LpSolve: A Package to Solve Linear Programming

Jiyanglin Li

School of Statistics and Management
Shanghai University of Finance and Economics

December 10, 2015

Outline

Linear Programming

$$\begin{aligned}
\max z &= c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{s.t.} \quad & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \quad \text{(Constraint 1)} \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \quad \text{(Constraint 2)} \\
& \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \quad \text{(Constraint m)} \\
& x_j \geq 0 (j = 1, 2, ..., n)
\end{aligned}$$

Remark:

- The direction of the constraint can be $<, \leq, =, \geq, >$.
- The variables can be unrestricted-in-sign.

Standard Form of Linear Programming

$$\max z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\begin{aligned}
\text{s. t.} \qquad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \\
x_j \geq 0(j = 1, 2, ..., n)&
\end{aligned}$$

How to convert a LP to Standard Form

Assume $b_i \geq 0, \forall i$, otherwise we can multiple all the terms in the equation by -1.

- If the constraints is $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$, add a slack variable $s_i \geq 0$, then the original constraint is equivalent to $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + s_i = b_i$.
- If the constraints is $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i$, add an excess variable $e_i \geq 0$, then the original constraint is equivalent to $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - e_i = b_i$.
- If x_i is unrestricted-in-sign, we can replace x_i with $x_i^{'} - x_i^{''}$, where $x_i^{'}, x_i^{''} \geq 0$.

Pure Integer Programming

$$
\begin{aligned}
\max z &= c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{s. t.} \quad & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\
& \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\
& x_j \geq 0 (j = 1, 2, ..., n); x_j \in \mathbb{Z}, \forall j
\end{aligned}
$$

Mixed Integer Programming

$$
\begin{aligned}
\max z &= c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{s. t.} \quad & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\
& \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\
& x_j \geq 0 (j = 1, 2, ..., n); x_j \in \mathbb{Z}, \forall j \in \mathcal{I} \\
& \mathcal{I} \subset \{1, 2, \cdots, n\}
\end{aligned}
$$

What can LpSolve do?

LpSolve is a package for solving linear, integer and mixed integer programs.

- lp
- lp.assign
- lp.transport
- make.q8 (8-queens problem)
- print.lp

lp

lp (direction = "min", objective.in, const.mat, const.dir, const.rhs, transpose.constraints = TRUE, int.vec, presolve=0, compute.sens=0, binary.vec, all.int=FALSE, all.bin=FALSE, scale = 196, dense.const, num.bin.solns=1, use.rw=FALSE)

lp.transport & lp.assign

- lp.transport (cost.mat, direction="min", row.signs, row.rhs, col.signs, col.rhs, presolve=0, compute.sens=0, integers = 1:(nc*nr))
- lp.assign (cost.mat, direction = "min", presolve = 0, compute.sens = 0)

8-Queens Problem

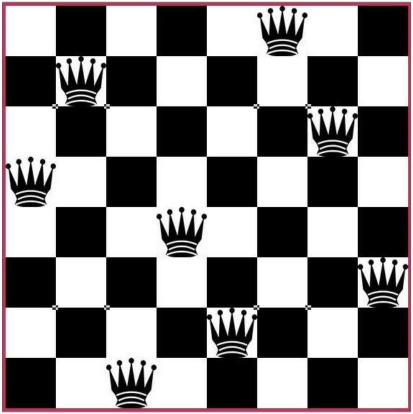

Figure: One solution to the eight queens puzzle.

make.q8

- make.q8()

Description:

Generate sparse constraint matrix for 8-queens problem

print.lp

```
print(x, ...)
```

A simple Linear Programming Problem

Example 1

$$\begin{aligned} \max z = 4x_1 + 3x_2 & \\ \text{s. t.} \qquad x_1 + x_2 &\leq 40 \\ 2x_1 + x_2 &\leq 60 \\ x_1, x_2 &\geq 0 \end{aligned}$$

