PCA on 2020 US Treasury Yields to Maturity

Yue Sun

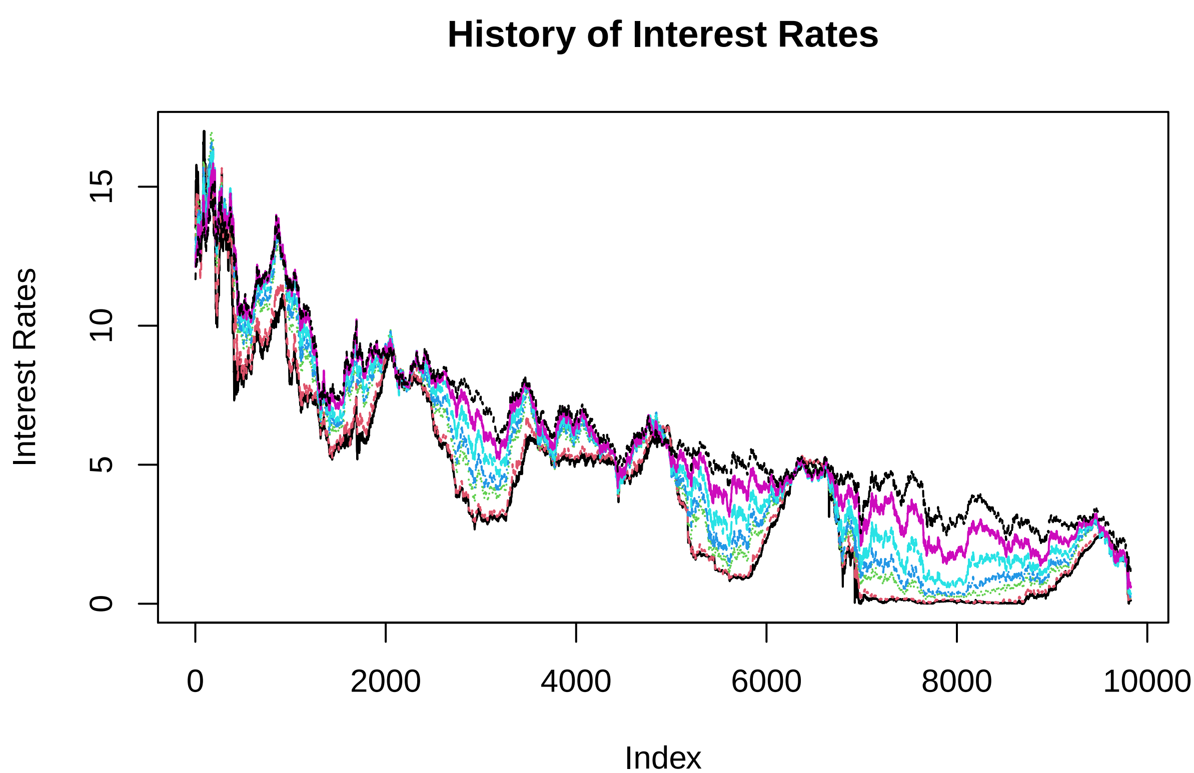
2020/11/30

1 Project data

```r
datapath = "C:/Users/Yue Sun/Desktop/project/PCA on 2020 US Treasury Yields to Maturity/"
Data<-
  read.csv(file=paste(datapath,"2020 US Treasury Yields to Maturity.csv",sep="/"),
           row.names=1,header=TRUE,sep=",")[,1:7]
dim(Data)
```

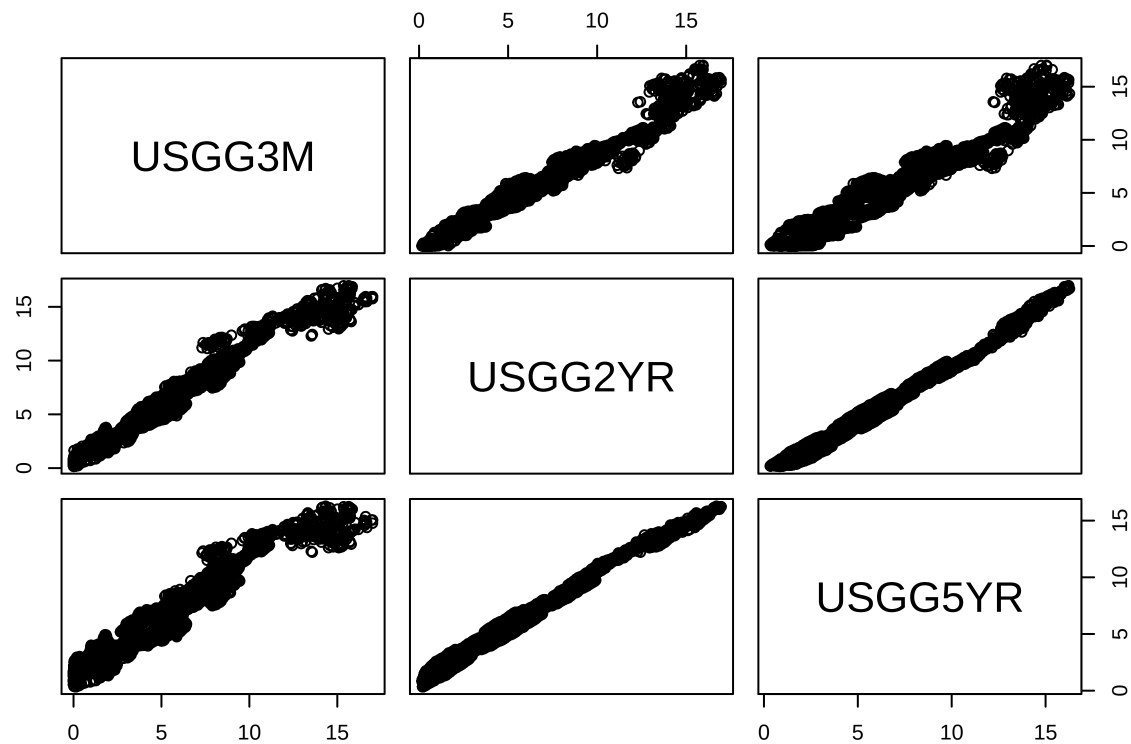
```
## [1] 9828    7
```

```r
View(Data)
matplot(Data[,-c(8,9,10)],type='l',ylab="Interest Rates",
        main="History of Interest Rates",xlab="Index")
```



2 Manual PCA # Perform PCA using eigenvalue decomposition ## 2.1 Dimension of the subset # Explore the dimensionality of the set of 3M, 2Y and 5Y yields

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




```r
#install.packages("rgl")
library(rgl);rgl.points(Data.3M_2Y_5Y)
```

