Final Project MSCA 31007 Statistical Analysis

By: Adham Suliman

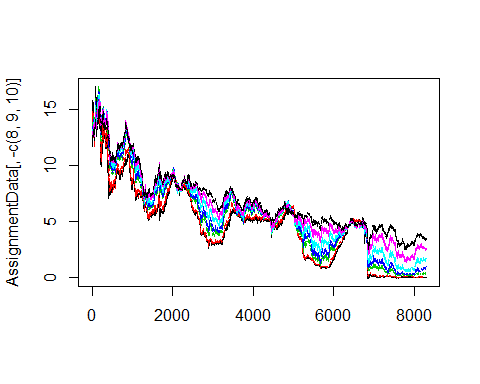
June 7, 2018

Step 1:

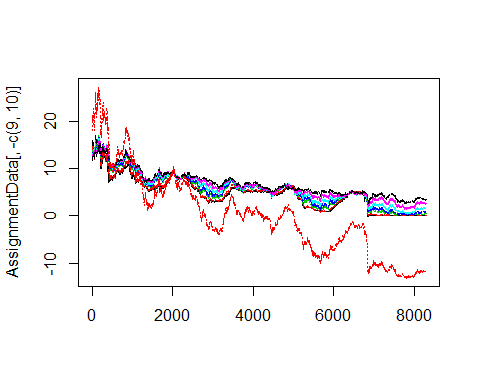
AssignmentData<- read.csv(file=paste("C:/Users/u353822/Documents/R/Statistical Analysis","Statistical Analysis Project.csv",sep="/"),  
 row.names=1,header=TRUE,sep=",")  
head(AssignmentData)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR Output1 Easing Tightening  
## 1/5/1981 13.52 13.09 12.289 12.28 12.294 12.152 11.672 18.01553 NA NA  
## 1/6/1981 13.58 13.16 12.429 12.31 12.214 12.112 11.672 18.09140 NA NA  
## 1/7/1981 14.50 13.90 12.929 12.78 12.614 12.382 11.892 19.44731 NA NA  
## 1/8/1981 14.76 14.00 13.099 12.95 12.684 12.352 11.912 19.74851 NA NA  
## 1/9/1981 15.20 14.30 13.539 13.28 12.884 12.572 12.132 20.57204 NA NA  
## 1/12/1981 15.22 14.23 13.179 12.94 12.714 12.452 12.082 20.14218 NA NA

matplot(AssignmentData[,-c(8,9,10)],type='l')



matplot(AssignmentData[,-c(9,10)],type='l')



The first visualization looks at all of the data with relation to the predictors(bonds). All of the bonds start at a yield of around 15%, and they follow the same general trend.There are periods of positive and negative slopes, but the general trend seems to be going towards a percent yield of below 5%. The second visualization has all of the predictors found in the first visualization, but it also includes Output 1. Output 1 seems to follow the same general trend of the predictors, but it begins to deviate from the predictors at around the index of 3000. The output 1 even falls below 0% which would be an interesting concept for a bond.

Step 2: Output1~USGG3M

Input1.Linear.Model<-lm(Output1~USGG3M,AssignmentData)  
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input1.Linear.Model)$sigma^2)

## Total.Variance Unexplained.Variance   
## 76.804438 2.857058

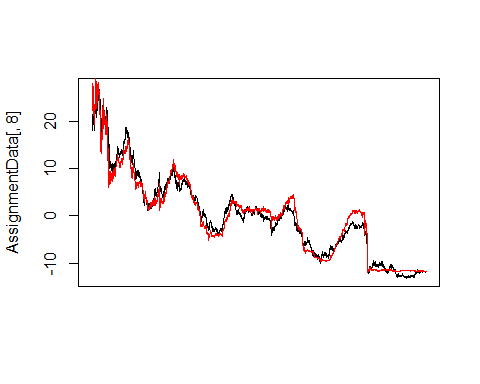
summary(Input1.Linear.Model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -11.723184 0.031369514 -373.7126 0  
## USGG3M 2.507561 0.005410479 463.4638 0

c(R2=(summary(Input1.Linear.Model)$r.squared),R2.adj=(summary(Input1.Linear.Model)$adj.r.squared))

## R2 R2.adj   
## 0.9628054 0.9628009

matplot(AssignmentData[,8],type="l",xaxt="n")  
 lines(Input1.Linear.Model$fitted.values,col="red")



The Linear Regresseion model here takes the Output against the USGG3M. The unexplained variance amounts to 2.8 out of 76.8 which shows that these two data sets seem to be fairly similar. All of the models below will have a total variance of 76.8 which is stemming from Output 1. The unexplained variance will be reduced to the extent that the predictor is capable of explaining the variance. From the summary of the coefficients, the Pr(>t) = 0 shows that there is significant evidence that the coefficient and intercept are not equal to 0. As expected, R2 and R2.adj are equal because there is only one predictor in the model. R2.adj is used to penalize correlation when more than one predictor is being utilized. R2 is not reduced as more predictors are utilized. R2 and R2.adj will be expected to be the same for all models utilizing one predictor.

Output1~USGG6M

Input2.Linear.Model<-lm(Output1~USGG6M,AssignmentData)  
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input2.Linear.Model)$sigma^2)

## Total.Variance Unexplained.Variance   
## 76.804438 1.967321

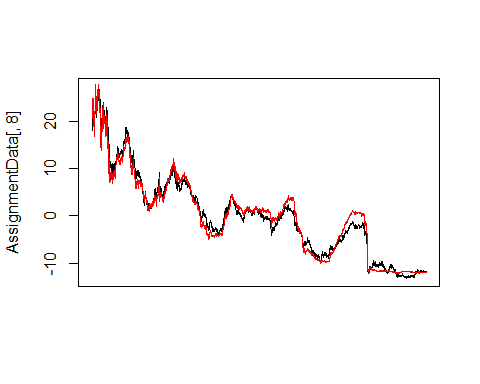
summary(Input2.Linear.Model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -12.097528 0.02646897 -457.0457 0  
## USGG6M 2.497235 0.00444452 561.8683 0

c(R2=(summary(Input2.Linear.Model)$r.squared),R2.adj=(summary(Input2.Linear.Model)$adj.r.squared))

## R2 R2.adj   
## 0.9743884 0.9743853

matplot(AssignmentData[,8],type="l",xaxt="n")  
lines(Input2.Linear.Model$fitted.values,col="red")



The Linear Regresseion model here takes the Output against the USGG6M. The unexplained variance amounts to 1.9 out of 76.8 which shows that these two data sets seem to be fairly similar. From the summary of the coefficients, the Pr(>t) = 0 shows that there is significant evidence that the coefficient and intercept are not equal to 0.

Output1~USGG2YR

Input3.Linear.Model<-lm(Output1~USGG2YR,AssignmentData)  
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input3.Linear.Model)$sigma^2)

## Total.Variance Unexplained.Variance   
## 76.8044379 0.2588092

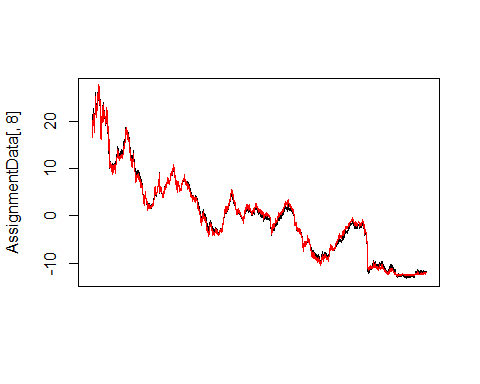
summary(Input3.Linear.Model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -13.055775 0.010031274 -1301.507 0  
## USGG2YR 2.400449 0.001532178 1566.691 0

c(R2=(summary(Input3.Linear.Model)$r.squared),R2.adj=(summary(Input3.Linear.Model)$adj.r.squared))

## R2 R2.adj   
## 0.9966307 0.9966303

matplot(AssignmentData[,8],type="l",xaxt="n")  
lines(Input3.Linear.Model$fitted.values,col="red")



The Linear Regresseion model here takes the Output against the USGG2YR. The unexplained variance amounts to .25 out of 76.8 which shows that these two data sets seem to be fairly similar. From the summary of the coefficients, the Pr(>t) = 0 shows that there is significant evidence that the coefficient and intercept are not equal to 0.

Output1~USGG3YR

Input4.Linear.Model<-lm(Output1~USGG3YR,AssignmentData)  
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input4.Linear.Model)$sigma^2)

## Total.Variance Unexplained.Variance   
## 76.804438 0.159657

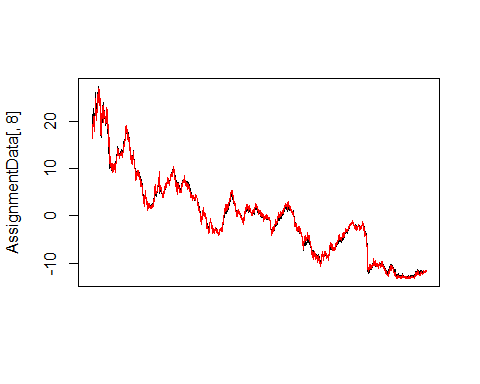
summary(Input4.Linear.Model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -13.861618 0.008213688 -1687.624 0  
## USGG3YR 2.455793 0.001230358 1995.999 0

c(R2=(summary(Input4.Linear.Model)$r.squared),R2.adj=(summary(Input4.Linear.Model)$adj.r.squared))

## R2 R2.adj   
## 0.9979215 0.9979213

matplot(AssignmentData[,8],type="l",xaxt="n")  
lines(Input4.Linear.Model$fitted.values,col="red")



The Linear Regresseion model here takes the Output against the USGG3YR. The unexplained variance amounts to .16 out of 76.8 which shows that these two data sets seem to be fairly similar. From the summary of the coefficients, the Pr(>t) = 0 shows that there is significant evidence that the coefficient and intercept are not equal to 0.

Output1~USGG5YR

Input5.Linear.Model<-lm(Output1~USGG5YR,AssignmentData)  
head(AssignmentData,3)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR Output1 Easing Tightening  
## 1/5/1981 13.52 13.09 12.289 12.28 12.294 12.152 11.672 18.01553 NA NA  
## 1/6/1981 13.58 13.16 12.429 12.31 12.214 12.112 11.672 18.09140 NA NA  
## 1/7/1981 14.50 13.90 12.929 12.78 12.614 12.382 11.892 19.44731 NA NA

c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input5.Linear.Model)$sigma^2)

## Total.Variance Unexplained.Variance   
## 76.8044379 0.6382572

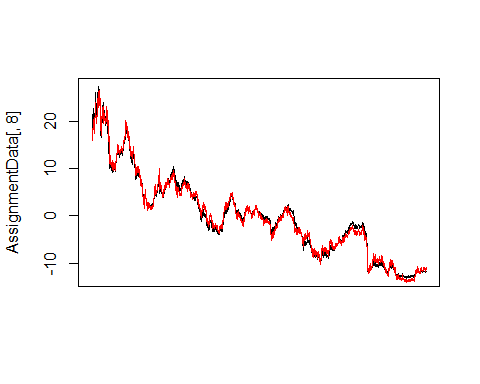
summary(Input5.Linear.Model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -15.436649 0.017818763 -866.3143 0  
## USGG5YR 2.568742 0.002581214 995.1678 0

c(R2=(summary(Input5.Linear.Model)$r.squared),R2.adj=(summary(Input5.Linear.Model)$adj.r.squared))

## R2 R2.adj   
## 0.9916908 0.9916898

matplot(AssignmentData[,8],type="l",xaxt="n")  
lines(Input5.Linear.Model$fitted.values,col="red")



The Linear Regresseion model here takes the Output against the USGG3YR. The unexplained variance amounts to .63 out of 76.8 which shows that these two data sets seem to be fairly similar. From the summary of the coefficients, the Pr(>t) = 0 shows that there is significant evidence that the coefficient and intercept are not equal to 0.

Output1~USGG10YR

Input6.Linear.Model<-lm(Output1~USGG10YR,AssignmentData)  
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input6.Linear.Model)$sigma^2)

## Total.Variance Unexplained.Variance   
## 76.804438 2.366752

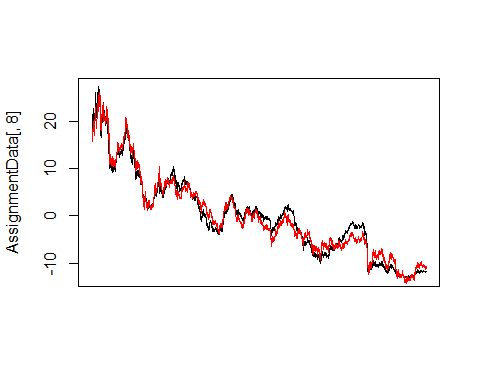
summary(Input6.Linear.Model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -18.063370 0.039181738 -461.0150 0  
## USGG10YR 2.786991 0.005455089 510.8975 0

c(R2=(summary(Input6.Linear.Model)$r.squared),R2.adj=(summary(Input6.Linear.Model)$adj.r.squared))

## R2 R2.adj   
## 0.9691884 0.9691847

matplot(AssignmentData[,8],type="l",xaxt="n")  
lines(Input6.Linear.Model$fitted.values,col="red")



The Linear Regresseion model here takes the Output against the USGG3YR. The unexplained variance amounts to 2.4 out of 76.8 which shows that these two data sets seem to be fairly similar. From the summary of the coefficients, the Pr(>t) = 0 shows that there is significant evidence that the coefficient and intercept are not equal to 0.

