Bond Value - Severly Adverse Scenario

Jingze Sun

11/14/2020

1. R Studio Environment Setup:

```
# Clear work space:
rm(list = ls())

# Import required packages:
library(data.table)
library(readxl)
library(pracma)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x purrr::cross()     masks pracma::cross()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```
library(data.table)
library(zoo)
```

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
```

```
##     yday, year
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(jrvFinance)
```

```
##
## Attaching package: 'jrvFinance'
## The following object is masked from 'package:lubridate':
##
##     duration
```

2. Regression Analysis: Mortgage Rate Spreads:

```
# Import the historic 30-year treasury rates data:
UST_30 <- read_xlsx("WRDS_UST_30.xlsx")
UST_30 <- as.data.table(UST_30)
UST_30 <- UST_30[,c(2,3)]
colnames(UST_30) <- c("Date","UST_30_r")
UST_30[,Date:=as.Date(Date)]

# Correct the dates:
month_end_list <- seq(as.Date("2008-02-01"),length=nrow(UST_30),by="months")-1
UST_30$Date <- month_end_list

# Import the domestic historic data:
his_dom <- fread("Historic_Domestic.csv")
his_dom <- his_dom[,c(2,13)]
colnames(his_dom) <- c("Quart","Mort_r")
his_dom[,Date:=as.Date(as.yearqtr(Quart), frac=1)] # Note: "frac=1" converts
                                                   # date to end-of-month
his_dom <- his_dom[,c(3,2)]

# Combine mortgage rates data and 30-year treasury rates data:
Mort_UST <- UST_30 %>% inner_join(his_dom, by="Date") %>% as.data.table()

# Compute the mortgage rate spread:
Mort_UST[,Mort_Spread:=Mort_r-UST_30_r]
# Get the lag mortgage rate spread column:
Mort_UST[,lag_Mort_Spread:=shift(Mort_Spread)]

# Prepare data for regression analysis:
Mort_UST <- drop_na(Mort_UST[,c(4,5)])

# Run the regression, then get the intercept and correlation coefficient:
reg <- lm(Mort_Spread ~ lag_Mort_Spread, data=Mort_UST)
reg_int <- reg$coefficients[[1]]
reg_coef <- reg$coefficients[[2]]
```

3. Scenario Data:

1) Set up severely adverse domestic scenario data table:

```
# Import the baseline domestic scenario data:
scen <- fread("Table_3A_Supervisory_Severely_Adverse_Domestic.csv")

# Set the maximum number of predicted quarters:
max_Q_len <- 4

# Tidy the scenario data:
scen <- scen[1:max_Q_len,c(2,9,10,11,13)]
colnames(scen) <- c("Quart","UST_3m","UST_5yr","UST_10yr","Mort_r")
scen[,Date:=as.Date(as.yearqtr(Quart), frac=1)]
scen <- scen[,c(6,2:5)]
```

2) Calculate predicated severely adverse domestic scenario data:

```
# Create a copy of the cleaned scenario data:
scen_up <- scen

# Compute the predicted mortgage rate spreads:
scen_up[,Mort_Spread:=NA]
ini_base_spread <- Mort_UST[nrow(Mort_UST),lag_Mort_Spread]
scen_up$Mort_Spread[1] <- reg_int+reg_coef*ini_base_spread

for (i in 2:nrow(scen_up)) {
  scen_up$Mort_Spread[i] <- scen_up$Mort_Spread[i-1]*reg_coef + reg_int
}

# Compute predicated 30-year Treasury rates:
scen_up[,Pred_UST_30yr:=Mort_r-Mort_Spread]
# Condense the updated scenario data table:
scen_up <- scen_up[,c(1:4,7)]
# Change the data table to a dataframe so that we can run our function later:
scen_up <- as.data.frame(scen_up)
```

4. Yield Curve Calculation:

1) Matching Yield Table Function:

```
# Write a function that, with a given pricing date and a scenario data set,
# returns a table of matched UST rates:
mat_yield_tab <- function(pricing_date, scenario) {
  yield_table <- data.frame(matrix(ncol=2, nrow=4))
  colnames(yield_table) <- c("t","Mat_Yield_r")
  t_list_eg <- c(0.25,5,10,30)
  yield_table$t <- t_list_eg

  for (i in 1:nrow(yield_table)) {
    yield_table$Mat_Yield_r[i] <- scenario[scenario$Date==pricing_date,i+1]
  }
```

