# Linear Programming

*Abhinav Anand, IIMB*

*2018/06/23*

## Background

Linear programming involves maximizing or minimizing a linear objective function subject to linear inequality constraints.

### Illustration 1:

We consider a trivial example: suppose the objective function is $f(x) = 2x$ and suppose that $x$ is constrained to be a positive number not more than 5. More formally,

$$\max 2x : x \leq 5, x > 0$$

While this is clearly an admissible linear program, it's fairly trivial to solve. Since $2x$ is linear and monotonic in $x$, its solution will occur at the end point of $x = 5$ where it attains its maximum of 10.

### Illustration 2:

For a linear program whose objective function has two variables, consider a classic portfolio analysis problem: bonds generate 5% returns, stocks generate 8% returns. The total budget is $1000. How much of each asset should be bought?

We can translate this problem into a linear programming problem:

$$\max 0.05b + 0.08s : b + s \leq 1000, b \geq 0, s \geq 0$$

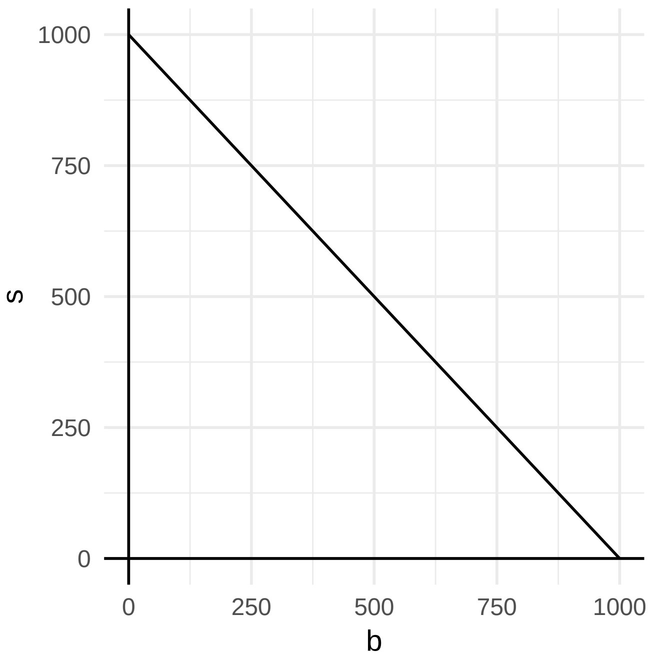
where $b, s$ respectively are the amount (in dollars) invested in bonds and stocks.

**The Feasible Set**

Any combination of bonds and stocks that satisfies the inequalities: $b, s \geq 0$ and $b + s \leq 1000$ is *feasible*. In the problem above, the feasible set is the following triangular region:

```
b <- 0:1000
s <- 1000 - b

ggplot(data.frame(cbind(b, s)), aes(b, s)) +
  geom_line() +
  geom_vline(xintercept = 0) + #vertical line
  geom_hline(yintercept = 0) + #horizontal line
  theme_minimal()
```




In the problem above and more generally, in any linear programming problem, the feasible set is an intersection of *half-spaces*. Clearly, the more constraints we have, the smaller the feasible set is. The feasible set in general can be of three varieties:

1. It is empty. In this case there is no solution.
2. It is not empty but the objective function is unbounded over it. $(f(x) \in$

$\{\infty, -\infty\}$.)

3. It is not empty *and* the objective function is bounded over it. ($f(x) \in (\infty, -\infty)$.)

Only the last case has practical value.

### Finding the Maximum

In principle, to find the maximum, all we need to do is to evaluate the objective function at all feasible points; and then see which point yields the maximum. Clearly, this is not practical since there are a continuum of points in this case.

The key idea is to consider a sequence of contour lines for the objective function. Here we consider the family of lines $0.05b + 0.08s = \{20, 35, 50, \ldots\}$ etc. Then we consider which of these intersect with our feasible set. The maximum of the objective function will be attained at a *corner point*. (Can you see why? Hint: See illustration 1.)

