SCO_Project1_Group1

Brian Lakey, Reshma Sekar, Julia Wu, Carlton Washburn

February 10, 2016

PROBLEM 1

Formulate the dedicated portfolio construction problem as a linear program

Setting up parameters for example portfolio decision variables: (x1), . . . , (x10) amount of each bond
(z1), . . . , (z7) excess cash at the end of each year (excluding the excess cash at the last period)

minimize: 102(x1) + 99(x2) + 101(x3) + 98(x4) + 98(x5) + 104(x6) + 100(x7) + 101(x8) + 102(x9) + 94(x10) cost of each bond

subject to:
year 1: 12000 = 100(x1) + 5(x1) + 3.5(x2) + 5(x3) + 3.5(x4) + 4(x5) + 9(x6) + 6(x7) + 8(x8) + 9(x9) + 7(x10) - (z1)
year 2: . . .

constraints = years (8)
variables = bond terms (10), excess cash (7)

```r
library('lpSolveAPI')
```

Setting up parameters for example portfolio

```
## Warning: package 'lpSolveAPI' was built under R version 3.2.3
```

```r
p = c(102,99,101,98,98,104,100,101,102,94) #price
c = c(5,3.5,5,3.5,4,9,6,8,9,7) #coupon
m = c(1,2,2,3,4,5,5,6,7,8) #maturity
l = c(12000,18000,20000,20000,16000,15000,12000,10000) #liabilities (RHS)
```

```r
my.lp = make.lp(length(l),length(p)+length(l)-1) #constraints, variables

i=1
while (i <= length(m)) { # cycles through the bonds
  v = c(rep(0,length(l))) # assigns all variables 0s
  v[1:m[i]] = c[i] # assigns coupons for all eligible years
  v[m[i]] = c[i]+100 # adds payout for final year
  set.column(my.lp,i,v) # writes to columns
  i = i+1 # increments
}
```

```r
k=1
while (k <= (length(l)-1)) { # cycles through years
  z = c(rep(0,length(l))) # assigns all excess cash variables to 0
  z[k] = -1 # assigns -1 to current year excess cash
  if (k < length(l)) {
    z[k+1] = 1 # assigns 1 to previous year excess cash
  }
  set.column(my.lp,k+length(m),z)
  k = k+1 #increments
}

set.objfn(my.lp, c(p, rep(0,(length(l)-1))))
set.constr.type(my.lp, rep(">=",length(l)))
set.rhs(my.lp, l)
```

Setting up code for solving optimization

PROBLEM 2

Solve the example portfolio case in R using lpSolveAPI() instead lpSolve()

```r
solve(my.lp)
```

Solving

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

```r
get.variables(my.lp)
```

Getting optimal variables

```
##  [1]  62.13613   0.00000 125.24293 151.50508 156.80776 123.08007   0.00000
##  [8] 124.15727 104.08986  93.45794   0.00000   0.00000   0.00000   0.00000
## [15]   0.00000   0.00000   0.00000
```

PROBLEM 3

Write a function in R that can construct a portfolio for any set of liabilities and bonds

```r
dedicate_g1 = function (p, c, m, l) {
  my.lp = make.lp(length(l),length(p)+length(l)-1)
  i=1
  while (i <= length(m)) {
    v = c(rep(0,length(l)))
    v[1:m[i]] = c[i]
    v[m[i]] = c[i]+100
    set.column(my.lp,i,v)
    i = i+1
  }

  k=1
  while (k <= (length(l)-1)) {
    z = c(rep(0,length(l)))
    z[k] = -1
    if (k < length(l)) {
      z[k+1] = 1
    }
    set.column(my.lp,k+length(m),z)
    k = k+1
  }

  set.objfn(my.lp, c(p, rep(0,(length(l)-1))))
  set.constr.type(my.lp, rep(">=",length(l)))
  set.rhs(my.lp, l)

  return(my.lp)
}
```

Function takes in bond portfolio parameters and return a .lp object

```
old_port = dedicate_g1(p,c,m,l)
solve(old_port)
```

Testing function on example portfolio

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

```
get.variables(old_port)
```

```
##  [1]  62.13613   0.00000 125.24293 151.50508 156.80776 123.08007   0.00000
##  [8] 124.15727 104.08986  93.45794   0.00000   0.00000   0.00000   0.00000
## [15]   0.00000   0.00000   0.00000
```

```
#looks right
```

```
get.objective(old_port)
```

```
## [1] 93944.5
```

```
#looks right
```

PROBLEM 4

```
setwd("C:/Users/brian/Desktop/MS-BA/03 Spring Semester/Stochastic Control and Optimization/sco_project
bonds = read.csv('treasury_bonds.csv', header=TRUE) # we creted a csv document containing all of the im
```

