APM466 Assignment 1

Xinqiao Li 1004761536

2/1/2022

```
library(tidyverse)
library(knitr)
library(lubridate)
library(rmarkdown)
library(readr)
library(dbplyr)
```

```
# Import data
# The 11th bond is for interpolation purposes.
data<-read.csv("C:/Users/Xinqiao/Desktop/Selected Bonds.csv")
```

4 (a). Calculate each of the selected bonds' yield (ytm).

- Step 1: Calculate the dirty price for each of the bonds selected.

```
# Use a date to store the # of days elapsed since 1/1/2022 to the record date.
date<-c(10,11,12,13,14,17,18,19,20,21)
t=1
while (t <= 10) {
  # Create an empty column for the dirty price of each bond
  calc_DP = paste("DP", toString(t), sep="")
  # Calculating the accrued interest, assuming there are 30.5 days in each month
  # (as an average of 30 and 31)
  Accrued<-(122 + (date[t]))/365 * data$Coupon*100
  # we get the dirty price by adding the accrued interest to the clean data
  Dirty_Price = Accrued + data[[paste("CP", toString(t), sep="")]]
  # Create a new column to the data frame to store the value of dirty price
  data[[calc_DP]] = Dirty_Price
  t = t + 1
}
```

```
# We have to modify ZU15(CAN 2.75 Jun 1, 2022) and A610(CAN 1.50 Jun 1, 2023)
# They are the 2nd and 4th bonds
new <- c(2,4)
t = 1
while (t<=10) {
  for (i in new) {
    Accrued<-(31+date[t])/365*data$Coupon[i]*100
    Dirty_Price= Accrued + data[[paste("CP", toString(t), sep="")]][i]
    # Replace dirty price with modified dirty price
    data[[paste("DP", toString(t), sep="")]][i] = Dirty_Price
  }
  t = t + 1
```

```
}
```

- Step 2: Using Newton-Rhapson Method to calculate the ytm.

```
# Create a vector to store the date when we collected the closed price.
observation_date<-c("2022-01-10","2022-01-11","2022-01-12","2022-01-13",
                    "2022-01-14","2022-01-17","2022-01-18","2022-01-19",
                    "2022-01-20","2022-01-21")
# Using function as.Date to convert the date for calculation convenience
dates<-as.Date(observation_date)
# Change the format of  Maturity.Date to match with that of observation_date
maturity <- as.Date(data$Maturity.Date, "%m/%d/%Y")
# calculate the number of payments remaining
n = (data$Month.To.Maturity-2)/6 + 1

# Calculate the total discounted cash flows minus the dirty price of the bond
sum_function<-function(i,j,r){
  term=as.numeric(maturity[i]-dates[j])/365
  # Discounted principal payment minus the dirty price
  sum=100*(1+r/2)**(-2*term)-data[[paste("DP", toString(j), sep="")]][i]
  # Time until the next latest coupon payment
  payment_time=as.numeric(0.5-data$Months.Since.Last.Coupon[i]/12-date[j]/365)
  m=1
  while (m<=n[i]) {
    sum=sum+(data$Coupon[i]*100/2)*(1+r)**(-2*payment_time)
    # Assume coupons are paid semi-annually
    payment_time=payment_time+0.5
    m = m + 1
  }
  sum # output the result
}

# Calculate the derivative of the above equation
derivative_function<-function(i,j,r){
  term=as.numeric(maturity[i]-dates[j])/365
  derivative=-100*2*term*(1+r/2)**(-2*term-1)
  payment_time=as.numeric(0.5-data$Months.Since.Last.Coupon[i]/12-date[j]/365)
  m=1
  while (m<=n[i]) {
    derivative=derivative-2*payment_time*(data$Coupon[i]*100/2)*
      (1+r/2)**(-2*payment_time-1)
    payment_time=payment_time+0.5
    m = m + 1
  }
  derivative # output the result
}

# Create 10 empty columns to store the ytm
empty_cols<-c('YTM1','YTM2','YTM3',"YTM4","YTM5",
              "YTM6","YTM7","YTM8","YTM9","YTM10")
data[ ,empty_cols]<-NA
for (j in c(1:10)){
  for(i in c(1:11)){
    require(numDeriv)
```

