Final Project

Weijie Gao

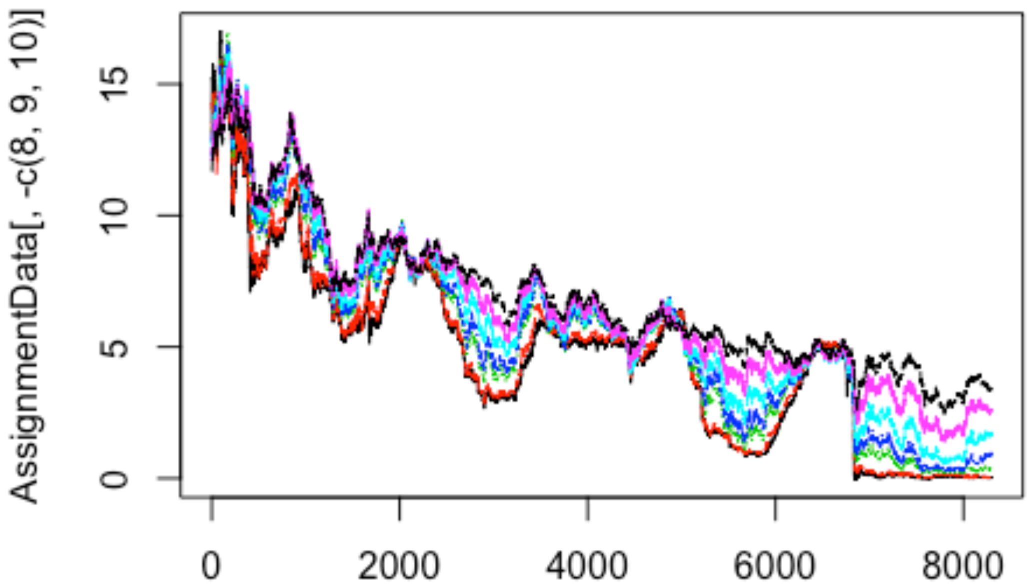
22 November 2016

Step 1

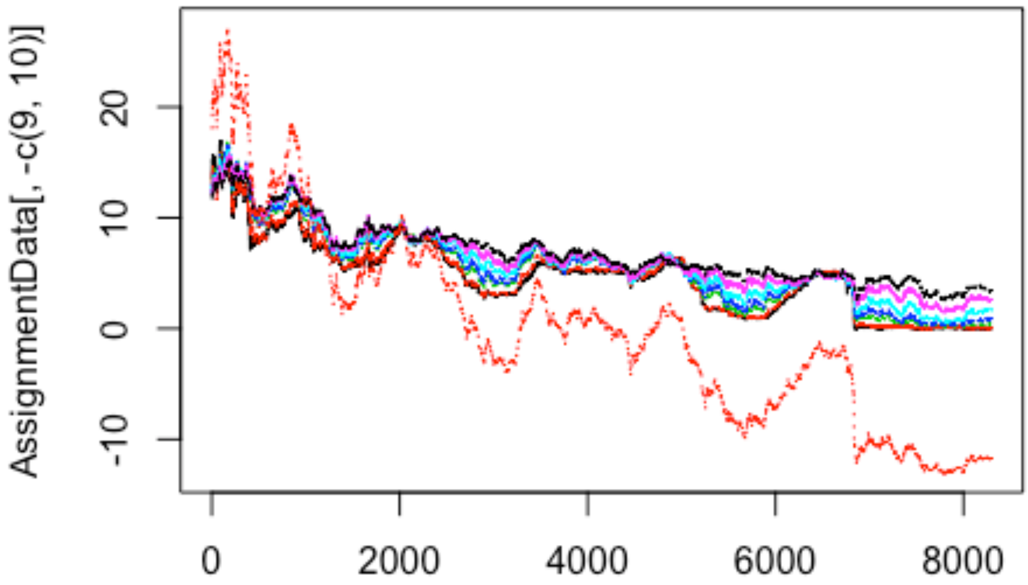
```
datapath <- "~/Documents/Chicago2016/Statistical Analysis/Project"
AssignmentData<-read.csv(file=paste(datapath,"regressionassignmentdata2
014.csv",sep="/"), row.names=1,header=TRUE,sep=",")
head(AssignmentData)
```

```
##            USGG3M USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  O
utput1
## 1/5/1981   13.52  13.09  12.289   12.28  12.294   12.152   11.672 18
.01553
## 1/6/1981   13.58  13.16  12.429   12.31  12.214   12.112   11.672 18
.09140
## 1/7/1981   14.50  13.90  12.929   12.78  12.614   12.382   11.892 19
.44731
## 1/8/1981   14.76  14.00  13.099   12.95  12.684   12.352   11.912 19
.74851
## 1/9/1981   15.20  14.30  13.539   13.28  12.884   12.572   12.132 20
.57204
## 1/12/1981  15.22  14.23  13.179   12.94  12.714   12.452   12.082 20
.14218
##            Easing Tightening
## 1/5/1981       NA         NA
## 1/6/1981       NA         NA
## 1/7/1981       NA         NA
## 1/8/1981       NA         NA
## 1/9/1981       NA         NA
## 1/12/1981      NA         NA
```

```
# Plot the input variables.
matplot(AssignmentData[,-c(8,9,10)],type='l')
```




```r
# Plot the input variables together with the output variable.
matplot(AssignmentData[,-c(9,10)],type='l')
```

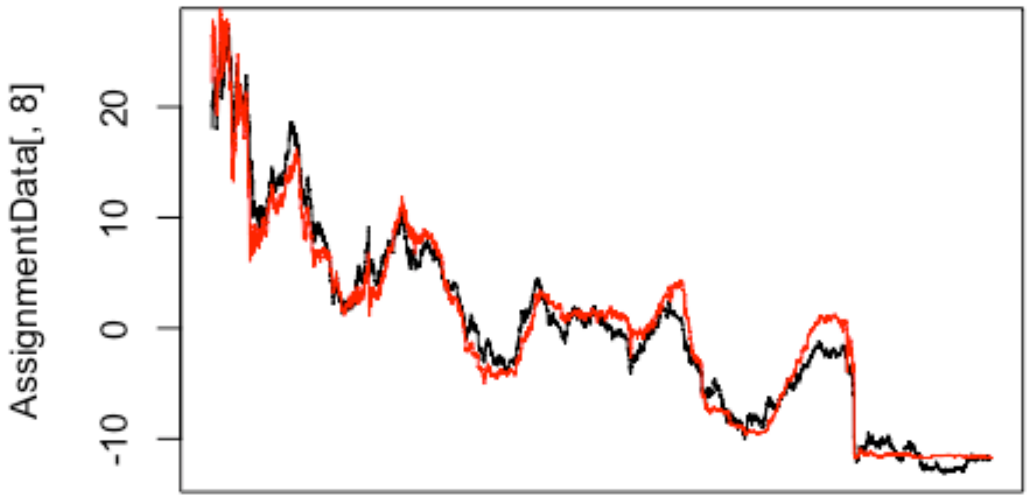


Step 2

Estimate simple regression model with each of the input variables and the output variable given in AssignmentData.

```
# Fit linear regression model with input variable USGG3M and Output1
Input1.linear.Model <- lm(AssignmentData[,8]~AssignmentData[,1])
# Check the summary
summary(Input1.linear.Model)

##
## Call:
## lm(formula = AssignmentData[, 8] ~ AssignmentData[, 1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9374 -1.2115 -0.0528  1.2640  7.7048
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -11.72318    0.03137  -373.7   <2e-16 ***
## AssignmentData[, 1]   2.50756    0.00541   463.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.69 on 8298 degrees of freedom
## Multiple R-squared:  0.9628, Adjusted R-squared:  0.9628
## F-statistic: 2.148e+05 on 1 and 8298 DF,  p-value: < 2.2e-16

# Check available names of fields returned by lm() and summary() functi
ons
names(Input1.linear.Model)

##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"

names(summary(Input1.linear.Model))

##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"

# Check the amount of correlation explained
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(I
nput1.linear.Model)$sigma^2)

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

# Return r.square of fitted model
summary(Input1.linear.Model)$r.squared

## [1] 0.9628054

# Return the estimated parameters
Coefficients.Input1 <- Input1.linear.Model$coefficients
Coefficients.Input1

##         (Intercept) AssignmentData[, 1]
##          -11.723184            2.507561

# Plot the output variable together with the fitted values.
matplot(AssignmentData[,8],type="l",xaxt="n")
lines(Input1.linear.Model$fitted.values,col="red")
```



From the result of summary table, the slope of fitted linear regression is positive, showing that the Input1 and Output1 are positively related. And both of the P values of t statistic are smaller than the significance level of 0.05, hence the intercept and slope is significant. And as the r square equals to 0.9628, it shows that the model explains 96.28% the variability of the response data around its mean. And in this case, it indicates that the model is a good fit.

```r
# Fit linear regression model with input variable USGG6M and Output1
Input2.linear.Model <- lm(AssignmentData[,8]~AssignmentData[,2])
# Check the summary
summary(Input2.linear.Model)
```

```
##
## Call:
## lm(formula = AssignmentData[, 8] ~ AssignmentData[, 2])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7529 -1.0385  0.0224  1.1443  4.1509
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -12.097528   0.026469  -457.0   <2e-16 ***
## AssignmentData[, 2]    2.497235   0.004445   561.9   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.403 on 8298 degrees of freedom
## Multiple R-squared:  0.9744, Adjusted R-squared:  0.9744
## F-statistic: 3.157e+05 on 1 and 8298 DF,  p-value: < 2.2e-16
```

```
# Check the amount of correlation explained
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(I
nput2.linear.Model)$sigma^2)
```

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

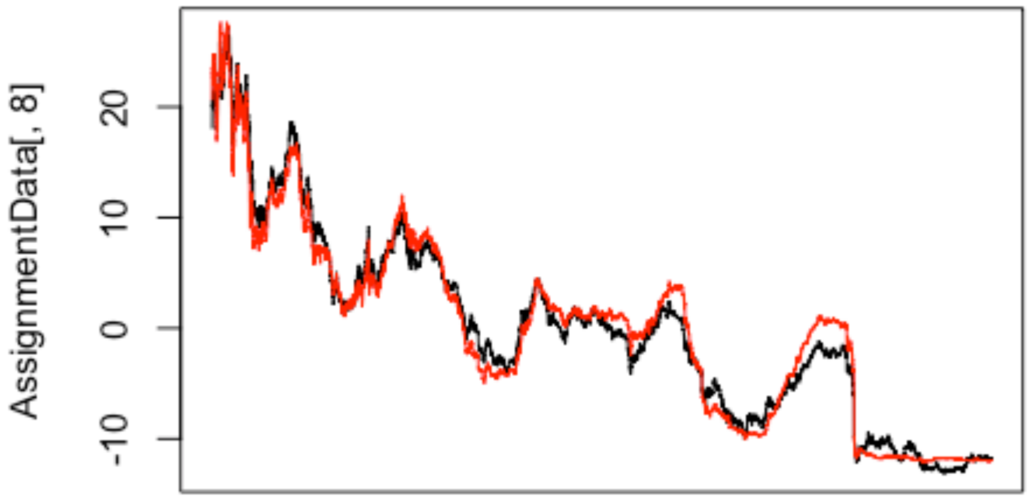
```
# Return r.square of fitted model
summary(Input2.linear.Model)$r.squared
```

```
## [1] 0.9743884
```

```
# Return the estimated parameters
Coefficients.Input2 <- Input2.linear.Model$coefficients
Coefficients.Input2
```

```
##         (Intercept) AssignmentData[, 2]
##          -12.097528            2.497235
```

```
# Plot the output variable together with the fitted values.
matplot(AssignmentData[,8],type="l",xaxt="n")
lines(Input2.linear.Model$fitted.values,col="red")
```



From the result of summary table, the slope of fitted linear regression is positive, showing that the Input2 and Output1 are positively related. And both of the P values of t statistic are smaller than the significance level of 0.05, hence the intercept and slope is significant. And as the r square equals to 0.9743, the model explains 97.43% the variability of the response data around its mean. Also, it shows that the model is a good fit.

```
# Fit linear regression model with input variable USGG2YR and Output1
Input3.linear.Model <- lm(AssignmentData[,8]~AssignmentData[,3])
# Check the summary
summary(Input3.linear.Model)

##
## Call:
## lm(formula = AssignmentData[, 8] ~ AssignmentData[, 3])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.43277 -0.38356 -0.00578  0.43362  1.72564
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -13.055775   0.010031   -1302   <2e-16 ***
## AssignmentData[, 3]     2.400449   0.001532    1567   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5087 on 8298 degrees of freedom
## Multiple R-squared:  0.9966, Adjusted R-squared:  0.9966
## F-statistic: 2.455e+06 on 1 and 8298 DF,  p-value: < 2.2e-16
```

```
# Check the amount of correlation explained
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(I
nput3.linear.Model)$sigma^2)
```

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

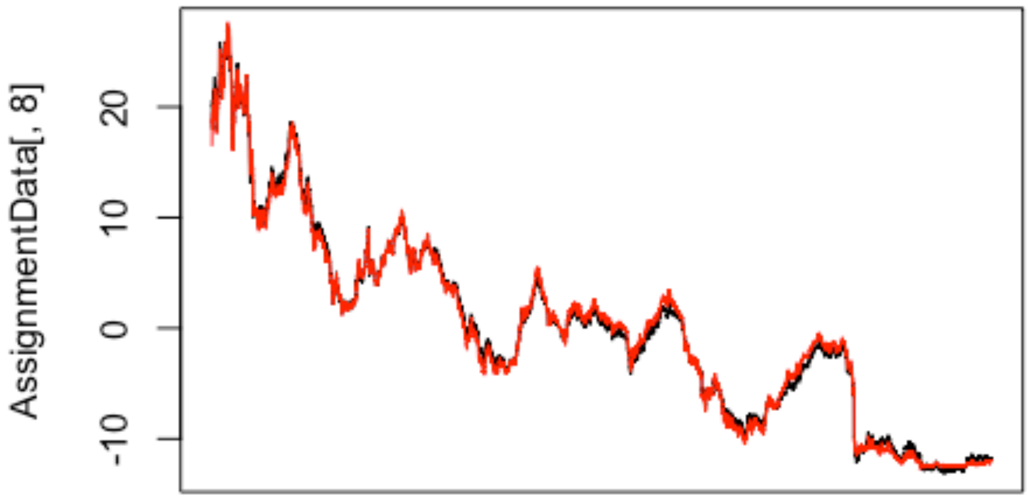
```
# Return r.square of fitted model
summary(Input3.linear.Model)$r.squared
```

```
## [1] 0.9966307
```

```
# Return the estimated parameters
Coefficients.Input3 <- Input3.linear.Model$coefficients
Coefficients.Input3
```

```
##         (Intercept) AssignmentData[, 3]
##          -13.055775            2.400449
```

```
# Plot the output variable together with the fitted values.
matplot(AssignmentData[,8],type="l",xaxt="n")
lines(Input3.linear.Model$fitted.values,col="red")
```



Similarly, the slope of fitted linear regression is positive, showing that the Input3 and Output1 are positively related. And both of the P values of t statistic are smaller than the significance level of 0.05, hence the intercept and slope is significant. And as the r square equals to 0.9966, it shows that the model explains 99.66% the variability of the response data around its mean. And in this case, it shows that the model is a good fit.

```
# Fit linear regression model with input variable USGG23R and Output1
Input4.linear.Model <- lm(AssignmentData[,8]~AssignmentData[,4])
# Check the summary
summary(Input4.linear.Model)

##
## Call:
## lm(formula = AssignmentData[, 8] ~ AssignmentData[, 4])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0160 -0.2459  0.0325  0.2638  3.0666
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -13.861618   0.008214   -1688   <2e-16 ***
## AssignmentData[, 4]   2.455793   0.001230    1996   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3996 on 8298 degrees of freedom
## Multiple R-squared:  0.9979, Adjusted R-squared:  0.9979
## F-statistic: 3.984e+06 on 1 and 8298 DF,  p-value: < 2.2e-16
```

```
# Check the amount of correlation explained
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(I
nput4.linear.Model)$sigma^2)
```

