Sample R Code from the reading "Principal Components Analysis in Finance Modeling the Dynamics of Interest Rates (2016)" and Principal Components Analysis (PCA) of US Treasury Yields over a 5-Year Period

Peter J. Kempthorne

September 19, 2022

This note is intensionally terse with the simple objective of illustrating R commands for plotting and computations. Some comments are included directly in the R code (after hashtags) to help explain the operations.

In section 1 we show the R code used to make the first two figures in the reading "Principal Components in Finance: Modeling the Dynamics of Interest Rates" (2016). Sweave files in RStudio can include "chunks" of R code. The graphical output from these code chunks are included in the Sweave document. All printed output from the code chunks are included in the Sweave document as well. In the reading, code chunks were executed with the option "echo=FALSE" to hide the R commands creating the output. In this note, the option "echo=TRUE" is used to show all commands.

Section 2 details the R commands for conducting the principal components analysis (PCA) of the Federal Reserve (FRED) yield data, extending the period of the PCA in the reading from 6 days to 5 years. These computation require the R data file"fm_freddata.Rdata", which is created by running (or compiling) the r script file "fm_freddata .r."

Section 3 illustrates the built-in R command *princomp*() and displays output that matches output produced in Section 2.

1 R Code for Figures 1 and 2 of the Reading

```
> # 0.1 Install packages ----
> #     Set ind.install0 to TRUE if running script for first time on a computer
> #     or updating the packages
> ind.install0<-FALSE
```

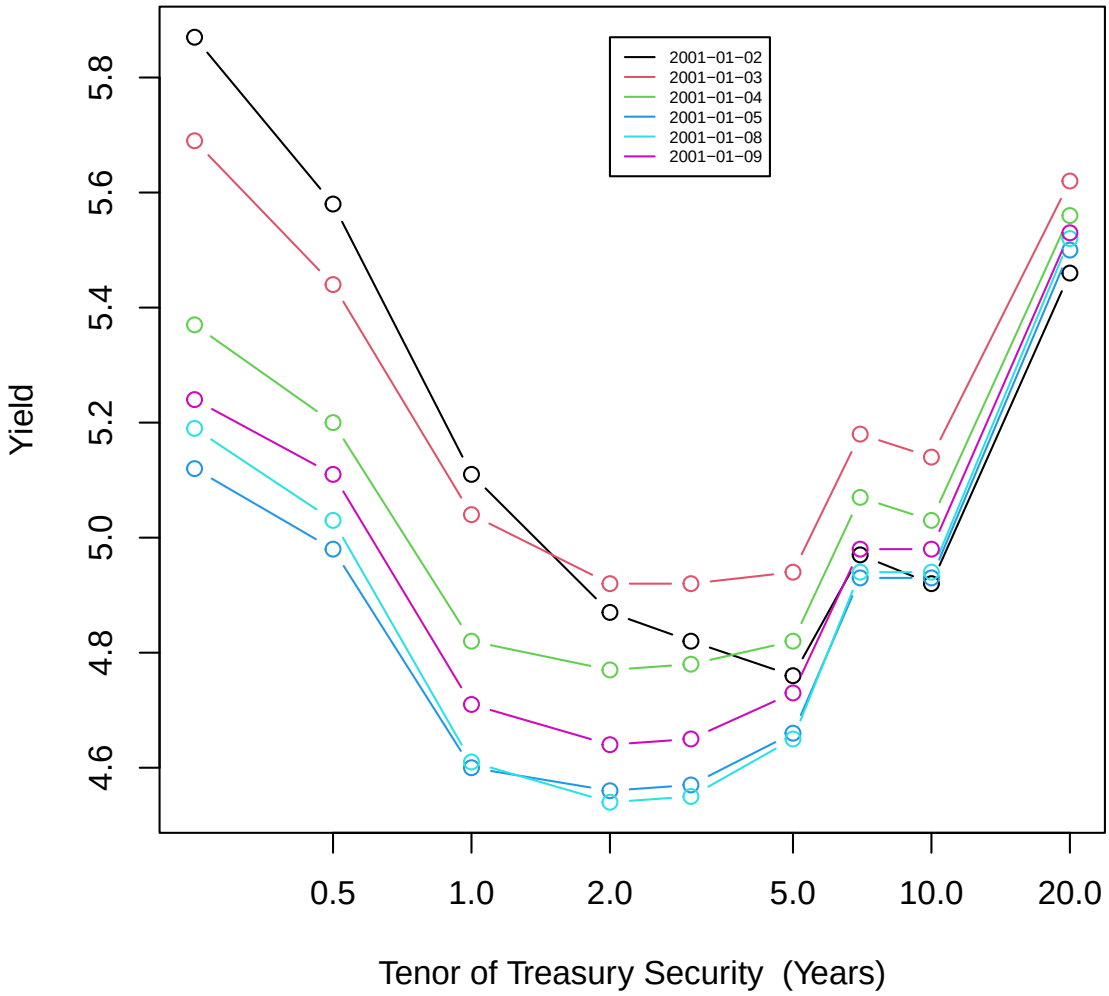
```
> #
> if (ind.install0){
+ install.packages("quantmod")
+ install.packages("tseries")
+ install.packages("vars")
+ install.packages("fxregime")
+ }
> # 0.1 Load packages into R session ----
>
> library("quantmod")
> library("tseries")
> library("vars")
> library("fxregime")
> library("graphics")
> library("tidyverse")
> # load the R workspace created by the script file
> #   fm_freddata.r
>
> dbnames0<-load(file="fm_freddata.RData")
> #print(dbnames0)
> #head(fred.data00)
>
>

> fred.data00.period1<-window(fred.data00, start=as.Date("2001-01-01"), end=as.Date("2005-12
> fred.data00.0<-na.omit(fred.data00.period1)
> # Create zoo objects:
> fred.data00.diff=zoo(diff((coredata(fred.data00.0))), order.by=time(fred.data00.0)[-1])
> fred.data00=zoo((coredata(fred.data00.0)), order.by=time(fred.data00.0))
>
> #head(fred.data00.diff)
> #head(fred.data00.0)
```

Figure 1 displays the Treasury Yield Curve for each of the first six (business) days of 2001. There is a curve for each day. Figure 2 displays how the yield curve changes from day to day by plotting the yield for each *tenor* over the six days.

```
> options(width=80)
> #
> par(mfcol=c(1,1))
> #
> #head(fred.data00)
> fred.tenors<-c(1/4, 1/2, 1,2,3,5,7,10,20)
> A00<-t(fred.data00.0[1:6,])
> A00.time<-time(fred.data00.0[1:6,])
> plot(fred.tenors, A00[,1],type="b", log="x",ylab="Yield", xlab="Tenor of Treasury Security
```