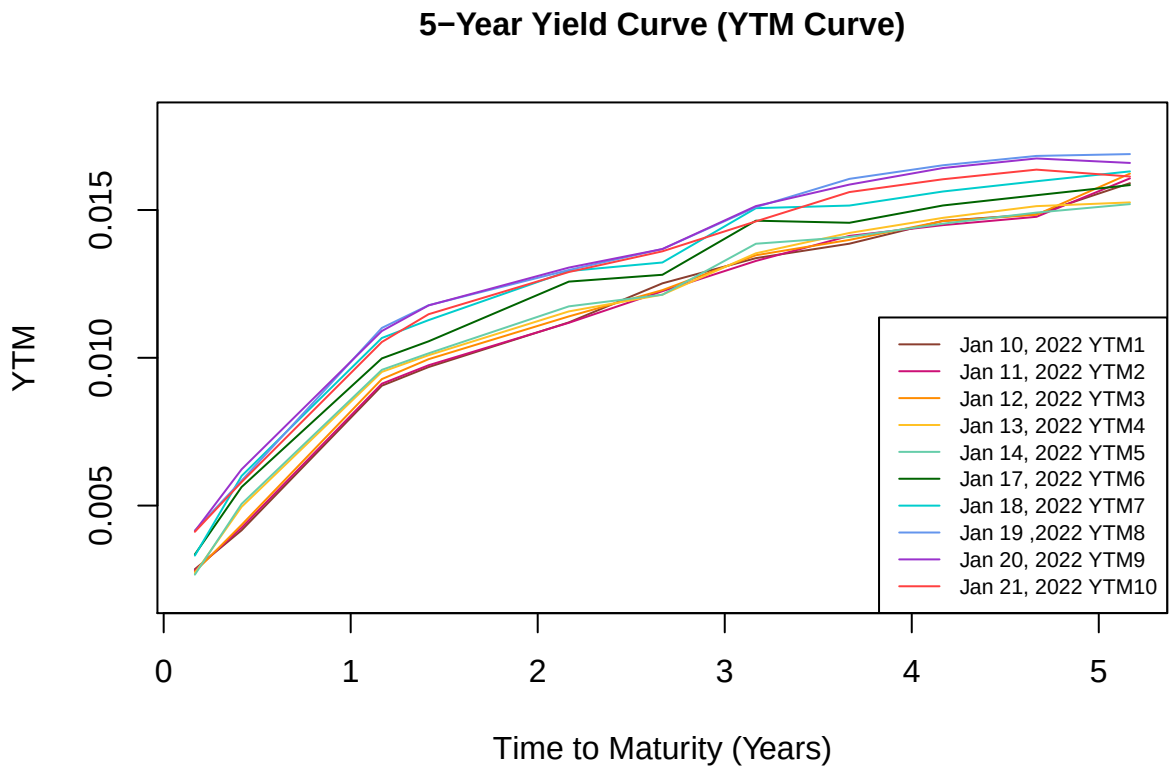
```
    k=c()
    x0=0
    for (t in 1:100) {
      dx<-derivative_function(i,j,x0)
      fx<-sum_function(i,j,x0)
      x1 <- x0 - (fx/dx)
      k[i] <- x1
      # When the difference between x0 and x1 becomes sufficiently small, output.
      if (abs(x1 - x0) < 0.000001) {
        root.approx <- tail(k, n=1)
        data[[paste("YTM",toString(j),sep="")]][i]=root.approx
        }
      x0 <- x1
      }
  }
}
```

```
## Loading required package: numDeriv
```

```
## Warning: package 'numDeriv' was built under R version 4.0.3
```

- Step 3: Plot the5-year yield curve.

```
col<-c("coral4","deeppink3","darkorange","goldenrod1","aquamarine3","darkgreen",
       "cyan3","cornflowerblue","darkorchid3","brown1")
plot(data$Month.To.Maturity/12,data$YTM1,type="l",col=col[1],
     main="5-Year Yield Curve (YTM Curve)",xlab="Time to Maturity (Years)",
     ylab="YTM",ylim=c(0.002,0.018),cex.main=1)
for (j in c(2:10)){
  lines(data$Month.To.Maturity/12,
        data[[paste("YTM", toString(j), sep="")]],type="l",col=col[j])
}
legend("bottomright",legend=c("Jan 10, 2022 YTM1","Jan 11, 2022 YTM2",
       "Jan 12, 2022 YTM3","Jan 13, 2022 YTM4","Jan 14, 2022 YTM5",
       "Jan 17, 2022 YTM6","Jan 18, 2022 YTM7", "Jan 19 ,2022 YTM8",
       "Jan 20, 2022 YTM9","Jan 21, 2022 YTM10"),lty=1.4,cex=0.7,col=col)
```



4 (b). Derive the spot curve with terms ranging from 1-5 years.

