DSA Project Code Final

There is no Yield Curve

11/8/2020

PART 1

#Set you working directory if required

# setwd('C:/Users/Ziyik/Desktop/New Folder')

#Load Libraries

library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(YieldCurve)

## Warning: package 'YieldCurve' was built under R version 3.6.3

## Loading required package: xts

## Warning: package 'xts' was built under R version 3.6.3

## Loading required package: zoo

##   
## Attaching package: 'zoo'

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

##   
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':  
##   
## first, last

library(xts)  
library(ustyc)

## Warning: package 'ustyc' was built under R version 3.6.3

library(forecast)

## Warning: package 'forecast' was built under R version 3.6.2

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':  
## method from   
## fitted.fracdiff fracdiff  
## residuals.fracdiff fracdiff

library(dsa)

## Warning: package 'dsa' was built under R version 3.6.3

library(lattice)  
library(plot3D)

## Warning: package 'plot3D' was built under R version 3.6.3

library(plot3Drgl)

## Warning: package 'plot3Drgl' was built under R version 3.6.3

## Loading required package: rgl

## Warning: package 'rgl' was built under R version 3.6.3

library(gridExtra)

## Warning: package 'gridExtra' was built under R version 3.6.2

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

theme\_set(theme\_light()) #do not have to run this if light theme is not preferred

#Load Part 1 Data

ZCBP <- read.table('ZCBP.txt', header = T)  
summary(ZCBP)

## time price   
## Min. : 0.3699 Min. :17.26   
## 1st Qu.: 7.5589 1st Qu.:25.55   
## Median :14.7479 Median :41.50   
## Mean :14.7479 Mean :47.16   
## 3rd Qu.:21.9370 3rd Qu.:65.98   
## Max. :29.1260 Max. :98.16

ZCBP <- ZCBP[order(ZCBP$time),]  
head(ZCBP)

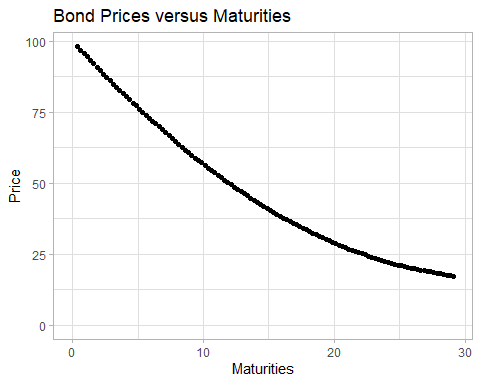
## time price  
## 39 0.3699 98.155  
## 1 0.6219 96.924  
## 40 0.8740 95.717  
## 2 1.1260 94.511  
## 41 1.3699 93.215  
## 3 1.6219 92.070

# 1.1) Plot the bond prices versus their maturities

ZCBP %>% rename(Time = time, Price = price)

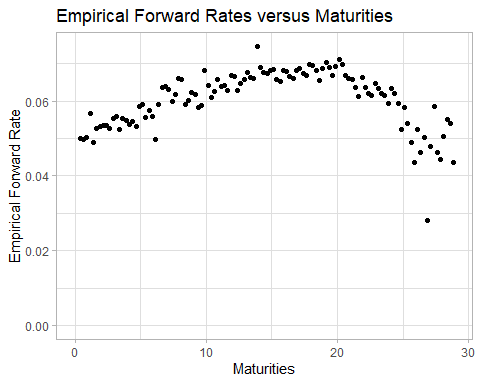
## Time Price  
## 39 0.3699 98.155  
## 1 0.6219 96.924  
## 40 0.8740 95.717  
## 2 1.1260 94.511  
## 41 1.3699 93.215  
## 3 1.6219 92.070  
## 42 1.8740 90.852  
## 4 2.1260 89.644  
## 43 2.3699 88.481  
## 5 2.6219 87.295  
## 44 2.8740 86.140  
## 6 3.1260 84.949  
## 45 3.3699 83.797  
## 7 3.6219 82.698  
## 46 3.8740 81.549  
## 8 4.1260 80.427  
## 47 4.3699 79.377  
## 9 4.6219 78.293  
## 48 4.8740 77.251  
## 10 5.1260 76.121  
## 49 5.3699 75.033  
## 11 5.6219 73.990  
## 50 5.8740 72.927  
## 12 6.1260 71.905  
## 51 6.3699 71.036  
## 13 6.6219 69.984  
## 52 6.8740 68.868  
## 14 7.1260 67.769  
## 53 7.3699 66.735  
## 15 7.6219 65.733  
## 54 7.8740 64.715  
## 16 8.1260 63.645  
## 55 8.3699 62.633  
## 17 8.6219 61.707  
## 56 8.8740 60.776  
## 18 9.1260 59.828  
## 57 9.3699 58.932  
## 19 9.6219 58.074  
## 58 9.8740 57.220  
## 20 10.1260 56.246  
## 59 10.3699 55.373  
## 21 10.6219 54.528  
## 60 10.8740 53.675  
## 22 11.1260 52.793  
## 61 11.3699 51.976  
## 23 11.6219 51.142  
## 62 11.8740 50.338  
## 24 12.1260 49.495  
## 63 12.3699 48.698  
## 25 12.6219 47.932  
## 64 12.8740 47.155  
## 26 13.1260 46.378  
## 65 13.3699 45.619  
## 27 13.6219 44.862  
## 66 13.8740 44.121  
## 28 14.1260 43.299  
## 68 14.3699 42.576  
## 29 14.6219 41.856  
## 89 14.8740 41.151  
## 30 15.1260 40.449  
## 69 15.3699 39.780  
## 31 15.6219 39.125  
## 70 15.8740 38.486  
## 32 16.1260 37.831  
## 71 16.3699 37.209  
## 33 16.6219 36.589  
## 72 16.8740 35.984  
## 34 17.1260 35.370  
## 73 17.3699 34.781  
## 35 17.6219 34.195  
## 74 17.8740 33.623  
## 36 18.1260 33.036  
## 75 18.3699 32.480  
## 37 18.6219 31.926  
## 76 18.8740 31.403  
## 38 19.1260 30.864  
## 77 19.3699 30.339  
## 67 19.6219 29.816  
## 78 19.8740 29.317  
## 79 20.1260 28.809  
## 80 20.3699 28.314  
## 81 20.6219 27.821  
## 82 20.8740 27.356  
## 83 21.1260 26.905  
## 84 21.3699 26.476  
## 85 21.6219 26.054  
## 86 21.8740 25.654  
## 96 22.1260 25.229  
## 87 22.3699 24.841  
## 97 22.6219 24.456  
## 88 22.8740 24.080  
## 90 23.1260 23.690  
## 91 23.3699 23.326  
## 92 23.6219 22.964  
## 93 23.8740 22.611  
## 94 24.1260 22.275  
## 95 24.3699 21.933  
## 98 24.6219 21.592  
## 100 24.8740 21.271  
## 99 25.1260 20.992  
## 101 25.3699 20.695  
## 102 25.6219 20.415  
## 103 25.8740 20.164  
## 104 26.1260 19.943  
## 108 26.3699 19.690  
## 105 26.6219 19.462  
## 106 26.8740 19.217  
## 107 27.1260 19.082  
## 110 27.3699 18.860  
## 109 27.6219 18.583  
## 111 27.8740 18.368  
## 115 28.1260 18.164  
## 112 28.3699 17.941  
## 116 28.6219 17.694  
## 113 28.8740 17.455  
## 114 29.1260 17.264

Maturities <- ZCBP[,1]  
Price <- ZCBP[,2]  
  
ZCBP %>% ggplot(aes(x = Maturities, y = Price)) +  
 geom\_point() + #can be geom\_lines too  
 ggtitle('Bond Prices versus Maturities') +  
 expand\_limits(x = 0, y = 0)



# 1.2) Plot the empirical forward rates as computed in equation (3) versus maturities.

j <- length(Price)-1  
ForwardRate <- matrix(nrow=j,0)  
TimeInterval <- Maturities[1:j]  
  
GGData <- cbind(TimeInterval, ForwardRate) %>%   
 as.data.frame()  
  
for(i in 1:j) #The i notation is used here but it essentially refers to j in equation (3)  
{  
ForwardRate[i] <- -(log(Price[i+1])-(log(Price[i])))/(Maturities[i+1]-Maturities[i])  
}  
  
GGData %>% ggplot(aes(x = TimeInterval, y = ForwardRate)) +   
 geom\_point() +  
 ggtitle('Empirical Forward Rates versus Maturities') +  
 labs(x = 'Maturities', y = 'Empirical Forward Rate') +  
 expand\_limits(x = 0, y = 0)



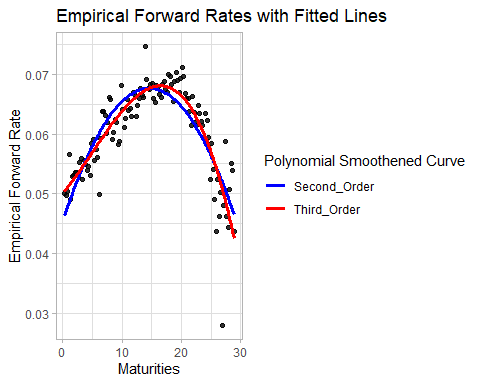
# 1.3) Smooth the empirical forward rates using second order and third order polynomials.Superimpose the smoothed curves versus the empirical forward rates.

Smoothen

SecondOrder <- lm(ForwardRate ~ TimeInterval + I(TimeInterval^2))  
SecondCoef <- SecondOrder$coefficients  
SecondSmoothen <- SecondCoef[1] + SecondCoef[2]\*TimeInterval + SecondCoef[3]\*TimeInterval^2  
  
ThirdOrder <- lm(ForwardRate ~ TimeInterval + I(TimeInterval^2)+ I(TimeInterval^3))  
ThirdCoef <- ThirdOrder$coefficients  
ThirdSmoothen <- ThirdCoef[1] + ThirdCoef[2]\*TimeInterval + ThirdCoef[3]\*TimeInterval^2 + ThirdCoef[4]\*TimeInterval^3

GGplot

GGData2 <- cbind(GGData,SecondSmoothen,ThirdSmoothen) %>%   
 as.data.frame()  
  
GGData2 %>% ggplot(aes(x = TimeInterval, y = ForwardRate)) +  
 geom\_point(alpha = 0.8) +  
 geom\_line(mapping = aes(y = SecondSmoothen, colour = 'Second\_Order'),size = 1.2) +  
 geom\_line(mapping = aes(y = ThirdSmoothen, colour = 'Third\_Order'), size = 1.2) +  
 ggtitle('Empirical Forward Rates with Fitted Lines') +  
 labs(x = 'Maturities', y = 'Empirical Forward Rate') +  
 theme(legend.position = 'right') +  
 scale\_colour\_manual(name = "Polynomial Smoothened Curve",  
 values = c(Second\_Order='blue', Third\_Order='red'))



# 1.4) Estimate the empirical spot rates of interest using Method 2 for any maturities up to 30 years.

Method 1: Defining t values based on the interval in the dataset. We choose t to be the upper value of each interval

SpotRate <- NULL  
for(i in 2:j){  
 SpotRate[i] = (1/Maturities[i])\*(-(log(0.01\*Price[1]))+cumsum((ForwardRate[1:i-1]%\*%(Maturities[2:i]-Maturities[1:i-1]))))  
}  
head(SpotRate); tail(SpotRate)

## [1] NA 0.05023801 0.05008497 0.05013673 0.05128953 0.05094089

## [1] 0.06092713 0.06079358 0.06064597 0.06056002 0.06051117 0.06045384

Method 2: Using cut() to split into 116 equal parts and define t values by picking the upper value of each interval

(Labels = levels(cut(Maturities,breaks=116)))