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

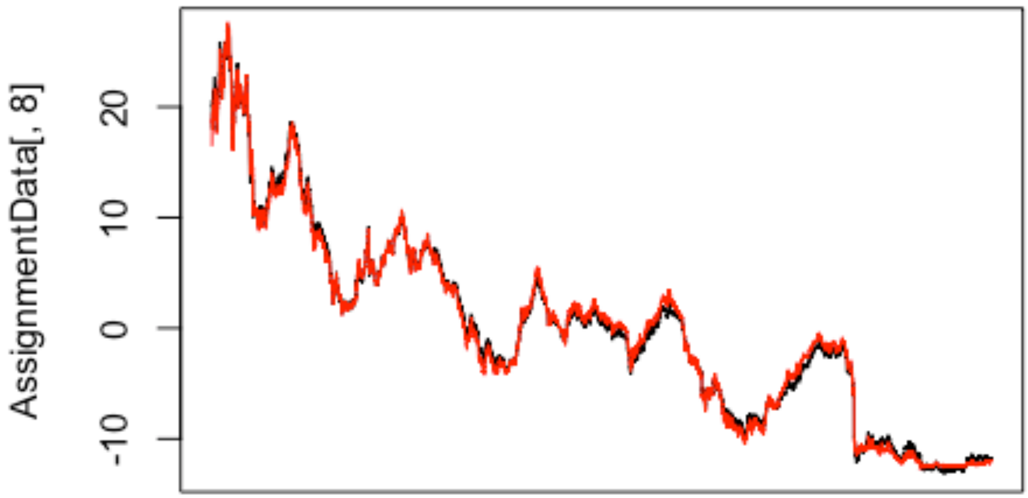
```
# Return r.square of fitted model
summary(Input4.linear.Model)$r.squared
```

```
## [1] 0.9979215
```

```
# Return the estimated parameters
Coefficients.Input4 <- Input4.linear.Model$coefficients
Coefficients.Input4
```

```
##         (Intercept) AssignmentData[, 4]
##          -13.861618            2.455793
```

```
# Plot the output variable together with the fitted values.
matplot(AssignmentData[,8],type="l",xaxt="n")
lines(Input3.linear.Model$fitted.values,col="red")
```



From the summary table, the slope of fitted linear regression is positive, showing that the Input4 and Output1 are positively related. And both of the P values of t statistic are smaller than the significance level of 0.05, hence the intercept and slope is significant.And as the r square equals to 0.9979, it shows that the model explains 99.79% the variability of the response data around its mean. And in this case, it appears that the model is a good fit.

```
# Fit linear regression model with input variable USGG5YR and Output1
Input5.linear.Model <- lm(AssignmentData[,8]~AssignmentData[,5])
# Check the summary
summary(Input5.linear.Model)
```

```
##
## Call:
## lm(formula = AssignmentData[, 8] ~ AssignmentData[, 5])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6517 -0.6216 -0.0147  0.6523  3.1720
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -15.436649   0.017819  -866.3   <2e-16 ***
## AssignmentData[, 5]   2.568742   0.002581   995.2   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7989 on 8298 degrees of freedom
## Multiple R-squared:  0.9917, Adjusted R-squared:  0.9917
## F-statistic: 9.904e+05 on 1 and 8298 DF,  p-value: < 2.2e-16
```

```r
# Check the amount of correlation explained
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(Input5.linear.Model)$sigma^2)
```

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

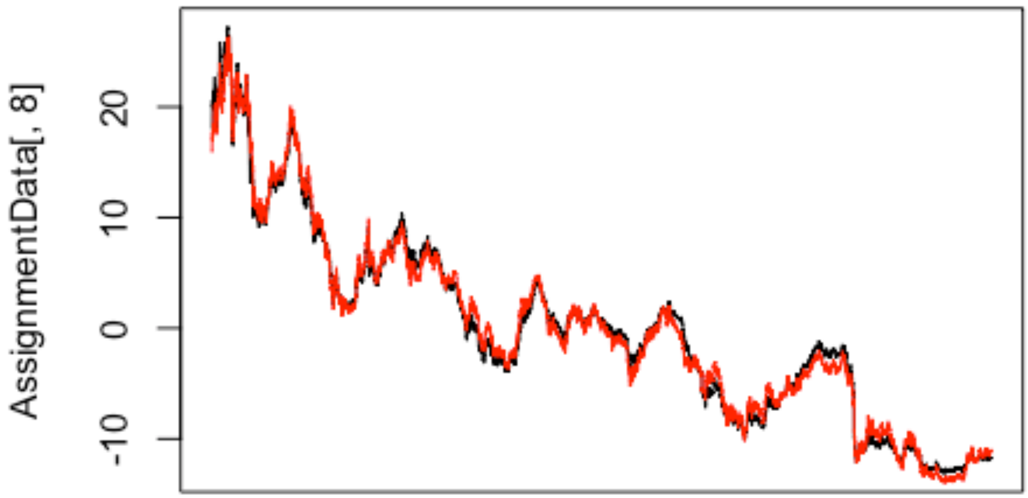
```r
# Return r.square of fitted model
summary(Input5.linear.Model)$r.squared
```

```
## [1] 0.9916908
```

```r
# Return the estimated parameters
Coefficients.Input5 <- Input5.linear.Model$coefficients
Coefficients.Input5
```

```
##         (Intercept) AssignmentData[, 5]
##          -15.436649            2.568742
```

```r
# Plot the output variable together with the fitted values.
matplot(AssignmentData[,8],type="l",xaxt="n")
lines(Input5.linear.Model$fitted.values,col="red")
```



From the summary table, the slope of fitted linear regression is positive, showing that the Input5 and Output1 are positively related. And both of the P values of t statistic are smaller than the significance level of 0.05, hence the intercept and slope is significant.And as the r square equals to 0.9917, it shows that the model explains 99.17% the variability of the response data around its mean. And in this case, it appears that the model is a good fit.

```
# Fit linear regression model with input variable USGG10YR and Output1
Input6.linear.Model <- lm(AssignmentData[,8]~AssignmentData[,6])
Input6.linear.Model$coefficients

##         (Intercept) AssignmentData[, 6]
##          -18.063370            2.786991

# Check the summary
summary(Input6.linear.Model)

##
## Call:
## lm(formula = AssignmentData[, 8] ~ AssignmentData[, 6])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0334 -1.2810 -0.1944  1.3561  4.2254
```

```
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -18.063370   0.039182  -461.0   <2e-16 ***
## AssignmentData[, 6]   2.786991   0.005455   510.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.538 on 8298 degrees of freedom
## Multiple R-squared:  0.9692, Adjusted R-squared:  0.9692
## F-statistic: 2.61e+05 on 1 and 8298 DF,  p-value: < 2.2e-16
```

```r
# Check the amount of correlation explained
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(I
nput6.linear.Model)$sigma^2)
```

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

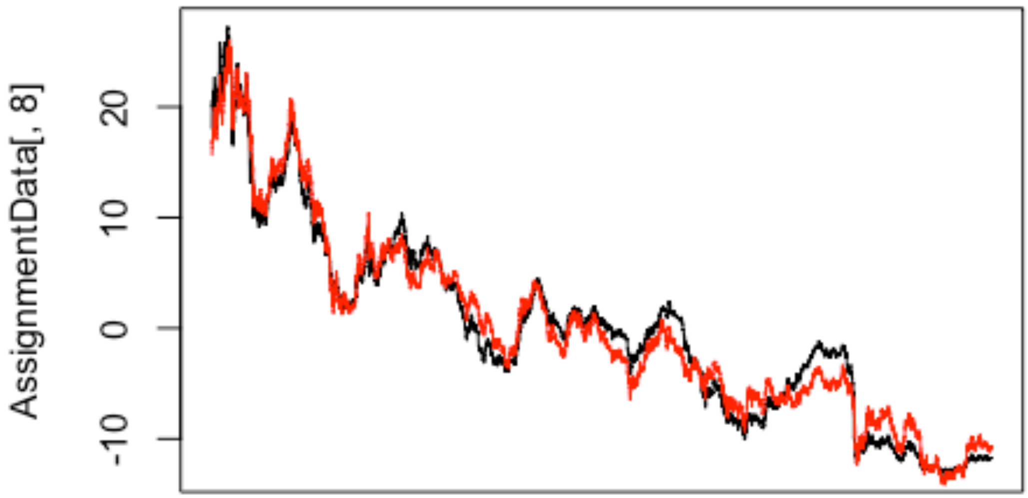
```r
# Return r.square of fitted model
summary(Input6.linear.Model)$r.squared
```

```
## [1] 0.9691884
```

```r
# Return the estimated parameters
Coefficients.Input6 <- Input6.linear.Model$coefficients
Coefficients.Input6
```

```
##         (Intercept) AssignmentData[, 6]
##          -18.063370            2.786991
```

```r
# Plot the output variable together with the fitted values.
matplot(AssignmentData[,8],type="l",xaxt="n")
lines(Input6.linear.Model$fitted.values,col="red")
```



From the summary table, the slope of fitted linear regression is positive, showing that the Input6 and Output1 are positively related. And both of the P values of t statistic are smaller than the significance level of 0.05, hence the intercept and slope is significant.And as the r square equals to 0.9692, it shows that the model explains 96.92% the variability of the response data around its mean. And in this case, it indicates that the model is a good fit.

```
# Fit linear regression model with input variable USGG10YR and Output1
Input7.linear.Model <- lm(AssignmentData[,8]~AssignmentData[,7])
Input7.linear.Model$coefficients

##         (Intercept) AssignmentData[, 7]
##          -21.085905            3.069561

# Check the summary
summary(Input7.linear.Model)

##
## Call:
## lm(formula = AssignmentData[, 8] ~ AssignmentData[, 7])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6447 -1.8426 -0.4731  1.9529  4.8734
```

```
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -21.08591    0.06560  -321.4   <2e-16 ***
## AssignmentData[, 7]    3.06956    0.00886   346.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.229 on 8298 degrees of freedom
## Multiple R-squared:  0.9353, Adjusted R-squared:  0.9353
## F-statistic: 1.2e+05 on 1 and 8298 DF,  p-value: < 2.2e-16
```

```
# Check the amount of correlation explained
c(Total.Variance=var(AssignmentData[,8]),Unexplained.Variance=summary(I
nput7.linear.Model)$sigma^2)
```

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

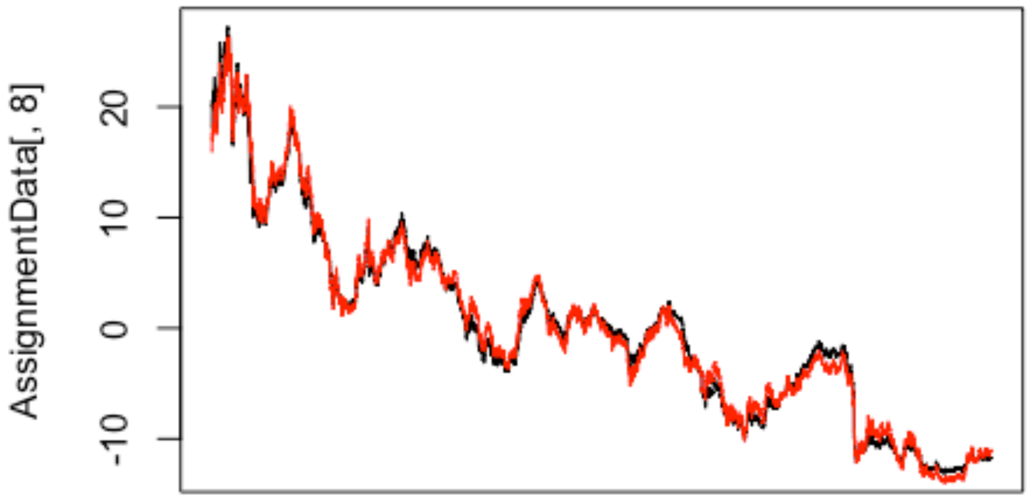
```
# Return r.square of fitted model
summary(Input7.linear.Model)$r.squared
```

```
## [1] 0.9353333
```

```
# Return the estimated parameters
Coefficients.Input7 <- Input7.linear.Model$coefficients
Coefficients.Input7
```

```
##         (Intercept) AssignmentData[, 7]
##          -21.085905            3.069561
```

```
# Plot the output variable together with the fitted values.
matplot(AssignmentData[,8],type="l",xaxt="n")
lines(Input5.linear.Model$fitted.values,col="red")
```



From the summary table, the slope of fitted linear regression is positive, showing that the Input7 and Output1 are positively related. And both of the P values of t statistic are smaller than the significance level of 0.05, hence the intercept and slope is significant.And as the r square equals to 0.9353, it shows that the model explains 93.53% the variability of the response data around its mean. And in this case, it appears that the model is a good fit.

```
# Collect all slopes and intercepts in one table and print this table.
Coefficients_table <- sapply(1:7, function(i) lm(AssignmentData[,8]~Ass
ignmentData[,i])$coefficients)
rownames(Coefficients_table) <- c("Intercept","Slope")
colnames(Coefficients_table) <- colnames(AssignmentData[,-c(8,9,10)])
Coefficients_table
```

```
##                USGG3M     USGG6M    USGG2YR    USGG3YR    USGG5YR
## Intercept -11.723184 -12.097528 -13.055775 -13.861618 -15.436649
## Slope       2.507561   2.497235   2.400449   2.455793   2.568742
##              USGG10YR   USGG30YR
## Intercept -18.063370 -21.085905
## Slope       2.786991   3.069561
```

Step 3.

```
# Fit linear regression models using single output (column 8 Output1) a
s input and each of the original inputs as outputs.
```

```r
# Collect all slopes and intercepts in one table and print this table.
Coefficients_table2 <- sapply(1:7, function(i) lm(AssignmentData[,i]~As
signmentData[,8])$coefficients)
rownames(Coefficients_table2) <- c("Intercept","Slope")
colnames(Coefficients_table2) <- colnames(AssignmentData[,-c(8,9,10)])
Coefficients_table2
```

```
##              USGG3M    USGG6M    USGG2YR    USGG3YR   USGG5YR   USGG10YR
## Intercept 4.6751341 4.844370 5.4388879 5.6444580 6.009421 6.4813160
## Slope     0.3839609 0.390187 0.4151851 0.4063541 0.386061 0.3477544
##             USGG30YR
## Intercept 6.8693554
## Slope     0.3047124
```

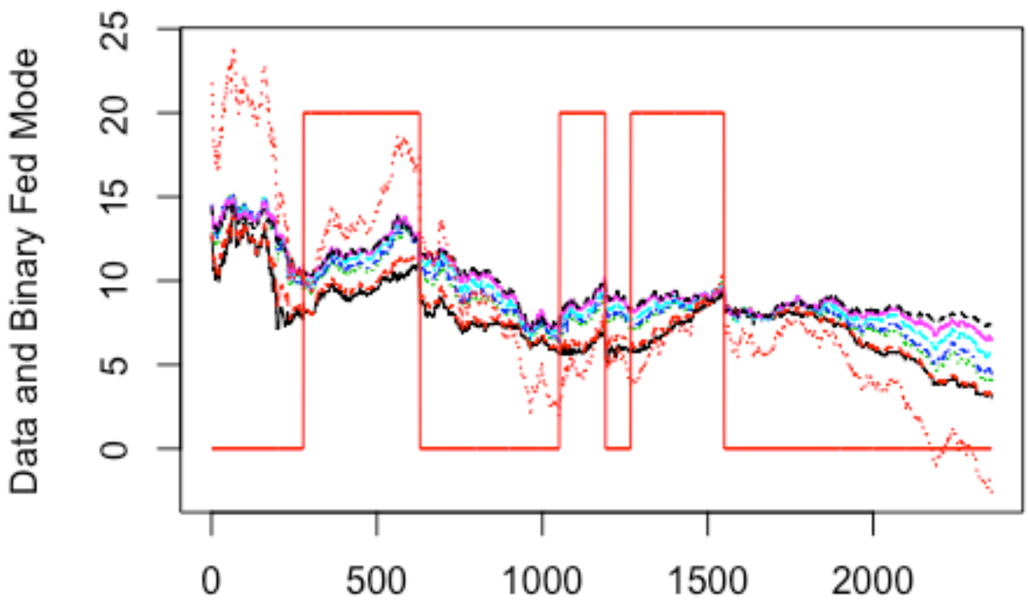
Step 4

```r
AssignmentDataLogistic<-data.matrix(AssignmentData,rownames.force="auto
matic")

# Create columns of easing periods (as 0s) and tightening periods (as 1
s)
EasingPeriods<-AssignmentDataLogistic[,9]
EasingPeriods[AssignmentDataLogistic[,9]==1]<-0
TighteningPeriods<-AssignmentDataLogistic[,10]
# Check easing and tightening periods
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
```

