Estimation of Term Structure in CN

201657015

2019.4.14

```
knitr::opts_chunk$set(echo = TRUE)
```

```
library(readxl)    # 程序包：文件读取
library(lubridate) # 程序包：时间数据
library(stringr)   # 程序包：文本数据
library(limSolve)  # 程序包：解方程组
```

在这篇报告中我们使用了在2019年4月15日还未到期的国债数据，市场上总共存在215类这样的债券。我们先从Wind获取了债券的各类指标以及在2019年4月14日之前最新的一次收盘价。总共用到的变量包括“证券代码”、“证券简称”、“债券面值”、“起息日期”、“到期日期”、“债券期限”、“票面利率” “计息方式” “利率类型”、“每年付息次数”、“年付息日1”、“年付息日2” 、“前收盘价”。

```
# 读取数据
bond <- read_xlsx('D://R/TermStructure/CN_bond_19-04-15.xlsx')
bond <- bond[!is.na(bond$前收盘价),]
row.names(bond) <- 1:nrow(bond)
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
bond$付息日二[is.na(bond$付息日二)] <- bond$付息日一[is.na(bond$付息日二)]

# 展示数据
head(bond)
```

在使用数据之前，需要对数据进行清洗，因为提取时的数据混合了字符和数值类型，需要先将数据类型统一。另外，我们只使用付息日（以及到期日）在4，7，10，1四个月份的债券，这样的债券总共有76只，如此，我们的现金流（利息及本金）只专注在每月的这四个月上，每次现金流的间隔是一个季度。

```
# 第一次付息日期处理
pay_1st <- unlist(bond[,11])
pay_1st_month <- unlist(strsplit(str_extract(pay_1st, '(.|..)月'),'月'))
na_loc <- which(is.na(pay_1st_month))
pay_1st_month[na_loc] <- substr(pay_1st,5,6)[na_loc]
pay_1st_month <- as.numeric(pay_1st_month)

bond_in_1st <- pay_1st_month %in% c(4,7,10,1)
pay_1st_month <- pay_1st_month[bond_in_1st]
# 第二次付息日期处理
pay_2cd <- unlist(bond[,12])
pay_2cd_month <- unlist(strsplit(str_extract(pay_2cd, '(.|..)月'),'月'))
na_loc <- which(is.na(pay_2cd_month))
pay_2cd_month[na_loc] <- substr(pay_2cd,5,6)[na_loc]
pay_2cd_month <- as.numeric(pay_2cd_month)

bond_in_2cd <- pay_2cd_month %in% c(4,7,10,1)
pay_2cd_month <- pay_2cd_month[bond_in_2cd]


# 债券筛选
bond_in <- bond_in_1st + bond_in_2cd
bond_fit <- bond[as.logical(bond_in),]
bond_fit <- bond_fit[1:74,]
```

我们每只债券现金流发生的时点，组成一个矩阵。该矩阵有76行（76只债券），有20列（5年，20个季度）。最初的矩阵只包含0-1元素，1表示该债券在此时点发生现金流（付息），否则没有发生。

```
# 提取债券指标
# 到期年月
n <- 4
nmax <- 4*n  # 估计年限*4 = 估计季度
end_year <- year(bond_fit$到期日期)
end_year_real <- end_year
end_year[end_year > (2019+n)] <- 2019+n
end_month <- month(bond_fit$到期日期)
# 票面利率
coupon <- bond_fit$票面利率


# 变量补充函数
fill_col <- function(lst){
  if(length(lst) < nmax){
    n_fill <- nmax - length(lst)
    lst <- c(lst, rep(0, n_fill))
  }
  else{
    lst <- lst[1:nmax]
  }
  return(lst)
}


# 付息一次和付息两次的债券分开处理
eql <- which(pay_1st_month==pay_2cd_month)

# Start here
df_fit <- c()
log_add <- c()
log_file <- c()
for(i in 1:nrow(bond_fit)){
  # 只付息一次
  if(i %in% eql){
    if(end_month[i]==1){
      real_paytime <- c(c(0,0,1), rep(c(0,0,0,1),end_year[i]-2020))
      fill_paytime <- fill_col(real_paytime)
      df_fit <- c(df_fit, fill_paytime)
      log_file <- c(log_file, '付息1次，1月到期')
    }
    else if(end_month[i]==4){
      real_paytime <- c(c(0,0,0,1), rep(c(0,0,0,1),end_year[i]-2020))
      fill_paytime <- fill_col(real_paytime)
      df_fit <- c(df_fit, fill_paytime)
      log_file <- c(log_file, '付息1次，4月到期')
    }
    else if(end_month[i]==7){
      real_paytime <- c(1, rep(c(0,0,0,1),end_year[i]-2019))
      fill_paytime <- fill_col(real_paytime)
      df_fit <- c(df_fit, fill_paytime)
      log_file <- c(log_file, '付息1次，7月到期')
    }
    else if(end_month[i]==10){
      real_paytime <- c(c(0,1),rep(c(0,0,0,1),end_year[i]-2019))
      fill_paytime <- fill_col(real_paytime)
      df_fit <- c(df_fit, fill_paytime)
```