## [1] "(0.341,0.618]" "(0.618,0.866]" "(0.866,1.11]" "(1.11,1.36]"   
## [5] "(1.36,1.61]" "(1.61,1.86]" "(1.86,2.11]" "(2.11,2.35]"   
## [9] "(2.35,2.6]" "(2.6,2.85]" "(2.85,3.1]" "(3.1,3.34]"   
## [13] "(3.34,3.59]" "(3.59,3.84]" "(3.84,4.09]" "(4.09,4.34]"   
## [17] "(4.34,4.58]" "(4.58,4.83]" "(4.83,5.08]" "(5.08,5.33]"   
## [21] "(5.33,5.58]" "(5.58,5.82]" "(5.82,6.07]" "(6.07,6.32]"   
## [25] "(6.32,6.57]" "(6.57,6.82]" "(6.82,7.06]" "(7.06,7.31]"   
## [29] "(7.31,7.56]" "(7.56,7.81]" "(7.81,8.05]" "(8.05,8.3]"   
## [33] "(8.3,8.55]" "(8.55,8.8]" "(8.8,9.05]" "(9.05,9.29]"   
## [37] "(9.29,9.54]" "(9.54,9.79]" "(9.79,10]" "(10,10.3]"   
## [41] "(10.3,10.5]" "(10.5,10.8]" "(10.8,11]" "(11,11.3]"   
## [45] "(11.3,11.5]" "(11.5,11.8]" "(11.8,12]" "(12,12.3]"   
## [49] "(12.3,12.5]" "(12.5,12.8]" "(12.8,13]" "(13,13.3]"   
## [53] "(13.3,13.5]" "(13.5,13.8]" "(13.8,14]" "(14,14.3]"   
## [57] "(14.3,14.5]" "(14.5,14.7]" "(14.7,15]" "(15,15.2]"   
## [61] "(15.2,15.5]" "(15.5,15.7]" "(15.7,16]" "(16,16.2]"   
## [65] "(16.2,16.5]" "(16.5,16.7]" "(16.7,17]" "(17,17.2]"   
## [69] "(17.2,17.5]" "(17.5,17.7]" "(17.7,18]" "(18,18.2]"   
## [73] "(18.2,18.5]" "(18.5,18.7]" "(18.7,19]" "(19,19.2]"   
## [77] "(19.2,19.5]" "(19.5,19.7]" "(19.7,20]" "(20,20.2]"   
## [81] "(20.2,20.4]" "(20.4,20.7]" "(20.7,20.9]" "(20.9,21.2]"   
## [85] "(21.2,21.4]" "(21.4,21.7]" "(21.7,21.9]" "(21.9,22.2]"   
## [89] "(22.2,22.4]" "(22.4,22.7]" "(22.7,22.9]" "(22.9,23.2]"   
## [93] "(23.2,23.4]" "(23.4,23.7]" "(23.7,23.9]" "(23.9,24.2]"   
## [97] "(24.2,24.4]" "(24.4,24.7]" "(24.7,24.9]" "(24.9,25.2]"   
## [101] "(25.2,25.4]" "(25.4,25.7]" "(25.7,25.9]" "(25.9,26.2]"   
## [105] "(26.2,26.4]" "(26.4,26.6]" "(26.6,26.9]" "(26.9,27.1]"   
## [109] "(27.1,27.4]" "(27.4,27.6]" "(27.6,27.9]" "(27.9,28.1]"   
## [113] "(28.1,28.4]" "(28.4,28.6]" "(28.6,28.9]" "(28.9,29.2]"

DF = cbind(lower = as.numeric(sub("\\((.+),.\*", "\\1", Labels)),  
 upper = as.numeric(sub("[^,]\*,([^]]\*)\\]", "\\1", Labels)))  
TimeCut = DF[,2]  
SpotRate2 <- NULL  
for(i in 2:j){  
 SpotRate2[i] = (1/TimeCut[i])\*(-(log(0.01\*Price[1]))+cumsum((ForwardRate[1:i-1]%\*%(Maturities[2:i]-Maturities[1:i-1])))-(ForwardRate[i-1]\*(Maturities[i]-Maturities[i-1]))+ForwardRate[i-1]\*((TimeCut[i]-Maturities[i-1])))  
}  
head(SpotRate2); tail(SpotRate2)

## [1] NA 0.05019407 0.05000470 0.05016761 0.05208322 0.05069827

## [1] 0.06092888 0.06077994 0.06066108 0.06054951 0.06051538 0.06044798

Remove all NAs

SpotRate <- na.omit(SpotRate)  
SpotRate2 <- na.omit(SpotRate2)  
  
head(SpotRate); tail(SpotRate)

## [1] 0.05023801 0.05008497 0.05013673 0.05128953 0.05094089 0.05119444

## [1] 0.06092713 0.06079358 0.06064597 0.06056002 0.06051117 0.06045384

head(SpotRate2); tail(SpotRate2)

## [1] 0.05019407 0.05000470 0.05016761 0.05208322 0.05069827 0.05137689

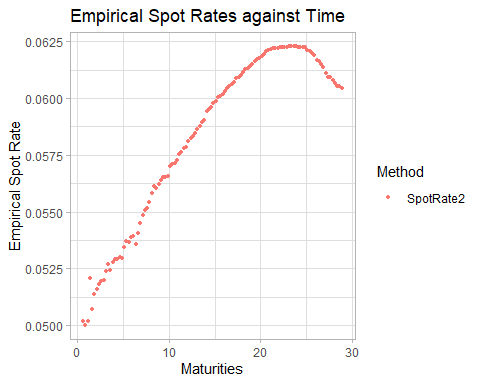
## [1] 0.06092888 0.06077994 0.06066108 0.06054951 0.06051538 0.06044798

# 1.5) Smooth the empirical spot rates using second order and third order polynomials.

# Superimpose the smoothed curves versus the empirical spot rates.

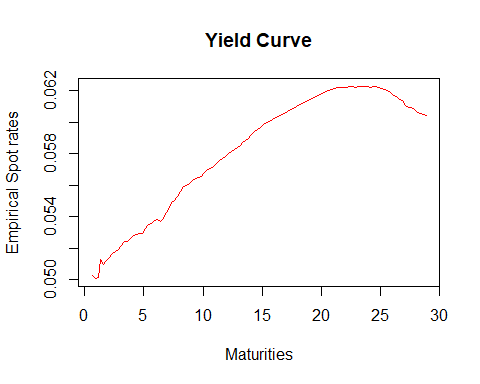
GGplot

TimeIntervalGG <- TimeInterval[-1]  
GGData2 <- cbind(TimeIntervalGG,SpotRate,SpotRate2) %>%   
 as.data.frame() %>%   
 pivot\_longer(cols = -TimeIntervalGG,  
 names\_to = 'Method',  
 values\_to = 'SpotRate')  
  
GGData2 %>% filter(Method == 'SpotRate2') %>% ggplot(aes(x = TimeIntervalGG, y = SpotRate, group = Method, colour = Method)) +  
 geom\_point(size = 1) +  
 ggtitle('Empirical Spot Rates against Time') +  
 labs(x = 'Maturities', y = 'Empirical Spot Rate')

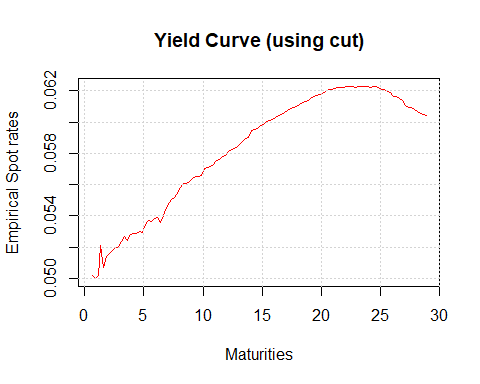


Normal Plot

plot(TimeIntervalGG, SpotRate, main = 'Yield Curve', xlab = 'Maturities', ylab='Empirical Spot rates', col = 'red',type='l')



plot(TimeIntervalGG, SpotRate2, main = 'Yield Curve (using cut)', xlab = 'Maturities', ylab='Empirical Spot rates', col = 'red',type='l')  
grid()

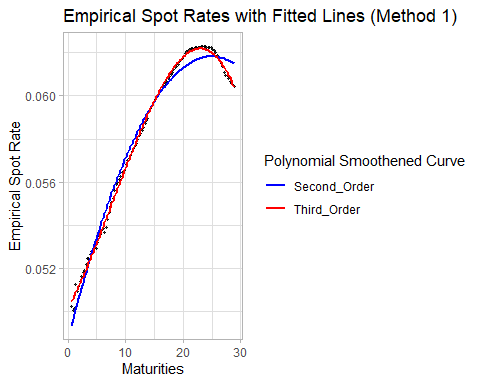


Smoothen

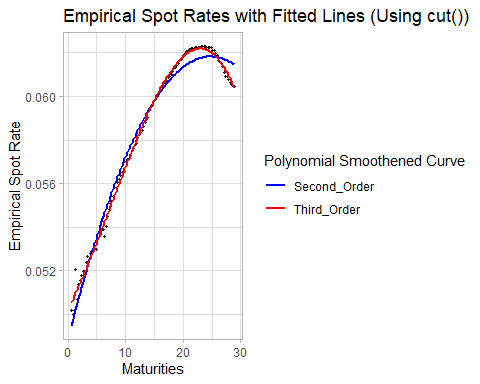
# Method 1  
SecondSpot = lm(SpotRate ~ TimeInterval[-1] + I(TimeInterval[-1]^2))  
SecondSpotCoef = SecondSpot$coefficients  
SecondSpotSmooth = SecondSpotCoef[1] + SecondSpotCoef[2]\*TimeInterval[-1] + SecondSpotCoef[3]\*TimeInterval[-1]^2  
  
ThirdSpot = lm(SpotRate ~ TimeInterval[-1] + I(TimeInterval[-1]^2) + I(TimeInterval[-1]^3))  
ThirdSpotCoef = ThirdSpot$coefficients  
ThirdSpotSmooth = ThirdSpotCoef[1] + ThirdSpotCoef[2]\*TimeInterval[-1] + ThirdSpotCoef[3]\*TimeInterval[-1]^2 +   
 ThirdSpotCoef[4]\*TimeInterval[-1]^3  
  
# Method 2  
SecondSpot2 = lm(SpotRate2 ~ TimeInterval[-1] + I(TimeInterval[-1]^2))  
SecondSpot2Coef = SecondSpot2$coefficients  
SecondSpot2Smooth = SecondSpot2Coef[1] + SecondSpot2Coef[2]\*TimeInterval[-1] + SecondSpot2Coef[3]\*TimeInterval[-1]^2  
  
ThirdSpot2 = lm(SpotRate2 ~ TimeInterval[-1] + I(TimeInterval[-1]^2) + I(TimeInterval[-1]^3))  
ThirdSpot2Coef = ThirdSpot2$coefficients  
ThirdSpot2Smooth = ThirdSpot2Coef[1] + ThirdSpot2Coef[2]\*TimeInterval[-1] + ThirdSpot2Coef[3]\*TimeInterval[-1]^2 +   
 ThirdSpot2Coef[4]\*TimeInterval[-1]^3

Superimpose

# Method 1  
GGData3 <- cbind(TimeIntervalGG,SpotRate,SecondSpotSmooth,ThirdSpotSmooth) %>%   
 as.data.frame()  
  
GGData3 %>% ggplot(aes(x = TimeIntervalGG, y = SpotRate)) +  
 geom\_point(alpha = 0.8, size = 0.7) +  
 geom\_line(mapping = aes(y = SecondSpotSmooth, colour = 'Second\_Order'),size = 1) +  
 geom\_line(mapping = aes(y = ThirdSpotSmooth, colour = 'Third\_Order'), size = 1) +  
 ggtitle('Empirical Spot Rates with Fitted Lines (Method 1)') +  
 labs(x = 'Maturities', y = 'Empirical Spot Rate') +  
 theme(legend.position = 'right') +  
 scale\_colour\_manual(name = "Polynomial Smoothened Curve",  
 values = c(Second\_Order='blue', Third\_Order='red'))