```
  return(yield_table)
}
```

2) Cubic-Spline Yield Curve Function:

```
# Write a function that, with a given list of points in time, and a list of
# matched UST rates, returns a table of cubic-spline fitted yield rates for
# all 60 points in time on the yield curve:
CS_yield_SR <- function(t_list, matched_YR, Min_Yr=0.5, Max_Yr=30, Comp=0.5) {
  xi_list <- seq(Min_Yr,Max_Yr,by=Comp)
  years_pt <- t_list
  rates_pt <- matched_YR

  # Get the cubic-spline fitted yield curve using the imported function:
  CS_output <- cubicspline(x=years_pt, y=rates_pt, xi=xi_list)

  # Initialize the table of cubic-spline fitted yields:
  CS_spot_table <- data.frame(matrix(ncol=2, nrow=length(xi_list)))
  colnames(CS_spot_table) <- c("t","YR")
  CS_spot_table$t <- xi_list
  CS_spot_table$YR <- CS_output
  return(CS_spot_table)
}
```

5. Boostrap Spot Rate Function:

```
# Write a function that, with the given CS-fitted Yield Table, returns the
# bootstrap spot rates:
Bootstrap_SR <- function(Yield_Table) {
  # Ensure that the yield table is in the data table format:
  Yield_Table <- as.data.table(Yield_Table)

  # Create the spot rate column:
  Yield_Table$Spot_Rate <- NA

  # 1. Set the first two spot rates to be equal to the Yields on Par Bonds:
  for (i in 1:2) {
    Yield_Table$Spot_Rate[i] = Yield_Table$YR[i]
  }

  # 2. Compute the rest of the spot rates:
  for (i in 3:nrow(Yield_Table)) {
    # Compute the sum of the values of previous discounted coupon payments:
    # i. Initialize a list of coupon payments:
    coupon_list <- rep(x=Yield_Table$YR[i], times=i-1)
    dis_coupon_list <- rep(x=0, times=i-1)
    # ii. Compute the discounted value of each coupon payments
    for (j in 1:(i-1)) {
      SR <- Yield_Table$Spot_Rate[j]
      TTM <- Yield_Table$t[j]
      dis_coupon_list[j] <- (coupon_list[j]/2)*((200/(200+SR))^(2*TTM))
    }
```

```
    # iii. Get the sum of the values of previous discounted coupon payments:
    dis_cou_sum <- sum(dis_coupon_list)

    # With the resultant sum, we can get the spot rate:
    A <- Yield_Table$YR[i]/2
    B <- 2*Yield_Table$t[i]
    Yield_Table$Spot_Rate[i] <- 200*(((100+A)/(100-dis_cou_sum))^(1/B)-1)
    # Note: The spot rate formula is derived through algebra.
  }
  return(Yield_Table)
}
```

6. DCF Computation Function:

```
# Write a function that returns a data table of discounted cashflows, given
# the pricing date, the bond's issue date, maturity date, and coupon rate, and
# the bootstrap spot rate table. The bootstrap spot rate table must be a result
# from the Bootstrap_SR function.
DCF_table <- function(pricing_date,issue_date,mat_date,coupon,BS_SR_Table) {
  # 1. Create the coupon payment time table of the bond:
  # i) Get the number of rows of the coupon payment time table:
  n_date_rows <- 2*round(as.numeric(mat_date - issue_date)/365)

  # ii) Build the inital structure of coupon payment time table:
  Coupon_Table <- data.frame(matrix(ncol=2, nrow=n_date_rows))
  colnames(Coupon_Table) <- c("Coupon_Date", "Coupon")
  Coupon_Table <- as.data.table(Coupon_Table)

  # iii) Get the coupon payments column of the table:
  bond_coupon <- coupon/2 # Note: coupon rate is semi-annual rate.
  bond_coupon_list <- rep(x=bond_coupon, times=n_date_rows)
  Coupon_Table$Coupon <- bond_coupon_list
  # Account for the principal payment at the maturity date:
  Coupon_Table$Coupon[n_date_rows] <- Coupon_Table$Coupon[n_date_rows]+100