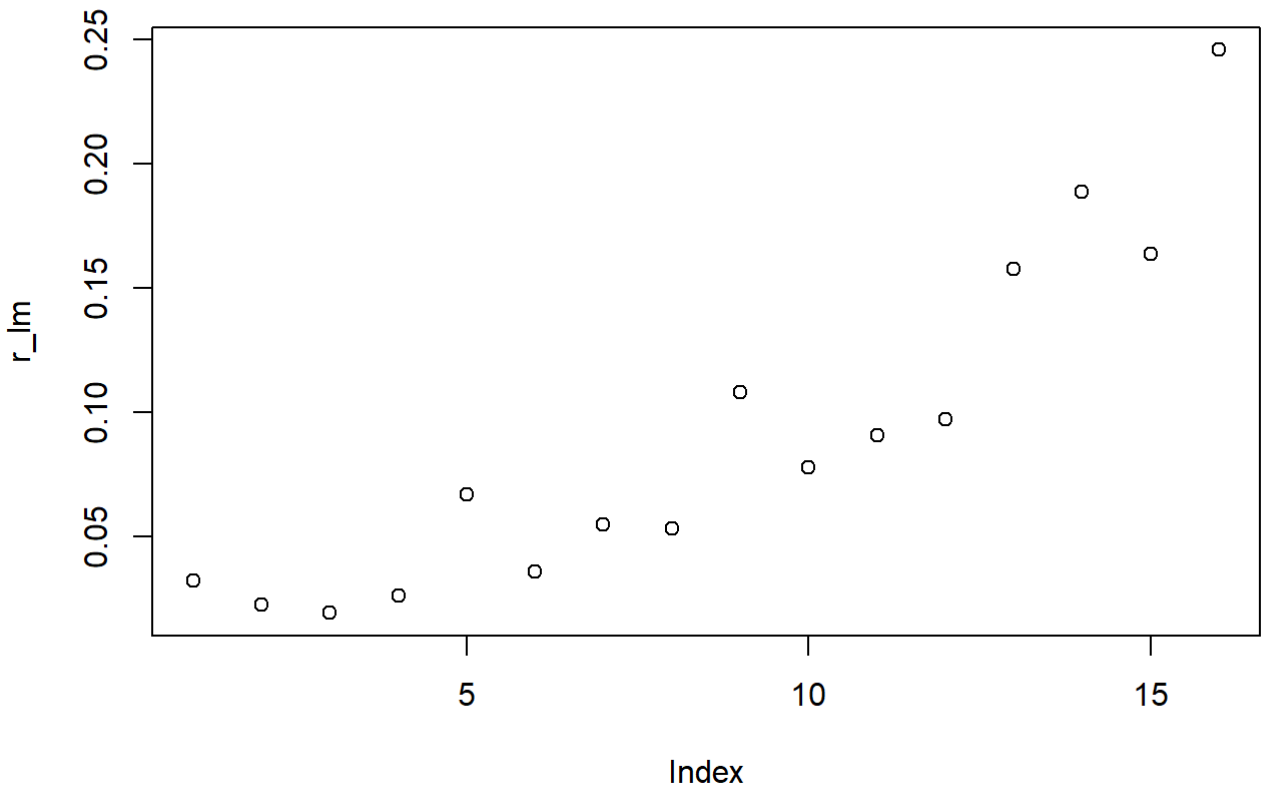
```
      log_file <- c(log_file, '付息1次，10月到期')
    }
  }
  # 付息两次
  else{
    # 在2019年7或10月到期的
    if(end_year[i] == 2019){
      if(end_month[i] == 7){
        real_paytime <- c(1,0,0,0)
        fill_paytime <- fill_col(real_paytime)
        df_fit <- c(df_fit, fill_paytime)
        log_file <- c(log_file, '付息两次，2019年7月到期')
      }
      else if(end_month[i] == 10){
        real_paytime <- c(0,1,0,0)
        fill_paytime <- fill_col(real_paytime)
        df_fit <- c(df_fit, fill_paytime)
        log_file <- c(log_file, '付息两次，2019年10月到期')
      }
    }
    else{
      if(end_month[i]==1){
        real_paytime <- c(c(1,0,1), rep(c(0,1,0,1),end_year[i]-2020))
        fill_paytime <- fill_col(real_paytime)
        df_fit <- c(df_fit, fill_paytime)
        log_file <- c(log_file, '付息两次，2020年以后1月到期')
      }
      else if(end_month[i]==4){
        real_paytime <- rep(c(0,1,0,1),end_year[i]-2019)
        fill_paytime <- fill_col(real_paytime)
        df_fit <- c(df_fit, fill_paytime)
        log_file <- c(log_file, '付息两次，2020年以后4月到期')
      }
      else if(end_month[i]==7){
        real_paytime <- c(1, rep(c(0,1,0,1),end_year[i]-2019))
        fill_paytime <- fill_col(real_paytime)
        df_fit <- c(df_fit, fill_paytime)
        log_file <- c(log_file, '付息两次，2020年以后7月到期')
      }
      else if(end_month[i]==10){
        real_paytime <- c(c(0,1),rep(c(0,1,0,1),end_year[i]-2019))
        fill_paytime <- fill_col(real_paytime)
        df_fit <- c(df_fit, fill_paytime)
        log_file <- c(log_file, '付息两次，2020年以后10月到期')
      }
    }
  }
  l <- length(df_fit)
  log_add <- c(log_add,l)
}

df_fit <- matrix(df_fit, nrow = nmax)
df_fit <- t(df_fit)
```