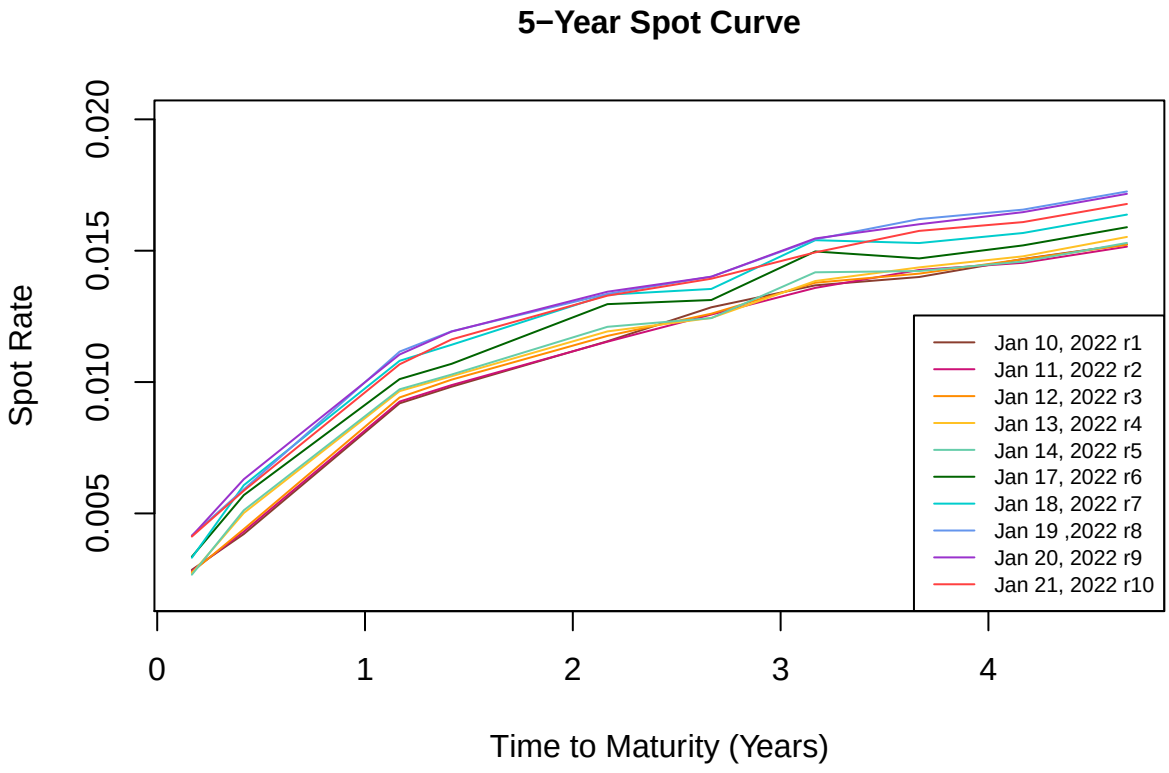
- Step 1: Calculate the spot rate of each bond i at time t.

```
t = 1
while(t <= 10){
  col_r = paste("r",toString(t),sep="")
  # calculate r_1 for bond1 with maturity 3/1/2022
  r_1 = -log(data[[paste("DP",toString(t),sep="")]][1] /(0.5*100*data$Coupon[1]+
  r = c(r_1)

  # calculate r_2 for bond2 with maturity 6/1/2022
  r_2 = -log(data[[paste("DP",toString(t),sep="")]][2] /(0.5*100*data$Coupon[2]+
  # Using linear extrapolation to find r_2 with maturity 9/1/2022, 2*r2-r1
  r_2_hat<-2*r_2-r_1
  r = append(r, 2*r_2-r_1)

  # calculate r_3 for bond3 with maturity 3/1/2023
  pmt_dis= 100*0.5*data$Coupon[3]*exp(-r_1*((60-date[t])/365)) +
    100*0.5*data$Coupon[3]*exp(-r_2_hat*((60-date[t])/365+1/2))
  r_3 = -log((data[[paste("DP", toString(t), sep="")]][3]-
                pmt_dis)/(100*0.5*data$Coupon[3]+100))/(1+(60-date[t])/365)
  r = append(r, r_3)

  # calculate r_4 for bond 4 with maturity 6/1/2023 via interpolation
  mid_r = c()
```

```
  w = 1
  while (w < 3) {
    mid_r = append(mid_r, (r[w]+r[w+1])/2 )
    w = w + 1
  }

  pmt = 0.5*100*data$Coupon[4]*exp(-mid_r[1]*((152-date[t])/365)) +
    0.5*100*data$Coupon[4]*exp(-mid_r[2]*(0.5+(152-date[t])/365))
  r_4 = -log((data[[paste("DP", toString(t), sep="")]][4]-
                pmt)/(100*0.5*data$Coupon[4]+100))/(1+(152-date[t])/365)
  mid_r<-append(mid_r,r_4)
  # Use extrapolation to get the rate with maturity 9/1/2023
  r <- append(r, 2*r_4-r_3)

  # Calculate the  r_i for bond 5 to 11 at each day t.
  i = 5
  while(i <= 11){
    pmt = 0.5*100*data$Coupon[i]*exp(-r_1*((60-date[t])/365))
    w = 2
    while(w<=i-1){
      pmt = pmt + 0.5*100*data$Coupon[i]*exp(-((w-1)/2+((60-date[t])/365))*r[w])
      w = w + 1
    }
    r_i = - log((data[[paste("DP", toString(t), sep="")]][i]-
                  pmt)/(0.5*100*data$Coupon[i]+100))/((i-1)/2+
                                                        ((60-date[t])/365))
    r = append(r, r_i)
    i = i + 1
  }

  spot_col = paste("r", toString(t), sep="")
  spot_r = append(r_1, append(r_2, append(r_3, append(r_4, r[5:11]))))
  data[[spot_col]] = spot_r
  t = t + 1
}
```