# Method 2  
GGData4 <- cbind(TimeIntervalGG,SpotRate2,SecondSpot2Smooth,ThirdSpot2Smooth) %>%   
 as.data.frame()  
  
GGData4 %>% ggplot(aes(x = TimeIntervalGG, y = SpotRate2)) +  
 geom\_point(alpha = 0.8, size = 0.7) +  
 geom\_line(mapping = aes(y = SecondSpot2Smooth, colour = 'Second\_Order'),size = 1) +  
 geom\_line(mapping = aes(y = ThirdSpot2Smooth, colour = 'Third\_Order'), size = 1) +  
 ggtitle('Empirical Spot Rates with Fitted Lines (Using cut())') +  
 labs(x = 'Maturities', y = 'Empirical Spot Rate') +  
 theme(legend.position = 'right') +  
 scale\_colour\_manual(name = "Polynomial Smoothened Curve",  
 values = c(Second\_Order='blue', Third\_Order='red'))



# Part 1.6) Comment on your results (refer to report)

PART 2

# Load data and stuff TO DELETE

library(ggplot2)  
library(dplyr)  
library(tidyr)  
library(YieldCurve)  
library(xts)  
library(ustyc)  
library(forecast)  
library(dsa)  
library(lattice)  
  
theme\_set(theme\_light()) #do not have to run this if light theme is not preferred

# 2.1) Presenting Data

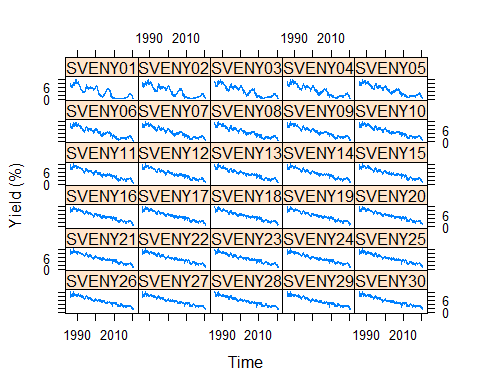
ZCBYF86 <- read.csv("ZCBYF86.csv")  
date=as.Date(ZCBYF86$Date,format="%d/%m/%Y")  
ZCBYF86=cbind(date,ZCBYF86[,-1])  
ZCBYF86.xts=xts(ZCBYF86[,2:31],order.by=ZCBYF86[,1])  
head(ZCBYF86.xts)

## SVENY01 SVENY02 SVENY03 SVENY04 SVENY05 SVENY06 SVENY07 SVENY08  
## 1986-01-02 7.6240 7.9553 8.2166 8.4281 8.6041 8.7544 8.8857 9.0028  
## 1986-01-03 7.6214 7.9657 8.2325 8.4454 8.6206 8.7692 8.8986 9.0138  
## 1986-01-06 7.6333 7.9740 8.2407 8.4551 8.6319 8.7818 8.9119 9.0271  
## 1986-01-07 7.6170 7.9314 8.1802 8.3820 8.5498 8.6928 8.8175 8.9283  
## 1986-01-08 7.6611 7.9863 8.2531 8.4753 8.6628 8.8233 8.9624 9.0842  
## 1986-01-09 7.8313 8.1833 8.4575 8.6753 8.8518 8.9981 9.1220 9.2289  
## SVENY09 SVENY10 SVENY11 SVENY12 SVENY13 SVENY14 SVENY15 SVENY16  
## 1986-01-02 9.1087 9.2056 9.2951 9.3782 9.4557 9.5280 9.5957 9.6591  
## 1986-01-03 9.1183 9.2142 9.3031 9.3860 9.4635 9.5362 9.6045 9.6686  
## 1986-01-06 9.1309 9.2257 9.3130 9.3940 9.4694 9.5400 9.6060 9.6680  
## 1986-01-07 9.0284 9.1198 9.2042 9.2826 9.3556 9.4240 9.4881 9.5482  
## 1986-01-08 9.1920 9.2881 9.3745 9.4526 9.5236 9.5884 9.6477 9.7022  
## 1986-01-09 9.3230 9.4070 9.4830 9.5523 9.6161 9.6752 9.7301 9.7812  
## SVENY17 SVENY18 SVENY19 SVENY20 SVENY21 SVENY22 SVENY23 SVENY24  
## 1986-01-02 9.7184 9.7741 9.8262 9.8750 9.9207 9.9636 10.0038 10.0414  
## 1986-01-03 9.7288 9.7852 9.8383 9.8880 9.9347 9.9784 10.0195 10.0580  
## 1986-01-06 9.7260 9.7805 9.8315 9.8795 9.9244 9.9666 10.0062 10.0433  
## 1986-01-07 9.6047 9.6577 9.7075 9.7542 9.7982 9.8395 9.8782 9.9147  
## 1986-01-08 9.7523 9.7987 9.8415 9.8812 9.9180 9.9523 9.9842 10.0139  
## 1986-01-09 9.8290 9.8737 9.9155 9.9547 9.9915 10.0260 10.0584 10.0888  
## SVENY25 SVENY26 SVENY27 SVENY28 SVENY29 SVENY30  
## 1986-01-02 10.0768 10.1100 10.1411 10.1704 10.1980 10.2240  
## 1986-01-03 10.0941 10.1281 10.1600 10.1900 10.2182 10.2447  
## 1986-01-06 10.0782 10.1110 10.1418 10.1708 10.1981 10.2239  
## 1986-01-07 9.9490 9.9812 10.0115 10.0401 10.0670 10.0924  
## 1986-01-08 10.0416 10.0676 10.0918 10.1146 10.1360 10.1560  
## 1986-01-09 10.1174 10.1443 10.1696 10.1934 10.2159 10.2370

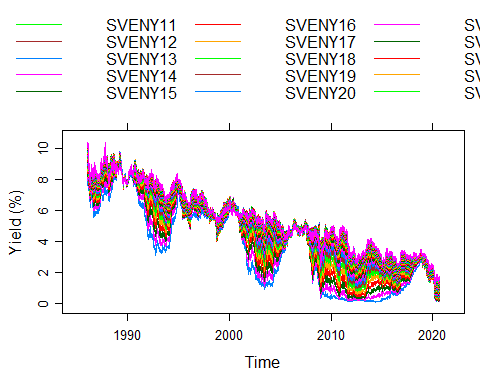
dim(ZCBYF86.xts)

## [1] 8650 30

par(mar=c(1,1,1,1))  
  
xyplot.ts(ZCBYF86.xts,scales=list(y=list(relation="same")),ylab="Yield (%)")



xyplot.ts(ZCBYF86.xts,superpose=TRUE,auto.key=list(columns=6), ylab="Yield (%)")



Defining functions for later. Rewriting Nelson.Siegel() and svensson() functions to retrieve SSR, AIC and BIC. New functions are NSFit() and NSSFit()

#Use NSFit() and NSSFit() instead  
  
### factorBeta1  
.FactorBeta1 <- function (lambda, maturity)   
{  
 as.numeric((1 - exp(-lambda \* maturity))/(lambda \* maturity))  
}  
  
### factorBeta2  
.FactorBeta2 <- function (lambda, maturity)   
{  
 as.numeric((1 - exp(-lambda \* maturity))/(lambda \* maturity) -   
 exp(-lambda \* maturity))  
}  
  
### .NS.estimator  
NSestimator <- function(rate, maturity, lambda)   
{  
 beta <- lm(rate ~ 1 + .FactorBeta1(lambda, maturity) + .FactorBeta2(lambda,   
 maturity))  
 betaPar <- coef(beta)  
 NaValues <- na.omit(betaPar)  
 AIC <- AIC(beta)  
 BIC <- BIC(beta)  
 if (length(NaValues) < 3)   
 betaPar <- c(0, 0, 0)  
 names(betaPar) <- c("beta\_0", "beta\_1", "beta\_2")  
 EstResults <- list(Par = betaPar, Res = resid(beta), AIC = AIC, BIC = BIC)  
 return(EstResults)  
}  
  
### .NS.estimator  
NSFit <- function (rate, maturity)   
{  
 rate <- try.xts(rate, error = as.matrix)  
 if (ncol(rate) == 1)   
 rate <- matrix(as.vector(rate), 1, nrow(rate))  
 pillars.number <- length(maturity)  
 lambdaValues <- seq(maturity[1], maturity[pillars.number],   
 by = 0.5)  
 FinalResults <- matrix(0, nrow(rate), 7)  
 colnames(FinalResults) <- c("beta\_0", "beta\_1",   
 "beta\_2", "lambda","SSR","AIC", "BIC")  
 j <- 1  
 while (j <= nrow(rate)) {  
 InterResults <- matrix(0, length(lambdaValues), 7)  
 colnames(InterResults) <- c("beta0", "beta1",   
 "beta2", "lambda", "SSR","AIC", "BIC")  
 for (i in 1:length(lambdaValues)) {  
 lambdaTemp <- optimize(.FactorBeta2, interval = c(0.001,   
 1), maturity = lambdaValues[i], maximum = TRUE)$maximum  
 InterEstimation <- NSestimator(as.numeric(rate[j,   
 ]), maturity, lambdaTemp)  
 BetaCoef <- InterEstimation$Par  
 AIC <- InterEstimation$AIC  
 BIC <- InterEstimation$BIC  
 if (BetaCoef[1] > 0 & BetaCoef[1] < 20) {  
 SSR <- sum(InterEstimation$Res^2)  
 InterResults[i, ] <- c(BetaCoef, lambdaTemp,   
 SSR,AIC,BIC)  
 }  
 else {  
 InterResults[i, ] <- c(BetaCoef, lambdaValues[i],   
 1e+05,AIC,BIC)  
 }  
 }  
 BestRow <- which.min(InterResults[, 5])  
 FinalResults[j, ] <- InterResults[BestRow,]  
 j <- j + 1  
 }  
 reclass(FinalResults, rate)  
}  
  
###.beta1Spot  
.Beta1Spot <- function (maturity, tau)   
{  
 as.numeric((1 - exp(-maturity/tau))/(maturity/tau))  
}  
  
###.beta2Spot  
.Beta2Spot <- function (maturity, tau)   
{  
 as.numeric(((1 - exp(-maturity/tau))/(maturity/tau) - exp(-maturity/tau)))  
}  
  
###.NSS.estimator  
NSSestimator <- function (rate, maturity, tau1, tau2)   
{  
 beta <- lm(rate ~ 1 + .Beta1Spot(maturity, tau1) + .Beta2Spot(maturity,   
 tau1) + .Beta2Spot(maturity, tau2))  
 betaPar <- coef(beta)  
 AIC <- AIC(beta)  
 BIC <- BIC(beta)  
 NaValues <- na.omit(betaPar)  
 if (length(NaValues) < 4)   
 betaPar <- c(0, 0, 0, 0)  
 names(betaPar) <- c("beta\_0", "beta\_1", "beta\_2",   
 "beta\_3")  
 EstResults <- list(Par = betaPar, Res = resid(beta), AIC = AIC, BIC = BIC)  
 return(EstResults)  
}  
  
  
###Svensson  
NSSFit <- function (rate, maturity)   
{  
 rate <- try.xts(rate, error = as.matrix)  
 if (ncol(rate) == 1)   
 rate <- matrix(as.vector(rate), 1, nrow(rate))  
 pillars.number <- length(maturity)  
 Tau1Values <- seq(maturity[1], median(maturity), by = 1)  
 Tau2Values <- seq(median(maturity), maturity[pillars.number],   
 by = 1.5)  
 FinalResults <- matrix(0, nrow(rate), 9)  
 FinalResultsTau2 <- matrix(0, length(Tau1Values), 9)  
 colnames(FinalResults) <- c("beta\_0", "beta\_1",   
 "beta\_2", "beta\_3", "tau1", "tau2", "SSR", "AIC", "BIC")  
 j <- 1  
 while (j <= nrow(rate)) {  
 InterResultsTau1 <- matrix(0, length(Tau1Values), 9)  
 InterResultsTau2 <- matrix(0, length(Tau2Values), 9)  
 for (i in 1:length(Tau1Values)) {  
 Tau1Temp <- optimize(.Beta2Spot, interval = c(0.001,   
 max(Tau1Values)), maturity = Tau1Values[i], maximum = TRUE)$maximum  
 for (a in 1:length(Tau2Values)) {  
 Tau2Temp <- optimize(.Beta2Spot, interval = c(0.001,   
 maturity[pillars.number]), maturity = Tau2Values[a],   
 maximum = TRUE)$maximum  
 InterEstimation <- NSSestimator(as.numeric(rate[j,   
 ]), maturity, Tau1Temp, Tau2Temp)  
 BetaCoef <- InterEstimation$Par  
 SSR <- sum(InterEstimation$Res^2)  
 AIC <- InterEstimation$AIC  
 BIC <- InterEstimation$BIC  
 InterResultsTau2[a, ] <- c(BetaCoef, Tau1Temp,   
 Tau2Temp, SSR, AIC, BIC)  
 }  
 BestRowTau2 <- which.min(InterResultsTau2[, 7])  
 FinalResultsTau2[i, ] <- InterResultsTau2[BestRowTau2, ]  
 }  
 BestRow <- which.min(FinalResultsTau2[, 7])  
 FinalResults[j, ] <- FinalResultsTau2[BestRow,]  
 j <- j + 1  
 }  
 reclass(FinalResults, rate)  
}

# 2.2 Fit the NS and NSS Models to the yield data by minimizing the sum of squared errors. Compare the two models using some suitable diagnostics and model selection criteria.