  # iv) Get the coupon payment dates column:
  coupon_date_list <- rep(x=issue_date, times=n_date_rows+1)

  for (i in 2:length(coupon_date_list)) {
    coupon_date_list[i] <- coupon_date_list[i-1] %m+% months(6)
  }
  coupon_date_list <- coupon_date_list[2:length(coupon_date_list)]
  Coupon_Table$Coupon_Date <- coupon_date_list

  # 2. Get the time to pricing date column in the Coupon_Table:
  Coupon_Table[,tPD:=as.numeric(Coupon_Date-pricing_date)/365]

  # 3. Get the Cubic-Spline fitted spot rates in the Coupon_Table:
  # i) Set up the inputs:
  t_pts <- BS_SR_Table$t
  SR_pts <- BS_SR_Table$Spot_Rate
  xi_list <- Coupon_Table$tPD
```

```
  # ii) Get the cubic-spline fitted yield curve using the imported function:
  fitted_SR <- cubicspline(x=t_pts, y=SR_pts, xi=xi_list)

  # iii) Add the fitted spot rate column to the Coupon Table:
  Coupon_Table$Spot_Rate <- fitted_SR

  # 4. Compute the DCF column:
  Coupon_Table[,DCF:=ifelse(tPD<0,0,Coupon/((1+Spot_Rate/200)^(2*tPD)))]

  # 5. Return the output DCF table:
  return(Coupon_Table)
}
```

7. Bond Price Computation Function:

```
# Using the following functions: 1) mat_yield_tab; 2) CS_yield_SR;
# 3) Bootstrap_SR; and 4) DCF_table, we can write a nested function that
# directly returns the price of a bond with the following inputs: 1) pricing
# date; 2) bond's issue date; 3) bond's maturity date; 4) bond's semi-annual
# coupon rate; and finally 5) the scenario data frame:
Bond_Price_d_s <- function(pricing_date,issue_date,mat_date,coupon,scenario) {
  MY_Tab <- mat_yield_tab(pricing_date,scenario)
  CS_YR_Tab <- CS_yield_SR(MY_Tab$t,MY_Tab$Mat_Yield_r)
  SR_Tab <- Bootstrap_SR(CS_YR_Tab)
  DCF_Tab <- DCF_table(pricing_date,issue_date,mat_date,coupon,SR_Tab)
  Price <- sum(DCF_Tab$DCF)
  return(Price)
}
```

8. Bond Book Value Computation Function:

```
# Write a function that returns the book value of a bond, given the bond's
# 1) price, 2) coupon rate, 3) pricing date, and 4) maturity date:
Bond_Price_b <- function(pricing_date,issue_date,price,mat_date,coupon) {
  # 1. Compute the yield of maturity of the bond at the pricing date:
  settle_b <- pricing_date
  coupon_b <- coupon/100
  mature_b <- mat_date
  freq_b <- 2
  price_b <- price

  yield_t <- bond.yield(settle = settle_b,mature = mature_b,coupon = coupon_b,
                        freq = freq_b,price = price_b)

  # 2. Create the coupon payment time table of the bond:
  # i) Get the number of rows of the coupon payment time table:
  n_date_rows <- 2*round(as.numeric(mat_date - issue_date)/365)

  # ii) Build the inital structure of coupon payment time table:
  Coupon_Table <- data.frame(matrix(ncol=2, nrow=n_date_rows))
  colnames(Coupon_Table) <- c("Coupon_Date", "Coupon")
```

```
  Coupon_Table <- as.data.table(Coupon_Table)

  # iii) Get the coupon payments column of the table:
  bond_coupon <- coupon/2 # Note: coupon rate is semi-annual rate.
  bond_coupon_list <- rep(x=bond_coupon, times=n_date_rows)
  Coupon_Table$Coupon <- bond_coupon_list
  # Account for the principal payment at the maturity date:
  Coupon_Table$Coupon[n_date_rows] <- Coupon_Table$Coupon[n_date_rows]+100

