MSDS_650_Linear_Programming_Transportation.R

SeanOMalley1

Sat Dec 10 12:37:11 2016

```
# MSDS 650
# Sean O'Malley
# Linear Programming using lpSolveAPI

# A trading company is looking for a way to maximize profit per transportation of their goods. The company ha
s a train available with 3 wagons. When stocking the wagons they can choose between 4 types of cargo, each wi
th its own specifications. How much of each cargo type should be loaded on which wagon in order to maximize p
rofit?

#### Step 1: Install lpSolveAPI and Load Packaages ####

require(lpSolveAPI)
```

```
## Loading required package: lpSolveAPI
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(reshape)
```

```
## Loading required package: reshape
```

```
require(gridExtra)
```

```
## Loading required package: gridExtra
```

```
#### Step 2: Set Parameters Specific to the Problem ####

# define datasets
# 3 train wagons and each wagon's space and weight capacity
train <- data.frame(wagon = c('w1','w2','w3'),
                    weightcapacity = c(10, 8, 12),
                    spacecapacity = c(5000, 4000, 8000))

# 4 types of cargo and the number available of each, volume, and profit of each cargo type
cargo <- data.frame(type = c('c1', 'c2','c3', 'c4'),
                   available = c(18,10,5,20),
                   volume = c(400, 300, 200, 500),
                   profit = c(2000, 2500, 5000, 3500))

#### Step 3: Create Linear Programming Model ####

# model with 10 constraints and 12 decision variables
lpmodel <- make.lp( 2*NROW(train) + NROW(cargo), 12)

column <- 0
row <- 0

# Build function that takes the model column per column using for loops to work through each wagon and cargo
type.
# Second function, takes arguments 'column','values' and indicies'
for (wg in train$wagon) {
  row <- row + 1
  for (type in seq(1, NROW(cargo$type))){
    column <- column + 1
    set.column(lpmodel, column, c(1, cargo[type, 'volume'], 1),
               indices = c(row,NROW(train) + row, NROW(train)*2 + type))
  }
}

#### Step 4: Set Constraints ####

# rhs weight constraints
set.constr.value(lpmodel, rhs=train$weightcapacity, constraints=seq(1,NROW(train)))

# rhs volume constraints
set.constr.value(lpmodel, rhs=train$spacecapacity, constraints=seq(NROW(train)+1,NROW(train)*2))

#set rhs volume constraints
set.constr.value(lpmodel, rhs=cargo$available, constraints=seq(NROW(train)*2+1,NROW(train)*2+NROW(cargo)))

#### Step 5: Set Objective Metrics ####

# set objective coefficients
set.objfn(lpmodel, rep(cargo$profit,NROW(train)))

# set objective direction
lp.control(lpmodel,sense='max')
```

```
## $anti.degen
## [1] "fixedvars" "stalling"
##
## $basis.crash
## [1] "none"
```

```
##
## $bb.depthlimit
## [1] -50
##
## $bb.floorfirst
## [1] "automatic"
##
## $bb.rule
## [1] "pseudononint" "greedy"       "dynamic"      "rcostfixing"
##
## $break.at.first
## [1] FALSE
##
## $break.at.value
## [1] 1e+30
##
## $epsilon
##       epsb       epsd      epsel     epsint epsperturb   epspivot
##      1e-10      1e-09      1e-12      1e-07      1e-05      2e-07
##
## $improve
## [1] "dualfeas" "thetagap"
##
## $infinite
## [1] 1e+30
##
## $maxpivot
## [1] 250
##
## $mip.gap
## absolute relative
##    1e-11    1e-11
##
## $negrange
## [1] -1e+06
##
## $obj.in.basis
## [1] TRUE
##
## $pivoting
## [1] "devex"    "adaptive"
##
## $presolve
## [1] "none"
##
## $scalelimit
## [1] 5
##
## $scaling
## [1] "geometric"   "equilibrate" "integers"
##
## $sense
## [1] "maximize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
## [1] 0
```

```
##
## $verbose
## [1] "neutral"
```