Fitting the NS model

# For this dataset, we have 8650 yield curves, 1 for each time period / day.   
# Each yield curve is fitted on 30 data points (30 maturities) with their corresponding 30 spot rates  
Maturity = c(1:30)  
head(ZCBYF86.xts)

## SVENY01 SVENY02 SVENY03 SVENY04 SVENY05 SVENY06 SVENY07 SVENY08  
## 1986-01-02 7.6240 7.9553 8.2166 8.4281 8.6041 8.7544 8.8857 9.0028  
## 1986-01-03 7.6214 7.9657 8.2325 8.4454 8.6206 8.7692 8.8986 9.0138  
## 1986-01-06 7.6333 7.9740 8.2407 8.4551 8.6319 8.7818 8.9119 9.0271  
## 1986-01-07 7.6170 7.9314 8.1802 8.3820 8.5498 8.6928 8.8175 8.9283  
## 1986-01-08 7.6611 7.9863 8.2531 8.4753 8.6628 8.8233 8.9624 9.0842  
## 1986-01-09 7.8313 8.1833 8.4575 8.6753 8.8518 8.9981 9.1220 9.2289  
## SVENY09 SVENY10 SVENY11 SVENY12 SVENY13 SVENY14 SVENY15 SVENY16  
## 1986-01-02 9.1087 9.2056 9.2951 9.3782 9.4557 9.5280 9.5957 9.6591  
## 1986-01-03 9.1183 9.2142 9.3031 9.3860 9.4635 9.5362 9.6045 9.6686  
## 1986-01-06 9.1309 9.2257 9.3130 9.3940 9.4694 9.5400 9.6060 9.6680  
## 1986-01-07 9.0284 9.1198 9.2042 9.2826 9.3556 9.4240 9.4881 9.5482  
## 1986-01-08 9.1920 9.2881 9.3745 9.4526 9.5236 9.5884 9.6477 9.7022  
## 1986-01-09 9.3230 9.4070 9.4830 9.5523 9.6161 9.6752 9.7301 9.7812  
## SVENY17 SVENY18 SVENY19 SVENY20 SVENY21 SVENY22 SVENY23 SVENY24  
## 1986-01-02 9.7184 9.7741 9.8262 9.8750 9.9207 9.9636 10.0038 10.0414  
## 1986-01-03 9.7288 9.7852 9.8383 9.8880 9.9347 9.9784 10.0195 10.0580  
## 1986-01-06 9.7260 9.7805 9.8315 9.8795 9.9244 9.9666 10.0062 10.0433  
## 1986-01-07 9.6047 9.6577 9.7075 9.7542 9.7982 9.8395 9.8782 9.9147  
## 1986-01-08 9.7523 9.7987 9.8415 9.8812 9.9180 9.9523 9.9842 10.0139  
## 1986-01-09 9.8290 9.8737 9.9155 9.9547 9.9915 10.0260 10.0584 10.0888  
## SVENY25 SVENY26 SVENY27 SVENY28 SVENY29 SVENY30  
## 1986-01-02 10.0768 10.1100 10.1411 10.1704 10.1980 10.2240  
## 1986-01-03 10.0941 10.1281 10.1600 10.1900 10.2182 10.2447  
## 1986-01-06 10.0782 10.1110 10.1418 10.1708 10.1981 10.2239  
## 1986-01-07 9.9490 9.9812 10.0115 10.0401 10.0670 10.0924  
## 1986-01-08 10.0416 10.0676 10.0918 10.1146 10.1360 10.1560  
## 1986-01-09 10.1174 10.1443 10.1696 10.1934 10.2159 10.2370

# NSParams <- NSFit(rate=ZCBYF86.xts,maturity=Maturity) # for our own nelson siegel function to retrieve SSR, AIC and BIC

Run and load pre-saved parameters.

# save(NSParams, file = 'NSParams.Rda')  
load(file = 'NSParams.Rda')  
head(NSParams)

## Warning: timezone of object (UTC) is different than current timezone ().

## beta\_0 beta\_1 beta\_2 lambda SSR AIC  
## 1986-01-02 10.86152 -3.425517 -0.01331838 0.1707952 0.024828237 -119.7728  
## 1986-01-03 10.88648 -3.438736 -0.04045802 0.1707952 0.032436207 -111.7540  
## 1986-01-06 10.82586 -3.380636 -0.05993100 0.1793200 0.029690427 -114.4075  
## 1986-01-07 10.69875 -3.257036 -0.03091129 0.1707952 0.024680149 -119.9523  
## 1986-01-08 10.64575 -3.265538 0.03020935 0.2109684 0.007936658 -153.9875  
## 1986-01-09 10.66024 -3.060790 -0.06672492 0.2241687 0.029487766 -114.6130  
## BIC  
## 1986-01-02 -114.1680  
## 1986-01-03 -106.1492  
## 1986-01-06 -108.8027  
## 1986-01-07 -114.3475  
## 1986-01-08 -148.3827  
## 1986-01-09 -109.0082

Fitting and plotting the yield curve (NS)

NSyield = NSrates(NSParams, Maturity)   
head(NSyield)

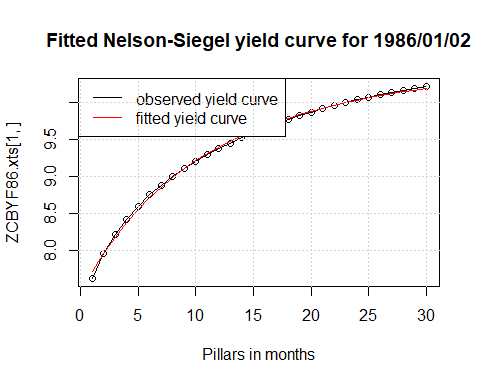
## Warning: timezone of object (UTC) is different than current timezone ().

## X1 X2 X3 X4 X5 X6 X7  
## 1986-01-02 7.711554 7.957954 8.178656 8.376674 8.554647 8.714884 8.859410  
## 1986-01-03 7.722295 7.968015 8.188296 8.386101 8.564023 8.724337 8.869038  
## 1986-01-06 7.726225 7.976534 8.199898 8.399581 8.578423 8.738899 8.883171  
## 1986-01-07 7.702318 7.935501 8.144491 8.332111 8.500834 8.652825 8.789985  
## 1986-01-08 7.704447 7.986491 8.232478 8.447592 8.636223 8.802095 8.948366  
## 1986-01-09 7.911811 8.182427 8.417529 8.622359 8.801333 8.958178 9.096042  
## X8 X9 X10 X11 X12 X13 X14  
## 1986-01-02 8.989999 9.108211 9.215416 9.312819 9.401480 9.482335 9.556209  
## 1986-01-03 8.999879 9.118398 9.225950 9.323728 9.412782 9.494040 9.568321  
## 1986-01-06 9.013127 9.130414 9.236478 9.332582 9.419836 9.499213 9.571569  
## 1986-01-07 8.913981 9.026278 9.128166 9.220777 9.305112 9.382052 9.452374  
## 1986-01-08 9.077723 9.192453 9.294508 9.385553 9.467015 9.540114 9.605901  
## 1986-01-09 9.217593 9.325089 9.420451 9.505310 9.581058 9.648882 9.709797  
## X15 X16 X17 X18 X19 X20 X21  
## 1986-01-02 9.623831 9.685845 9.742823 9.795270 9.843634 9.888314 9.929665  
## 1986-01-03 9.636349 9.698765 9.756136 9.808965 9.857701 9.902741 9.944438  
## 1986-01-06 9.637657 9.698140 9.753604 9.804565 9.851480 9.894754 9.934744  
## 1986-01-07 9.516768 9.575841 9.630133 9.680122 9.726232 9.768841 9.808284  
## 1986-01-08 9.665277 9.719020 9.767801 9.812201 9.852722 9.889801 9.923819  
## 1986-01-09 9.764675 9.814261 9.859199 9.900042 9.937270 9.971297 10.002481  
## X22 X23 X24 X25 X26 X27 X28  
## 1986-01-02 9.968002 10.003607 10.036732 10.067600 10.096413 10.123352 10.14858  
## 1986-01-03 9.983108 10.019032 10.052462 10.083622 10.112714 10.139919 10.16540  
## 1986-01-06 9.971770 10.006115 10.038029 10.067739 10.095443 10.121322 10.14554  
## 1986-01-07 9.844860 9.878836 9.910451 9.939917 9.967426 9.993149 10.01724  
## 1986-01-08 9.955107 9.983954 10.010615 10.035311 10.058238 10.079569 10.09946  
## 1986-01-09 10.031137 10.057535 10.081913 10.104481 10.125419 10.144890 10.16303  
## X29 X30  
## 1986-01-02 10.17223 10.19445  
## 1986-01-03 10.18930 10.21175  
## 1986-01-06 10.16823 10.18953  
## 1986-01-07 10.03983 10.06106  
## 1986-01-08 10.11803 10.13542  
## 1986-01-09 10.17998 10.19583

Dates <- matrix(rev(date))  
  
NSyieldDF <- cbind(Dates, NSyield) %>%   
 as.data.frame() %>% rename(Date = Dates) %>%  
 mutate(Date = as.Date(Date)) %>%   
 pivot\_longer(cols = -Date,  
 names\_to = 'Maturity',  
 values\_to = 'NSEstimated') %>%   
 mutate(Maturity = as.numeric(gsub(pattern = 'X', replacement = '', x = Maturity)))  
  
ObservedYield <- ZCBYF86[nrow(ZCBYF86):1,] %>%   
 rename(Date = date) %>%   
 pivot\_longer(cols = -Date,  
 names\_to = 'Maturity',  
 values\_to = 'ObservedYield') %>%   
 mutate(Maturity = as.numeric(gsub(pattern = 'SVENY', replacement = '', x = Maturity)))

Normal plot

y.1 = NSyield[1,]  
plot(Maturity,ZCBYF86.xts[1,],main="Fitted Nelson-Siegel yield curve for 1986/01/02",  
xlab=c("Pillars in months"), type="o")  
lines(Maturity,y.1, col=2)  
legend("topleft",legend=c("observed yield curve","fitted yield curve"),  
col=c(1,2),lty=1)  
grid()



#Fitting NSS Model Load pre-saved parameters

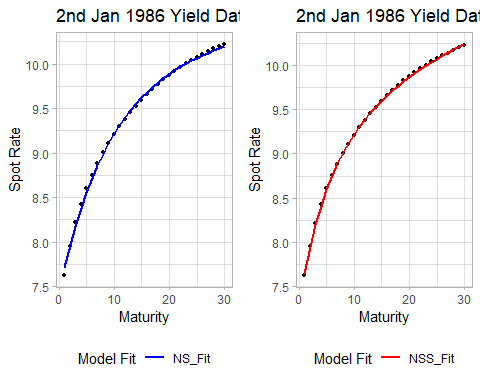
# NSSParams = NSSFit(rate=ZCBYF86.xts,maturity=Maturity) # Unhash to   
# head(NSSParams)  
# save(NSSParams, file = 'NSSParams.Rda')  
load(file = 'NSSParams.Rda')  
NSSyield = Srates(NSSParams, Maturity, whichRate = "Spot")   
head(NSSyield)