  # iv) Get the coupon payment dates column:
  coupon_date_list <- rep(x=issue_date, times=n_date_rows+1)

  for (i in 2:length(coupon_date_list)) {
    coupon_date_list[i] <- coupon_date_list[i-1] %m+% months(6)
  }
  coupon_date_list <- coupon_date_list[2:length(coupon_date_list)]
  Coupon_Table$Coupon_Date <- coupon_date_list

  # 3. Get the time to pricing date column in the Coupon_Table:
  Coupon_Table[,tPD:=as.numeric(Coupon_Date-pricing_date)/365]

  # 4. Fill the yields for each coupon date:
  Coupon_Table$YTM <- yield_t

  # 5. Compute the DCF column:
  Coupon_Table[,DCF:=ifelse(tPD<0,0,Coupon/((1+yield_t/2)^(2*tPD)))]

  # 6. Compute the book value and return it:
  book_value <- sum(Coupon_Table$DCF)
  if (is.na(book_value) == T) {
    return(0)
  }
  else {
    return(book_value)
  }
}
```

9. Bond Sample Data:

```
Bonds_Info <- read_xlsx("Bond_Sample.xlsx",
                        col_types = c("text",   "date",   "date",
                                      "numeric","numeric","numeric"))
# Convert the Bonds dataframe to a datatable:
Bonds_Info <- as.data.table(Bonds_Info)

# Rename columns:
colnames(Bonds_Info) <- c("CUSIP","Issue_Date","Mat_Date",
                          "Coupon","End_2019_Price", "End_2019_Value")

# Reverse engineer the multiplier:
Bonds_Info[,Multiplier:=End_2019_Value/End_2019_Price]
```

```r
# Correct the date format:
Bonds_Info$Issue_Date <- as.Date(Bonds_Info$Issue_Date)
Bonds_Info$Mat_Date <- as.Date(Bonds_Info$Mat_Date )
```

10. Market Values, Book Values, P&L Calculations:

```r
# Specify the quarter dates:
Q1_2020_Date <- as.Date("2020-03-31")
Q2_2020_Date <- as.Date("2020-06-30")
Q3_2020_Date <- as.Date("2020-09-30")
Q4_2020_Date <- as.Date("2020-12-31")
```

Bond 1:

```r
issue_date_1 <- Bonds_Info[1,Issue_Date]
mat_date_1 <- Bonds_Info[1,Mat_Date]
coupon_1 <- Bonds_Info[1,Coupon]
```

```r
# Market prices:
Bond1_Q1 <- Bond_Price_d_s(Q1_2020_Date,issue_date_1,mat_date_1,
                           coupon_1,scen_up)
Bond1_Q2 <- Bond_Price_d_s(Q2_2020_Date,issue_date_1,mat_date_1,
                           coupon_1,scen_up)
Bond1_Q3 <- Bond_Price_d_s(Q3_2020_Date,issue_date_1,mat_date_1,
                           coupon_1,scen_up)
Bond1_Q4 <- Bond_Price_d_s(Q4_2020_Date,issue_date_1,mat_date_1,
                           coupon_1,scen_up)
```

```r
# Book values:
Bond1_Q1_b <- Bond_Price_b(Q1_2020_Date,issue_date_1,Bond1_Q1,
                           mat_date_1,coupon_1)
Bond1_Q2_b <- Bond_Price_b(Q2_2020_Date,issue_date_1,Bond1_Q2,
                           mat_date_1,coupon_1)
Bond1_Q3_b <- Bond_Price_b(Q3_2020_Date,issue_date_1,Bond1_Q3,
                           mat_date_1,coupon_1)
Bond1_Q4_b <- Bond_Price_b(Q1_2020_Date,issue_date_1,Bond1_Q4,
                           mat_date_1,coupon_1)
```

```
## Warning in irr.solve(f = fn): Both Newton-Raphson and Bisection failed to find
## IRR/YTM
```

Bond 2:

```r
issue_date_2 <- Bonds_Info[2,Issue_Date]
mat_date_2 <- Bonds_Info[2,Mat_Date]
coupon_2 <- Bonds_Info[2,Coupon]
```

```r
# Market prices:
Bond2_Q1 <- Bond_Price_d_s(Q1_2020_Date,issue_date_2,mat_date_2,
                           coupon_2,scen_up)
Bond2_Q2 <- Bond_Price_d_s(Q2_2020_Date,issue_date_2,mat_date_2,
                           coupon_2,scen_up)
Bond2_Q3 <- Bond_Price_d_s(Q3_2020_Date,issue_date_2,mat_date_2,
```

