Adil-Gokturk_HW5.R

HAG

2020-03-28

```r
# Adil Gokturk
# FIN 659

# HW5 SWAPS
# Textbook Reference:       Section 7.5, pp. 165-167; Section 7.7, pp. 169-172

## Set working directory
setwd("/Users/HAG/Desktop/Spring2020/FIN659/Assignments/hw5")
getwd()
```

```
## [1] "/Users/HAG/Desktop/Spring2020/FIN659/Assignments/hw5"
```

```r
## Libraries
library(tidyverse)
```

```
## -- Attaching packages ------------------------------ tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts --------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(quantmod)
```

```
## Loading required package: xts

## 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

## Loading required package: TTR
```

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

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
library(ggplot2)
library(jrvFinance)
library(knitr)

#############
# Problem1 ##
#############
## The first key principle to take away from this problem is that
## the swap rate at the outset of a contract is computed from
## the observed term structure of interest rates at that time,
## such that the initial value of the swap is zero to both parties involved.
## After the contract is initiated, however,
## the yield curve will change and the swap will have a positive value to one party and
## a negative value to the other - a zero-sum game.
## It should be observed that
## the floating rate is set at the beginning of each period,
##but paid at the end - thus,
## the first floating cash flow will be fixed at the outset of the swap.
## The second key principle to take away from this problem is to recognize that
## swaps can be valued in one of two ways:
## (i) as a portfolio of Forward Rate Agreements (FRAs), or
## (ii) as the difference in the prices of a floating-rate bond and a fixed-rate bond.
## A third key principle to note is that a corporate treasurer is
## likely to use an interest-rate swap
## such as this one to change the nature of a liability or
## an asset - instead of paying (or receiving) a fixed rate,
## the financial manager can pay (or receive) a floating rate.

## In a three-year interest rate swap, a financial institution agrees to pay
##a fixed rate per annum and
## to receive six-month LIBOR in return on a notional principal
##of $10 million with payments being exchanged every six months.

# Notional principal
(notional.principal <- 10000000)    #$10,000,000
```

```
## [1] 1e+07
```

```r
# Time to maturity of swap (years)
(time.to.maturity.of.swap <- 3) # years
```

```
## [1] 3
```

```r
# Frequency (payments per year)
(frequency <-   2) # payments per year
```

```
## [1] 2
```

```r
## Complete the tables below in order to calculate the swap rate
## when the financial institution enters into the contract.
## (At this time, the value of the swap should be $0.)
```

```r
##################################
## Calculation of Forward Rates##
##################################
# Time
(time <- c(0.0, 0.5, 1.0,  1.5,  2.0,  2.5, 3.0))
```

```
## [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0
```

```r
# LIBOR Rates (sa)
(LIBOR.rates.sa <-(c(3.25, 4.25, 4.95, 5.55, 5.95, 6.25))/100) #sa
```

```
## [1] 0.0325 0.0425 0.0495 0.0555 0.0595 0.0625
```

```r
# LIBOR Rates (cc)
(LIBOR.rates.cc <-frequency * log(1+(LIBOR.rates.sa/frequency))) # cc
```

```
## [1] 0.03223876 0.04205473 0.04889736 0.05474389 0.05863211 0.06154332
```

```r
# Forward Rates (cc)
(forward.rates.cc <- c(((LIBOR.rates.cc[1] * time[2]) - (0 * time[1]))/(time[2] - time[1]),
                       ((LIBOR.rates.cc[2] * time[3]) - (LIBOR.rates.cc[1] * time[2]))/(time[3] - time[2
                       ((LIBOR.rates.cc[3] * time[4]) - (LIBOR.rates.cc[2] * time[3]))/(time[4] - time[3
                       ((LIBOR.rates.cc[4] * time[5]) - (LIBOR.rates.cc[3] * time[4]))/(time[5] - time[4
                       ((LIBOR.rates.cc[5] * time[6]) - (LIBOR.rates.cc[4] * time[5]))/(time[6] - time[5
                       ((LIBOR.rates.cc[6] * time[7]) - (LIBOR.rates.cc[5] * time[6]))/(time[7] - time[6
```