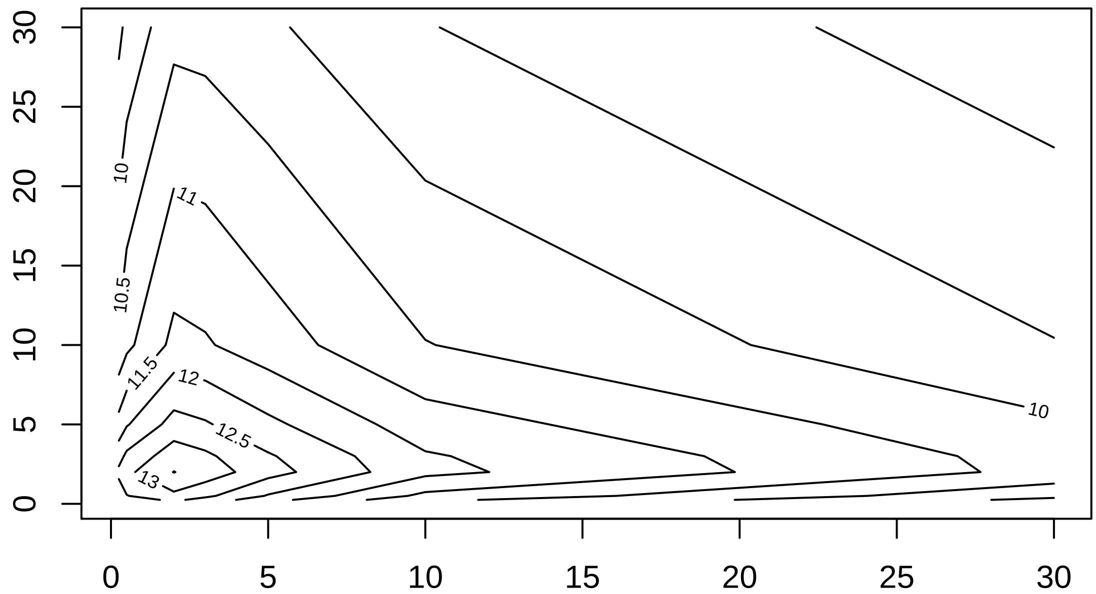
2.2 Covariance matrix

Analyze covariance matrix of the data. Compare results of manual calculation and cov().

```r
# Manual
Manual.Covariance.Matrix<-t(apply(Data,2,
                                  function(z) z-mean(z)))%*%
  (apply(Data,2,
         function(z) z-mean(z)))/(length(Data[,1])-1)
# Using cov
Covariance.Matrix<-cov(Data)
# Check if the 2 matrices are identical up to 11 decimals.
identical(round(Manual.Covariance.Matrix,11),round(Covariance.Matrix,11))
```

```
## [1] TRUE
```

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



2.3 Eigenvalue decomposition # Perform the PCA by calculating factors, loadings and analyzing the importance of factors.

Find eigenvalues and eigenvectors. Calculate vector of means (zero loading), first 3 loadings and 3 factors.

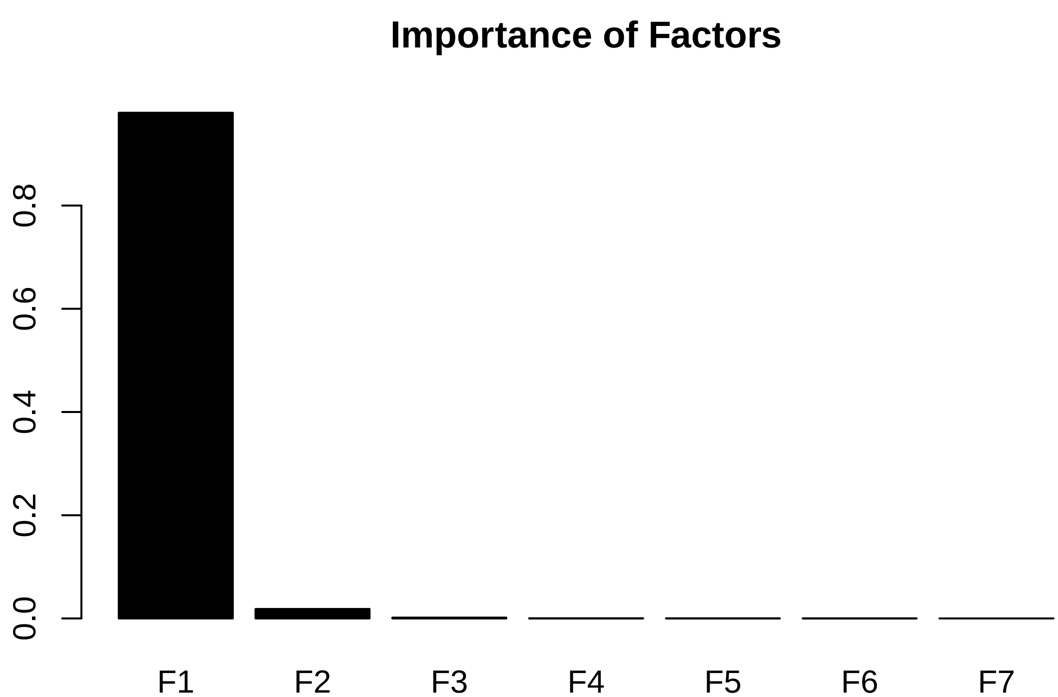
Print and visualize importance of the factors.

```r
Eigen.Decomposition = eigen(Covariance.Matrix)
Eigen.Decomposition
```

```
## eigen() decomposition
## $values
## [1] 8.080542e+01 1.514295e+00 1.290601e-01 1.505182e-02 7.664463e-03
## [6] 2.227497e-03 8.972045e-04
##
## $vectors
##           [,1]         [,2]       [,3]        [,4]       [,5]        [,6]
## [1,] 0.3799165 -0.496752921  0.5199053  0.51282309  0.2649450 -0.07290584
## [2,] 0.3883198 -0.433437102  0.1552497 -0.56402655 -0.5393875  0.16226922
## [3,] 0.4080861 -0.139091164 -0.4117364 -0.28964140  0.4560002 -0.31142982
```

```
## [4,] 0.4021569 -0.001448617 -0.4516241  0.06051751  0.1906191  0.10794874
## [5,] 0.3857975  0.212796304 -0.2841039  0.46253897 -0.3422184  0.49070565
## [6,] 0.3557289  0.434704099  0.1342314  0.11793691 -0.3840652 -0.70623885
## [7,] 0.3181572  0.558364077  0.4830856 -0.32168401  0.3605191  0.34662173
##              [,7]
## [1,] -0.009993044
## [2,] -0.042911952
## [3,]  0.505742601
## [4,] -0.763316716
## [5,]  0.391554964
## [6,] -0.079264001
## [7,] -0.005713468
```