Construct a dedicated portfolio using this liability stream and current bond information

```
bonds['Matur_Date'] = NaN
bonds$Matur_Date = as.Date(as.character(bonds$Maturity),format="%m/%d/%Y")
```

Creating a column of date objects

```
liability_dates = c('6/30/2016','12/31/2016','6/30/2017','12/31/2017','6/30/2018','12/31/2018','6/30/20
                    '12/31/2019','6/30/2020','12/31/2020','6/30/2021','12/31/2021')
liability_dates = as.Date(liability_dates,format="%m/%d/%Y")
```

Creating a list of liability dates

Adding portfolio start date Note: we set out portfolio start date a 12/31/2015, six months prior to the first liability date

```
start_date = as.Date("12/31/2015", format="%m/%d/%Y")
portfolio_dates = c(start_date,liability_dates)
```

Assigning maturity periods using the date lists Goal is to have list of maturity due periods (numbering 1-12), similar to format given in example portfolio

```
bonds['period'] = NaN
for (i in c(1:nrow(bonds))) { #check each bond
  k=1
  while (k <= length(portfolio_dates)) {
    if (bonds$Matur_Date[i] < portfolio_dates[k]) { # against each liability date until the maturity da
      bonds$period[i] = k-1 # assign that bond to the previous period
      k = length(portfolio_dates)+1  # exit loop
    } else {
      k = k+1
    }
  }
}
```

```
w = sum(!is.na(bonds$period)) # find number of bonds that mature during project
```

Remove all bonds maturing after last liability date

```
pp = bonds$Asked[1:w] #price, removing all bonds that do not mature during the life of the project
cc = bonds$Coupon[1:w] #coupon
mm = bonds$period[1:w] #maturity
ll = c(9000000,9000000,10000000,10000000,6000000,6000000,9000000,9000000,10000000,10000000,5000000,3000

new_port = dedicate_g1(pp,cc,mm,ll)
solve(new_port)
```

Using function to optimize new portfolio

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

The chart below specifies how much and of which bonds the portfolio should invest in. The last 11 variables are excess cash carried over, indicating that approximately $4.7m should be carried over from period 10 to 11.

```
get.variables(new_port)
```

```
##   [1]     0.00     0.00 37703.66     0.00     0.00     0.00
##   [7]     0.00     0.00     0.00     0.00     0.00     0.00
##  [13]     0.00     0.00     0.00     0.00     0.00     0.00
##  [19]     0.00     0.00     0.00     0.00     0.00     0.00
##  [25]     0.00     0.00     0.00     0.00     0.00     0.00
##  [31]     0.00     0.00     0.00     0.00     0.00     0.00
##  [37]     0.00     0.00     0.00     0.00     0.00     0.00
##  [43]     0.00 41191.25     0.00     0.00     0.00     0.00
##  [49]     0.00     0.00     0.00     0.00     0.00     0.00
##  [55]     0.00     0.00     0.00     0.00     0.00     0.00
##  [61]     0.00     0.00     0.00     0.00     0.00     0.00
##  [67]     0.00     0.00     0.00     0.00 54280.59     0.00
##  [73]     0.00     0.00     0.00     0.00     0.00     0.00
##  [79]     0.00     0.00     0.00     0.00 59030.14     0.00
##  [85]     0.00     0.00     0.00     0.00     0.00     0.00
##  [91]     0.00     0.00     0.00     0.00     0.00     0.00
##  [97]     0.00     0.00     0.00     0.00     0.00     0.00
## [103]     0.00     0.00     0.00     0.00     0.00     0.00
## [109]     0.00     0.00     0.00     0.00     0.00     0.00
## [115]     0.00     0.00 24269.07     0.00     0.00     0.00
## [121]     0.00     0.00     0.00     0.00     0.00     0.00
## [127]     0.00     0.00     0.00     0.00     0.00     0.00
## [133]     0.00     0.00     0.00 26483.62     0.00     0.00
## [139]     0.00     0.00     0.00     0.00     0.00     0.00
## [145]     0.00 58867.14     0.00     0.00     0.00     0.00
## [151]     0.00     0.00     0.00     0.00     0.00     0.00
## [157]       0.00       0.00       0.00       0.00   64091.60       0.00
## [163]       0.00       0.00       0.00       0.00       0.00       0.00
## [169]       0.00       0.00       0.00       0.00       0.00       0.00
## [175]       0.00   79299.05       0.00       0.00       0.00       0.00
## [181]       0.00       0.00       0.00       0.00       0.00       0.00
## [187]       0.00       0.00       0.00       0.00       0.00  133785.13
## [193]       0.00       0.00       0.00       0.00       0.00       0.00
## [199]       0.00       0.00       0.00       0.00       0.00       0.00
## [205]       0.00       0.00       0.00       0.00       0.00       0.00
## [211]       0.00       0.00       0.00       0.00       0.00       0.00
## [217]   27745.66       0.00       0.00       0.00       0.00       0.00
## [223]       0.00       0.00       0.00       0.00       0.00       0.00
## [229]       0.00       0.00       0.00       0.00 4774566.47       0.00
```