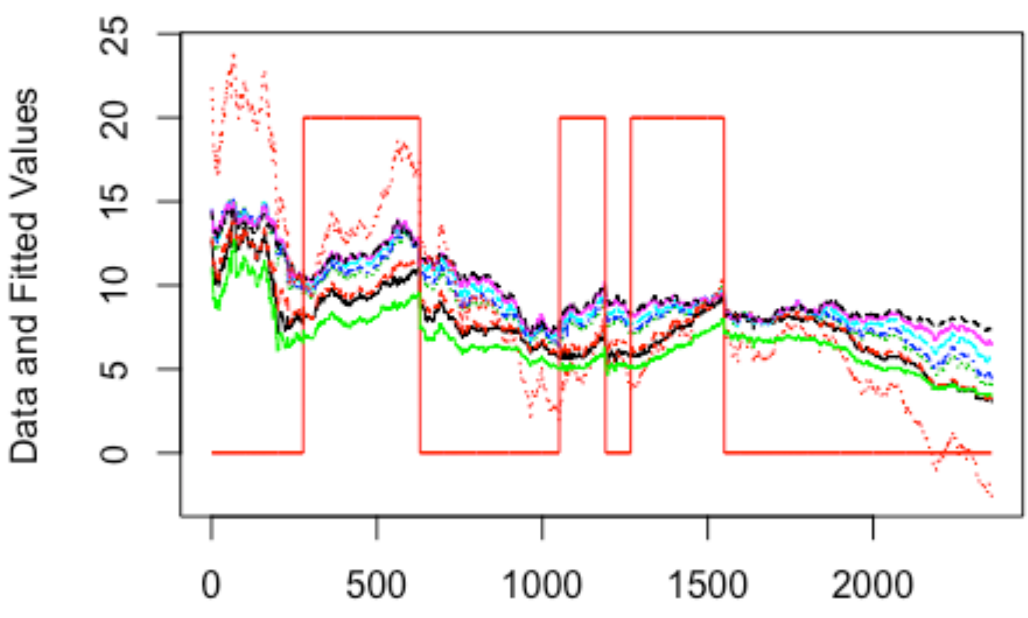
```
# Remove the periods of neither easing nor tightening
All.NAs<-is.na(EasingPeriods)&is.na(TighteningPeriods)
AssignmentDataLogistic.EasingTighteningOnly<-AssignmentDataLogistic
AssignmentDataLogistic.EasingTighteningOnly[,9]<-EasingPeriods
AssignmentDataLogistic.EasingTighteningOnly<-AssignmentDataLogistic.Eas
ingTighteningOnly[!All.NAs,]
AssignmentDataLogistic.EasingTighteningOnly[is.na(AssignmentDataLogisti
c.EasingTighteningOnly[,10]),10]<-0
# Binary output for logistic regression is now in column 10

matplot(AssignmentDataLogistic.EasingTighteningOnly[,-c(9,10)],type="l"
,ylab="Data and Binary Fed Mode")
lines(AssignmentDataLogistic.EasingTighteningOnly[,10]*20,col="red")
```



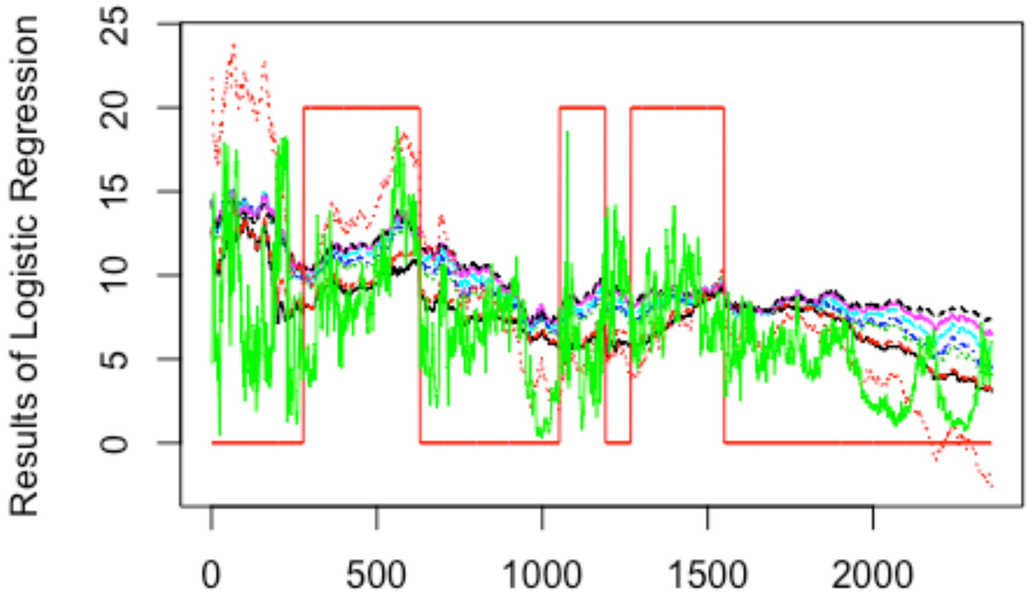

```
# Estimate logistic regression with 3M yields as predictors for easing/
tightening output.

LogisticModel.TighteningEasing_3M<-glm(AssignmentDataLogistic.EasingTig
htenningOnly[,10]~
                                AssignmentDataLogistic.EasingTightening
Only[,1],family=binomial(link=logit))
summary(LogisticModel.TighteningEasing_3M)

##
## Call:
## glm(formula = AssignmentDataLogistic.EasingTighteningOnly[, 10] ~
##     AssignmentDataLogistic.EasingTighteningOnly[, 1], family = binom
ial(link = logit))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4239  -0.9014  -0.7737   1.3548   1.6743
##
## Coefficients:
##                                                  Estimate Std. Error
## (Intercept)                                      -2.15256    0.17328
## AssignmentDataLogistic.EasingTighteningOnly[, 1]  0.18638    0.02144
```

```
##                                                          z value Pr(>|z|)

## (Intercept)                                              -12.422   <2e-16 **
*
## AssignmentDataLogistic.EasingTighteningOnly[, 1]   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
```

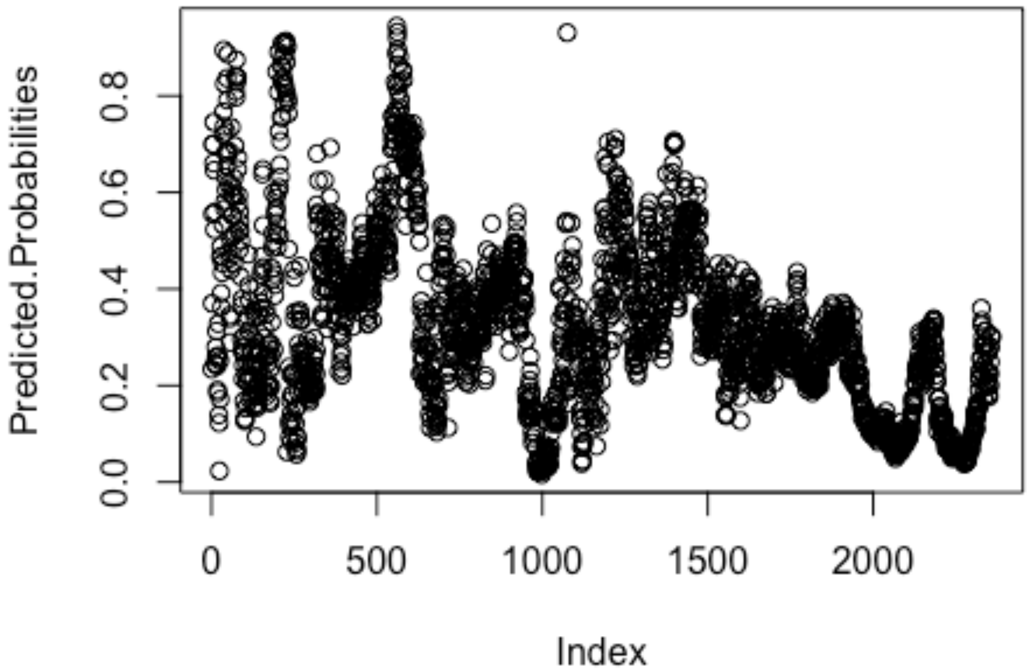
```
matplot(AssignmentDataLogistic.EasingTighteningOnly[,-c(9,10)],type="l"
,ylab="Data and Fitted Values")
lines(AssignmentDataLogistic.EasingTighteningOnly[,10]*20,col="red")
lines(LogisticModel.TighteningEasing_3M$fitted.values*20,col="green")
```



```
# Use all inputs as predictors for logistic regression.
LogisticModel.TighteningEasing_All<-glm(AssignmentDataLogistic.EasingTi
ghteningOnly[,10]~
                                AssignmentDataLogistic.EasingTightening
Only[,1]+AssignmentDataLogistic.EasingTighteningOnly[,2] +AssignmentDat
aLogistic.EasingTighteningOnly[,3]+AssignmentDataLogistic.EasingTighten
ingOnly[,4]
+AssignmentDataLogistic.EasingTighteningOnly[,5]+AssignmentDataLogistic
.EasingTighteningOnly[,6]
+AssignmentDataLogistic.EasingTighteningOnly[,7],family=binomial(link=l
ogit))

# Explore the estimated model
summary(LogisticModel.TighteningEasing_All)$aic

## [1] 2645.579

summary(LogisticModel.TighteningEasing_All)$coefficients[,c(1,4)]

##                                                     Estimate     Pr(>
|z|)
## (Intercept)                                       -4.7551928 2.784283
e-28
## AssignmentDataLogistic.EasingTighteningOnly[, 1] -3.3456116 4.073045
e-36
## AssignmentDataLogistic.EasingTighteningOnly[, 2]  4.1558535 1.422964
e-28
## AssignmentDataLogistic.EasingTighteningOnly[, 3]  3.9460296 1.751687
e-07
## AssignmentDataLogistic.EasingTighteningOnly[, 4] -3.4642455 2.080617
e-04
## AssignmentDataLogistic.EasingTighteningOnly[, 5] -3.2115319 3.786229
e-05
## AssignmentDataLogistic.EasingTighteningOnly[, 6] -0.9705444 3.202140
e-01
## AssignmentDataLogistic.EasingTighteningOnly[, 7]  3.3253517 6.036041
e-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")
```




```
# Verify that the fitted values are the predicted probabilityies of the
 logistic regression
head(LogisticModel.TighteningEasing_All$fitted.values)

##  11/2/1981  11/4/1981  11/5/1981  11/6/1981  11/9/1981 11/10/1981
##  0.2339726  0.3701609  0.5553356  0.6996382  0.7000135  0.6607022

Predicted.Probabilities<- predict(LogisticModel.TighteningEasing_All,ty
pe="response")
head(Predicted.Probabilities)

##  11/2/1981  11/4/1981  11/5/1981  11/6/1981  11/9/1981 11/10/1981
##  0.2339726  0.3701609  0.5553356  0.6996382  0.7000135  0.6607022

plot(Predicted.Probabilities)
```



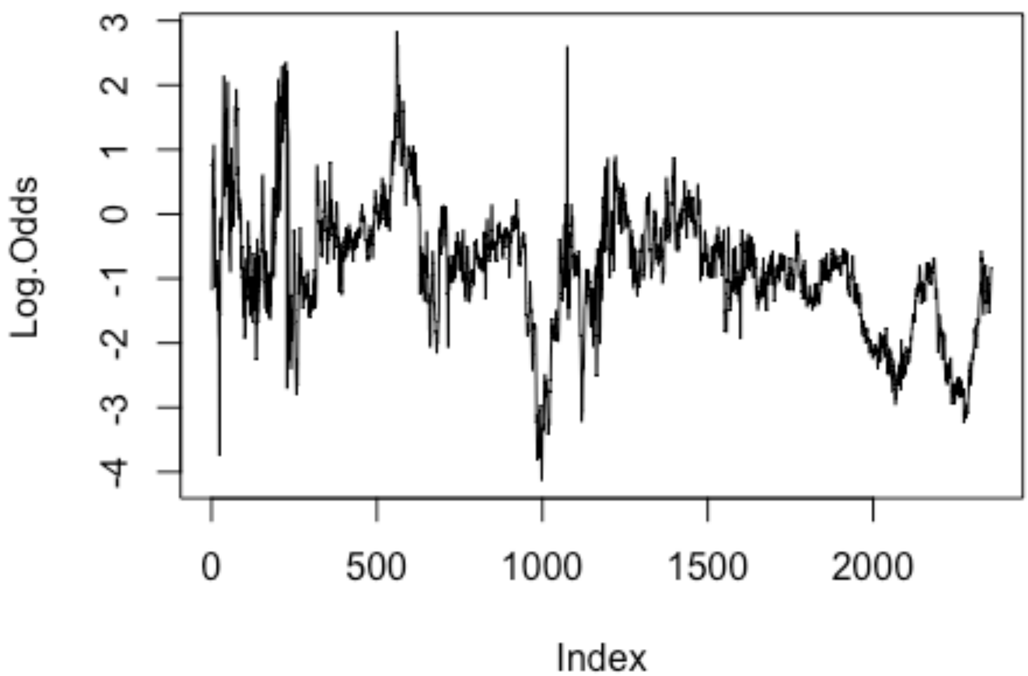
Interpret the coefficients of the model and the fitted values

The estimate intercept is -4.7551928, and it is the expected value of the log-odds of Output1 when all of the predictor variables equal zero. The parameter estimate for the variable USGG3M is -3.3456116. This means that for a one-unit increase in USGG3M, we expect a 3.3456116 decrease in the log-odds of the dependent variable Output1, holding all other independent variables constant.

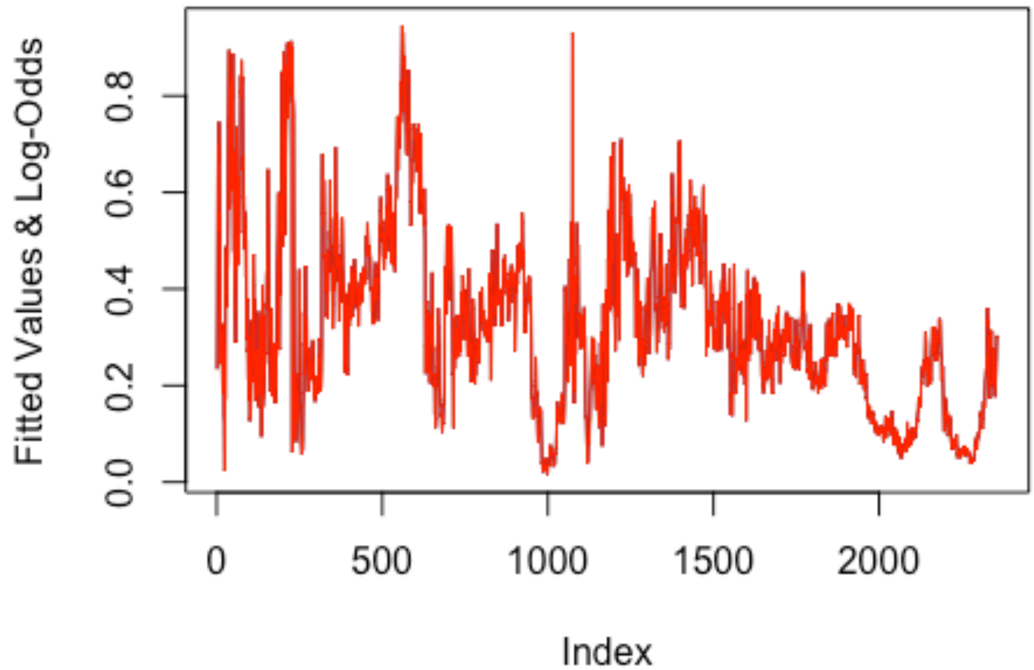
And the parameter estimate for the variable USGG6M is 4.1558535. This means that for a one-unit increase in USGG6M, we expect a 4.1558535 increase in the log-odds of the dependent variable Output1, holding all other independent variables constant. Similarly, the rest five coefficients of variables in our model can be interpreted in the same way.