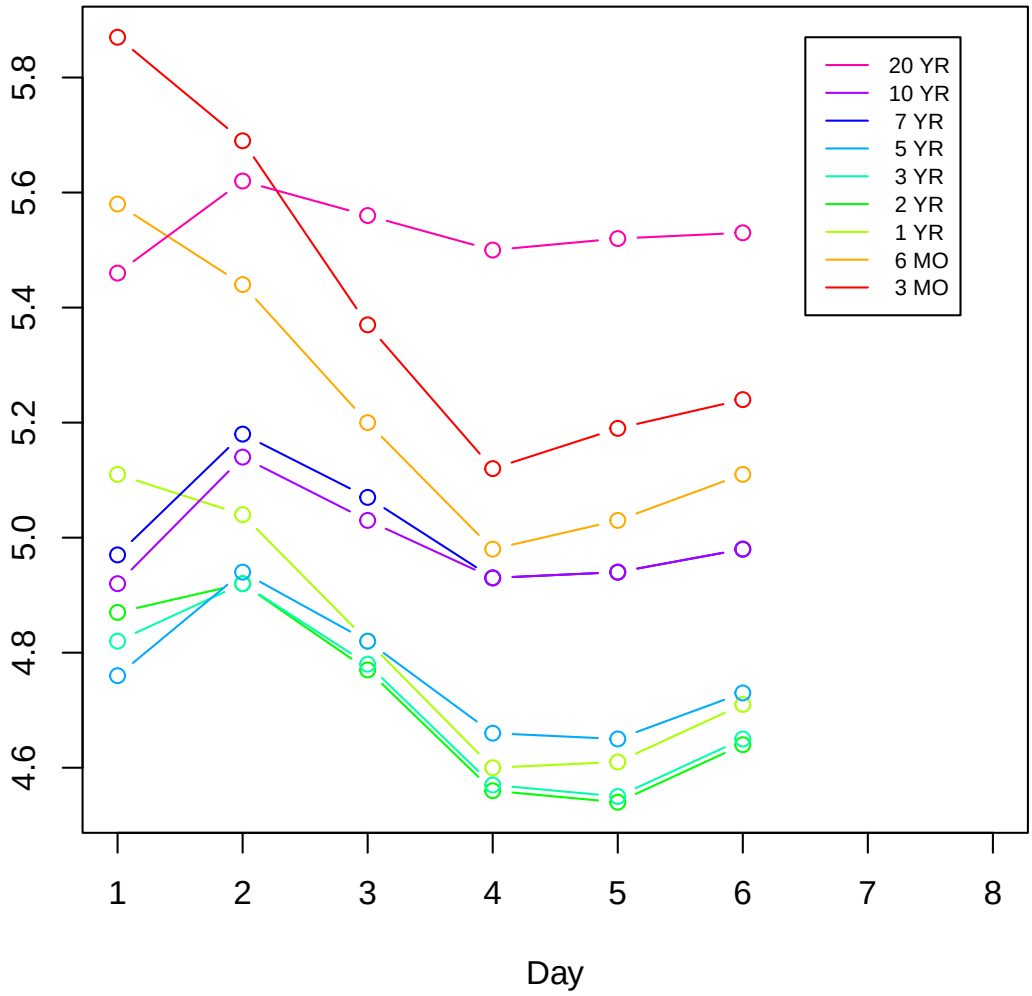
```
+      main="Figure 1. Yield Curve For 9 US Treasury Securities\n on 6 Days in 2001",
+      ylim=c(min(A00), max(A00))
+      )
> lines(fred.tenors, A00[,2], type="b", col=2)
> lines(fred.tenors, A00[,3], type="b", col=3)
> lines(fred.tenors, A00[,4], type="b", col=4)
> lines(fred.tenors, A00[,5], type="b", col=5)
> lines(fred.tenors, A00[,6], type="b", col=6)
> #legend(x=0.25,y=7,
>  legend(x=2,y=max(A00),
+        legend=dimnames(A00)[[2]],
+
+        lty=rep(1,times=6),
+        col=c(1:6),
+        cex=0.5)
>
```

Figure 1. Yield Curve For 9 US Treasury Securities on 6 Days in 2001



```
> ts.plot(as.ts(t(A00)), xlab="Day",type="b",
+         col=rainbow(NROW(A00)),bg="black",
+         main=paste("Figure 2. Six Days of Yields\nfor Nine Treasury Securities", "(https:/
> #)
> legend.fred<-c(" 3 MO"," 6 MO", paste(c(" 1"," 2"," 3" ," 5"," 7" ,"10","20")," YR",sep="'
> legend(x=6.5,y=max(A00),
+        legend=rev(legend.fred),
+        lty=rep(1,times=NROW(A00)),
+        col=rev(rainbow(NROW(A00))),
+        cex=0.70)
```

Figure 2. Six Days of Yields for Nine Treasury Securities (https://research.stlouisfed.org)

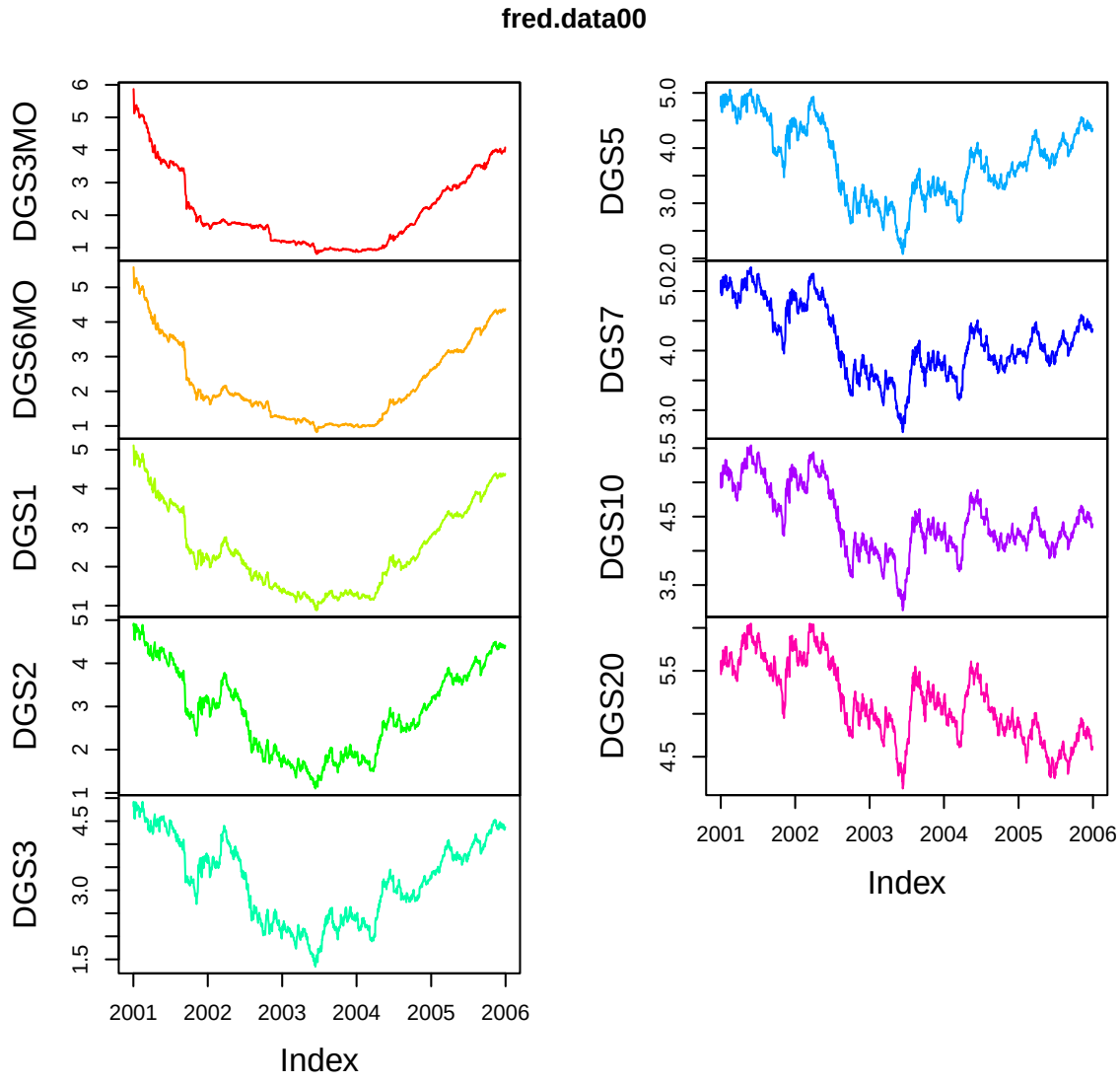



```
> dimnames(A00)[[1]]<-legend.fred
> dimnames(A00)[[2]]<-format(A00.time, format="%b %d")
```