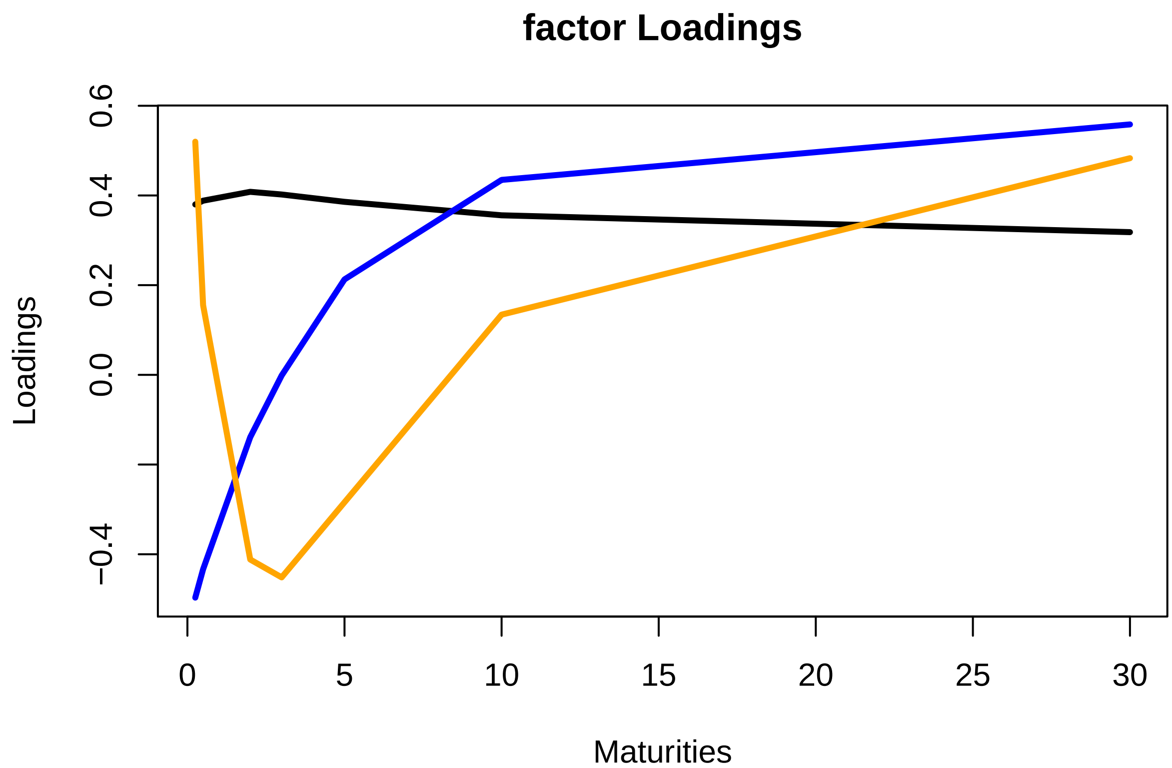
```
barplot(Eigen.Decomposition$values/sum(Eigen.Decomposition$values),
        width=2,col = "black",names.arg=c("F1","F2","F3","F4","F5","F6","F7"),
        main="Importance of Factors")
```



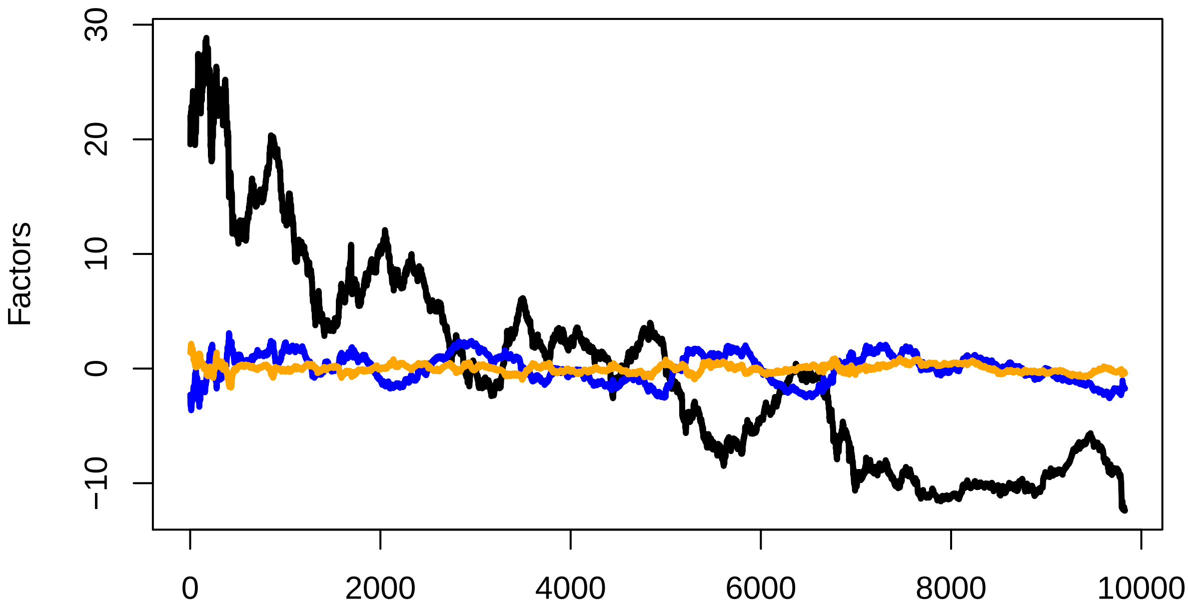

```
(cumulativeImportance<-cumsum(Eigen.Decomposition$values)/
  sum((Eigen.Decomposition$values)))
```

```
## [1] 0.9797611 0.9981218 0.9996867 0.9998692 0.9999621 0.9999891 1.0000000
```

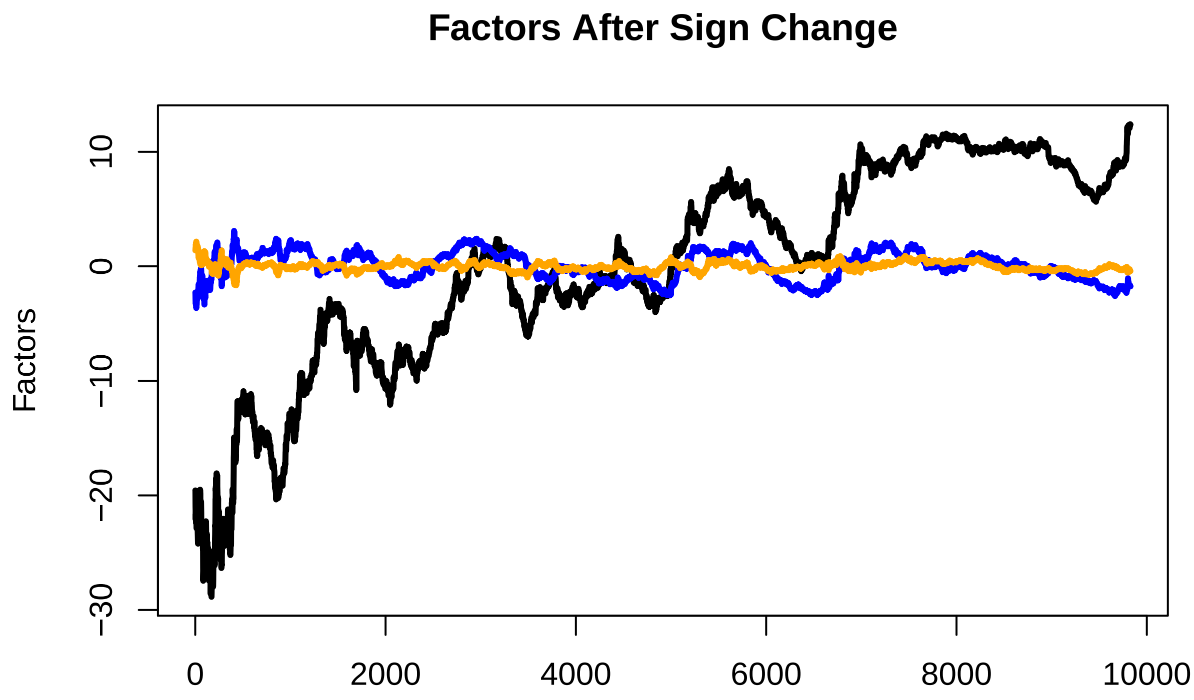
```
# Plot the loadings.
Loadings = Eigen.Decomposition$vectors[,1:3]
matplot(Maturities,Loadings,type="l",lty=1,col=c("black","blue","orange"),
        lwd=3,main="factor Loadings")
```



