Project1 - Optimization

Henry Chang, Catherine McNabb, Shan Qin

2/10/2019

The Linear Program

1.Formulate the budget allocation as a linear programming problem.

For this problem we will:

Let xi be the amount invested in the ith platform. We have: X_1 through X_{10}

To optimize $3.1X_1 + 4.9X_2 + 2.4X_3 + 3.9X_4 + 1.6X_5 + 2.4X_6 + 4.6X_7 + 2.6X_8 + 3.3X_9 + 4.4X_{10}$

Subject to $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} <= 10$ and $x_1 + x_2 <= x_6 + x_{10}$ and $x_5 + x_6 + x_7 + x_8 + x_9 >= 2(x_3 + x_4)$ and $x_1 <= 3$ and $x_2 <= 3$ and $x_3 <= 3$ and $x_4 <= 3$ and $x_5 <= 3$ and $x_6 <= 3$ and $x_7 <= 3$ and $x_8 <= 3$ and $x_9 <= 3$ and $x_{10} <= 3$

2. Solve lp() in R.

The constraints are formulated into the A matrix size 13 X 10. In which the first two rows are contraints a), b), and the constraint c) is constructed as 10 X 10 identity matrix. The last row of A matrix forumulates the total budget of $10 million.

```
c = c(3.1, 4.9, 2.4, 3.9, 1.6, 2.4, 4.6, 2.6, 3.3, 4.4)/100
A = matrix(0, 13, 10)
A[1, c(1, 2, 5, 10)] = c( 1,  1, -1, -1)
A[2, c(3, 4, 5, 6, 7, 8, 9)] = c( 2, 2, -1, -1, -1, -1, -1)
for( i in c(1:10)){
  A[2+i, i] = 1
}
A[13, ] = rep(1, 10)

d = c(0, 0, rep(3, 10), 10)
dir = rep("<=", 13)
```

```
alc1 = lp("max",c,A, dir, d)
#Allocation 1
alc1$solution
```

```
##  [1] 0 3 0 1 0 0 3 0 0 3
```

3. Write a function in R for allocation for any ROI vector

```
#Solve the allocation using the same structure as Q1, having upper_bound, budget as inpu
t.

#Enter the budget amount at the upper_bound input if there is no upper bound
allocation = function(ROI_vec, upper_bound, budget){
c = ROI_vec
A = matrix(0, 13, 10)
A[1, c(1, 2, 5, 10)] = c( 1,  1, -1, -1)
A[2, c(3, 4, 5, 6, 7, 8, 9)] = c( 2, 2, -1, -1, -1, -1, -1)
for( i in c(1:10)){
  A[2+i, i] = 1
}
A[13, ] = rep(1, 10)

d = c(0, 0, rep(upper_bound, 10), budget)
dir = rep("<=", 13)

s = lp("max", c, A, dir, d)
return(s)
}
```

4. Use allocation() function to calculate the optimal objective without the third constraint.

```
# alc1's optimal soultion(allocation), and objective value(revenue)
alc1 = allocation(c, 3, 10)
alc1$solution
```

```
##  [1] 0 3 0 1 0 0 3 0 0 3
```

```
alc1$objval
```

```
## [1] 0.456
```

```
# alc2's optimal soultion(allocation), and objective value(revenue)
alc2 = allocation(c, 10, 10)
alc2$solution
```

```
##  [1] 0 5 0 0 0 0 0 0 0 5
```

```
alc2$objval
```

```
## [1] 0.465
```

The Optimizer's Curse

1. Get the optimal objective value and the corresponding solution.

```
new_ROI_vec = c(4.9, 2.3, 2.4, 3.9, 4.4, 4.6, 2.6, 1.9, 3.7, 2.6)/100
alc3 = allocation(new_ROI_vec, 3, 10)
# They are different from using previous ROI vecto
alc3$solution
```

```
##  [1] 3 0 0 1 3 3 0 0 0 0
```

```
alc3$objval
```

```
## [1] 0.456
```

2. Disappointment of allocation

```
print(paste("disappointment for alc1:", round(alc1$solution %*% c - alc1$solution %*% ne
w_ROI_vec, 6), "M"))
```

```
## [1] "disappointment for alc1: 0.192 M"
```

```
print(paste("disappointment for alc2:", round(alc2$solution %*% c - alc2$solution %*% ne
w_ROI_vec, 6) , "M"))
```

```
## [1] "disappointment for alc2: 0.22 M"
```

3rd constraint is useful because it distributes the risk - if one of the large investments fails, that is okay because they are not that large anyways; they can't be over 3MM.

