\_ridge$lambda.min   
  
#Fitting ridge model with best lambda value   
my\_ridge <- glmnet(train\_x,train\_y,alpha=0,lambda = best\_lambda\_ridge)  
  
#Calculating test error(test MSE)   
ridge.pred <- predict(my\_ridge, s=best\_lambda\_ridge,newx = test\_x)   
ridge.mse <- mean ((test$Count - ridge.pred)^2)   
ridge.mse

## [1] 306187.2

Lasso Model  
The MSE of this model with all predictors is 204254.3, with only most significant predictors is 205228.5, and the MSE using this model with only the Temperature variable is 307582.2

set.seed(1)  
train\_x <- model.matrix(Count ~.,train)  
train\_y <- train$Count   
  
test\_x <- model.matrix(Count ~.,test)  
test\_y <- test$Count  
  
#Doing cross validation to find the best lambda to use for lasso  
cv\_lasso <- cv.glmnet(train\_x, train\_y, alpha=1)  
best\_lambda\_lasso <- cv\_lasso$lambda.min   
  
#Fitting lasso model with best lambda value   
my\_lasso <- glmnet(train\_x,train\_y,alpha=1,lambda = best\_lambda\_ridge)  
  
#Calculating test error(test MSE)   
lasso.pred <- predict(my\_lasso, s=best\_lambda\_ridge,newx = test\_x)   
lasso.mse <- mean ((test$Count - lasso.pred)^2)   
lasso.mse

## [1] 204254.3

set.seed(1)  
train\_x <- model.matrix(Count ~.,tr)  
train\_y <- tr$Count   
  
test\_x <- model.matrix(Count ~.,te)  
test\_y <- te$Count  
  
#Doing cross validation to find the best lambda to use for lasso  
cv\_lasso <- cv.glmnet(train\_x, train\_y, alpha=1)  
best\_lambda\_lasso <- cv\_lasso$lambda.min   
  
#Fitting lasso model with best lambda value   
my\_lasso <- glmnet(train\_x,train\_y,alpha=1,lambda = best\_lambda\_ridge)  
  
#Calculating test error(test MSE)   
lasso.pred <- predict(my\_lasso, s=best\_lambda\_ridge,newx = test\_x)   
lasso.mse <- mean ((te$Count - lasso.pred)^2)   
lasso.mse

## [1] 205228.5

set.seed(1)  
train\_x <- model.matrix(Count ~ Temperature,train)  
train\_y <- train$Count   
  
test\_x <- model.matrix(Count ~ Temperature,test)  
test\_y <- test$Count  
  
#Doing cross validation to find the best lambda to use for lasso  
cv\_lasso <- cv.glmnet(train\_x, train\_y, alpha=1)  
best\_lambda\_lasso <- cv\_lasso$lambda.min   
  
#Fitting lasso model with best lambda value   
my\_lasso <- glmnet(train\_x,train\_y,alpha=1,lambda = best\_lambda\_ridge)  
  
#Calculating test error(test MSE)   
lasso.pred <- predict(my\_lasso, s=best\_lambda\_ridge,newx = test\_x)   
lasso.mse <- mean ((test$Count - lasso.pred)^2)   
lasso.mse

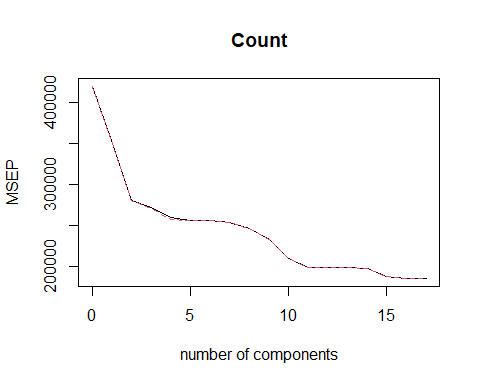
## [1] 307582.2

PCR-Dimension Reduction  
The MSE of this model using all predictors is 682003. The MSE using only the most significant predictors is 667469.9. Upon conducting a cross validation test, I also discovered that the ncomp with minimum MSE is 1. Furthermore, when I fit this model with only the most significant variable-Temperature-, the MSE was 531729.7

#Fitting PCR model, reducing number of predictors by using transformations of multiple predictors   
  
train\_x <- model.matrix(Count ~.,train)  
train\_y <- train$Count   
  
test\_x <- model.matrix(Count ~.,test)  
test\_y <- test$Count   
  
   
set.seed(1)   
pcr.fit <- pcr(Count~.,data = train,scale=TRUE,validation="CV")   
#Analyzing the resulting fit and cross validation results(which M produces least error)   
summary(pcr.fit)

## Data: X dimension: 5242 17   
## Y dimension: 5242 1  
## Fit method: svdpc  
## Number of components considered: 17  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 647.5 594.4 529.4 521.2 510.1 506.5 506.5  
## adjCV 647.5 594.4 529.4 521.2 507.8 506.5 506.5  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 503.6 496.1 483.3 458.9 446.7 446.7 445.9  
## adjCV 503.6 496.4 483.4 458.8 446.6 446.6 445.9  
## 14 comps 15 comps 16 comps 17 comps  
## CV 444.7 433.8 431.6 431.5  
## adjCV 444.6 433.7 431.5 431.4  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 21.08 34.94 45.69 52.92 60.08 66.36 72.19 77.52  
## Count 15.75 33.21 35.29 38.36 38.91 38.92 39.63 41.38  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 82.65 87.44 91.39 94.97 97.63 99.01 99.90  
## Count 44.39 50.03 52.62 52.63 52.82 53.10 55.36  
## 16 comps 17 comps  
## X 99.97 100.00  
## Count 55.84 55.89

#Plotting the number of components(M) with the cross validation MSEP. Based on the plot and summary, the lowest cross-validation error occurs when M = 18.   
validationplot(pcr.fit,val.type = "MSEP")



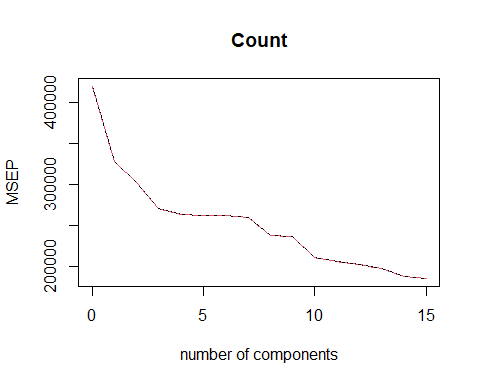
pcr.pred <- predict (pcr.fit,newx=test\_x, ncomp = 17)  
pcr.mse <- mean (( test\_y - pcr.pred ) ^2)   
pcr.mse

## [1] 682003

#Fitting PCR model with only most significant predictors  
  
train\_x <- model.matrix(Count ~.,tr)  
train\_y <- tr$Count   
  
test\_x <- model.matrix(Count ~.,te)  
test\_y <- te$Count   
  
   
set.seed(1)   
pcr.fit <- pcr(Count~.,data = tr,scale=TRUE,validation="CV")   
#Analyzing the resulting fit and cross validation results(which M produces least error)   
summary(pcr.fit)

## Data: X dimension: 5242 15   
## Y dimension: 5242 1  
## Fit method: svdpc  
## Number of components considered: 15  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 647.5 573.0 549.8 520.8 514.0 512.6 512.6  
## adjCV 647.5 572.9 549.8 520.7 513.9 512.6 512.6  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 509.5 488.2 486.5 460.0 454.9 450.1 444.6  
## adjCV 509.6 488.1 486.8 459.9 454.9 450.0 444.5  
## 14 comps 15 comps  
## CV 433.5 431.3  
## adjCV 433.5 431.2  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 18.12 32.15 43.53 51.71 58.92 66.03 72.58 78.51  
## Count 21.71 27.94 35.38 37.08 37.43 37.44 38.21 43.35  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 84.18 89.58 93.77 97.08 99.10 99.92 100.00  
## Count 43.64 49.82 50.89 51.94 53.09 55.38 55.88

#Plotting the number of components(M) with the cross validation MSEP. Based on the plot and summary, the lowest cross-validation error occurs when M = 13.   
validationplot(pcr.fit,val.type = "MSEP")



pcr.pred <- predict (pcr.fit,newx=test\_x, ncomp = 13)  
pcr.mse <- mean (( test\_y - pcr.pred ) ^2)   
pcr.mse

## [1] 667469.9

#Finding the ncomp which gives the minimum mse  
minMSE = 1000000000   
optNComp = 0  
  
for (i in 1:15)   
{  
 pcr.pred <- predict (pcr.fit,newx=test\_x, ncomp = i)  
 pcr.mse <- mean (( test\_y - pcr.pred ) ^2)   
 if(pcr.mse < minMSE)   
 {   
 minMSE = pcr.mse   
 optNComp = i  
 }  
}   
  
#Optimal degree was found to be 1  
optNComp

## [1] 1

#Temperature is automatically chosen to be the predictor when ncomp is 1 because it is the most signficant predictor  
pcr.mse = minMSE   
pcr.mse

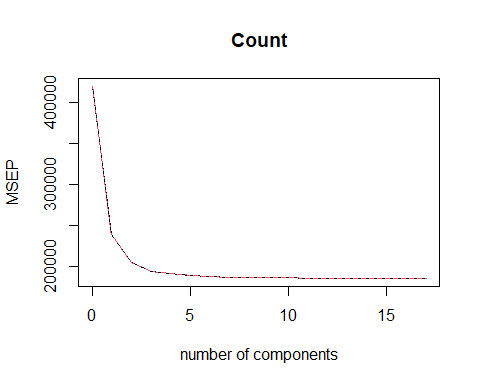
## [1] 531729.7

PLS  
The MSE is 682003 when all variables are used, is 682061.8 when only most significant predictors are considered, and is 562903.4 when only most signficant predictor-Temperature- is used to build the model.

train\_x <- model.matrix(Count ~.,train)  
train\_y <- train$Count   
  
test\_x <- model.matrix(Count ~.,test)  
test\_y <- test$Count   
  
set.seed(1)   
pls.fit <- plsr(Count~.,data = train,scale=TRUE,validation="CV")   
#Analyzing the resulting fit and cross validation results(which M produces least error)   
summary(pls.fit)