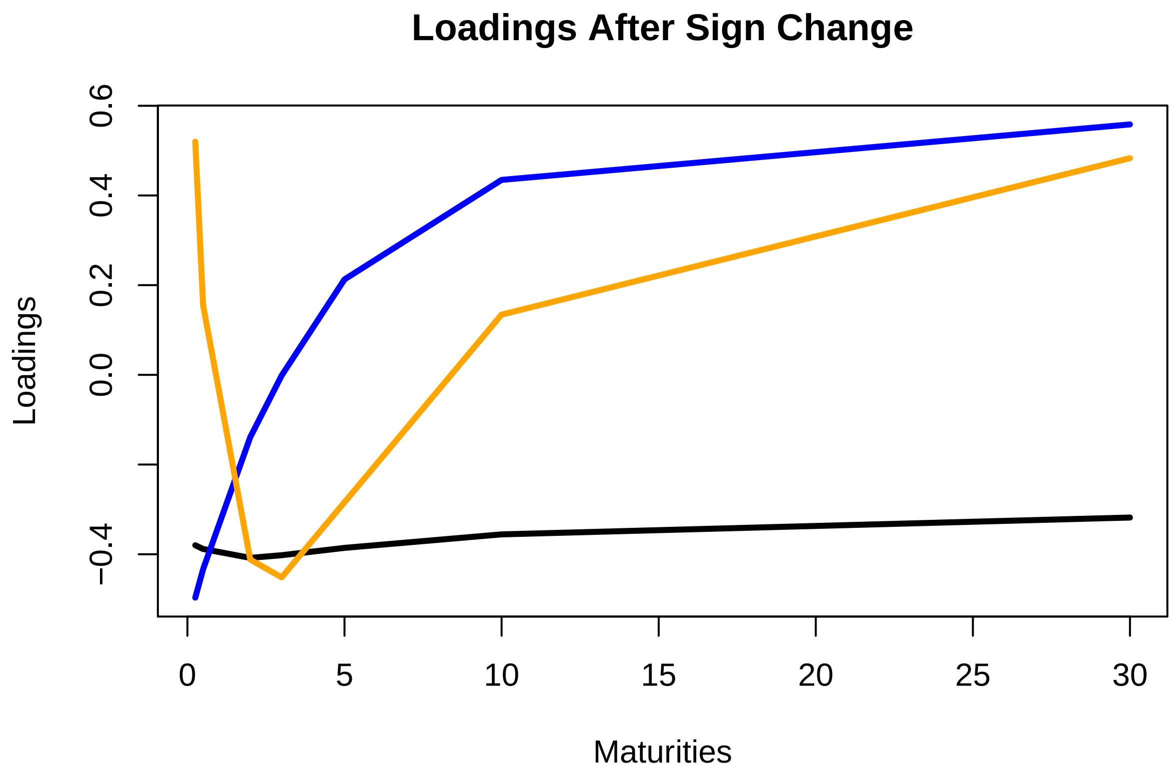

```
View(Loadings)

# abline(h=0)
# legend("bottomright",legend=c("L1","L2","L3"),
#        col=c("black","blue","orange"),lty=1,cex=.5)

y0 = apply(Data,2, function(z) z-mean(z))
#View(y0)
Factors = y0 %*% Loadings


# Interpret the factors by looking at the shapes of the loadings.
# Calculate and plot 3 selected factors
matplot(Factors,type="l",col=c("black","blue","orange"),lty=1,lwd=3)
```



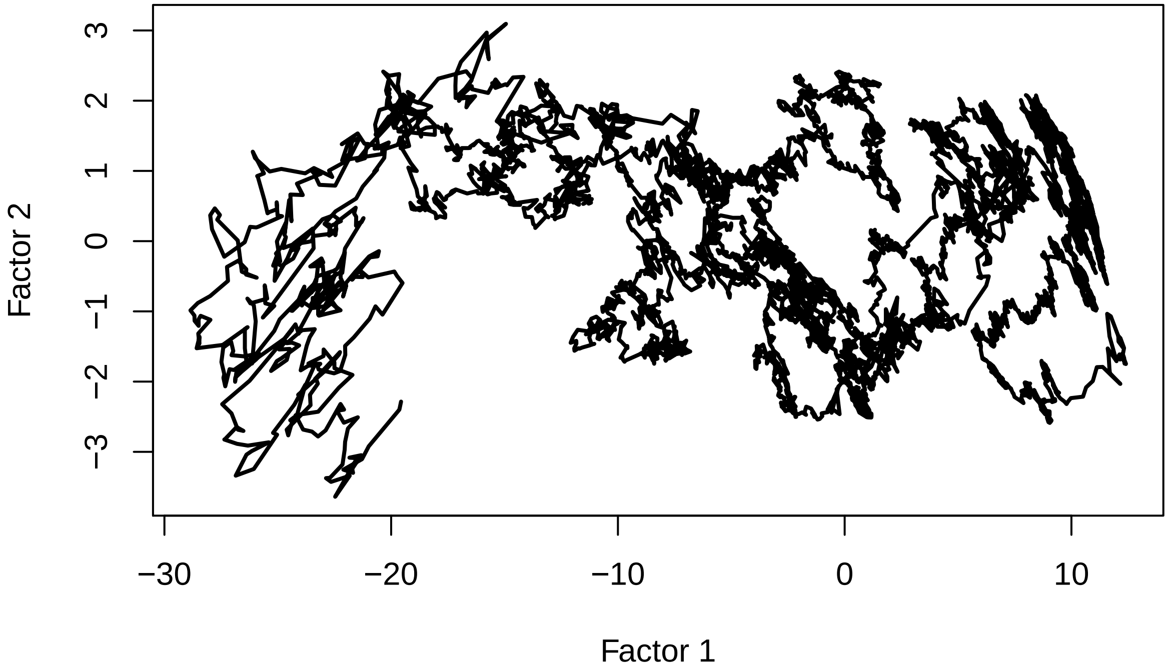

```
# legend("topright",legend=c("F1","F2","F3"),
#        col=c("black","blue","orange"),lty=1,cex=.5)

# Change the signs of the first factor and the corresponding factor loading.
Loadings[,1]<--Loadings[,1]
Factors[,1]<--Factors[,1]

matplot(Factors,type="l",col=c("black","blue","orange"),lty=1,lwd=3,
        main="Factors After Sign Change")
```




```r
# legend("bottomright",legend=c("F1","F2","F3"),
#        col=c("black","blue","orange"),lty=1,cex=.5)

matplot(Maturities,Loadings,type="l",lty=1,col=c("black","blue","orange"),lwd=3,main="Loadings After Si
```