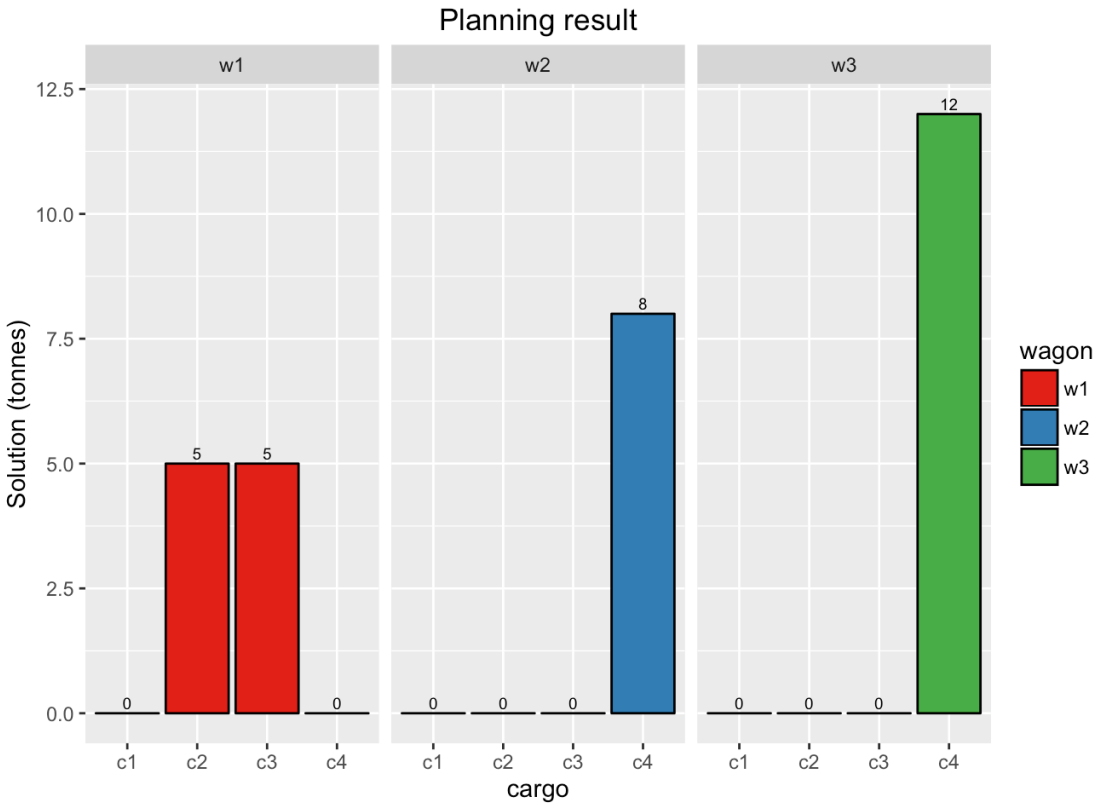
```
# examine lpmodel
write.lp(lpmodel,'/Users/SeanOMalley1/Desktop/MSDS_650/model.lp',type='lp')

#### Step 6: Solve Model ####

# Returns a 0, this implies that an optimal solution was found.
solve(lpmodel)
```

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

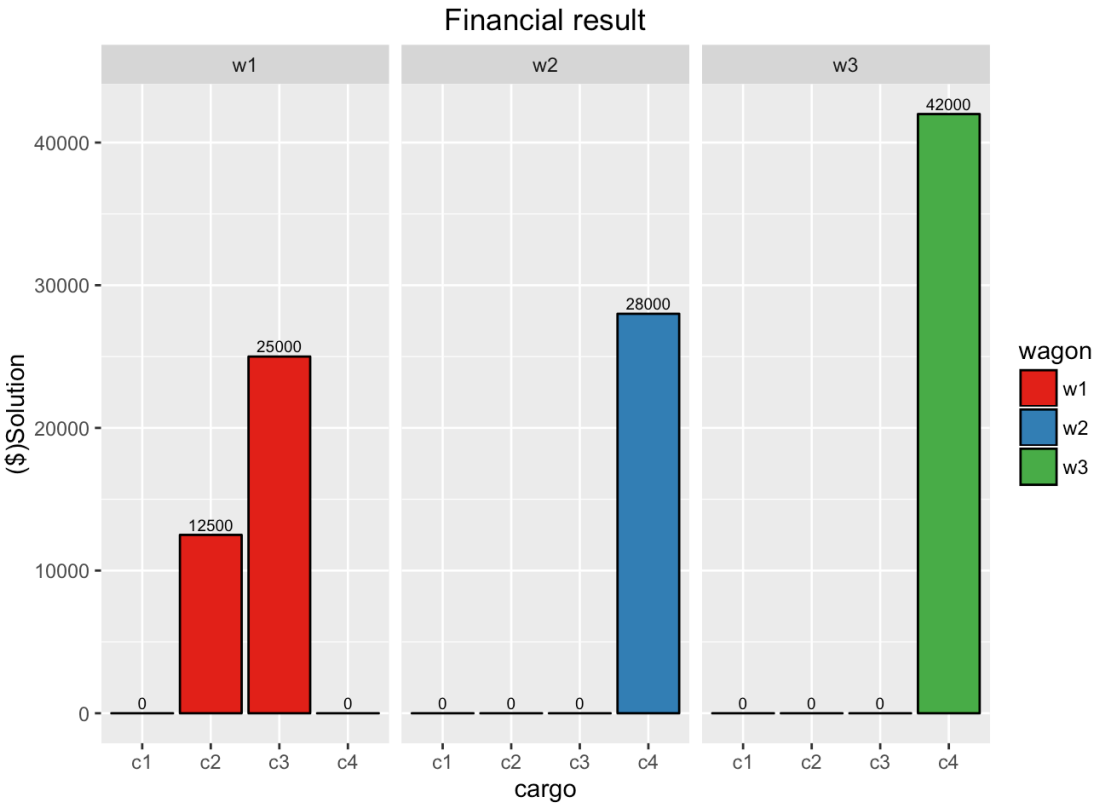
```
# The value of the solution is returned, showing maximum total profit of $107,500
get.objective(lpmodel)
```

```
## [1] 107500
```