## Data: X dimension: 5242 17   
## Y dimension: 5242 1  
## Fit method: kernelpls  
## Number of components considered: 17  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 647.5 488.9 453.5 440.7 437.8 434.9 433.6  
## adjCV 647.5 488.9 453.4 440.7 437.7 434.8 433.5  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 433.3 433.0 432.7 432.2 431.9 431.5 431.5  
## adjCV 433.2 432.9 432.7 432.1 431.8 431.4 431.4  
## 14 comps 15 comps 16 comps 17 comps  
## CV 431.5 431.5 431.5 431.5  
## adjCV 431.4 431.5 431.5 431.4  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 17.96 30.89 40.0 47.18 52.80 58.30 62.65 65.84  
## Count 43.07 51.14 53.9 54.55 55.14 55.42 55.47 55.52  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 69.3 71.98 75.91 79.55 84.88 89.91 92.58  
## Count 55.6 55.74 55.83 55.88 55.88 55.88 55.89  
## 16 comps 17 comps  
## X 95.27 100.00  
## Count 55.89 55.89

#Plotting the number of components(M) with the cross validation MSE. Based on the plot, the lowest cross-validation error occurs when M = 5  
validationplot(pls.fit,val.type = "MSEP")



#Computing the test MSE of PLS with M=5   
pls.pred <- predict (pls.fit, newx=test\_x, ncomp = 17)  
pls.mse <- mean ((test$Count - pls.pred)^2)

## Warning in test$Count - pls.pred: longer object length is not a multiple of  
## shorter object length

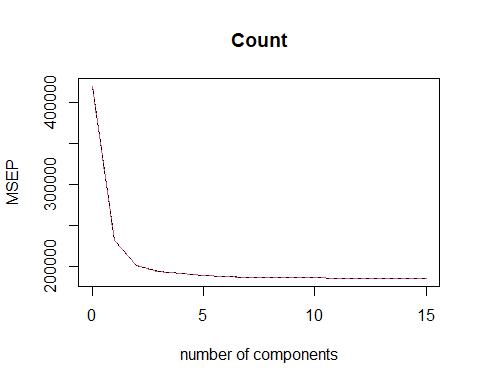
pls.mse

## [1] 682003

train\_x <- model.matrix(Count ~.,tr)  
train\_y <- tr$Count   
  
test\_x <- model.matrix(Count ~.,te)  
test\_y <- te$Count   
  
set.seed(1)   
pls.fit <- plsr(Count~.,data = tr,scale=TRUE,validation="CV")   
#Analyzing the resulting fit and cross validation results(which M produces least error)   
summary(pls.fit)

## Data: X dimension: 5242 15   
## Y dimension: 5242 1  
## Fit method: kernelpls  
## Number of components considered: 15  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 647.5 482.1 449.1 441.3 437.8 434.9 433.6  
## adjCV 647.5 482.1 449.0 441.2 437.7 434.9 433.5  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 433 432.8 432.6 432.2 431.4 431.3 431.3  
## adjCV 433 432.7 432.6 432.1 431.4 431.2 431.2  
## 14 comps 15 comps  
## CV 431.3 431.3  
## adjCV 431.2 431.2  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 16.26 26.32 36.07 44.19 51.02 56.69 61.30 65.43  
## Count 44.67 52.13 53.79 54.52 55.10 55.36 55.49 55.54  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 70.08 74.05 76.78 81.08 87.22 93.16 100.00  
## Count 55.58 55.67 55.84 55.88 55.88 55.88 55.88

#Plotting the number of components(M) with the cross validation MSE. Based on the plot, the lowest cross-validation error occurs when M = 5  
validationplot(pls.fit,val.type = "MSEP")



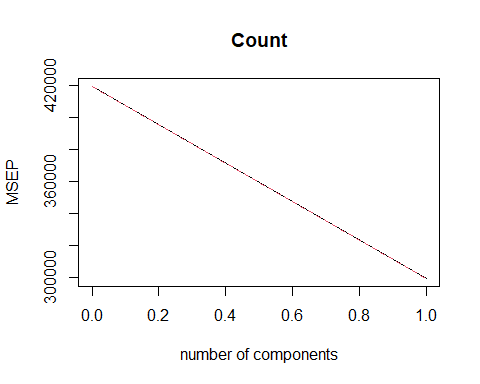
#Computing the test MSE of PLS with M=5   
pls.pred <- predict (pls.fit, newx=test\_x, ncomp = 13)  
pls.mse <- mean ((test$Count - pls.pred)^2)

## Warning in test$Count - pls.pred: longer object length is not a multiple of  
## shorter object length

pls.mse

## [1] 682061.8

set.seed(1)   
pls.fit <- plsr(Count~Temperature,data = train,scale=TRUE,validation="CV")   
#Analyzing the resulting fit and cross validation results(which M produces least error)   
#summary(pls.fit)   
  
  
#Plotting the number of components(M) with the cross validation MSE. Based on the plot, the lowest cross-validation error occurs when M = 5  
validationplot(pls.fit,val.type = "MSEP")



#Computing the test MSE of PLS with M=5   
pls.pred <- predict (pls.fit, newx=test\_x, ncomp = 1)  
pls.mse <- mean ((test$Count - pls.pred)^2)

## Warning in test$Count - pls.pred: longer object length is not a multiple of  
## shorter object length

pls.mse

## [1] 562903.4

Cubic Regression Splines  
The MSE of a cubic regression spline model with 4 degrees of freedom is 301163.1

#We use temperature as the predictor variable because based on best subset selection, that was the most significant variable  
min.mse = 10000000000   
optDegree = 0  
set.seed(1)   
  
#Using a for loop to find the optimal degrees of freedom which gives minimum MSE  
for(i in 1:20)   
{   
 spline.mod <- lm(Count ~ bs(Temperature,df=i),data=train)  
 spline.pred <- predict(spline.mod,test)  
 curr.mse = mean((test$Count - spline.pred)^2)   
 if(curr.mse < min.mse)   
 {   
 min.mse = curr.mse   
 optDegree = i  
 }  
}

## Warning in bs(Temperature, df = i): 'df' was too small; have used 3  
  
## Warning in bs(Temperature, df = i): 'df' was too small; have used 3

spline.mse = min.mse   
  
#Optimal degree was found to be 4 degrees of freedom  
optDegree

## [1] 4

spline.mse

## [1] 301163.1

Smoothing Spline The MSE of a cubic smoothing spline with 6 degrees of freedom is 618431.1

#We use temperature as the predictor variable because based on best subset selection, that was the most significant variable  
  
min.mse = 10000000000   
optDegree = 0  
set.seed(1)   
  
#Using a for loop to find the optimal degrees of freedom which gives minimum MSE  
for(i in 1:20)   
{   
 smooth.spline.mod <- smooth.spline(train[,'Temperature'],train[,'Count'],df=i,cv=TRUE)  
 smooth.spline.pred <- predict(smooth.spline.mod,test[,'Temperature'])   
 smooth.spline.pred <- unlist(smooth.spline.pred)   
 smooth.spline.pred <- as.numeric(smooth.spline.pred)  
 smooth.curr.mse = mean((test[,'Count'] - smooth.spline.pred)^2)   
 if(smooth.curr.mse < min.mse)   
 {   
 min.mse = smooth.curr.mse   
 optDegree = i  
 }  
}   
smooth.spline.mse = min.mse   
  
#Optimal degree was found to be 6 degrees of freedom  
optDegree

## [1] 6

smooth.spline.mse

## [1] 618431.1

GAM with regression splines I used only the most significant predictors with degrees of freedom of 4 for each predictor and found the MSE to be 144964.5. I used the single most significant predictor with degrees of freedom of 4 and found the MSE to be 301682.3

#Including Year variable causes error  
set.seed(1)   
  
gam.mod <- lm(Count ~ ns(Day,4)+ ns(Month,4) + ns( Temperature , 4) + ns(Hour , 4) + ns(Wind,4) + ns(Snowfall,4) + ns(Humidity,4) + ns(Solar,4) + ns(Rainfall,4) + Seasons + Functioning + Holiday,  
data = train )   
gam.pred <- predict(gam.mod,newdata=test)   
gam.mse = mean((test$Count - gam.pred)^2)   
gam.mse

## [1] 144964.5

set.seed(1)  
gam.mod <- lm(Count ∼ ns( Temperature , 4), data = train )   
gam.pred <- predict(gam.mod,newdata=test)   
gam.mse = mean((test$Count - gam.pred)^2)   
gam.mse

## [1] 301682.3

GAM with smoothing splines I used the 7 most significant predictors with degrees of freedom of 6 for each predictor and found the MSE to be 188317.2. I used the single most significant predictor with degrees of freedom of 6 and found the MSE to be 305366.3

set.seed(1)  
gam.mod <- lm(Count ∼ s(Day,6) + s(Month,6) + s( Temperature , 6) + s(Hour , 6) + s(Wind,6) + s(Snowfall,6) + s(Humidity,6) + s(Solar,6) + s(Rainfall,6) + Seasons + Functioning + Holiday,  
data = train )   
gam.pred <- predict(gam.mod,newdata=test)   
gam.mse = mean((test$Count - gam.pred)^2)   
gam.mse

## [1] 188317.2

set.seed(1)  
gam.mod <- lm(Count ∼ s( Temperature , 6), data = train )   
gam.pred <- predict(gam.mod,newdata=test)   
gam.mse = mean((test$Count - gam.pred)^2)   
gam.mse

## [1] 305366.3

Tree model  
The MSE of a full tree model with all predictors is 199860.7

set.seed(1)  
tree.mod <- tree(Count~.,data=train)

## Warning in tree(Count ~ ., data = train): NAs introduced by coercion

#summary(tree.mod)   
#plot(tree.mod)   
#text(tree.mod,pretty=0)   
#tree.mod  
  
#Finding the MSE   
tree.pred <- predict(tree.mod,test)