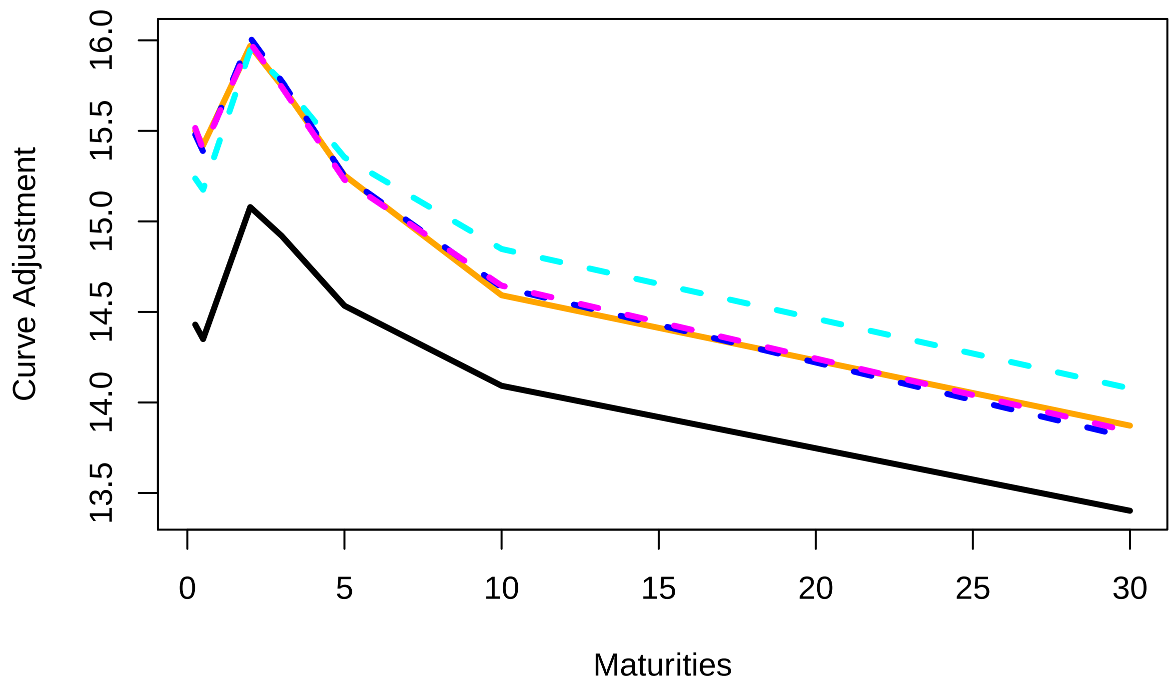



```
# abline(h=0)
# legend("right",legend=c("L1","L2","L3"),
#        col=c("black","blue","orange"),lty=1,cex=.5)

plot(Factors[,1:2],type="l",lwd=2, xlab="Factor 1",ylab="Factor 2")
```



Analyze the adjustments that each factor makes to the term curve.

```
OldCurve<-Data[135,]
NewCurve<-Data[136,]
CurveChange<-NewCurve-OldCurve
FactorsChange<-Factors[136,]-Factors[135,]
ModelCurveAdjustment.1Factor<-OldCurve+t(Loadings[,1])*FactorsChange[1]
ModelCurveAdjustment.2Factors<-ModelCurveAdjustment.1Factor+
  t(Loadings[,2])*FactorsChange[2]
ModelCurveAdjustment.3Factors<-ModelCurveAdjustment.2Factors+
  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","orange","cyan","blue","magenta"),
        lwd=3,ylab="Curve Adjustment")
```



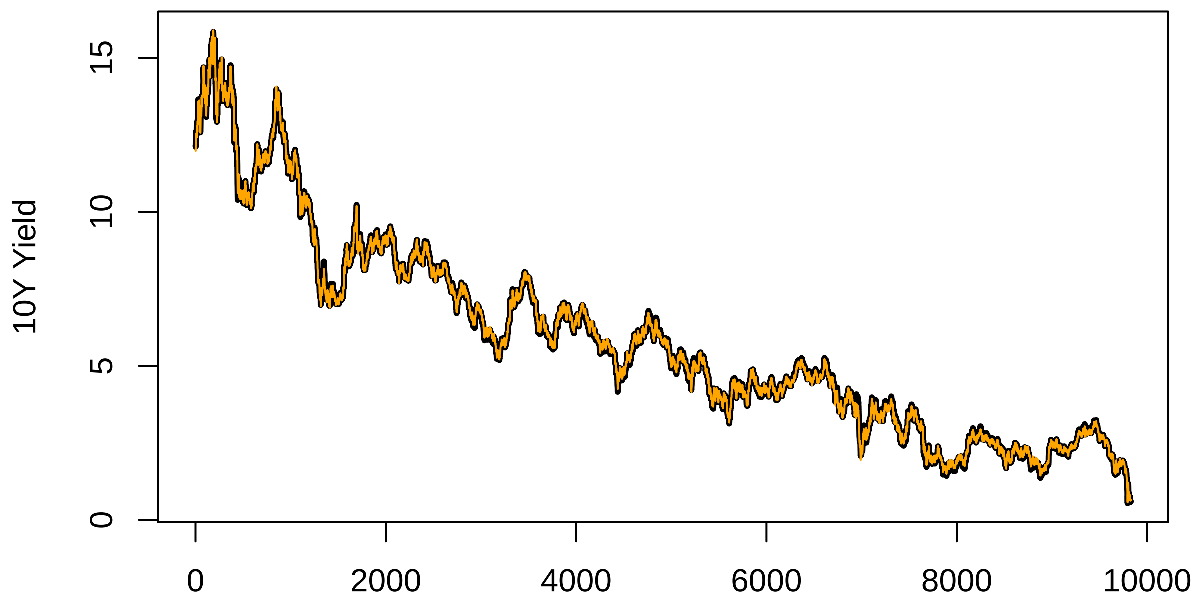

```
#legend(x="topright",
#       c("Old Curve","New Curve","1-Factor Adj.",
#         "2-Factor Adj.","3-Factor Adj."),
#       lty=c(1,1,2,2,2),lwd=3,
#       col=c("black","orange","cyan","blue","magenta"),cex=.5)
rbind(CurveChange,ModelCurveAdjustment.3Factors-OldCurve)
```

```
##              USGG3M   USGG6M   USGG2YR   USGG3YR   USGG5YR  USGG10YR USGG30YR
## 7/20/1981 1.070000 1.070000 0.8900000 0.8300000 0.7200000 0.5000000   0.4700
## 7/17/1981 1.085851 1.047054 0.9055009 0.8230115 0.6957089 0.5532926   0.4379
```

```
# Explain how shapes of the loadings affect the adjustnents using only
# factor 1, factors 1 and 2, and all 3 factors.
# See the goodness of fit for the example of 10Y yield.

# How close is the approximation for each maturity?
# 10Y
cbind(Maturities,Loadings)
```

```
##      Maturities
## [1,]       0.25 -0.3799165 -0.496752921  0.5199053
## [2,]       0.50 -0.3883198 -0.433437102  0.1552497
## [3,]       2.00 -0.4080861 -0.139091164 -0.4117364
## [4,]       3.00 -0.4021569 -0.001448617 -0.4516241
## [5,]       5.00 -0.3857975  0.212796304 -0.2841039
## [6,]      10.00 -0.3557289  0.434704099  0.1342314
## [7,]      30.00 -0.3181572  0.558364077  0.4830856
```

```r
Means = apply(Data, 2, mean)
#View(Means)

Model.10Y<-Means[6]+Loadings[6,1]*Factors[,1]+Loadings[6,2]*Factors[,2]+
  Loadings[6,3]*Factors[,3]

matplot(cbind(Data[,6],Model.10Y),type="l",lty=1,lwd=c(3,1),col=c("black","orange"),ylab="10Y Yield")
```




```r
# legend("topright",legend=c("Actual","Approximation"),
#        col=c("black","orange"),lty=1,cex=.5)
```