And based on the code above, it could be verified that the fitted values are the predicted probabilities of logistic regression, and by comparing the probability plot and the periods of easing and tightening period, we could notice that a comparatively high probability (maybe we could define it to be larger than 0.5) corresponds to the tightening period and a comparatively low probability (smaller than 0.5) corresponds to the easing period. Also, it could be noticed that the predicted results for years after 1989 is more accurate than the predicted results for years before that.

```r
# Calculate and plot log-odds and probabilities. Compare probabilities with fitted values.
Log.Odds<-predict(LogisticModel.TighteningEasing_All)
plot(Log.Odds,type="l")
```




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



Step 5

```
# In this part, we compare linear regression models with different comb
inations of predictors and select the best combination.
AssignmentDataRegressionComparison<-data.matrix(AssignmentData[,-c(9,10
)],rownames.force="automatic")
AssignmentDataRegressionComparison<-AssignmentData[,-c(9,10)]

# Estimate the full model by using all 7 predictors
RegressionModelComparison.Full<-lm(Output1~1+USGG3M+USGG6M+USGG2YR+USGG
3YR+USGG5YR+USGG10YR+USGG30YR,
                                   data=AssignmentDataRegressionCompari
son)
# Check the coefficients
(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

# Check the R2
(R2 <- summary(RegressionModelComparison.Full)$r.squared)

## [1] 1

# Check the adjusted R2
Adjusted.R2 <- summary(RegressionModelComparison.Full)$adj.r.squared
c(R2 = R2, Adjusted.R2 = Adjusted.R2)

##          R2 Adjusted.R2
##           1           1

# Check the degrees of freedom
summary(RegressionModelComparison.Full)$df

## [1]    8 8292    8
```

Intepret the fitted model. How good is the fit? How significant are the parameters?

Since we use all the 7 inputs to construct the model, the result is expected to be good. And from the results, all the p value of t statistic are 0, less than the significance level of 0.05, hence we may say they are all significant. And since both R square and adjusted.R2 are equal to 1, it appears that the model is perfectly fitted. But this may also give us a hint that the model could be overfitted.

```
# Estimate the Null model by including only intercept.
RegressionModelComparison.Null<-lm(Output1~1,data=AssignmentDataRegress
ionComparison)
summary(RegressionModelComparison.Null)

##
## Call:
## lm(formula = Output1 ~ 1, data = AssignmentDataRegressionComparison)
##
## 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

# Check the coefficients
summary(RegressionModelComparison.Null)$coefficients

##                 Estimate Std. Error      t value Pr(>|t|)
## (Intercept) 1.420082e-11 0.09619536 1.476248e-10        1
```

```r
# Check the R2
R2 <- summary(RegressionModelComparison.Null)$r.squared

# Check the adjusted R2
Adjusted.R2 <- summary(RegressionModelComparison.Null)$adj.r.squared
c(R2 = R2, Adjusted.R2 = Adjusted.R2)

##           R2 Adjusted.R2
##            0           0

# Check the degrees of freedom
summary(RegressionModelComparison.Null)$df

## [1]    1 8299    1
```

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

```r
Output1 <- AssignmentDataRegressionComparison[,8]
ybar <- mean(Output1)
yhat <- RegressionModelComparison.Null$fitted.values
# Claculate the regression sum of square(SSR)
(SSR <- sum((yhat-ybar)^2))

## [1] 7.10687e-24

# sum of squares of the residual error(SSE)
(SSE <- sum((Output1-yhat)^2))

## [1] 637400

(rsquare <- 1-  SSE/(SSE+SSR))

## [1] 0
```

Recall that the definition of R2 is 1 - sum of square of the residual error/ total sum of square, and from the calculation above, it could be seen that the regression sum of is quite small, hence SSE/(SSE+SSR) equals almost 1, and therefore r square equals 0 in this case. This result is not surprising since R square is monotone increasing with the number of variables included, and it is expected to get a low value with only intercept included. And with the value of r square being 0, we could say the model explains none of the variability of the response data around its mean. And the model does not fit well.

```r
# Compare models pairwise using anova()

anova(RegressionModelComparison.Full,RegressionModelComparison.Null)

## Analysis of Variance Table
##
## Model 1: Output1 ~ 1 + USGG3M + USGG6M + USGG2YR + USGG3YR + USGG5YR
 +
##     USGG10YR + USGG30YR
## Model 2: Output1 ~ 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
```

As can be see from the table, for Model 2, the p value of F statistic is smaller than 2.2e-16, much smaller than the significance level of 0.05, so we could reject the null hypothesis that the two models have the same performance. It means that the fitted model 1 is significantly different from model2 at the level of α=0.05.This result seems to be obvious as the full model with all variables are expected to give more information than the null model including only intercept.

Repeat the analysis for different combinations of input variables and select the one you think is the best.

```
# Using add1()
ma0<-lm(Output1~1,data=AssignmentDataRegressionComparison)
summary(ma0)

## 
## Call:
## lm(formula = Output1 ~ 1, data = AssignmentDataRegressionComparison)
## 
## 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

anova(ma0)

## Analysis of Variance Table
## 
## Response: Output1
##             Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 8299 637400  76.804

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

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

add1(ma0,scope=myScope)
```

```
## Single term additions
##
## Model:
## Output1 ~ 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
```

```
# Add USGG3YR
ma1<-lm(Output1~USGG3YR,data=AssignmentDataRegressionComparison)
summary(ma1)
```

```
##
## Call:
## lm(formula = Output1 ~ USGG3YR, data = AssignmentDataRegressionCompa
rison)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0160 -0.2459  0.0325  0.2638  3.0666
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -13.861618   0.008214   -1688   <2e-16 ***
## USGG3YR       2.455793   0.001230    1996   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3996 on 8298 degrees of freedom
## Multiple R-squared:  0.9979, Adjusted R-squared:  0.9979
## F-statistic: 3.984e+06 on 1 and 8298 DF,  p-value: < 2.2e-16
```

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

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

```
add1(ma1,scope=myScope)
```

```
## Single term additions
##
## Model:
## Output1 ~ 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
```

```
# Add USGG3M
ma2<-lm(Output1~USGG3YR+USGG3M,data=AssignmentDataRegressionComparison)
summary(ma2)
```

```
##
## Call:
## lm(formula = Output1 ~ USGG3YR + USGG3M, data = AssignmentDataRegres
sionComparison)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.19592 -0.25503  0.01678  0.24577  1.77420
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -13.671658   0.007515 -1819.36   <2e-16 ***
## USGG3YR       2.171934   0.004782   454.14   <2e-16 ***
## USGG3M        0.302081   0.004972    60.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3324 on 8297 degrees of freedom
## Multiple R-squared:  0.9986, Adjusted R-squared:  0.9986
## F-statistic: 2.88e+06 on 2 and 8297 DF,  p-value: < 2.2e-16
```

```
(myScope<-myScope[-which(myScope=="USGG3M")])
```

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

```
add1(ma2,scope=myScope)
```

```
## Single term additions
##
## Model:
## Output1 ~ 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
```

```r
# Add USGG10YR
ma3<-lm(Output1~USGG3YR+USGG3M+USGG10YR,data=AssignmentDataRegressionCo
mparison)
summary(ma3)

##
## Call:
## lm(formula = Output1 ~ USGG3YR + USGG3M + USGG10YR, data = Assignmen
tDataRegressionComparison)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42997 -0.04207  0.00512  0.04825  0.69394
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14.723583   0.003369 -4370.4   <2e-16 ***
## USGG3YR       1.120774   0.003068   365.3   <2e-16 ***
## USGG3M        0.705571   0.001616   436.7   <2e-16 ***
## USGG10YR      0.786689   0.002130   369.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0796 on 8296 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 3.353e+07 on 3 and 8296 DF,  p-value: < 2.2e-16
```

Model selection.

The method add1 was used to choose the input variables, and the variable USGG3YR,USGG3M and USGG10YR were selected. From the table add1(ma0,scope=myScope), variable USGG3YR has the smallest AIC that is -15226, so we include this variable in linear model, and from the result of summary(ma1) the r square equals 0.9979. The r square here is already pretty high but we would like to see if it could be improved a little bit more.

Then from the table add1(ma1,scope=myScope),variable USGG3M has the smallest AIC that is -18279, so we include this variable and from the result of summary(ma2) the r square equals 0.9989.

And from the table add1(ma2,scope=myScope),variable USGG10YR has the smallest AIC that is -42007, so we include this variable and from the result of summary(ma2) the r square equals 0.9999.

Now we only include three variables but the r square is high enough. So we may stop the selection and choose these three variables. It could be noticed that this selection also matches the common sense that a more accurate rate could be obtained when we include variables combining a near term rate, a middle term rate and a long-term rate.

Step 6

```
# Perform rolling window analysis of the yields data.
# Set the window width and window shift parameters for rolling window.
# install.packages("zoo")
library(zoo)

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

Window.width<-20; Window.shift<-5

# Calculate rolling mean values for each variable
all.means<-rollapply(AssignmentDataRegressionComparison,width=Window.wi
dth,by=Window.shift,by.column=TRUE, mean)
head(all.means,10)

##         USGG3M  USGG6M USGG2YR USGG3YR USGG5YR USGG10YR USGG30YR  Out
put1
##  [1,] 15.0405 14.0855 13.2795 12.9360 12.7825  12.5780  12.1515 20.1
4842
##  [2,] 15.1865 14.1440 13.4855 13.1085 12.9310  12.7370  12.3370 20.5
5208
##  [3,] 15.2480 14.2755 13.7395 13.3390 13.1500  12.9480  12.5500 21.0
4895
##  [4,] 14.9345 14.0780 13.7750 13.4765 13.2385  13.0515  12.6610 21.0
2611
##  [5,] 14.7545 14.0585 13.9625 13.6890 13.4600  13.2295  12.8335 21.3
1356
##  [6,] 14.6025 14.0115 14.0380 13.7790 13.5705  13.3050  12.8890 21.3
9061
##  [7,] 14.0820 13.5195 13.8685 13.6710 13.4815  13.1880  12.7660 20.7
7200
##  [8,] 13.6255 13.0635 13.6790 13.5735 13.4270  13.1260  12.6950 20.2
3626
##  [9,] 13.2490 12.6810 13.5080 13.4575 13.3680  13.0770  12.6470 19.7
6988
## [10,] 12.9545 12.4225 13.4140 13.4240 13.3850  13.1115  12.6755 19.5
3054

# Create points at which rolling means are calculated
Count<-1:length(AssignmentDataRegressionComparison[,1])
Rolling.window.matrix<-rollapply(Count,width=Window.width,by=Window.shi
ft,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]
##  [1,]    1    2    3    4    5    6    7    8    9    10    11    12
    13
##  [2,]    6    7    8    9   10   11   12   13   14    15    16    17
    18
##  [3,]   11   12   13   14   15   16   17   18   19    20    21    22
    23
##  [4,]   16   17   18   19   20   21   22   23   24    25    26    27
    28
##  [5,]   21   22   23   24   25   26   27   28   29    30    31    32
    33
##  [6,]   26   27   28   29   30   31   32   33   34    35    36    37
    38
##  [7,]   31   32   33   34   35   36   37   38   39    40    41    42
    43
##  [8,]   36   37   38   39   40   41   42   43   44    45    46    47
    48
##  [9,]   41   42   43   44   45   46   47   48   49    50    51    52
    53
## [10,]   46   47   48   49   50   51   52   53   54    55    56    57
    58
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20]
##  [1,]    14    15    16    17    18    19    20
##  [2,]    19    20    21    22    23    24    25
##  [3,]    24    25    26    27    28    29    30
##  [4,]    29    30    31    32    33    34    35
##  [5,]    34    35    36    37    38    39    40
##  [6,]    39    40    41    42    43    44    45
##  [7,]    44    45    46    47    48    49    50
##  [8,]    49    50    51    52    53    54    55
##  [9,]    54    55    56    57    58    59    60
## [10,]    59    60    61    62    63    64    65

# Take middle of each window
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

# Incert means into the total length vector to plot the rolling mean wi
th the original data
Means.forPlot<-rep(NA,length(AssignmentDataRegressionComparison[,1]))
```

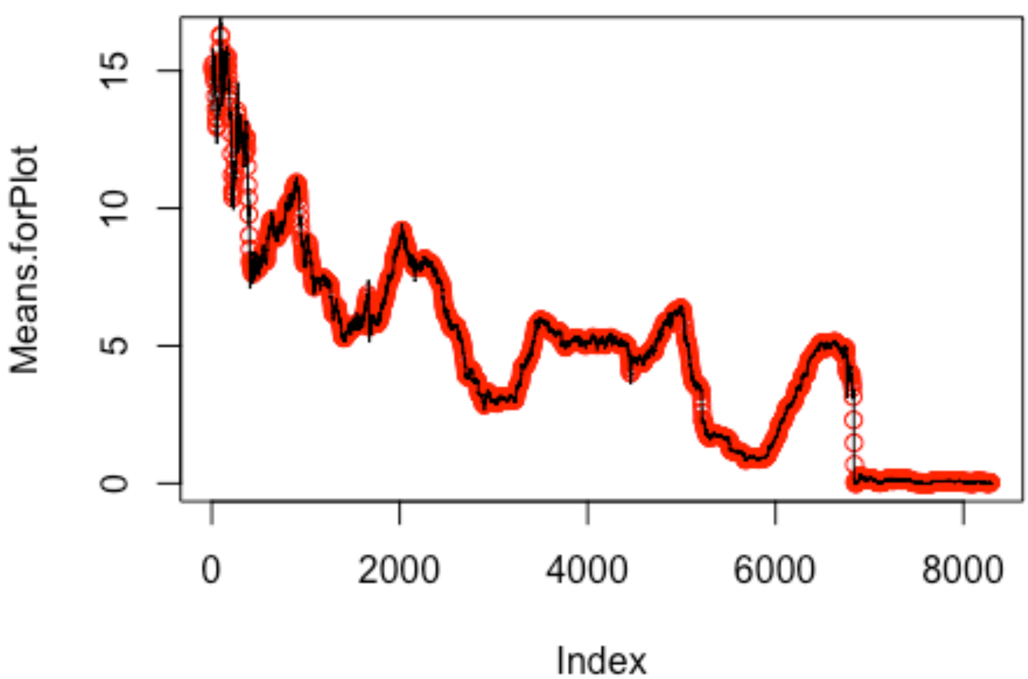
```r
Means.forPlot[Points.of.calculation]<-all.means[,1]
Means.forPlot[1:50]
```

```
##  [1]      NA      NA      NA      NA      NA      NA      NA      NA
##  [9]      NA 15.0405      NA      NA      NA      NA 15.1865      NA
## [17]      NA      NA      NA 15.2480      NA      NA      NA      NA
## [25] 14.9345      NA      NA      NA      NA 14.7545      NA      NA
## [33]      NA      NA 14.6025      NA      NA      NA      NA 14.0820
## [41]      NA      NA      NA      NA 13.6255      NA      NA      NA
## [49]      NA 13.2490
```

```r
# Assemble the matrix to plot the rolling means
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])
```




```
# Run rolling daily difference standard deviation of each variable
daily_difference <- apply(AssignmentDataRegressionComparison,2,FUN = di
ff)
head(daily_difference,10)
```

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

```
rolling.sd<-rollapply(daily_difference,width=Window.width,by=Window.shi
ft,by.column=TRUE, sd)
head(rolling.sd)
```

```
##         USGG3M    USGG6M   USGG2YR   USGG3YR   USGG5YR  USGG10YR  US
GG30YR
## [1,] 0.3433258 0.3262462 0.2748258 0.2030161 0.1713192 0.1299585 0.1
202147
## [2,] 0.2933383 0.2907504 0.2261811 0.1499219 0.1450082 0.1146895 0.1
192201
## [3,] 0.2613180 0.2437530 0.2006201 0.1632596 0.1654110 0.1459308 0.1
351909
## [4,] 0.2551754 0.2469663 0.1989446 0.1692794 0.1717219 0.1551052 0.1
422183
## [5,] 0.2480551 0.2481595 0.2102004 0.1786057 0.1744767 0.1643960 0.1
516540
## [6,] 0.1963884 0.2363672 0.2095082 0.1809180 0.1822917 0.1664956 0.1
537351
##        Output1
## [1,] 0.5639875
## [2,] 0.4707427
## [3,] 0.4681168
## [4,] 0.4786189
## [5,] 0.4888569
## [6,] 0.4788897
```