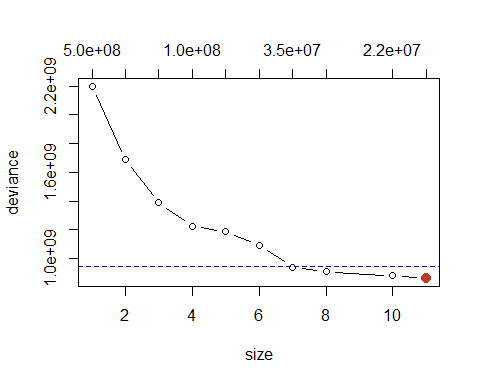
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

mean((tree.pred-test$Count)^2)

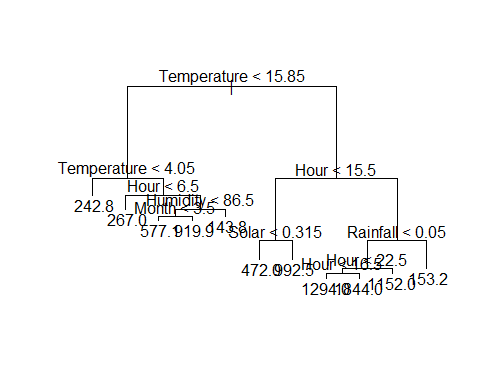
## [1] 199860.7

Tree Model with Pruning  
Pruning didn’t improve the error, the MSE is still the exact same as the tree model’s -> 199860.7

#Using cross-validation in order to determine the optimal level of tree complexity.   
set.seed(1)   
  
#Based on the plot, 11 is the optimal number of nodes, gives the least amount of cross validation error  
cv.mod <- cv.tree(tree.mod)   
plot(cv.mod, type = "b")   
abline(h = min(cv.mod$dev) + 0.2 \* sd(cv.mod$dev), col = "blue", lty = 2)  
points(cv.mod$size[which.min(cv.mod$dev)], min(cv.mod$dev),   
 col = "#BC3C29FF", cex = 2, pch = 20)



#Pruning the tree model to use only 11 nodes  
set.seed(1)  
prune.mod = prune.tree(tree.mod, best = 11)  
plot(prune.mod)  
text(prune.mod, pretty = 0)



prune.pred <- predict(prune.mod,test)

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

mean((prune.pred-test$Count)^2)

## [1] 199860.7

Bagging  
The MSE of the Bagging model is 57543.61

bag.mod <- randomForest(Count~.,data=train,mtry = 16,importance = TRUE)

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid  
## range

#Calculating test MSE   
bag.pred <- predict(bag.mod,test)   
MSE <- mean ((test$Count - bag.pred)^2)   
MSE

## [1] 57543.61

Random Forest  
The optimal mtry value was found to be 6, and the MSE of the Random Forest model with this mtry is 52028.69

minMSE = 100000000000000   
opt\_mtry = 0  
for(i in 1:16)  
{  
 randfor.mod <- randomForest(Count~.,data=train,mtry = i,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 if(MSE < minMSE)  
 {  
 minMSE = MSE  
 opt\_mtry = i   
 }  
}

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid  
## range

minMSE

## [1] 52122.06

opt\_mtry

## [1] 8

Temperature had the highest IncNodePurity of 658608257.9 while Hour had the highest %IncMSE of 212.669337.

importance(randfor.mod)

## %IncMSE IncNodePurity  
## Day 29.72908 30610151.7  
## Month 56.45362 43930244.9  
## Year 7.71744 674354.6  
## Hour 217.05592 627487156.9  
## Temperature 155.90560 659163960.5  
## Humidity 66.87101 125963584.6  
## Wind 23.11546 29927647.9  
## Visibility 31.93035 31027896.5  
## Dew 57.14900 65582441.7  
## Solar 95.54847 177287991.8  
## Rainfall 60.81195 137242296.9  
## Snowfall 17.65923 2369179.0  
## Seasons 52.12713 49135798.7  
## Holiday 23.11325 4804435.5  
## Functioning 145.32617 199326173.2

Creating null model to compare MSE to best model’s MSE. The MSE of this null model is very high-> 2643199. It was expected that the MSE of this model be extremely high because it is a null model and randomly assigns predictions.

range(train$Count)

## [1] 0 3556

testLength = nrow(test)   
null.pred = numeric()   
  
#Using sample function to randomly assign each observation in the test dataset a Count value in the same range as the range of the Count values of the training dataset.   
for(i in 1:testLength)   
{   
 null.pred[i] = sample(0:3556,replace = T)   
}  
  
#Finding the MSE of the null model  
MSE <- mean ((test$Count - null.pred)^2)   
MSE

## [1] 2556108

Using best model to predict actual test observations

#GAM with Regression Splines using only the most significant predictors seemed to perform best out of all the models   
  
  
#Loading data in   
testData <- read.csv("testData.csv")   
test.pred <- predict(gam.mod,newdata=testData)   
test.pred