3 PCA using princomp()

Repeat PCA using princomp.

```r
PCA.Yields<-princomp(Data)
names(PCA.Yields)
```

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

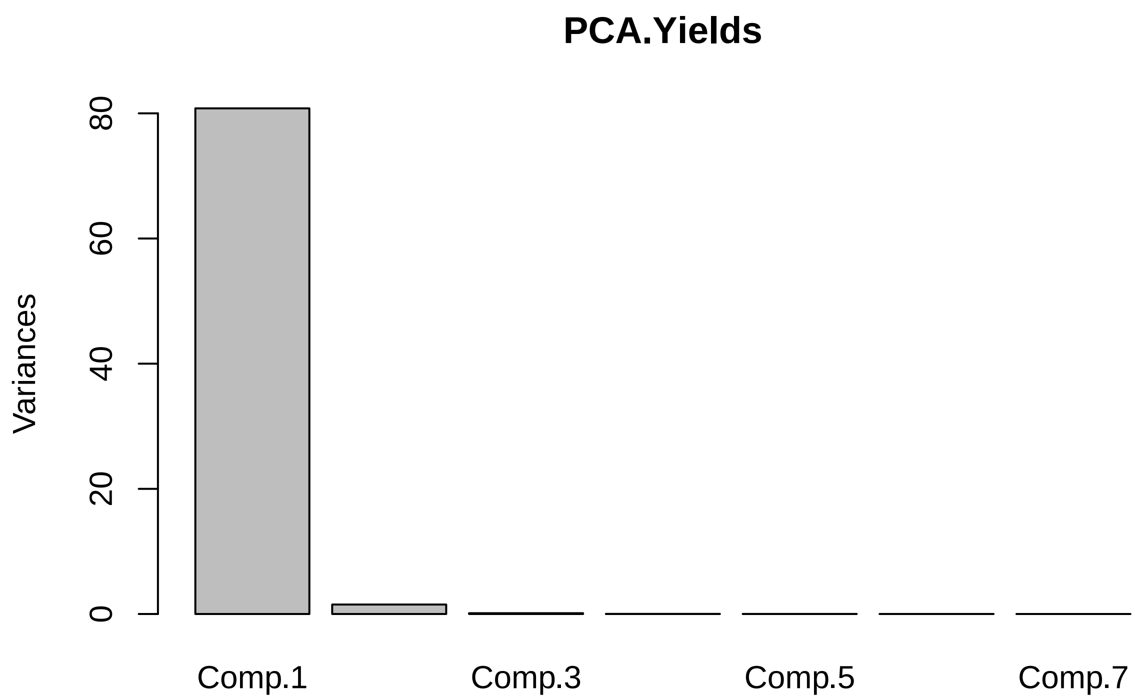
3.1 Importance of factors

```r
summary(PCA.Yields)
```

```
## Importance of components:
```

```
##                          Comp.1     Comp.2      Comp.3       Comp.4
## Standard deviation     8.9887263 1.23050422 0.359231033 0.1226796260
## Proportion of Variance 0.9797611 0.01836074 0.001564846 0.0001825025
## Cumulative Proportion  0.9797611 0.99812183 0.999686680 0.9998691820
##                              Comp.5       Comp.6       Comp.7
## Standard deviation     8.754246e-02 4.719396e-02 2.995185e-02
## Proportion of Variance 9.293117e-05 2.700827e-05 1.087855e-05
## Cumulative Proportion  9.999621e-01 9.999891e-01 1.000000e+00
```

```r
plot(PCA.Yields)
```



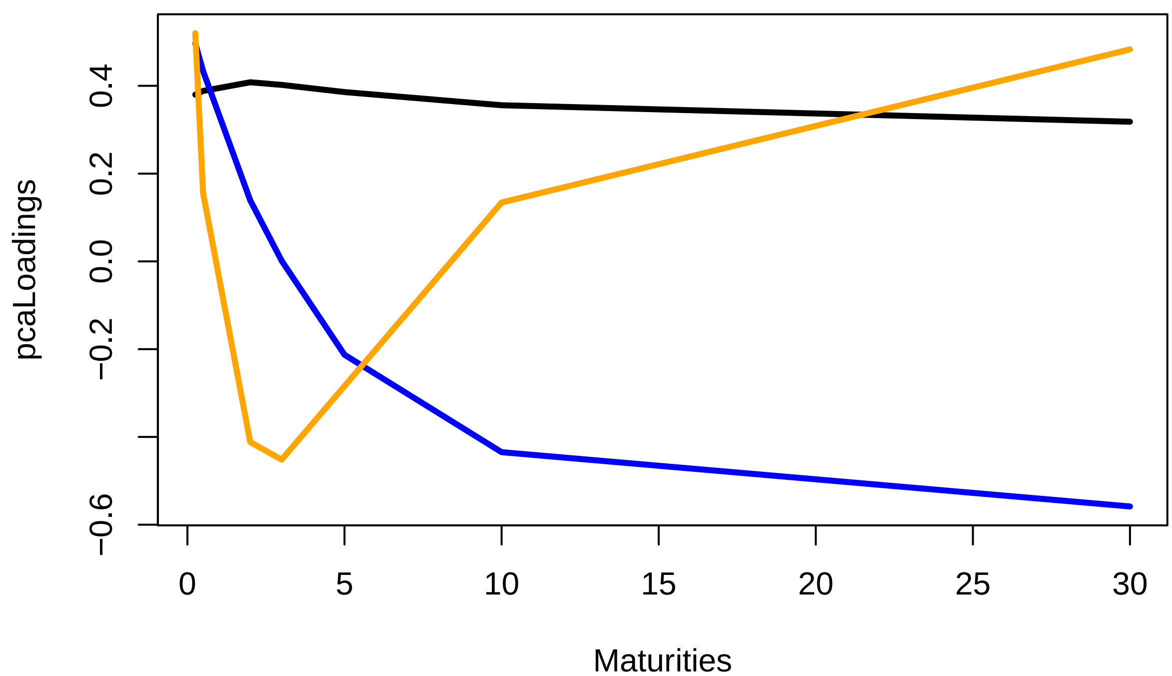
3.2 Factors and Loadings

```r
# Create first 3 loadings and factors.
pcaLoadings<-PCA.Yields$loadings[,1:3]
pcaLoading0<-PCA.Yields$center
pcaFactors<-PCA.Yields$scores[,1:3]
# Compare the loadings.
loadingsComparison<-cbind(pcaLoadings,Maturities,Eigen.Decomposition$vectors[,1:3])
colnames(loadingsComparison)<-c(paste0("PCA",1:3),"Maturity",paste0("Manual",1:3))
loadingsComparison
```

