Project1_g1

Caroline Nelson, Valerie Roth, Wenduo Wang, Jon Zeller

February 4, 2017

Part 1

The portfolio construction problem can be formulated as a linear program where the objective is to minimize the total costs of bonds purchased subject to the liability constraints. The decision boils down to how much of each bond type to purchase at the onset of the problem. Using the provided numbers, the decision is as follows:

Choice Variables Choose how much of each bond to buy - call them x_{1-10}

Objective Function

Here, our goal is to minimize the total cost, defined by the prices of each bond multiplied by the quantity purchased:

$$102x_1 + 99x_2 + 101x_3 + 98x_4 + 98x_5 + 104x_6 + 100x_7 + 101x_8 + 109x_9 + 94x_{10}$$

Constraints The constraints in this problem correspond to the liabilities that must be paid in each period. The available funds, which will be exactly equal to the liabilities, are the coupons of each surviving bond plus the the face values of all bonds that expire in that period. They are as follows:

Year 1: $12000 = 105x_1 + 3.5x_2 + 5x_3 + 3.5x_4 + 4x_5 + 9x_6 + 6x_7 + 8x_8 + 9x_9 + 7x_{10}$
Year 2: $18000 = 103.5x_2 + 105x_3 + +3.5x_4 + 4x_5 + 9x_6 + 6x_7 + 8x_8 + 9x_9 + 7x_{10}$
Year 3: $20000 = 103.5x_4 + 4x_5 + 9x_6 + 6x_7 + 8x_8 + 9x_9 + 7x_{10}$
Year 4: $20000 = 104x_5 + 109x_6 + 6x_7 + 8x_8 + 9x_9 + 7x_{10}$
Year 5: $16000 = 106x_7 + 8x_8 + 9x_9 + 7x_{10}$
Year 6: $15000 = 108x_8 + 9x_9 + 7x_{10}$
Year 7: $12000 = 109x_9 + 7x_{10}$
Year 8: $10000 = 107x_{10}$

For example, in year 3, the liabilities of $20000 must be met by a combination of the coupon+face values of bond 4, which reaches maturity, plus the coupon payments of bonds 5-10 which are yet to mature.

Part 2

```
# define constraints matrix
A = matrix(0, 8, 10)  # 10 input vars, 8 constraints

# Bond 1 2 3 4 5 6 7 8 9 10 Coupon 5 3.5 5 3.5 4 9 6 8 9 7
# Maturity 1 2 2 3 4 5 5 6 7 8
```

```r
A[1, 1] = c(105)  # first column is payoffs of bond 1 in each year - matures at t=1
A[1:2, 2] = c(rep(3.5, 1), 103.5)  # second column is payoffs of bond 2 in each year - matures at t=2
A[1:2, 3] = c(rep(5, 1), 105)  # third col is payoffs of bond 3, etc...
A[1:3, 4] = c(rep(3.5, 2), 103.5)
A[1:4, 5] = c(rep(4, 3), 104)
A[1:5, 6] = c(rep(9, 4), 109)
A[1:5, 7] = c(rep(6, 4), 106)
A[1:6, 8] = c(rep(8, 5), 108)
A[1:7, 9] = c(rep(9, 6), 109)
A[1:8, 10] = c(rep(7, 7), 107)


# define values for constraints matrix - liabilities in each
# time period
B = c(12000, 18000, 20000, 20000, 16000, 15000, 12000, 10000)

# define obj function - minimize costs of bonds
C = c(102, 99, 101, 98, 98, 104, 100, 101, 102, 94)

# define direction of inequalities - equalities
dir = rep("=", 8)

# solve linear program and print solution and obj vals
soln = lp(C, A, direction = "min", dir, B)

soln$status
```