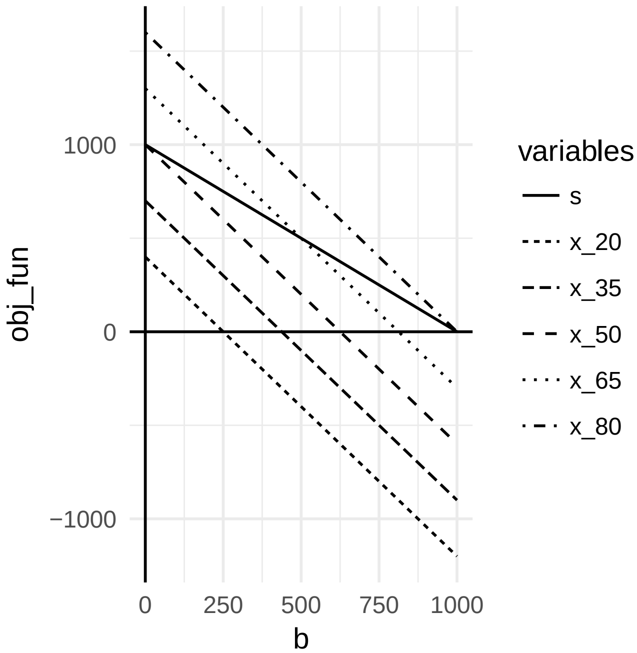
```
y_s <- 0:1000

x_20 <- (20-0.08*y_s)/0.05 #contour line: .05b+.08s=20
x_35 <- (35-0.08*y_s)/0.05 #contour line: .05b+.08s=35
x_50 <- (50-0.08*y_s)/0.05
x_65 <- (65-0.08*y_s)/0.05
x_80 <- (80-0.08*y_s)/0.05

data_plot_w <- cbind(b, s, x_20,
                     x_35, x_50,
                     x_65, x_80) %>%
  dplyr::as_tibble() #wide format

data_plot_l <- tidyr::gather(data_plot_w,
                             s:x_80,
                             key = "variables",
                             value = "obj_fun") #long format
```

```r
ggplot(data = data_plot_l, aes(b, obj_fun)) +
  geom_line(aes(linetype = variables)) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  theme_minimal()
```




This plot suggests that the optimal cannot occur at a strictly interior point in the feasible set and that it must occur at some corner point.[1] This is simple to see since the contour lines increase steadily upwards. This yields a tempting tentative solution which computes the objective function at all (finitely many) corners and just compares all values to find the optimal. However, for general problems, there could be several million corner points and this approach does not scale. Hence we prefer to reach the maximum in a more systematic way.

### The Simplex Idea

Devised by George Dantzig, the simplex method relies on a simple insight: for a canonical cost minimization program, look for an optimal solution by starting from

[1]In general the solution could occur along some edge as well.

a corner and visiting some accessible corner with lower cost until we reach a corner for which there is no accessible corner with cost any lower.

### Corners

In Illustration 1, we have to maximize $2x : x \leq 5, x > 0$. (What (if any) are the corner(s) here?) In one-dimension ($x \in \mathbb{R}$), corners are merely end points. The problem above has a corner at $x = 5$. (Is $x = 0$ also a corner?)

In two dimensions, corners are intersection points of two lines. This is clearly the case in Illustration 2 which possesses three corners: (0,0), (1000,0) and (0,1000).

In three dimensions corners are intersection points of three planes—for example, ceiling corners are intersections of the three walls (planes).

Similarly in $n$ dimensions corners are intersections of $n$ hyperplanes. In general for a linear program there are $\binom{n+m}{n}$ possible corners. Hence brute-force evaluation of the objective function over all corners is rendered impractical.

## Linear Programming: Simplex

Consider the canonical cost minimization linear program for two variables, say $x_1, x_2$:

$$\min\{c_1x_1 + c_2x_2\} :$$

$$a_{11}x_1 + a_{12}x_2 \geq b_1$$

$$a_{21}x_1 + a_{22}x_2 \geq b_2$$

$$x_1, x_2 \geq 0$$

Or in vector notation:

$$\min c^\top x :$$

$$Ax \geq b, x \geq 0$$

where $c = (c_1, c_2), x = (x_1, x_2), b = (b_1, b_2)$ and the matrix $A$ is the $2 \times 2$ matrix of $a_{ij}$.