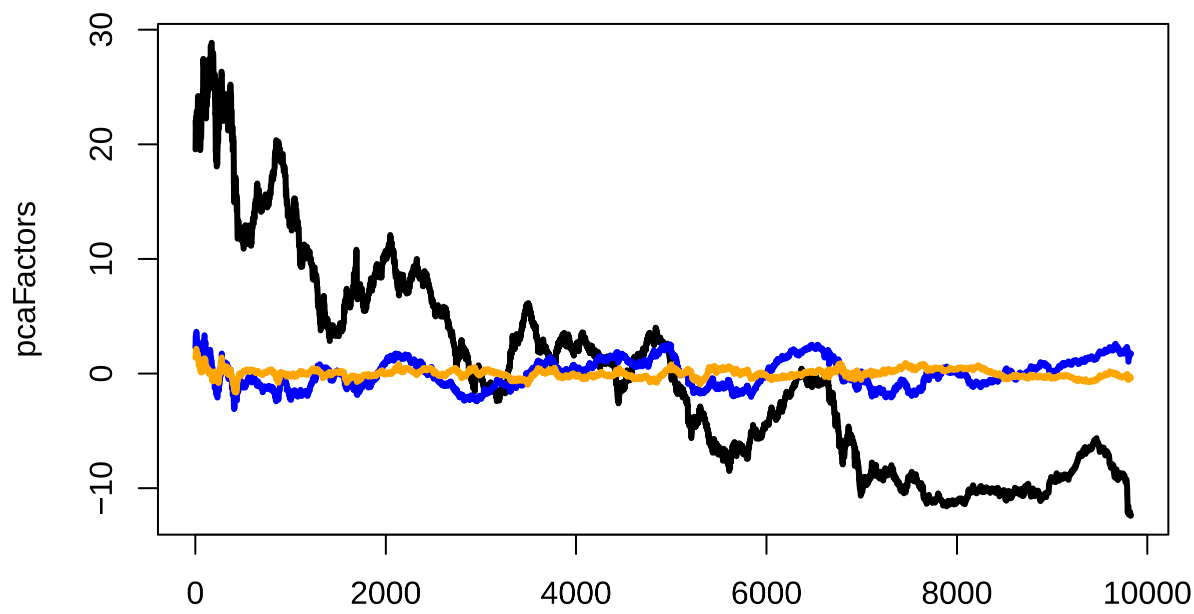
```
##              PCA1         PCA2       PCA3 Maturity   Manual1      Manual2
## USGG3M   0.3799165  0.496752921  0.5199053     0.25 0.3799165 -0.496752921
## USGG6M   0.3883198  0.433437102  0.1552497     0.50 0.3883198 -0.433437102
## USGG2YR  0.4080861  0.139091164 -0.4117364     2.00 0.4080861 -0.139091164
## USGG3YR  0.4021569  0.001448617 -0.4516241     3.00 0.4021569 -0.001448617
## USGG5YR  0.3857975 -0.212796304 -0.2841039     5.00 0.3857975  0.212796304
```

```
## USGG10YR 0.3557289 -0.434704099  0.1342314    10.00 0.3557289  0.434704099
## USGG30YR 0.3181572 -0.558364077  0.4830856    30.00 0.3181572  0.558364077
##             Manual3
## USGG3M    0.5199053
## USGG6M    0.1552497
## USGG2YR  -0.4117364
## USGG3YR  -0.4516241
## USGG5YR  -0.2841039
## USGG10YR  0.1342314
## USGG30YR  0.4830856
```

```r
matplot(Maturities,pcaLoadings,type="l",col=c("black","blue","orange"),lty=1,lwd=3)
```




```r
# legend("right",legend=c("L1","L2","L3"),col=c("black","blue","orange"),lty=1,cex=.5)
matplot(pcaFactors,type="l",col=c("black","blue","orange"),lwd=3,lty=1)
```




```
# legend("topright",legend=c("F1","F2","F3"),col=c("black","blue","orange"),lty=1,cex=.5)
#Change the signs of the first factor and factor loading again if necessary.
```

3.3 Loadings by regression

```
# Obtain factor loadings using linear regression.
centeredData<-t(apply(Data,1,function(z) z-pcaLoading0))
# Reconstruct loading 1.
reconstructLoading1<-lm(pcaFactors[,1]~centeredData)$coefficients[-1]
cbind(reconstructLoading1,pcaLoadings[,1])
```

```
##                        reconstructLoading1
## centeredDataUSGG3M               0.3799165 0.3799165
## centeredDataUSGG6M               0.3883198 0.3883198
## centeredDataUSGG2YR              0.4080861 0.4080861
## centeredDataUSGG3YR              0.4021569 0.4021569
## centeredDataUSGG5YR              0.3857975 0.3857975
## centeredDataUSGG10YR             0.3557289 0.3557289
## centeredDataUSGG30YR             0.3181572 0.3181572
```

```
# Reconstruct loading 2.
reconstructLoading2<-lm(pcaFactors[,2]~centeredData)$coefficients[-1]
cbind(reconstructLoading2,pcaLoadings[,2])
```

```
##                        reconstructLoading2
## centeredDataUSGG3M             0.496752921  0.496752921
## centeredDataUSGG6M             0.433437102  0.433437102
```

```
## centeredDataUSGG2YR          0.139091164  0.139091164
## centeredDataUSGG3YR          0.001448617  0.001448617
## centeredDataUSGG5YR         -0.212796304 -0.212796304
## centeredDataUSGG10YR        -0.434704099 -0.434704099
## centeredDataUSGG30YR        -0.558364077 -0.558364077
```

```
# Reconstruct loading 3.
reconstructLoading3<-lm(pcaFactors[,3]~centeredData)$coefficients[-1]
cbind(reconstructLoading3,pcaLoadings[,3])
```