## 1 2 3 4 5 6   
## 235.714207 229.972429 212.747096 198.392651 184.038207 169.683762   
## 7 8 9 10 11 12   
## 163.941984 149.587539 135.233095 143.845762 184.038207 229.972429   
## 13 14 15 16 17 18   
## 278.777541 287.390208 298.873763 313.228208 304.615541 267.293985   
## 19 20 21 22 23 24   
## 247.197763 241.455985 241.455985 241.455985 238.585096 235.714207   
## 25 26 27 28 29 30   
## 20.397538 14.655760 6.043093 3.172204 3.172204 3.172204   
## 31 32 33 34 35 36   
## -2.569573 -11.182240 -16.924018 -2.569573 34.751983 66.331761   
## 37 38 39 40 41 42   
## 97.911539 138.103984 181.167318 169.683762 163.941984 135.233095   
## 43 44 45 46 47 48   
## 115.136872 100.782428 95.040650 83.557094 77.815316 69.202650   
## 49 50 51 52 53 54   
## 307.486430 278.777541 258.681319 255.810430 252.939541 224.230651   
## 55 56 57 58 59 60   
## 212.747096 209.876207 186.909095 184.038207 204.134429 235.714207   
## 61 62 63 64 65 66   
## 264.423096 273.035763 273.035763 298.873763 284.519319 258.681319   
## 67 68 69 70 71 72   
## 227.101540 207.005318 195.521762 192.650873 178.296429 169.683762   
## 73 74 75 76 77 78   
## 287.390208 290.261097 304.615541 318.969986 327.582653 330.453542   
## 79 80 81 82 83 84   
## 333.324431 327.582653 324.711764 344.807986 376.387764 425.192876   
## 85 86 87 88 89 90   
## 465.385321 491.223321 505.577766 522.803099 508.448655 491.223321   
## 91 92 93 94 95 96   
## 476.868877 485.481543 505.577766 511.319544 511.319544 502.706877   
## 97 98 99 100 101 102   
## 502.706877 499.835988 496.965099 491.223321 488.352432 479.739766   
## 103 104 105 106 107 108   
## 482.610655 482.610655 482.610655 485.481543 491.223321 499.835988   
## 109 110 111 112 113 114   
## 537.157544 557.253767 583.091767 583.091767 571.608211 557.253767   
## 115 116 117 118 119 120   
## 525.673988 508.448655 505.577766 496.965099 488.352432 494.094210   
## 121 122 123 124 125 126   
## 109.395095 100.782428 89.298872 74.944427 74.944427 66.331761   
## 127 128 129 130 131 132   
## 54.848205 46.235538 43.364649 54.848205 92.169761 140.974873   
## 133 134 135 136 137 138   
## 178.296429 201.263540 221.359763 247.197763 244.326874 215.617985   
## 139 140 141 142 143 144   
## 186.909095 178.296429 166.812873 152.458428 152.458428 155.329317   
## 145 146 147 148 149 150   
## 324.711764 321.840875 310.357319 304.615541 304.615541 301.744652   
## 151 152 153 154 155 156   
## 304.615541 316.099097 324.711764 341.937097 359.162431 393.613098   
## 157 158 159 160 161 162   
## 428.063765 451.030876 459.643543 459.643543 453.901765 445.289099   
## 163 164 165 166 167 168   
## 422.321987 402.225765 390.742209 387.871320 387.871320 376.387764   
## 169 170 171 172 173 174   
## 244.326874 241.455985 229.972429 221.359763 215.617985 209.876207   
## 175 176 177 178 179 180   
## 207.005318 201.263540 192.650873 212.747096 290.261097 339.066208   
## 181 182 183 184 185 186   
## 379.258653 405.096654 442.418210 422.321987 422.321987 387.871320   
## 187 188 189 190 191 192   
## 359.162431 333.324431 327.582653 310.357319 298.873763 290.261097   
## 193 194 195 196 197 198   
## 227.101540 224.230651 209.876207 198.392651 189.779984 175.425540   
## 199 200 201 202 203 204   
## 178.296429 175.425540 184.038207 227.101540 284.519319 350.549764   
## 205 206 207 208 209 210   
## 367.775098 376.387764 390.742209 413.709320 410.838431 402.225765   
## 211 212 213 214 215 216   
## 387.871320 382.129542 373.516875 370.645987 367.775098 367.775098   
## 217 218 219 220 221 222   
## 422.321987 422.321987 439.547321 462.514432 476.868877 479.739766   
## 223 224 225 226 227 228   
## 476.868877 462.514432 453.901765 448.159987 456.772654 471.127099   
## 229 230 231 232 233 234   
## 514.190433 511.319544 540.028433 548.641100 551.511989 522.803099   
## 235 236 237 238 239 240   
## 485.481543 462.514432 442.418210 433.805543 422.321987 405.096654   
## 241 242 243 244 245 246   
## 413.709320 428.063765 430.934654 439.547321 430.934654 410.838431   
## 247 248 249 250 251 252   
## 399.354876 379.258653 367.775098 370.645987 376.387764 387.871320   
## 253 254 255 256 257 258   
## 407.967543 436.676432 462.514432 456.772654 439.547321 416.580209   
## 259 260 261 262 263 264   
## 393.613098 379.258653 367.775098 359.162431 356.291542 353.420653   
## 265 266 267 268 269 270   
## 49.106427 37.622871 31.881094 29.010205 17.526649 11.784871   
## 271 272 273 274 275 276   
## 14.655760 11.784871 17.526649 51.977316 120.878650 207.005318   
## 277 278 279 280 281 282   
## 244.326874 298.873763 304.615541 290.261097 273.035763 267.293985   
## 283 284 285 286 287 288   
## 270.164874 287.390208 293.131986 290.261097 287.390208 287.390208   
## 289 290 291 292 293 294   
## 186.909095 181.167318 178.296429 158.200206 132.362206 112.265983   
## 295 296 297 298 299 300   
## 103.653317 86.427983 89.298872 109.395095 192.650873 261.552207   
## 301 302 303 304 305 306   
## 304.615541 344.807986 359.162431 356.291542 367.775098 344.807986   
## 307 308 309 310 311 312   
## 307.486430 284.519319 267.293985 244.326874 224.230651 207.005318   
## 313 314 315 316 317 318   
## 264.423096 270.164874 238.585096 209.876207 175.425540 155.329317   
## 319 320 321 322 323 324   
## 112.265983 83.557094 63.460872 66.331761 92.169761 115.136872   
## 325 326 327 328 329 330   
## 163.941984 169.683762 192.650873 207.005318 215.617985 175.425540   
## 331 332 333 334 335 336   
## 146.716651 118.007761 92.169761 74.944427 57.719094 49.106427   
## 337 338 339 340 341 342   
## 379.258653 385.000431 390.742209 393.613098 390.742209 387.871320   
## 343 344 345 346 347 348   
## 382.129542 382.129542 382.129542 382.129542 387.871320 410.838431   
## 349 350 351 352 353 354   
## 419.451098 416.580209 399.354876 393.613098 344.807986 310.357319   
## 355 356 357 358 359 360   
## 278.777541 247.197763 209.876207 175.425540 158.200206 143.845762   
## 361 362 363 364 365 366   
## 138.103984 135.233095 135.233095 129.491317 120.878650 112.265983   
## 367 368 369 370 371 372   
## 95.040650 72.073539 66.331761 89.298872 138.103984 192.650873   
## 373 374 375 376 377 378   
## 224.230651 267.293985 273.035763 261.552207 232.843318 215.617985   
## 379 380 381 382 383 384   
## 204.134429 186.909095 172.554651 175.425540 169.683762 135.233095   
## 385 386 387 388 389 390   
## 241.455985 232.843318 218.488874 209.876207 207.005318 192.650873   
## 391 392 393 394 395 396   
## 186.909095 181.167318 181.167318 212.747096 298.873763 364.904209   
## 397 398 399 400 401 402   
## 399.354876 436.676432 456.772654 451.030876 430.934654 407.967543   
## 403 404 405 406 407 408   
## 376.387764 353.420653 341.937097 339.066208 318.969986 278.777541   
## 409 410 411 412 413 414   
## 258.681319 247.197763 227.101540 207.005318 186.909095 172.554651   
## 415 416 417 418 419 420   
## 158.200206 140.974873 132.362206 149.587539 212.747096 255.810430   
## 421 422 423 424 425 426   
## 298.873763 324.711764 333.324431 364.904209 373.516875 353.420653   
## 427 428 429 430 431 432   
## 327.582653 293.131986 275.906652 270.164874 258.681319 252.939541   
## 433 434 435 436 437 438   
## 468.256210 451.030876 442.418210 436.676432 413.709320 387.871320   
## 439 440 441 442 443 444   
## 370.645987 353.420653 336.195319 353.420653 359.162431 399.354876   
## 445 446 447 448 449 450   
## 451.030876 488.352432 534.286655 528.544877 505.577766 476.868877   
## 451 452 453 454 455 456   
## 436.676432 396.483987 367.775098 356.291542 339.066208 318.969986   
## 457 458 459 460 461 462   
## 482.610655 479.739766 485.481543 488.352432 491.223321 485.481543   
## 463 464 465 466 467 468   
## 485.481543 494.094210 499.835988 514.190433 522.803099 540.028433   
## 469 470 471 472 473 474   
## 554.382878 554.382878 499.835988 473.997988 456.772654 445.289099   
## 475 476 477 478 479 480   
## 439.547321 407.967543 413.709320 405.096654 402.225765 387.871320   
## 481 482 483 484 485 486   
## 350.549764 353.420653 362.033320 359.162431 353.420653 359.162431   
## 487 488 489 490 491 492   
## 350.549764 339.066208 347.678875 399.354876 482.610655 568.737322   
## 493 494 495 496 497 498   
## 643.380434 686.443768 749.603324 795.537547 789.795769 766.828658   
## 499 500 501 502 503 504   
## 735.248880 700.798213 654.863990 631.896879 629.025990 606.058878   
## 505 506 507 508 509 510   
## 554.382878 537.157544 517.061322 499.835988 482.610655 471.127099   
## 511 512 513 514 515 516   
## 456.772654 445.289099 439.547321 442.418210 462.514432 511.319544   
## 517 518 519 520 521 522   
## 554.382878 574.479100 623.284212 631.896879 651.993101 623.284212   
## 523 524 525 526 527 528   
## 568.737322 502.706877 459.643543 425.192876 402.225765 390.742209   
## 529 530 531 532 533 534   
## 416.580209 416.580209 410.838431 413.709320 410.838431 407.967543   
## 535 536 537 538 539 540   
## 405.096654 399.354876 410.838431 445.289099 485.481543 565.866433   
## 541 542 543 544 545 546   
## 611.800656 620.413323 617.542434 629.025990 634.767767 634.767767   
## 547 548 549 550 551 552   
## 611.800656 591.704434 580.220878 562.995544 542.899322 508.448655   
## 553 554 555 556 557 558   
## 436.676432 419.451098 413.709320 402.225765 416.580209 430.934654   
## 559 560 561 562 563 564   
## 416.580209 422.321987 445.289099 473.997988 525.673988 600.317100   
## 565 566 567 568 569 570   
## 672.089323 726.636213 763.957769 775.441325 752.474213 729.507102   
## 571 572 573 574 575 576   
## 686.443768 640.509545 606.058878 585.962656 560.124655 531.415766   
## 577 578 579 580 581 582   
## 442.418210 422.321987 402.225765 387.871320 373.516875 364.904209   
## 583 584 585 586 587 588   
## 359.162431 