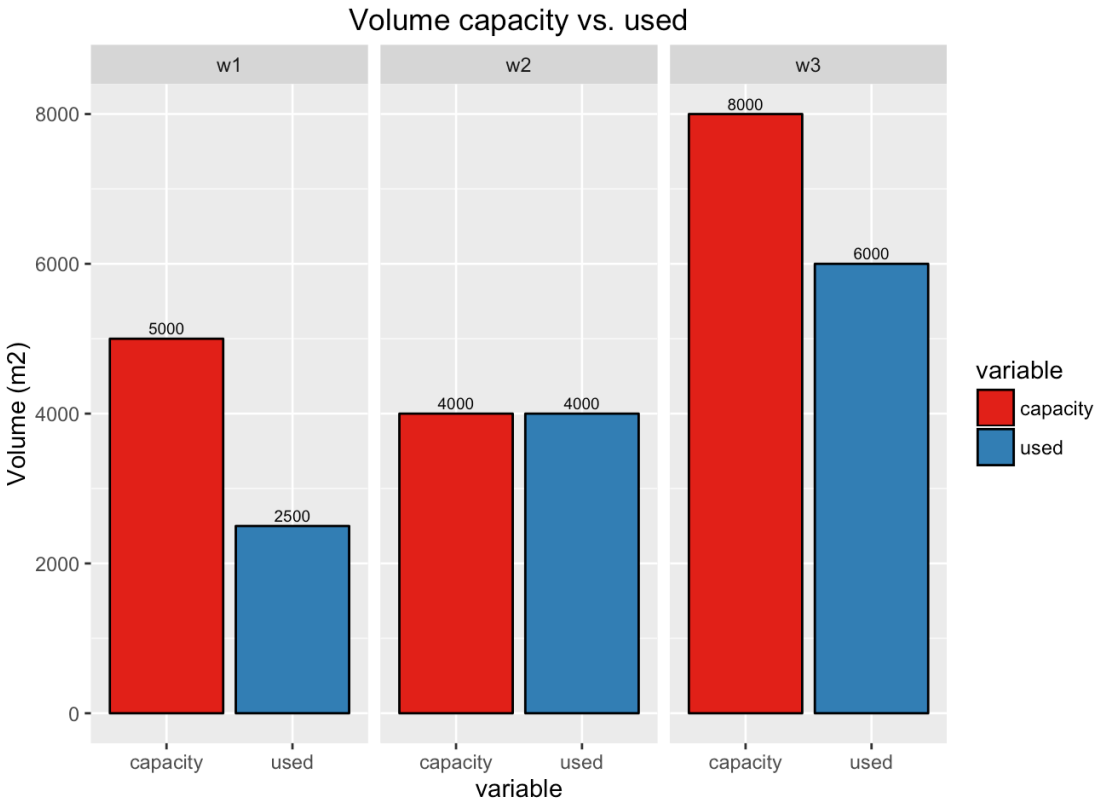
```
#### Step 7: Visualize Solution ####

# Map results to data frame
results <- data.frame(cargo = rep(cargo$type, 3),
                      wagon = as.vector(sapply(train$wagon, FUN = function(x) rep(x, NROW(cargo)))), solution
= get.variables(lpmodel))

# Create plot of results
ggplot(results, aes(x=cargo, y=solution, fill=wagon)) +
             geom_bar(color='black', position='dodge', stat = 'identity') +
             geom_text(aes(label=solution), size=2.5, position=position_dodge(width=1), vjust=-.4) +
             scale_fill_brewer(palette='Set1') +
             facet_grid(.~wagon) +
             ggtitle('Planning result') +
             ylab('Solution (tonnes)')
```



