MAT1856/APM466 Assignment 1

Justin Leo, Student #: 1006376459

February, 2020

Fundamental Questions - 25 points

1.

   (a) Governments issue bonds to raise funds to finance day to day operations and to cover deficits in the annual budget.

   (b) The slope of the yield curve gives a loose forecast of future interest rate changes and economic activity. When short term interest rates are higher than long term bonds, an inverted yield curve occurs which usually precedes an economic recession.

   (c) When the government sells bonds to the public the public is giving the government money in exchange for the bonds, this reduces the overall money supply in the economy.

2. Bonds are chosen with maturity dates between 2020 and 2025 (maturity date up to 5 years from today's date): 01-Mar-20 01-Sep-20 01-Mar-21 01-Sep-21 01-Mar-22 01-Jun-22 01-Mar-23 01-Jun-23 01-Mar-24 01-Jun-24 01-Sep-24 01-Mar-25.

   The bonds are ordered by maturity date starting with 3/2020 up to 03/2025. These bonds are selected in such a way that we will be able to calculate the spot rates and yield rates for $r_{1/2}$,r_1 ... r_5 so that we will be able to plot our spot and yield curves over the 5 year span.

3. The eigenvectors represent a set of uncorrelated portfolios, the eigenvalues correspond to the variances of these uncorrelated portfolios. PCA is a process that takes a set of observations of potentially correlated variables and converts them to a set of values of linearly uncorrelated variables through projecting the original feature space onto a smaller (lower dimensional) subspace. The first component accounts for as much variability in the data as possible. The eigenvectors with smallest eigenvalues contain the least information about the distribution of the data. In summary, PCA aims to find patterns in data through analyzing how rates move together.

Empirical Questions - 75 points

4.

   (a) Before we can calculate the yields we first need to obtain the dirty prices. The dirty price is found using the following formula:

$$DirtyPrice = AccruedInterest + CleanPrice \tag{1}$$

   where

$$AccruedInterest = \frac{Days.since.last.payment}{365} \times AnnualCouponRate \tag{2}$$

   We create a table of our dirty prices that we will use in our following calculations:

Coupon	Maturity	20-01-02	20-01-03	20-01-04	20-01-05	20-01-06	20-01-07	20-01-08	20-01-09	20-01-10	20-01-11
0.75	1-Mar-2020	100.35137	100.365479	100.377808	100.381918	100.386027	100.390137	100.394247	100.406575	100.410685	100.414795
0.38	1-Sep-2020	99.5106849	99.5327397	99.5389041	99.5309589	99.5430137	99.5450685	99.5471233	99.5432877	99.5553425	99.5773973
0.38	1-Mar-2021	99.1406849	99.1827397	99.2089041	99.2009589	99.1830137	99.1850685	99.1471233	99.1732877	99.1753425	99.2073973
0.38	1-Sep-2021	98.6606849	98.7027397	98.7489041	98.7209589	98.7230137	98.6950685	98.6971233	98.6532877	98.6853425	98.6973973
0.25	1-Mar-2022	97.7371233	97.7984932	97.8326027	97.8239726	97.8153425	97.7767123	97.7880822	97.7521918	97.7635616	97.7949315
1.38	1-Jun-2022	102.763562	102.831096	102.883699	102.861233	102.858767	102.806301	102.813836	102.776438	102.793973	102.841507
0.88	1-Mar-2023	100.894932	101.009726	101.08411	101.058904	101.053699	100.968493	100.933288	100.907671	100.952466	101.02726
0.75	1-Jun-2023	99.6073973	99.7215068	99.7938356	99.7579452	99.7720548	99.6961644	99.690274	99.6126027	99.6667123	99.7408219
1.13	1-Mar-2024	102.972055	103.378219	103.566712	103.362877	103.509041	103.215205	103.29137	103.369863	103.266027	103.372192
1.25	1-Jun-2024	103.742329	104.079178	104.229726	104.126575	104.073425	103.910274	103.967123	103.897671	103.994521	104.10137
0.75	1-Sep-2024	99.2213699	99.4554795	99.8078082	99.6319178	99.7760274	99.520137	99.5642466	99.6065753	99.5406849	99.6547945
0.63	1-Mar-2025	98.4489041	98.6423288	98.8026027	98.7060274	98.7094521	98.5328767	98.4863014	98.4865753	98.59	98.7234247

The dirty price will be used so that we can isolate for our r(t_i) in the following equation:

$$DirtyPrice = \sum_i p_i e^{-r(t_i)t_i} \tag{3}$$

We will first need to calculate the yield for our first zero coupon bond ($r_{2/12}$), once we obtain this value we use equation (3) to calculate $r_{8/12}$, and we continue this iterative process using the previous r_is to calculate the next r_i.