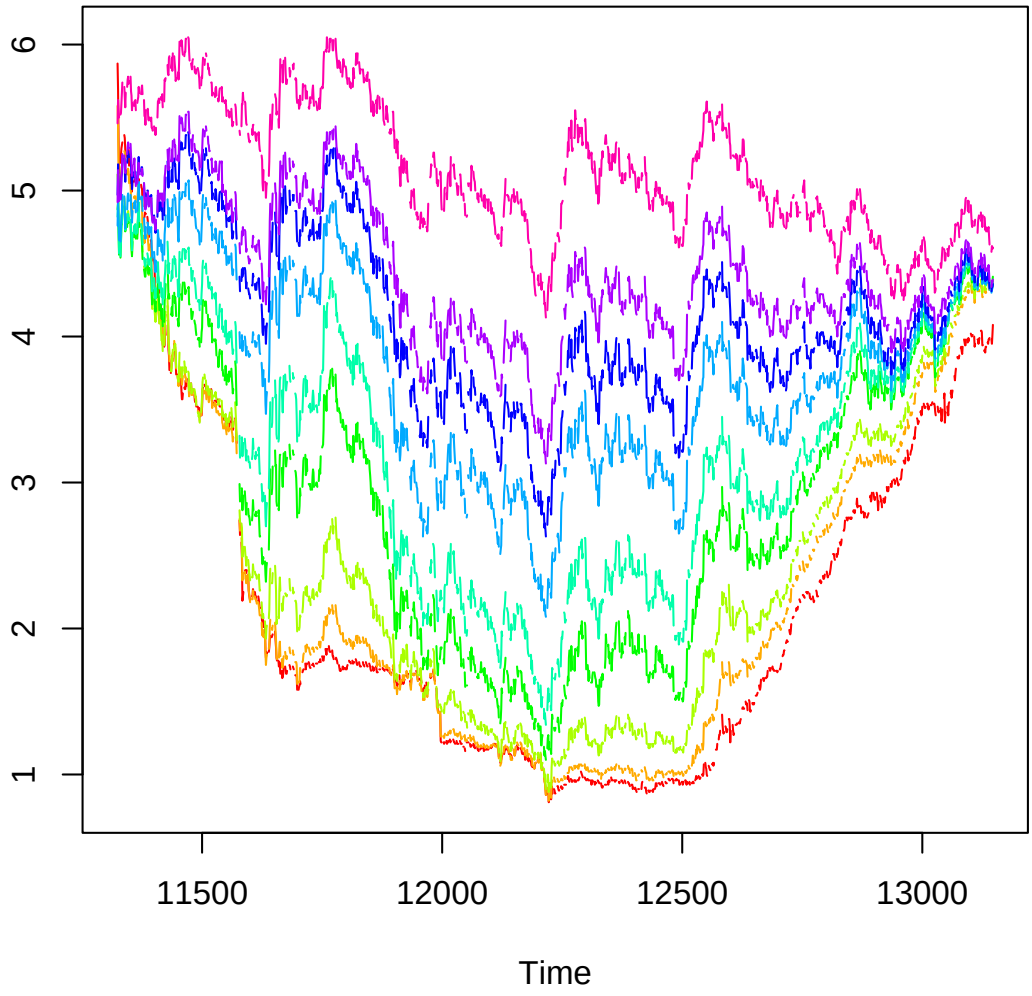
2 Analysis of Yields for 5-Year Period (2001-2005)

```
> par(mfcol=c(1,1)) # create plots in  1x1 panels
> # Default plot() displays each column as time series
```

```
> plot(fred.data00,col=rainbow(NCOL(fred.data00)))
```

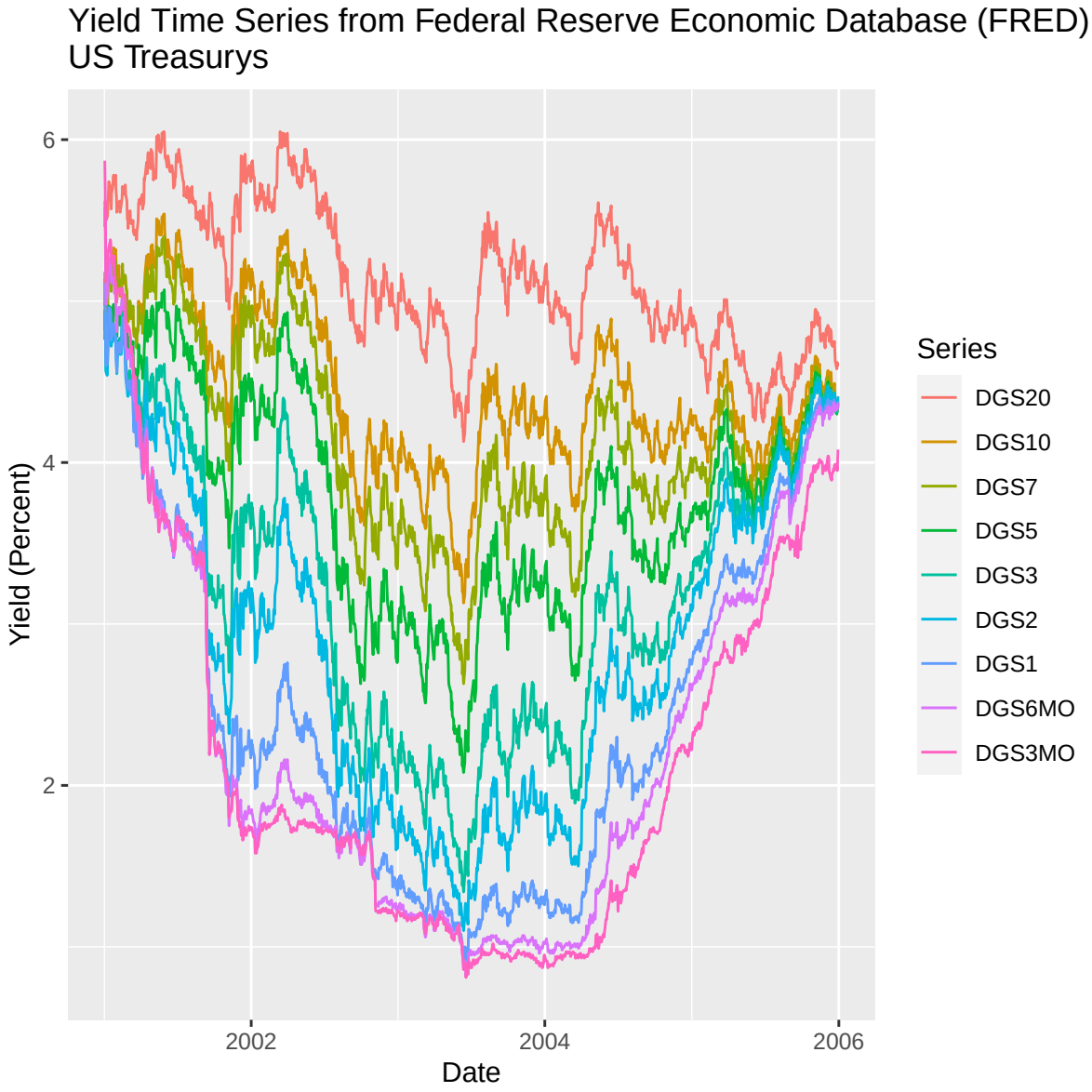



```
> # Default ts.plot displays all time series together
> #     (time axis is julian date  which equals
> #       day count since default origin, 1/1/1970 ))
> ts.plot(fred.data00,col=rainbow(NCOL(fred.data00)))
>
>
```




```
> # Replot using ggplot
>
> list_yields<-names(fred.data00)
> yield_data <- data.frame(date=time(fred.data00),
+                          yield=fred.data00[,1],
+                          Series=list_yields[1])
> for (j in c(2:ncol(fred.data00))){
+   yield_data<- rbind(yield_data,
```

```
+                   data.frame(date=time(fred.data00),
+                              yield=fred.data00[,j],
+                              Series=list_yields[j])
+                   )
+   }
> yield_data<-data.frame(yield_data)
> yield_data$Series<- factor(yield_data$Series, levels=rev(list_yields))
> ggplot(yield_data, mapping=aes(date, y=yield , color=Series))  +
+   geom_line() +
+   xlab("Date") + ylab("Yield (Percent)") +
+   ggtitle("Yield Time Series from Federal Reserve Economic Database (FRED) \nUS Treasurys'
>
```




```
> # Set X0 to coredata of zoo object (i.e. data matrix)
> X0= coredata(fred.data00.diff)
> class(X0)

[1] "matrix" "array"

> head(X0)
```

```
     DGS3MO DGS6MO  DGS1  DGS2  DGS3  DGS5  DGS7 DGS10 DGS20
[1,]  -0.18  -0.14 -0.07  0.05  0.10  0.18  0.21  0.22  0.16
[2,]  -0.32  -0.24 -0.22 -0.15 -0.14 -0.12 -0.11 -0.11 -0.06
[3,]  -0.25  -0.22 -0.22 -0.21 -0.21 -0.16 -0.14 -0.10 -0.06
[4,]   0.07   0.05  0.01 -0.02 -0.02 -0.01  0.01  0.01  0.02
[5,]   0.05   0.08  0.10  0.10  0.10  0.08  0.04  0.04  0.01
[6,]   0.05   0.05  0.11  0.12  0.13  0.10  0.11  0.12  0.07
```

```
> tail(X0)
```

```
        DGS3MO DGS6MO  DGS1  DGS2  DGS3  DGS5  DGS7 DGS10 DGS20
[1242,]   0.00  -0.03 -0.03 -0.03 -0.05 -0.04 -0.05 -0.05 -0.06
[1243,]   0.01   0.00 -0.02 -0.03 -0.04 -0.06 -0.06 -0.06 -0.06
[1244,]  -0.01   0.03  0.02 -0.02 -0.03 -0.02 -0.02 -0.04 -0.05
[1245,]  -0.03  -0.02 -0.01  0.02  0.03  0.02  0.03  0.04  0.04
[1246,]   0.06   0.01  0.01  0.01  0.01  0.01  0.00 -0.01 -0.02
[1247,]   0.07   0.03  0.03  0.03  0.02  0.02  0.02  0.02  0.01
```

```
> # Subtract off column-means of X
> X=X0 - matrix(1, nrow=NROW(X0),ncol=1)%*% t(as.matrix(as.vector(colMeans(X0))))
```