```
## [1] 0.03223876 0.05187071 0.06258261 0.07228349 0.07418497 0.07609936
```

```r
# Forward Rates (sa)
(forward.rates.sa <- (frequency * (exp((forward.rates.cc)/frequency)- 1)))
```

```
## [1] 0.03250000 0.05254920 0.06357205 0.07360560 0.07557799 0.07756568
```

```r
# let's add ZEROs to values to make calculation easier
(LIBOR.rates.sa <- c(0,LIBOR.rates.sa))
```

```
## [1] 0.0000 0.0325 0.0425 0.0495 0.0555 0.0595 0.0625
```

```r
(LIBOR.rates.cc <- c(0,LIBOR.rates.cc))
```

```
## [1] 0.00000000 0.03223876 0.04205473 0.04889736 0.05474389 0.05863211 0.06154332
```

```r
(forward.rates.cc <- c(forward.rates.cc, 0))
```

```
## [1] 0.03223876 0.05187071 0.06258261 0.07228349 0.07418497 0.07609936 0.00000000
```

```r
(forward.rates.sa <- c(forward.rates.sa, 0))
```

```
## [1] 0.03250000 0.05254920 0.06357205 0.07360560 0.07557799 0.07756568 0.00000000
```

```r
# Let's put all data in a table
(analysis1.df <- data.frame(time, LIBOR.rates.sa, LIBOR.rates.cc, forward.rates.cc, forward.rates.sa))
```

```
##   time LIBOR.rates.sa LIBOR.rates.cc forward.rates.cc forward.rates.sa
## 1  0.0         0.0000     0.00000000       0.03223876       0.03250000
## 2  0.5         0.0325     0.03223876       0.05187071       0.05254920
## 3  1.0         0.0425     0.04205473       0.06258261       0.06357205
## 4  1.5         0.0495     0.04889736       0.07228349       0.07360560
## 5  2.0         0.0555     0.05474389       0.07418497       0.07557799
## 6  2.5         0.0595     0.05863211       0.07609936       0.07756568
```

```
## 7  3.0         0.0625     0.06154332       0.00000000        0.00000000
```

```r
kable(analysis1.df, align = "c")
```

time	LIBOR.rates.sa	LIBOR.rates.cc	forward.rates.cc	forward.rates.sa
0.0	0.0000	0.0000000	0.0322388	0.0325000
0.5	0.0325	0.0322388	0.0518707	0.0525492
1.0	0.0425	0.0420547	0.0625826	0.0635721
1.5	0.0495	0.0488974	0.0722835	0.0736056
2.0	0.0555	0.0547439	0.0741850	0.0755780
2.5	0.0595	0.0586321	0.0760994	0.0775657
3.0	0.0625	0.0615433	0.0000000	0.0000000

```r
## Calculation of Swap Rate
# At this time, the value of the swap should be 0
(swap.rate <- 0)
```

```
## [1] 0
```

```r
# Floating Cash Flow
(floating.cash.flow <- ((notional.principal * analysis1.df$forward.rates.sa[-7])/1) / frequency) # $
```

```
## [1] 162500.0 262746.0 317860.3 368028.0 377890.0 387828.4
```

```r
# Fixed Cash Flow
(fixed.cash.flow <- (swap.rate * notional.principal) / frequency)
```

```
## [1] 0
```

```r
# Net Cash Flow
(net.cash.flow <- floating.cash.flow - fixed.cash.flow)
```

```
## [1] 162500.0 262746.0 317860.3 368028.0 377890.0 387828.4
```

```r
# Discount Factor
(discount.factor <- (exp(-LIBOR.rates.cc[2:7] * time[2:7])))
```

```
## [1] 0.9840098 0.9588173 0.9292792 0.8962931 0.8636564 0.8314119
```

```r
# PV of Net Cash Flow
(PV.of.net.cash.flow <- net.cash.flow * discount.factor)
```

```
## [1] 159901.6 251925.4 295380.9 329860.9 326367.1 322445.1
```

```r
# Let's calculate the sum of PV of Net Cash Flow
(Sum.PV.of.net.cash.flow <- sum(PV.of.net.cash.flow))
```