接着，我们把实际发生的现金流数值加入矩阵，使用的是债券面值乘以票面利率取得。

```
# 现金流矩阵
cashflow <- df_fit
# 利息与到期现金加入现金流
for(i in 1:nrow(cashflow)){
  if(end_year[i] == end_year_real[i]){
  cashflow[i,] <- cashflow[i,] * coupon[i]
  cashflow[i,max(which((cashflow[i,] == coupon[i])))] <- 100 + coupon[i]
  }
  else{
  cashflow[i,] <- cashflow[i,] * coupon[i]
  }
}
```

最后，我们使用受约束的回归，令回归的系数x1>x2>..x20，这符合对债券期限结构的假设。随后，把估计的回归系数转换为利率。（x=1/(1+r)）得到利率的期限结构。

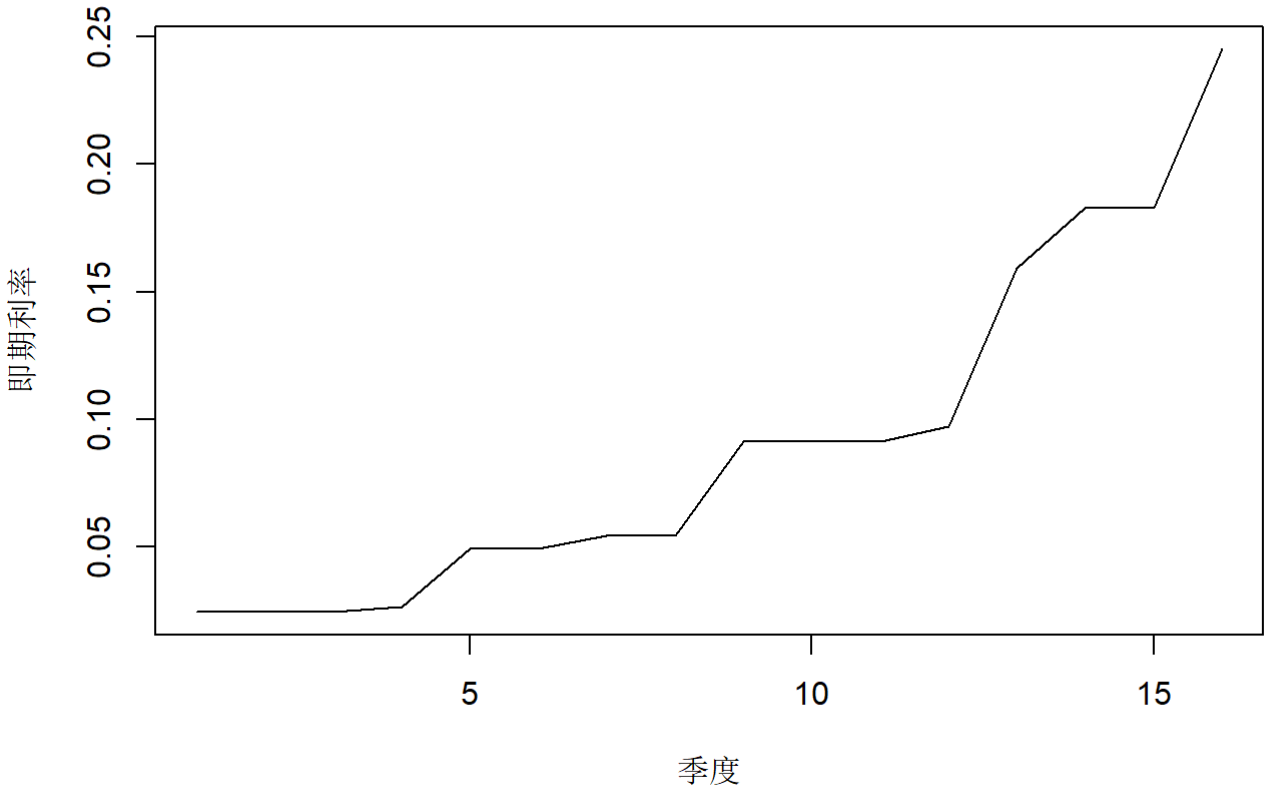
```
# 回归
y <- bond_fit$前收盘价
# 这里选择使用样本的范围，受样本选择的影响极大
y <- y[30:length(y)]
X <- cashflow
X <- X[30:nrow(X),]
ml <-lm(y~0+X)
r_lm <- 1/ml$coefficients - 1
plot(r_lm)
```