## Warning: timezone of object (UTC) is different than current timezone ().

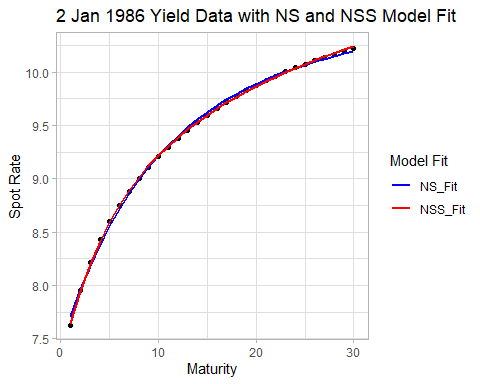
## X1 X2 X3 X4 X5 X6 X7  
## 1986-01-02 7.647572 7.945813 8.197194 8.410848 8.594049 8.752590 8.891087  
## 1986-01-03 7.649188 7.954302 8.209624 8.425203 8.608990 8.767272 8.905023  
## 1986-01-06 7.657082 7.964441 8.221224 8.437631 8.621727 8.779896 8.917188  
## 1986-01-07 7.637643 7.923323 8.163331 8.366719 8.540675 8.690901 8.821927  
## 1986-01-08 7.671395 7.982248 8.244705 8.467665 8.658297 8.822383 8.964592  
## 1986-01-09 7.846888 8.176396 8.443996 8.663364 8.845058 8.997235 9.126202  
## X8 X9 X10 X11 X12 X13 X14  
## 1986-01-02 9.013226 9.121945 9.219595 9.308054 9.388827 9.463122 9.531911  
## 1986-01-03 9.026180 9.133858 9.230526 9.318139 9.398248 9.472084 9.540628  
## 1986-01-06 9.037603 9.144309 9.239816 9.326115 9.404789 9.477094 9.544032  
## 1986-01-07 8.937355 9.040044 9.132273 9.215854 9.292234 9.362567 9.427775  
## 1986-01-08 9.088703 9.197777 9.294304 9.380306 9.457437 9.527047 9.590245  
## 1986-01-09 9.236843 9.332941 9.417432 9.492599 9.560214 9.621663 9.678025  
## X15 X16 X17 X18 X19 X20 X21  
## 1986-01-02 9.595976 9.655950 9.712346 9.765584 9.816007 9.863896 9.909488  
## 1986-01-03 9.604660 9.664802 9.721555 9.775321 9.826424 9.875129 9.921654  
## 1986-01-06 9.606400 9.664839 9.719862 9.771882 9.821233 9.868189 9.912974  
## 1986-01-07 9.488600 9.545635 9.599363 9.650170 9.698376 9.744238 9.787972  
## 1986-01-08 9.647942 9.700891 9.749716 9.794935 9.836982 9.876221 9.912958  
## 1986-01-09 9.730146 9.778689 9.824177 9.867024 9.907558 9.946043 9.982690  
## X22 X23 X24 X25 X26 X27 X28  
## 1986-01-02 9.952978 9.994532 10.03429 10.072377 10.108895 10.14394 10.17759  
## 1986-01-03 9.966176 10.008847 10.04979 10.089120 10.126923 10.16328 10.19828  
## 1986-01-06 9.955772 9.996740 10.03601 10.073683 10.109867 10.14464 10.17808  
## 1986-01-07 9.829755 9.869739 9.90805 9.944796 9.980074 10.01397 10.04654  
## 1986-01-08 9.947453 9.979928 10.01057 10.039549 10.067001 10.09305 10.11781  
## 1986-01-09 10.017671 10.051129 10.08318 10.113921 10.143438 10.17180 10.19908  
## X29 X30  
## 1986-01-02 10.20992 10.24100  
## 1986-01-03 10.23196 10.26441  
## 1986-01-06 10.21026 10.24122  
## 1986-01-07 10.07788 10.10802  
## 1986-01-08 10.14137 10.16382  
## 1986-01-09 10.22532 10.25058

Fit and plot the yield curve (NSS)

NSSyieldDF <- cbind(Dates,NSSyield) %>%   
 as.data.frame() %>% rename(Date = Dates) %>%  
 mutate(Date = as.Date(Date)) %>%   
 pivot\_longer(cols = -Date,  
 names\_to = 'Maturity',  
 values\_to = 'NSSEstimated') %>%   
 mutate(Maturity = as.numeric(gsub(pattern = 'X', replacement = '', x = Maturity)))  
  
  
  
GGData5 <- cbind(NSyieldDF$NSEstimated,NSSyieldDF$NSSEstimated,ObservedYield) %>%   
 as.data.frame() %>%   
 rename(NSEstimated = 'NSyieldDF$NSEstimated', NSSEstimated = 'NSSyieldDF$NSSEstimated')  
  
#NS Fit plot  
p1 <- GGData5 %>% filter(Date == '1986-01-02') %>%   
 ggplot(aes(x = Maturity, y = ObservedYield)) +  
 geom\_point(size = 1.2) +  
 geom\_line(mapping = aes(y = NSEstimated, colour = 'NS\_Fit'),size = 1) +  
 ggtitle('2nd Jan 1986 Yield Data with NS Model Fit') +  
 labs(x = 'Maturity', y = 'Spot Rate') +  
 theme(legend.position = 'bottom') +  
 scale\_colour\_manual(name = "Model Fit",  
 values = c(NS\_Fit = 'blue'))  
  
#NSS Fit plot  
p2 <- GGData5 %>% filter(Date == '1986-01-02') %>%   
 ggplot(aes(x = Maturity, y = ObservedYield)) +  
 geom\_point(size = 1.2) +  
 geom\_line(mapping = aes(y = NSSEstimated, colour = 'NSS\_Fit'),size = 1) +  
 ggtitle('2nd Jan 1986 Yield Data with NSS Model Fit') +  
 labs(x = 'Maturity', y = 'Spot Rate') +  
 theme(legend.position = 'bottom') +  
 scale\_colour\_manual(name = "Model Fit",  
 values = c(NSS\_Fit = 'red'))  
  
grid.arrange(p1,p2, nrow = 1, ncol = 2)

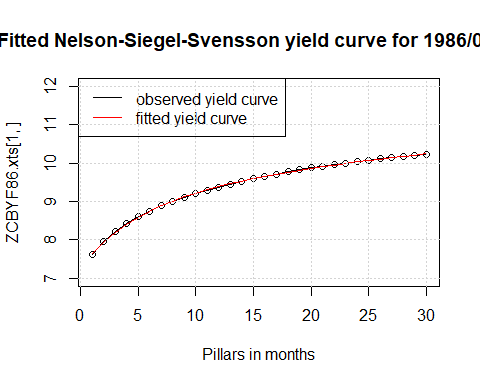


#Plot with both fit  
GGData5 %>% filter(Date == '1986-01-02') %>%   
 ggplot(aes(x = Maturity, y = ObservedYield)) +  
 geom\_point(size = 1.5) +  
 geom\_line(mapping = aes(y = NSEstimated, colour = 'NS\_Fit'),size = 1) +  
 geom\_line(mapping = aes(y = NSSEstimated, colour = 'NSS\_Fit'), size = 1) +  
 ggtitle('2 Jan 1986 Yield Data with NS and NSS Model Fit') +  
 labs(x = 'Maturity', y = 'Spot Rate') +  
 theme(legend.position = 'right') +  
 scale\_colour\_manual(name = "Model Fit",  
 values = c(NS\_Fit = 'blue', NSS\_Fit = 'red'))



#normal plot

y.2 = NSSyield[1,]  
plot(Maturity,ZCBYF86.xts[1,],main="Fitted Nelson-Siegel-Svensson yield curve for 1986/01/02",  
xlab=c("Pillars in months"), ylim=c(7,12), type="o")  
lines(Maturity,y.2, col=2)  
legend("topleft",legend=c("observed yield curve","fitted yield curve"),  
col=c(1,2),lty=1)  
grid()



Choosing best model based on MSE

# We calculate the MSE for each model using the entire data set.   
(NSmse= mean(((NSyield-ZCBYF86.xts)^2))) # 0.00229

## [1] 0.002289602

(NSSmse = mean(((NSSyield-ZCBYF86.xts)^2))) # 0.0000644, NSS model is better

## [1] 6.441476e-05

## Choosing best model based on BIC  
NSbic = NSParams[,7]  
NSSbic = NSSParams[,9]  
BICdata = cbind(NSbic,NSSbic)  
colnames(BICdata) = c('NS\_BIC', 'NSS\_BIC'); head(BICdata)

## Warning: timezone of object (UTC) is different than current timezone ().

## NS\_BIC NSS\_BIC  
## 1986-01-02 -114.1680 -169.3957  
## 1986-01-03 -106.1492 -160.0713  
## 1986-01-06 -108.8027 -168.6452  
## 1986-01-07 -114.3475 -176.2442  
## 1986-01-08 -148.3827 -218.0805  
## 1986-01-09 -109.0082 -187.7066

BICdata$NSScount = ifelse(BICdata$NS\_BIC>BICdata$NSS\_BIC,1,0); sum(BICdata$NSScount) # NSS was chosen 8399 out of 8650 times.

## [1] 8399

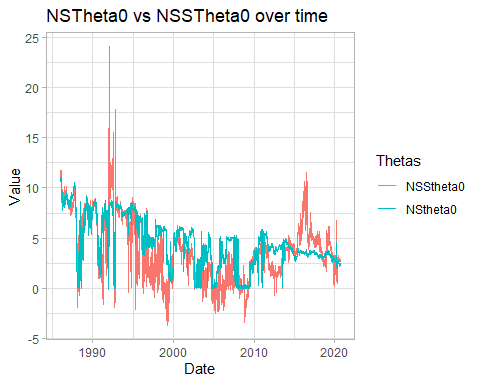
# 2.3) The changing patterns of the yield curve can be studied through the parameters tetha.

# What information can you extract from the estimates of tetha?

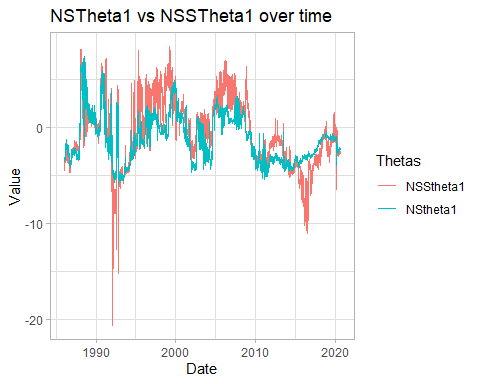
# NS Model  
NStheta0 = NSParams[,1]  
NStheta1 = NSParams[,2]  
NStheta3 = NSParams[,4]  
NStheta2 = NSParams[,3]\*NStheta3  
  
# NSS Model  
NSStheta0 = NSSParams[,1]   
NSStheta1 = NSSParams[,2]   
NSStheta3 = 1/NSSParams[,5]   
NSStheta2 = NSSParams[,3]\*NSStheta3   
NSStheta5 = 1/NSSParams[,6]   
NSStheta4 = NSSParams[,4]\*NSStheta5  
  
Thetas <- cbind(NStheta0,NStheta1,NStheta2,NStheta3,NSStheta0,NSStheta1,NSStheta2,NSStheta3,  
 NSStheta4,NSStheta5) %>%   
 as.data.frame()  
  
colnames(Thetas) <- c('NStheta0','NStheta1','NStheta2','NStheta3','NSStheta0','NSStheta1','NSStheta2',  
 'NSStheta3','NSStheta4','NSStheta5')  
ThetaDF <- cbind(matrix(rev(date)),Thetas) %>%   
 as.data.frame() %>% rename(Date = 'matrix(rev(date))') %>%  
 mutate(Date = as.Date(Date),  
 NS\_theta2theta3 = NStheta2/NStheta3,  
 NSS\_theta2theta3 = NSStheta2/NSStheta3,  
 NSS\_theta4theta5 = NSStheta4/NSStheta5) %>%   
 pivot\_longer(cols = -Date,  
 names\_to = 'Thetas',  
 values\_to = 'Value')