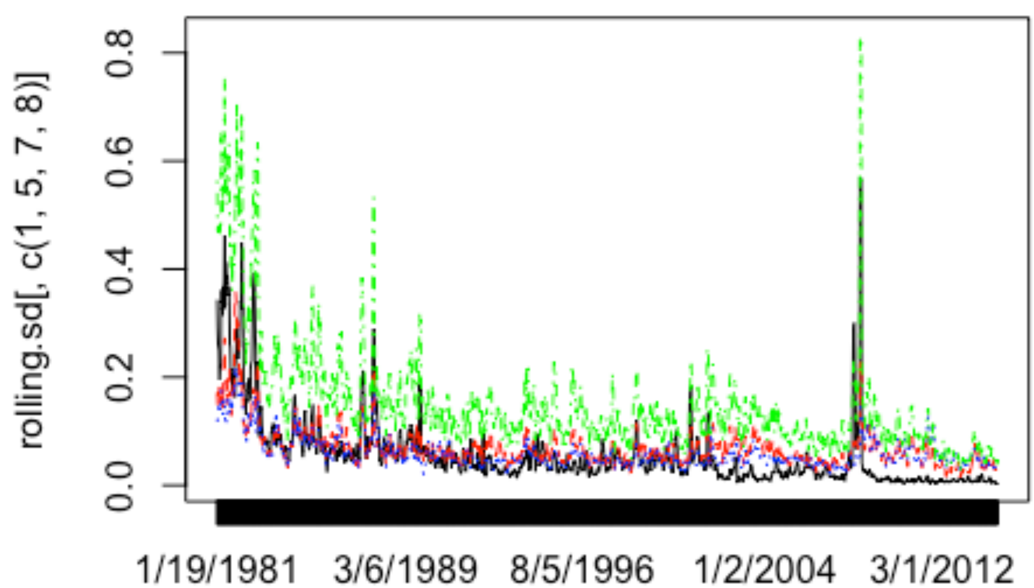
```
rolling.dates<-rollapply(AssignmentDataRegressionComparison[-1,],width=
Window.width,by=Window.shift,
                         by.column=FALSE,FUN=function(z) rownames(z))
head(rolling.dates)
```

```
##      [,1]        [,2]        [,3]        [,4]        [,5]
## [1,] "1/6/1981"  "1/7/1981"  "1/8/1981"  "1/9/1981"  "1/12/1981"
## [2,] "1/13/1981" "1/14/1981" "1/15/1981" "1/16/1981" "1/19/1981"
## [3,] "1/20/1981" "1/21/1981" "1/22/1981" "1/23/1981" "1/26/1981"
## [4,] "1/27/1981" "1/28/1981" "1/29/1981" "1/30/1981" "2/2/1981"
## [5,] "2/3/1981"  "2/4/1981"  "2/5/1981"  "2/6/1981"  "2/9/1981"
## [6,] "2/10/1981" "2/11/1981" "2/13/1981" "2/17/1981" "2/18/1981"
##      [,6]        [,7]        [,8]        [,9]        [,10]
## [1,] "1/13/1981" "1/14/1981" "1/15/1981" "1/16/1981" "1/19/1981"
## [2,] "1/20/1981" "1/21/1981" "1/22/1981" "1/23/1981" "1/26/1981"
## [3,] "1/27/1981" "1/28/1981" "1/29/1981" "1/30/1981" "2/2/1981"
## [4,] "2/3/1981"  "2/4/1981"  "2/5/1981"  "2/6/1981"  "2/9/1981"
## [5,] "2/10/1981" "2/11/1981" "2/13/1981" "2/17/1981" "2/18/1981"
## [6,] "2/19/1981" "2/20/1981" "2/23/1981" "2/24/1981" "2/25/1981"
##      [,11]       [,12]       [,13]       [,14]       [,15]
## [1,] "1/20/1981" "1/21/1981" "1/22/1981" "1/23/1981" "1/26/1981"
## [2,] "1/27/1981" "1/28/1981" "1/29/1981" "1/30/1981" "2/2/1981"
## [3,] "2/3/1981"  "2/4/1981"  "2/5/1981"  "2/6/1981"  "2/9/1981"
## [4,] "2/10/1981" "2/11/1981" "2/13/1981" "2/17/1981" "2/18/1981"
## [5,] "2/19/1981" "2/20/1981" "2/23/1981" "2/24/1981" "2/25/1981"
## [6,] "2/26/1981" "2/27/1981" "3/2/1981"  "3/3/1981"  "3/4/1981"
##      [,16]       [,17]       [,18]       [,19]       [,20]
## [1,] "1/27/1981" "1/28/1981" "1/29/1981" "1/30/1981" "2/2/1981"
## [2,] "2/3/1981"  "2/4/1981"  "2/5/1981"  "2/6/1981"  "2/9/1981"
## [3,] "2/10/1981" "2/11/1981" "2/13/1981" "2/17/1981" "2/18/1981"
## [4,] "2/19/1981" "2/20/1981" "2/23/1981" "2/24/1981" "2/25/1981"
## [5,] "2/26/1981" "2/27/1981" "3/2/1981"  "3/3/1981"  "3/4/1981"
## [6,] "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  USGG10Y
R
## 1/19/1981 0.3433258 0.3262462 0.2748258 0.2030161 0.1713192 0.129958
5
## 1/26/1981 0.2933383 0.2907504 0.2261811 0.1499219 0.1450082 0.114689
5
## 2/2/1981  0.2613180 0.2437530 0.2006201 0.1632596 0.1654110 0.145930
8
## 2/9/1981  0.2551754 0.2469663 0.1989446 0.1692794 0.1717219 0.155105
2
## 2/18/1981 0.2480551 0.2481595 0.2102004 0.1786057 0.1744767 0.164396
0
## 2/25/1981 0.1963884 0.2363672 0.2095082 0.1809180 0.1822917 0.166495
```

```
6
##               USGG30YR   Output1
## 1/19/1981 0.1202147 0.5639875
## 1/26/1981 0.1192201 0.4707427
## 2/2/1981  0.1351909 0.4681168
## 2/9/1981  0.1422183 0.4786189
## 2/18/1981 0.1516540 0.4888569
## 2/25/1981 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))
```



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

The high volatility could be noticed around the year of 1981, 1987 and 2008. As can be seen from the plot, the volatility increases when the level of rates increases, and this is obvious especially for those periods with high volatility. And in real life, when stock market get out of balance, people will be in panic and start to selling and then the market becomes tanking, rates will be changing, and therefore increase the volatility.

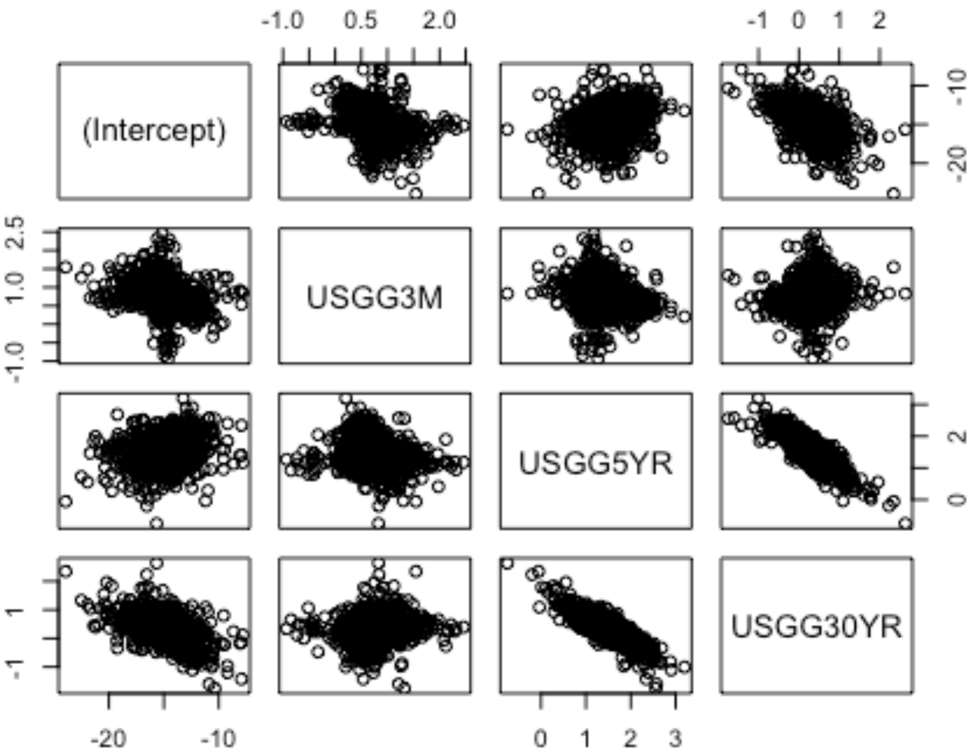
```
# Show periods of high volatility
high.volatility.periods<-rownames(rolling.sd)[rolling.sd[,8]>.5]
high.volatility.periods
```

```
##  [1] "1/19/1981"  "3/25/1981"  "4/1/1981"   "4/8/1981"   "4/23/1981"

##  [6] "4/30/1981"  "5/7/1981"   "5/14/1981"  "5/21/1981"  "5/29/1981"

## [11] "6/5/1981"   "6/12/1981"  "6/19/1981"  "6/26/1981"  "7/6/1981"

## [16] "7/13/1981"  "7/20/1981"  "7/27/1981"  "10/28/1981" "11/5/1981"

## [21] "11/13/1981" "11/20/1981" "11/30/1981" "12/7/1981"  "12/14/1981
"
## [26] "12/29/1981" "1/14/1982"  "1/21/1982"  "1/28/1982"  "2/4/1982"

## [31] "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
"
## [41] "10/13/1987" "10/20/1987" "10/27/1987" "11/19/2007" "11/26/2007
"
## [46] "11/12/2008" "11/19/2008"
```

```
# Fit linear model to rolling window data using 3 months, 5 years and 3
0 years variables as predictors
# Rolling lm coefficients
Coefficients<-rollapply(AssignmentDataRegressionComparison,width=Window
.width,by=Window.shift,by.column=FALSE,
         FUN=function(z) coef(lm(Output1~USGG3M+USGG5YR+USGG30YR,data=a
s.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
```

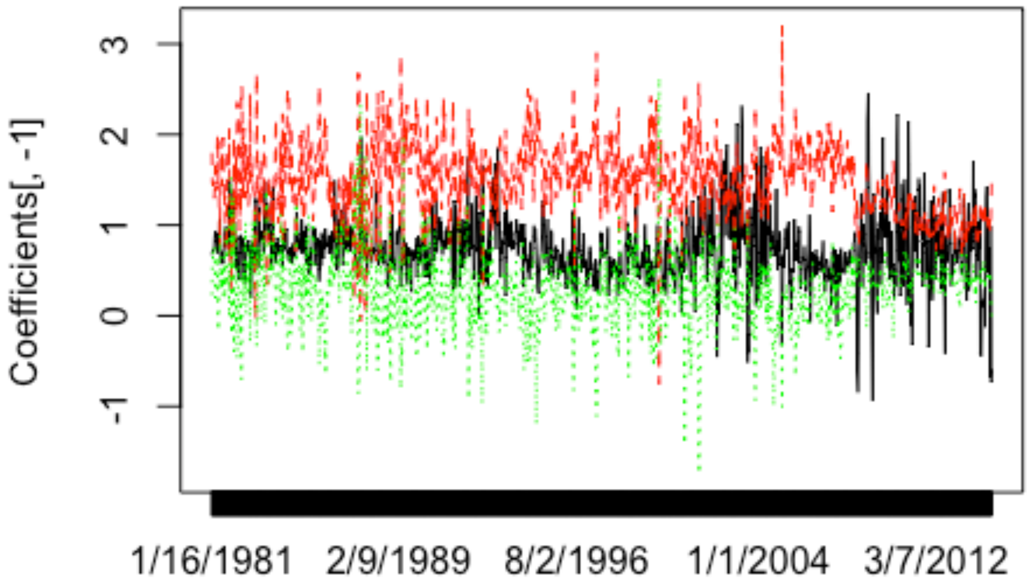
```r
# Look at pairwise X-Y plots of regression coefficients for the 3M, 5Yr
 and 30Yr yields as inputs.
# Pairs plot of Coefficients
pairs(Coefficients)
```



Interpret the pairs plot.

From the pairs plot we could notice that the coefficients of USGG5YR and USGG30YR is negatively correlated. And for Intercept and the coefficients of USGG30YR, it seems that they are somewhat negatively correlated as well but the pattern is not that obvious. But for USGG3M and USGG30YR, there is a diamond shape plot showing there is a weak relationship between them, also this is somewhat unusual and may imply some interesting connection. And for the rest of comparisons, we see no obvious correlation between them.

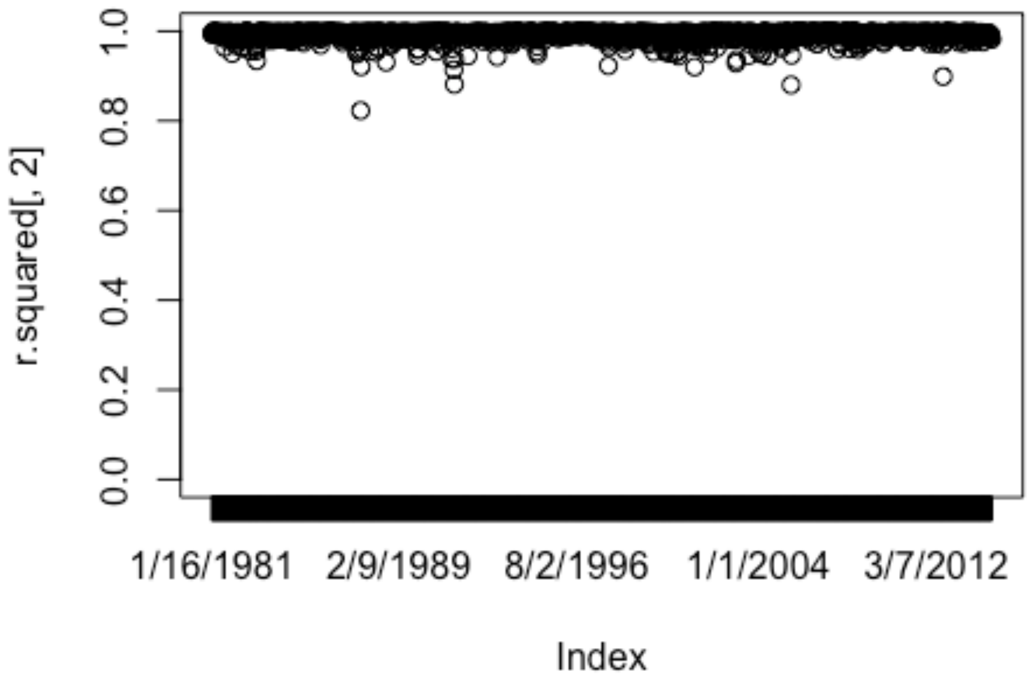
```r
# Plot of coefficients
matplot(Coefficients[,-1],xaxt="n",type="l",col=c("black","red","green"
))
axis(side=1,at=1:1657,rownames(Coefficients))
```




```
high.slopespread.periods<-rownames(Coefficients)[Coefficients[,3]-Coeff
icients[,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"

##  [6] "5/27/1987"  "9/25/1987"  "3/15/1988"  "9/27/1988"  "10/5/1988"

## [11] "3/10/1989"  "2/5/1992"   "8/3/1994"   "12/8/1994"  "6/14/1996"

## [16] "5/9/1997"   "5/16/1997"  "5/23/1997"  "5/30/1997"  "12/26/2000
"
## [21] "1/2/2001"   "7/25/2001"  "8/1/2001"   "11/13/2003" "8/12/2004"

## [26] "12/16/2004"

jump.slopes

## [1] "12/16/2004"
```

For most of the time the coefficients are consistent with the picture of the pairs. From the pairs plot above, the coefficients of 5 year rate and coefficients of 30-year rate are negatively correlated. And in our graph here the red line representing the coefficients of 5 year rate and green line representing the coefficients of 30-year rate seems to move in opposite direction, and specifically this pattern is obvious during the period from 1991 to 2007. When the red line goes up, the green line will goes down, that is when the coefficients of 5 year rates increase, the coefficients of 30-year rates will decrease. But after the financial crisis of year 2008, the red line goes down and its relationship between the green line becomes not that obvious. Also, from the pairs plot above, the coefficients of 3 month rate and 30-year rate is sort of weak correlated and from the graph here the we see somewhat relationship between the black line and the green line but the relationship is not obvious. Besides, it appears that the black line and red line are sort of negatively correlated, especially for period from 1991 to 2007, but this correlation was not obvious from the pairs plot.

```
# R-squared
r.squared<-rollapply(AssignmentDataRegressionComparison,width=Window.wi
dth,by=Window.shift,by.column=FALSE,
         FUN=function(z) summary(lm(Output1~USGG3M+USGG5YR+USGG30YR,dat
a=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))
```




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

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

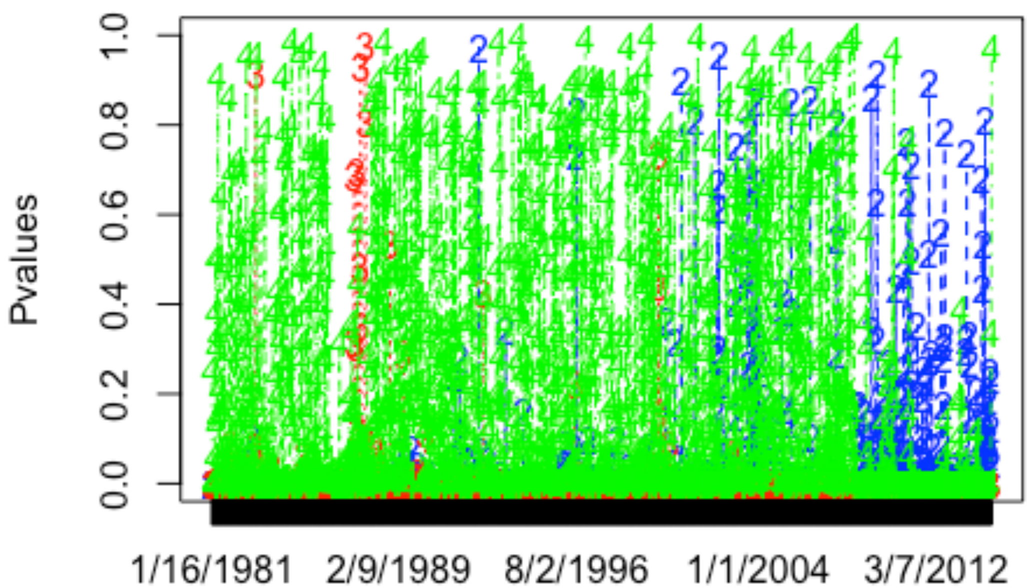
What could cause decrease of R2?