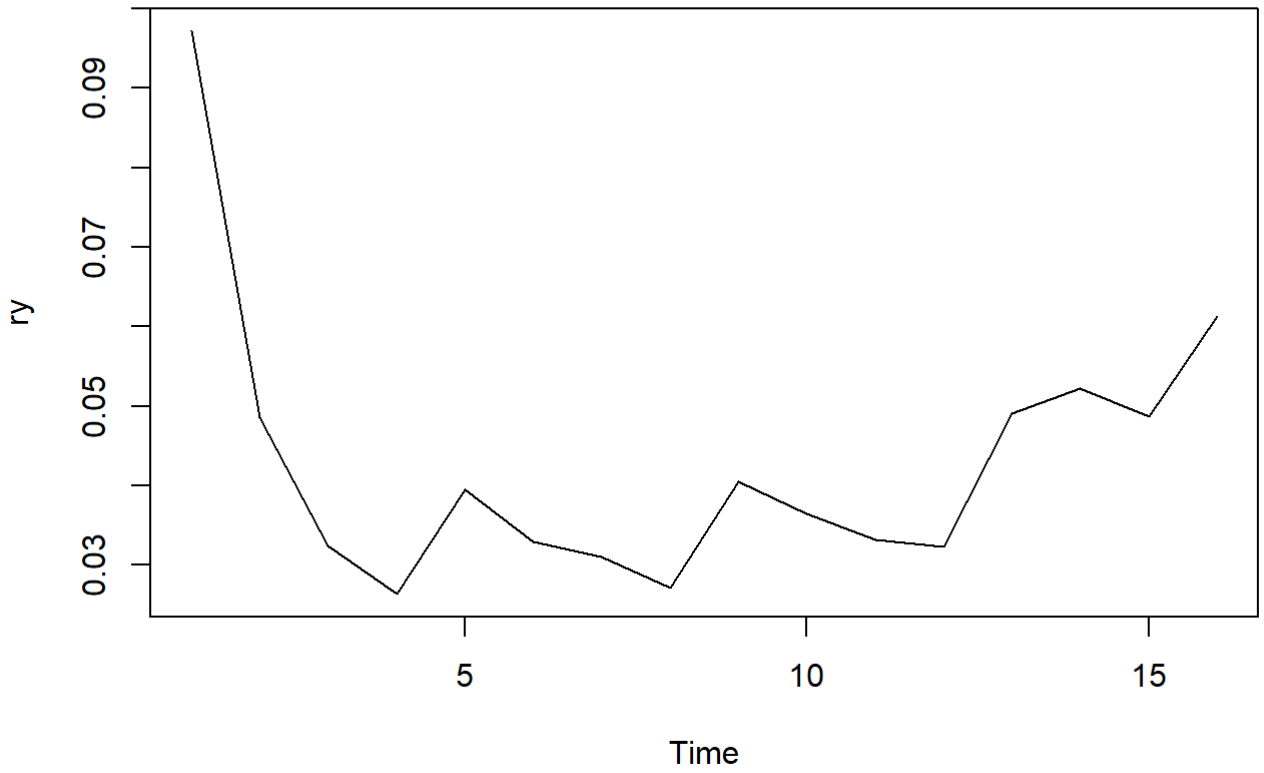
```
# 受约束回归
A <- X
B <- y
N <- nmax
G_0 <- rbind(cbind(rep(0,N-1),-1*diag(N-1)),rep(0,N))
G_1 <- G_0 + diag(N)
G_2 <- diag(c(-1,rep(0,N-1)))
G <- rbind(G_2,G_1)
H <- c(-1,rep(0,N-1),rep(0, N))


reg <- lsei(A = A, B = B, G = G, H = H, type=2)
# 系数
reg
```

```
## $X
##  [1] 0.9762762 0.9762762 0.9762762 0.9743394 0.9529883 0.9529883 0.9485771
##  [8] 0.9485771 0.9164270 0.9164270 0.9164270 0.9115567 0.8625214 0.8455031
## [15] 0.8455031 0.8030561
##
## $residualNorm
## [1] 0
##
## $solutionNorm
## [1] 469.6209
##
## $IsError
## [1] FALSE
##
## $type
## [1] "lsei"
```

```
coe <- reg$X
r <- 1/coe-1
plot(r, type='l', xlab = '季度', ylab = '即期利率')
```




```
# 年化利率
# 年化利率
m <- seq(3,n*12,3)
ry <- r * 12/m
ts.plot(ry)
```




```
ry
```

```
##  [1] 0.09720098 0.04860049 0.03240033 0.02633640 0.03946466 0.03288722
##  [7] 0.03097747 0.02710529 0.04053082 0.03647773 0.03316158 0.03234150
## [13] 0.04904355 0.05220793 0.04872740 0.06131077
```