Output1~USGG30YR

Input7.Linear.Model<-lm(Output1~USGG30YR,AssignmentData)  
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input7.Linear.Model)$sigma^2)

## Total.Variance Unexplained.Variance   
## 76.804438 4.967286

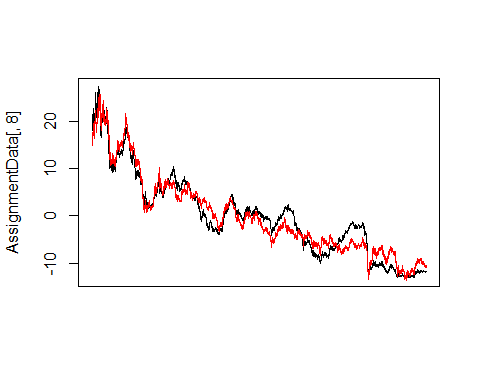
summary(Input7.Linear.Model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -21.085905 0.065596726 -321.4475 0  
## USGG30YR 3.069561 0.008860263 346.4413 0

c(R2=(summary(Input7.Linear.Model)$r.squared),R2.adj=(summary(Input7.Linear.Model)$adj.r.squared))

## R2 R2.adj   
## 0.9353333 0.9353255

matplot(AssignmentData[,8],type="l",xaxt="n")  
lines(Input7.Linear.Model$fitted.values,col="red")



The Linear Regresseion model here takes the Output against the USGG3YR. The unexplained variance amounts to 4.9 out of 76.8 which shows that these two data sets seem to be fairly similar. From the summary of our coefficients, the Pr(>t) = 0 shows that there is significant evidence that the coefficient and intercept are not equal to 0.

If one predictor must be selected from the above to best represent Output 1, USGG3YR would be used because of it’s minimal unexplained variance and it’s maximized R2. All of the models fitted values seemed to correspond to the output 1 values quite well, which is why all the models had such a high adj.R2 and a low amount of unexplained variance.

Collect all slopes and intercepts in one table and print this table. Try to do it in one line using apply() function.

apply(AssignmentData,2, function(z) lm(Output1~z,AssignmentData)$coefficients)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  
## (Intercept) -11.723184 -12.097528 -13.055775 -13.861618 -15.436649 -18.063370 -21.085905  
## z 2.507561 2.497235 2.400449 2.455793 2.568742 2.786991 3.069561  
## Output1 Easing Tightening  
## (Intercept) -4.766731e-26 7.473583 9.986771  
## z 1.000000e+00 NA NA

It’s interesting to see that all of the bonds have negative intecepts with positive slopes ranging from 2.4 to slightly above 3.

Step 3: Fit linear regression models using single output as input and each ofthe original inputs as outputs. Collect all slopes and intercepts in one table and print this table

apply(AssignmentData,2, function(z) lm(z~Output1,AssignmentData)$coefficients)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR Output1  
## (Intercept) 4.6751341 4.844370 5.4388879 5.6444580 6.009421 6.4813160 6.8693554 -4.766731e-26  
## Output1 0.3839609 0.390187 0.4151851 0.4063541 0.386061 0.3477544 0.3047124 1.000000e+00  
## Easing Tightening  
## (Intercept) 1.000000e+00 1.000000e+00  
## Output1 2.310393e-16 -2.087244e-17

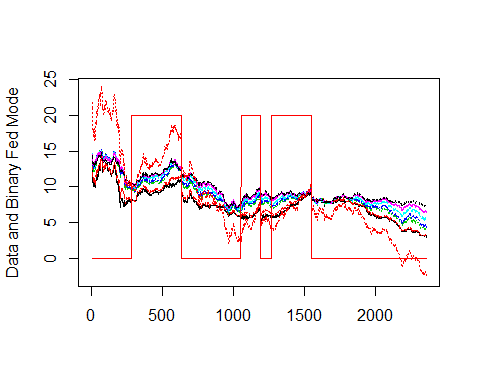
When Output 1 is made to be the input, all of the intercepts for the bonds are now positive ranging from 4.6 to 4.9 with positive slopes ranging from .3 to .4.

Step 4: Estimate logistic regression using all inputs and the data on FED tightening and easing cycles.

AssignmentDataLogistic<-data.matrix(AssignmentData,rownames.force="automatic")  
EasingPeriods<-AssignmentDataLogistic[,9]  
EasingPeriods[AssignmentDataLogistic[,9]==1]<-0  
TighteningPeriods<-AssignmentDataLogistic[,10]  
cbind(EasingPeriods,TighteningPeriods)[c(550:560,900:910,970:980),]

## EasingPeriods TighteningPeriods  
## 3/29/1983 NA NA  
## 3/30/1983 NA NA  
## 3/31/1983 NA NA  
## 4/4/1983 NA 1  
## 4/5/1983 NA 1  
## 4/6/1983 NA 1  
## 4/7/1983 NA 1  
## 4/8/1983 NA 1  
## 4/11/1983 NA 1  
## 4/12/1983 NA 1  
## 4/13/1983 NA 1  
## 8/27/1984 NA 1  
## 8/28/1984 NA 1  
## 8/29/1984 NA 1  
## 8/30/1984 NA 1  
## 8/31/1984 NA 1  
## 9/4/1984 NA NA  
## 9/5/1984 NA NA  
## 9/6/1984 NA NA  
## 9/7/1984 NA NA  
## 9/10/1984 NA NA  
## 9/11/1984 NA NA  
## 12/10/1984 0 NA  
## 12/11/1984 0 NA  
## 12/12/1984 0 NA  
## 12/13/1984 0 NA  
## 12/14/1984 0 NA  
## 12/17/1984 0 NA  
## 12/18/1984 0 NA  
## 12/19/1984 0 NA  
## 12/20/1984 0 NA  
## 12/21/1984 0 NA  
## 12/24/1984 0 NA

All.NAs<-is.na(EasingPeriods)&is.na(TighteningPeriods)  
AssignmentDataLogistic.EasingTighteningOnly<-AssignmentDataLogistic  
AssignmentDataLogistic.EasingTighteningOnly[,9]<-EasingPeriods  
AssignmentDataLogistic.EasingTighteningOnly<-AssignmentDataLogistic.EasingTighteningOnly[!All.NAs,]  
AssignmentDataLogistic.EasingTighteningOnly[is.na(AssignmentDataLogistic.EasingTighteningOnly[,10]),10]<-0  
matplot(AssignmentDataLogistic.EasingTighteningOnly[,-c(9,10)],type="l",ylab="Data and Binary Fed Mode")  
lines(AssignmentDataLogistic.EasingTighteningOnly[,10]\*20,col="red")

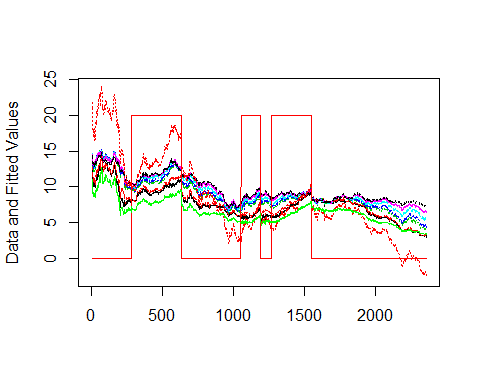


It is quite visible that when the treasurey has decided to go through a period of tightening within the past, interest rates would rise gradually. This makes sense in that if there is not hat much free cashflow in the market place, investors should receive higher returns on their free capital because cash is in demand. The initial tightening period takes place between index 250 and about index 600. This is the longest tightening period within the dataset, and there also seems to be substantial growth with the rates and especially output 1. The same relation seems to occur during the tightening period between 1100 and 1200 and the tightening period between 1300 and 1550.

LogisticModel.TighteningEasing\_3M<-glm(AssignmentDataLogistic.EasingTighteningOnly[,10]~  
 AssignmentDataLogistic.EasingTighteningOnly[,1],family=binomial(link=logit))  
summary(LogisticModel.TighteningEasing\_3M)

##   
## Call:  
## glm(formula = AssignmentDataLogistic.EasingTighteningOnly[, 10] ~   
## AssignmentDataLogistic.EasingTighteningOnly[, 1], family = binomial(link = logit))  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4239 -0.9014 -0.7737 1.3548 1.6743   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.15256 0.17328 -12.422 <2e-16 \*\*\*  
## AssignmentDataLogistic.EasingTighteningOnly[, 1] 0.18638 0.02144 8.694 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2983.5 on 2357 degrees of freedom  
## Residual deviance: 2904.8 on 2356 degrees of freedom  
## AIC: 2908.8  
##   
## Number of Fisher Scoring iterations: 4

matplot(AssignmentDataLogistic.EasingTighteningOnly[,-c(9,10)],type="l",ylab="Data and Fitted Values")  
lines(AssignmentDataLogistic.EasingTighteningOnly[,10]\*20,col="red")  
lines(LogisticModel.TighteningEasing\_3M$fitted.values\*20,col="green")



The fitted values of the glm is the green line above. The fitted values for the Easining and Tightening model against the USGG3M follows the trend of the rest of the bonds in that it has a positive slope in periods of tightening while having a negative slope in periods of easing. The only period where this doesn’t apply seems to be towards the very begining of our data set and right before the first tigteining period where the fitted values have a positive slope for a short time span. This green line is the probability created through the logistic regression which will determine if the date was in either an easining or a tightening period. Periods of tightening are identified by the red bars that flatten out at 20 on the y-axis. By taking the fitted line and multiplying it by 20, it can be thought of as a probability that it is within a tighetining period.

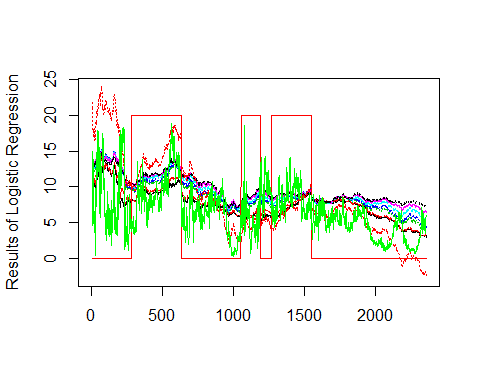
AssignmentDataLogistic.EasingTighteningOnly<-data.frame(AssignmentDataLogistic.EasingTighteningOnly)  
LogisticModel.TighteningEasing\_All<-glm(AssignmentDataLogistic.EasingTighteningOnly[,10]~.,AssignmentDataLogistic.EasingTighteningOnly[,-c(8,9,10)],family=binomial(link=logit))  
summary(LogisticModel.TighteningEasing\_All)$aic

## [1] 2645.579

summary(LogisticModel.TighteningEasing\_All)$coefficients[,c(1,4)]

## Estimate Pr(>|z|)  
## (Intercept) -4.7551928 2.784283e-28  
## USGG3M -3.3456116 4.073045e-36  
## USGG6M 4.1558535 1.422964e-28  
## USGG2YR 3.9460296 1.751687e-07  
## USGG3YR -3.4642455 2.080617e-04  
## USGG5YR -3.2115319 3.786229e-05  
## USGG10YR -0.9705444 3.202140e-01  
## USGG30YR 3.3253517 6.036041e-08

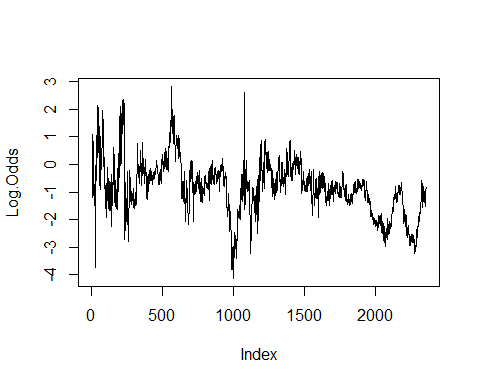
matplot(AssignmentDataLogistic.EasingTighteningOnly[,-c(9,10)],type="l",ylab="Results of Logistic Regression")  
lines(AssignmentDataLogistic.EasingTighteningOnly[,10]\*20,col="red")  
lines(LogisticModel.TighteningEasing\_All$fitted.values\*20,col="green")



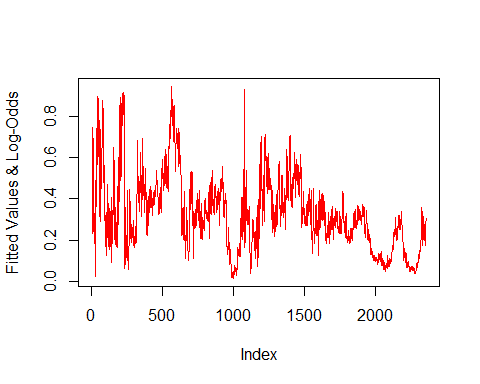
USGG3M has the smallest Pr(>|z|) which means it has the largest affect upon the glm. Due to such a large affect upon the glm model that takes into account easing and tightening, it may be wise to utilize USGG3m in the predictive model. USGG10YR has the largest Pr(>|z|) which means it has the smallest effect upon the glm. In the previous model, a logistic regression of Output 1 was taken against USGG3M. In this model, a logistic regression of Output 1 is taken against all bonds. What’s interesting is that even though USGG3M has the largest affect of any of the bonds, when the other bonds are included, there is a much larger variance within the fitted values. The fitted values still follow the trend spoken to in the previous model.

Lop.odds

Log.Odds<-predict(LogisticModel.TighteningEasing\_All)  
plot(Log.Odds,type="l")



Probabilities<-1/(exp(-Log.Odds)+1)  
plot(LogisticModel.TighteningEasing\_All$fitted.values,type="l",ylab="Fitted Values & Log-Odds")  
lines(Probabilities,col="red")



The log odds uses the fitted values of the Tightening periods against all bonds and puts it into a predictive model. By looking at the probabilities, the Y axis is reduced to a scale between 0 adn 1 and keeps a similar figure.