The table below is a more condensed version of the investment strategy:

```
bond_list = get.variables(new_port)[1:w]
buy_bonds = matrix(NaN, length(which(bond_list!=0)), 2)
colnames(buy_bonds) = c("bond", "amount")
buy_bonds[,1] = which(bond_list!=0)
buy_bonds[,2] = bond_list[which(bond_list!=0)]
buy_bonds
```

```
##       bond    amount
##  [1,]    3  37703.66
##  [2,]   44  41191.25
##  [3,]   71  54280.59
##  [4,]   83  59030.14
##  [5,]  117  24269.07
##  [6,]  136  26483.62
##  [7,]  146  58867.14
##  [8,]  161  64091.60
##  [9,]  176  79299.05
## [10,]  192 133785.13
## [11,]  217  27745.66
```

Savings (below) of $21,957,772 (ignoring interest/time value of money)

```
get.objective(new_port)
```

```
## [1] 74042228
```

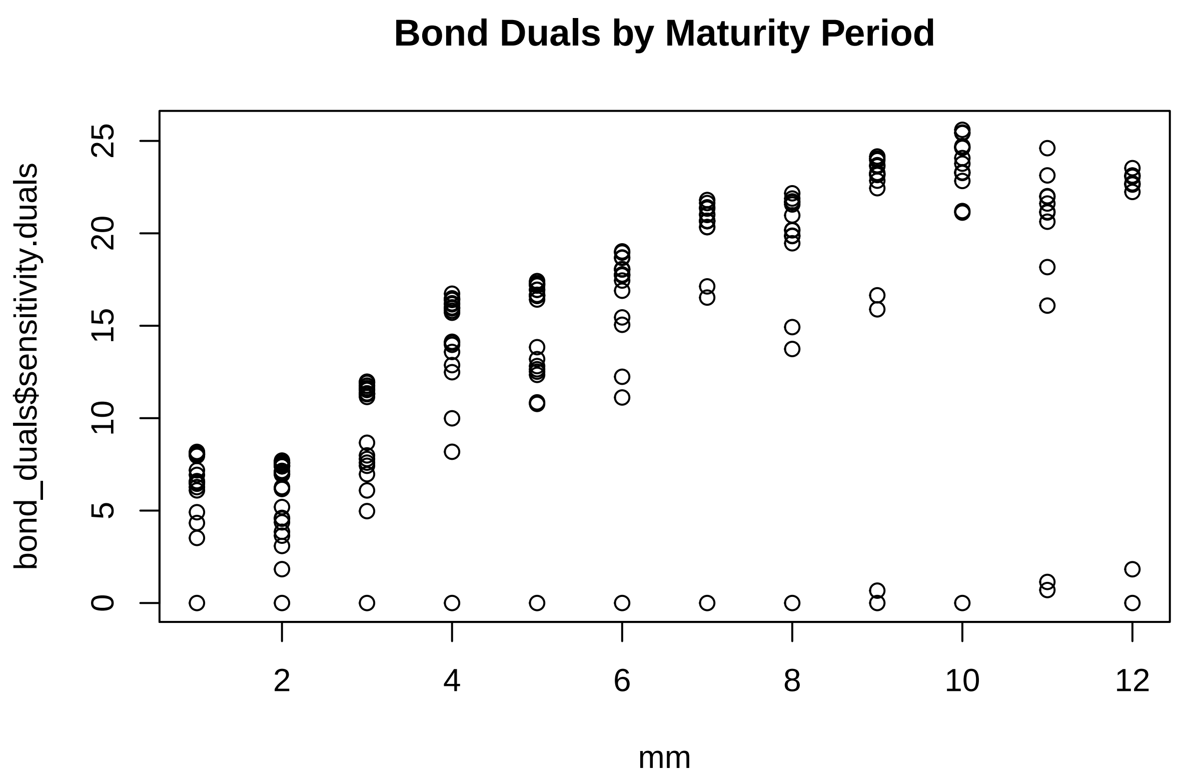
```
#sum(l) #pv of liabilities = $96,000,000
#optimal portfolio = $74,042,228
```

```


duals = get.dual.solution(new_port)
sensitivity = get.sensitivity.rhs(new_port)
dual_df = data.frame(sensitivity$duals,sensitivity$dualsfrom,sensitivity$dualstill)

liab_duals = dual_df[1:12,]
bond_duals = dual_df[13:235,]
```

Plotting bond and liability duals versus libability period (analagous to date) The graph below shows the shadow prices for each of the 223 bonds by maturity period. The shadow price represents the increase in the objective value (cost of the portfolio) from a purchase of an additional bond. We see a positive relationship between maturity period and the duals, such that buying additional bonds with later maturity dates has a greater impact on the total portfolio cost than buying bonds with shorter term maturities.