For the first bond, since it is zero coupon (maturity date within 6 months) we calculate $r_{1/2}$ with:

$$r_T = \frac{-\ln \frac{P}{N}}{T} \tag{4}$$

Where P is our dirty price, N our notional value which is 100 and T = 1/2.

With $r_{1/2}$ we plug into the equation:

$$DirtyPrice = \frac{C}{2} e^{-r(t_{2/12})t_{2/12}} + (100 + \frac{C}{2}) e^{-r(t_{8/12})t_{8/12}} \tag{5}$$

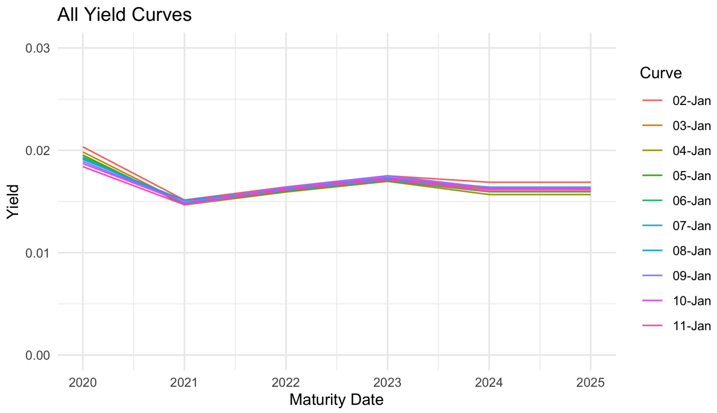
and isolate for $r(t_{8/12})$ to solve for it.

$$r_{\frac{8}{12}} = -\frac{12}{8} log[\frac{P - \frac{c}{2} e^{r_{2/12} \times \frac{2}{12}}}{100 + \frac{c}{2}}] \tag{6}$$

Using this process we obtain up to $-r(t_{62/12})$ (or $-r(t_5)$). Please see python code provided in Github for how I looped through the data to iteratively calculate the 10 r_is $\forall i$.

To save space, the results of each calculation are provided within the code as lists, for example the calculations for $r_{2/12}$ (r2) are provided under the comment "R2" and likewise for $r_{8/12}$ (r8) under "R8" up to "R62". Once we have these numbers we can calculate $r_{6/12}$ up to $r_{60/12}$ by interpolation and extrapolation.

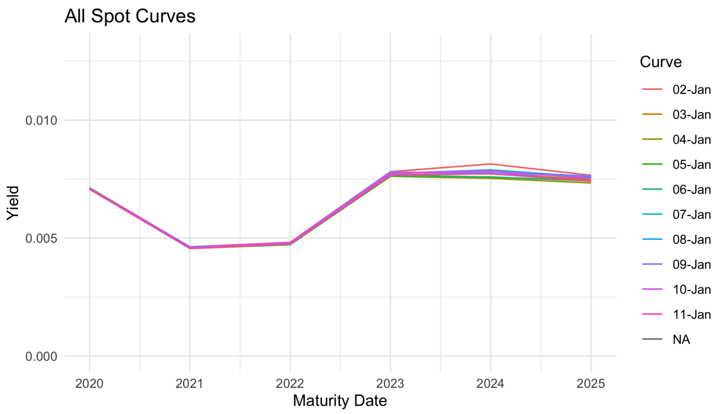
For interpolation we use a weighted average, i.e. using r2 and r8 to find r6, r14 and r8 to find r12 and so on... Up to when we arrive at r32. We are unable to interpolate to find r32 so we will use an extrapolation here. To do so we will assume the difference between r29 and r26 is the same as the difference between r32 and r29, which is a reasonable assumption since the length of time for both is 3 months.