- Step 2: Plot the spot curve with terms ranging from 1-5 yrs from chosen bonds.

```
data_10<-data[c(1:10), ]
col<-c("coral4","deeppink3","darkorange","goldenrod1","aquamarine3","darkgreen",
       "cyan3","cornflowerblue","darkorchid3","brown1")

plot(data_10$Month.To.Maturity/12,data_10$r1,type="l",col=col[1],
     main="5-Year Spot Curve",xlab="Time to Maturity (Years)",
     ylim=c(0.002,0.020),ylab="Spot Rate",cex.main=1)
for (j in c(2:10)){
  lines(data_10$Month.To.Maturity/12,
        data_10[[paste("r", toString(j), sep="")]],type="l",col=col[j])
}
# Adding additional legend
legend("bottomright",legend=c("Jan 10, 2022 r1","Jan 11, 2022 r2",
       "Jan 12, 2022 r3","Jan 13, 2022 r4","Jan 14, 2022 r5",
       "Jan 17, 2022 r6","Jan 18, 2022 r7", "Jan 19 ,2022 r8",
       "Jan 20, 2022 r9","Jan 21, 2022 r10"),lty=1.4,cex=0.7,col=col)
```



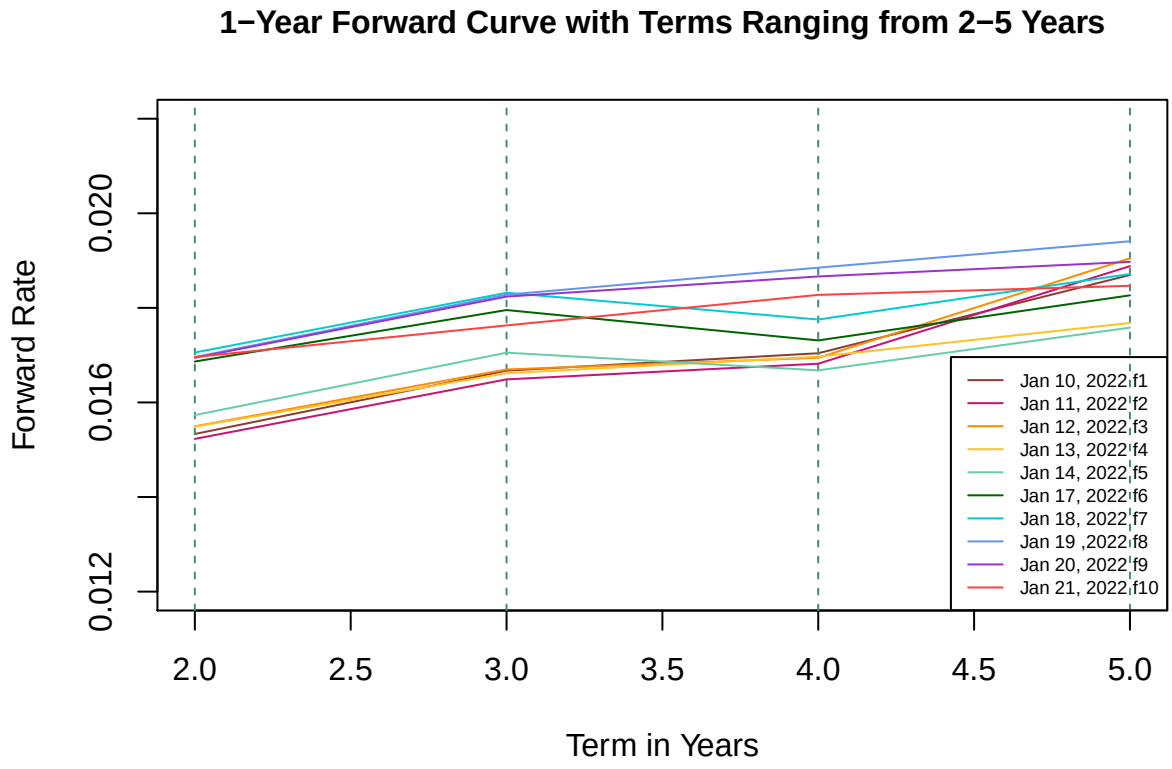
4(c) Derive the 1-year forward curve with terms ranging from 2-5 years.

- Step 1: Calculate the forward rates.

```
date<-c(10,11,12,13,14,17,18,19,20,21)
t = 1
term = data$Month.To.Maturity/12
while (t <= 10) {
  forward = c()
  # Estimate the annual spot rate for year 1.
  r_year1 = (data[[paste("r", toString(t), sep="")]][2]+
               data[[paste("r", toString(t), sep="")]][4])/2
  i = 1
  while (i <= 11) {
    f_01i=(data[[paste("r",toString(t),sep="")]][i]*(term[i]-date[t]/365)-
             r_year1*(1-date[t]/365))/(term[i]-1)
    forward = append(forward, f_01i)
    i = i + 1
  }
  data[[paste("f", toString(t), sep="")]] = forward
  t= t + 1
}