A simple Linear Programming Problem I

```
> f.obj <- c(4,3)
> f.con <- matrix (c(1,2,1,1), nrow=2)
> f.dir <- c("<=", "<=")
> f.rhs <- c(40,60)
> b = lp("max", f.obj, f.con, f.dir, f.rhs)
> print(b)
Success: the objective function is 140
> b$solution
[1] 20 20
> b$objval
[1] 140
> b$x.count
[1] 2
```

A simple Linear Programming Problem II

```
> b$direction # Optimization direction
[1] 1
> b$const.count
[1] 2
> b$objective
[1] 4 3
> b$constraints
              [,1] [,2]
                 1    2
                 1    1
const.dir.num    1    1
const.rhs       40   60
```

- Remark: the value of const.dir.num $1 :<, <=; 2 :>, >=; 3 :=$.

A simple Linear Programming Problem

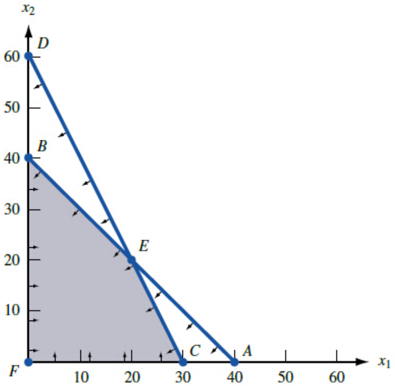

Figure: Feasible Region.

Unbounded Linear Programming

Example 2

$$
\begin{aligned}
\max z = 50x_1 + 100x_2 \\
\text{s. t.} \qquad 7x_1 + 2x_2 \geq 28 \\
2x_1 + 12x_2 \geq 24 \\
x_1, x_2 \geq 0
\end{aligned}
$$

Unbounded Linear Programming

```
> f.obj <- c(50,100)
> f.con <- matrix (c(7,2,2,12), nrow=2)
> f.dir <- c(">=", ">=")
> f.rhs <- c(28,24)
> (b=lp ("max", f.obj, f.con, f.dir, f.rhs))
Error: status 3
> b$solution
[1] 0 0
> b$objval
[1] 0
```

Unbounded Linear Programming Problem

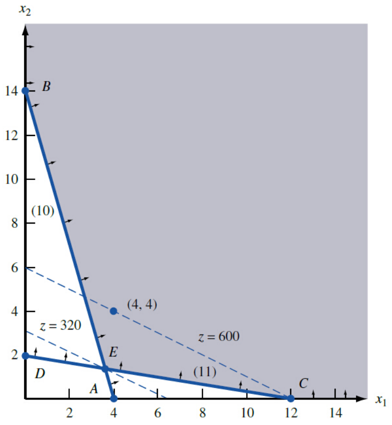

Figure: Unbounded Feasible Region.

Infeasible Linear Programming

Example 3

$$
\begin{aligned}
\max z &= 3x_1 + 2x_2 \\
\text{s. t.} \quad \frac{1}{40}x_1 + \frac{1}{60}x_2 &\leq 1 \\
\frac{1}{50}x_1 + \frac{1}{50}x_2 &\leq 1 \\
x_1 &\geq 30 \\
x_2 &\geq 20 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

Infeasible Linear Programming

```
> f.obj <- c(3,2)
> f.con <- matrix (c(1/40,1/50,1,0,1/60,1/50,0,1),
+                    nrow=4)
> f.dir <- c("<=","<=",">=", ">=")
> f.rhs <- c(1,1,30,20)
> (b=lp ("max", f.obj, f.con, f.dir, f.rhs))
Error: no feasible solution found>
> b$status
[1] 2
> b$solution
[1] 0 0
> b$objval
[1] 0
```

Infeasible Linear Programming

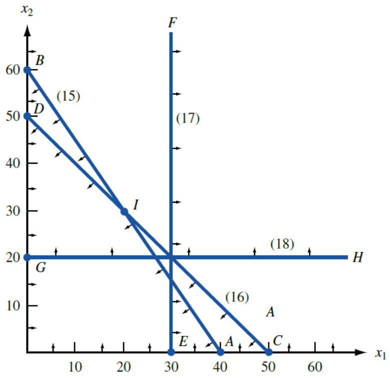

Figure: Empty Feasible Region((Infeasible LP)).