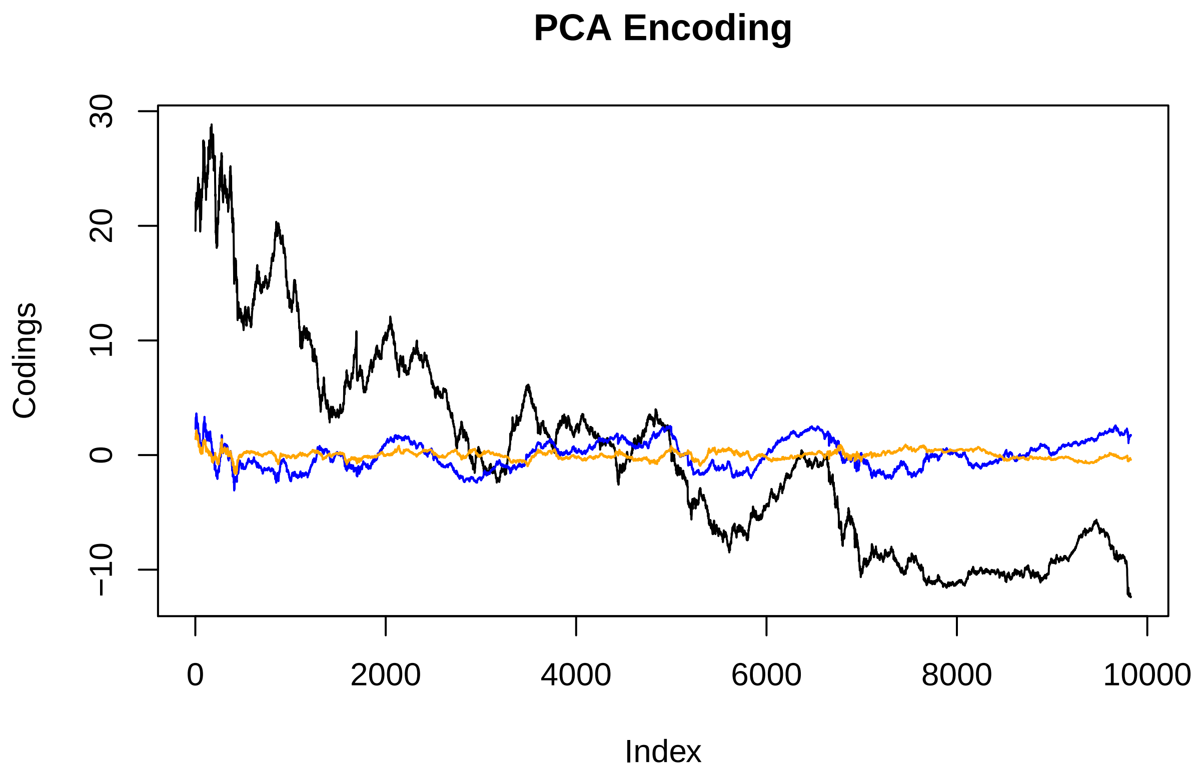
```
##                      reconstructLoading3
## centeredDataUSGG3M             0.5199053  0.5199053
## centeredDataUSGG6M             0.1552497  0.1552497
## centeredDataUSGG2YR           -0.4117364 -0.4117364
## centeredDataUSGG3YR           -0.4516241 -0.4516241
## centeredDataUSGG5YR           -0.2841039 -0.2841039
## centeredDataUSGG10YR           0.1342314  0.1342314
## centeredDataUSGG30YR           0.4830856  0.4830856
```

##4 Encoder-decoder interpretation # PCA has an interpretation of encoder-decoder, a powerful concept in Machine Learning.

4.1 Encoder

Obtain 3 factors from the original data.

```
codings<-centeredData%*% pcaLoadings
matplot(1:dim(centeredData)[1],codings,type="l",lty=1,
        col=c("black","blue","orange"),
        main="PCA Encoding",ylab="Codings",xlab="Index")
```

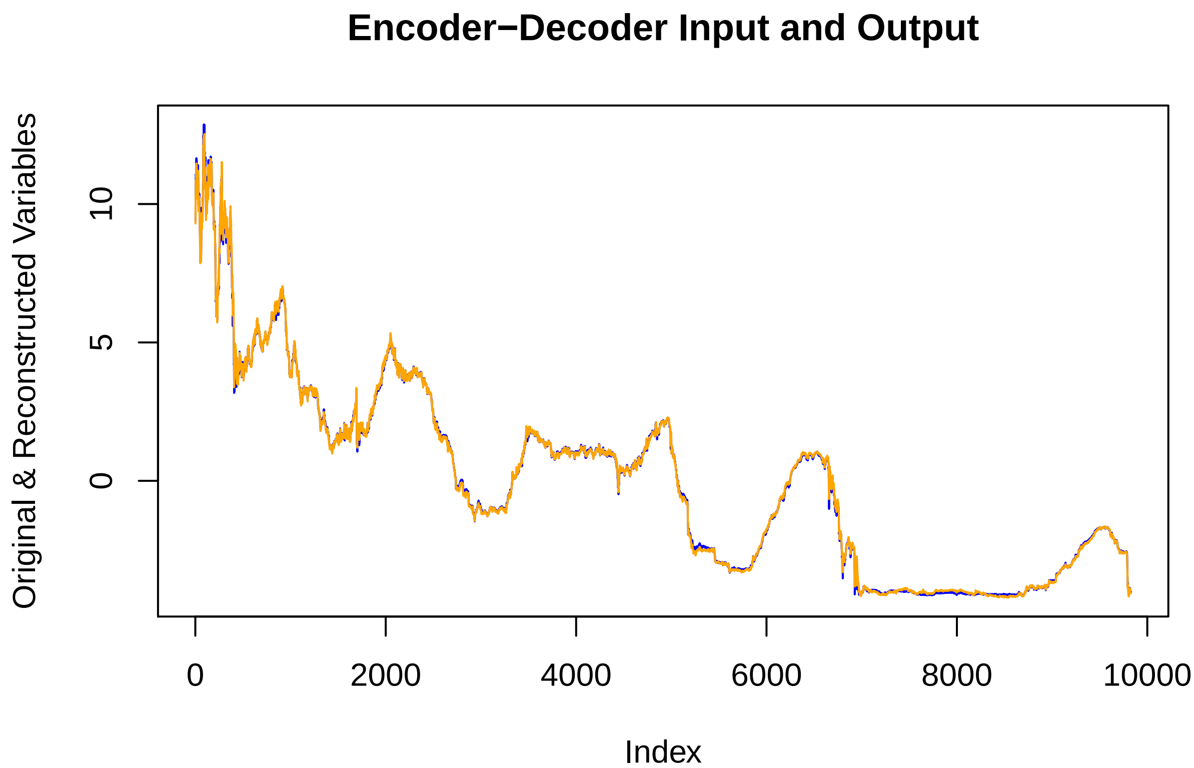



```
# legend("topright",legend=paste0("Code",1:3),lty=1,
#        col=c("black","blue","orange"),cex=.5)
```

4.2 Decoder

Obtain reconstructions of the original centered data from encodings.

```
decodings<-codings%*%t(pcaLoadings)
matplot(1:dim(centeredData)[1],cbind(centeredData[,1],decodings[,1]),
        type="l",lty=1,col=c("blue","orange"),
        main="Encoder-Decoder Input and Output",
        ylab="Original & Reconstructed Variables",xlab="Index")
```



Encoder−Decoder Input and Output
Original & Reconstructed Variables
0
5
10
0
2000
4000
6000
8000
10000
Index