The singular-value decomposition (SVD) of X has 3 components:

```
> X.svd=svd(X)
> names(X.svd)
```

```
[1] "d" "u" "v"
```

```
> print(X.svd$d)
```

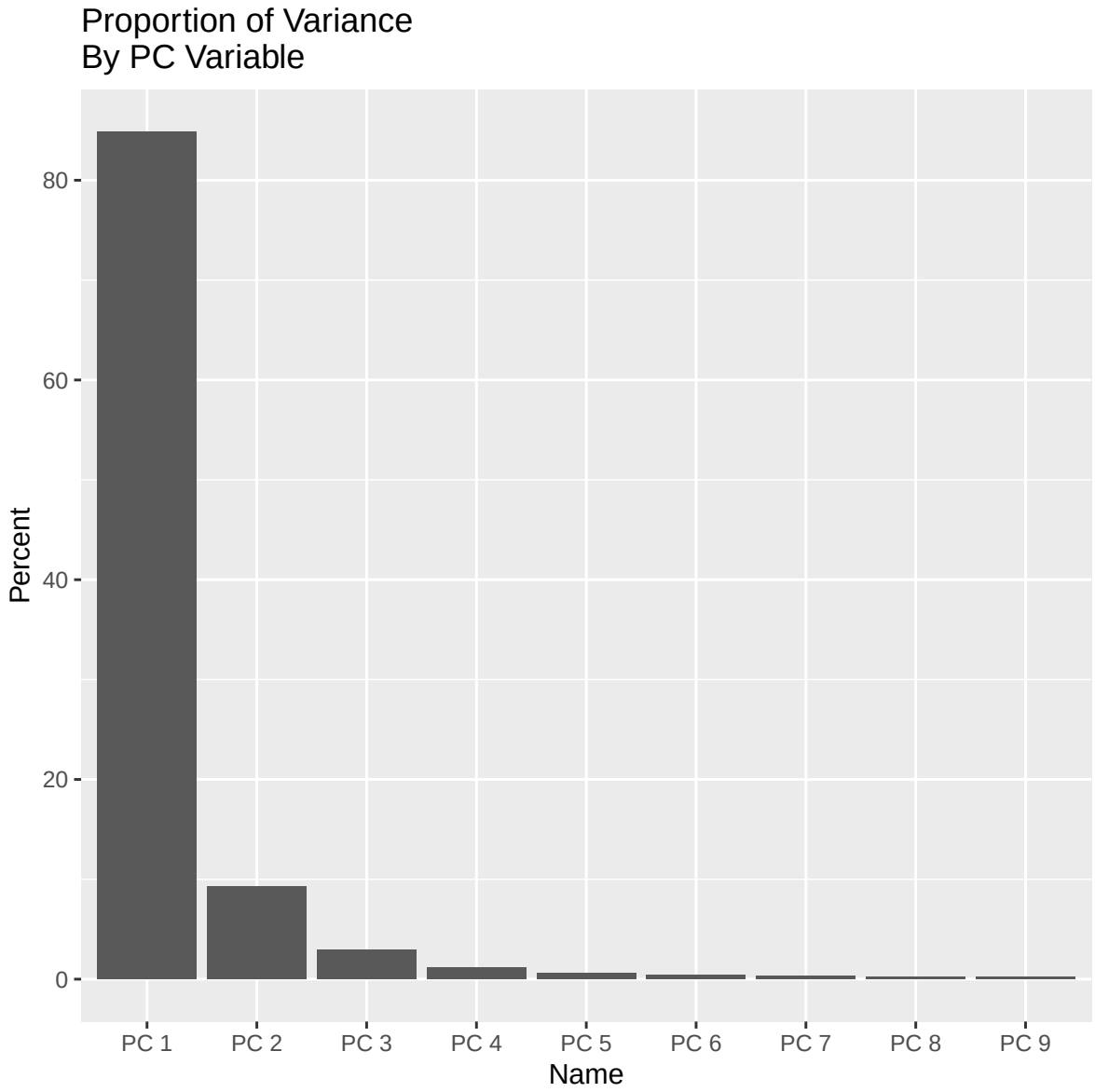
```
[1] 5.6888548 1.8798646 1.0543847 0.6744861 0.4750723 0.3954230 0.3384820
[8] 0.3183946 0.2864309
```

```
> print(X.svd$v)
```

```
           [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
 [1,] 0.1094755 -0.52396980 -0.53930197 -0.54201274 -0.21467693  0.28578205
 [2,] 0.1575183 -0.48180741 -0.25175245  0.25384580  0.26016949 -0.73398351
 [3,] 0.2518511 -0.41018365  0.04840772  0.66906075  0.03427485  0.52532563
 [4,] 0.3900745 -0.20717611  0.47643680 -0.06108921 -0.45192896 -0.05140573
 [5,] 0.4195167 -0.08743782  0.40477867 -0.29360720 -0.06738322 -0.20401940
 [6,] 0.4206809  0.10773798  0.03153752 -0.21324550  0.59211406  0.18452629
 [7,] 0.4015145  0.22814155 -0.14679307 -0.01942643  0.21896797  0.03626624
 [8,] 0.3698262  0.28336162 -0.27240399  0.06144588  0.01411828  0.03376057
 [9,] 0.3109083  0.36131631 -0.39477364  0.23362042 -0.52680665 -0.15118990
             [,7]         [,8]        [,9]
 [1,] -0.004104455  0.007699493 -0.03182216
```

```
 [2,]  0.063313981 -0.045791330  0.05203054
 [3,] -0.169017256  0.091333958 -0.06545449
 [4,]  0.509853514 -0.277972602  0.17347706
 [5,] -0.564952136  0.409818830 -0.18172120
 [6,] -0.170142226 -0.487184807  0.33764695
 [7,]  0.385047468  0.019729453 -0.75292689
 [8,]  0.276436525  0.615915210  0.49784275
 [9,] -0.366959204 -0.356639741 -0.01127971

> # X.svd$u is too big to print
> print(head(X.svd$u))

             [,1]        [,2]         [,3]         [,4]         [,5]
[1,]  0.051798111  0.19032041  0.041610849 -0.013276357  0.087538062
[2,] -0.069963895  0.17285040  0.150905076  0.033185886  0.022261305
[3,] -0.081764407  0.15365595  0.056737694  0.034788518  0.005203164
[4,]  0.002293465 -0.02594221 -0.076818619 -0.006526149 -0.011986163
[5,]  0.033977523 -0.05510211  0.025180409  0.016958119  0.026780182
[6,]  0.052072586 -0.01996219  0.001076536  0.020447249 -0.019173474
             [,6]         [,7]         [,8]         [,9]
[1,]  0.039305122  0.073803107  0.064502486  0.013340849
[2,] -0.039108298 -0.013208688 -0.054564054 -0.001239131
[3,] -0.002288638  0.010287500 -0.014774323  0.051776365
[4,] -0.026803704  0.009477146  0.001742798 -0.022194108
[5,] -0.003756958 -0.024913360  0.005835135  0.041226627
[6,]  0.046777981  0.014399696  0.095454705  0.002826924

> print(tail(X.svd$u))

                [,1]         [,2]         [,3]         [,4]         [,5]
[1242,] -0.020663452 -0.008123324  0.013124722 -0.028058884  0.008599237
[1243,] -0.021295046 -0.025113870  0.008522503 -0.013537461 -0.010020902
[1244,] -0.010027503 -0.025439817  0.008499638  0.039731750  0.065197283
[1245,]  0.010648686  0.030014156  0.010268938  0.001860706 -0.026396161
[1246,]  0.002549732 -0.028374810 -0.014946244 -0.051032594  0.001913039
[1247,]  0.012031067 -0.030032839 -0.032510237 -0.028480382 -0.021347515
              [,6]         [,7]         [,8]         [,9]
[1242,]  0.04070209  0.034899107 -0.014073460  0.014387733
[1243,] -0.01073502  0.009215867  0.008232294 -0.004573769
[1244,] -0.01397549  0.024357529 -0.012189584 -0.029956509
[1245,] -0.01596199 -0.005129421  0.024056328  0.007421237
[1246,]  0.04284073  0.002835272 -0.005429623 -0.012536570
[1247,]  0.02934648  0.019605102  0.003451145  0.001360156

>
> # NOTE: The X matrix in this note is the transpose
> #       of the X matrix in the reading.
```

```
> #       Statisticians like to have rows correspond
> #       to cases (days) and columns to variables


> df0<-data.frame(
+   Name= paste("PC ",c(1:NCOL(X)),sep=""),
+   Percent=round(as.matrix(X.svd$d^2 / sum(X.svd$d^2)*100), digits=2))
> ggplot(df0, aes(x=Name, y=Percent)) + geom_bar(stat="identity") +
+   ggtitle("Proportion of Variance\nBy PC Variable")
```