Transportation Problem

Example 4(Powerco Problem)

Shipping Costs, Supply, and Demand for Powerco

From	To: City 1	City 2	City 3	City 4	Supply (million kwh)
Plant 1	\$8	\$6	\$10	\$9	35
Plant 2	\$9	\$12	\$13	\$7	50
Plant 3	\$14	\$9	\$16	\$5	40
Demand (million kwh)	45	20	30	30	

Transportation Problem

```
> costs <- matrix (c(8,6,10,9,9,12,13,7,14,9,16,5),
+                   byrow=TRUE,nrow=3)
> row.signs <- rep ("<=", 3)
> row.rhs <- c(35,50,40)
> col.signs <- rep (">=", 4)
> col.rhs <- c(45,20,30,30)
> b=lp.transport(costs, "min", row.signs, row.rhs,
+                col.signs, col.rhs)
> b$solution
     [,1] [,2] [,3] [,4]
[1,]    0   10   25    0
[2,]   45    0    5    0
[3,]    0   10    0   30
> b$objval
[1] 1020
```

Transportation Problem

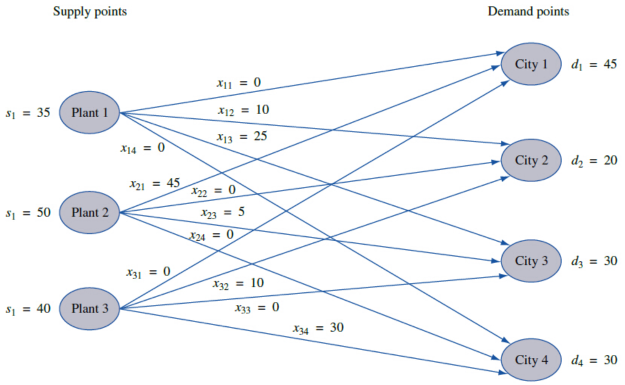

Figure: Graphical Representation of Powerco Problem and Its Optimal Solution.

Assignment Problem

Example 5(Machine Assignment Problem)

Setup Times for Machineco

	Time (Hours)			
Machine	Job 1	Job 2	Job 3	Job 4
1	14	5	8	7
2	2	12	6	5
3	7	8	3	9
4	2	4	6	10

Assignment Problem

```
> assign.costs<-matrix(c(14,5,8,7,2,12,6,5,
+ 7,8,3,9,2,4,6,10), ncol=4, byrow=TRUE)
> b = lp.assign(assign.costs)
> b$solution
     [,1] [,2] [,3] [,4]
[1,]    0    1    0    0
[2,]    0    0    0    1
[3,]    0    0    1    0
[4,]    1    0    0    0
> b$objval
[1] 15
```

Sensitivity Analysis

Once the optimal solution is found, we also hope to know the range of the objective function coefficient that will keep the current basis optimal.
Remark: more specific explanation of sensitivity analysis can be found in WL Winston & JB Goldberg(2004).

Sensitivity Analysis(Continued with Example 1)

```
> f.obj <- c(4,3)
> f.con <- matrix (c(1,2,1,1), nrow=2)
> f.dir <- c("<=", "<=")
> f.rhs <- c(40,60)
> b=lp ("max", f.obj, f.con, f.dir, f.rhs,
+       compute.sens=TRUE)
> b$sens.coef.from
[1] 3 2
> b$sens.coef.to
[1] 6 4
```

Pure Integer Programming

Example 6

$$
\begin{aligned}
\max z &= 8x_1 + 5x_2 \\
\text{s. t.} \qquad x_1 + x_1 &\le 6 \\
9x_1 + 5x_2 &\le 45 \\
x_1, x_2 \ge 0;\, x_1, x_2 &\text{ integer}
\end{aligned}
$$

Pure Integer Programming

```
> f.obj <- c(8,5)
> f.con <- matrix (c(1,9,1,5), nrow=2)
> f.dir <- c("<=", "<=")
> f.rhs <- c(6,45)
> b=lp ("max", f.obj, f.con, f.dir, f.rhs,all.int=TRUE)
> ### all.int=TRUE can be substituted by int.vec=1:2
> b$solution
[1] 5 0
> b$objval
[1] 40
```

- Remark: We can get sensitivities in the integer case, but they're harder to interpret.