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

```r
soln$solution
```

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

```r
soln$objval
```

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

Part 3

```r
# Inputs: P is the vector containing the prices of  = 1, … ,
#  bonds C is the vector containing the coupon payments for
# the N bonds. M is the vector containing the maturity (in
# years) for the N bonds. L is the vector of non-negative
# liabilities for  = 1, … ,  years
```

```r
solvebonds <- function(p, c, m, l) {

    # Inputs: P is the vector containing the prices of  = 1, … ,
    #  bonds C is the vector containing the coupon payments for
    # the N bonds. M is the vector containing the maturity (in
    # years) for the N bonds. L is the vector of non-negative
    # liabilities for  = 1, … ,  y


    # initialize matrix of zeroes - one row for each constraint
    # (liability), one column for each bond
    A = matrix(0, length(l), length(p))  # l constraints, p = m input
vars

    # fill in martrix with payments and maturities for each bond,
    # do the following:
    for (i in seq(1, length(c))) {
        A[1:m[i], i] = c(rep(c[i], m[i] - 1), 100 + c[i])
        # each column - corresponding to a bond - is filled with each
        # row through its maturity with its corresponding payment
    }

    # define values for constraints matrix - liabilities in each
    # time period
    B = l

    # define obj function - minimize costs of bonds
    C = p

    # define direction of equalities vector
    dir = rep("=", length(l))


    # solve LP and return results
    soln = lp(C, A, direction = "min", dir, B, compute.sens = 1)
    return(soln)
}

### test function liabilities
liab = c(12000, 18000, 20000, 20000, 16000, 15000, 12000, 10000)

# prices
price = c(102, 99, 101, 98, 98, 104, 100, 101, 102, 94)

# maturities
mat = c(1, 2, 2, 3, 4, 5, 5, 6, 7, 8)

# coupon values
```

```
coup = c(5, 3.5, 5, 3.5, 4, 9, 6, 8, 9, 7)

## Test the function
test = solvebonds(p = price, c = coup, m = mat, l = liab)

test$solution

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

Part 4

To develop the optimal combination of bonds to meet this liability schedule, we will assume a date of 12/31/2016 and use historical bond price data from this date. Then, we can look at the menu of bonds available with maturities corresponding to the dates of liabilities and our problem becomes similar to the one above. To account for the six-month coupon payments we will divide the listed coupons by 2 and multiply all of the maturities by 2, so that each time period represents six months rather than one year.

Looking at the bond data from 12/30/2016, we see the following schedule of bonds relevant to our dates:

Maturity	Coupon	Asked
6/30/2017	0.625	99.9922
6/30/2017	0.75	100.0547
6/30/2017	2.5	100.9063
12/31/2017	0.75	99.8438
12/31/2017	1	100.0938
12/31/2017	2.75	101.8047
6/30/2018	0.625	99.3828
6/30/2018	1.375	100.4688
6/30/2018	2.375	101.9453
12/31/2018	1.25	100.1016
12/31/2018	1.375	100.3359
12/31/2018	1.5	100.5781
6/30/2019	1	99.2266
6/30/2019	1.625	100.7656
12/31/2019	1.125	99.0469
12/31/2019	1.625	100.5156
6/30/2020	1.625	100.0547
6/30/2020	1.875	100.9688

12/31/2020	1.75	99.9688
12/31/2020	2.375	102.4453
6/30/2021	1.125	96.7734
6/30/2021	2.125	101.0703
12/31/2021	2	100.3672
12/31/2021	2.125	100.8516
6/30/2022	2.125	100.3125
12/31/2022	2.125	99.9219

We will let these 26 bonds be equal to x_1 to x_{26}, with maturities of 1 to 12 for each of the six-month intervals. We will define C as a the coupon vector and P as the price vector with entries for each of x_1 to x_{26}.

Thus, our constraints become, generally, $liability(t) = [(100 + c/2) * x_i]$ (if x_i matures in time t) $+[c/2 * x_i]$, if x_i matures in time > t. The liabilities are as follows:

Date	Liability
6/30/17	9000000
12/31/17	9000000
6/30/18	10000000
12/31/18	10000000
6/30/19	6000000
12/31/19	6000000
6/30/20	9000000
12/31/20	9000000
6/30/21	10000000
12/31/21	10000000
6/30/22	5000000
12/31/22	3000000

Using this information as inputs to our linear program, we can formulate the problem as follows:

Choices: x_{1-26} Objective: min $\sum_{i=1}^{26} p_i * x_i$, where p spot price.
Constraints: for each due date of liabilities, the payment from the bonds is equal to the liability.

We have used the input data from the WSJ site as a raw input, and extracted the relevant prices and coupons for each bond above. Solving below, we can see the optimal solution and objective function value.