From the plot of r square, it could be seen that the overall performance of r square is good. The decrease of R2 implies us that the fitted model may not perform well during that time. One significant decrease of R2 is around year 1986, and the R square almost dropped to 0.8, and this may due to the Black Monday at year 1987. Besides, we could also see another three obvious decrease in the year around 1991, 2006 and 2012. This may also due to the instability of financial market during that specific time.

```
# P-values
Pvalues<-rollapply(AssignmentDataRegressionComparison,width=Window.widt
h,by=Window.shift,by.column=FALSE,
                        FUN=function(z) summary(lm(Output1~USGG3M+USGG5
YR+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))
```




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

## [1] "7/15/1992" "7/26/1996" "8/2/1996"  "11/7/2000" "5/30/2001" "5/2
/2002"

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

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

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

```
## [1] "3/17/1981"  "4/22/1981"  "4/29/1981"  "6/4/1981"   "10/13/1981"
## [6] "11/19/1981"
```

We know the blue line represents the 3-month rate and red line is the 5-year rate and the green line is the 30-year rate. And from the plot we could see the p value of green line deviate from 0 a lot, it means that the 30-year rate is not a powerful predictor especially for the year before 2006. And after the year 2007, it seems that the 30-year rate comes to be a powerful predictor. This may be because for some reason the market changes and 30-year rate provide a better prediction.

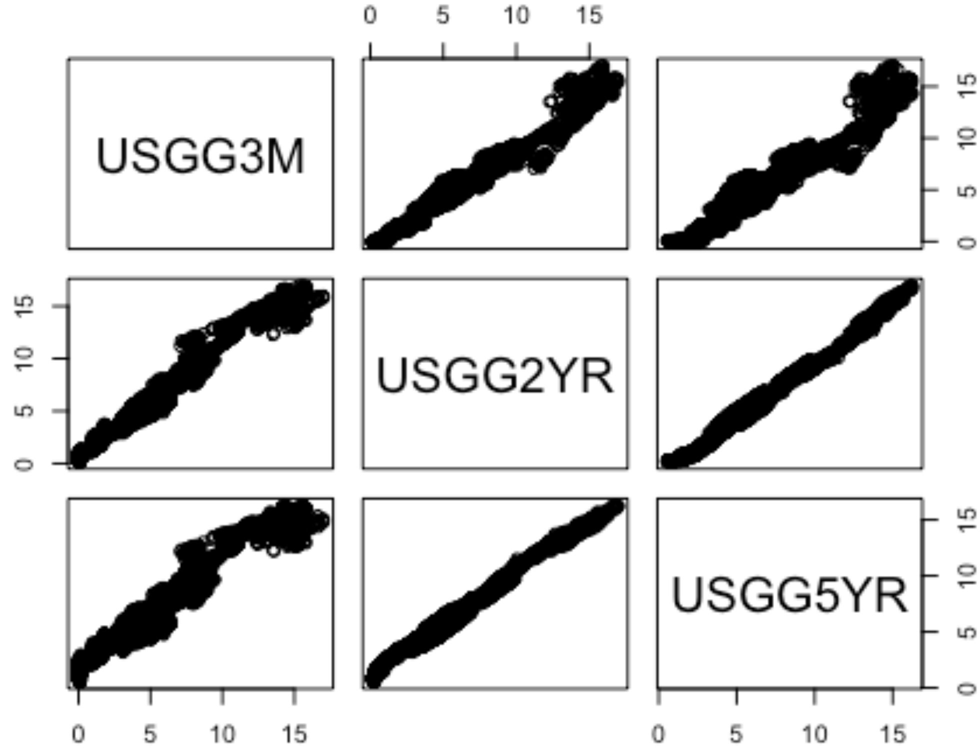
And for the p value of blue line, basically the p value is low before 2001, except for only a couple of years like 1991 and 1996, so the 3-month rate is a powerful predictor during that period. But it started to deviates from 0 a lot after the year around 2001.

The 5-year rate seems to be a powerful predictor all the time except for the years around 1986 and 1998. This may due to the Black Monday at year 1987 and the Russian financial crisis on 1998.

On the whole, we could say the 5-year rate has the best predicting performance. The 30-year rate has a better predicting performance for recent years. And for 3-month rate, it is initially a very powerful predictor but overtime especially since the financial crisis it seems it does not have a significant effect.

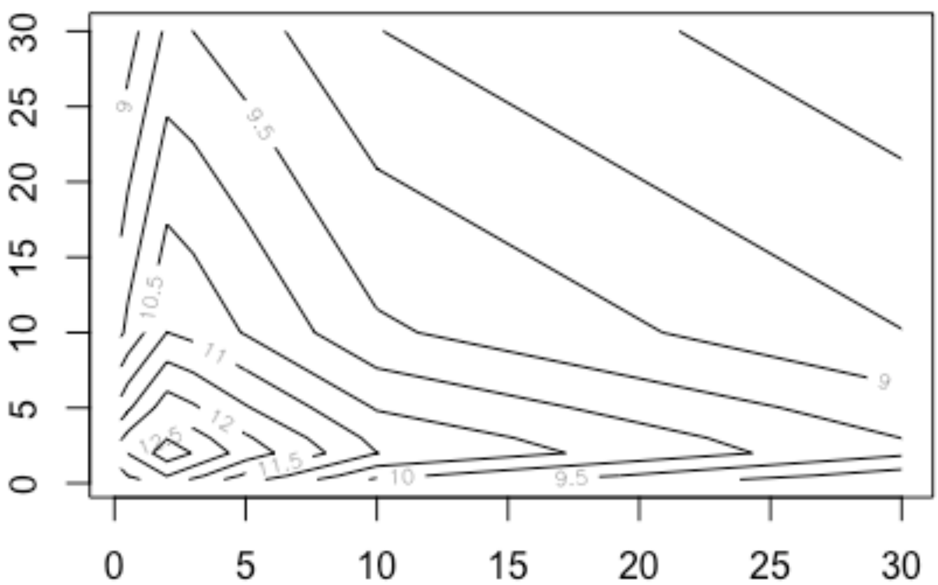
Step 7

```
# Perform PCA with the inputs (columns 1-7)
AssignmentData.Output<-AssignmentData$Output1
AssignmentData<-data.matrix(AssignmentData[,1:7],rownames.force="automa
tic")
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

# Select 3 variables. Explore dimensionality and correlation
AssignmentData.3M_2Y_5Y<-AssignmentData[,c(1,3,5)]
pairs(AssignmentData.3M_2Y_5Y)
```




```
# library(rgl)
# rgl.points(AssignmentData.3M_2Y_5Y)

# Analyze the covariance matrix of the data.

# Manual calculation
n <- nrow(AssignmentData) # number of subjects

Transposed_Assignmentdata <- t(as.matrix(AssignmentData))
dim(Transposed_Assignmentdata)

## [1]    7 8300

Means <- rowMeans(Transposed_Assignmentdata)

# Another way to calculate means for each column
# AssignmentData_mean <- matrix(data=1, nrow=n) %*% cbind(mean(Assignme
ntData[,1]),mean(AssignmentData[,2]),mean(AssignmentData[,3]),mean(Assi
gnmentData[,4]),mean(AssignmentData[,5]),mean(AssignmentData[,6]),mean(
AssignmentData[,7]))

# Creates a centered matrix
Centered_matrix <- Transposed_Assignmentdata - matrix(rep(Means,dim(Tra
```

```
nsposed_Assignmentdata)[2]),nrow=dim(Transposed_Assignmentdata)[1])

# Creates the covariance matrix
(Manual.Covariance.Matrix <- (n-1)^-1*Centered_matrix %*% t(Centered_ma
trix))

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

# Caculate using cov()
(Covariance.Matrix <- cov(AssignmentData))

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

# Plot the covariance matrix.
Maturities<-c(.25,.5,2,3,5,10,30)
contour(Maturities,Maturities,Covariance.Matrix)
```




```
# Find eigenvalues and eigenvectors
Eigen.Decomposition <- eigen(Covariance.Matrix,TRUE)

# Return eigenvalues
Eigenvalues <- Eigen.Decomposition$values

# Return eigenvector, and in this case they are loadings
(Eigenvectors<- Eigen.Decomposition$vectors)

##             [,1]        [,2]       [,3]        [,4]       [,5]
 [,6]
## [1,] -0.3839609 -0.50744508  0.5298222  0.40373501  0.3860878 -0.039
76285
## [2,] -0.3901870 -0.43946144  0.1114737 -0.40526448 -0.6787624  0.094
75452
## [3,] -0.4151851 -0.11112721 -0.4187873 -0.40896949  0.3787209 -0.298
48638
## [4,] -0.4063541  0.01696988 -0.4476561  0.06433748  0.2362448  0.197
60026
## [5,] -0.3860610  0.23140317 -0.2462364  0.53357656 -0.2868460  0.421
25768
## [6,] -0.3477544  0.43245979  0.1500903  0.19856539 -0.2562426 -0.735
61857
```

```
## [7,] -0.3047124  0.54421228  0.4979195 -0.42098839  0.2074508  0.377
76687
##              [,7]
## [1,]  0.026742547
## [2,] -0.090913541
## [3,]  0.490009873
## [4,] -0.731570606
## [5,]  0.438559615
## [6,] -0.152627535
## [7,]  0.009199827
```

```
# Calculate the factors
# dim(Eigenvectors)
# dim(Centered_matrix)
pcafactor <- t(Centered_matrix)%*%Eigenvectors
colnames(pcafactor) <- c("F1","F2","F3","F4","F5","F6","F7")
head(pcafactor)
```

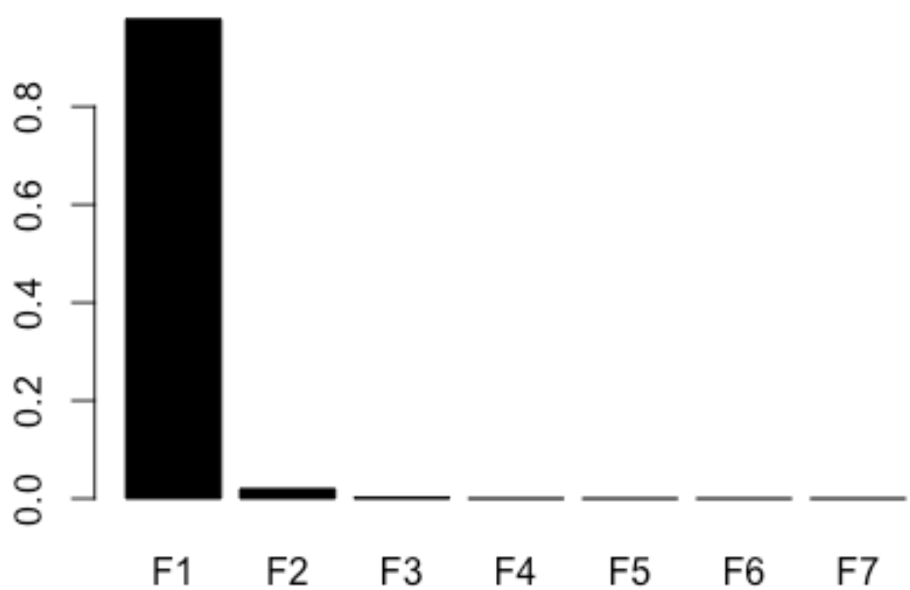
```
##                  F1        F2       F3        F4          F5
  F6
## 1/5/1981  -18.01553 -2.240277 1.461149 0.3121963 -0.27950095 -0.0136
1981
## 1/6/1981  -18.09140 -2.352346 1.442377 0.2020975 -0.21054340 -0.0495
0872
## 1/7/1981  -19.44731 -2.862932 1.644084 0.2738177 -0.19551603 -0.0193
4850
## 1/8/1981  -19.74851 -3.040712 1.633909 0.3026486 -0.06670820  0.0217
4990
## 1/9/1981  -20.57204 -3.177974 1.661795 0.2577797  0.07603629 -0.0279
2115
## 1/12/1981 -20.14218 -3.241569 1.966508 0.3260907 -0.01625095  0.0026
9386
##                   F7
## 1/5/1981  -0.07600647
## 1/6/1981  -0.06309127
## 1/7/1981  -0.06835902
## 1/8/1981  -0.07610064
## 1/9/1981  -0.06126411
## 1/12/1981 -0.03873468
```

```
# Calculate vector of means
(Vectorofmeans <- rowMeans(Transposed_Assignmentdata))
```