3. Find an allocation that dominates alc1, alc2, alc3 regarding the average ojbective values.

An allocation that dominates alc1, alc2, alc3 should have the best peformance on the average ROI weights we have.

To answer this question, we first compute the maximum return of the alc1, alc2, alc3 on the averaged ROI. Then try to find a maximum performance allocation on the averaged ROI, which is trained on the original ROI vector(our best information avaliable) with different upper_bound inputs.

```
load("Project1.Rdata")
avg_ROI = (ROI_vec1+ROI_vec2)/(2*100)
MaxReturnFirst3 = max(c(alc1$solution %*% avg_ROI,
                        alc2$solution %*% avg_ROI,
                        alc3$solution %*% avg_ROI))

print(paste("The maximum return among first three alcs",
            MaxReturnFirst3)  )
```

```
## [1] "The maximum return among first three alcs 0.36"
```

```
alc_domin = c()
return_domin = MaxReturnFirst3
upper_bound_domin = 0

for( i in c(1:20)){
  temp = allocation(ROI_vec1, i*0.5, 10)

  if(temp$solution %*% avg_ROI > return_domin){
    alc_domin = temp$solution
    return_domin = temp$solution %*% avg_ROI
    upper_bound_domin = i
  }
}
print(paste("The optimal upper bound is", upper_bound_domin,
      "yielding", return_domin))
```

```
## [1] "The optimal upper bound is 4 yielding 0.362"
```

```
# The optimal allocation
alc_domin
```

```
##  [1] 0 2 0 2 0 0 2 0 2 2
```

Multi-Period Allocation

1. What's the optimal allocation for the whole year?

The optimal allocation for each year is a 12 X 10 matrix where each row is the amount spent in the 10 channels.

```
# Column 1 through 10 are allocation for each platform for each month, column 11 are bud
get of that month, and column 12 are return on investments in millions.
year_alc = matrix(0, nrow = 12, ncol = 12)
budget_now = 10
for( i in c(1:12) ){
  month_alc = allocation(c(ROI_mat[i,]/100), 3, budget_now)
  objval = month_alc$objval
  year_alc[i, ] = c(month_alc$solution,budget_now,objval)
  budget_now = budget_now + month_alc$objval * 0.5
}
round(year_alc, 3)
```

```
##        [,1] [,2] [,3]  [,4]  [,5]  [,6]  [,7] [,8]  [,9] [,10]  [,11]
##  [1,] 3.000    0    0 1.333 0.000 0.000 2.667    0 0.000 3.000 10.000
##  [2,] 3.000    0    0 2.395 3.000 0.000 0.000    0 1.791 0.000 10.187
##  [3,] 0.000    0    0 3.000 0.000 3.000 1.390    0 3.000 0.000 10.390
##  [4,] 0.000    0    0 3.000 0.000 3.000 3.000    0 1.597 0.000 10.597
##  [5,] 1.804    0    0 0.000 0.000 0.000 3.000    0 3.000 3.000 10.804
##  [6,] 3.000    0    0 0.000 0.000 0.000 3.000    0 2.020 3.000 11.020
##  [7,] 0.000    0    0 3.000 2.248 0.000 3.000    0 3.000 0.000 11.248
##  [8,] 3.000    0    0 1.827 0.000 0.655 0.000    0 3.000 3.000 11.482
##  [9,] 1.363    0    0 3.000 0.000 3.000 0.000    0 3.000 1.363 11.726
## [10,] 0.000    0    0 3.000 0.000 3.000 3.000    0 0.000 2.955 11.955
## [11,] 3.000    0    0 2.056 0.000 1.113 3.000    0 0.000 3.000 12.169
## [12,] 3.000    3    0 0.428 3.000 0.000 0.000    0 0.000 3.000 12.428
##       [,12]
##  [1,] 0.373
##  [2,] 0.406
##  [3,] 0.414
##  [4,] 0.414
##  [5,] 0.432
##  [6,] 0.455
##  [7,] 0.469
##  [8,] 0.488
##  [9,] 0.459
## [10,] 0.428
## [11,] 0.517
## [12,] 0.517
```

2. What's the connection between the multi-period and previous single-period problem?

The multi-period problem can be solved as 12 single-period problem having budgets yielded from the previous period. Because we can't borrow from the future, and aren't able to change the ROI in the future. We just find the optmize allocation in this period to gain the maximum profit that can be carried to the next month.

3. Does the previous connection still hold?

The previous connection no longer holds. Because now there are more constraints to add in our A matrix for the allocation in last period now affect how we allocate in this period. Which makes the multi-period problem different from solving multiple single-period problem. In other words, we have to rewrite our allocation function to solve the new problem.