In general, $c = (c_1, c_2, \ldots, c_n), x = (x_1, x_2, \ldots, x_n), b = (b_1, b_2, \ldots, b_m)$; and $A$ is $m \times n$. However the general form remains the same:

$$\min c^\top x :$$

$$Ax \geq b, x \geq 0$$

## Locating a Starting Corner

In the general formulation of the canonical cost minimization linear program, where $x \in \mathbb{R}^n$, at a typical corner there are $n$ edges that connect it to an adjacent corner. The simplex idea is to compute the values of the objective function at the accessible corners and choose the next one that has cost lower than that of the current corner. Then at the next corner, repeat the process and choose another less costly corner. Eventually there will be a special corner all of whose adjacent corners are more expensive. This is the optimal as reached by the simplex method.

Corners are simply the intersection points of inequalities. In general there are $n$ inequalities from the $x \geq 0$ constraints and $m$ inequalities from $Ax \geq b$ leading to a total of $n + m$ inequalities.

### Slack Variables

In Illustration 2, the inequality could be re-written after the introduction of an auxiliary "slack" variable $w$:

$$b + s \leq 1000, b, s \geq 0 \iff b + s + w = 1000, b, s, w \geq 0$$

This transforms the inequalities $Ax \geq 0$ to $A\tilde{x} = 0$ where $\tilde{x} = (x, w)$; while retaining the other inequalities in the form $(x, w) = \tilde{x} \geq 0$. As may be seen, the slack variables $w = Ax - b$ turn "loose" inequalities into "tight" inequalities (equations). The new costs become $\tilde{c} = (c, 0)$.

The new linear program now becomes:

$$\max (c, 0)^\top (x, w) : Ax - Iw = b, (x, w) \geq 0$$

or,

$$\max \tilde{c}^\top \tilde{x} : [A - I] \begin{bmatrix} x \\ w \end{bmatrix} = b, \begin{bmatrix} x \\ w \end{bmatrix} \geq 0$$

or, simply without loss of generality,

$$\max c^\top x : Ax = b, x \geq 0$$

This is a general canonical form of a linear program. The equality constraints are $Ax = b$ and the inequality constraints are $x \geq 0$. There are still $n + m$ inequalities but the slack variables have moved them from $Ax \geq b$ to $x \geq 0$. The augmented vector now is $x = (x_1, \ldots, x_n, w_1, \ldots, w_m)$.

#### Starting Corner

Translating the idea of a corner into algebra, we say that a corner of the feasible set occurs when exactly $n$ out of the $n + m$ components of $x$ equal 0. These are exactly the *basic feasible* solutions of $Ax = b$. Basic variables are computed by solving $Ax = b$ and free free components of $x$ are set to 0.[2]

Once the starting corner is found, we move along the edge to an adjacent corner with lower cost.

## Duality

Every linear programming problem (called the *primal* problem, with $x \in \mathbb{R}^n$) can be converted into a *dual* problem (with $y \in \mathbb{R}^m$) the solution of which provides an upper bound for the solution of the primal.

[2]'Basic', if $n$ components equal 0; and 'feasible' if $x \geq 0$. While solving $Ax = b$ (with $A_{(m+n)\times n}$ and $b_{m\times 1}$) after Gaussian elimination there will be left exactly $m$ pivots and $n$ free variables. We stipulate the free variables to be all 0; and confirm if all pivots are at least 0. If so, we have a basic feasible solution.

**Primal** : $\max\{c^\top x\} : Ax \leq b, x \geq 0$

The dual problem contains the same fundamental building blocks $A, b, c$ but reverses everything: in the primal, $c$ captures the costs while $b$ contains the constraints. In the dual problem, $b$ enters the objective function and $c$ forms the constraints. Also, if the primal is a maximization program, its dual is a minimization program.