```
data <- read.csv("wsj (1).csv")
# Convert Maturity dates to date format
```

```r
data$Maturity <- as.Date(data$Maturity, format = "%m/%d/%Y")

#### Input - Maturity dates
maturity_dates <- c("2017-06-30", "2017-12-31", "2018-06-30",
    "2018-12-31", "2019-06-30", "2019-12-31", "2020-06-30", "2020-12-
31",
    "2021-06-30", "2021-12-31", "2022-06-30", "2022-12-31")

### Input - liabilities at each maturity

liabilities <- c(9e+06, 9e+06, 1e+07, 1e+07, 6e+06, 6e+06, 9e+06,
    9e+06, 1e+07, 1e+07, 5e+06, 3e+06)

# Step 1: find valid bonds: which bonds mature at the defined
# dates
find_options <- function(date) {
    return(data[data$Maturity == date, ])
}
options <- lapply(maturity_dates, find_options)

# Step 2: convert maturities to periods Step 3: get the price
# of each bond Step 4: get the coupon of each bond

# Define a helper to calculate maturity periods in half years
timedelta <- function(t1, t2) {
    return(as.integer(round(difftime(toString(t1), toString(t2),
        units = "days")/(365/2))))
}

# Transform the input to vectors for solver
maturity <- c()
price <- c()
coupon <- c()
today <- "2016-12-30"
for (i in 1:length(options)) {
    maturities <- options[[i]]$Maturity
    prices <- options[[i]]$Asked
    coupons <- options[[i]]$Coupon
    for (j in 1:length(maturities)) {
        maturity <- c(maturity, timedelta(maturities[j], today))
        price <- c(price, prices[j])
        # Important: semi-annual coupon should be halved
        coupon <- c(coupon, coupons[j]/2)
    }
}

result <- solvebonds(p = price, c = coupon, m = maturity, l =
liabilities)
Purchase <- result$solution
```

```r
Periods_to_maturity <- maturity
bonds <- data_frame()
for (i in 1:length(options)) {
    bonds <- rbind(bonds, options[[i]])
}

bonds <- cbind(bonds, Purchase, Periods_to_maturity)

# spot check that bonds list equals manual version above
bonds[, c("Maturity", "Coupon", "Asked", "Purchase")]
```

```
##       Maturity Coupon    Asked Purchase
## 24  2017-06-30  0.625  99.9922     0.00
## 25  2017-06-30  0.750 100.0547     0.00
## 26  2017-06-30  2.500 100.9063 81093.12
## 47  2017-12-31  0.750  99.8438     0.00
## 48  2017-12-31  1.000 100.0938     0.00
## 49  2017-12-31  2.750 101.8047 82106.78
## 73  2018-06-30  0.625  99.3828     0.00
## 74  2018-06-30  1.375 100.4688     0.00
## 75  2018-06-30  2.375 101.9453 93235.75
## 98  2018-12-31  1.250 100.1016     0.00
## 99  2018-12-31  1.375 100.3359     0.00
## 100 2018-12-31  1.500 100.5781 94342.92
## 120 2019-06-30  1.000  99.2266 55050.50
## 121 2019-06-30  1.625 100.7656     0.00
## 141 2019-12-31  1.125  99.0469 55325.75
## 142 2019-12-31  1.625 100.5156     0.00
## 157 2020-06-30  1.625 100.0547 85636.96
## 158 2020-06-30  1.875 100.9688     0.00
## 172 2020-12-31  1.750  99.9688 86332.76
## 173 2020-12-31  2.375 102.4453     0.00
## 188 2021-06-30  1.125  96.7734     0.00
## 189 2021-06-30  2.125 101.0703 97088.17
## 204 2021-12-31  2.000 100.3672     0.00
## 205 2021-12-31  2.125 100.8516 98119.73
## 213 2022-06-30  2.125 100.3125 49162.25
## 223 2022-12-31  2.125  99.9219 29684.60
```

```r
# create new solution object and check values
soln = solvebonds(p = price, c = coupon, m = maturity, l = liabilities)
soln$status
```

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