The Total Variance is also the sum of eigen values of $S = \frac{X^T X}{n-1}$. These eigen values (λ_k) are related to the singular values of X (d_i) by $\lambda_k = d_k^2/(n-1)$, where n is the number of columns (days) in A. These eigen values are the variances of the PC Variables.

```
> X.colVariances=apply(X,2,var)
> X.colVariances
```

```
     DGS3MO      DGS6MO        DGS1        DGS2        DGS3        DGS5
0.001475386 0.001463504 0.002328630 0.004347215 0.004821780 0.004744274
       DGS7       DGS10       DGS20
0.004414126 0.003902147 0.003115827
```

```
> sum(X.colVariances)
```

```
[1] 0.03061289
```

```
> sum((X.svd$d)^2)/(NROW(X)-1)
```

```
[1] 0.03061289
```

The Principal Components Variables are an orthogonal transformation of the original coordinate axes (columns) of X that corresponds to a simple 'rotation' of the data about the origin (data mean). Principal Component Variable PC 1 is the axis of the data with the greatest variability (highest standard deviation), and Principal Component Variable PC 2 is the axis of the data orthogonal to the first with next greatest variability.

The importance of principal components variables can be measured by the proportion of total variability of each component k and the cumulative proportion of the first k components

Loadings of Principal Components Variables:

The j-th column of matrix v in the Singular-Value Decomposition (SVD) of X contains the "loadings" of the j-th Principal Component Variable. Each PC variable is a weighted sum of yield changes in different rows (tenors) of X and the loadings are the weights applied for each PC variable.

The next table presents the loadings for the first 3 PC variables and the stacked plots display and compare their loadings on different tenors. Note that

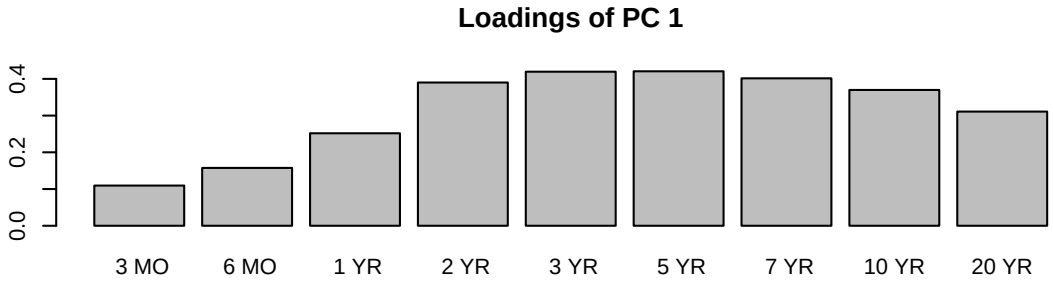
- PC 1 measures level shifts in the yield curve – its daily values are a weighted sum of all 9 yield changes.
- PC 2 measures the spread in the yield curve between short and long maturities – its daily values are a difference between a weighted sum of the shorter-tenor yield changes and a weighted sum of the longer-tenor yield changes.
- PC 3 measures the curvature in the yield curve – it corresponds to a second difference of yield changes: the difference between two first differences (mid-tenor minus short-tenor, and long-tenor minus mid-tenor).