**Dual**: $\min\{b^\top y\} : A^\top y \geq c, y \geq 0$

There are two fundamental ideas in duality:

1. The dual of the dual (the so-called *bidual*) is the primal. (Can you see how?)
2. Every feasible solution for a primal gives a bound on the optimal value for the dual.

## Duality Theorem

When primal and dual are both feasible with optimal vectors $x^*, y^*$ respectively, the maximum of the primal $c^\top x^*$ is equal to the minimum of the dual $b^\top y^*$.

### Weak Duality Theorem

If $x$ and $y$ are feasible in the primal and dual respectively, then $c^\top x \leq b^\top y$.

In plain language, the maximal primal value is no more than the minimal dual value for feasible points ($x \in \mathbb{R}^n, y \in \mathbb{R}^m$).

*Sketch of Proof*: Suppose $x \in \mathbb{R}^n, y \in \mathbb{R}^m$ are feasible. Then $x : Ax \leq b$ and $y : A^\top y \geq c$. Also, since $x, y \geq 0$, we can construct inner products and preserve the inequalities.[3]

1. $y^\top Ax \leq y^\top b$, from the primal constraints
2. $x^\top A^\top y \geq x^\top c$, from the dual constraints. Since inner products are commutative, we can rewrite it as: $y^\top (Ax) \geq c^\top x$.

Hence, $y^\top b \geq y^\top (Ax) \geq c^\top x$.

[3] Recall that for $a, b \in \mathbb{R}^n$, the inner product is $a^\top b = b^\top a = \sum a_i b_i$.

Weak duality implies that if the primal (max) is unbounded then the dual (min) is infeasible. Likewise, if the dual is unbounded, then the primal must be infeasible.[4]

To each variable in the primal space corresponds a half-space (a constraint/inequality) to satisfy in the dual space. To each constraint to satisfy in the primal space there exists a variable in the dual space. The coefficients that bound the inequalities in the primal space are used to compute the objective in the dual space. Both the primal and the dual problems make use of the same matrix.

Additionally, since each inequality can be replaced by an equality and a slack variable, this means each primal variable corresponds to a dual slack variable, and each dual variable corresponds to a primal slack variable. This relation allows us to speak about *complementary slackness.*

#### Complementary Slackness

Take $x = (x_1, \dots, x_n)$ as primal feasible and $y = (y_1, \dots, y_m)$ as dual feasible. Denote by $(w_1, \dots, w_m)$ the corresponding primal slacks and by $(z_1, \dots, z_n)$ the corresponding dual slacks. Then $x^*$ and $y^*$ are optimal for their respective problems if and only if

1. $x_j^* z_j = 0, j \in \{1, \dots, n\}$
2. $y_i^* w_i = 0, i \in \{1, \dots, m\}$

Put simply, if some component of the primal slack is positive, then the corresponding component of the dual variable must be zero; and if some component of the dual slack is positive then the corresponding primal variable is zero.

It's easy to see why this is necessary: if there is a primal slack in maximization, (i.e., there are "leftovers"), then additional units of that variable have no value. Likewise, if there is slack in the dual minimum, there must be no leftovers in the primal.

## Computation: Linear Programming

Let's reconsider the old portfolio optimization program:

[4] However, it is possible for both the dual and the primal to be infeasible.

$$\max 0.05b + 0.08s : b + s \leq 1000, b \geq 0, s \geq 0$$

This may be rewritten as:

$$\max\,[0.05, 0.08] \begin{bmatrix} b \\ s \end{bmatrix} :$$

$$[1, 1] \begin{bmatrix} b \\ s \end{bmatrix} \leq 1000$$

$$\begin{bmatrix} b \\ s \end{bmatrix} \geq 0$$

To solve such problems, we use the library `lpSolve()` containing the function `lp()`. The general syntax for `lp()` is:

```
lp(direction = "max", #default is "min"
   objective.in = ..., #the objective function coefficients
   const.mat = ..., #constraint matrix
   const.dir = ..., #directions of constraints: >=, <= etc.
   const.rhs = ...  #the constraint Right Hand Side (RHS)
   )
```

This is a well tested library and can solve small to mid-sized problems, i.e., with several hundred variables, very efficiently. The fact that the variables have to be greater than 0 is always implicitly assumed and does not need to be stated.

