R4 Optimization

Author: Richard T. Watson

Date: 2020-04-14

```
library(lpSolveAPI)
```

a.

Solve a transportation problem.

Transportation cost minimization

```
#model with 0 constraints and 16 decision
lpmodel <- make.lp(0,16)
# Shipping cost per unit laid out for ease of checking
set.objfn(lpmodel, c(15,20,25,15,
                     25,15,20,10,
                     15,30,20,20,
                     30,30,35,45))
# Supply constraints
add.constraint(lpmodel, c(1,1,1,1,
                          0,0,0,0,
                          0,0,0,0,
                          0,0,0,0), "<=", 10)
add.constraint(lpmodel, c(0,0,0,0,
                          1,1,1,1,
                          0,0,0,0,
                          0,0,0,0), "<=", 11)
add.constraint(lpmodel, c(0,0,0,0,
                          0,0,0,0,
                          1,1,1,1,
                          0,0,0,0), "<=", 19)
add.constraint(lpmodel, c(0,0,0,0,
                          0,0,0,0,
                          0,0,0,0,
                          1,1,1,1), "<=", 15)
# Demand constraints
add.constraint(lpmodel, c(1,0,0,0,
                          1,0,0,0,
                          1,0,0,0,
                          1,0,0,0), ">=", 11)
```

```
add.constraint(lpmodel, c(0,1,0,0,
                          0,1,0,0,
                          0,1,0,0,
                          0,1,0,0), ">=", 18)
add.constraint(lpmodel, c(0,0,1,0,
                          0,0,1,0,
                          0,0,1,0,
                          0,0,1,0), ">=", 16)
add.constraint(lpmodel, c(0,0,0,1,
                          0,0,0,1,
                          0,0,0,1,
                          0,0,0,1), ">=", 10)
#row names (constraints)
RowNames<- c("Alpha","Beta","Gamma","Delta","Phi","Chi","Psi","Omega")
#columns (decision variables)
ColNames<- c("AlphatoPhi","AlphatoChi", "AlphatoPsi", "AlphatoOmega", "BetatoPhi", "BetatoChi","BetatoPs
dimnames(lpmodel)<- list(RowNames, ColNames)
write.lp(lpmodel, "lpmodel.lp", type= c("lp"), use.names = c(TRUE,TRUE))
solve(lpmodel)
```

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

Minimum cost

```
get.objective(lpmodel) # this is the minimum cost
```

```
## [1] 1090
```

Solution

```
mat <-  t(matrix(get.variables(lpmodel),4,4)) # transpose results matrix so row is source
rows <-  c("Alpha","Beta","Gamma","Delta")
cols <-  c("Phi","Chi","Psi","Omega")
dimnames(mat) <-  list(rows, cols)
mat
```

```
##       Phi Chi Psi Omega
## Alpha   8   0   0     2
## Beta    0   3   0     8
## Gamma   3   0  16     0
## Delta   0  15   0     0
```

Shadow price analysis

```
get.variables(lpmodel)
```

```
##  [1]  8  0  0  2  0  3  0  8  3  0 16  0  0 15  0  0
```

```
get.dual.solution(lpmodel)
```

```
##  [1]   1 -10 -15 -10   0  25  30  30  25   0   0   5   0  15   0   5   0   0  10
## [20]   0   5   5   0   5  20
```

Production increase decision

1. Shadow prices are -10 (Alpha), -15 (Beta), -10 (Gamma), 0 (Delta)
2. Increasing production at Beta will reduce cost by 15 for the first additional unit

b.

A pizza chain in Athens has three locations, and in the last five minutes has received orders for pizzas to be delivered to five addresses in Athens. Assuming ordering is centrally managed, which store should assigned the making of the five orders to minimize delivery distance? What is the shortest route in miles?

```
library(TSP)
library(ggplot2)
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library(mapproj)
```

```
## Loading required package: maps
```

```
library(measurements)
library(readr)
library(googleway)
k <- read_csv("~/Dropbox/R/API keys/GoogleAPIkey.txt")
```

```
## Parsed with column specification:
## cols(
##   key = col_character()
## )
```

Use a loop to find the delivery distance from each store and select the minimum as the pizza making store.

```
maker <-  c('700 E. Broad, Athens, GA', '1591 S. Lumpkin, Athens, GA','120 Alps, Athens, GA')
mindist <-  9999999
num_makers = length(maker)
for(s in 1:num_makers) {
  locations <-  c(maker[s], "170 River Rd, Athens, GA", "785 S. Milledge Ave., Athens, GA", "558 West B
# Set a pickup store as the starting point for the tour
start_location <- 1
  l <-  length(locations) # number of locations is length of vector
  # set up distance as symmetric matrix
  distMat <-  matrix(0,l,l)
  # compute distances and enter into cells
  for(i in 1:(l-1)){
    for(j in (i+1):l){
     t <-  google_distance(locations[i],locations[j],mode=c('driving'),simplify = TRUE, key = k$key)
     tx <- t$rows$elements[[1]]$distance$value*conv_unit(1,'m','mi') # convert meters to miles
     distMat[i,j] <-  tx
      distMat[j,i] <-  tx
    }
  }
  tsp <-  TSP(distMat,locations)
  # Set store as the starting point
  solve_TSP(tsp,start = start_location)
  tour <- solve_TSP(tsp)
  labels(tour)
  as.integer(tour)
  tour_length(tour)
  print(maker[s])
  print(tour_length(tour))
  if(tour_length(tour) < mindist) {
    mindist <-  tour_length(tour)
    mindeliver <-  s
    mintour <-  tour
  }
}
```

```
## [1] "700 E. Broad, Athens, GA"
## [1] 6.922696
## [1] "1591 S. Lumpkin, Athens, GA"
## [1] 7.10103
## [1] "120 Alps, Athens, GA"
## [1] 8.413366
```

The best store for delivery

```
print("Minimum distance solution")
```

```
## [1] "Minimum distance solution"
```

```r
print(maker[mindeliver])
```

```
## [1] "700 E. Broad, Athens, GA"
```

```r
print(round(mindist,2))
```

```
## [1] 6.92
```