# We have just calculated the forward rate for all T2 over the 0-5 year span,
# but we only want T2=2,3,4,5.
# So we select bonds mature in march 2024,2025,2026,2027
annual_forward<-data[c(5,7,9,11),]
```

Step 2: Plot the 1-year forward curve with terms 2-5 years for selected bonds.

```
col<-c("coral4","deeppink3","darkorange","goldenrod1","aquamarine3","darkgreen",
       "cyan3","cornflowerblue","darkorchid3","brown1")
t<-c(2,3,4,5)
plot(t,annual_forward$f1,type="l",col=col[1],
     main="1-Year Forward Curve with Terms Ranging from 2-5 Years",
     xlab="Term in Years",ylim=c(0.012,0.022),xlim=c(2,5),
     ylab="Forward Rate",cex.main=1)
abline(v=2, col="aquamarine4",lty=2)
abline(v=3, col="aquamarine4",lty=2)
abline(v=4, col="aquamarine4",lty=2)
abline(v=5, col="aquamarine4",lty=2)
for (j in c(2:10)){
  lines(t,annual_forward[[paste("f", toString(j), sep="")]],type="l",col=col[j])
}
legend("bottomright",legend=c("Jan 10, 2022 f1","Jan 11, 2022 f2",
       "Jan 12, 2022 f3","Jan 13, 2022 f4","Jan 14, 2022 f5",
       "Jan 17, 2022 f6","Jan 18, 2022 f7", "Jan 19 ,2022 f8",
       "Jan 20, 2022 f9","Jan 21, 2022 f10"),lty=1.5,cex=0.6,col=col)
```



5. Calculate two covariance matrices for the time series of daily log-returns of yield, and forward rates.

- Part 1: Calculate the covariance matrix of daily log-returns of yield.

```
X_log_ytm<-c()
# Find the time series X for yield.
i=1
while (i<=10){
  X_log_ytm_i<-log(as.numeric(data[i,30:38]/data[i,29:37]))
  X_log_ytm = append(X_log_ytm, X_log_ytm_i)
  i = i + 2
}
X_log_ytm = matrix(X_log_ytm, ncol=5, byrow=FALSE)
# Find the covariance matrix
covariance_ytm = cov(X_log_ytm)
```

