Demonstration of R’s Optimization Function

## Details

FRE 501 September 17, 2021

## Learning Outcomes

* Use R’s optim() function to maximize an expression without using matrices.
* Use R’s optim() function to maximize an expression using matrices.

## Objectives

Demonstrate the use of R’s optim() function to help students better understand how the optimization procedure works in Module 1B (softwood lumber) and Module 1C (tomatoes).

## Optimization without Matrices

Consider the simple quadratic function with two variables:

We can maximize Z by solving the pair of first order conditions:

and

This system of equations can be solved to give as the optimizing values of and .

If we want R to choose the values of and which maximize then we need to put and into a R vector. The vector has two elements, which in R language is and . We can rewrite the above objective function as

To write the R code the first step is to create an R function which contains this objective function.

objective1 <-function(z) {   
 W <- -1\*(z[1] + z[2] - 0.5\*z[1]\*z[1] - 0.5\*z[2]\*z[2] - 2\*z[1]\*z[2])  
 return(W)  
}

In the above function we multiplied the objective function by -1 because R’s optimization function is designed to minimize. We trick R into maximizing by telling it to minimize the negative of the objective function.

To maximize in the function we need to supply starting (guess) values for and . Let’s guess as the pair of starting values.

It is often useful to put upper and lower limits on the range of values for and . The most common limit is to restrict these variables to non-negative values only. If we don’t want any restrictions then we set negative infinity as the lower limit and positive infinity as the upper limit. For this example we will tell R to choose only non-negative values for and .

Coding R’s optimization function is very straight forward for simple problems. We feed into the function three things: (1) the name of the function to be minimized; (2) the vector of starting values; and (3) the upper and lower limits for the choice variables. In this example we will use

Coding this function in R gives:

optim( c(1,1), objective1, lower=0, upper=Inf)

## Warning in optim(c(1, 1), objective1, lower = 0, upper = Inf): bounds can only  
## be used with method L-BFGS-B (or Brent)

## $par  
## [1] 0.3333333 0.3333333  
##   
## $value  
## [1] -0.3333333  
##   
## $counts  
## function gradient   
## 3 3   
##   
## $convergence  
## [1] 0  
##   
## $message  
## [1] "CONVERGENCE: NORM OF PROJECTED GRADIENT <= PGTOL"

In the above results the top row, , gives the optimal values for and . These match the optimal values derived manually at the beginning of this section. The next row, , gives the value of the optimized objective function. We need to multiply this value by -1 to get the actual value because of the way we tricked R into maximizing rather than minimizing.

## Optimization with Matrices - Previous Problem

Consider again the objective function

I will let you verify that this function can be written in matrix notation as

where

To construct the objective function in R we need to use R’s method of multiplying matrices. Specifically, we use if we want to transpose the matrix , and use if we want to multiply two matrices together.

We first need to create the and matrices in R:

A <- matrix(c(1,0.5,0.5,1), nrow=2, ncol=2)  
B <- matrix(c(1,1), nrow=2, ncol=1)

The R function that we previously specified can be coded as

objective2 <-function(z) {   
 W <- -1\*(t(B)%\*%z - t(z)%\*%A%\*%z)  
 return(W)  
}

Using R’s optimization function is the same as before:

optim( c(1,1), objective2, lower=0, upper=Inf)

## Warning in optim(c(1, 1), objective2, lower = 0, upper = Inf): bounds can only  
## be used with method L-BFGS-B (or Brent)

## $par  
## [1] 0.3333333 0.3333333  
##   
## $value  
## [1] -0.3333333  
##   
## $counts  
## function gradient   
## 7 7   
##   
## $convergence  
## [1] 0  
##   
## $message  
## [1] "CONVERGENCE: NORM OF PROJECTED GRADIENT <= PGTOL"

As expected we get the same results as above.

## Optimization with Matrices - More Complex Problem

In Module 1C we will need to maximize a welfare function where the set of choice variables are in a matrix rather than a vector. Unfortunately, R’s optim() function only works with vectors. This means we must convert a matrix into a long vector before we can use the optim() function.The example below will demonstate this technique.

Let

and

Our objective function is

Note that is the same as that used above. We want to maximize using starting values .

The objective function in R can be written as

objective3 <-function(X) {   
 Xhat = matrix(X, 2, 2) # Reshape 4x1 vector into 2x2 matrix.  
 Z <- -1\*(t(B)%\*%Xhat%\*%B - t(B)%\*%Xhat%\*%Xhat%\*%B)  
 return(Z)  
}

The line of code in the above function is very important. The function accepts as input a 4 x 1 column vector of values and the line in question converts this vector into a 2 x 2 matrix. This is done because the optim() function can only work with the 4 x 1 column vector but the expression for within the function is designed to work with a 2 x 2 matrix.

We can now use R’s optim() function in the usual way. Specifically, we will feed into the function the function to be minimized, a vector of starting values, , and the non-negative restrictions on the choice variables.

optim( c(1,1,1,1), objective3, lower=0, upper=Inf)

## Warning in optim(c(1, 1, 1, 1), objective3, lower = 0, upper = Inf): bounds can  
## only be used with method L-BFGS-B (or Brent)

## $par  
## [1] 0.25 0.25 0.25 0.25  
##   
## $value  
## [1] -0.5  
##   
## $counts  
## function gradient   
## 4 4   
##   
## $convergence  
## [1] 0  
##   
## $message  
## [1] "CONVERGENCE: REL\_REDUCTION\_OF\_F <= FACTR\*EPSMCH"

We can see from the R output that the optimizing values are and the value of the objective function (after reversing the sign) is 0.5.