350.549764 373.516875 433.805543 508.448655 565.866433   
## 589 590 591 592 593 594   
## 623.284212 649.122212 683.572879 715.152657 729.507102 683.572879   
## 595 596 597 598 599 600   
## 640.509545 588.833545 562.995544 548.641100 537.157544 519.932211   
## 601 602 603 604 605 606   
## 382.129542 379.258653 376.387764 373.516875 373.516875 373.516875   
## 607 608 609 610 611 612   
## 376.387764 376.387764 382.129542 407.967543 439.547321 473.997988   
## 613 614 615 616 617 618   
## 473.997988 451.030876 390.742209 399.354876 405.096654 407.967543   
## 619 620 621 622 623 624   
## 393.613098 393.613098 385.000431 370.645987 370.645987 379.258653   
## 625 626 627 628 629 630   
## 471.127099 471.127099 479.739766 479.739766 482.610655 471.127099   
## 631 632 633 634 635 636   
## 456.772654 451.030876 465.385321 522.803099 542.899322 594.575323   
## 637 638 639 640 641 642   
## 640.509545 692.185546 700.798213 692.185546 689.314657 654.863990   
## 643 644 645 646 647 648   
## 594.575323 557.253767 540.028433 528.544877 522.803099 519.932211   
## 649 650 651 652 653 654   
## 540.028433 517.061322 491.223321 494.094210 494.094210 491.223321   
## 655 656 657 658 659 660   
## 499.835988 508.448655 519.932211 554.382878 623.284212 697.927324   
## 661 662 663 664 665 666   
## 755.345102 795.537547 835.729992 841.471770 844.342659 827.117325   
## 667 668 669 670 671 672   
## 795.537547 746.732435 706.539991 669.218435 643.380434 617.542434   
## 673 674 675 676 677 678   
## 720.894435 709.410879 706.539991 700.798213 703.669102 700.798213   
## 679 680 681 682 683 684   
## 697.927324 703.669102 718.023546 723.765324 766.828658 798.408436   
## 685 686 687 688 689 690   
## 858.697103 896.018659 916.114882 933.340215 930.469327 918.985771   
## 691 692 693 694 695 696   
## 884.535104 838.600881 784.053991 749.603324 718.023546 700.798213   
## 697 698 699 700 701 702   
## 818.504659 827.117325 824.246436 824.246436 821.375547 821.375547   
## 703 704 705 706 707 708   
## 818.504659 818.504659 832.859103 861.567992 878.793326 881.664215   
## 709 710 711 712 713 714   
## 898.889548 913.243993 939.081993 907.502215 901.760437 901.760437   
## 715 716 717 718 719 720   
## 878.793326 852.955326 832.859103 824.246436 795.537547 766.828658   
## 721 722 723 724 725 726   
## 608.929767 594.575323 594.575323 591.704434 588.833545 574.479100   
## 727 728 729 730 731 732   
## 551.511989 540.028433 528.544877 531.415766 540.028433 540.028433   
## 733 734 735 736 737 738   
## 545.770211 557.253767 565.866433 568.737322 568.737322 565.866433   
## 739 740 741 742 743 744   
## 554.382878 554.382878 545.770211 545.770211 542.899322 540.028433   
## 745 746 747 748 749 750   
## 419.451098 405.096654 399.354876 393.613098 385.000431 373.516875   
## 751 752 753 754 755 756   
## 362.033320 362.033320 419.451098 482.610655 517.061322 519.932211   
## 757 758 759 760 761 762   
## 534.286655 548.641100 562.995544 545.770211 511.319544 459.643543   
## 763 764 765 766 767 768   
## 422.321987 407.967543 413.709320 416.580209 413.709320 419.451098   
## 769 770 771 772 773 774   
## 663.476657 623.284212 614.671545 620.413323 614.671545 597.446211   
## 775 776 777 778 779 780   
## 574.479100 588.833545 640.509545 715.152657 786.924880 821.375547   
## 781 782 783 784 785 786   
## 858.697103 881.664215 873.051548 870.180659 861.567992 855.826215   
## 787 788 789 790 791 792   
## 838.600881 821.375547 801.279325 807.021103 804.150214 781.183103   
## 793 794 795 796 797 798   
## 761.086880 740.990658 723.765324 683.572879 631.896879 608.929767   
## 799 800 801 802 803 804   
## 585.962656 571.608211 557.253767 551.511989 557.253767 568.737322   
## 805 806 807 808 809 810   
## 583.091767 588.833545 591.704434 583.091767 591.704434 603.187989   
## 811 812 813 814 815 816   
## 611.800656 608.929767 606.058878 597.446211 588.833545 577.349989   
## 817 818 819 820 821 822   
## 611.800656 594.575323 577.349989 560.124655 554.382878 548.641100   
## 823 824 825 826 827 828   
## 534.286655 534.286655 585.962656 680.701990 758.215991 821.375547   
## 829 830 831 832 833 834   
## 852.955326 884.535104 904.631326 907.502215 896.018659 861.567992   
## 835 836 837 838 839 840   
## 827.117325 752.474213 695.056435 672.089323 651.993101 626.155101   
## 841 842 843 844 845 846   
## 617.542434 603.187989 588.833545 583.091767 574.479100 562.995544   
## 847 848 849 850 851 852   
## 554.382878 557.253767 611.800656 692.185546 758.215991 798.408436   
## 853 854 855 856 857 858   
## 847.213548 887.405993 924.727549 916.114882 904.631326 913.243993   
## 859 860 861 862 863 864   
## 884.535104 835.729992 781.183103 743.861547 720.894435 706.539991   
## 865 866 867 868 869 870   
## 692.185546 674.960212 672.089323 660.605768 663.476657 663.476657   
## 871 872 873 874 875 876   
## 657.734879 660.605768 709.410879 789.795769 858.697103 927.598438   
## 877 878 879 880 881 882   
## 982.145327 1030.950439 1079.755551 1085.497328 1074.013773 1071.142884   
## 883 884 885 886 887 888   
## 1045.304883 993.628883 941.952882 887.405993 858.697103 832.859103   
## 889 890 891 892 893 894   
## 792.666658 792.666658 778.312214 772.570436 763.957769 761.086880   
## 895 896 897 898 899 900   
## 755.345102 766.828658 804.150214 821.375547 804.150214 835.729992   
## 901 902 903 904 905 906   
## 878.793326 924.727549 930.469327 898.889548 884.535104 850.084437   
## 907 908 909 910 911 912   
## 798.408436 772.570436 718.023546 680.701990 669.218435 660.605768   
## 913 914 915 916 917 918   
## 674.960212 666.347546 657.734879 643.380434 629.025990 620.413323   
## 919 920 921 922 923 924   
## 620.413323 626.155101 669.218435 715.152657 775.441325 844.342659   
## 925 926 927 928 929 930   
## 896.018659 939.081993 973.532660 987.887105 1002.241550 962.049105   
## 931 932 933 934 935 936   
## 918.985771 873.051548 821.375547 786.924880 758.215991 738.119769   
## 937 938 939 940 941 942   
## 672.089323 657.734879 649.122212 640.509545 631.896879 614.671545   
## 943 944 945 946 947 948   
## 608.929767 620.413323 686.443768 763.957769 838.600881 898.889548   
## 949 950 951 952 953 954   
## 924.727549 964.919994 956.307327 990.757994 973.532660 964.919994   
## 955 956 957 958 959 960   
## 913.243993 855.826215 821.375547 798.408436 784.053991 772.570436   
## 961 962 963 964 965 966   
## 758.215991 749.603324 740.990658 746.732435 755.345102 761.086880   
## 967 968 969 970 971 972   
## 763.957769 769.699547 789.795769 838.600881 896.018659 918.985771   
## 973 974 975 976 977 978   
## 973.532660 1005.112439 1025.208661 1028.079550 1048.175772 1025.208661   
## 979 980 981 982 983 984   
## 996.499772 944.823771 910.373104 884.535104 864.438881 850.084437   
## 985 986 987 988 989 990   
## 901.760437 898.889548 881.664215 867.309770 864.438881 864.438881   
## 991 992 993 994 995 996   
## 850.084437 841.471770 838.600881 835.729992 844.342659 852.955326   
## 997 998 999 1000 1001 1002   
## 832.859103 829.988214 838.600881 824.246436 778.312214 735.248880   
## 1003 1004 1005 1006 1007 1008   
## 726.636213 697.927324 654.863990 637.638656 634.767767 608.929767   
## 1009 1010 1011 1012 1013 1014   
## 781.183103 769.699547 761.086880 758.215991 749.603324 743.861547   
## 1015 1016 1017 1018 1019 1020   
## 738.119769 743.861547 761.086880 792.666658 838.600881 916.114882   
## 1021 1022 1023 1024 1025 1026   
## 993.628883 1016.595994 1028.079550 1042.433995 1039.563106 1068.271995   
## 1027 1028 1029 1030 1031 1032   
## 1056.788439 1010.854216 936.211104 898.889548 870.180659 838.600881   
## 1033 1034 1035 1036 1037 1038   
## 807.021103 795.537547 781.183103 784.053991 778.312214 766.828658   
## 1039 1040 1041 1042 1043 1044   
## 761.086880 775.441325 821.375547 867.309770 910.373104 939.081993   
## 1045 1046 1047 1048 1049 1050   
## 953.436438 959.178216 973.532660 979.274438 967.790883 959.178216   
## 1051 1052 1053 1054 1055 1056   
## 930.469327 850.084437 789.795769 752.474213 715.152657 686.443768   
## 1057 1058 1059 1060 1061 1062   
## 855.826215 838.600881 827.117325 812.762881 798.408436 786.924880   
## 1063 1064 1065 1066 1067 1068   
## 778.312214 801.279325 864.438881 933.340215 987.887105 1048.175772   
## 1069 1070 1071 1072 1073 1074   
## 1091.239106 1134.302440 1154.398663 1171.623996 1163.011329 1154.398663   
## 1075 1076 1077 1078 1079 1080   
## 1125.689773 1108.464440 1091.239106 1071.142884 1053.917550 1048.175772   
## 1081 1082 1083 1084 1085 1086   
## 729.507102 706.539991 695.056435 683.572879 672.089323 660.605768   
## 1087 1088 1089 1090 1091 1092   
## 663.476657 695.056435 772.570436 841.471770 907.502215 953.436438   
## 1093 1094 1095 1096 1097 1098   
## 970.661771 1005.112439 1002.241550 1010.854216 1010.854216 1007.983327   
## 1099 1100 1101 1102 1103 1104   
## 985.016216 941.952882 910.373104 890.276882 870.180659 855.826215   
## 1105 1106 1107 1108 1109 1110   
## 936.211104 916.114882 898.889548 878.793326 861.567992 847.213548   
## 1111 1112 1113 1114 1115 1116   
## 844.342659 855.826215 921.856660 1005.112439 1068.271995 1119.947995   
## 1117 1118 1119 1120 1121 1122   
## 1148.656885 1185.978441 1191.720219 1191.720219 1185.978441 1165.882218   
## 1123 1124 1125 1126 1127 1128   
## 1122.818884 1065.401106 1007.983327 970.661771 950.565549 927.598438   
## 1129 1130 1131 1132 1133 1134   
## 924.727549 910.373104 901.760437 884.535104 867.309770 855.826215   
## 1135 1136 1137 1138 1139 1140   
## 847.213548 875.922437 953.436438 1010.854216 1068.271995 1105.593551   
## 1141 1142 1143 1144 1145 1146   
## 1145.785996 1157.269551 1165.882218 1168.753107 1163.011329 1142.915107   
## 1147 1148 1149 1150 1151 