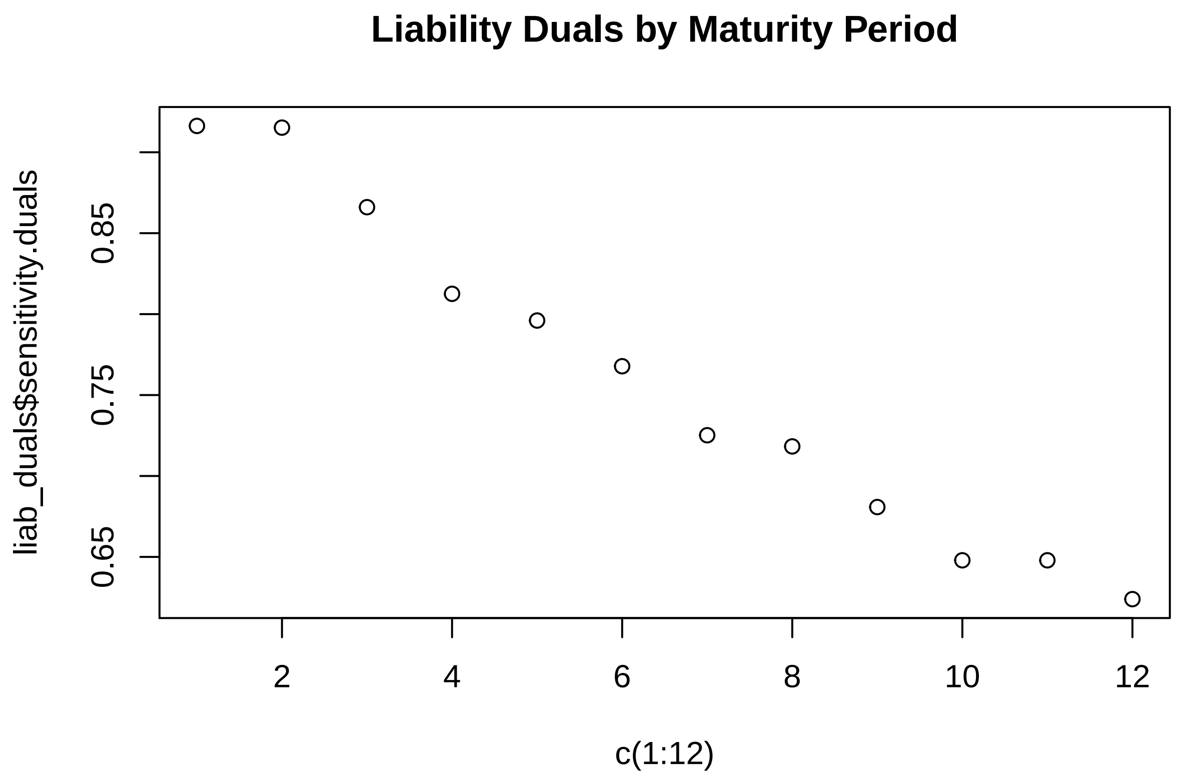
This makes sense, because the further into the future a bond matures, the more coupons it will generate and the more expensive it is. For example, bond #1 has a maturity of 2/15/2016 and a shadow price of 8.05. This means buying an additional unit of bond 1 would increase the objective value by 8.05. Buying an additional bond that matures in a later period would increase the objective value by a greater amount.

```r
plot(mm, bond_duals$sensitivity.duals, main="Bond Duals by Maturity Period")
```



Looking at liability duals by maturity shows a negative linear relationship. The first liability has a shadow price of 0.92, which means that an increase in the first liability by $1 requires a $0.92 increase in the objective value, or cost of the portfolio. Shadow prices fall as maturity dates rise because there is more time to accumulate return to cover the liability.

```r
plot(c(1:12), liab_duals$sensitivity.duals, main="Liability Duals by Maturity Period")
```


Liability Duals by Maturity Period
liab_duals$sensitivity.duals
0.65
0.75
0.85
2
4
6
8
10
12
c(1:12)