GGplot

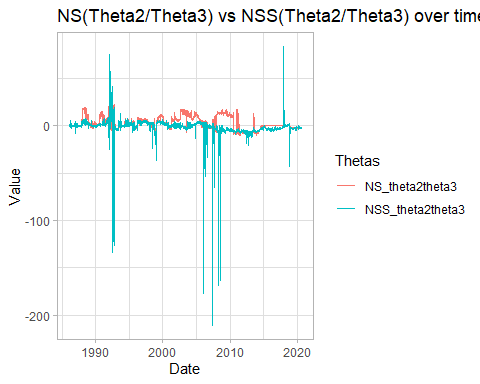
#Theta0 plot  
ThetaDF %>% filter(Thetas == 'NStheta0' | Thetas == 'NSStheta0') %>%   
 ggplot(aes(x = Date, y = Value, group = Thetas, colour = Thetas)) +   
 geom\_line() +   
 labs(title = 'NSTheta0 vs NSSTheta0 over time')



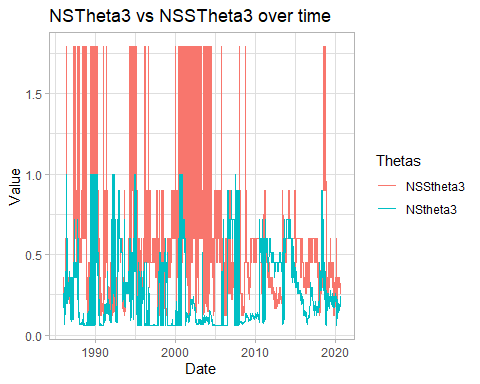
#Theta1 plot  
ThetaDF %>% filter(Thetas == 'NStheta1' | Thetas == 'NSStheta1') %>%   
 ggplot(aes(x = Date, y = Value, group = Thetas, colour = Thetas)) +   
 geom\_line() +   
 labs(title = 'NSTheta1 vs NSSTheta1 over time')



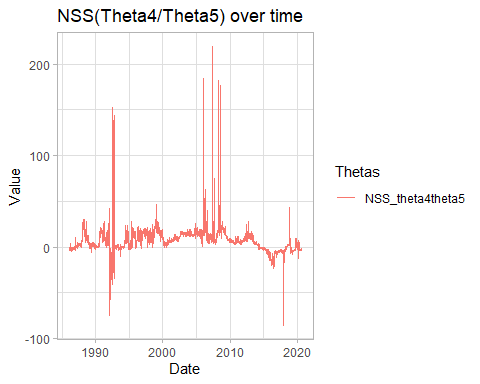
#Theta2/Theta 3 plot  
ThetaDF %>% filter(Thetas == 'NS\_theta2theta3' | Thetas == 'NSS\_theta2theta3') %>%   
 ggplot(aes(x = Date, y = Value, group = Thetas, colour = Thetas)) +   
 geom\_line() +   
 labs(title = 'NS(Theta2/Theta3) vs NSS(Theta2/Theta3) over time')



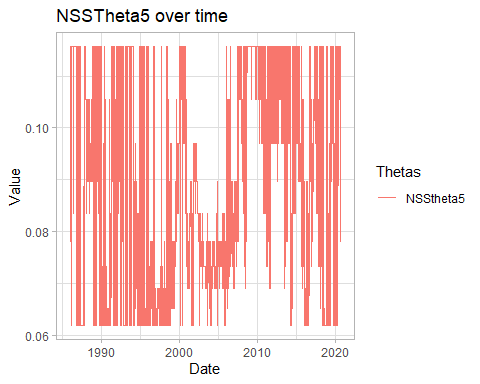
#Theta3  
ThetaDF %>% filter(Thetas == 'NStheta3' | Thetas == 'NSStheta3') %>%   
 ggplot(aes(x = Date, y = Value, group = Thetas, colour = Thetas)) +   
 geom\_line() +   
 labs(title = 'NSTheta3 vs NSSTheta3 over time')



#Theta4/Theta5 (only NSS)  
ThetaDF %>% filter(Thetas == 'NSS\_theta4theta5') %>%   
 ggplot(aes(x = Date, y = Value, colour = Thetas)) +   
 geom\_line() +   
 labs(title = 'NSS(Theta4/Theta5) over time')



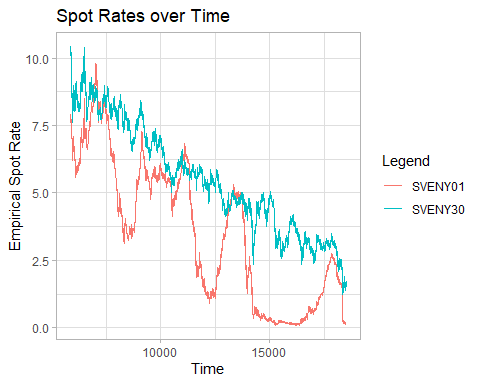
#Theta5 (only NSS)  
ThetaDF %>% filter(Thetas == 'NSStheta5') %>%   
 ggplot(aes(x = Date, y = Value, colour = Thetas)) +   
 geom\_line() +   
 labs(title = 'NSSTheta5 over time')

 Tetha0 = level (long term component) Tetha1 = Slope (short term component) Tetha2/Tetha3 = Curvature (medium term component 1) Tetha4/Tetha5 = Curvature (medium term component 2) Tetha3 = rate of decay for first medium term Tetha5 = rate of decay for second medium term

# 2.4) How may the data tell you about the response of the spot rates at the long end

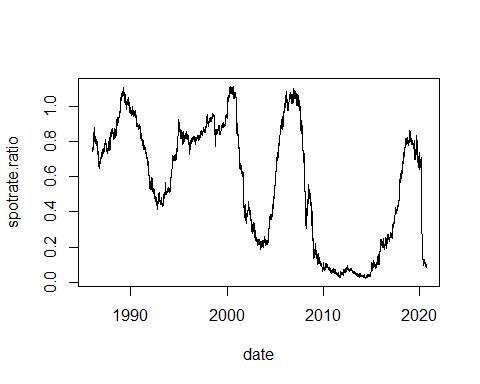
# with respect to the spot rates at the short end?

# comparing spot rates of 1 yr and 30 yr maturity  
ZCBYF86 <- as.data.frame(ZCBYF86)  
GGData6 <- cbind(date, ZCBYF86$SVENY01, ZCBYF86$SVENY30) %>% as.data.frame()  
colnames(GGData6) <- c("date", "SVENY01", "SVENY30")  
GGData6 %>%   
 pivot\_longer(cols = -date,   
 names\_to="maturity",  
 values\_to = "value") %>%   
 ggplot(aes(x = date)) +   
 geom\_line(aes(y = value, col = maturity)) +   
 ggtitle('Spot Rates over Time') +  
 labs(x = 'Time', y = 'Empirical Spot Rate', color="Legend") +  
 theme(legend.position = 'right')

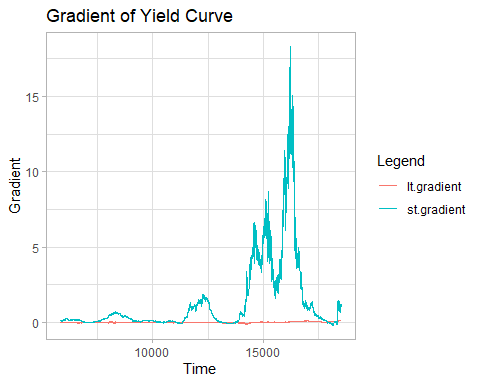


Other plots that were not included in the final report

# ratio of 1 yr maturity to 30 yr maturity spot rates  
spotrate.ratio <- ZCBYF86$SVENY01/ZCBYF86$SVENY30  
ZCBYF86.ratio <- cbind(ZCBYF86,spotrate.ratio)  
plot(date, spotrate.ratio, type = "l") # idk what to infer from this tho LOL



# gradient of yield curve for 1 yr vs 30 yr maturities  
st.gradient <- NULL  
lt.gradient <- NULL  
for (i in 1:nrow(ZCBYF86)){  
 st.gradient[i] <- (ZCBYF86$SVENY05[i] - ZCBYF86$SVENY01[i])/ZCBYF86$SVENY01[i]  
}  
for (i in 1:nrow(ZCBYF86)){  
 lt.gradient[i] <- (ZCBYF86$SVENY30[i] - ZCBYF86$SVENY25[i])/ZCBYF86$SVENY25[i]  
}  
ZCBYF86.gradient <- cbind(date, st.gradient, lt.gradient)  
ZCBYF86.gradient %>% as.data.frame() %>%   
 gather(key="term", value="value",-date) %>%  
 ggplot(aes(x=date)) +  
 geom\_line(aes(y=value, col = term)) +   
 ggtitle('Gradient of Yield Curve') +  
 labs(x = 'Time', y = 'Gradient', color = "Legend") +  
 theme(legend.position = 'right')

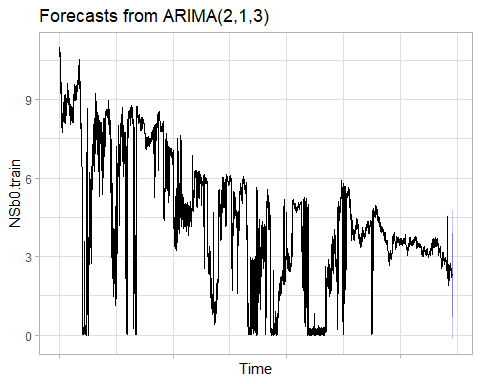


# 2.5) Forecasting and MSFE

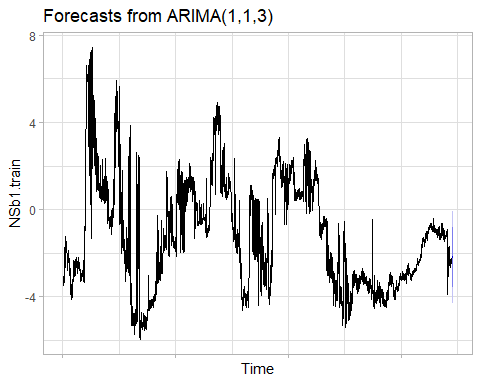
Choosing best model based on MSFE: 1) Split Dataset into train and testing dataset (test set is August 2020 only, h = 20) 2) For the training dataset, fit the NS and NSS models and get a dataframe of the parameters (We can use the objects from the ftting that have already been created from above and do step 1 here.)

#NS Model Load Parameters

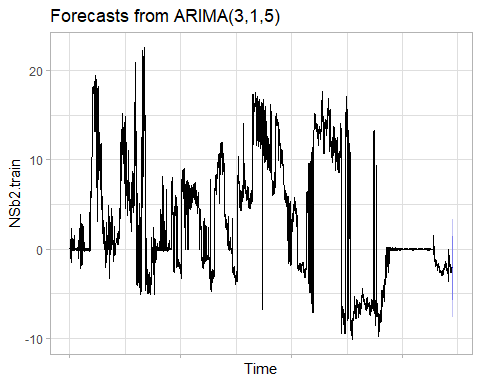
h=20  
NSb0 = NSParams[,1]  
NSb1 = NSParams[,2]  
NSb2 = NSParams[,3]  
NSlambda1 = NSParams[,4]  
  
NSb0.ts = ts(NSb0, start=c(1986,1), frequency=252)  
NSb0.train = NSb0.ts[1:(8650-h),]  
  
NSb1.ts = ts(NSb1, start=c(1986,1), frequency=252)  
NSb1.train = NSb1.ts[1:(8650-h),]  
  
NSb2.ts = ts(NSb2, start=c(1986,1), frequency=252)  
NSb2.train = NSb2.ts[1:(8650-h),]  
  
NSlambda1.ts = ts(NSlambda1, start=c(1986,1), frequency=252)  
NSlambda1.train = NSlambda1.ts[1:(8650-h),]  
  
# 3) Fit an ARIMA model to these parameters individually  
# auto.arima(NSlambda1.train, ic="bic") # Unhash to run; Find best arima model based on BIC  
NSb0.arima = Arima(NSb0.train,  
 order=c(2,1,3),  
 )  
NSb1.arima = Arima(NSb1.train,  
 order=c(1,1,3),  
 )  
NSb2.arima = Arima(NSb2.train,  
 order=c(3,1,5),  
 )  
NSlambda1.arima = Arima(NSlambda1.train,  
 order=c(1,1,2),  
 )  
# 4) Use the Arima model in step 3 to predict the parameters for the test set  
NSb0.forecast = forecast(NSb0.arima, h=h)  
autoplot(NSb0.forecast) + theme(axis.text.x=element\_blank())