```
## [1] 1685881
```

```r
# put it in a df
(Sum.PV.of.net.cash.flow <- as.data.frame(Sum.PV.of.net.cash.flow))
```

```
##   Sum.PV.of.net.cash.flow
## 1                 1685881
```

```r
# Let's put all data in a data frame
(analysis2.df <- data.frame(floating.cash.flow, net.cash.flow, discount.factor, PV.of.net.cash.flow))
```

```
##   floating.cash.flow net.cash.flow discount.factor PV.of.net.cash.flow
## 1           162500.0      162500.0       0.9840098            159901.6
```

```
## 2           262746.0       262746.0       0.9588173            251925.4
## 3           317860.3       317860.3       0.9292792            295380.9
## 4           368028.0       368028.0       0.8962931            329860.9
## 5           377890.0       377890.0       0.8636564            326367.1
## 6           387828.4       387828.4       0.8314119            322445.1
```

```r
# this visualization would be better
kable(analysis2.df, align = "c")
```

floating.cash.flow	net.cash.flow	discount.factor	PV.of.net.cash.flow
162500.0	162500.0	0.9840098	159901.6
262746.0	262746.0	0.9588173	251925.4
317860.3	317860.3	0.9292792	295380.9
368028.0	368028.0	0.8962931	329860.9
377890.0	377890.0	0.8636564	326367.1
387828.4	387828.4	0.8314119	322445.1

```r
kable(Sum.PV.of.net.cash.flow)
```

Sum.PV.of.net.cash.flow
1685881

```r
# Three months after the swap is initiated, the LIBOR rates have changed,
# as shown in the table below.
# (Yields have shifted upwards, and the yield curve has flattened.)

## Complete the tables below
## to find the value of the swap at this time
## from the financial institution's perspective.

#####################################
## Calculation of New Forward Rates##
#####################################
# Time
(time <- c(0.0, 0.25, 0.75,  1.25,  1.75, 2.25,  2.75))
```

```
## [1] 0.00 0.25 0.75 1.25 1.75 2.25 2.75
```

```r
# LIBOR Rates (sa)
(LIBOR.rates.sa <-(c(5, 5.50, 5.85, 6.15, 6.35, 6.50))/100) #sa
```

```
## [1] 0.0500 0.0550 0.0585 0.0615 0.0635 0.0650
```

```r
# LIBOR Rates (cc)
(LIBOR.rates.cc <-frequency * log(1+(LIBOR.rates.sa/frequency))) # cc
```

```
## [1] 0.04938523 0.05425733 0.05766076 0.06057339 0.06251278 0.06396609
```

```r
# Forward Rates (cc)
(forward.rates.cc <- c(((LIBOR.rates.cc[2] * time[3]) - (LIBOR.rates.cc[1] * time[2]))/(time[3] - time[2
                       ((LIBOR.rates.cc[3] * time[4]) - (LIBOR.rates.cc[2] * time[3]))/(time[4] - time[3
                       ((LIBOR.rates.cc[4] * time[5]) - (LIBOR.rates.cc[3] * time[4]))/(time[5] - time[4
                       ((LIBOR.rates.cc[5] * time[6]) - (LIBOR.rates.cc[4] * time[5]))/(time[6] - time[5
                       ((LIBOR.rates.cc[6] * time[7]) - (LIBOR.rates.cc[5] * time[6]))/(time[7] - time[6
```

```
## [1] 0.05669339 0.06276591 0.06785494 0.06930066 0.07050600
```

```
# Forward Rates (sa)
(forward.rates.sa <- (frequency * (exp((forward.rates.cc)/frequency)- 1)))
```

```
## [1] 0.05750457 0.06376118 0.06901914 0.07051529 0.07176350
```

```
# let's add ZEROs to values
(LIBOR.rates.sa <- c(0 ,LIBOR.rates.sa))
```

```
## [1] 0.0000 0.0500 0.0550 0.0585 0.0615 0.0635 0.0650
```

```
(LIBOR.rates.cc <- c(0,LIBOR.rates.cc))
```

```
## [1] 0.00000000 0.04938523 0.05425733 0.05766076 0.06057339 0.06251278 0.06396609
```