Mixed Integer Programming

Example 7

$$
\begin{aligned}
\max z &= 2x_1 + x_2 \\
\text{s. t.} \quad & 5x_1 + x_2 \leq 8 \\
& 2x_1 + x_2 \leq 3 \\
& x_1, x_2 \geq 0; x_1 \text{ integer}
\end{aligned}
$$

Mixed Integer Programming

```
> f.obj <- c(2,1)
> f.con <- matrix (c(5,1,2,1), nrow=2)
> f.dir <- c("<=", "<=")
> f.rhs <- c(8,3)
> b=lp ("max", f.obj, f.con, f.dir, f.rhs,int.vec=1)
> b$solution
[1] 1.0 1.5
> b$objval
[1] 3.5
```

0-1 Integer Programming

Example 8

$$
\begin{aligned}
\max z = 16x_1 + 22x_2 + 12x_3 + 8x_4 \\
\text{s. t.} \qquad 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\
x_i = 0 \text{ or } 1, i = 1, 2{,}3, 4
\end{aligned}
$$

0-1 Integer Programming

```
> f.obj <- c(16,22,12,8)
> f.con <- matrix (c(5,7,4,3), nrow=1)
> f.dir <- "<="
> f.rhs <- 14
> b=lp ("max", f.obj, f.con, f.dir, f.rhs,all.bin=1)
> ### all.bin=TRUE can be substituted by binary.vec=1:2
> b$solution
[1] 0 1 1 1
> b$objval
[1] 42
```

Number of solutions $>$ 1 case

Example 9

$$\begin{aligned} \max z &= x_1 + x_2 + x_3 \\ \text{s. t.} \quad & x_1 + x_2 + x_3 \leq 1 \\ x_i &= 0 \text{ or } 1, i = 1, 2,3 \end{aligned}$$

Number of solutions > 1 case

```
> f.obj = c(1,1,1)
> f.con = matrix(c(1,1,1),nrow=1)
> f.dir = "<="
> f.rhs = 1
> x = lp ("max", f.obj, f.con, f.dir, f.rhs, all.bin=TRUE,
+         num.bin.solns=3)
> x$solution
 [1]  0  0  1  0  1  0  1  0  0 -1
> x$num.bin.solns
[1] 3
> y = lp ("max", f.obj, f.con, f.dir, f.rhs, all.bin=TRUE,
+         num.bin.solns=2)
> y$solution
[1] 0 0 1 0 1 0 1
> y$num.bin.solns
[1] 2
```

Eight Queens Problem I

```
> chess.obj <- rep (1, 64)
> q8 <- make.q8 ()
> chess.dir <- rep (c("=", "<="), c(16, 26))
> chess.rhs <- rep (1, 42)
> b = lp ('max', chess.obj, , chess.dir, chess.rhs,
+          dense.const = q8, all.bin=TRUE,
+          num.bin.solns=100)
> b$num.bin.solns
[1] 92
> a = matrix(b$solution[1:64],ncol=8,byrow=T)
```

Eight Queens Problem II

```
> a
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0    0    0    1    0    0    0    0
[2,]    1    0    0    0    0    0    0    0
[3,]    0    0    0    0    1    0    0    0
[4,]    0    0    0    0    0    0    0    1
[5,]    0    1    0    0    0    0    0    0
[6,]    0    0    0    0    0    0    1    0
[7,]    0    0    1    0    0    0    0    0
[8,]    0    0    0    0    0    1    0    0
```

Eight Queens Problem III

```
> q8
    const.ctr
 [1,]         1  1 1
 [2,]         1  2 1
 [3,]         1  3 1
 [4,]         1  4 1
 [5,]         1  5 1
 [6,]         1  6 1
 [7,]         1  7 1
 [8,]         1  8 1
```

Remark: There are 252 rows in this matrix, we only show the first 8 rows here.

Thanks!