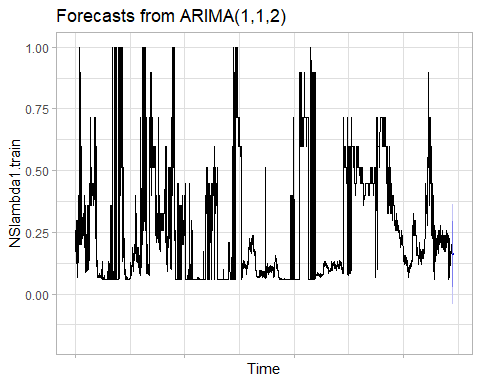
NSb1.forecast = forecast(NSb1.arima, h=h)  
autoplot(NSb1.forecast) + theme(axis.text.x=element\_blank())



NSb2.forecast = forecast(NSb2.arima, h=h)  
autoplot(NSb2.forecast) + theme(axis.text.x=element\_blank())



NSlambda1.forecast = forecast(NSlambda1.arima, h=h)  
autoplot(NSlambda1.forecast) + theme(axis.text.x=element\_blank())



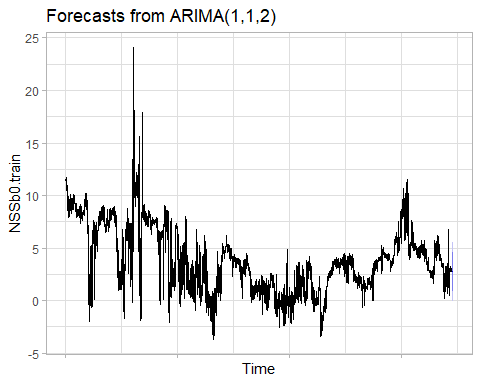
# 5) Using the predicted parameters in 4), get the NS and NSS equations for each observation in the test set.  
paramsNS = cbind(NSb0.forecast$mean,NSb1.forecast$mean,NSb2.forecast$mean,NSlambda1.forecast$mean); head(paramsNS)

## Time Series:  
## Start = 8631   
## End = 8636   
## Frequency = 1   
## NSb0.forecast$mean NSb1.forecast$mean NSb2.forecast$mean  
## 8631 2.233389 -2.131832 -2.094085  
## 8632 2.241851 -2.139460 -2.088540  
## 8633 2.248841 -2.144929 -2.091947  
## 8634 2.255623 -2.149842 -2.082903  
## 8635 2.261474 -2.154253 -2.085548  
## 8636 2.266971 -2.158215 -2.085252  
## NSlambda1.forecast$mean  
## 8631 0.1618160  
## 8632 0.1620160  
## 8633 0.1621825  
## 8634 0.1623210  
## 8635 0.1624364  
## 8636 0.1625324

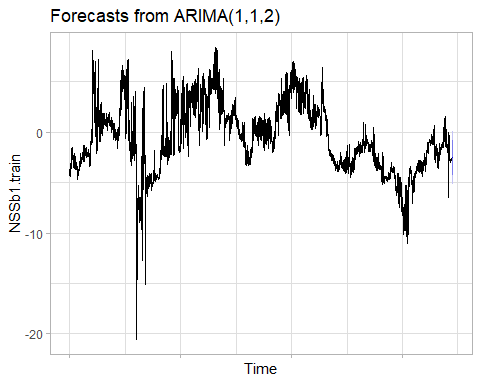
# 6) Use the NS and NSS equations in 5) to get the spot rates for each ZCB  
NSPredicted = matrix(nrow=h,ncol=30)  
  
# For all the bonds (double for loop - summing cross sectionally first, then summing across time) - can use NSrate() function instead !!!!  
for(i in 1:h)  
{  
 for(j in 1:30)  
 {  
 NSPredicted[i,j] = paramsNS[i,1]+  
 paramsNS[i,2]\*((1-exp(-paramsNS[i,4]\*j))/(paramsNS[i,4]\*j))+  
 paramsNS[i,3]\*(((1-exp(-paramsNS[i,4]\*j))/(paramsNS[i,4]\*j))-exp(-paramsNS[i,4]\*j))  
 }  
}  
# NSrates(params.xts, maturity) # alternative way to predict (faster LOL)  
ZCBYF86.test = as.matrix(ZCBYF86.xts[(8650-h+1):8650,])

#NSS Model

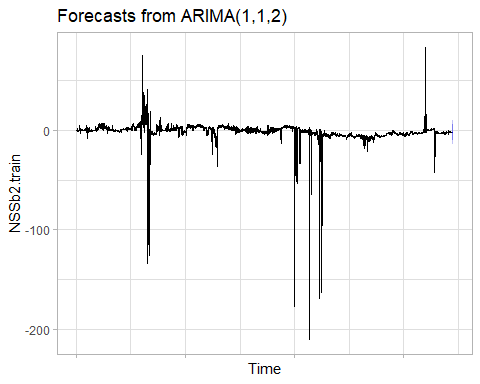
NSSb0 = NSSParams[,1]  
NSSb1 = NSSParams[,2]  
NSSb2 = NSSParams[,3]  
NSSb3 = NSSParams[,4]  
NSStau1 = NSSParams[,5]  
NSStau2 = NSSParams[,6]  
  
NSSb0.ts = ts(NSSb0, start=c(1986,1), frequency=252)  
NSSb0.train = NSSb0.ts[1:(8650-h),]  
  
NSSb1.ts = ts(NSSb1, start=c(1986,1), frequency=252)  
NSSb1.train = NSSb1.ts[1:(8650-h),]  
  
NSSb2.ts = ts(NSSb2, start=c(1986,1), frequency=252)  
NSSb2.train = NSSb2.ts[1:(8650-h),]  
  
NSSb3.ts = ts(NSSb3, start=c(1986,1), frequency=252)  
NSSb3.train = NSSb3.ts[1:(8650-h),]  
  
NSStau1.ts = ts(NSStau1, start=c(1986,1), frequency=252)  
NSStau1.train = NSStau1.ts[1:(8650-h),]  
  
NSStau2.ts = ts(NSStau2, start=c(1986,1), frequency=252)  
NSStau2.train = NSStau2.ts[1:(8650-h),]  
  
# 3) Fit an ARIMA model to these parameters individually  
# auto.arima(NSStau2.train, ic="bic") # Find best arima model based on BIC  
NSSb0.arima = Arima(NSSb0.train,  
 order=c(1,1,2),  
 )  
NSSb1.arima = Arima(NSSb1.train,  
 order=c(1,1,2),  
 )  
NSSb2.arima = Arima(NSSb2.train,  
 order=c(1,1,2),  
 )  
NSSb3.arima = Arima(NSSb3.train,  
 order=c(2,1,2),  
 )  
NSStau1.arima = Arima(NSStau1.train,  
 order=c(1,1,2),  
 )  
NSStau2.arima = Arima(NSStau2.train,  
 order=c(1,1,2),  
 )  
  
# 4) Use the Arima model in step 3 to predict the parameters for the test set  
NSSb0.forecast = forecast(NSSb0.arima, h=h)  
autoplot(NSSb0.forecast) + theme(axis.text.x=element\_blank())



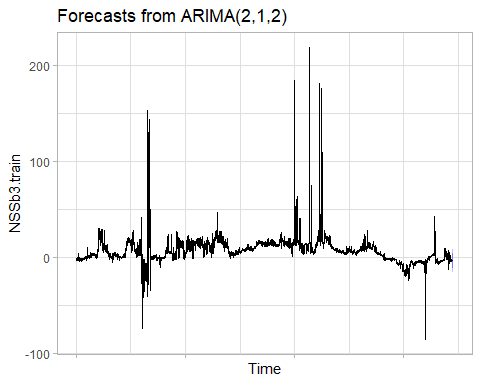
NSSb1.forecast = forecast(NSSb1.arima, h=h)  
autoplot(NSSb1.forecast) + theme(axis.text.x=element\_blank())



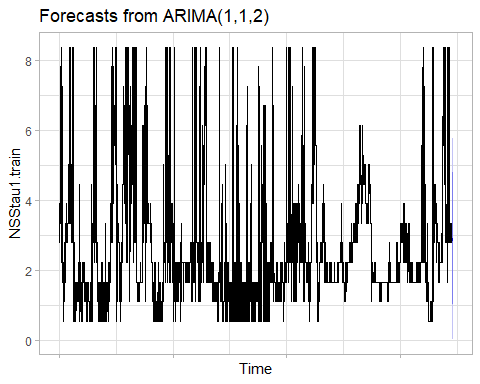
NSSb2.forecast = forecast(NSSb2.arima, h=h)  
autoplot(NSSb2.forecast) + theme(axis.text.x=element\_blank())



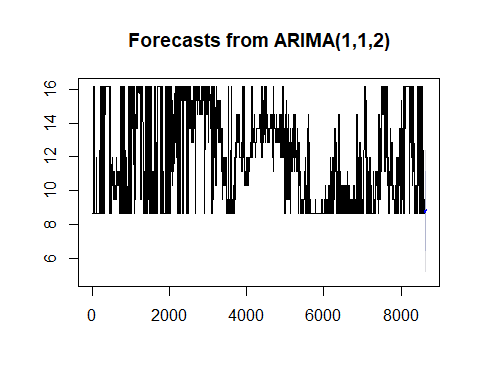
NSSb3.forecast = forecast(NSSb3.arima, h=h)  
autoplot(NSSb3.forecast) + theme(axis.text.x=element\_blank())



NSStau1.forecast = forecast(NSStau1.arima, h=h)  
autoplot(NSStau1.forecast) + theme(axis.text.x=element\_blank())



NSStau2.forecast = forecast(NSStau2.arima, h=h)  
plot(NSStau2.forecast) + theme(axis.text.x=element\_blank())



## NULL

# 5) Using the predicted parameters in 4), get the NS and NSS equations for each observation in the test set.  
paramsNSS = cbind(NSSb0.forecast$mean,NSSb1.forecast$mean,NSSb2.forecast$mean,NSSb3.forecast$mean,NSStau1.forecast$mean,NSStau2.forecast$mean)  
  
# 6) Use the NS and NSS equations in 5) to get the spot rates for each ZCB (use SRates() function instead!!!)  
NSSPredicted = matrix(nrow=h,ncol=30)  
  