1152   
## 1114.206218 1068.271995 1025.208661 1005.112439 979.274438 964.919994   
## 1153 1154 1155 1156 1157 1158   
## 898.889548 884.535104 875.922437 864.438881 855.826215 847.213548   
## 1159 1160 1161 1162 1163 1164   
## 855.826215 870.180659 878.793326 924.727549 967.790883 1007.983327   
## 1165 1166 1167 1168 1169 1170   
## 1053.917550 1102.722662 1145.785996 1177.365774 1177.365774 1145.785996   
## 1171 1172 1173 1174 1175 1176   
## 1071.142884 1025.208661 990.757994 953.436438 936.211104 918.985771   
## 1177 1178 1179 1180 1181 1182   
## 916.114882 901.760437 887.405993 870.180659 861.567992 864.438881   
## 1183 1184 1185 1186 1187 1188   
## 861.567992 864.438881 881.664215 887.405993 916.114882 947.694660   
## 1189 1190 1191 1192 1193 1194   
## 1002.241550 1030.950439 1028.079550 1048.175772 1048.175772 1042.433995   
## 1195 1196 1197 1198 1199 1200   
## 1028.079550 1007.983327 990.757994 970.661771 953.436438 936.211104   
## 1201 1202 1203 1204 1205 1206   
## 918.985771 904.631326 887.405993 875.922437 875.922437 875.922437   
## 1207 1208 1209 1210 1211 1212   
## 870.180659 884.535104 930.469327 973.532660 1016.595994 1002.241550   
## 1213 1214 1215 1216 1217 1218   
## 1002.241550 1016.595994 1022.337772 973.532660 967.790883 930.469327   
## 1219 1220 1221 1222 1223 1224   
## 904.631326 893.147771 884.535104 881.664215 881.664215 858.697103   
## 1225 1226 1227 1228 1229 1230   
## 941.952882 933.340215 933.340215 933.340215 939.081993 918.985771   
## 1231 1232 1233 1234 1235 1236   
## 896.018659 893.147771 878.793326 884.535104 901.760437 924.727549   
## 1237 1238 1239 1240 1241 1242   
## 944.823771 990.757994 1019.466883 1042.433995 1053.917550 1042.433995   
## 1243 1244 1245 1246 1247 1248   
## 1025.208661 1013.725105 985.016216 973.532660 962.049105 956.307327   
## 1249 1250 1251 1252 1253 1254   
## 939.081993 927.598438 913.243993 875.922437 852.955326 835.729992   
## 1255 1256 1257 1258 1259 1260   
## 829.988214 858.697103 873.051548 913.243993 967.790883 993.628883   
## 1261 1262 1263 1264 1265 1266   
## 1036.692217 1062.530217 1094.109995 1102.722662 1096.980884 1085.497328   
## 1267 1268 1269 1270 1271 1272   
## 1079.755551 1033.821328 985.016216 950.565549 921.856660 898.889548   
## 1273 1274 1275 1276 1277 1278   
## 973.532660 973.532660 979.274438 979.274438 976.403549 970.661771   
## 1279 1280 1281 1282 1283 1284   
## 967.790883 987.887105 1030.950439 1048.175772 1062.530217 1096.980884   
## 1285 1286 1287 1288 1289 1290   
## 1111.335329 1117.077107 1114.206218 1094.109995 1059.659328 973.532660   
## 1291 1292 1293 1294 1295 1296   
## 927.598438 936.211104 939.081993 947.694660 956.307327 959.178216   
## 1297 1298 1299 1300 1301 1302   
## 964.919994 967.790883 973.532660 976.403549 976.403549 970.661771   
## 1303 1304 1305 1306 1307 1308   
## 964.919994 967.790883 973.532660 993.628883 1036.692217 1048.175772   
## 1309 1310 1311 1312 1313 1314   
## 1085.497328 1105.593551 1134.302440 1151.527774 1168.753107 1154.398663   
## 1315 1316 1317 1318 1319 1320   
## 1142.915107 1114.206218 1088.368217 1059.659328 1030.950439 1010.854216   
## 1321 1322 1323 1324 1325 1326   
## 1010.854216 999.370661 990.757994 985.016216 985.016216 982.145327   
## 1327 1328 1329 1330 1331 1332   
## 979.274438 985.016216 1002.241550 1005.112439 1016.595994 1022.337772   
## 1333 1334 1335 1336 1337 1338   
## 1074.013773 1088.368217 1111.335329 1108.464440 1099.851773 1099.851773   
## 1339 1340 1341 1342 1343 1344   
## 1079.755551 1053.917550 1019.466883 999.370661 982.145327 967.790883   
## 1345 1346 1347 1348 1349 1350   
## 921.856660 921.856660 924.727549 927.598438 927.598438 927.598438   
## 1351 1352 1353 1354 1355 1356   
## 930.469327 941.952882 959.178216 976.403549 985.016216 1002.241550   
## 1357 1358 1359 1360 1361 1362   
## 1028.079550 1059.659328 1068.271995 1074.013773 1091.239106 1096.980884   
## 1363 1364 1365 1366 1367 1368   
## 1088.368217 1074.013773 1062.530217 1056.788439 1053.917550 1051.046661   
## 1369 1370 1371 1372 1373 1374   
## 1042.433995 1042.433995 1039.563106 1039.563106 1039.563106 1039.563106   
## 1375 1376 1377 1378 1379 1380   
## 1039.563106 1042.433995 1056.788439 1071.142884 1091.239106 1117.077107   
## 1381 1382 1383 1384 1385 1386   
## 1142.915107 1160.140440 1174.494885 1180.236663 1160.140440 1160.140440   
## 1387 1388 1389 1390 1391 1392   
## 1151.527774 1145.785996 1131.431551 1119.947995 1114.206218 1108.464440   
## 1393 1394 1395 1396 1397 1398   
## 1142.915107 1119.947995 1105.593551 1091.239106 1079.755551 1068.271995   
## 1399 1400 1401 1402 1403 1404   
## 1062.530217 1076.884662 1137.173329 1211.816441 1260.621553 1306.555775   
## 1405 1406 1407 1408 1409 1410   
## 1355.360887 1389.811554 1409.907777 1412.778666 1421.391332 1415.649555   
## 1411 1412 1413 1414 1415 1416   
## 1378.327999 1326.651998 1292.201331 1263.492442 1246.267108 1231.912663   
## 1417 1418 1419 1420 1421 1422   
## 1214.687330 1197.461996 1188.849330 1185.978441 1177.365774 1177.365774   
## 1423 1424 1425 1426 1427 1428   
## 1174.494885 1177.365774 1197.461996 1226.170886 1249.137997 1274.975997   
## 1429 1430 1431 1432 1433 1434   
## 1315.168442 1306.555775 1320.910220 1352.489998 1335.264665 1335.264665   
## 1435 1436 1437 1438 1439 1440   
## 1309.426664 1272.105108 1240.525330 1220.429108 1206.074663 1197.461996   
## 1441 1442 1443 1444 1445 1446   
## 1174.494885 1154.398663 1140.044218 1122.818884 1102.722662 1094.109995   
## 1447 1448 1449 1450 1451 1452   
## 1088.368217 1105.593551 1163.011329 1211.816441 1254.879775 1289.330442   
## 1453 1454 1455 1456 1457 1458   
## 1300.813998 1332.393776 1349.619109 1355.360887 1381.198887 1352.489998   
## 1459 1460 1461 1462 1463 1464   
## 1384.069776 1346.748220 1309.426664 1286.459553 1269.234219 1246.267108   
## 1465 1466 1467 1468 1469 1470   
## 1252.008886 1240.525330 1240.525330 1229.041775 1229.041775 1217.558219   
## 1471 1472 1473 1474 1475 1476   
## 1208.945552 1211.816441 1243.396219 1280.717775 1323.781109 1349.619109   
## 1477 1478 1479 1480 1481 1482   
## 1386.940665 1389.811554 1407.036888 1386.940665 1424.262221 1409.907777   
## 1483 1484 1485 1486 1487 1488   
## 1389.811554 1346.748220 1309.426664 1292.201331 1280.717775 1266.363331   
## 1489 1490 1491 1492 1493 1494   
## 1257.750664 1246.267108 1240.525330 1231.912663 1226.170886 1211.816441   
## 1495 1496 1497 1498 1499 1500   
## 1211.816441 1220.429108 1249.137997 1283.588664 1320.910220 1361.102665   
## 1501 1502 1503 1504 1505 1506   
## 1392.682443 1392.682443 1409.907777 1409.907777 1389.811554 1363.973554   
## 1507 1508 1509 1510 1511 1512   
## 1323.781109 1297.943109 1260.621553 1231.912663 1214.687330 1203.203774   
## 1513 1514 1515 1516 1517 1518   
## 1171.623996 1154.398663 1148.656885 1148.656885 1140.044218 1137.173329   
## 1519 1520 1521 1522 1523 1524   
## 1128.560662 1145.785996 1157.269551 1203.203774 1243.396219 1274.975997   
## 1525 1526 1527 1528 1529 1530   
## 1315.168442 1323.781109 1300.813998 1326.651998 1343.877331 1341.006443   
## 1531 1532 1533 1534 1535 1536   
## 1326.651998 1272.105108 1226.170886 1188.849330 1183.107552 1171.623996   
## 1537 1538 1539 1540 1541 1542   
## 1165.882218 1163.011329 1154.398663 1148.656885 1142.915107 1134.302440   
## 1543 1544 1545 1546 1547 1548   
## 1131.431551 1137.173329 1157.269551 1180.236663 1208.945552 1220.429108   
## 1549 1550 1551 1552 1553 1554   
## 1229.041775 1188.849330 1220.429108 1231.912663 1243.396219 1163.011329   
## 1555 1556 1557 1558 1559 1560   
## 1142.915107 1145.785996 1137.173329 1134.302440 1125.689773 1119.947995   
## 1561 1562 1563 1564 1565 1566   
## 1191.720219 1168.753107 1154.398663 1137.173329 1122.818884 1108.464440   
## 1567 1568 1569 1570 1571 1572   
## 1096.980884 1102.722662 1140.044218 1188.849330 1234.783552 1240.525330   
## 1573 1574 1575 1576 1577 1578   
## 1289.330442 1315.168442 1326.651998 1341.006443 1358.231776 1363.973554   
## 1579 1580 1581 1582 1583 1584   
## 1323.781109 1306.555775 1277.846886 1243.396219 1234.783552 1217.558219   
## 1585 1586 1587 1588 1589 1590   
## 1194.591107 1180.236663 1168.753107 1154.398663 1142.915107 1131.431551   
## 1591 1592 1593 1594 1595 1596   
## 1117.077107 1122.818884 1157.269551 1188.849330 1243.396219 1277.846886   
## 1597 1598 1599 1600 1601 1602   
## 1306.555775 1312.297553 1341.006443 1338.135554 1315.168442 1303.684887   
## 1603 1604 1605 1606 1607 1608   
## 1295.072220 1254.879775 1180.236663 1165.882218 1163.011329 1160.140440   
## 1609 1610 1611 1612 1613 1614   
## 1151.527774 1145.785996 1142.915107 1137.173329 1134.302440 1140.044218   
## 1615 1616 1617 1618 1619 1620   
## 1125.689773 1128.560662 1145.785996 1214.687330 1252.008886 1297.943109   
## 1621 1622 1623 1624 1625 1626   
## 1326.651998 1343.877331 1349.619109 1341.006443 1326.651998 1306.555775   
## 1627 1628 1629 1630 1631 1632   
## 1266.363331 1191.720219 1137.173329 1105.593551 1079.755551 1053.917550   
## 1633 1634 1635 1636 1637 1638   
## 1056.788439 1042.433995 1030.950439 1028.079550 1013.725105 1007.983327   
## 1639 1640 1641 1642 1643 1644   
## 1016.595994 1033.821328 1074.013773 1140.044218 1203.203774 1257.750664   
## 1645 1646 1647 1648 1649 1650   
## 1309.426664 1349.619109 1375.457110 1392.682443 1409.907777 1381.198887   
## 1651 1652 1653 1654 1655 1656   
## 1352.489998 1346.748220 1289.330442 1286.459553 1257.750664 1231.912663   
## 1657 1658 1659 1660 1661 1662   
## 990.757994 982.145327 970.661771 956.307327 944.823771 930.469327   
## 1663 1664 1665 1666 1667 1668   
## 921.856660 930.469327 970.661771 1030.950439 1091.239106 1140.044218   
## 1669 1670 1671 1672 1673 1674   