(b) We can use the formula to solve for the corresponding spot rate r_s:

$$P_i = \sum_i \frac{C_i}{(1+r)^i} + \frac{100}{(1+r)^n} \tag{7}$$

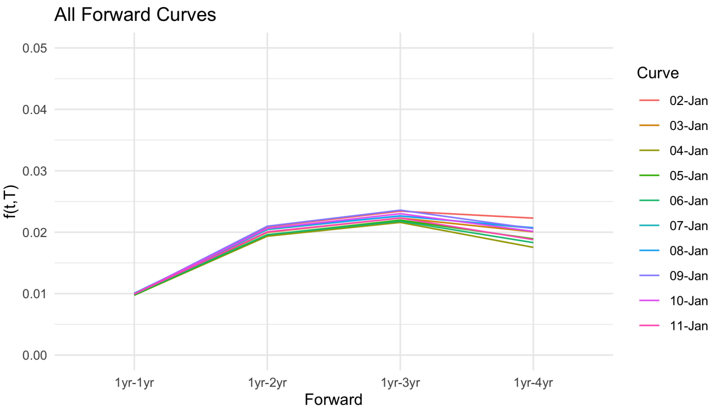
i = periods, n = coupon rate, where the 'r's we input are r1/2, r1, ... , r5 from the yield calculations. Below is the plot for each day's spot curve:



(c) The forward rates 1-1,1-2,1-3 and 1-4 are calculated using the formula:

$$F_{i,j} = [\frac{(1+r_j)^j}{(1+r_i)^i}]^{\frac{1}{j-i}} - 1, \;\; i = 1, j = 1, ..., 4 \tag{8}$$

where r_i, r_j are the spot rates (see python code for calculation).



5. Using the March bonds for each year 2020 to 2025, we obtain the data for the time series of the daily log returns and input this data into the equation given for $X_{i,j}$. For Yield, we obtain the 5 by 9 matrix of X_is. See R code in Github for the covariance calculation used to obtain the 5 by 5 matrix. For Forward, we obtain a 4 by 9 matrix and the covariance matrix is 4 by 4.

Table 1: Covariance Matrix (Yield)

	a	b	c	d	e
a	0.0001	0.00004	0.00000	-0.00001	-0.00004
b	0.00004	0.0001	0.0001	0.00004	0.0001
c	0.00000	0.0001	0.0002	0.0001	0.0003
d	-0.00001	0.00004	0.0001	0.0001	0.0002
e	-0.00004	0.0001	0.0003	0.0002	0.0005

Table 2: Covariance Matrix (Forward)

	f	g	h	j
f	0.0005	0.0001	-0.00000	0.0001
g	0.0001	0.0002	0.0001	0.0002
h	-0.00000	0.0001	0.0001	0.0001
j	0.0001	0.0002	0.0001	0.0004

6. See R code in Github. First table is for yield, second for forward.

```
eigen() decomposition
$values
[1] 7.453676e-04 1.765434e-04 6.437654e-05 2.182958e-05 1.524468e-06

$vectors
            [,1]        [,2]        [,3]         [,4]        [,5]
[1,] -0.03552927  0.80179071  0.57472862  0.158060295  0.02394322
[2,]  0.27447260  0.55741437 -0.76798190  0.003053247  0.15539787
[3,]  0.44539372  0.09161928  0.08966609 -0.590611205 -0.66058212
[4,]  0.28208636 -0.04709785  0.21241332 -0.580907649  0.73187155
[5,]  0.80340163 -0.18923153  0.16349738  0.537338189  0.05721469
```

```
$values
[1] 6.966548e-04 3.915131e-04 3.157457e-05 1.295220e-06

$vectors
          [,1]       [,2]        [,3]        [,4]
[1,] 0.4908704  0.8635656 -0.08764175  0.07496438
[2,] 0.4662851 -0.2721676 -0.58654569 -0.60371117
[3,] 0.2487783 -0.2603040 -0.49521986  0.79063802
[4,] 0.6926282 -0.3352926  0.63485461  0.06931583
```

The eigenvectors give us information about the distribution of our data, the eigenvalues returned in R are in order of size and the first eigenvalue corresponds to the first eigenvector, so the first eigenvector has the most information.

References and GitHub Link to Code

code: https://github.com/leojusti/1856

References:

"Principal Component Analysis", Raschka, Sebastian. Retrieved: https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html