Step 5:

Compare linear regression models with different combinations of predicotrs. Selec tthe best combination.  
Estiamte the full model by using all 7 predictors

AssignmentDataRegressionComparison<-data.matrix(AssignmentData[,-c(9,10)],rownames.force="automatic")  
AssignmentDataRegressionComparison<-AssignmentData[,-c(9,10)]  
RegressionModelComparison.Full<-lm(AssignmentDataRegressionComparison[,8]~.,AssignmentDataRegressionComparison[,-8])  
summary(RegressionModelComparison.Full)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -14.9041591 1.056850e-10 -141024294891 0  
## USGG3M 0.3839609 9.860401e-11 3893968285 0  
## USGG6M 0.3901870 1.500111e-10 2601053702 0  
## USGG2YR 0.4151851 2.569451e-10 1615851177 0  
## USGG3YR 0.4063541 3.299038e-10 1231735395 0  
## USGG5YR 0.3860610 2.618339e-10 1474449865 0  
## USGG10YR 0.3477544 2.800269e-10 1241860763 0  
## USGG30YR 0.3047124 1.566487e-10 1945195584 0

c(R2=summary(RegressionModelComparison.Full)$r.squared, Adjusted.R2=summary(RegressionModelComparison.Full)$adj.r.squared)

## R2 Adjusted.R2   
## 1 1

summary(RegressionModelComparison.Full)$df

## [1] 8 8292 8

Q: Interpret the fitted Model. How good is the fit? How significant are the paramters?

A: Because R2 and asjursted.R2 are 1, the model is a perfect fit which would mean the model is overfitting. This is not an ideal scenario in statistics. Since all of the Pr(>|t|) =0, the current model decrees that all parameters are significant. All predictors shouldn’t be used because as a perfect model in Analytics isn’t very useful for future predictive analysis.

Estimate the null model by including only the intercept

RegressionModelComparison.Null<-lm(AssignmentDataRegressionComparison[,8]~+1,AssignmentDataRegressionComparison[,-8])  
summary(RegressionModelComparison.Null)

##   
## Call:  
## lm(formula = AssignmentDataRegressionComparison[, 8] ~ +1, data = AssignmentDataRegressionComparison[,   
## -8])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -13.173 -6.509 -0.415 4.860 27.298   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 1.42e-11 9.62e-02 0 1  
##   
## Residual standard error: 8.764 on 8299 degrees of freedom

c(R2=summary(RegressionModelComparison.Null)$r.squared, Adjusted.R2=summary(RegressionModelComparison.Null)$adj.r.squared, df=summary(RegressionModelComparison.Null)$df)

## R2 Adjusted.R2 df1 df2 df3   
## 0 0 1 8299 1

anova(RegressionModelComparison.Full,RegressionModelComparison.Null)

## Analysis of Variance Table  
##   
## Model 1: AssignmentDataRegressionComparison[, 8] ~ USGG3M + USGG6M + USGG2YR +   
## USGG3YR + USGG5YR + USGG10YR + USGG30YR  
## Model 2: AssignmentDataRegressionComparison[, 8] ~ +1  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 8292 0   
## 2 8299 637400 -7 -637400 3.73e+22 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Q: Why summary(RegressionModelComparison.Null) does not show R2?

A: When a regression model is taken for an Output against the intercept, the model doesn’t take into account any of the predictors. Without the predictors, the model cannot be explained simply by the intercept therefore there is no correlation to show for. This is why R2 and adjusted.R2 are equal to zero. When the anova is taken between the two models, RegressionModelComparison.Full has a large RSS because it explains the model fully, but it also has 7 degrees of freedom less because it ulizes all predictors.

The best combination is attempted to be found by utilizing the add1() process

(myScope<-names(AssignmentDataRegressionComparison)[-which(names(AssignmentDataRegressionComparison)=="Output1")])

## [1] "USGG3M" "USGG6M" "USGG2YR" "USGG3YR" "USGG5YR" "USGG10YR" "USGG30YR"

anova(RegressionModelComparison.Null)

## Analysis of Variance Table  
##   
## Response: AssignmentDataRegressionComparison[, 8]  
## Df Sum Sq Mean Sq F value Pr(>F)  
## Residuals 8299 637400 76.804

add1(RegressionModelComparison.Null,scope=myScope)

## Single term additions  
##   
## Model:  
## AssignmentDataRegressionComparison[, 8] ~ +1  
## Df Sum of Sq RSS AIC  
## <none> 637400 36033  
## USGG3M 1 613692 23708 8715  
## USGG6M 1 621075 16325 5618  
## USGG2YR 1 635252 2148 -11217  
## USGG3YR 1 636075 1325 -15226  
## USGG5YR 1 632104 5296 -3725  
## USGG10YR 1 617761 19639 7153  
## USGG30YR 1 596181 41219 13306

(myScope<-myScope[-which(myScope=="USGG3YR")])

## [1] "USGG3M" "USGG6M" "USGG2YR" "USGG5YR" "USGG10YR" "USGG30YR"

RegressionModelComparison.3yr<-lm(Output1~1+USGG3YR, AssignmentDataRegressionComparison)  
add1(RegressionModelComparison.3yr,myScope)

## Single term additions  
##   
## Model:  
## Output1 ~ 1 + USGG3YR  
## Df Sum of Sq RSS AIC  
## <none> 1324.83 -15226  
## USGG3M 1 407.98 916.85 -18279  
## USGG6M 1 270.17 1054.66 -17117  
## USGG2YR 1 82.93 1241.91 -15761  
## USGG5YR 1 20.94 1303.89 -15356  
## USGG10YR 1 64.05 1260.78 -15636  
## USGG30YR 1 100.79 1224.04 -15881

myScope<-myScope[-which(myScope=="USGG3M")]  
RegressionModelComparison.3yr.3m<-lm(Output1~1+USGG3YR+USGG3M, AssignmentDataRegressionComparison)  
add1(RegressionModelComparison.3yr.3m, myScope)

## Single term additions  
##   
## Model:  
## Output1 ~ 1 + USGG3YR + USGG3M  
## Df Sum of Sq RSS AIC  
## <none> 916.85 -18279  
## USGG6M 1 139.91 776.95 -19652  
## USGG2YR 1 176.99 739.86 -20058  
## USGG5YR 1 736.49 180.36 -31773  
## USGG10YR 1 864.29 52.56 -42007  
## USGG30YR 1 863.62 53.23 -41902

RegressionModelComparison.3yr.3m.30yr<-lm(Output1~1+USGG3YR+USGG3M+USGG30YR, AssignmentDataRegressionComparison)  
myScope<-myScope[-which(myScope=="USGG30YR")]  
add1(RegressionModelComparison.3yr.3m.30yr, myScope)

## Single term additions  
##   
## Model:  
## Output1 ~ 1 + USGG3YR + USGG3M + USGG30YR  
## Df Sum of Sq RSS AIC  
## <none> 53.230 -41902  
## USGG6M 1 26.5758 26.654 -47641  
## USGG2YR 1 7.5370 45.693 -43167  
## USGG5YR 1 9.5669 43.663 -43544  
## USGG10YR 1 10.9537 42.276 -43812

RegressionModelComparison.3yr.3m.30yr.6m<-lm(Output1~1+USGG3YR+USGG3M+USGG30YR+USGG6M, AssignmentDataRegressionComparison)  
myScope<-myScope[-which(myScope=="USGG6M")]  
add1(RegressionModelComparison.3yr.3m.30yr.6m, myScope)

## Single term additions  
##   
## Model:  
## Output1 ~ 1 + USGG3YR + USGG3M + USGG30YR + USGG6M  
## Df Sum of Sq RSS AIC  
## <none> 26.6542 -47641  
## USGG2YR 1 0.0556 26.5986 -47656  
## USGG5YR 1 17.6312 9.0230 -56629  
## USGG10YR 1 17.2110 9.4433 -56251

The objective is to find a combination of predictors that produces the lowest AIC. Using the add1() process, the first most important predictor is USGG3YR. When USGG3YR is taken out of our predictors and the add1() process is run again, the most important predictor is USGG3M. When USGG3YR and USGG3m are taken out of he predictors and the add1() process is run again, the most important predictor is USGG30YR.

Using the MuMIn Package, all combinations of predictors’ AICs can be seen to determine which combination of predictors will lead to the most accurate model while attempting to reserve degrees of freedom.

lm.test<-lm(Output1~., AssignmentDataRegressionComparison, na.action='na.fail')  
summary(model.avg(dredge(lm.test)))

## Fixed term is "(Intercept)"

##   
## Call:  
## model.avg(object = dredge(lm.test))  
##   
## Component model call:   
## lm(formula = Output1 ~ <128 unique rhs>, data = AssignmentDataRegressionComparison,   
## na.action = na.fail)  
##   
## Component models:   
## df logLik AICc delta weight  
## 1234567 9 156526.06 -313034.10 0.0 1  
## 123467 8 20238.68 -40461.34 272572.8 0  
## 234567 8 20170.73 -40325.44 272708.7 0  
## 123457 8 18745.84 -37475.65 275558.5 0  
## 134567 8 17985.75 -35955.48 277078.6 0  
## 23467 7 17330.88 -34647.75 278386.4 0  
## 34567 7 16543.38 -33072.75 279961.4 0  
## 124567 8 16446.09 -32876.17 280157.9 0  
## 13457 7 16354.45 -32694.89 280339.2 0  
## 12457 7 15737.16 -31460.30 281573.8 0  
## 12467 7 15637.22 -31260.43 281773.7 0  
## 123456 8 14034.49 -28052.97 284981.1 0  
## 12346 7 13670.88 -27327.74 285706.4 0  
## 14567 7 13353.62 -26693.22 286340.9 0  
## 1457 6 13340.73 -26669.45 286364.7 0  
## 23456 7 13020.92 -26027.83 287006.3 0  
## 12345 7 12876.94 -25739.87 287294.2 0  
## 12347 7 12847.23 -25680.44 287353.7 0  
## 1247 6 12827.07 -25642.13 287392.0 0  
## 2346 6 12742.30 -25472.59 287561.5 0  
## 12456 7 12691.31 -25368.61 287665.5 0  
## 1246 6 12563.03 -25114.05 287920.0 0  
## 1245 6 12298.23 -24584.44 288449.7 0  
## 23457 7 12056.89 -24099.77 288934.3 0  
## 3457 6 12048.22 -24084.43 288949.7 0  
## 1234 6 11216.36 -22420.72 290613.4 0  
## 124 5 11211.32 -22412.64 290621.5 0  
## 123567 8 10685.34 -21354.67 291679.4 0  
## 13567 7 10502.01 -20990.00 292044.1 0  
## 23567 7 10394.31 -20774.61 292259.5 0  
## 3567 6 10269.39 -20526.76 292507.3 0  
## 13456 7 10233.89 -20453.76 292580.3 0  
## 12367 7 10169.33 -20324.64 292709.5 0  
## 12357 7 10155.92 -20297.83 292736.3 0  
## 1345 6 10133.90 -20255.80 292778.3 0  
## 1357 6 10097.33 -20182.65 292851.5 0  
## 3456 6 9999.96 -19987.90 293046.2 0  
## 2367 6 9913.34 -19814.67 293219.4 0  
## 2345 6 9811.38 -19610.75 293423.4 0  
## 12567 7 9272.05 -18530.08 294504.0 0  
## 1456 6 9245.30 -18478.58 294555.5 0  
## 145 5 9230.13 -18450.26 294583.8 0  
## 1257 6 9205.88 -18399.74 294634.4 0  
## 345 5 9177.77 -18345.53 294688.6 0  
## 1267 6 9079.89 -18147.78 294886.3 0  
## 1567 6 8960.01 -17908.02 295126.1 0  
## 157 5 8955.09 -17900.16 295133.9 0  
## 2357 6 8718.56 -17425.10 295609.0 0  
## 357 5 8660.38 -17310.76 295723.3 0  
## 1237 6 8583.79 -17155.57 295878.5 0  
## 127 5 8486.64 -16963.28 296070.8 0  
## 3467 6 8156.96 -16301.92 296732.2 0  
## 13467 7 8157.25 -16300.49 296733.6 0  
## 1467 6 7459.47 -14906.94 298127.2 0  
## 367 5 7356.42 -14702.82 298331.3 0  
## 1367 6 7356.59 -14701.16 298332.9 0  
## 167 5 6643.89 -13277.78 299756.3 0  
## 2347 6 6073.10 -12134.19 300899.9 0  
## 234 5 5884.79 -11759.58 301274.5 0  
## 24567 7 5129.54 -10245.06 302789.0 0  
## 2456 6 5056.79 -10101.57 302932.5 0  
## 237 5 4997.85 -9985.68 303048.4 0  
## 2467 6 4787.23 -9562.45 303471.7 0  
## 246 5 4690.45 -9370.89 303663.2 0  
## 4567 6 4511.98 -9011.95 304022.2 0  
## 467 5 4460.30 -8910.59 304123.5 0  
## 456 5 4113.28 -8216.56 304817.5 0  
## 1346 6 3912.61 -7813.21 305220.9 0  
## 346 5 3857.91 -7705.81 305328.3 0  
## 146 5 3737.79 -7465.57 305568.5 0  
## 46 4 3432.00 -6856.00 306178.1 0  
## 2567 6 3359.55 -6707.10 306327.0 0  
## 1347 6 3330.45 -6648.89 306385.2 0  
## 137 5 3319.59 -6629.18 306404.9 0  
## 567 5 3148.48 -6286.95 306747.2 0  
## 267 5 3074.48 -6138.95 306895.2 0  
## 67 4 3071.92 -6135.83 306898.3 0  
## 147 5 2606.61 -5203.21 307830.9 0  
## 17 4 2553.65 -5099.29 307934.8 0  
## 134 5 -456.32 922.66 313956.8 0  
## 12356 7 -596.25 1206.52 314240.6 0  
## 1235 6 -619.53 1251.07 314285.2 0  
## 2356 6 -619.92 1251.85 314286.0 0  
## 1256 6 -663.07 1338.15 314372.2 0  
## 125 5 -667.10 1344.21 314378.3 0  
## 235 5 -733.35 1476.71 314510.8 0  
## 1236 6 -801.23 1614.47 314648.6 0  
## 236 5 -824.49 1658.99 314693.1 0  
## 23 4 -831.00 1670.01 314704.1 0  
## 123 5 -830.87 1671.75 314705.9 0  
## 126 5 -911.05 1832.10 314866.2 0  
## 12 4 -1124.06 2256.13 315290.2 0  
## 256 5 -1499.12 3008.25 316042.4 0  
## 2457 6 -1639.19 3290.40 316324.5 0  
## 245 5 -1744.38 3498.77 316532.9 0  
## 347 5 -1869.60 3749.20 316783.3 0  
## 457 5 -1947.34 3904.69 316938.8 0  
## 26 4 -2211.92 4431.84 317465.9 0  
## 37 4 -2389.78 4787.56 317821.7 0  
## 14 4 -2549.22 5106.44 318140.5 0  
## 257 5 -2594.10 5198.20 318232.3 0  
## 45 4 -2634.48 5276.96 318311.1 0  
## 57 4 -3215.59 6439.18 319473.3 0  
## 1356 6 -3272.40 6556.80 319590.9 0  
## 356 5 -3285.02 6580.05 319614.2 0  
## 135 5 -3428.53 6867.06 319901.2 0  
## 156 5 -3504.78 7019.57 320053.7 0  
## 35 4 -3833.69 7675.39 320709.5 0  
## 25 4 -3893.82 7795.64 320829.7 0  
## 15 4 -3956.41 7920.83 320954.9 0  
## 56 4 -4095.95 8199.90 321234.0 0  
## 5 3 -4162.07 8330.14 321364.2 0  
## 247 5 -4871.50 9753.00 322787.1 0  
## 27 4 -5813.59 11635.19 324669.3 0  
## 24 4 -6085.36 12178.73 325212.8 0  
## 2 3 -6166.78 12339.57 325373.7 0  
## 34 4 -6443.05 12894.10 325928.2 0  
## 136 5 -7264.89 14539.79 327573.9 0  
## 16 4 -7423.44 14854.89 327889.0 0  
## 36 4 -7861.05 15730.11 328764.2 0  
## 6 3 -9912.78 19831.57 332865.7 0  
## 13 4 -11175.06 22358.12 335392.2 0  
## 47 4 -13959.90 27927.80 340961.9 0  
## 7 3 -14584.38 29174.76 342208.9 0  
## 1 3 -15351.49 30708.99 343743.1 0  
## 4 3 -16132.83 32271.66 345305.8 0  
## 3 3 -18428.12 36862.23 349896.3 0  
## (Null) 2 -29792.93 59589.86 372624.0 0  
##   
## Term codes:   
## USGG10YR USGG2YR USGG30YR USGG3M USGG3YR USGG5YR USGG6M   
## 1 2 3 4 5 6 7   
##   
## Model-averaged coefficients:   
## (full average)   
## Estimate Std. Error Adjusted SE z value Pr(>|z|)   
## (Intercept) -1.490e+01 1.057e-10 1.057e-10 1.410e+11 <2e-16 \*\*\*  
## USGG10YR 3.478e-01 2.800e-10 2.801e-10 1.242e+09 <2e-16 \*\*\*  
## USGG2YR 4.152e-01 2.569e-10 2.570e-10 1.616e+09 <2e-16 \*\*\*  
## USGG30YR 3.047e-01 1.566e-10 1.567e-10 1.945e+09 <2e-16 \*\*\*  
## USGG3M 3.840e-01 9.860e-11 9.862e-11 3.893e+09 <2e-16 \*\*\*  
## USGG3YR 4.064e-01 3.299e-10 3.299e-10 1.232e+09 <2e-16 \*\*\*  
## USGG5YR 3.861e-01 2.618e-10 2.619e-10 1.474e+09 <2e-16 \*\*\*  
## USGG6M 3.902e-01 1.500e-10 1.500e-10 2.601e+09 <2e-16 \*\*\*  
##   
## (conditional average)   
## Estimate Std. Error Adjusted SE z value Pr(>|z|)   
## (Intercept) -1.490e+01 1.057e-10 1.057e-10 1.410e+11 <2e-16 \*\*\*  
## USGG10YR 3.478e-01 2.800e-10 2.801e-10 1.242e+09 <2e-16 \*\*\*  
## USGG2YR 4.152e-01 2.569e-10 2.570e-10 1.616e+09 <2e-16 \*\*\*  
## USGG30YR 3.047e-01 1.566e-10 1.567e-10 1.945e+09 <2e-16 \*\*\*  
## USGG3M 3.840e-01 9.860e-11 9.862e-11 3.893e+09 <2e-16 \*\*\*  
## USGG3YR 4.064e-01 3.299e-10 3.299e-10 1.232e+09 <2e-16 \*\*\*  
## USGG5YR 3.861e-01 2.618e-10 2.619e-10 1.474e+09 <2e-16 \*\*\*  
## USGG6M 3.902e-01 1.500e-10 1.500e-10 2.601e+09 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Relative variable importance:   
## USGG10YR USGG2YR USGG30YR USGG3M USGG3YR USGG5YR USGG6M  
## Importance: 1 1 1 1 1 1 1   
## N containing models: 64 64 64 64 64 64 64