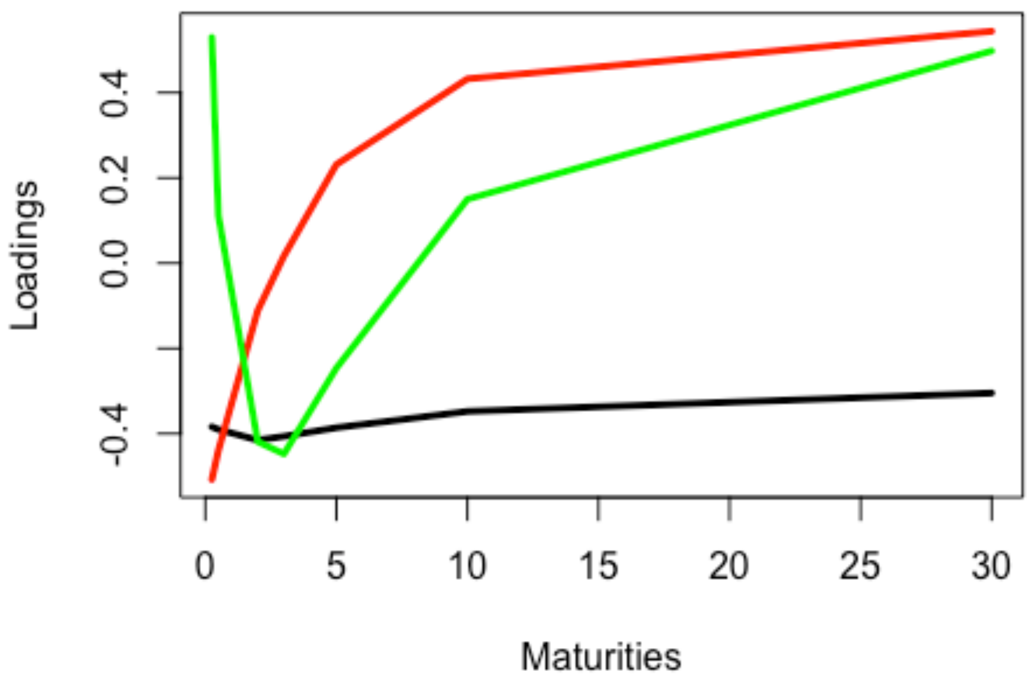
```
##   USGG3M   USGG6M  USGG2YR  USGG3YR  USGG5YR USGG10YR USGG30YR
## 4.675134 4.844370 5.438888 5.644458 6.009421 6.481316 6.869355
```

```
# Calculate the first 3 loadings
Loadings <- Eigenvectors[,1:3]
colnames(Loadings) <- c("L1","L2","L3")
rownames(Loadings) <- colnames(AssignmentData)
Loadings
```

```
##                 L1          L2         L3
## USGG3M   -0.3839609 -0.50744508  0.5298222
## USGG6M   -0.3901870 -0.43946144  0.1114737
## USGG2YR  -0.4151851 -0.11112721 -0.4187873
## USGG3YR  -0.4063541  0.01696988 -0.4476561
## USGG5YR  -0.3860610  0.23140317 -0.2462364
## USGG10YR -0.3477544  0.43245979  0.1500903
## USGG30YR -0.3047124  0.54421228  0.4979195

# Calculate the first 3 factors
Factors <- pcafactor[,1:3]

# See importance of factors
barplot(Eigen.Decomposition$values/sum(Eigen.Decomposition$values), wid
th = 2,col = "black",
        names.arg=c("F1","F2","F3","F4","F5","F6","F7"))
```




```
# As can be seen, the first three factors are the most important.

# Plot the Loadings
matplot(Maturities,Loadings,type="l",lty=1,col=c("black","red","green")
,lwd=3)
```



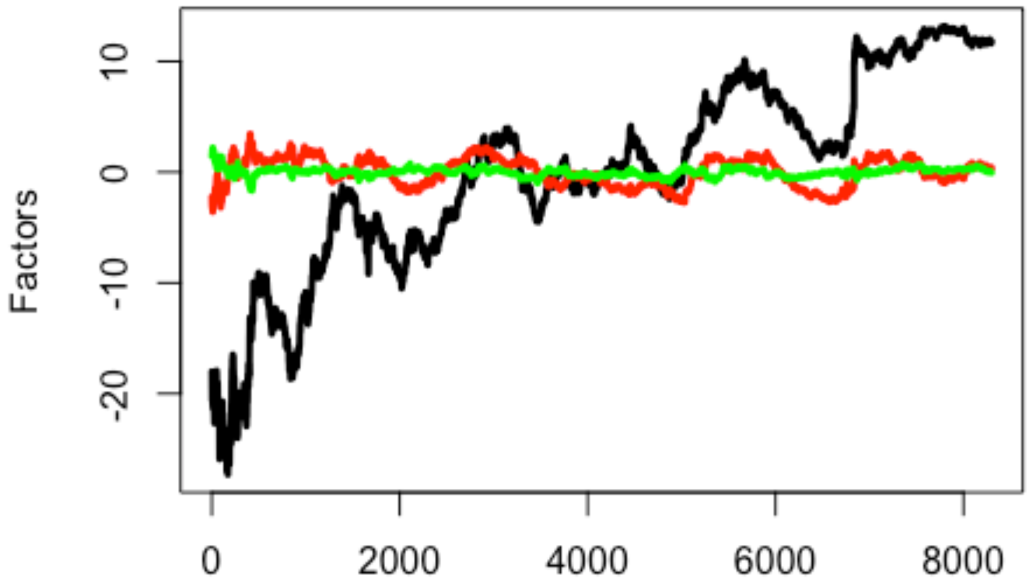
Interpret the factors by looking at the shapes of the loadings.

The black line is the first factor and since the shape of the loadings are sort of flat, we may say factor 1 has roughly equal weight for the seven predictors, and the weights are always between range about -0.35 and -0.45. This means that factor 1 describe the movements when all the movement go up together and down together at the same time. And when the factor 1 is high, that implies all the rates will be high, likewise, if factor 1 is low then the rates will be low.

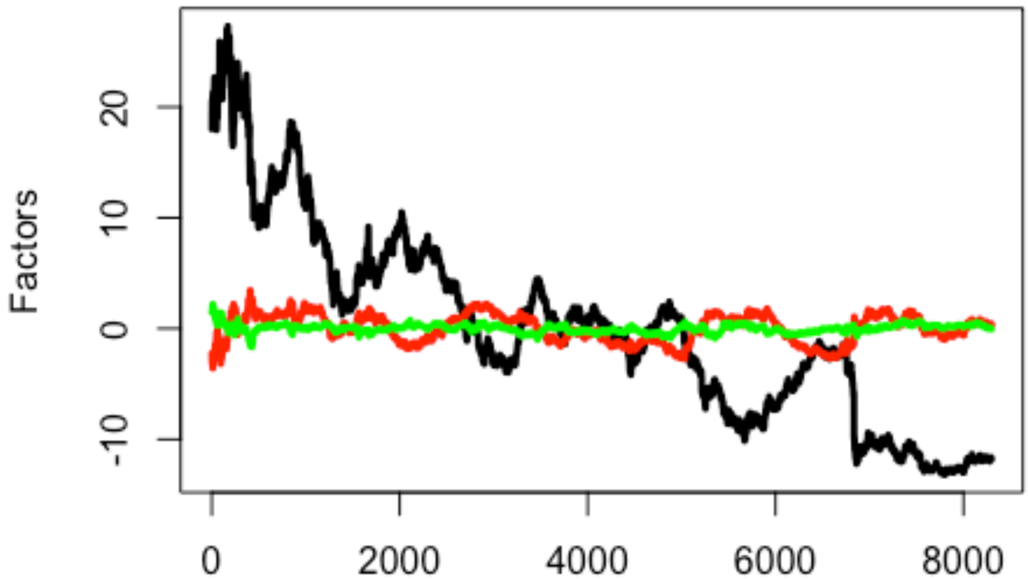
And the second factor is the red line, we could see there is a negative rate for the near term rates and positive rate for the long-term rates. That means the second factor describes the near term and long term movement in different directions. That is if short term rates goes up, then the long term rates goes down and vice versa.

The third factor is the green line. As can be seen, it has the same sign of weight for both the near term and long-term. It describes the curvature behavior of movement, that is when short-term rate and long-term rate have the same direction but the middle term rate moves to the opposite direction.

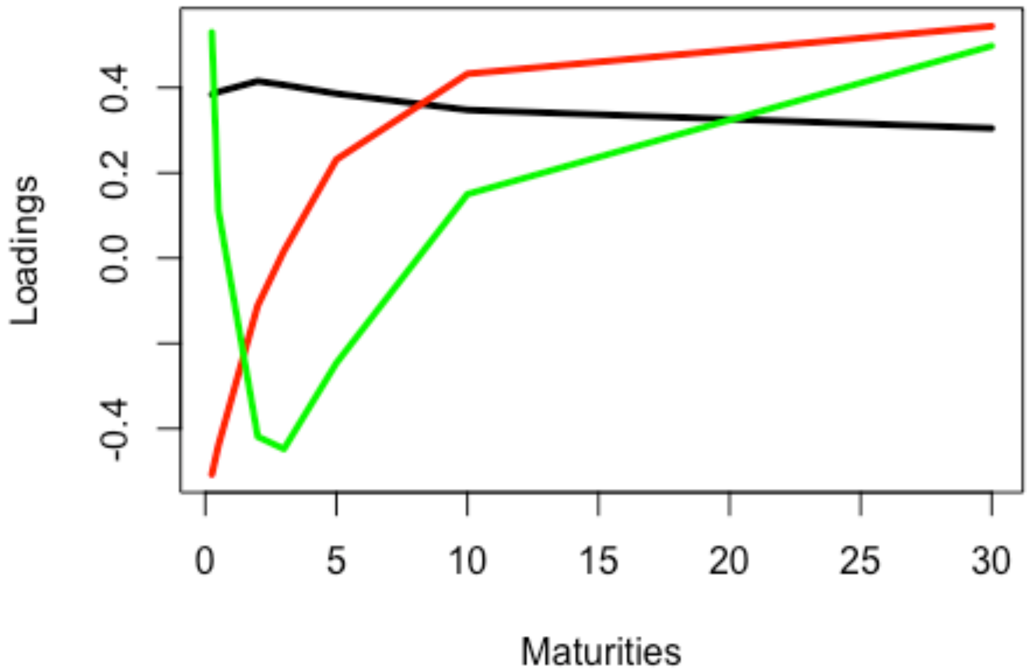
```
# Calculate and plot 3 selected factors
matplot(Factors,type="l",col=c("black","red","green"),lty=1,lwd=3)
```



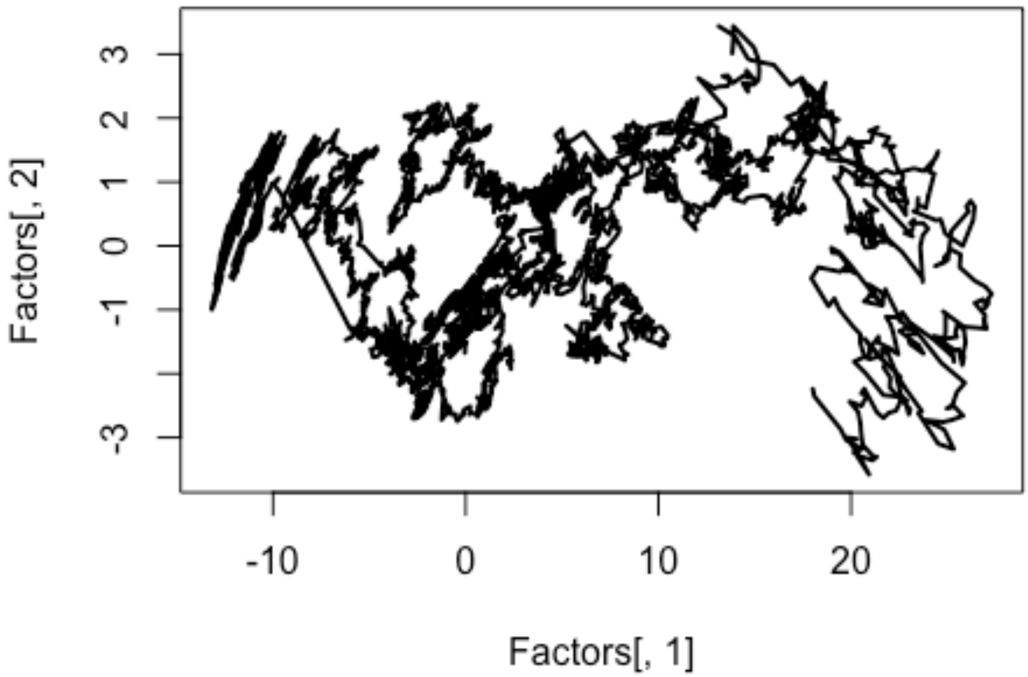

```
# Change the signs of the first factor and the corresponding factor loa
ding.
Loadings[,1]<--Loadings[,1]
Factors[,1]<--Factors[,1]
matplot(Factors,type="l",col=c("black","red","green"),lty=1,lwd=3)
```




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




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




```
rownames(AssignmentDataRegressionComparison[c(7135,8300),])

## [1] "1/5/2010"  "6/26/2014"

rownames(AssignmentDataRegressionComparison[c(1,506),])

## [1] "1/5/1981"  "1/21/1983"

rownames(AssignmentDataRegressionComparison[c(742,1737),])

## [1] "1/11/1984" "1/27/1988"
```

Draw at least three conclusions from the plot of the first two factors above.

From the plot we could notice that from 1/5/2010 to 6/26/2014, the factor 1 and factor 2 are significantly linear correlated with each other.

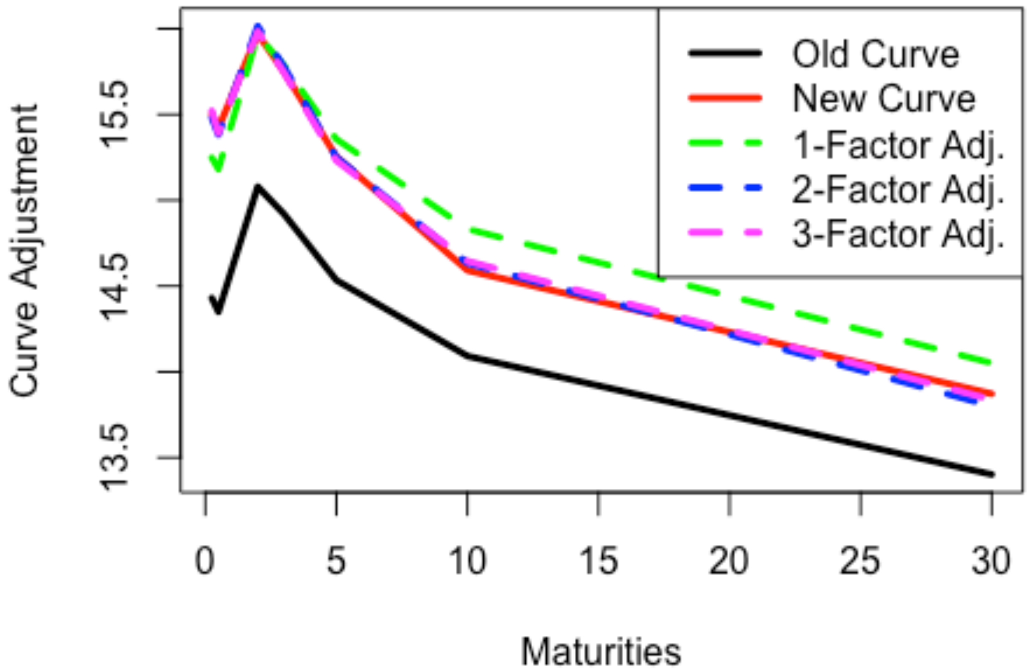
Also we could notice that generally speaking the plot is very dense, which means the day to day change is relatively small. But from 1/5/1981 to 8/12/1985, during that time the volatility is relatively high.

Besides, on the whole we could see in recent years the volatility is low compared to the volatility during the time of 1980s.

And from 1/11/1984 to 1/27/1988, we could notice a comparatively flat line between factor 1 and factor 2, and based on this we may say during that time, there is no obvious relationship between these two factors, and the score of factor 2 remain near 0 when the score of factor 1 change from -18 to -3.

Also, for the whole period, the range of factor 1 is greater than the range of factor 2. For score of factor 1, it change from about -15 to 25 and for score of factor 2, it change from -4 to 4.

```
# Analyze the adjustments that each factor makes to the term curve.
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,ModelCur
veAdjustment.2Factors,
                ModelCurveAdjustment.3Factors)),
        type="l",lty=c(1,1,2,2,2),col=c("black","red","green","blue","m
agenta"),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("bl
ack","red","green","blue","magenta"))
```




```
rbind(CurveChange,ModelCurveAdjustment.3Factors-OldCurve)

##                USGG3M   USGG6M   USGG2YR   USGG3YR   USGG5YR  USGG10Y
R
## CurveChange 1.070000 1.070000 0.8900000 0.8300000 0.7200000 0.500000
0
##             1.090063 1.041267 0.9046108 0.8248257 0.6979317 0.553173
4
##              USGG30YR
## CurveChange 0.4700000
##             0.4357793
```

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

From the plot we could see, with an old rate, when using only factor 1 it's just parallel shift of the curve. And by comparing the green line and red line, we could notice that it is too low for the near term and too high for the long term.