```
                            coupon_2,scen_up)
Bond2_Q4 <- Bond_Price_d_s(Q4_2020_Date,issue_date_2,mat_date_2,
                            coupon_2,scen_up)
```

```
# Book values:
Bond2_Q1_b <- Bond_Price_b(Q1_2020_Date,issue_date_2,Bond2_Q1,
                            mat_date_2,coupon_2)
Bond2_Q2_b <- Bond_Price_b(Q2_2020_Date,issue_date_2,Bond2_Q2,
                            mat_date_2,coupon_2)
Bond2_Q3_b <- Bond_Price_b(Q3_2020_Date,issue_date_2,Bond2_Q3,
                            mat_date_2,coupon_2)
Bond2_Q4_b <- Bond_Price_b(Q4_2020_Date,issue_date_2,Bond2_Q4,
                            mat_date_2,coupon_2)
```

Bond 3:

```
issue_date_3 <- Bonds_Info[3,Issue_Date]
mat_date_3 <- Bonds_Info[3,Mat_Date]
coupon_3 <- Bonds_Info[3,Coupon]
```

```
# Market prices:
Bond3_Q1 <- Bond_Price_d_s(Q1_2020_Date,issue_date_3,mat_date_3,
                            coupon_3,scen_up)
Bond3_Q2 <- Bond_Price_d_s(Q2_2020_Date,issue_date_3,mat_date_3,
                            coupon_3,scen_up)
Bond3_Q3 <- Bond_Price_d_s(Q3_2020_Date,issue_date_3,mat_date_3,
                            coupon_3,scen_up)
Bond3_Q4 <- Bond_Price_d_s(Q4_2020_Date,issue_date_3,mat_date_3,
                            coupon_3,scen_up)
```

```
# Book values:
Bond3_Q1_b <- Bond_Price_b(Q1_2020_Date,issue_date_3,Bond3_Q1,
                            mat_date_3,coupon_3)
Bond3_Q2_b <- Bond_Price_b(Q2_2020_Date,issue_date_3,Bond3_Q2,
                            mat_date_3,coupon_3)
Bond3_Q3_b <- Bond_Price_b(Q3_2020_Date,issue_date_3,Bond3_Q3,
                            mat_date_3,coupon_3)
Bond3_Q4_b <- Bond_Price_b(Q4_2020_Date,issue_date_3,Bond3_Q4,
                            mat_date_3,coupon_3)
```

```
## Warning in irr.solve(f = fn): Both Newton-Raphson and Bisection failed to find
## IRR/YTM
```

Bond 4:

```
issue_date_4 <- Bonds_Info[4,Issue_Date]
mat_date_4 <- Bonds_Info[4,Mat_Date]
coupon_4 <- Bonds_Info[4,Coupon]
```

```
# Market prices:
Bond4_Q1 <- Bond_Price_d_s(Q1_2020_Date,issue_date_4,mat_date_4,
                            coupon_4,scen_up)
Bond4_Q2 <- Bond_Price_d_s(Q2_2020_Date,issue_date_4,mat_date_4,
                            coupon_4,scen_up)
```

```
Bond4_Q3 <- Bond_Price_d_s(Q3_2020_Date,issue_date_4,mat_date_4,
                           coupon_4,scen_up)
Bond4_Q4 <- Bond_Price_d_s(Q4_2020_Date,issue_date_4,mat_date_4,
                           coupon_4,scen_up)
```

```
# Book values:
Bond4_Q1_b <- Bond_Price_b(Q1_2020_Date,issue_date_4,Bond4_Q1,
                           mat_date_4,coupon_4)
Bond4_Q2_b <- Bond_Price_b(Q2_2020_Date,issue_date_4,Bond4_Q2,
                           mat_date_4,coupon_4)
Bond4_Q3_b <- Bond_Price_b(Q3_2020_Date,issue_date_4,Bond4_Q3,
                           mat_date_4,coupon_4)
Bond4_Q4_b <- Bond_Price_b(Q4_2020_Date,issue_date_4,Bond4_Q4,
                           mat_date_4,coupon_4)
```

Bond 5:

```
issue_date_5 <- Bonds_Info[5,Issue_Date]
mat_date_5 <- Bonds_Info[5,Mat_Date]
coupon_5 <- Bonds_Info[5,Coupon]
```

```
# Market prices:
Bond5_Q1 <- Bond_Price_d_s(Q1_2020_Date,issue_date_5,mat_date_5,
                           coupon_5,scen_up)
Bond5_Q2 <- Bond_Price_d_s(Q2_2020_Date,issue_date_5,mat_date_5,
                           coupon_5,scen_up)
Bond5_Q3 <- Bond_Price_d_s(Q3_2020_Date,issue_date_5,mat_date_5,
                           coupon_5,scen_up)
Bond5_Q4 <- Bond_Price_d_s(Q4_2020_Date,issue_date_5,mat_date_5,
                           coupon_5,scen_up)
```

```
# Book values:
Bond5_Q1_b <- Bond_Price_b(Q1_2020_Date,issue_date_5,Bond5_Q1,
                           mat_date_5,coupon_5)
Bond5_Q2_b <- Bond_Price_b(Q2_2020_Date,issue_date_5,Bond5_Q2,
                           mat_date_5,coupon_5)
Bond5_Q3_b <- Bond_Price_b(Q3_2020_Date,issue_date_5,Bond5_Q3,
                           mat_date_5,coupon_5)
Bond5_Q4_b <- Bond_Price_b(Q4_2020_Date,issue_date_5,Bond5_Q4,
                           mat_date_5,coupon_5)
```

Bond 6:

```
issue_date_6 <- Bonds_Info[6,Issue_Date]
mat_date_6 <- Bonds_Info[6,Mat_Date]
coupon_6 <- Bonds_Info[6,Coupon]
```

```
# Market prices:
Bond6_Q1 <- Bond_Price_d_s(Q1_2020_Date,issue_date_6,mat_date_6,
                           coupon_6,scen_up)
Bond6_Q2 <- Bond_Price_d_s(Q2_2020_Date,issue_date_6,mat_date_6,
                           coupon_6,scen_up)
Bond6_Q3 <- Bond_Price_d_s(Q3_2020_Date,issue_date_6,mat_date_6,
```

```
                            coupon_6,scen_up)
Bond6_Q4 <- Bond_Price_d_s(Q4_2020_Date,issue_date_6,mat_date_6,
                            coupon_6,scen_up)
```

```
# Book values:
Bond6_Q1_b <- Bond_Price_b(Q1_2020_Date,issue_date_6,Bond6_Q1,
                            mat_date_6,coupon_6)
Bond6_Q2_b <- Bond_Price_b(Q2_2020_Date,issue_date_6,Bond6_Q2,
                            mat_date_6,coupon_6)
Bond6_Q3_b <- Bond_Price_b(Q3_2020_Date,issue_date_6,Bond6_Q3,
                            mat_date_6,coupon_6)
Bond6_Q4_b <- Bond_Price_b(Q4_2020_Date,issue_date_6,Bond6_Q4,
                            mat_date_6,coupon_6)
```

Bond 7:

```
issue_date_7 <- Bonds_Info[7,Issue_Date]
mat_date_7 <- Bonds_Info[7,Mat_Date]
coupon_7 <- Bonds_Info[7,Coupon]
```

```
# Market prices:
Bond7_Q1 <- Bond_Price_d_s(Q1_2020_Date,issue_date_7,mat_date_7,
                            coupon_7,scen_up)
Bond7_Q2 <- Bond_Price_d_s(Q2_2020_Date,issue_date_7,mat_date_7,
                            coupon_7,scen_up)
Bond7_Q3 <- Bond_Price_d_s(Q3_2020_Date,issue_date_7,mat_date_7,
                            coupon_7,scen_up)
Bond7_Q4 <- Bond_Price_d_s(Q4_2020_Date,issue_date_7,mat_date_7,
                            coupon_7,scen_up)
```