my.model<-lm(Output1~1+USGG10YR+USGG2YR+USGG3M, AssignmentDataRegressionComparison)  
cbind("Using MuMIn"= AIC(my.model), "Using Stepwise Model" = AIC(RegressionModelComparison.3yr.3m.30yr))

## Using MuMIn Using Stepwise Model  
## [1,] -22412.65 -18345.53

cbind('Adjusted.R.2'=(summary(my.model)$adj.r.squared),'Adjusted.R.2'=(summary(RegressionModelComparison.3yr.3m.30yr)$adj.r.squared ))

## Adjusted.R.2 Adjusted.R.2  
## [1,] 0.9999488 0.9999165

If the add1() process is utilized, it would show that the USGG3YR is first most important predictor. Using the MuMIn package, if only one predictor had to be used, USGG3YR would be the best predictor as well. The problem with the add1() process is that it doesn’t take into account combinations of predictors because it is a stepwise function. With the MuMIn package, we are able to run all combinations of predictors to determine which has the lowest AIC while reserving degrees of freedom. From the table above, the AIC of the best combinations of predictors with their respective degrees of freedome can bee seen below: DF AIC: 9 -313034 8 -40461 7 -34647 6 -26669 5 -22412 4 -6856

When determining the selection of the best predictors to utilize to predict Output 1, R2 and R2.adjusted would not suffice when distinguishing between models because they don’t penalize larger data sets enough to determine which model is truly the best. Utilizing AIC, the lowest AIC was the model with df=9 where all predictors were used. This should not be the chosen model becuase the model would be overfitted. There is justification to utilize any other combination of predictors because they all produce a high R2.adjusted while not overfitting the model. A final decision to utilize 3 predictors was decided upon because there is a massive reduction in AIC between the model with 4 degrees of freedom and the model with 5 degrees of freedom. Something that was very interesting is that the model with 5 degrees of freedom didn’t include USGG3YR as a predictor though it was considered to be the most important predictor of all predictors utilizng the add1() process if only one predictor were to be utilized.

BIC is an even stricter method that is used to differentiate predictors. The equation for BIC is -2log-likelihood+n\_parameters(log(n\_observations)). Below, the model chosen utilizing MuMIn vs add1() are observed using BIC.

c('Using MuMIn'=(BIC(my.model)),'Using add1()'=(BIC(RegressionModelComparison.3yr.3m.30yr)))

## Using MuMIn Using add1()   
## -22377.53 -18310.41

BIC still identifies the combination created utilizing the MuMIn package as a better set of predictors than what would have been produced with the add1() process.

anova(my.model,RegressionModelComparison.3yr.3m.30yr)

## Analysis of Variance Table  
##   
## Model 1: Output1 ~ 1 + USGG10YR + USGG2YR + USGG3M  
## Model 2: Output1 ~ 1 + USGG3YR + USGG3M + USGG30YR  
## Res.Df RSS Df Sum of Sq F Pr(>F)  
## 1 8296 32.61   
## 2 8296 53.23 0 -20.62

Using anova, it can be seen that the model utilizing MuMIn has a smaller residual sum of squares meaning that it is a better combination of predictors for output 1.

Step 6 Rolling mean values for each variable.

Window.width<-20; Window.shift<-5  
all.means<-rollapply(AssignmentDataRegressionComparison,width=Window.width,by=Window.shift,by.column=TRUE, mean)  
head(all.means,10)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR Output1  
## [1,] 15.0405 14.0855 13.2795 12.9360 12.7825 12.5780 12.1515 20.14842  
## [2,] 15.1865 14.1440 13.4855 13.1085 12.9310 12.7370 12.3370 20.55208  
## [3,] 15.2480 14.2755 13.7395 13.3390 13.1500 12.9480 12.5500 21.04895  
## [4,] 14.9345 14.0780 13.7750 13.4765 13.2385 13.0515 12.6610 21.02611  
## [5,] 14.7545 14.0585 13.9625 13.6890 13.4600 13.2295 12.8335 21.31356  
## [6,] 14.6025 14.0115 14.0380 13.7790 13.5705 13.3050 12.8890 21.39061  
## [7,] 14.0820 13.5195 13.8685 13.6710 13.4815 13.1880 12.7660 20.77200  
## [8,] 13.6255 13.0635 13.6790 13.5735 13.4270 13.1260 12.6950 20.23626  
## [9,] 13.2490 12.6810 13.5080 13.4575 13.3680 13.0770 12.6470 19.76988  
## [10,] 12.9545 12.4225 13.4140 13.4240 13.3850 13.1115 12.6755 19.53054

Above are the rolling means where the width of the windows consisted of 20 days and the shifts consisted of 5 days.

Count<-1:length(AssignmentDataRegressionComparison[,1])  
Rolling.window.matrix<-rollapply(Count,width=Window.width,by=Window.shift,by.column=FALSE,  
 FUN=function(z) z)  
Rolling.window.matrix[1:10,]

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]  
## [1,] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
## [2,] 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22  
## [3,] 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27  
## [4,] 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32  
## [5,] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37  
## [6,] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  
## [7,] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47  
## [8,] 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52  
## [9,] 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57  
## [10,] 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62  
## [,18] [,19] [,20]  
## [1,] 18 19 20  
## [2,] 23 24 25  
## [3,] 28 29 30  
## [4,] 33 34 35  
## [5,] 38 39 40  
## [6,] 43 44 45  
## [7,] 48 49 50  
## [8,] 53 54 55  
## [9,] 58 59 60  
## [10,] 63 64 65

Points.of.calculation<-Rolling.window.matrix[,10]  
Points.of.calculation[1:10]

## [1] 10 15 20 25 30 35 40 45 50 55

length(Points.of.calculation)

## [1] 1657

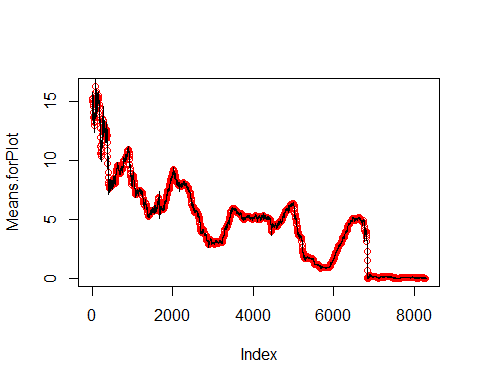
Means.forPlot<-rep(NA,length(AssignmentDataRegressionComparison[,1]))  
Means.forPlot[Points.of.calculation]<-all.means[,1]  
Means.forPlot[1:50]

## [1] NA NA NA NA NA NA NA NA NA 15.0405 NA NA  
## [13] NA NA 15.1865 NA NA NA NA 15.2480 NA NA NA NA  
## [25] 14.9345 NA NA NA NA 14.7545 NA NA NA NA 14.6025 NA  
## [37] NA NA NA 14.0820 NA NA NA NA 13.6255 NA NA NA  
## [49] NA 13.2490

cbind(AssignmentDataRegressionComparison[,1],Means.forPlot)[1:50,]

## Means.forPlot  
## [1,] 13.52 NA  
## [2,] 13.58 NA  
## [3,] 14.50 NA  
## [4,] 14.76 NA  
## [5,] 15.20 NA  
## [6,] 15.22 NA  
## [7,] 15.24 NA  
## [8,] 15.08 NA  
## [9,] 15.25 NA  
## [10,] 15.15 15.0405  
## [11,] 15.79 NA  
## [12,] 15.32 NA  
## [13,] 15.71 NA  
## [14,] 15.72 NA  
## [15,] 15.70 15.1865  
## [16,] 15.35 NA  
## [17,] 15.20 NA  
## [18,] 15.06 NA  
## [19,] 14.87 NA  
## [20,] 14.59 15.2480  
## [21,] 14.90 NA  
## [22,] 14.85 NA  
## [23,] 14.67 NA  
## [24,] 14.74 NA  
## [25,] 15.32 14.9345  
## [26,] 15.52 NA  
## [27,] 15.46 NA  
## [28,] 15.54 NA  
## [29,] 15.51 NA  
## [30,] 15.14 14.7545  
## [31,] 15.02 NA  
## [32,] 14.48 NA  
## [33,] 14.09 NA  
## [34,] 14.23 NA  
## [35,] 14.15 14.6025  
## [36,] 14.20 NA  
## [37,] 14.14 NA  
## [38,] 14.22 NA  
## [39,] 14.52 NA  
## [40,] 14.39 14.0820  
## [41,] 14.49 NA  
## [42,] 14.51 NA  
## [43,] 14.29 NA  
## [44,] 14.16 NA  
## [45,] 13.99 13.6255  
## [46,] 13.92 NA  
## [47,] 13.66 NA  
## [48,] 13.21 NA  
## [49,] 13.02 NA  
## [50,] 12.95 13.2490

plot(Means.forPlot,col="red")  
lines(AssignmentDataRegressionComparison[,1])



It can be seen here that the means for the rolling means function seems to fit the USGG3M data which would be expected due to the large number of datapoints that are utilizd. The means for the rolling means function might not fit as well if there was less data within the given dataset.