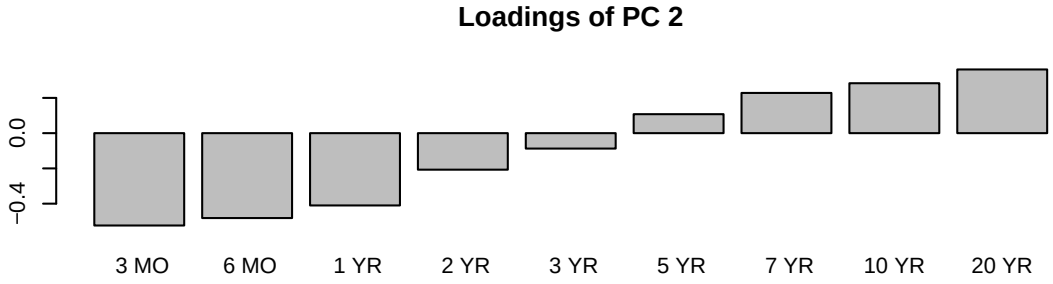
```
> X.Loadings<-X.svd$v
> dimnames(X.Loadings)<-list(legend.fred, paste("PC",c(1:NCOL(X))))
> print(round(X.Loadings[,1:3], digits=2))

      PC 1  PC 2  PC 3
 3 MO 0.11 -0.52 -0.54
 6 MO 0.16 -0.48 -0.25
 1 YR 0.25 -0.41  0.05
 2 YR 0.39 -0.21  0.48
 3 YR 0.42 -0.09  0.40
 5 YR 0.42  0.11  0.03
 7 YR 0.40  0.23 -0.15
10 YR 0.37  0.28 -0.27
20 YR 0.31  0.36 -0.39

>
```

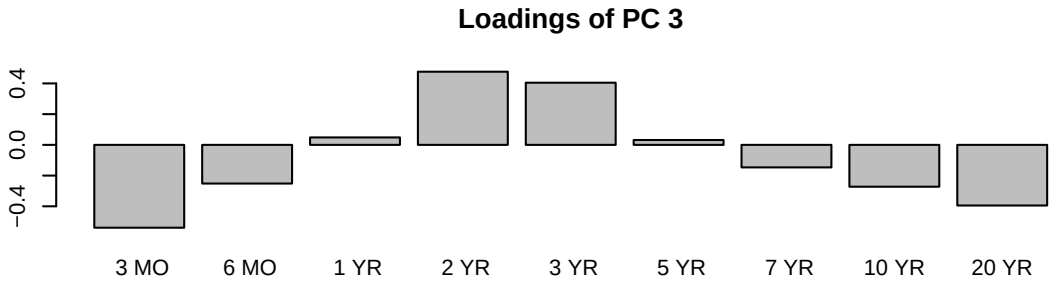
Graphical display of loadings for the top 3 PC variables.

```
> #
> ylimabs0<-max(abs(X.Loadings[,1:3]))
> ylim0<-c(-1,1)*ylimabs0
```

```
> par(mfcol=c(3,1))
> barplot(X.Loadings[,1], main="Loadings of PC 1")
> barplot(X.Loadings[,2], main="Loadings of PC 2")
> barplot(X.Loadings[,3], main="Loadings of PC 3")
> #par(mfcol=c(3,1))
> #barplot(A.Loadings[,1], main="",ylim=ylim0,axes=FALSE)
> #abline(h=0,col=1)
> #barplot(A.Loadings[,2], main="",ylim=ylim0,axes=FALSE)
> #abline(h=0,col=1)
> #barplot(A.Loadings[,3], main="",ylim=ylim0,axes=FALSE)
> #abline(h=0,col=1)
```


Loadings of PC 1
0.0
0.2
0.4
3 MO
6 MO
1 YR
2 YR
3 YR
5 YR
7 YR
10 YR
20 YR

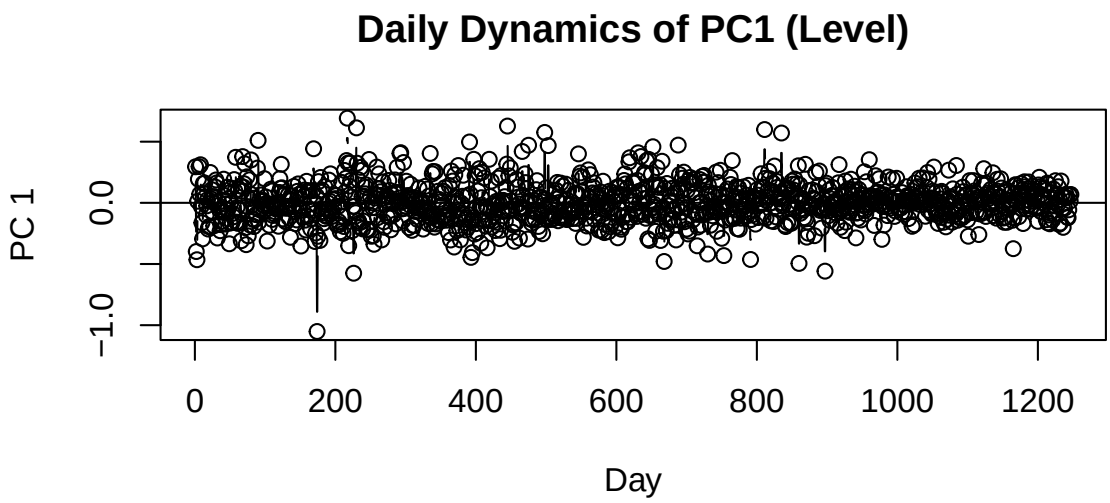

Loadings of PC 2
−0.4
0.0
3 MO
6 MO
1 YR
2 YR
3 YR
5 YR
7 YR
10 YR
20 YR

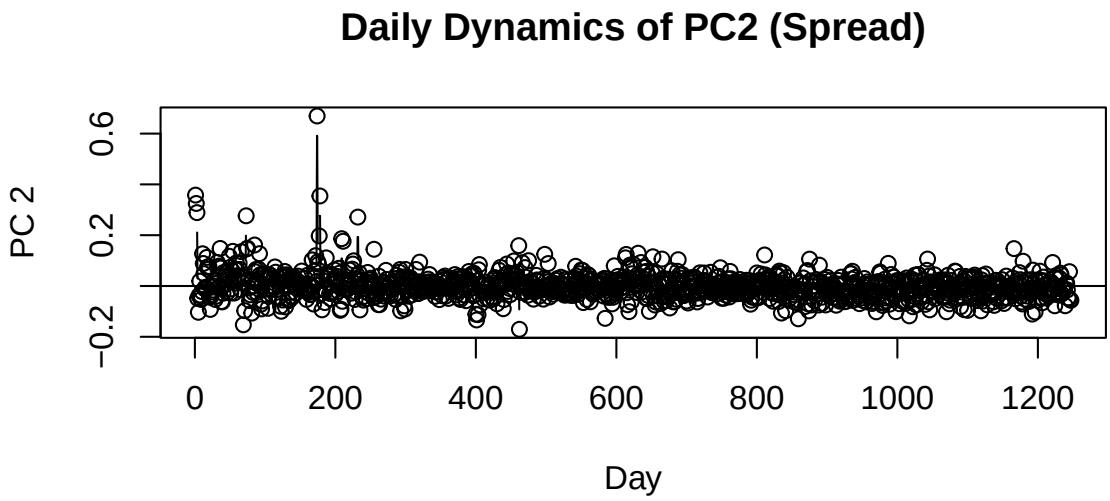

Loadings of PC 3
−0.4
0.0
0.4
3 MO
6 MO
1 YR
2 YR
3 YR
5 YR
7 YR
10 YR
20 YR

Principal Components Analysis allows us to approximate daily changes in the Treasury yield curve with changes in the first and second Principal Components Variables. The first plot shows how the level of the curve (PC1) shifted daily over the period. The second plot shows how the spread of the curve (shorter-tenor yields minus longer-tenor yields) changed on a daily basis.