## 1165.882218 1134.302440 1076.884662 1033.821328 1010.854216 1010.854216   
## 1675 1676 1677 1678 1679 1680   
## 1007.983327 987.887105 979.274438 982.145327 944.823771 933.340215   
## 1681 1682 1683 1684 1685 1686   
## 1045.304883 1051.046661 1045.304883 1028.079550 1022.337772 1025.208661   
## 1687 1688 1689 1690 1691 1692   
## 1030.950439 1033.821328 1042.433995 1053.917550 1085.497328 1088.368217   
## 1693 1694 1695 1696 1697 1698   
## 1111.335329 1145.785996 1125.689773 1117.077107 1128.560662 1105.593551   
## 1699 1700 1701 1702 1703 1704   
## 1085.497328 1056.788439 1051.046661 1048.175772 1042.433995 1033.821328   
## 1705 1706 1707 1708 1709 1710   
## 1022.337772 1016.595994 1007.983327 1005.112439 1002.241550 999.370661   
## 1711 1712 1713 1714 1715 1716   
## 993.628883 993.628883 999.370661 1028.079550 1068.271995 1102.722662   
## 1717 1718 1719 1720 1721 1722   
## 1134.302440 1160.140440 1177.365774 1206.074663 1188.849330 1171.623996   
## 1723 1724 1725 1726 1727 1728   
## 1151.527774 1096.980884 1065.401106 1039.563106 1028.079550 1019.466883   
## 1729 1730 1731 1732 1733 1734   
## 1045.304883 1039.563106 1033.821328 1030.950439 1025.208661 1022.337772   
## 1735 1736 1737 1738 1739 1740   
## 1019.466883 1022.337772 1030.950439 1013.725105 1005.112439 1016.595994   
## 1741 1742 1743 1744 1745 1746   
## 1022.337772 1051.046661 1071.142884 1074.013773 1068.271995 987.887105   
## 1747 1748 1749 1750 1751 1752   
## 964.919994 959.178216 953.436438 956.307327 959.178216 959.178216   
## 1753 1754 1755 1756 1757 1758   
## 950.565549 947.694660 936.211104 927.598438 921.856660 916.114882   
## 1759 1760 1761 1762 1763 1764   
## 910.373104 913.243993 924.727549 956.307327 999.370661 1039.563106   
## 1765 1766 1767 1768 1769 1770   
## 1076.884662 1096.980884 1122.818884 1102.722662 1114.206218 1128.560662   
## 1771 1772 1773 1774 1775 1776   
## 1105.593551 1056.788439 1016.595994 993.628883 973.532660 959.178216   
## 1777 1778 1779 1780 1781 1782   
## 913.243993 896.018659 878.793326 858.697103 844.342659 829.988214   
## 1783 1784 1785 1786 1787 1788   
## 824.246436 821.375547 847.213548 904.631326 964.919994 1016.595994   
## 1789 1790 1791 1792 1793 1794   
## 1039.563106 1074.013773 1076.884662 1082.626439 1068.271995 1065.401106   
## 1795 1796 1797 1798 1799 1800   
## 1045.304883 1016.595994 993.628883 973.532660 956.307327 936.211104   
## 1801 1802 1803 1804 1805 1806   
## 990.757994 990.757994 990.757994 985.016216 982.145327 973.532660   
## 1807 1808 1809 1810 1811 1812   
## 967.790883 964.919994 964.919994 976.403549 993.628883 1007.983327   
## 1813 1814 1815 1816 1817 1818   
## 1016.595994 1013.725105 1007.983327 1002.241550 982.145327 967.790883   
## 1819 1820 1821 1822 1823 1824   
## 953.436438 944.823771 941.952882 944.823771 950.565549 947.694660   
## 1825 1826 1827 1828 1829 1830   
## 878.793326 881.664215 884.535104 867.309770 850.084437 847.213548   
## 1831 1832 1833 1834 1835 1836   
## 841.471770 841.471770 847.213548 858.697103 864.438881 878.793326   
## 1837 1838 1839 1840 1841 1842   
## 884.535104 904.631326 950.565549 962.049105 953.436438 944.823771   
## 1843 1844 1845 1846 1847 1848   
## 936.211104 918.985771 910.373104 907.502215 901.760437 896.018659   
## 1849 1850 1851 1852 1853 1854   
## 829.988214 807.021103 795.537547 789.795769 801.279325 804.150214   
## 1855 1856 1857 1858 1859 1860   
## 809.891992 809.891992 824.246436 864.438881 890.276882 913.243993   
## 1861 1862 1863 1864 1865 1866   
## 933.340215 987.887105 1030.950439 1053.917550 1042.433995 1028.079550   
## 1867 1868 1869 1870 1871 1872   
## 976.403549 939.081993 916.114882 884.535104 867.309770 844.342659   
## 1873 1874 1875 1876 1877 1878   
## 689.314657 672.089323 654.863990 646.251323 634.767767 617.542434   
## 1879 1880 1881 1882 1883 1884   
## 608.929767 600.317100 620.413323 686.443768 766.828658 850.084437   
## 1885 1886 1887 1888 1889 1890   
## 887.405993 916.114882 916.114882 927.598438 918.985771 910.373104   
## 1891 1892 1893 1894 1895 1896   
## 855.826215 807.021103 781.183103 769.699547 749.603324 723.765324   
## 1897 1898 1899 1900 1901 1902   
## 597.446211 580.220878 565.866433 554.382878 540.028433 528.544877   
## 1903 1904 1905 1906 1907 1908   
## 525.673988 514.190433 548.641100 623.284212 697.927324 772.570436   
## 1909 1910 1911 1912 1913 1914   
## 812.762881 861.567992 890.276882 904.631326 893.147771 873.051548   
## 1915 1916 1917 1918 1919 1920   
## 809.891992 761.086880 732.377991 697.927324 672.089323 651.993101   
## 1921 1922 1923 1924 1925 1926   
## 663.476657 643.380434 637.638656 629.025990 623.284212 606.058878   
## 1927 1928 1929 1930 1931 1932   
## 600.317100 591.704434 614.671545 677.831101 755.345102 815.633770   
## 1933 1934 1935 1936 1937 1938   
## 858.697103 893.147771 927.598438 916.114882 901.760437 878.793326   
## 1939 1940 1941 1942 1943 1944   
## 832.859103 784.053991 752.474213 720.894435 700.798213 680.701990   
## 1945 1946 1947 1948 1949 1950   
## 646.251323 629.025990 608.929767 606.058878 594.575323 580.220878   
## 1951 1952 1953 1954 1955 1956   
## 568.737322 560.124655 568.737322 626.155101 709.410879 786.924880   
## 1957 1958 1959 1960 1961 1962   
## 818.504659 829.988214 841.471770 852.955326 858.697103 824.246436   
## 1963 1964 1965 1966 1967 1968   
## 769.699547 735.248880 712.281768 683.572879 669.218435 651.993101   
## 1969 1970 1971 1972 1973 1974   
## 433.805543 422.321987 396.483987 385.000431 382.129542 367.775098   
## 1975 1976 1977 1978 1979 1980   
## 359.162431 373.516875 376.387764 442.418210 511.319544 551.511989   
## 1981 1982 1983 1984 1985 1986   
## 580.220878 617.542434 643.380434 649.122212 651.993101 620.413323   
## 1987 1988 1989 1990 1991 1992   
## 577.349989 565.866433 554.382878 554.382878 554.382878 557.253767   
## 1993 1994 1995 1996 1997 1998   
## 540.028433 525.673988 519.932211 496.965099 479.739766 453.901765   
## 1999 2000 2001 2002 2003 2004   
## 439.547321 428.063765 442.418210 482.610655 542.899322 585.962656   
## 2005 2006 2007 2008 2009 2010   
## 623.284212 657.734879 677.831101 680.701990 674.960212 660.605768   
## 2011 2012 2013 2014 2015 2016   
## 606.058878 577.349989 554.382878 554.382878 531.415766 519.932211   
## 2017 2018 2019 2020 2021 2022   
## 542.899322 525.673988 511.319544 499.835988 485.481543 473.997988   
## 2023 2024 2025 2026 2027 2028   
## 465.385321 453.901765 471.127099 537.157544 623.284212 703.669102   
## 2029 2030 2031 2032 2033 2034   
## 763.957769 809.891992 852.955326 870.180659 875.922437 824.246436   
## 2035 2036 2037 2038 2039 2040   
## 755.345102 703.669102 672.089323 646.251323 623.284212 614.671545   
## 2041 2042 2043 2044 2045 2046   
## 588.833545 580.220878 554.382878 540.028433 517.061322 511.319544   
## 2047 2048 2049 2050 2051 2052   
## 494.094210 485.481543 491.223321 562.995544 651.993101 752.474213   
## 2053 2054 2055 2056 2057 2058   
## 809.891992 847.213548 855.826215 867.309770 873.051548 815.633770   
## 2059 2060 2061 2062 2063 2064   
## 743.861547 720.894435 683.572879 654.863990 623.284212 600.317100   
## 2065 2066 2067 2068 2069 2070   
## 623.284212 617.542434 614.671545 606.058878 597.446211 588.833545   
## 2071 2072 2073 2074 2075 2076   
## 580.220878 574.479100 585.962656 626.155101 672.089323 709.410879   
## 2077 2078 2079 2080 2081 2082   
## 752.474213 798.408436 835.729992 804.150214 812.762881 775.441325   
## 2083 2084 2085 2086 2087 2088   
## 738.119769 718.023546 715.152657 709.410879 689.314657 686.443768   
## 2089 2090 2091 2092 2093 2094   
## 677.831101 677.831101 680.701990 646.251323 640.509545 654.863990   
## 2095 2096 2097 2098 2099 2100   
## 651.993101 646.251323 660.605768 657.734879 677.831101 692.185546   
## 2101 2102 2103 2104 2105 2106   
## 715.152657 752.474213 781.183103 778.312214 772.570436 763.957769   
## 2107 2108 2109 2110 2111 2112   
## 749.603324 732.377991 718.023546 723.765324 715.152657 706.539991   
## 2113 2114 2115 2116 2117 2118   
## 491.223321 479.739766 473.997988 468.256210 451.030876 451.030876   
## 2119 2120 2121 2122 2123 2124   
## 445.289099 433.805543 430.934654 485.481543 574.479100 631.896879   
## 2125 2126 2127 2128 2129 2130   
## 689.314657 718.023546 743.861547 732.377991 706.539991 674.960212   
## 2131 2132 2133 2134 2135 2136   
## 640.509545 617.542434 594.575323 580.220878 554.382878 534.286655   
## 2137 2138 2139 2140 2141 2142   
## 439.547321 428.063765 419.451098 407.967543 407.967543 402.225765   
## 2143 2144 2145 2146 2147 2148   
## 390.742209 396.483987 410.838431 436.676432 473.997988 554.382878   
## 2149 2150 2151 2152 2153 2154   
## 600.317100 620.413323 626.155101 611.800656 606.058878 585.962656   
## 2155 2156 2157 2158 2159 2160   
## 577.349989 562.995544 548.641100 540.028433 531.415766 537.157544   
## 2161 2162 2163 2164 2165 2166   
## 542.899322 534.286655 534.286655 522.803099 528.544877 531.415766   
## 2167 2168 2169 2170 2171 2172   
## 528.544877 522.803099 519.932211 519.932211 534.286655 548.641100   
## 2173 2174 2175 2176 2177 2178   
## 557.253767 565.866433 528.544877 499.835988 511.319544 499.835988   
## 2179 2180 2181 2182 2183 2184   
## 491.223321 482.610655 482.610655 468.256210 468.256210 448.159987   
## 2185 2186 2187 2188 2189 2190   
## 430.934654 425.192876 416.580209 405.096654 399.354876 396.483987   
## 2191 2192 2193 2194 2195 2196   
## 387.871320 385.000431 393.613098 425.192876 482.610655 542.899322   
## 2197 2198 2199 2200 2201 2202   
## 597.446211 672.089323 703.669102 715.152657 715.152657 663.476657   
## 2203 2204 2205 2206 2207 2208   
## 608.929767 577.349989 545.770211 528.544877 514.190433 491.223321