Run rolling daily difference standard deviation of each variable

AssignmentData.SD<-AssignmentData[,-c(9,10)]  
AssignmentData.SD2<-AssignmentData.SD[-1,]  
AssignmentData.SD1<-AssignmentData.SD[1:8299,]  
AssignmentData.SD.Diff<-AssignmentData.SD2-AssignmentData.SD1  
head(AssignmentData.SD.Diff,10)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR Output1  
## 1/6/1981 0.06 0.07 0.14 0.03 -0.08 -0.04 0.00 0.07587222  
## 1/7/1981 0.92 0.74 0.50 0.47 0.40 0.27 0.22 1.35591615  
## 1/8/1981 0.26 0.10 0.17 0.17 0.07 -0.03 0.02 0.30119608  
## 1/9/1981 0.44 0.30 0.44 0.33 0.20 0.22 0.22 0.82353207  
## 1/12/1981 0.02 -0.07 -0.36 -0.34 -0.17 -0.12 -0.05 -0.42985741  
## 1/13/1981 0.02 -0.13 0.13 0.03 -0.03 0.08 0.00 0.03935812  
## 1/14/1981 -0.16 -0.20 -0.35 -0.22 -0.07 0.00 -0.01 -0.40425521  
## 1/15/1981 0.17 0.19 0.30 0.27 0.16 0.09 0.18 0.52159590  
## 1/16/1981 -0.10 -0.11 -0.17 -0.17 -0.11 -0.09 -0.12 -0.33130842  
## 1/19/1981 0.64 0.75 0.54 0.07 0.26 0.11 0.12 0.96621424

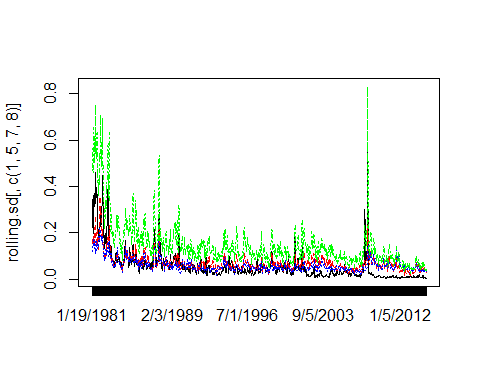
rolling.sd<-rollapply(AssignmentData.SD.Diff,width=Window.width,by=Window.shift,by.column=TRUE, sd)  
rolling.dates<-rollapply(AssignmentDataRegressionComparison[-1,],width=Window.width,by=Window.shift,  
 by.column=FALSE,FUN=function(z) rownames(z))  
head(rolling.dates)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]   
## [1,] "1/6/1981" "1/7/1981" "1/8/1981" "1/9/1981" "1/12/1981" "1/13/1981" "1/14/1981"  
## [2,] "1/13/1981" "1/14/1981" "1/15/1981" "1/16/1981" "1/19/1981" "1/20/1981" "1/21/1981"  
## [3,] "1/20/1981" "1/21/1981" "1/22/1981" "1/23/1981" "1/26/1981" "1/27/1981" "1/28/1981"  
## [4,] "1/27/1981" "1/28/1981" "1/29/1981" "1/30/1981" "2/2/1981" "2/3/1981" "2/4/1981"   
## [5,] "2/3/1981" "2/4/1981" "2/5/1981" "2/6/1981" "2/9/1981" "2/10/1981" "2/11/1981"  
## [6,] "2/10/1981" "2/11/1981" "2/13/1981" "2/17/1981" "2/18/1981" "2/19/1981" "2/20/1981"  
## [,8] [,9] [,10] [,11] [,12] [,13] [,14]   
## [1,] "1/15/1981" "1/16/1981" "1/19/1981" "1/20/1981" "1/21/1981" "1/22/1981" "1/23/1981"  
## [2,] "1/22/1981" "1/23/1981" "1/26/1981" "1/27/1981" "1/28/1981" "1/29/1981" "1/30/1981"  
## [3,] "1/29/1981" "1/30/1981" "2/2/1981" "2/3/1981" "2/4/1981" "2/5/1981" "2/6/1981"   
## [4,] "2/5/1981" "2/6/1981" "2/9/1981" "2/10/1981" "2/11/1981" "2/13/1981" "2/17/1981"  
## [5,] "2/13/1981" "2/17/1981" "2/18/1981" "2/19/1981" "2/20/1981" "2/23/1981" "2/24/1981"  
## [6,] "2/23/1981" "2/24/1981" "2/25/1981" "2/26/1981" "2/27/1981" "3/2/1981" "3/3/1981"   
## [,15] [,16] [,17] [,18] [,19] [,20]   
## [1,] "1/26/1981" "1/27/1981" "1/28/1981" "1/29/1981" "1/30/1981" "2/2/1981"   
## [2,] "2/2/1981" "2/3/1981" "2/4/1981" "2/5/1981" "2/6/1981" "2/9/1981"   
## [3,] "2/9/1981" "2/10/1981" "2/11/1981" "2/13/1981" "2/17/1981" "2/18/1981"  
## [4,] "2/18/1981" "2/19/1981" "2/20/1981" "2/23/1981" "2/24/1981" "2/25/1981"  
## [5,] "2/25/1981" "2/26/1981" "2/27/1981" "3/2/1981" "3/3/1981" "3/4/1981"   
## [6,] "3/4/1981" "3/5/1981" "3/6/1981" "3/9/1981" "3/10/1981" "3/11/1981"

rownames(rolling.sd)<-rolling.dates[,10]  
head(rolling.sd)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR Output1  
## 1/19/1981 0.3433258 0.3262462 0.2748258 0.2030161 0.1713192 0.1299585 0.1202147 0.5639875  
## 1/26/1981 0.2933383 0.2907504 0.2261811 0.1499219 0.1450082 0.1146895 0.1192201 0.4707427  
## 2/2/1981 0.2613180 0.2437530 0.2006201 0.1632596 0.1654110 0.1459308 0.1351909 0.4681168  
## 2/9/1981 0.2551754 0.2469663 0.1989446 0.1692794 0.1717219 0.1551052 0.1422183 0.4786189  
## 2/18/1981 0.2480551 0.2481595 0.2102004 0.1786057 0.1744767 0.1643960 0.1516540 0.4888569  
## 2/25/1981 0.1963884 0.2363672 0.2095082 0.1809180 0.1822917 0.1664956 0.1537351 0.4788897

matplot(rolling.sd[,c(1,5,7,8)],xaxt="n",type="l",col=c("black","red","blue","green"))  
axis(side=1,at=1:1656,rownames(rolling.sd))



Q: Show periods of high volatility. How is volatility related to the level of rates?

A: The coding used to find the rolling means is also utlized to find the rolling standard deviations. The visualization above shows the standard deviations on a line graph of USGG3M, USGG5YR, USGG3OYR, and Output 1. The standard deviations seem to have a sharp rise at around 2008. This rise could be do to the precipitous drop Output 1 has within its data set at this time period. Another interesting inference to obtain by looking at the visualization is that while USGG3M, USGG5YR,USGG3OYR, and Output 1 all seem to follow the same trend, Output 1 seems to always have a higher standard deviation.

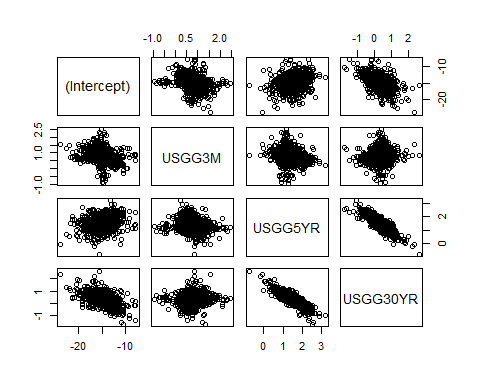
high.volatility.periods<-rownames(rolling.sd)[rolling.sd[,8]>.5]  
high.volatility.periods

## [1] "1/19/1981" "3/25/1981" "4/1/1981" "4/8/1981" "4/23/1981" "4/30/1981" "5/7/1981"   
## [8] "5/14/1981" "5/21/1981" "5/29/1981" "6/5/1981" "6/12/1981" "6/19/1981" "6/26/1981"   
## [15] "7/6/1981" "7/13/1981" "7/20/1981" "7/27/1981" "10/28/1981" "11/5/1981" "11/13/1981"  
## [22] "11/20/1981" "11/30/1981" "12/7/1981" "12/14/1981" "12/29/1981" "1/14/1982" "1/21/1982"   
## [29] "1/28/1982" "2/4/1982" "2/11/1982" "7/29/1982" "8/5/1982" "8/12/1982" "8/19/1982"   
## [36] "8/26/1982" "9/24/1982" "10/1/1982" "10/8/1982" "10/18/1982" "10/13/1987" "10/20/1987"  
## [43] "10/27/1987" "11/19/2007" "11/26/2007" "11/12/2008" "11/19/2008"

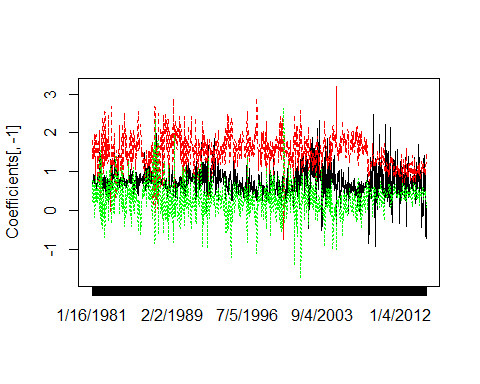
Coefficients<-rollapply(AssignmentDataRegressionComparison,width=Window.width,by=Window.shift,by.column=FALSE,  
 FUN=function(z) coef(lm(Output1~USGG3M+USGG5YR+USGG30YR,data=as.data.frame(z))))  
rolling.dates<-rollapply(AssignmentDataRegressionComparison[,1:8],width=Window.width,by=Window.shift,by.column=FALSE,  
 FUN=function(z) rownames(z))  
  
rownames(Coefficients)<-rolling.dates[,10]  
Coefficients[1:10,]

## (Intercept) USGG3M USGG5YR USGG30YR  
## 1/16/1981 -15.70877 0.6723609 1.797680 0.2276011  
## 1/23/1981 -15.96684 0.6948992 1.480514 0.5529139  
## 1/30/1981 -16.77273 0.7078197 1.434388 0.6507280  
## 2/6/1981 -16.90734 0.7279033 1.470083 0.6003377  
## 2/17/1981 -17.46896 0.7343406 1.361289 0.7499705  
## 2/24/1981 -17.04722 0.7357663 1.295641 0.7844908  
## 3/3/1981 -17.67901 0.8544681 1.396653 0.5945022  
## 3/10/1981 -17.72402 0.9162385 1.654274 0.2571200  
## 3/17/1981 -17.00260 0.9265767 1.647852 0.1951273  
## 3/24/1981 -16.84302 0.9102780 1.477727 0.3788401

pairs(Coefficients)



matplot(Coefficients[,-1],xaxt="n",type="l",col=c("black","red","green"))  
axis(side=1,at=1:1657,rownames(Coefficients))



Pairs plot  
The pairs plot of coefficients represents the coefficients between a short term bond(USGG3M), a medium term bond(USGG5YR), and a long term bond(USGG30YR). The medium and long term bonds share a stong negative correlation with respect to their coefficients. This means that their yields could be attempting to counteract one another throughout the dataset. The short term bond doesnt seem to share much correlation with respect to it’s coefficients with the medium and long term bonds. The intercept doesn’t seem to share much correlation with the short term or medium term bonds, but it does seem to have a slight negative correlation with the long term bond. The third visualization looks at the coefficients for the short, medium and long term bonds. The coefficients seem to diverge at times from their means. The dates where this phenomon can be seen are within the years of 1987,1991, 2005 and 2012.

Q: Is the picture of coefficients consistent with the picture of pairs? If yes, explain why.

A: Yes, it was stated that the 30 yr and 5 yr shared a stronger netagtive correlation with one another, and neither seemed to correlate with the 3 months. It can be seen that the green line(Long term bond) and the red line (medium term bond) seem to reflect one another. The black line remains between the two for a majority of the visualization.

high.slopespread.periods<-rownames(Coefficients)[Coefficients[,3]-Coefficients[,4]>3]  
jump.slopes<-rownames(Coefficients)[Coefficients[,3]>3]  
high.slopespread.periods

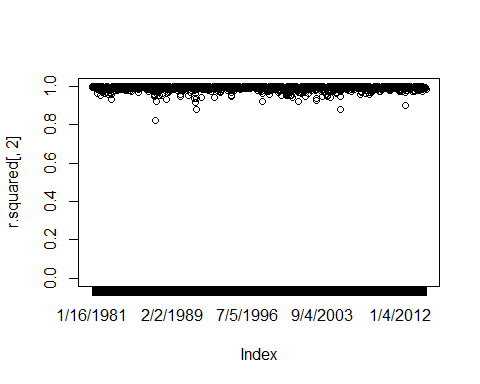
## [1] "4/26/1982" "12/15/1982" "9/16/1985" "5/12/1987" "5/19/1987" "5/27/1987" "9/25/1987"   
## [8] "3/15/1988" "9/27/1988" "10/5/1988" "3/10/1989" "2/5/1992" "8/3/1994" "12/8/1994"   
## [15] "6/14/1996" "5/9/1997" "5/16/1997" "5/23/1997" "5/30/1997" "12/26/2000" "1/2/2001"   
## [22] "7/25/2001" "8/1/2001" "11/13/2003" "8/12/2004" "12/16/2004"

This shows when dates where USGG5YR had a greater coefficeint than USGG30YR.This would be where the green line goes above the red line in the previous visualizaiton.

r.squared<-rollapply(AssignmentDataRegressionComparison,width=Window.width,by=Window.shift,by.column=FALSE,  
 FUN=function(z) summary(lm(Output1~USGG3M+USGG5YR+USGG30YR,data=as.data.frame(z)))$r.squared)  
r.squared<-cbind(rolling.dates[,10],r.squared)  
r.squared[1:10,]

## r.squared   
## [1,] "1/16/1981" "0.995046300986446"  
## [2,] "1/23/1981" "0.992485868136646"  
## [3,] "1/30/1981" "0.998641209587999"  
## [4,] "2/6/1981" "0.998849080081881"  
## [5,] "2/17/1981" "0.997958757207598"  
## [6,] "2/24/1981" "0.996489757136839"  
## [7,] "3/3/1981" "0.99779753570421"   
## [8,] "3/10/1981" "0.998963395226792"  
## [9,] "3/17/1981" "0.998729445388789"  
## [10,] "3/24/1981" "0.997073000898673"

plot(r.squared[,2],xaxt="n",ylim=c(0,1))  
axis(side=1,at=1:1657,rownames(Coefficients))



Q: How often the R-Squared is not conidered high? All of the R-squared seem to be very high. They will not share that level of correlation when their coffeicients diverge from their average which was spoken to with the last visualization. In the second visualization, the R-squared is near 100% for a majority of the data set. It does diverge a bit lower than 95%, and this can be attributed to the fact that the coefficients diverge from their means at certain points of the data set which can be seen in the previous visualization. The dates where this phenomon can be seen are within the years of 1987,1991, 2005 and 2012. This is determined through the code below.