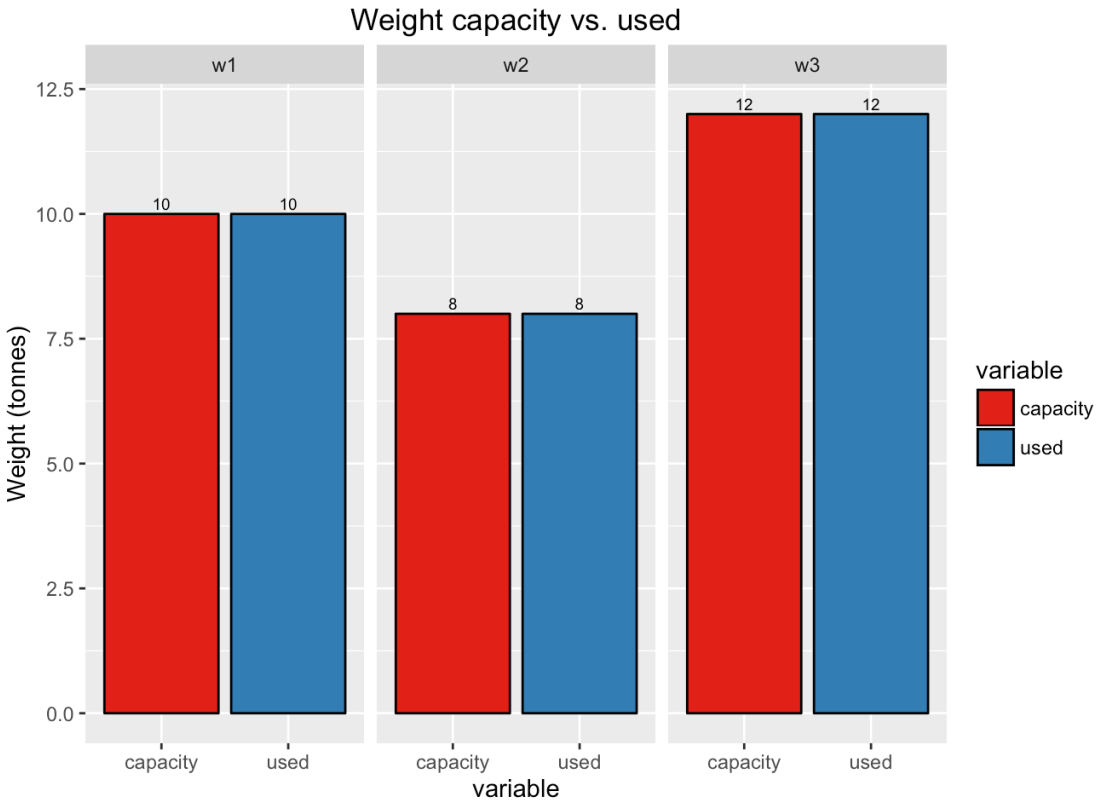

```
# Map financial result to data frame
financialresult <- data.frame(cargo = rep(cargo$type, 3),
                              wagon = as.vector(sapply(train$wagon, FUN=function(x) rep(x, NROW(cargo)))),
                              solution = get.variables(lpmodel),
                              profit_unit=rep(cargo$profit, 3))

# multiply profit unit column by solution column to get new column called profit
financialresult$profit <- financialresult$profit_unit*financialresult$solution

# Create plot of financial results
ggplot(financialresult, aes(x=cargo, y=profit, fill=wagon)) +
            geom_bar(color='black', position='dodge', stat = 'identity') +
            geom_text(aes(label=profit), size=2.5, position=position_dodge(width=1), vjust=-.4) +
            scale_fill_brewer(palette='Set1') +
            facet_grid(.~wagon) +
            ggtitle('Financial result') +
            ylab('($)Solution')
```




```
# Map space and weight capacity constraints to data frame
spacecapacity  <- data.frame(wagon=train$wagon, capacity=train$spacecapacity, used=get.constraints(lpmodel)[4
:6])
weightcapacity <- data.frame(wagon=train$wagon, capacity=train$weightcapacity, used=get.constraints(lpmodel)[
1:3])

# Create plot of each capacity constraints
ggplot(melt(spacecapacity, id='wagon'), aes(x=variable, y=value, fill=variable)) +
            geom_bar(color='black', stat = 'identity') +
            facet_grid(.~wagon) +
            ggtitle('Volume capacity vs. used') +
            scale_fill_brewer(palette='Set1') +
            geom_text(aes(label=value), size=2.5, vjust=-.4) +
            ylab('Volume (m2)')
```




```
ggplot(melt(weightcapacity, id='wagon'), aes(x=variable, y=value, fill=variable)) +
            geom_bar(color='black', stat = 'identity') +
            facet_grid(.~wagon) +
            ggtitle('Weight capacity vs. used') +
            scale_fill_brewer(palette='Set1') +
            geom_text(aes(label=value), size=2.5, vjust=-.4) +
            ylab('Weight (tonnes)')
```


Weight capacity vs. used
w1
w2
w3
12.5
10.0
7.5
5.0
2.5
0.0
Weight (tonnes)
10
10
8
8
12
12
capacity
used
capacity
used
capacity
used
variable
variable
capacity
used