```
(forward.rates.cc <- c(0,forward.rates.cc, 0))
```

```
## [1] 0.00000000 0.05669339 0.06276591 0.06785494 0.06930066 0.07050600 0.00000000
```

```
(forward.rates.sa <- c(0,forward.rates.sa, 0))
```

```
## [1] 0.00000000 0.05750457 0.06376118 0.06901914 0.07051529 0.07176350 0.00000000
```

```
# Let's put all data in a data frame
(analysis3.df <- data.frame(time, LIBOR.rates.sa, LIBOR.rates.cc, forward.rates.cc, forward.rates.sa))
```

```
##   time LIBOR.rates.sa LIBOR.rates.cc forward.rates.cc forward.rates.sa
## 1 0.00         0.0000     0.00000000       0.00000000       0.00000000
## 2 0.25         0.0500     0.04938523       0.05669339       0.05750457
## 3 0.75         0.0550     0.05425733       0.06276591       0.06376118
## 4 1.25         0.0585     0.05766076       0.06785494       0.06901914
## 5 1.75         0.0615     0.06057339       0.06930066       0.07051529
## 6 2.25         0.0635     0.06251278       0.07050600       0.07176350
## 7 2.75         0.0650     0.06396609       0.00000000       0.00000000
```

```
# Visualize the df
kable(analysis3.df, align = "c")
```

time	LIBOR.rates.sa	LIBOR.rates.cc	forward.rates.cc	forward.rates.sa
0.00	0.0000	0.0000000	0.0000000	0.0000000
0.25	0.0500	0.0493852	0.0566934	0.0575046
0.75	0.0550	0.0542573	0.0627659	0.0637612
1.25	0.0585	0.0576608	0.0678549	0.0690191
1.75	0.0615	0.0605734	0.0693007	0.0705153
2.25	0.0635	0.0625128	0.0705060	0.0717635
2.75	0.0650	0.0639661	0.0000000	0.0000000

```
## Calculation of Value of the Swap
# At this time, the value of the swap should be 0
(swap.rate <- 0)
```

```
## [1] 0
```

```
# Floating Cash Flow
# Need to get first Floating cash value from the previous analysys
(floating.cash.flow <- c(((analysis1.df$forward.rates.sa[1] * notional.principal) / frequency),
                         (notional.principal * analysis3.df$forward.rates.sa[2:6]) / frequency)) # $
```

```
## [1] 162500.0 287522.9 318805.9 345095.7 352576.5 358817.5
```

```r
# Fixed Cash Flow
(fixed.cash.flow <- (swap.rate * notional.principal) / frequency)
```

```
## [1] 0
```

```r
# Net Cash Flow
(net.cash.flow <- floating.cash.flow - fixed.cash.flow)
```

```
## [1] 162500.0 287522.9 318805.9 345095.7 352576.5 358817.5
```

```r
# Discount Factor
(discount.factor <- (exp(-LIBOR.rates.cc[2:7] * time[2:7])))
```

```
## [1] 0.9877296 0.9601238 0.9304602 0.8994216 0.8687901 0.8386962
```

```r
# PV of Net Cash Flow
(PV.of.net.cash.flow <- net.cash.flow * discount.factor)
```

```
## [1] 160506.1 276057.5 296636.2 310386.5 306314.9 300938.9
```

```r
# Let;s calculate the sum of the PV of Net Cash Flow
(Sum.PV.of.net.cash.flow <- sum(PV.of.net.cash.flow))
```

```
## [1] 1650840
```

```r
# Put it in a df
(Sum.PV.of.net.cash.flow <- as.data.frame(Sum.PV.of.net.cash.flow))
```

```
##   Sum.PV.of.net.cash.flow
## 1                 1650840
```

```r
# # Let's put all the data in a data frame
(analysis4.df <- data.frame(floating.cash.flow, fixed.cash.flow, net.cash.flow, discount.factor, PV.of.r
```