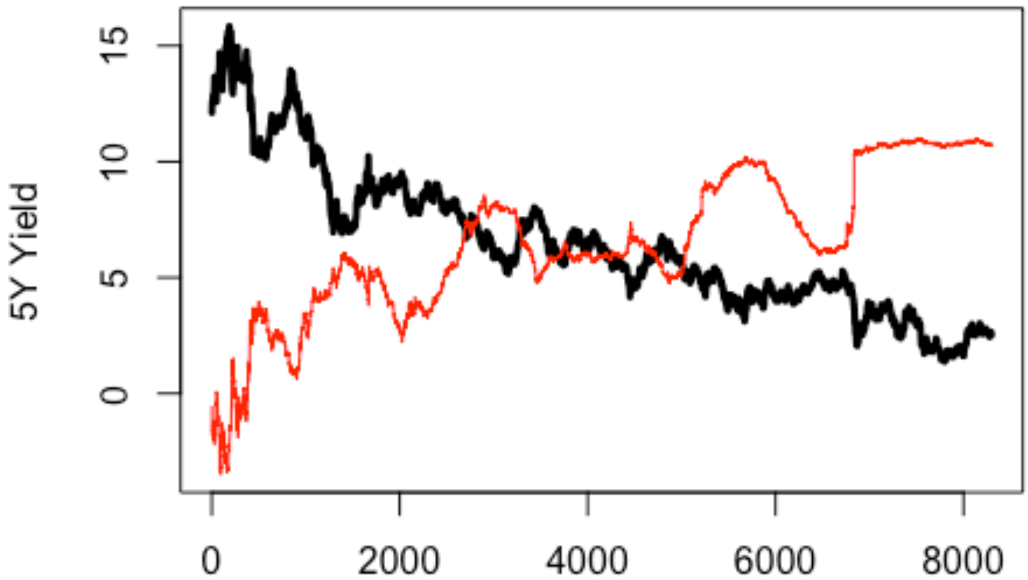
And when using factor 2, we tilt the curve that is we change the slope of the curve, and by comparing the blue line and the red line we could see the blue line is a little bit too low at the beginning and at the end, and little bit too high in the middle.

And when we adding factor 3, the curvature is changed. Then when we compare the magenta line and the red line we could notice that they are very close to each other. Hence, the factor 1 change the shift of the curve and factor 2 twist the curve and the factor change the curvature and it is also called the butterfly. And when we combine these three factors, we see the old curve could almost perfectly match the new curve.

```
# See the goodness of fit for the example of 10Y yield.
Loadings[,1]<--Loadings[,1]
cbind(Maturities,Loadings)
```

```
##          Maturities         L1          L2         L3
## USGG3M         0.25 -0.3839609 -0.50744508  0.5298222
## USGG6M         0.50 -0.3901870 -0.43946144  0.1114737
## USGG2YR        2.00 -0.4151851 -0.11112721 -0.4187873
## USGG3YR        3.00 -0.4063541  0.01696988 -0.4476561
## USGG5YR        5.00 -0.3860610  0.23140317 -0.2462364
## USGG10YR      10.00 -0.3477544  0.43245979  0.1500903
## USGG30YR      30.00 -0.3047124  0.54421228  0.4979195
```

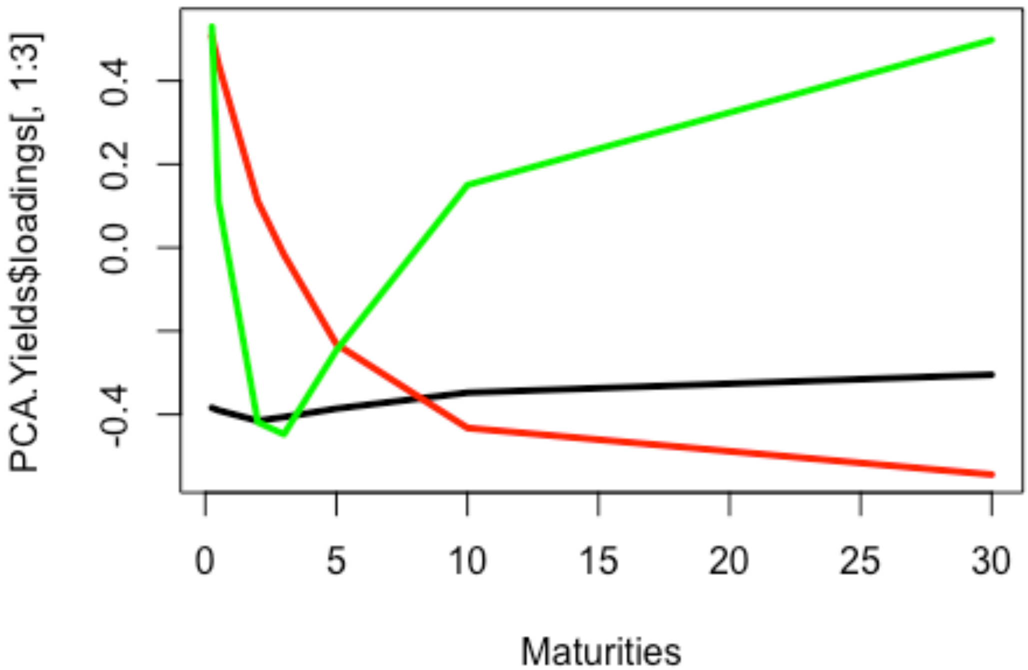
```
Model.10Y<-Means[6]+Loadings[6,1]*Factors[,1]+Loadings[6,2]*Factors[,2]
+Loadings[6,3]*Factors[,3]
matplot(cbind(AssignmentData[,6],Model.10Y),type="l",lty=1,lwd=c(3,1),c
ol=c("black","red"),ylab="5Y Yield")
```



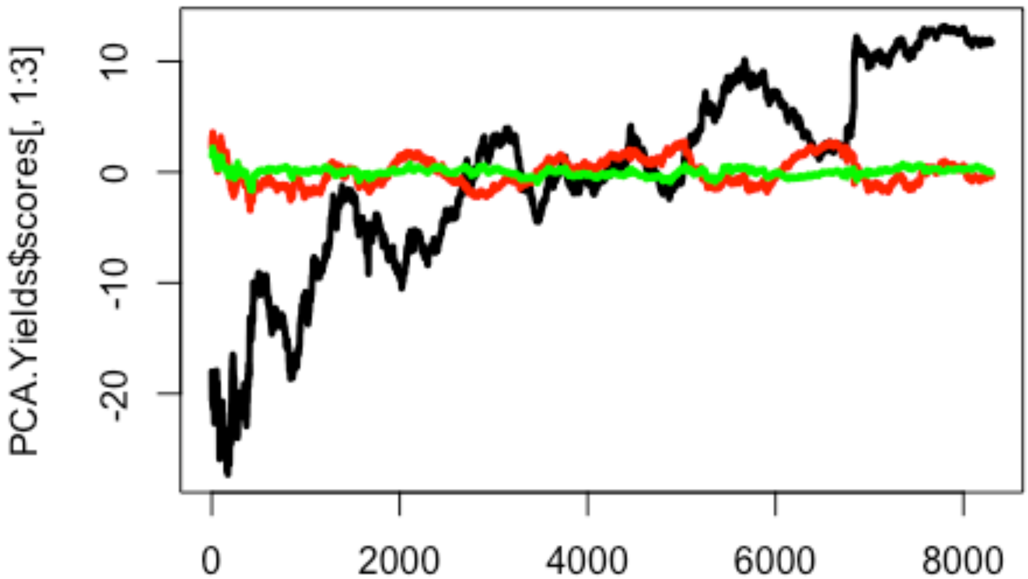

```
# Repeat the PCA using princomp
PCA.Yields<-princomp(AssignmentData)
names(PCA.Yields)

## [1] "sdev"     "loadings" "center"   "scale"    "n.obs"    "scores"

## [7] "call"

# Check that the Loadings are the same
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
## USGG6M   -0.3901870  0.43946144  0.1114737       0.50 -0.3901870
## USGG2YR  -0.4151851  0.11112721 -0.4187873       2.00 -0.4151851
## USGG3YR  -0.4063541 -0.01696988 -0.4476561       3.00 -0.4063541
## USGG5YR  -0.3860610 -0.23140317 -0.2462364       5.00 -0.3860610
## USGG10YR -0.3477544 -0.43245979  0.1500903      10.00 -0.3477544
## USGG30YR -0.3047124 -0.54421228  0.4979195      30.00 -0.3047124
##
## USGG3M   -0.50744508  0.5298222
## USGG6M   -0.43946144  0.1114737
## USGG2YR  -0.11112721 -0.4187873
```

```
## USGG3YR   0.01696988 -0.4476561
## USGG5YR   0.23140317 -0.2462364
## USGG10YR  0.43245979  0.1500903
## USGG30YR  0.54421228  0.4979195
```

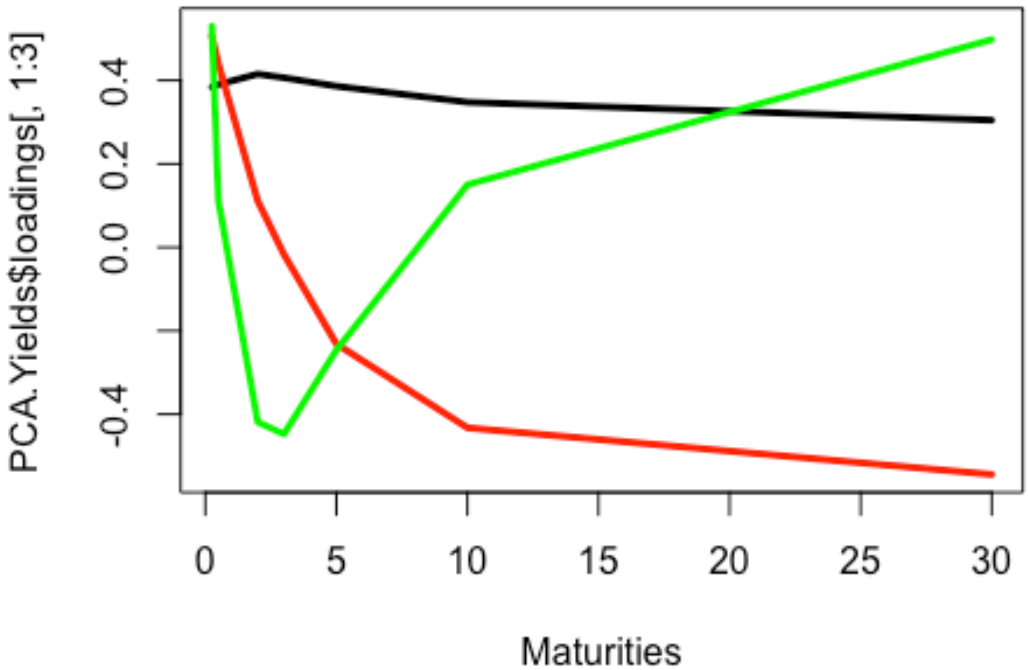
```r
matplot(Maturities,PCA.Yields$loadings[,1:3],type="l",col=c("black","red","green"),lty=1,lwd=3)
```



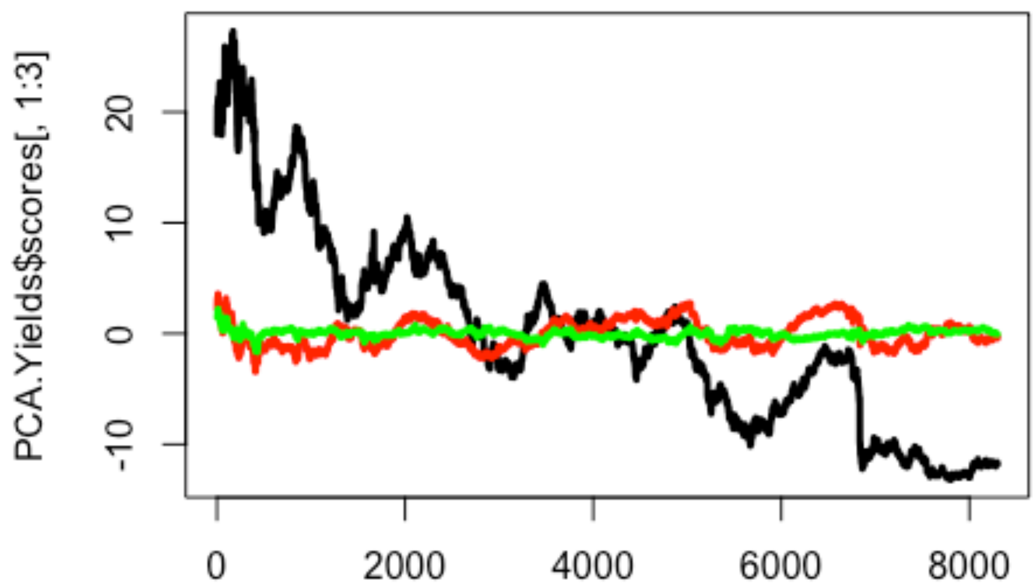

```r
matplot(PCA.Yields$scores[,1:3],type="l",col=c("black","red","green"),lwd=3,lty=1)
```



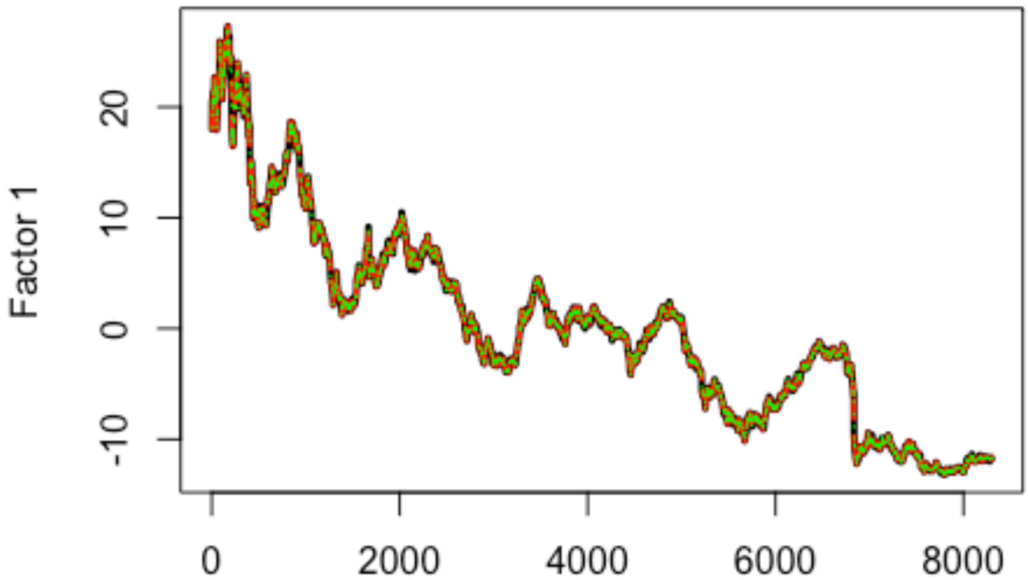

```
# Change the signs of the first factor and factor loading again.
PCA.Yields$loadings[,1]<--PCA.Yields$loadings[,1]
PCA.Yields$scores[,1]<--PCA.Yields$scores[,1]
matplot(Maturities,PCA.Yields$loadings[,1:3],type="l",col=c("black","re
d","green"),lty=1,lwd=3)
```




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




```
# What variable we had as Output?
# The Output is actually the first factor.
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="F
actor 1")
```




```
# Compare the regression coefficients from Step 2 and Step 3 with facto
r loadings.
t(apply(AssignmentData, 2, function(AssignmentData.col) lm(AssignmentDa
ta.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 m
odels Y~Output1, where Y is one of the columns of yields in the data. A
lso, 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 corre
spondence between the coefficients of models Output1~Yield and the firs
t loading.
# Yes, the first loading is the same as the slopes of the model.
AssignmentData.Centered<-t(apply(AssignmentData,1,function(AssignmentDa
ta.row) AssignmentData.row-PCA.Yields$center))
dim(AssignmentData.Centered)

## [1] 8300    7

t(apply(AssignmentData.Centered, 2, function(AssignmentData.col) lm(Ass
ignmentData.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 a
ll inputs together.
t(lm(AssignmentData.Output~AssignmentData.Centered)$coef)[-1]

## [1] 0.3839609 0.3901870 0.4151851 0.4063541 0.3860610 0.3477544 0.30
47124

PCA.Yields$loadings[,1]

##    USGG3M    USGG6M   USGG2YR   USGG3YR   USGG5YR  USGG10YR  USGG30Y
R
## 0.3839609 0.3901870 0.4151851 0.4063541 0.3860610 0.3477544 0.304712
4

# This means that the factor is a portfolio of all input variables with
 weights.
PCA.Yields$loadings[,1]

##    USGG3M    USGG6M   USGG2YR   USGG3YR   USGG5YR  USGG10YR  USGG30Y
R
## 0.3839609 0.3901870 0.4151851 0.4063541 0.3860610 0.3477544 0.304712
4
```