- Table 1: Covariance Matrix of Daily Log-Returns of Yield

	X1	X2	X3	X4	X5
X1	0.011326872	0.0010629733	0.0011841299	0.0010780640	0.0022814963
X2	0.001062973	0.0008579410	0.0004315453	0.0005414344	0.0006173138
X3	0.001184130	0.0004315453	0.0005342130	0.0005366878	0.0003448639
X4	0.001078064	0.0005414344	0.0005366878	0.0006242361	0.0003866526
X5	0.002281496	0.0006173138	0.0003448639	0.0003866526	0.0007234129

- Part 2: Calculate the covariance matrix of daily log-returns of 1-yr forward rate.

```
t = 1
f_year = c()
while (t <= 10) {
  for_i = data[[paste("f", toString(t), sep = "")]]
  # Interpolate the forward rate
  inter_f = c(for_i[2],for_i[3]*(2/3)+for_i[2]*(1/3),for_i[4],for_i[5]*(2/3)+
                for_i[4]*(1/3),(for_i[5:10]+for_i[6:11])/2)

  f_year = append(f_year, inter_f[4:10])
  t = t + 1
}
f_year = matrix(f_year,ncol=10,byrow=FALSE)

i = 1
X_f = c()
while (i <= 10-4+1) {
  X_log_for = log(as.numeric(f_year[i, 2:10]/f_year[i, 1:9]))
  X_f = append(X_f, X_log_for)
  i = i + 2
}
X_for = matrix(X_f, ncol=4, byrow=FALSE)
covariance_forward = cov(X_for)
```

- Table 2: Covariance Matrix of Daily Log-Returns of 1-Year Forward Rate

	X1	X2	X3	X4
X1	0.0004046832	0.0004360880	0.0003337612	0.0004485543
X2	0.0004360880	0.0005228843	0.0003493111	0.0004738488
X3	0.0003337612	0.0003493111	0.0007197948	0.0006319142
X4	0.0004485543	0.0004738488	0.0006319142	0.0007325842

6. Calculate the eigenvalues and eigenvectors of both covariance matrices.

- Part 1: Calculate the eigen of yield rate

```
eigenspace_ytm = eigen(covariance_ytm)
# Eigenvalues of yield rate
# The eigenvalues are presented in descending order
eigenspace_ytm$values
```

```
## [1] 1.219243e-02 1.435480e-03 3.846263e-04 3.292288e-05 2.121836e-05
```

```
# Eigenvectors of yield rate
# Note that each column corresponds to an eigenvector
eigenspace_ytm$vectors
```

```
##           [,1]       [,2]        [,3]        [,4]        [,5]
## [1,] 0.9602722  0.2401391 -0.04440926  0.05264139  0.12436723
## [2,] 0.1105478 -0.6511283  0.49310976  0.03437068  0.56521909
## [3,] 0.1125727 -0.4125453 -0.55543191 -0.71252424  0.03063289
## [4,] 0.1067035 -0.5206825 -0.45692235  0.66323132 -0.26239338
## [5,] 0.2039565 -0.2775993  0.48742993 -0.22018393 -0.77153841
```

```
# Percentage of variation explained by the eigenvector with largest eigenvalue.
eigenspace_ytm$values[1]/sum(eigenspace_ytm$values)
```

```
## [1] 0.8667597
```

- Part 2: Calculate the eigen of forward rates

```
eigenspace_forward = eigen(covariance_forward)
# Eigenvalues of forward rate
# The eigenvalues are presented in descending order
eigenspace_forward$values
```

```
## [1] 1.968373e-03 3.330345e-04 5.859162e-05 1.994755e-05
```

```
# Eigenvectors of forward rate
# Note that each column corresponds to an eigenvector
eigenspace_forward$vectors
```

```
##           [,1]       [,2]         [,3]        [,4]
## [1,] -0.4079717  0.4393583 -0.005141119  0.80031055
## [2,] -0.4459639  0.5925921 -0.376611025 -0.55508094
## [3,] -0.5333365 -0.6564089 -0.526723779  0.08509762
## [4,] -0.5918002 -0.1578804  0.762036591 -0.21011087
```

```
# Percentage of variation explained by the eigenvector with largest eigenvalue.
eigenspace_forward$values[1]/sum(eigenspace_forward$values)
```

```
## [1] 0.827066
```