Let's try to solve the LP above using the function `lp()` from the package `lpSolve()`.

```
optim_port <- lpSolve::lp(direction = "max",
                          objective.in = c(0.05, 0.08),
                          const.mat = matrix(c(1,1), #must be matrix
                                             nrow = 1,
                                             byrow = T
                                             ),
                          const.dir = c("<=", "<="),
                          const.rhs = 1000
                          )
```

```
## Warning in rbind(const.mat, const.dir.num, const.rhs): number of columns of
## result is not a multiple of vector length (arg 2)
```

```
optim_port$solution
```

```
## [1]    0 1000
```

```
optim_port$objval
```

```
## [1] 80
```

It's easy to have guessed the solution of the above program even without computing it formally. (How? Is there a simple way to see this?)

Here is a more involved program:

$$\max 25x_1 + 20x_2 :$$

$$20x_1 + 12x_2 \leq 1800$$

$$1/15x_1 + 1/15x_2 \leq 8$$

$$x_1, x_2 \geq 0$$

```
optim_lp <- lpSolve::lp(direction = "max",
                        objective.in = c(25, 20),
                        const.mat = matrix(c(20, 12, 1/15, 1/15),
                                           nrow = 2,
                                           byrow = T),
                        const.dir = c("<=", "<="),
                        const.rhs = c(1800, 8)
                        )

optim_lp$solution
```

```
## [1] 45 75
```

```
optim_lp$objval
```

```
## [1] 2625
```

# Application: Quantile Regressions

Quantile regression aims to estimate the conditional median (or other quantiles) of the dependent variable as opposed to its conditional mean, which is the objective of an ordinary least squares exercise. Quantile estimates, somewhat unsurprisingly, are more robust measures of conditional dependence. Median regressions for large data sets were unpopular until recently since they are quite tedious compared to the least squares method.

## Mean and Medians as Optimization Programs

### Argmin of Sum of Squared Deviations

Given observations $b_1, \ldots, b_m$, find $b$ such that the sum of observations' squared deviations from $b$ is minimum:

$$\min_b (b - b_1)^2 + \ldots + (b - b_m)^2$$

$$\min_b \sum_{i=1}^{m} (b - b_i)^2$$

Then

$$f(x) = \sum_{i=1}^{m} (x - b_i)^2$$

$$f'(x) = \sum_{i=1}^{m} 2(x - b_i)$$

$$f''(x) = 2m > 0$$

Hence there is a minimum $x^*$ such that

$$f'(x) = \sum_{i=1}^{m} 2(x - b_i) = 0 \Rightarrow x^* = \frac{\sum_{i=1}^{m} b_i}{m}$$

Hence the mean $x^* = \frac{\sum_{i=1}^{m} b_i}{m}$ minimizes the sum of the squared distance between itself and the observations.

**Argmin of Sum of Absolute Deviations**

Given observations $b_1, \ldots, b_m$, find $b$ such that the sum of observations' absolute deviations from $b$ is minimum:

$$\min_b |b - b_1| + \ldots + |b - b_m|$$

$$\min_b \sum_{i=1}^{m} |b - b_i|$$

Then

$$g(x) = \sum_{i=1}^{m} |x - b_i|$$

$$g'(x) = \sum_{i=1}^{m} sgn(x - b_i) \cdot 1$$

$$g'(x) = |\{i : b_i < x\}| - |\{i : b_i \geq x\}|$$

Hence there is a minimum $x_*$ such that

$$g'(x) = 0 \Rightarrow x_* = b_{\frac{1+m}{2}}$$

Hence the median $x_* = b_{\frac{1+m}{2}}$ minimizes the sum of the absolute distance between itself and the observations.