(low.r.squared.periods<-r.squared[r.squared[,2]<.9,1])

## [1] "6/24/1987" "6/27/1991" "4/28/2005" "6/20/2012"

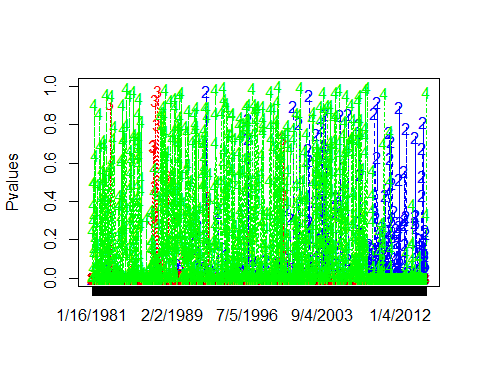
Q: What could cause the decrease of R2

A: There are certain dates where the data for the coeffieicnets extend past one another. With the plot of coefficients above, the extension of these models can be seen which would cause for low R2. The dates where this phenomon can be seen are within the years of 1987,1991, 2005 and 2012.

Pvalues<-rollapply(AssignmentDataRegressionComparison,width=Window.width,by=Window.shift,by.column=FALSE,  
 FUN=function(z) summary(lm(Output1~USGG3M+USGG5YR+USGG30YR,data=as.data.frame(z)))$coefficients[,4])  
rownames(Pvalues)<-rolling.dates[,10]  
Pvalues[1:10,]

## (Intercept) USGG3M USGG5YR USGG30YR  
## 1/16/1981 1.193499e-10 3.764585e-10 2.391260e-07 2.538852e-01  
## 1/23/1981 3.751077e-12 1.008053e-11 2.447369e-07 5.300949e-03  
## 1/30/1981 3.106359e-18 1.406387e-14 4.040035e-09 3.626961e-05  
## 2/6/1981 2.591522e-19 3.360104e-19 3.828054e-11 2.221691e-05  
## 2/17/1981 1.897239e-16 6.578118e-17 1.461743e-09 1.331767e-04  
## 2/24/1981 2.341158e-13 1.000212e-13 9.008221e-07 4.733543e-03  
## 3/3/1981 5.435581e-14 1.535503e-11 3.357199e-06 6.010473e-02  
## 3/10/1981 6.227624e-16 1.178498e-16 1.679479e-05 3.851840e-01  
## 3/17/1981 9.592582e-17 7.065226e-20 1.459692e-05 5.025726e-01  
## 3/24/1981 8.248747e-16 6.689840e-16 6.413371e-04 3.052705e-01

matplot(Pvalues,xaxt="n",col=c("black","blue","red","green"),type="o")  
axis(side=1,at=1:1657,rownames(Coefficients))



Q: What does this graph tell us?

A: The USGG30YR bonds were non-significant against the output up until 2008. After 2008, the USGG3M began to have strong insignificance. The USGG5YR has a short period of insiginificance at the end of the 80s.

Below, the dates of insignificance can be observed for the USGG3M, USGG%YR, and USGG30YR can be seen.

c("USGG3M")

## [1] "USGG3M"

rownames(Pvalues)[Pvalues[,2]>.5]

## [1] "7/15/1992" "7/26/1996" "8/2/1996" "11/7/2000" "5/30/2001" "5/2/2002" "5/16/2002"   
## [8] "5/23/2002" "1/30/2003" "2/6/2003" "7/24/2003" "7/31/2003" "8/7/2003" "11/20/2003"  
## [15] "12/18/2003" "4/28/2005" "2/10/2006" "3/9/2007" "3/16/2007" "7/21/2009" "10/6/2009"   
## [22] "10/13/2009" "12/28/2010" "1/11/2011" "3/1/2011" "11/16/2011" "11/23/2011" "5/23/2012"   
## [29] "7/11/2012" "6/6/2013" "1/16/2014" "1/30/2014" "3/6/2014"

c("USGG5YR")

## [1] "USGG5YR"

rownames(Pvalues)[Pvalues[,3]>.5]

## [1] "12/1/1982" "3/16/1987" "4/28/1987" "6/24/1987" "9/3/1987" "9/11/1987" "9/20/1988" "12/3/1999"

c("USGG30YR")

## [1] "USGG30YR"

rownames(Pvalues)[Pvalues[,4]>.5]

## [1] "3/17/1981" "4/22/1981" "4/29/1981" "6/4/1981" "10/13/1981" "11/19/1981" "2/3/1982"   
## [8] "2/26/1982" "4/2/1982" "4/12/1982" "5/3/1982" "7/7/1982" "9/1/1982" "9/23/1982"   
## [15] "12/8/1982" "2/18/1983" "3/1/1983" "5/4/1983" "12/30/1983" "1/9/1984" "2/6/1984"   
## [22] "3/14/1984" "3/21/1984" "4/11/1984" "6/22/1984" "6/29/1984" "10/31/1984" "11/16/1984"  
## [29] "11/26/1984" "12/17/1984" "3/25/1985" "5/14/1985" "5/21/1985" "8/30/1985" "9/9/1985"   
## [36] "9/23/1985" "10/1/1985" "10/8/1985" "10/16/1985" "10/23/1985" "10/30/1985" "11/14/1985"  
## [43] "11/21/1985" "1/22/1986" "1/29/1986" "5/5/1987" "12/16/1987" "1/25/1988" "2/1/1988"   
## [50] "2/16/1988" "3/1/1988" "3/22/1988" "5/18/1988" "6/3/1988" "6/10/1988" "6/24/1988"   
## [57] "7/25/1988" "8/15/1988" "12/5/1988" "2/2/1989" "3/3/1989" "4/10/1989" "5/1/1989"   
## [64] "6/13/1989" "8/16/1989" "9/14/1989" "9/21/1989" "10/3/1989" "10/11/1989" "10/18/1989"  
## [71] "11/1/1989" "11/30/1989" "12/7/1989" "1/8/1990" "1/16/1990" "6/15/1990" "7/30/1990"   
## [78] "8/6/1990" "10/2/1990" "10/10/1990" "11/23/1990" "3/1/1991" "5/13/1991" "5/20/1991"   
## [85] "6/13/1991" "7/5/1991" "7/19/1991" "9/16/1991" "2/12/1992" "2/20/1992" "3/12/1992"   
## [92] "4/16/1992" "4/24/1992" "5/1/1992" "5/8/1992" "6/2/1992" "6/9/1992" "6/16/1992"   
## [99] "8/19/1992" "8/26/1992" "10/23/1992" "5/20/1993" "6/11/1993" "6/18/1993" "8/30/1993"   
## [106] "12/15/1993" "12/22/1993" "3/16/1994" "3/30/1994" "4/6/1994" "4/20/1994" "4/27/1994"   
## [113] "6/29/1994" "8/17/1994" "9/21/1994" "9/28/1994" "12/22/1994" "12/29/1994" "1/5/1995"   
## [120] "1/12/1995" "1/26/1995" "2/2/1995" "2/9/1995" "4/6/1995" "4/13/1995" "8/25/1995"   
## [127] "9/29/1995" "10/27/1995" "11/3/1995" "11/10/1995" "12/29/1995" "1/5/1996" "1/12/1996"   
## [134] "1/19/1996" "3/29/1996" "5/31/1996" "6/21/1996" "7/12/1996" "7/19/1996" "7/26/1996"   
## [141] "8/2/1996" "8/9/1996" "8/16/1996" "8/23/1996" "9/27/1996" "10/4/1996" "12/6/1996"   
## [148] "2/28/1997" "3/7/1997" "4/18/1997" "4/25/1997" "5/2/1997" "6/13/1997" "6/20/1997"   
## [155] "6/27/1997" "7/4/1997" "10/10/1997" "10/17/1997" "12/12/1997" "12/19/1997" "12/26/1997"  
## [162] "1/9/1998" "1/16/1998" "8/14/1998" "8/21/1998" "8/28/1998" "9/18/1998" "9/25/1998"   
## [169] "12/4/1998" "12/11/1998" "1/8/1999" "3/12/1999" "4/2/1999" "5/14/1999" "6/25/1999"   
## [176] "7/9/1999" "7/30/1999" "8/20/1999" "9/10/1999" "9/24/1999" "10/15/1999" "12/31/1999"  
## [183] "3/3/2000" "3/31/2000" "4/7/2000" "4/14/2000" "4/21/2000" "7/25/2000" "11/21/2000"  
## [190] "11/28/2000" "3/7/2001" "5/30/2001" "7/11/2001" "10/11/2001" "12/13/2001" "1/24/2002"   
## [197] "1/31/2002" "8/29/2002" "9/26/2002" "12/26/2002" "1/16/2003" "1/23/2003" "3/6/2003"   
## [204] "3/13/2003" "3/20/2003" "3/27/2003" "5/1/2003" "6/19/2003" "6/26/2003" "7/3/2003"   
## [211] "9/18/2003" "9/25/2003" "10/16/2003" "10/23/2003" "10/30/2003" "11/20/2003" "1/1/2004"   
## [218] "2/5/2004" "2/26/2004" "3/4/2004" "4/15/2004" "4/22/2004" "5/13/2004" "5/27/2004"   
## [225] "6/3/2004" "6/17/2004" "6/24/2004" "7/8/2004" "10/14/2004" "10/21/2004" "10/28/2004"  
## [232] "11/18/2004" "12/23/2004" "3/17/2005" "3/24/2005" "3/31/2005" "4/7/2005" "7/21/2005"   
## [239] "8/11/2005" "9/22/2005" "10/6/2005" "11/3/2005" "12/8/2005" "12/15/2005" "1/20/2006"   
## [246] "5/12/2006" "5/19/2006" "5/26/2006" "6/2/2006" "6/9/2006" "6/16/2006" "7/7/2006"   
## [253] "7/21/2006" "10/6/2006" "11/3/2006" "1/12/2007" "2/23/2007" "3/16/2007" "4/27/2007"   
## [260] "5/25/2007" "9/7/2007" "9/14/2007" "9/21/2007" "9/28/2007" "11/16/2007" "5/5/2009"   
## [267] "6/1/2010" "6/15/2010" "1/25/2011" "2/1/2011" "6/12/2014"

Step 7

AssignmentData.Output<-(AssignmentData$Output1)  
AssignmentData<-data.matrix(AssignmentData[,1:7],rownames.force="automatic")  
AssignmentData.cov<-data.matrix(AssignmentData[,1:7], rownames.force="NA")  
dim(AssignmentData)

## [1] 8300 7

head(AssignmentData)

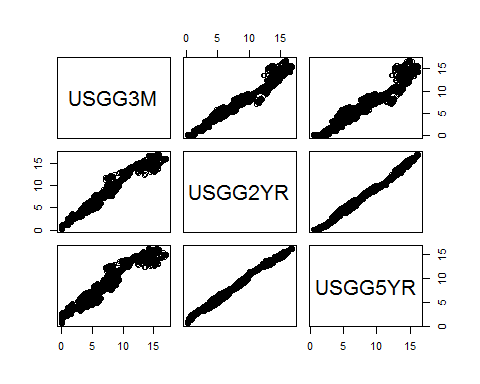
## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  
## 1/5/1981 13.52 13.09 12.289 12.28 12.294 12.152 11.672  
## 1/6/1981 13.58 13.16 12.429 12.31 12.214 12.112 11.672  
## 1/7/1981 14.50 13.90 12.929 12.78 12.614 12.382 11.892  
## 1/8/1981 14.76 14.00 13.099 12.95 12.684 12.352 11.912  
## 1/9/1981 15.20 14.30 13.539 13.28 12.884 12.572 12.132  
## 1/12/1981 15.22 14.23 13.179 12.94 12.714 12.452 12.082

Explore the dimensionality of the set of 3, 2Y and 5Y yields.