```r
soln$objval
```

```
## [1] 91282713
```

```r
Purchase <- soln$solution
```

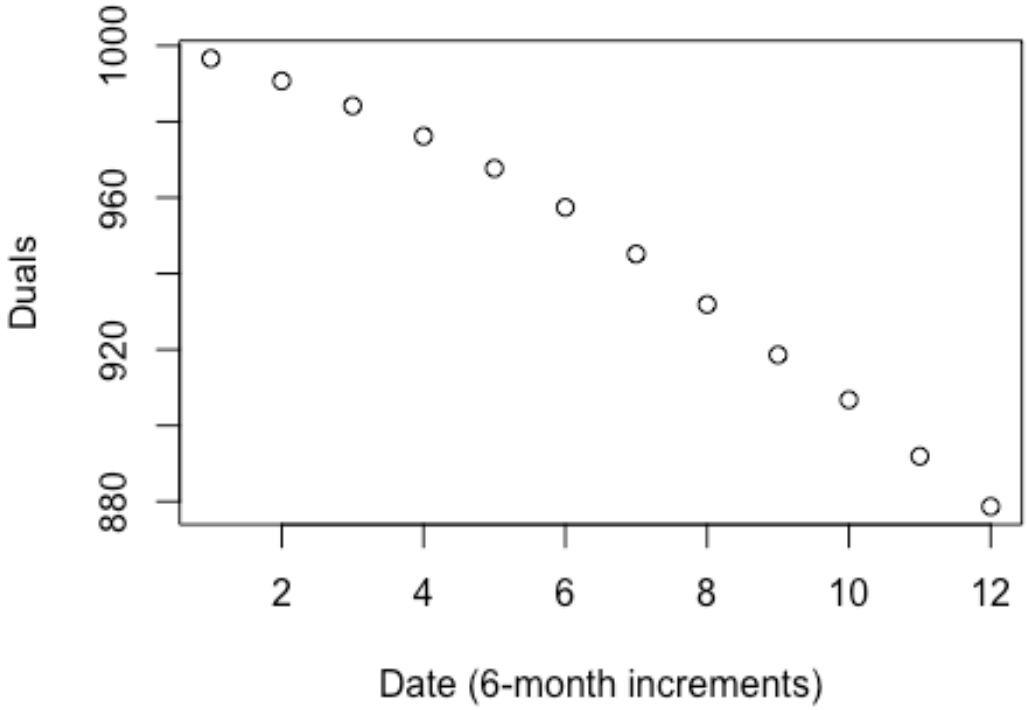
```
bonds <- cbind(bonds, Purchase)
bonds
```

```
##       Maturity Coupon      Bid    Asked     Chg Asked.yield Purchase
## 24  2017-06-30  0.625  99.9766  99.9922 -0.0078       0.641     0.00
## 25  2017-06-30  0.750 100.0391 100.0547  0.0078       0.638     0.00
## 26  2017-06-30  2.500 100.8906 100.9063 -0.0391       0.651 81093.12
## 47  2017-12-31  0.750  99.8281  99.8438  0.0234       0.909     0.00
## 48  2017-12-31  1.000 100.0781 100.0938  0.0156       0.905     0.00
## 49  2017-12-31  2.750 101.7891 101.8047 -0.0078       0.918 82106.78
## 73  2018-06-30  0.625  99.3672  99.3828  0.0313       1.043     0.00
## 74  2018-06-30  1.375 100.4531 100.4688  0.0078       1.057     0.00
## 75  2018-06-30  2.375 101.9297 101.9453  0.0469       1.057 93235.75
## 98  2018-12-31  1.250 100.0859 100.1016  0.0547       1.198     0.00
## 99  2018-12-31  1.375 100.3203 100.3359  0.0234       1.204     0.00
## 100 2018-12-31  1.500 100.5625 100.5781  0.0078       1.205 94342.92
## 120 2019-06-30  1.000  99.2109  99.2266  0.0391       1.317 55050.50
## 121 2019-06-30  1.625 100.7500 100.7656  0.0313       1.312     0.00
## 141 2019-12-31  1.125  99.0313  99.0469  0.0703       1.452 55325.75
## 142 2019-12-31  1.625 100.5000 100.5156  0.0313       1.448     0.00
## 157 2020-06-30  1.625 100.0391 100.0547   0.125       1.609 85636.96
## 158 2020-06-30  1.875 100.9531 100.9688  0.1094       1.589     0.00
## 172 2020-12-31  1.750  99.9531  99.9688  0.1172       1.758 86332.76
## 173 2020-12-31  2.375 102.4297 102.4453  0.1094       1.738     0.00
## 188 2021-06-30  1.125  96.7578  96.7734  0.1172       1.877     0.00
## 189 2021-06-30  2.125 101.0547 101.0703  0.1406       1.875 97088.17
## 204 2021-12-31  2.000 100.3516 100.3672  0.1797       1.923     0.00
## 205 2021-12-31  2.125 100.8359 100.8516  0.1563       1.945 98119.73
## 213 2022-06-30  2.125 100.2969 100.3125  0.1953       2.065 49162.25
## 223 2022-12-31  2.125  99.9063  99.9219  0.2344       2.139 29684.60
##     Periods_to_maturity Purchase
## 24                    1     0.00
## 25                    1     0.00
## 26                    1 81093.12
## 47                    2     0.00
## 48                    2     0.00
## 49                    2 82106.78
## 73                    3     0.00
## 74                    3     0.00
## 75                    3 93235.75
## 98                    4     0.00
## 99                    4     0.00
## 100                   4 94342.92
## 120                   5 55050.50
## 121                   5     0.00
## 141                   6 55325.75
## 142                   6     0.00
## 157                   7 85636.96
## 158                   7     0.00
## 172                   8 86332.76
```