## Regressions: Least Squares and Median

In general, a least squares regression is:

$$y = X\beta + u; \text{and } \mathbb{E}(u|X) = 0$$

and the problem is to find $\beta^*$ such that the sum of the squares of the residuals are minimum:

$$\min_\beta \|u\|_2 = \min_\beta \|y - X\beta\|_2$$

where $\|u\|_2 := \sqrt{\sum_{i=1}^{m} u_i^2}$ is the *Euclidean* norm (also called the $L^2$ norm).

Analogously, a *median* regression is:

$$y = X\beta_{0.50} + u; \text{and } \mathbb{Q}_{0.50}(u|X) = 0$$

where $Q_\tau(Y|X)$ is the $\tau_{\text{th}}$ conditional quantile of Y given X and the problem is to find $\beta^*_{0.50}$ such that the sum of the absolute values of the residuals are minimum:

$$\min_{\beta_{0.50}} \|u\|_1 = \min_{\beta_{0.50}} \|y - X\beta_{0.50}\|_1$$

where $\|u\|_1 := \sum_{i=1}^{m} |u_i|$ is the $L^1$ norm.

Hence the median regression problem is:

$$\min_{\beta_{0.50}} \sum_{i=1}^{m} |u_i| = \sum_{i=1}^{m} |y_i - X_i^\top \beta_{0.50}|$$

**Quantile Regression: The General Case**

In the same way as the median, a general quantile $\tau \in (0,1)$ for a sequence of observations $y_1, \ldots, y_m$ can be computed as the minimum of the following program:

$$\min_{y_\tau}\{\sum_{i:y_i > y_\tau} |y_i - y_\tau| \cdot \tau + \sum_{i:y_i \leq y_\tau} |y_i - y_\tau| \cdot (1-\tau)\}$$

Hence the general quantile regression is:

$$\min_{\beta_\tau}\{\sum_{i:y_i > \beta_\tau} \tau \cdot |y_i - X_i^\top \beta_\tau| + \sum_{i:y_i \leq \beta_\tau} (1-\tau) \cdot |y_i - X_i^\top \beta_\tau|\}$$

We can recast the above as follows:

$$u_i := \max(y_i - X_i^\top \beta_\tau, 0)$$

$$v_i := \min(y_i - X_i^\top \beta_\tau, 0)$$

$$b_+ := \max(\beta_\tau, 0)$$

$$b_- := \min(\beta_\tau, 0)$$

giving the generic following linear program:

$$\min \sum_{i=1}^{m} (\tau \cdot u_i + (1-\tau) \cdot v_i) :$$

$$u_i, v_i \geq 0, i \in \mathbb{N}$$

$$b_+, b_- \geq 0$$

$$y_i = X_i^\top (b_+ - b_-) + u_i - v_i$$

Hence in particular, for the median regression, with $\tau = 0.50$, we have the program:

$$\min \sum_{i=1}^{m} |y_i - y|$$

$$\min \sum_{i=1}^{m} (a_i + b_i) :$$

$$y_i - y = a_i - b_i$$

$$a_i, b_i \geq 0$$

### Computation: Median Regressions

Find the median of a sequence of observations (residuals) which (we know now) is the solution for the median regression problem above. We assume the observations (residuals) are normally distributed.

```
num <- 201 #number of points
u <- rnorm(num, 0, 1) #normal 0, 1 residuals

# One constraint per row: a[i], b[i] >= 0
A1 <- cbind(diag(2*num),0)

# a[i] - b[i] = y[i] - y
A2 <- cbind(diag(num), -diag(num), 1)

quant_med <- lpSolve::lp("min",
                         c(rep(1,2*num),0),
                         rbind(A1, A2),
```

```r
                         c(rep(">=", 2*num), rep("=", num)),
                         c(rep(0,2*num), u)
                         )

quant_med$solution %>% tail()
```

```
## [1] 1.09864620 2.97629643 0.00000000 0.78742939 0.00000000 0.07821771
```

```r
median(u)
```

```
## [1] 0.07821771
```