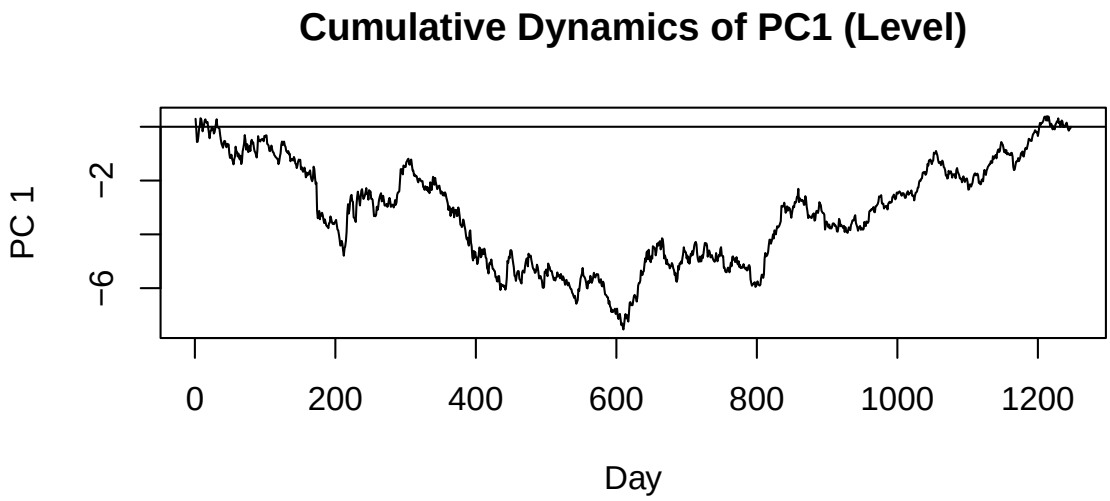
These are followed by plots of the cumulative daily changes of the PC variables over the 5-year time period. The overall level of rates (PC1) declined over the first half of the period and then increased. The spread of rates (PC2) increased at the first part of the period and then decreased over the last third of the period.

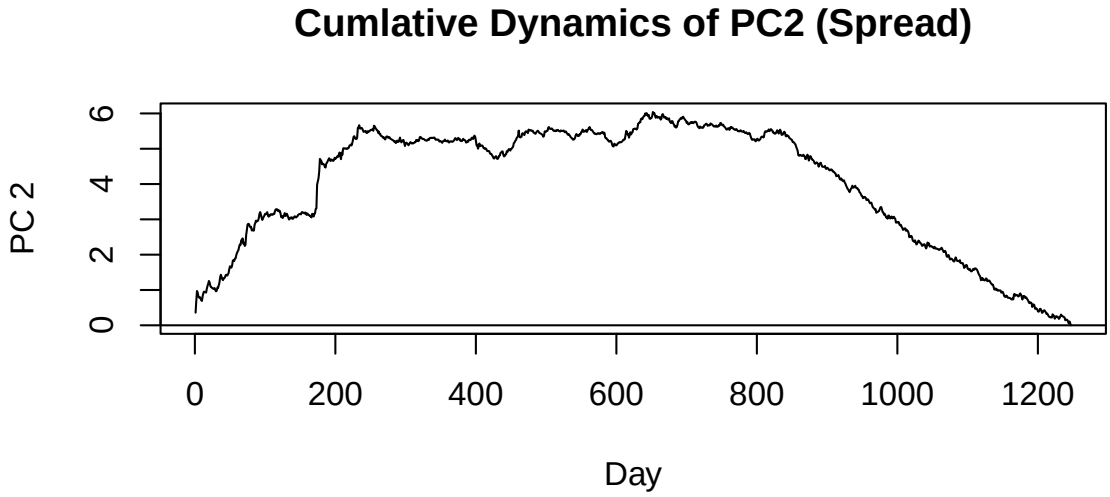
```
> # Compute PC variables individually
> PC1=X.svd$d[1]*X.svd$u[,1]
> PC2=X.svd$d[2]*X.svd$u[,2]
> PC3=X.svd$d[3]*X.svd$u[,3]
> # Note: the PC variables could be computed
> # all at once.
> #   The matrix PCmatrix has  columns equal to PC variables
> PCmatrix=X.svd$u %*%diag(X.svd$d)
> par(mfcol=c(2,1))
> plot(c(1:NROW(X)),PC1,type="b",xlab="Day", ylab="PC 1",
+      main="Daily Dynamics of PC1 (Level)")
> abline(h=0)
> plot(c(1:NROW(X)),PC2,type="b",xlab="Day", ylab="PC 2",
+ main="Daily Dynamics of PC2 (Spread)")
> abline(h=0)
>
```






```
> par(mfcol=c(2,1))
> plot(c(1:NROW(X)),cumsum(PC1),type="l",xlab="Day", ylab="PC 1",
+      main="Cumulative Dynamics of PC1 (Level)")
> abline(h=0)
> plot(c(1:NROW(X)),cumsum(PC2),type="l",xlab="Day", ylab="PC 2",
+ main="Cumlative Dynamics of PC2 (Spread)")
> abline(h=0)
>
```



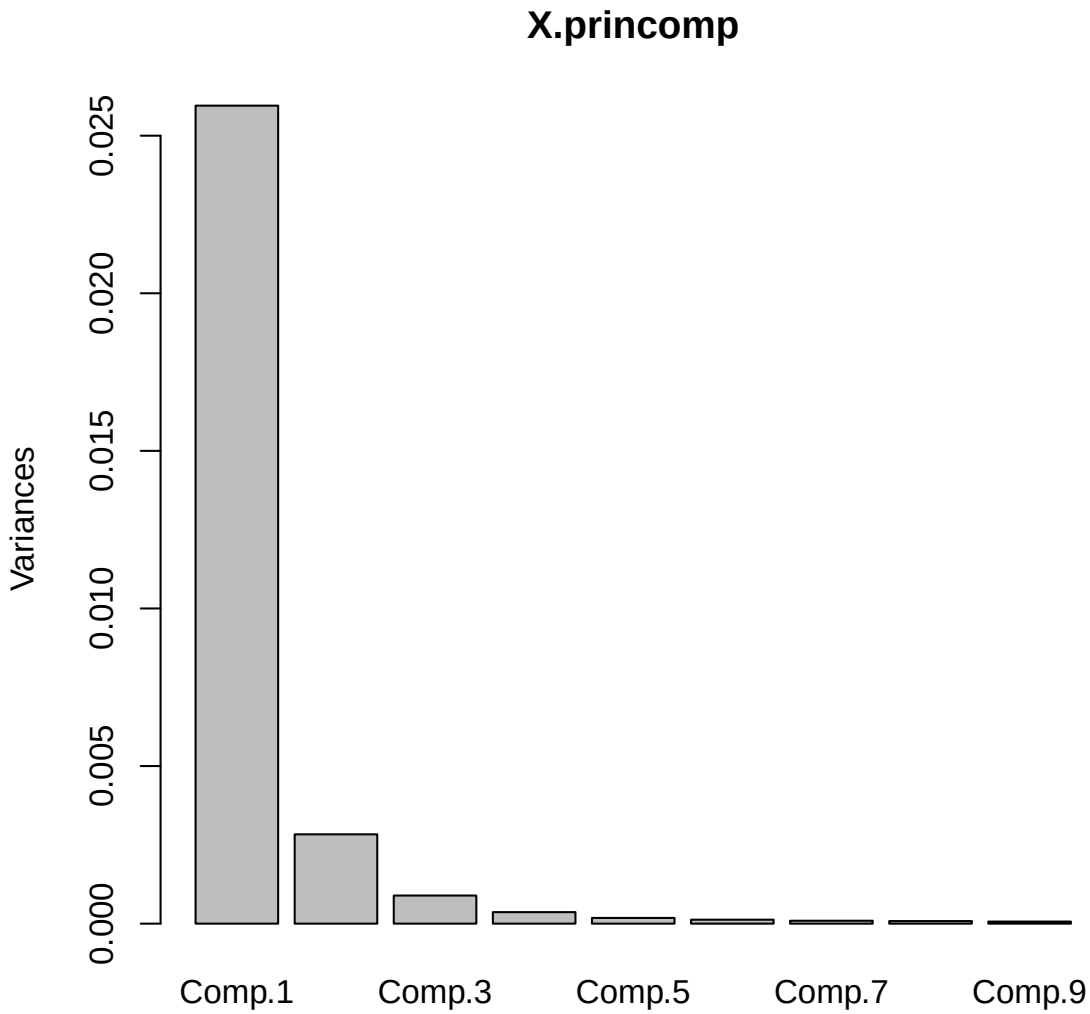


3 R's Built-in Functions for PCA

The following R code illustrates the use of the built-in R functions *princomp*() and *screeplot*(). Compare these results with those computed directly above.

```
> par(mfcol=c(1,1))
> # Compare these results to the output of R's built-in
```