AssignmentData.3M\_2Y\_5Y<-AssignmentData[,c(1,3,5)]  
length(AssignmentData.3M\_2Y\_5Y)

## [1] 24900

pairs(AssignmentData.3M\_2Y\_5Y)



cov(AssignmentData[,-c(8,9)])

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  
## USGG3M 11.760393 11.855287 12.303031 11.942035 11.188856 9.924865 8.587987  
## USGG6M 11.855287 12.000510 12.512434 12.158422 11.406959 10.128890 8.768702  
## USGG2YR 12.303031 12.512434 13.284203 12.977542 12.279514 11.005377 9.600181  
## USGG3YR 11.942035 12.158422 12.977542 12.708647 12.068078 10.856033 9.497246  
## USGG5YR 11.188856 11.406959 12.279514 12.068078 11.543082 10.463386 9.212159  
## USGG10YR 9.924865 10.128890 11.005377 10.856033 10.463386 9.583483 8.510632  
## USGG30YR 8.587987 8.768702 9.600181 9.497246 9.212159 8.510632 7.624304

The covariance matrix’s values initially increase as you move to the bottom right of the matrix, but then they begin to decrease. This means that USGG2YR, USGG3YR, and USGG5YR share a strong amount of variance within their data sets with other short and mid term bonds. The longer term bonds share less variance with one another. The USGG2YR and the USGG5YR share a very strong positive correlation wtih one anotehr. The USGG3M also shares a strong polsitive correlation with the two otehr predictors but there seems to be some deviation from that correlation towards the end of each line. The deviations for the USGG3M could be due to the insignificance of the coefficient towards the end of the data set as seen previously in the visualization which looked at the dates of insignificance for USGG3M.

Observe the 3d plot of the set.

rgl.points(AssignmentData.3M\_2Y\_5Y)

The three dimensional plot produced through the code above combines the data sets for the USGG3M, USGG2YR, and the USGG5YR data. It is flat due to the fact that USGG5YR and USGG2YR share such a strong correlation. We can see deviations which reflect USGG3M in the pair plot above which is representative of the deviation of USGG3M to both USGG2YR and USGG5YR.

Manually Solving for Covariance

AssignmentData.means<-cbind(rep(mean(AssignmentData[,1]),8300),rep(mean(AssignmentData[,2]),8300),rep(mean(AssignmentData[,3]),8300),rep(mean(AssignmentData[,4]),8300),rep(mean(AssignmentData[,5]),8300),rep(mean(AssignmentData[,6]),8300),rep(mean(AssignmentData[,7]),8300))  
D<-AssignmentData.cov-AssignmentData.means  
D<-as.matrix(D)  
n<-nrow(AssignmentData)  
C<-(t(D)%\*%(D))/n  
C

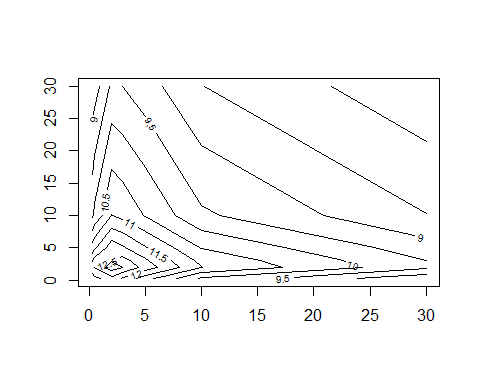
## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  
## USGG3M 11.758976 11.853859 12.301548 11.940597 11.187507 9.923669 8.586952  
## USGG6M 11.853859 11.999064 12.510927 12.156957 11.405584 10.127670 8.767646  
## USGG2YR 12.301548 12.510927 13.282602 12.975978 12.278035 11.004051 9.599024  
## USGG3YR 11.940597 12.156957 12.975978 12.707115 12.066624 10.854725 9.496102  
## USGG5YR 11.187507 11.405584 12.278035 12.066624 11.541691 10.462125 9.211049  
## USGG10YR 9.923669 10.127670 11.004051 10.854725 10.462125 9.582328 8.509607  
## USGG30YR 8.586952 8.767646 9.599024 9.496102 9.211049 8.509607 7.623386

In object D, the data is centered by subtracting the mean. By taking the dot product between the transpose of D and D and then diving by n, the covariance matrix is produced.

Covariance.Matrix<-cov(AssignmentData[,-c(8,9)])  
Covariance.Matrix

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  
## USGG3M 11.760393 11.855287 12.303031 11.942035 11.188856 9.924865 8.587987  
## USGG6M 11.855287 12.000510 12.512434 12.158422 11.406959 10.128890 8.768702  
## USGG2YR 12.303031 12.512434 13.284203 12.977542 12.279514 11.005377 9.600181  
## USGG3YR 11.942035 12.158422 12.977542 12.708647 12.068078 10.856033 9.497246  
## USGG5YR 11.188856 11.406959 12.279514 12.068078 11.543082 10.463386 9.212159  
## USGG10YR 9.924865 10.128890 11.005377 10.856033 10.463386 9.583483 8.510632  
## USGG30YR 8.587987 8.768702 9.600181 9.497246 9.212159 8.510632 7.624304

Maturities<-c(.25,.5,2,3,5,10,30)  
contour(Maturities,Maturities,Covariance.Matrix)



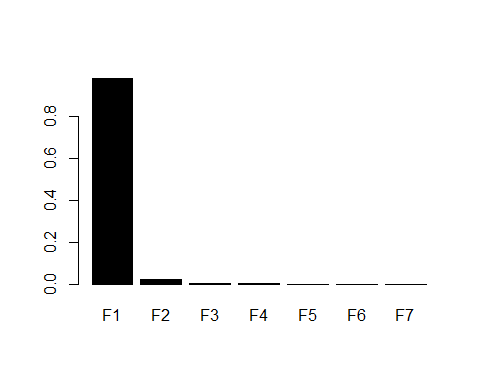
Cov() can also be utilized to create the covariance matrix. The countour plot attempts to create a 3d image using a 2d image. It can be seen that the lower left part of the graph has the highest elements of the graph with a height of 12.5. This number gradually decreases as we move to the top right of the graph. This means that the longer bonds tend to share less covariance with other bonds than the shorter term bonds. This would make sense becuase predicting the market in the long run is incredibly difficult therefore rates shouldn’t fluctuate nearly as much as they would with the short term bonds.

PCA Manually Done

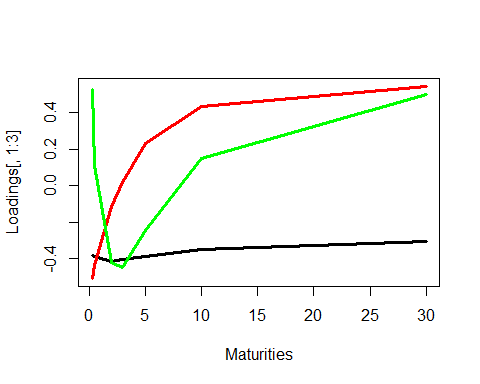
AssignmentData.Centered<-AssignmentData-AssignmentData.means  
Eigen.Decomposition<-eigen(Covariance.Matrix)  
Loadings <- Eigen.Decomposition$vectors  
Factors <- AssignmentData.Centered%\*%Loadings  
AssignmentData.Output.Factors<-as.data.frame(cbind(AssignmentData.Output,Factors))  
LinMod.PCA<-lm(AssignmentData.Output~.,AssignmentData.Output.Factors)  
calc.relimp(LinMod.PCA)

## Response variable: AssignmentData.Output   
## Total response variance: 76.80444   
## Analysis based on 8300 observations   
##   
## 7 Regressors:   
## V2 V3 V4 V5 V6 V7 V8   
## Proportion of variance explained by model: 100%  
## Metrics are not normalized (rela=FALSE).   
##   
## Relative importance metrics:   
##   
## lmg  
## V2 1  
## V3 0  
## V4 0  
## V5 0  
## V6 0  
## V7 0  
## V8 0  
##   
## Average coefficients for different model sizes:   
##   
## 1X 2Xs 3Xs 4Xs 5Xs 6Xs  
## V2 -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00 -1.000000e+00  
## V3 -8.736166e-12 -8.738461e-12 -8.740756e-12 -8.743050e-12 -8.745345e-12 -8.747640e-12  
## V4 4.488087e-11 4.489814e-11 4.491541e-11 4.493269e-11 4.494996e-11 4.496723e-11  
## V5 -1.077551e-11 -1.060446e-11 -1.043341e-11 -1.026236e-11 -1.009131e-11 -9.920260e-12  
## V6 2.189490e-10 2.190589e-10 2.191688e-10 2.192786e-10 2.193885e-10 2.194984e-10  
## V7 -2.103928e-10 -2.099526e-10 -2.095124e-10 -2.090722e-10 -2.086320e-10 -2.081917e-10  
## V8 3.369576e-10 3.377625e-10 3.385675e-10 3.393725e-10 3.401774e-10 3.409824e-10  
## 7Xs  
## V2 -1.000000e+00  
## V3 -8.749935e-12  
## V4 4.498450e-11  
## V5 -9.749210e-12  
## V6 2.196083e-10  
## V7 -2.077515e-10  
## V8 3.417874e-10

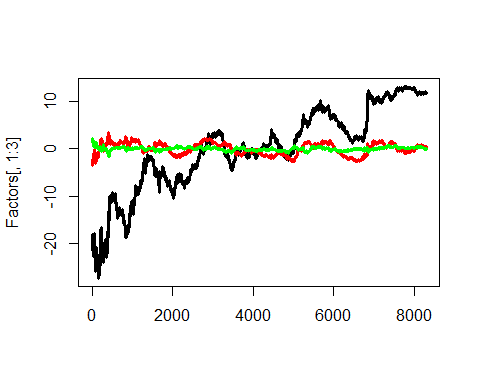
barplot(Eigen.Decomposition$values/sum(Eigen.Decomposition$values),width=2,col = "black",  
 names.arg=c("F1","F2","F3","F4","F5","F6","F7"))



#Identify columns you want, and then remove the [,1:3] below  
matplot(Maturities,Loadings[,1:3],type="l",lty=1,col=c("black","red","green"),lwd=3)



matplot(Factors[,1:3],type="l",col=c("black","red","green"),lty=1,lwd=3)



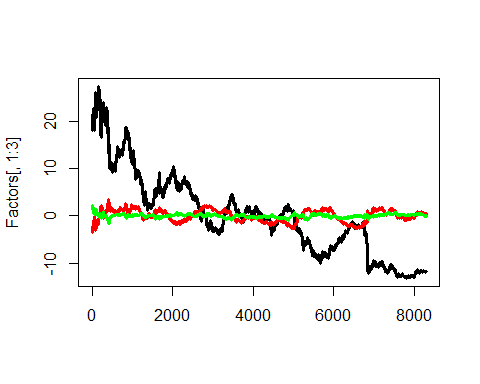
1) In the first output from the data above, the first factor has a relative importance of 1 compared to the rest of the factors. We can see the Loadings of the eigenvalues by looking at the coefficients for the different model sizes.

2) The barplot indicates again that the relative importance of the 1st factor is one, which is incredibly high compared to the other factors.

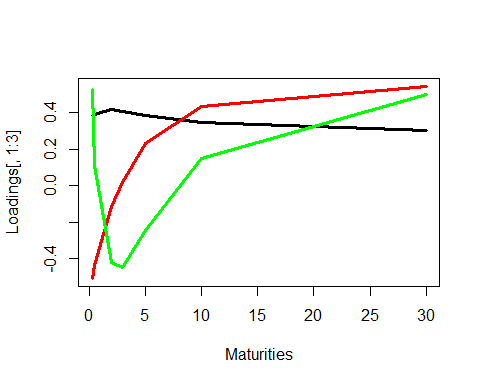
3) The third visualization looks at how the first three loadings effect each of the bonds. Eigenvalues are the equivalents to our loadings. The first loading follows a squared function, the second loading follows a quadratic function, and the third loading follows a cubic function. The first loading remains positive for the entirety of the visualization. The first loading seems to peak at the USGG2YR, and then has a negative slope for the remainder of the dataset. The second loading initially has a very strong positive slope, and it tapers out as it approaches the USGG30YR. The third loading initially starts as a positive effect, but it shoots down into the negatives at around the USGG2YR. It seemingly continues to have an even stronger negative effect on the USGG3YR.The third loading has a negative effect on the USGG5YR,but it is much less than the two preceding maturities. The third loading has a positive effect on both USGG10YR and USGG30Yr. It seems that the first and third loadings attempt to counterbalance the 2nd loading at the very beggiing of the data set. As the second loading increases its effect in a positive direction, the third loading moves to have a neggative effect with a very strong negative slope. They both end as positive effects at the USGG30YR. The second loading seems to have a possitive effect at the USGG3YR, while the 3rd Loading initially becomes negative at around the USGG6M.  
4)In the Fourth Visualization, Factor 1 has a massive impact on the dataset. It has a stronger impact than any of the factors for a majority of the dataset execpt for between the index of 2500 and 5000. Even within that range, there are several points where Factor 1 effect exceeds that of both factor 2 and factor 3. A question that could be asked is why does factor 1 initially have a negative effect and then a positive effect towards the end of our data set. This visualization occurs because during the process of PCA, a line is placed on the axis where there is the most variation. This line doesn’t necessarily point in the right direction which is why the magniuted of factor 1 is correct, yet it’s sign is incorrect. This is why we multiple factor 1 by -1 below.

Change the signs of the first facor and the corresponding factor loading

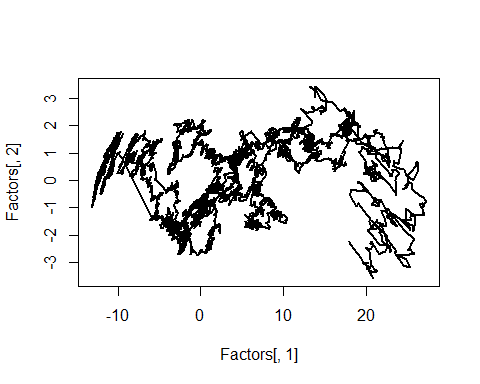
Loadings[,1]<--Loadings[,1]  
Factors[,1]<--Factors[,1]  
matplot(Factors[,1:3],type="l",col=c("black","red","green"),lty=1,lwd=3)



matplot(Maturities,Loadings[,1:3],type="l",lty=1,col=c("black","red","green"),lwd=3)



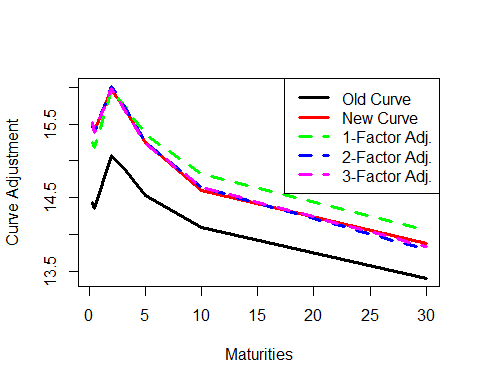
plot(Factors[,1],Factors[,2],type="l",lwd=2)



Now, both of Factor 1’s magnitude and sign correlate with how the predictors act within the data set.