for(i in 1:h)  
{  
 for(j in 1:30)  
 {  
 NSSPredicted[i,j] = paramsNSS[i,1]+  
 paramsNSS[i,2]\*((1-exp(-j\*(1/paramsNSS[i,5])))/(j\*(1/paramsNSS[i,5])))+  
 paramsNSS[i,3]\*(((1-exp(-j\*(1/paramsNSS[i,5])))/(j\*(1/paramsNSS[i,5])))-exp(-j\*(1/paramsNSS[i,5])))+  
 paramsNSS[i,4]\*(((1-exp(-j\*(1/paramsNSS[i,6])))/(j\*(1/paramsNSS[i,6])))-exp(-j\*(1/paramsNSS[i,6])))  
 }  
}  
# can use SRates() function instead!  
head(NSSPredicted)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.1441568 0.1404559 0.1706684 0.2204270 0.2805922 0.3454697 0.4116111  
## [2,] 0.1488716 0.1478477 0.1801519 0.2316520 0.2933332 0.3595634 0.4269233  
## [3,] 0.1523614 0.1532376 0.1870204 0.2397488 0.3024994 0.3696863 0.4379120  
## [4,] 0.1553656 0.1579374 0.1930593 0.2469021 0.3106158 0.3786538 0.4476391  
## [5,] 0.1580382 0.1621881 0.1985740 0.2534713 0.3180899 0.3869180 0.4565986  
## [6,] 0.1604274 0.1660447 0.2036190 0.2595092 0.3249755 0.3945368 0.4648554  
## [,8] [,9] [,10] [,11] [,12] [,13] [,14]  
## [1,] 0.4770133 0.5405871 0.6018084 0.6604913 0.7166430 0.7703727 0.8218356  
## [2,] 0.4934223 0.5579768 0.6200658 0.6795068 0.7363115 0.7905949 0.8425189  
## [3,] 0.5051952 0.5704560 0.6331756 0.6931733 0.7504637 0.8051650 0.8574438  
## [4,] 0.5156013 0.5814667 0.6447205 0.7051860 0.7628813 0.8179289 0.8704999  
## [5,] 0.5251735 0.5915769 0.6553002 0.7161717 0.7742145 0.8295561 0.8823721  
## [6,] 0.5339857 0.6008709 0.6650096 0.7262364 0.7845799 0.8401727 0.8931956  
## [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]  
## [1,] 0.8712001 0.9186298 0.9642738 1.008264 1.050712 1.091715 1.131354 1.169697  
## [2,] 0.8922597 0.9399887 0.9858632 1.030022 1.072587 1.113660 1.153328 1.191667  
## [3,] 0.9074809 0.9554530 1.0015228 1.045835 1.088514 1.129669 1.169392 1.207760  
## [4,] 0.9207797 0.9689499 1.0151777 1.059612 1.102383 1.143602 1.183365 1.221753  
## [5,] 0.9328530 0.9811846 1.0275388 1.072069 1.114908 1.156172 1.195959 1.234354  
## [6,] 0.9438436 0.9923071 1.0387619 1.083365 1.126254 1.167547 1.207345 1.245737  
## [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30]  
## [1,] 1.206802 1.242718 1.277489 1.311153 1.343744 1.375295 1.405835 1.435392  
## [2,] 1.228738 1.264597 1.299291 1.332862 1.365347 1.396781 1.427196 1.456624  
## [3,] 1.244840 1.280690 1.315360 1.348895 1.381335 1.412715 1.443071 1.472433  
## [4,] 1.258837 1.294675 1.329322 1.362823 1.395220 1.426552 1.456853 1.486157  
## [5,] 1.271430 1.307248 1.341864 1.375326 1.407676 1.438955 1.469200 1.498443  
## [6,] 1.282795 1.318586 1.353165 1.386582 1.418883 1.450107 1.480292 1.509473

# 7) Compare the predicted spot rate with the actual spot rate and get MSFE for each model:

(NSmsfe = mean(((NSPredicted-ZCBYF86.test)^2))) # 0.009869873

## [1] 0.009869873

(NSSmsfe = mean(((NSSPredicted-ZCBYF86.test)^2))) # 0.00351771, NSS model better

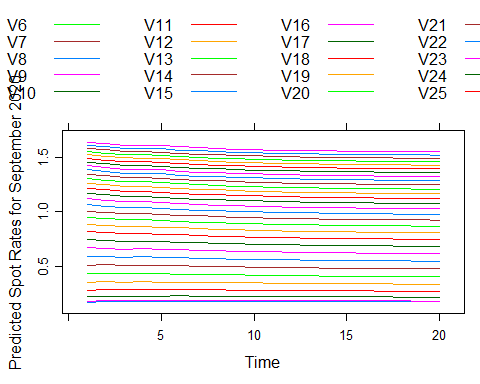
## [1] 0.003517712

# 2.6) Forecasting for September 2020

NSSb0\_ts = ts(NSSb0, start=c(1986,1), frequency=252)  
NSSb1\_ts = ts(NSSb1, start=c(1986,1), frequency=252)  
NSSb2\_ts = ts(NSSb2, start=c(1986,1), frequency=252)  
NSSb3\_ts = ts(NSSb3, start=c(1986,1), frequency=252)  
NSStau1\_ts = ts(NSStau1, start=c(1986,1), frequency=252)  
NSStau2\_ts = ts(NSStau2, start=c(1986,1), frequency=252)  
  
# auto.arima(NSSb1.ts, ic="bic") # Find best arima model based on BIC  
NSSb0.pred = Arima(NSSb0\_ts,  
 order=c(1,1,2),  
 )  
NSSb1.pred = Arima(NSSb1\_ts,  
 order=c(1,1,2),  
 )  
NSSb2.pred = Arima(NSSb2\_ts,  
 order=c(5,1,2),  
 )  
NSSb3.pred = Arima(NSSb3\_ts,  
 order=c(5,1,1),  
 )  
NSStau1.pred = Arima(NSStau1\_ts,  
 order=c(5,1,2),  
 )  
NSStau2.pred = Arima(NSStau2\_ts,  
 order=c(1,1,2),  
 )  
  
NSSb0\_forecast = forecast(NSSb0.pred, h=h)  
NSSb1\_forecast = forecast(NSSb1.pred, h=h)  
NSSb2\_forecast = forecast(NSSb2.pred, h=h)  
NSSb3\_forecast = forecast(NSSb3.pred, h=h)  
NSStau1\_forecast = forecast(NSStau1.pred, h=h)  
NSStau2\_forecast = forecast(NSStau2.pred, h=h)  
  
paramsPred = cbind(NSSb0\_forecast$mean,NSSb1\_forecast$mean,NSSb2\_forecast$mean,NSSb3\_forecast$mean,NSStau1\_forecast$mean,NSStau2\_forecast$mean)  
  
NSSPredict = matrix(nrow=h,ncol=30)  
  
for(i in 1:h)  
{  
 for(j in 1:30)  
 {  
 NSSPredict[i,j] = paramsPred[i,1]+  
 paramsPred[i,2]\*((1-exp(-j\*(1/paramsPred[i,5])))/(j\*(1/paramsPred[i,5])))+  
 paramsPred[i,3]\*(((1-exp(-j\*(1/paramsPred[i,5])))/(j\*(1/paramsPred[i,5])))-exp(-j\*(1/paramsPred[i,5])))+  
 paramsPred[i,4]\*(((1-exp(-j\*(1/paramsPred[i,6])))/(j\*(1/paramsPred[i,6])))-exp(-j\*(1/paramsPred[i,6])))  
 }  
}  
View(NSSPredict)

Other plot that was not included in Final report - plotting predicted spot rates for each maturity over time

nss.predicted.spotrate.ts = ts(NSSPredict)  
nss.predicted.spotrate.xts = ts2xts(NSSPredict)  
par(mar=c(1,1,1,1))  
xyplot.ts(NSSPredict,superpose=TRUE,auto.key=list(columns=6), ylab="Predicted Spot Rates for September 2020")

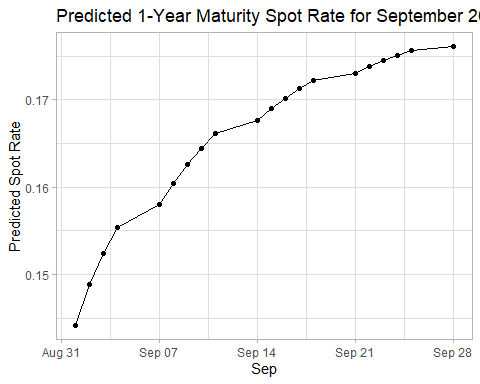


Plotting

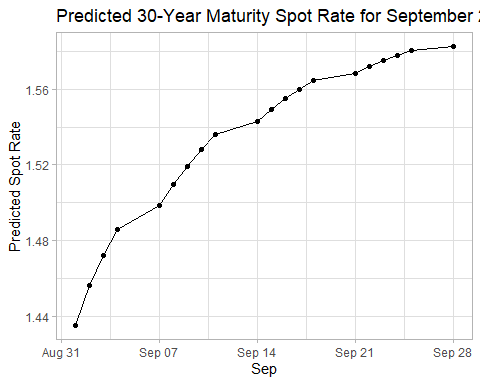
Sep <- seq(as.Date('2020-09-01'),as.Date('2020-09-30'),by = 1)  
Sep <- Sep[!weekdays(Sep) %in% c('Saturday','Sunday')]  
Sep <- Sep[1:20]  
  
GGDataForecast <- cbind(NSSPredicted,Sep) %>% as.data.frame() %>% mutate(Sep = as.Date(Sep)) %>% pivot\_longer(cols = -Sep, names\_to = 'Maturity', values\_to = 'SpotRateValue') %>% mutate(Maturity = as.numeric(gsub(x = Maturity, pattern = 'V', replacement = '')))  
head(GGDataForecast)

## # A tibble: 6 x 3  
## Sep Maturity SpotRateValue  
## <date> <dbl> <dbl>  
## 1 2020-09-01 1 0.144  
## 2 2020-09-01 2 0.140  
## 3 2020-09-01 3 0.171  
## 4 2020-09-01 4 0.220  
## 5 2020-09-01 5 0.281  
## 6 2020-09-01 6 0.345

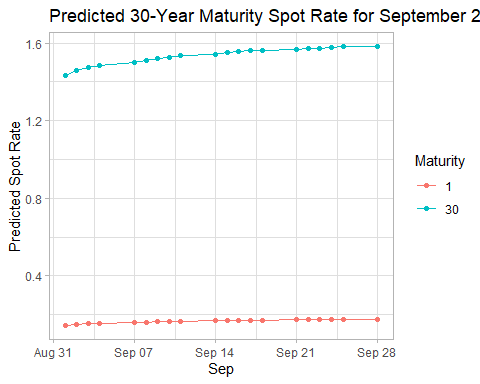
#plot by maturity  
GGDataForecast %>% filter(Maturity == 1) %>%   
 ggplot(aes(x=Sep, y = SpotRateValue)) +   
 geom\_point() + geom\_line() +   
 labs(title = 'Predicted 1-Year Maturity Spot Rate for September 2020', y = 'Predicted Spot Rate')



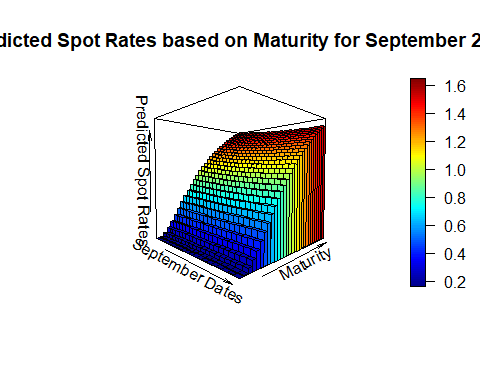
GGDataForecast %>% filter(Maturity == 30) %>%   
 ggplot(aes(x=Sep, y = SpotRateValue)) +   
 geom\_point() + geom\_line() +   
 labs(title = 'Predicted 30-Year Maturity Spot Rate for September 2020', y = 'Predicted Spot Rate')



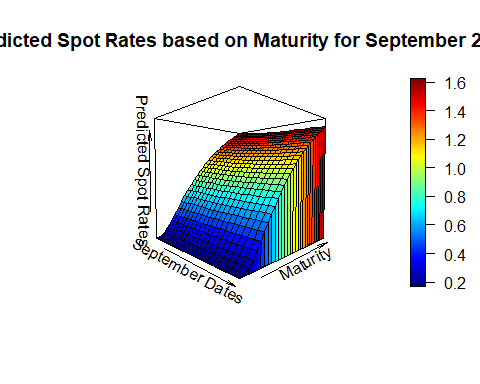
#plot both  
GGDataForecast %>% filter(Maturity == 1 | Maturity == 30) %>%   
 mutate(Maturity = as.factor(Maturity)) %>%   
 ggplot(aes(x = Sep, y = SpotRateValue, group = Maturity, colour = Maturity)) +   
 geom\_point() + geom\_line() +   
 labs(title = 'Predicted 30-Year Maturity Spot Rate for September 2020', y = 'Predicted Spot Rate')



hist3D(x = c(1:20), y = c(1:30), z = NSSPredict, xlab = 'September Dates', ylab = 'Maturity', zlab = 'Predicted Spot Rates', main = 'Predicted Spot Rates based on Maturity for September 2020', theta = 45, phi = 2, border = 'black', scale = TRUE)



persp3D(x = c(1:20), y = c(1:30), z = NSSPredict, xlab = 'September Dates', ylab = 'Maturity', zlab = 'Predicted Spot Rates', theta = 45, phi = 2, border = 'black', main = 'Predicted Spot Rates based on Maturity for September 2020', scale = TRUE, curtain = TRUE)



plotrgl() #run this line to generate an interactive 3D plot