```
## 173                    8     0.00
## 188                    9     0.00
## 189                    9 97088.17
## 204                   10     0.00
## 205                   10 98119.73
## 213                   11 49162.25
## 223                   12 29684.60
```

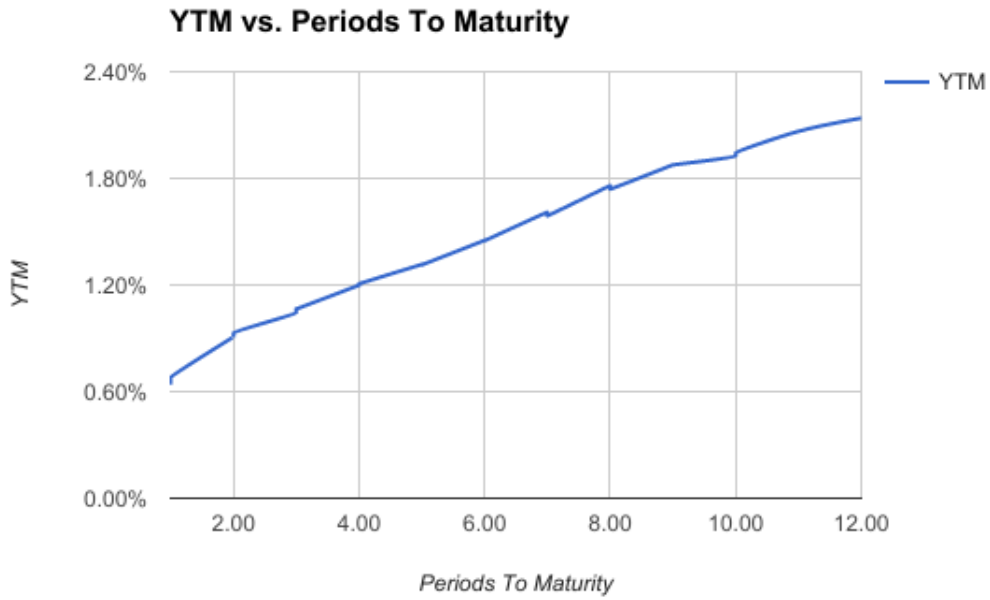
We can see that the above bond combination results in the cheapest possible way to meet our liabilities. Essentially, it involves buying only the cheapest bond option at each liability time period, which makes intuitive sense. Next, we will plot the duals/shadow prices for each time period (equivalent to each liability/constraint).

```
plotter <- function(p, c, m, l) {
    soln1 = solvebonds(p, c, m, l)
    # print(soln1$duals)
    y_duals = soln1$duals[1:12] * 1000

    x_dates = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)

    plot(x_dates, y_duals, xlab = "Date (6-month increments)",
        ylab = "Duals")
}

plotter(p = price, c = coupon, m = maturity, l = liabilities)
```



Above we can see a plot that shows, on the x-axis, each date which corresponds to a liability. The y-axis shows the associated dual or shadow price. These values represent the change in the objective function for a given unit change in the liability/cosntraint. They are relevant only in a neighborhood around the current values, but they represent how much more or less we would have to pay upfront to meet a one-unit change in the corresponding liability. As such, they are a way to see which constraints are most sensitive.

We can see that as the possible combinations of bonds increases, so does the liability constraint. For every $1000 increase in liability, the cost needed to offset the liability also increases, and since there are more options available in year 1, it makes sense for the cost increase to be close to $1000. As time passes, there is a decrease in cost increase, this exhibits the time value of money-short term bonds are more expensive, but the bond will innately be cheaper. This relationship can be characterised by the yield to maturity curve below.



Yield to Maturity Curve