Q:Draw at least three conclusions from the plot of the first two factors above A: 1) There seems to be a cyclical relationship between the two factors at the 0 mark on the x-axis and between 0 and 2 marks on the y-axis. 2) There is also a strong negative correlation between the -10 mark and 0 on the x-axis which could be representative that when factor 1 begins to move to have a strong positive weight from an originally negative weight, factor 2 begins to have a negative weigh on the data. 3) At 20 on the x axis, it seems that when Factor 1 is having a very strong weight on the data, Factor 2’s weight can range between -3 and positive 1 which is quite large. 4) There also seems to be a genera trend from the 0 to the 16 mark on the x-axis where factor 1 and factor 2 both shar a negative slope. Factor 2 only increases around 4 units while factor 1 increases by 16. This shows that factor 1 has a much stronger effect on the data than factor 2 for this period.

OldCurve<-AssignmentData[135,]  
NewCurve<-AssignmentData[136,]  
CurveChange<-NewCurve-OldCurve  
FactorsChange<-Factors[136,]-Factors[135,]  
ModelCurveAdjustment.1Factor<-OldCurve+t(Loadings[,1])\*FactorsChange[1]  
ModelCurveAdjustment.2Factors<-OldCurve+t(Loadings[,1])\*FactorsChange[1]+t(Loadings[,2])\*FactorsChange[2]  
ModelCurveAdjustment.3Factors<-OldCurve+t(Loadings[,1])\*FactorsChange[1]+t(Loadings[,2])\*FactorsChange[2]+  
 t(Loadings[,3])\*FactorsChange[3]  
matplot(Maturities,  
 t(rbind(OldCurve,NewCurve,ModelCurveAdjustment.1Factor,ModelCurveAdjustment.2Factors,  
 ModelCurveAdjustment.3Factors)),  
 type="l",lty=c(1,1,2,2,2),col=c("black","red","green","blue","magenta"),lwd=3,ylab="Curve Adjustment")  
legend(x="topright",c("Old Curve","New Curve","1-Factor Adj.","2-Factor Adj.",  
 "3-Factor Adj."),lty=c(1,1,2,2,2),lwd=3,col=c("black","red","green","blue","magenta"))



rbind(CurveChange,ModelCurveAdjustment.3Factors-OldCurve)

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  
## CurveChange 1.070000 1.070000 0.8900000 0.8300000 0.7200000 0.5000000 0.4700000  
## 1.090063 1.041267 0.9046108 0.8248257 0.6979317 0.5531734 0.4357793

Q: Explain how shapes of the loadings affect the adjustments using only factor 1, factors 1 and 2, and all 3 factors.

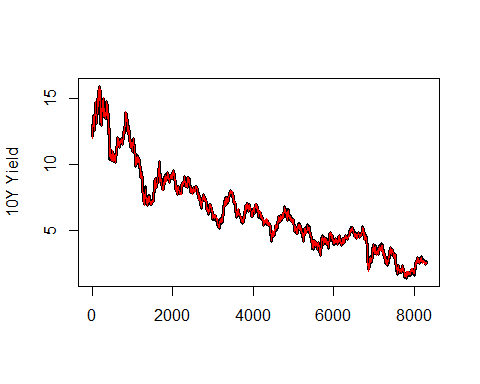
A: The graph above compares rows 136 and 135, and aims to see how the factors affect each of the maturities. The first factor: The first factor takes the old curve and inreases the new curve’s Y greatly. It can be seen that the first factor under weights the new curve from 0 to 2.5 on the x axis, and overshoots the new curve from around 5 to 30 on the x-axis. The first and second factor and all 3 factors combined do a nearly perfect job of fitting the new curve.

See the goodness of fit for the example of 10Y yield.

cbind(Maturities,Loadings)

## Maturities   
## [1,] 0.25 0.3839609 -0.50744508 0.5298222 0.40373501 0.3860878 -0.03976285 0.026742547  
## [2,] 0.50 0.3901870 -0.43946144 0.1114737 -0.40526448 -0.6787624 0.09475452 -0.090913541  
## [3,] 2.00 0.4151851 -0.11112721 -0.4187873 -0.40896949 0.3787209 -0.29848638 0.490009873  
## [4,] 3.00 0.4063541 0.01696988 -0.4476561 0.06433748 0.2362448 0.19760026 -0.731570606  
## [5,] 5.00 0.3860610 0.23140317 -0.2462364 0.53357656 -0.2868460 0.42125768 0.438559615  
## [6,] 10.00 0.3477544 0.43245979 0.1500903 0.19856539 -0.2562426 -0.73561857 -0.152627535  
## [7,] 30.00 0.3047124 0.54421228 0.4979195 -0.42098839 0.2074508 0.37776687 0.009199827

Model.10Y<-AssignmentData.means[,6]+Loadings[6,1]\*Factors[,1]+Loadings[6,2]\*Factors[,2]+Loadings[6,3]\*Factors[,3]  
matplot(cbind(AssignmentData[,6],Model.10Y),type="l",lty=1,lwd=c(3,1),col=c("black","red"),ylab="10Y Yield")



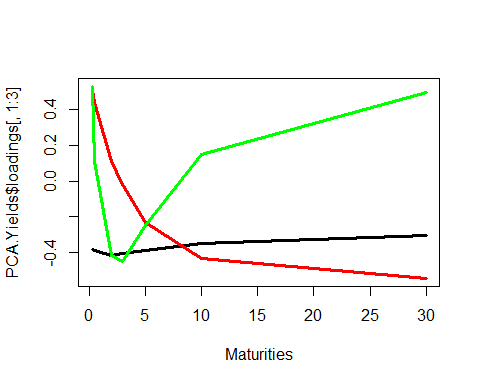
As expectd the red line fits the black line because the eigenvalues multiplied by their vectors will reproduce numbers very close to the original data set.

PCA not manually done

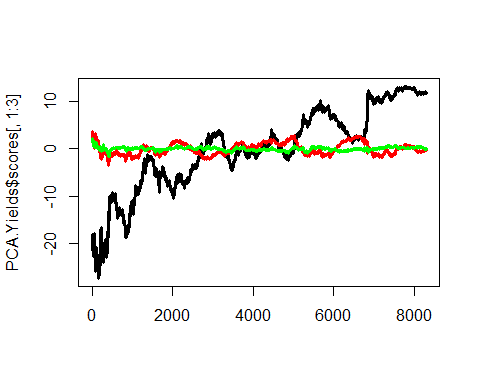
PCA.Yields<-princomp(AssignmentData[,1:7])  
cbind(PCA.Yields$loadings[,1:3],Maturities,Eigen.Decomposition$vectors[,1:3])

## Comp.1 Comp.2 Comp.3 Maturities   
## USGG3M -0.3839609 0.50744508 0.5298222 0.25 -0.3839609 -0.50744508 0.5298222  
## USGG6M -0.3901870 0.43946144 0.1114737 0.50 -0.3901870 -0.43946144 0.1114737  
## USGG2YR -0.4151851 0.11112721 -0.4187873 2.00 -0.4151851 -0.11112721 -0.4187873  
## USGG3YR -0.4063541 -0.01696988 -0.4476561 3.00 -0.4063541 0.01696988 -0.4476561  
## USGG5YR -0.3860610 -0.23140317 -0.2462364 5.00 -0.3860610 0.23140317 -0.2462364  
## USGG10YR -0.3477544 -0.43245979 0.1500903 10.00 -0.3477544 0.43245979 0.1500903  
## USGG30YR -0.3047124 -0.54421228 0.4979195 30.00 -0.3047124 0.54421228 0.4979195

matplot(Maturities,PCA.Yields$loadings[,1:3],type="l",col=c("black","red","green"),lty=1,lwd=3)



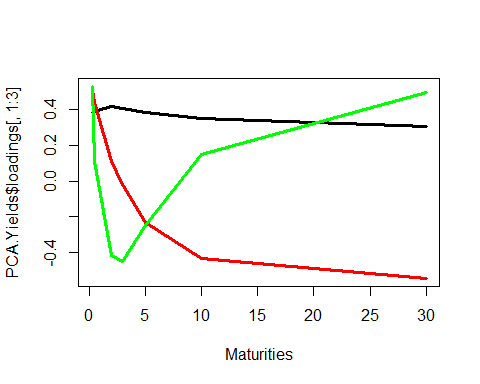
matplot(PCA.Yields$scores[,1:3],type="l",col=c("black","red","green"),lwd=3,lty=1)



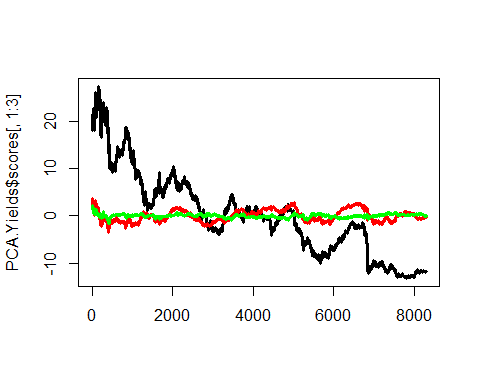
As expected, these are the same graphs that were seen earlier.

Change the signs fo the first factor and factor loading again.

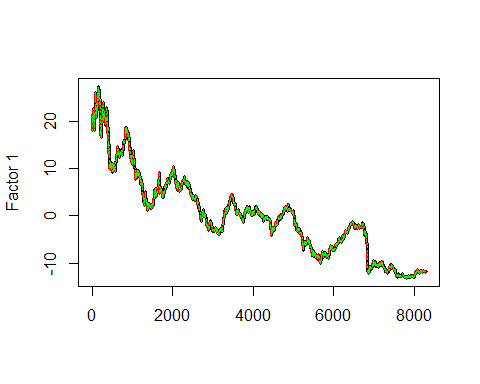
PCA.Yields$loadings[,1]<--PCA.Yields$loadings[,1]  
PCA.Yields$scores[,1]<--PCA.Yields$scores[,1]  
matplot(Maturities,PCA.Yields$loadings[,1:3],type="l",col=c("black","red","green"),lty=1,lwd=3)



matplot(PCA.Yields$scores[,1:3],type="l",col=c("black","red","green"),lwd=3,lty=1)



matplot(cbind(PCA.Yields$scores[,1],AssignmentData.Output,Factors[,1]),type="l",col=c("black","red","green"),lwd=c(3,2,1),lty=c(1,2,3),ylab="Factor 1")

 OMG!!OUTPUT1 is the first eigenvector!!

Compare the regression coefficients from Step 2 and Step 3 with factor loadings. First, look at the slopes for AssignmentData.Input~AssignmentData.Output

t(apply(AssignmentData, 2, function(AssignmentData.col) lm(AssignmentData.col~AssignmentData.Output)$coef))

## (Intercept) AssignmentData.Output  
## USGG3M 4.675134 0.3839609  
## USGG6M 4.844370 0.3901870  
## USGG2YR 5.438888 0.4151851  
## USGG3YR 5.644458 0.4063541  
## USGG5YR 6.009421 0.3860610  
## USGG10YR 6.481316 0.3477544  
## USGG30YR 6.869355 0.3047124

cbind(PCA.Yields$center,PCA.Yields$loadings[,1])

## [,1] [,2]  
## USGG3M 4.675134 0.3839609  
## USGG6M 4.844370 0.3901870  
## USGG2YR 5.438888 0.4151851  
## USGG3YR 5.644458 0.4063541  
## USGG5YR 6.009421 0.3860610  
## USGG10YR 6.481316 0.3477544  
## USGG30YR 6.869355 0.3047124

This shows that the zero loading equals the vector of intercepts of models Y~Output1, where Y is one of the columns of yields in the data. Also, the slopes of the same models are equal to the first loading.

Check if the same is true in the opposite direction: is there a correspondence between the coefficients of models Output1~Yield and the first loading? Yes there is, as shall be seen below.

AssignmentData.Centered<-t(apply(AssignmentData,1,function(AssignmentData.row) AssignmentData.row-PCA.Yields$center))  
dim(AssignmentData.Centered)

## [1] 8300 7

t(apply(AssignmentData.Centered, 2, function(AssignmentData.col) lm(AssignmentData.Output~AssignmentData.col)$coef))

## (Intercept) AssignmentData.col  
## USGG3M 1.420077e-11 2.507561  
## USGG6M 1.421187e-11 2.497235  
## USGG2YR 1.419747e-11 2.400449  
## USGG3YR 1.419989e-11 2.455793  
## USGG5YR 1.419549e-11 2.568742  
## USGG10YR 1.420297e-11 2.786991  
## USGG30YR 1.420965e-11 3.069561

To recover the loading of the first factor by doing regression, use all inputs togehter.

t(lm(AssignmentData.Output~AssignmentData.Centered)$coef)[-1]

## [1] 0.3839609 0.3901870 0.4151851 0.4063541 0.3860610 0.3477544 0.3047124

PCA.Yields$loadings[,1]

## USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR   
## 0.3839609 0.3901870 0.4151851 0.4063541 0.3860610 0.3477544 0.3047124

This means that the factor is a portfolio of all input variables with weights!