```
##   floating.cash.flow fixed.cash.flow net.cash.flow discount.factor
## 1           162500.0               0      162500.0       0.9877296
## 2           287522.9               0      287522.9       0.9601238
## 3           318805.9               0      318805.9       0.9304602
## 4           345095.7               0      345095.7       0.8994216
## 5           352576.5               0      352576.5       0.8687901
## 6           358817.5               0      358817.5       0.8386962
##   PV.of.net.cash.flow
## 1            160506.1
## 2            276057.5
## 3            296636.2
## 4            310386.5
## 5            306314.9
## 6            300938.9
```

```r
# Visualize it
kable(analysis4.df, align = "c")
```

floating.cash.flow	fixed.cash.flow	net.cash.flow	discount.factor	PV.of.net.cash.flow
162500.0	0	162500.0	0.9877296	160506.1
287522.9	0	287522.9	0.9601238	276057.5
318805.9	0	318805.9	0.9304602	296636.2
345095.7	0	345095.7	0.8994216	310386.5
352576.5	0	352576.5	0.8687901	306314.9

floating.cash.flow	fixed.cash.flow	net.cash.flow	discount.factor	PV.of.net.cash.flow
358817.5	0	358817.5	0.8386962	300938.9

```
kable(Sum.PV.of.net.cash.flow)
```

Sum.PV.of.net.cash.flow
1650840

```
#############
# Problem2 ##
#############

# The key principle to take away from this problem is that
# even though one party has an absolute advantage in borrowing
# in to different markets relative to the other party in the swap,
# using its comparative advantage allows both parties to make savings
# and obtain funds at lower net borrowing costs.

# Jaguar, a British manufacturer, wishes to borrow U.S. dollars at a fixed rate of interest.
# Ford, a U.S. multinational, wishes to borrow sterling at a fixed rate of interest.
# They have been quoted the following rates per annum (adjusted for differential tax effects):

# Let put the values in a table
(Jaguar <- c(0.08, 0.05)) # Sterling, US$ borrowing rates per anum respectively
```

```
## [1] 0.08 0.05
```

```
(Ford <- c(0.072, 0.038)) # Sterling, US$ borrowing rates per anum respectively
```

```
## [1] 0.072 0.038
```

```
(rates <- rbind(Jaguar, Ford))
```

```
##         [,1]  [,2]
## Jaguar 0.080 0.050
## Ford   0.072 0.038
```

```
# rename column names
c("GBP", "US$") -> colnames(rates)

# convert data frame
rates <- as.data.frame(rates)
#let's take a look at it
kable(rates)
```

	GBP	US$
Jaguar	0.080	0.050
Ford	0.072	0.038

```
## Design a swap that will net Citibank, a financial intermediary,
# 10 basis points (0.001) per annum .
# Make the swap equally attractive to the two companies and ensure that
```

```r
# all foreign exchange risk is assumed by the bank.

# Let's calculate the Differentials
(Differentials <- c(rates$GBP[1]-rates$GBP[2],rates$US[1]-rates$US[2]))
```

```
## [1] 0.008 0.012
```

```r
# Let's calculate the difference in Differentials
(difference.in.Differentials <- abs(Differentials[1] - Differentials[2]))
```

```
## [1] 0.004
```

```r
# Profit made by intermediary (Citibank)

(profit.made.by.Citibank <- 0.001)
```

```
## [1] 0.001
```

```r
# Savings made on borrowing costs (Jaguar)

(savings.made.on.borrowing.costs.Jaguar <- (difference.in.Differentials- profit.made.by.Citibank)/2)
```

```
## [1] 0.0015
```

```r
# Savings made on borrowing costs (Ford)
(savings.made.on.borrowing.costs.Ford <- (difference.in.Differentials- profit.made.by.Citibank)/2)
```

```
## [1] 0.0015
```

```r
print("Company with a comparative advantage at borrowing in U.S. dollars:FORD")
```

```
## [1] "Company with a comparative advantage at borrowing in U.S. dollars:FORD"
```

```r
# Let's take a look at the borrowing rate Continuum for the companies
print("£: 8.00% <- Jaguar <- £: 8.00%, -> $: 4.85% Citibank <- £: 7.05%, -> $: 3.80% Ford  -> $: 3.80%
```

```
## [1] "£: 8.00% <- Jaguar <- £: 8.00%, -> $: 4.85% Citibank <- £: 7.05%, -> $: 3.80% Ford  -> $: 3.80%
```