```
# Book values:
Bond7_Q1_b <- Bond_Price_b(Q1_2020_Date,issue_date_7,Bond7_Q1,
                            mat_date_7,coupon_7)
Bond7_Q2_b <- Bond_Price_b(Q2_2020_Date,issue_date_7,Bond7_Q2,
                            mat_date_7,coupon_7)
Bond7_Q3_b <- Bond_Price_b(Q3_2020_Date,issue_date_7,Bond7_Q3,
                            mat_date_7,coupon_7)
Bond7_Q4_b <- Bond_Price_b(Q4_2020_Date,issue_date_7,Bond7_Q4,
                            mat_date_7,coupon_7)
```

Group up the market prices by quarter:

```
Q1_prices <- c(Bond1_Q1,Bond2_Q1,Bond3_Q1,Bond4_Q1,Bond5_Q1,Bond6_Q1,Bond7_Q1)
Q2_prices <- c(Bond1_Q2,Bond2_Q2,Bond3_Q2,Bond4_Q2,Bond5_Q2,Bond6_Q2,Bond7_Q2)
Q3_prices <- c(Bond1_Q3,Bond2_Q3,Bond3_Q3,Bond4_Q3,Bond5_Q3,Bond6_Q3,Bond7_Q3)
Q4_prices <- c(Bond1_Q4,Bond2_Q4,Bond3_Q4,Bond4_Q4,Bond5_Q4,Bond6_Q4,Bond7_Q4)
```

```
# Display the prices:
print(Q1_prices)
```

```
## [1]  99.91872 102.34749 101.77149 106.13192 108.99971 113.05466 109.69993
```

```
print(Q2_prices)
```

```
## [1]  99.95994 102.39954 100.87957 106.14818 107.69460 111.03873 101.42517
```

```
print(Q3_prices)
```

```
## [1]  99.99129 101.20043 100.91747 105.14432 107.96569 111.10277  97.60018
```

```
print(Q4_prices)
```

```
## [1]   0.00000 101.23562   0.00000 105.19040 106.74937 109.55577  96.87535
```

```
# Append the predicted prices to the Bonds Info data able:
Bonds_Info$Q1_Prices <- Q1_prices
Bonds_Info$Q2_Prices <- Q2_prices
Bonds_Info$Q3_Prices <- Q3_prices
Bonds_Info$Q4_Prices <- Q4_prices
```

Group up the book values by quarter:

```
Q1_prices_b <- c(Bond1_Q1_b,Bond2_Q1_b,Bond3_Q1_b,Bond4_Q1_b,
                 Bond5_Q1_b,Bond6_Q1_b,Bond7_Q1_b)
Q2_prices_b <- c(Bond1_Q2_b,Bond2_Q2_b,Bond3_Q2_b,Bond4_Q2_b,
                 Bond5_Q2_b,Bond6_Q2_b,Bond7_Q2_b)
Q3_prices_b <- c(Bond1_Q3_b,Bond2_Q3_b,Bond3_Q3_b,Bond4_Q3_b,
                 Bond5_Q3_b,Bond6_Q3_b,Bond7_Q3_b)
Q4_prices_b <- c(Bond1_Q4_b,Bond2_Q4_b,Bond3_Q4_b,Bond4_Q4_b,
                 Bond5_Q4_b,Bond6_Q4_b,Bond7_Q4_b)
```

```
# Display the prices:
print(Q1_prices_b)
```

```
## [1]  99.91759 102.56959 102.32437 106.52670 109.93641 113.90204 110.70623
```

```
print(Q2_prices_b)
```

```
## [1]  99.95891 103.23873 100.95778 107.13779 108.06793 111.31578 101.73201
```

```
print(Q3_prices_b)
```

```
## [1]  99.99067 101.40979 101.46599 105.53953 108.90277 111.94414  98.59915
```

```
print(Q4_prices_b)
```

```
## [1]   0.00000 102.07877   0.00000 106.18017 107.12471 109.84217  97.19993
```

```
# Append the book values to the Bonds Info data able:
Bonds_Info$Q1_Prices_b <- Q1_prices_b
Bonds_Info$Q2_Prices_b <- Q2_prices_b
Bonds_Info$Q3_Prices_b <- Q3_prices_b
Bonds_Info$Q4_Prices_b <- Q4_prices_b
```

Compute End-of-Quarter Bonds Portion Value:

```
# Create the market value column:
Bonds_Info[,End_2020_Q1_MV:=Q1_Prices*Multiplier]
Bonds_Info[,End_2020_Q2_MV:=Q2_Prices*Multiplier]
Bonds_Info[,End_2020_Q3_MV:=Q3_Prices*Multiplier]
Bonds_Info[,End_2020_Q4_MV:=Q4_Prices*Multiplier]
```

```
# Create the book value column:
Bonds_Info[,End_2020_Q1_BV:=Q1_Prices_b*Multiplier]
Bonds_Info[,End_2020_Q2_BV:=Q2_Prices_b*Multiplier]
Bonds_Info[,End_2020_Q3_BV:=Q3_Prices_b*Multiplier]
Bonds_Info[,End_2020_Q4_BV:=Q4_Prices_b*Multiplier]
```

```
# Condense the result table:
Results_Table <- Bonds_Info[,c(1,16:23)]
```

```
# Profit & Loss Calculations:
Results_Table[,Q1_PnL:=End_2020_Q1_MV-End_2020_Q1_BV]
Results_Table[,Q2_PnL:=End_2020_Q2_MV-End_2020_Q2_BV]
Results_Table[,Q3_PnL:=End_2020_Q3_MV-End_2020_Q3_BV]
Results_Table[,Q4_PnL:=End_2020_Q4_MV-End_2020_Q4_BV]
```

```
# Compute the End-of-Quarter Bonds Portion Market Value for each quarter:
Q1_Bond_Port_MV <- sum(Results_Table$End_2020_Q1_MV)
Q2_Bond_Port_MV <- sum(Results_Table$End_2020_Q2_MV)
Q3_Bond_Port_MV <- sum(Results_Table$End_2020_Q3_MV)
Q4_Bond_Port_MV <- sum(Results_Table$End_2020_Q4_MV)

# Compute the End-of-Quarter Bonds Portion Book Value for each quarter:
Q1_Bond_Port_BV <- sum(Results_Table$End_2020_Q1_BV)
Q2_Bond_Port_BV <- sum(Results_Table$End_2020_Q2_BV)
Q3_Bond_Port_BV <- sum(Results_Table$End_2020_Q3_BV)
Q4_Bond_Port_BV <- sum(Results_Table$End_2020_Q4_BV)

# Compute the End-of-Quarter Bonds Portion P&L:
Q1_Bond_Port_PL <- sum(Results_Table$Q1_PnL)
Q2_Bond_Port_PL <- sum(Results_Table$Q2_PnL)
Q3_Bond_Port_PL <- sum(Results_Table$Q3_PnL)
Q4_Bond_Port_PL <- sum(Results_Table$Q4_PnL)
```

11. Final Results:

2020 Q1 Bond Portion Market Value:

```
print(Q1_Bond_Port_MV)
```

```
## [1] 242328245
```

2020 Q2 Bond Portion Market Value:

```
print(Q2_Bond_Port_MV)
```

```
## [1] 239315023
```

2020 Q3 Bond Portion Market Value:

```
print(Q3_Bond_Port_MV)
```

```
## [1] 237543646
```

2020 Q4 Bond Portion Market Value:

```
print(Q4_Bond_Port_MV)
```

```
## [1] 146508445
```

2020 Q1 Bond Portion Book Value:

```
print(Q1_Bond_Port_BV)
```

```
## [1] 243356817
```

2020 Q2 Bond Portion Book Value:

```
print(Q2_Bond_Port_BV)
```

```
## [1] 240255936
```

2020 Q3 Bond Portion Book Value:

```
print(Q3_Bond_Port_BV)
```

```
## [1] 238561398
```

2020 Q4 Bond Portion Book Value:

```
print(Q4_Bond_Port_BV)
```

```
## [1] 147427402
```

2020 Q1 Bond Portion P&L:

```
print(Q1_Bond_Port_PL)
```

```
## [1] -1028572
```

2020 Q2 Bond Portion P&L:

```
print(Q2_Bond_Port_PL)
```

```
## [1] -940912.8
```

2020 Q3 Bond Portion P&L:

```
print(Q3_Bond_Port_PL)
```

```
## [1] -1017752
```

2020 Q4 Bond Portion P&L:

```
print(Q4_Bond_Port_PL)
```

```
## [1] -918957.1
```