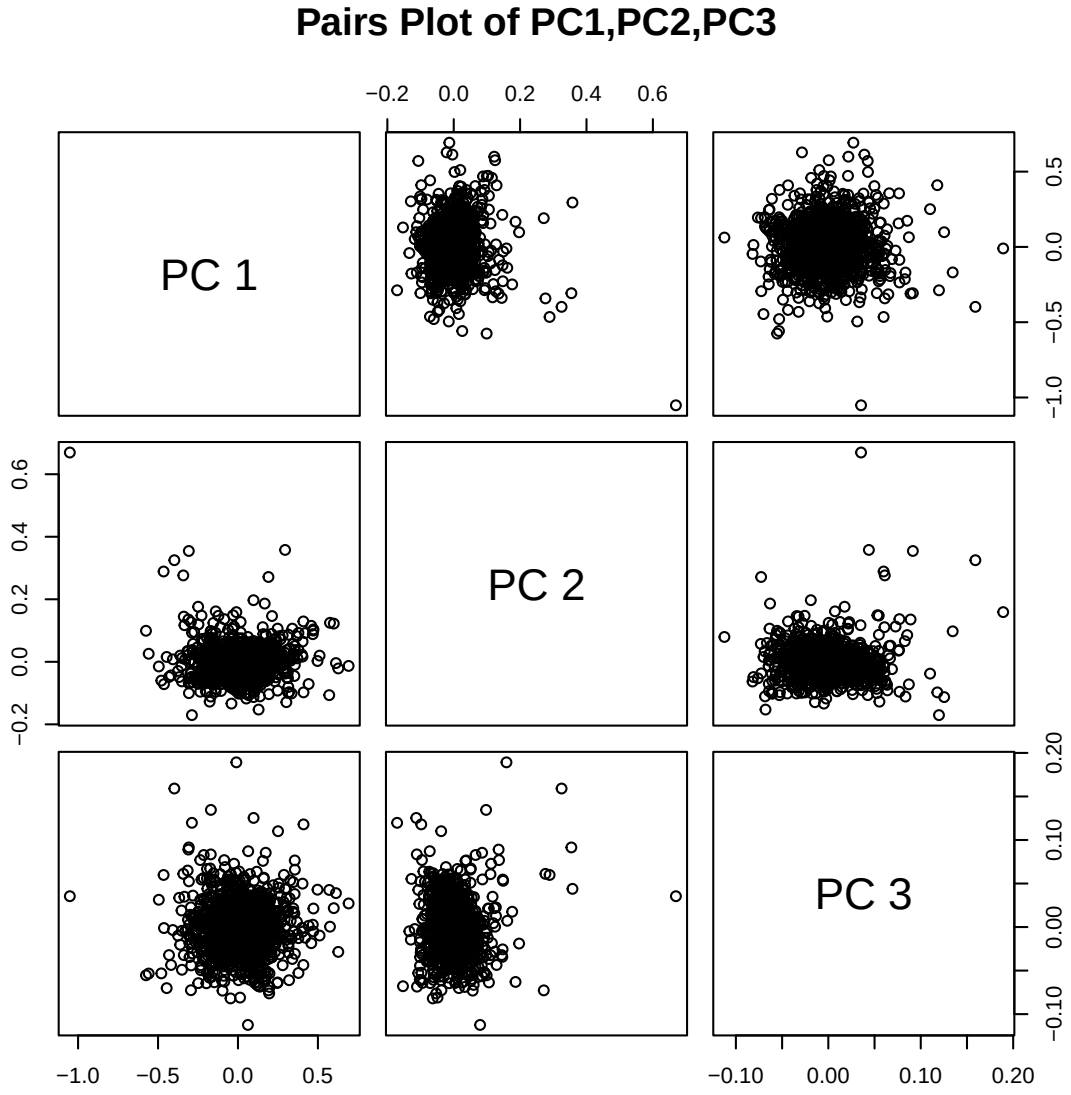
```
> #   function princomp(), and related function screeplot()
> X.princomp=princomp(X)
> print(names(X.princomp)) #output from princomp() is a list

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

> print(summary(X.princomp))

Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4      Comp.5
Standard deviation     0.1610985 0.05323452 0.02985836 0.01910028 0.013453227
Proportion of Variance 0.8484521 0.09264687 0.02914579 0.01192679 0.005916939
Cumulative Proportion  0.8484521 0.94109900 0.97024480 0.98217159 0.988088525
                            Comp.6      Comp.7      Comp.8      Comp.9
Standard deviation     0.011197698 0.009585226 0.009016386 0.008111228
Proportion of Variance 0.004099223 0.003003646 0.002657719 0.002150886
Cumulative Proportion  0.992187748 0.995191395 0.997849114 1.000000000

> print(X.princomp$loadings)

Loadings:
       Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
DGS3MO  0.109  0.524  0.539  0.542  0.215  0.286
DGS6MO  0.158  0.482  0.252 -0.254 -0.260 -0.734
DGS1    0.252  0.410        -0.669         0.525  0.169
DGS2    0.390  0.207 -0.476         0.452        -0.510 -0.278 -0.173
DGS3    0.420        -0.405  0.294        -0.204  0.565  0.410  0.182
DGS5    0.421 -0.108         0.213 -0.592  0.185  0.170 -0.487 -0.338
DGS7    0.402 -0.228  0.147        -0.219        -0.385         0.753
DGS10   0.370 -0.283  0.272                      -0.276  0.616 -0.498
DGS20   0.311 -0.361  0.395 -0.234  0.527 -0.151  0.367 -0.357

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111
Cumulative Var  0.111  0.222  0.333  0.444  0.556  0.667  0.778  0.889  1.000

> # The screeplot is the same as the barplot constructed above.
>
> screeplot(X.princomp)
```



4 Interactive 3-D Displays with Yield PCA Data

Now, we load R libraries for 3-d plotting. Note: the packages need to be installed; these lines have been commented out, but need to be run once before compiling the document.

```
> # install.packages("scales")
> library(scales)
> # install.packages("plot3D")  #Note the capital D
> library(plot3D)
> # install.packages("plot3Drgl")  #Note the capital D
> library("plot3Drgl")
> # install.packages("scatterplot3d") #Note small d
> library("scatterplot3d")
> library(rgl)

> dimnames(PCmatrix)[[2]]<-paste("PC",1:NCOL(PCmatrix))
> pairs(PCmatrix[,1:3], main="Pairs Plot of PC1,PC2,PC3")
> # Since the commands are interactive they are not in the document
> #  These command can be executed from inside the Sweave file
> #   when editing this document
> if (FALSE){
+
+ plot3d(PC1,PC2,PC3) #creates RGL device outside RStudio
+ plot3d(PC1,PC2,PC3,
+        col='blue',size=5,
+        aspect=c(1,1,1))
+
+ aspect0=X.svd$d[1:3]
+ aspect0=aspect0/sum(aspect0)
+
+ plot3d(PC1,PC2,PC3,
+        col='blue',size=5,
+        aspect=aspect0)
+
+ planes3d(a=c(0,0,1),alpha=.5)
+
+ }
>
```


Pairs Plot of PC1,PC2,PC3
PC 1
PC 2
PC 3
−0.2
0.0
0.2
0.4
0.6
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
−0.10
0.00
0.10
0.20

18.642 Topics in Mathematics with Applications in Finance

Fall 2024