test.pred <- data.frame(test.pred)  
  
write.csv(test.pred,"testing\_predictions\_4407409.csv")

Running Permutation Test on each of the predictors using Random Forests in order to find out which predictors are most significant and how significant they are.

The variables are Day,Month,Year,Hour,Temperature,Humidity,Wind,Solar,Rainfall,Snowfall,Seasons,Holiday,Functioning, Dew, Visibility

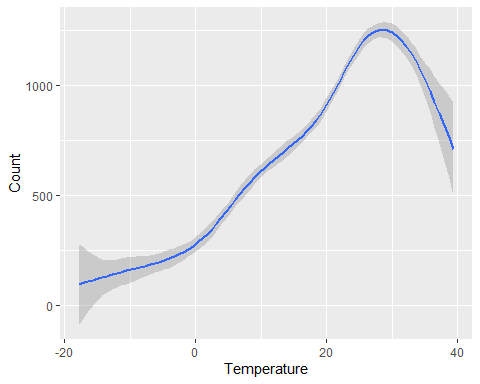
mse.perm <- numeric()   
set.seed(1)   
for (i in 1:15)   
{   
 if(i == 1)   
 {   
 shuffleTrain = train  
 shuffleTrain$Day = sample(train$Day,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 2)  
 {   
 shuffleTrain = train  
 shuffleTrain$Month = sample(train$Month,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 3 )  
 {   
 shuffleTrain = train  
 shuffleTrain$Year = sample(train$Year,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 4)  
 {   
 shuffleTrain = train  
 shuffleTrain$Hour = sample(train$Hour,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 5)  
 {   
 shuffleTrain = train  
 shuffleTrain$Temperature = sample(train$Temperature,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 6)  
 {   
 shuffleTrain = train  
 shuffleTrain$Humidity = sample(train$Humidity,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 7)  
 {   
 shuffleTrain = train  
 shuffleTrain$Wind = sample(train$Wind,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 8)  
 {   
 shuffleTrain = train  
 shuffleTrain$Solar = sample(train$Solar,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 9)  
 {   
 shuffleTrain = train  
 shuffleTrain$Rainfall = sample(train$Rainfall,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 10)  
 {   
 shuffleTrain = train  
 shuffleTrain$Snowfall = sample(train$Snowfall,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 11)   
 {   
 shuffleTrain = train  
 shuffleTrain$Seasons = sample(train$Seasons,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 12)   
 {   
 shuffleTrain = train  
 shuffleTrain$Holiday = sample(train$Holiday,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 13)   
 {   
 shuffleTrain = train  
 shuffleTrain$Functioning = sample(train$Functioning,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else if(i == 14)   
 {   
 shuffleTrain = train  
 shuffleTrain$Dew = sample(train$Dew,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }   
 else   
 {   
 shuffleTrain = train  
 shuffleTrain$Visibility = sample(train$Visibility,replace = FALSE)   
 randfor.mod <- randomForest(Count~.,data=shuffleTrain,mtry = 8,importance = TRUE)   
   
 #Calculating test MSE   
 randfor.pred <- predict(randfor.mod,test)   
 MSE <- mean ((test$Count - randfor.pred)^2)   
 mse.perm[i] <- MSE  
 }  
}   
plot(mse.perm,type="b",axes=F,ann=F,ylim=c(0,max(mse.perm)+1))  
axis(1,at=1:15,lab=names(train)[-1])  
axis(2,at=seq(0,max(mse.perm)+1,0.25),las=1)  
box()

Analyzing the trends between Count and most significant variables-Temperature,Hour,Seasons

Temperature

ggplot(data = bikeData\_new) +   
 geom\_smooth(mapping = aes(x = Temperature, y = Count))

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



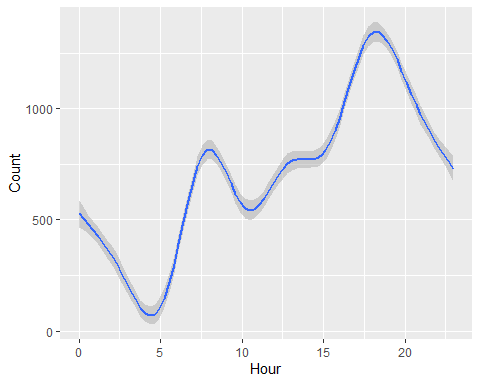
range(bikeData\_new$Temperature)

## [1] -17.8 39.4

Hour

ggplot(data = bikeData\_new) +   
 geom\_smooth(mapping = aes(x = Hour, y = Count))

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



Continuing Data Exploration to better understand how season affects the bike counts

bikeData\_new %>%   
 group\_by(Seasons) %>%   
 summarise()

## # A tibble: 4 x 1  
## Seasons  
## <chr>   
## 1 Autumn   
## 2 Spring   
## 3 Summer   
## 4 Winter

#Calculating the total counts for each season   
  
  
##Winter  
only\_Winter <- bikeData\_new %>%   
 select(Count,Seasons) %>%   
 filter(Seasons == "Winter")   
  
winter\_sum = sum(only\_Winter$Count)   
winter\_sum

## [1] 389211

##Summer  
only\_Summer <- bikeData\_new %>%   
 select(Count,Seasons) %>%   
 filter(Seasons == "Summer")   
  
summer\_sum = sum(only\_Summer$Count)   
summer\_sum

## [1] 1634082

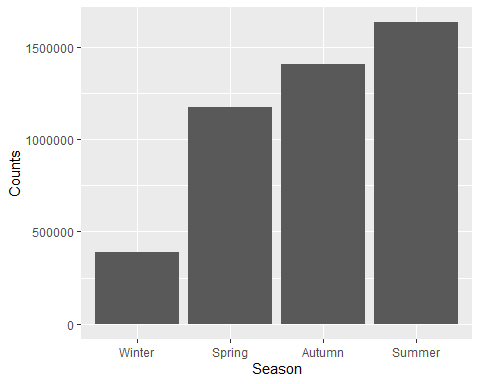
##Spring  
only\_Spring <- bikeData\_new %>%   
 select(Count,Seasons) %>%   
 filter(Seasons == "Spring")   
  
spring\_sum = sum(only\_Spring$Count)   
spring\_sum

## [1] 1175208

##Fall   
only\_Autumn <- bikeData\_new %>%   
 select(Count,Seasons) %>%   
 filter(Seasons == "Autumn")   
  
autumn\_sum = sum(only\_Autumn$Count)   
autumn\_sum

## [1] 1406891

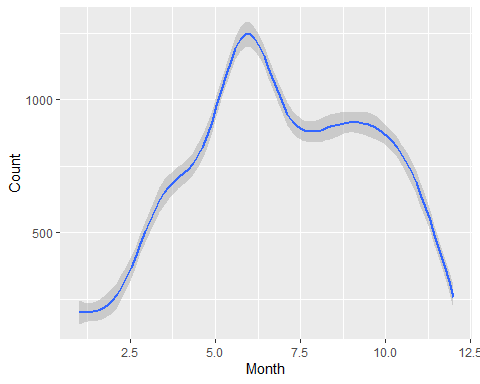
Seasons\_x <- c("Winter","Summer","Spring","Autumn")   
Counts\_y <- c(winter\_sum,summer\_sum,spring\_sum,autumn\_sum)   
  
my\_data <- data.frame(Seasons\_x,Counts\_y)   
ggplot(data = my\_data) +   
 geom\_bar(mapping = aes(x=reorder(Seasons\_x,Counts\_y),y=Counts\_y),stat="identity") +   
 labs(x = "Season",y = "Counts")



Analyzing the trends between Count and other variables(not considered the most significant)

ggplot(data = bikeData\_new) +   
 geom\_smooth(mapping = aes(x = Month, y = Count))

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



ggplot(data = bikeData\_new) +   
 geom\_bar(mapping = aes(x = Year, y = Count),stat = "identity")

