dualassign package

Stefan Rampertshammer

July 6, 2018

1 Summary

This package uses a fast and approximately optimal method to assign resources to tasks.

1.1 Included functions

- dual_assignment
- iterate_dual_process

1.2 Package requirements

- dplyr
- lpSolve

2 Use and examples

2.1 dual_assignment

Input and output is an object (ss) containing:

- *ss$tasks* - vector of length K showing the list of work to be done
- *ss$assigns* - vector of length J showing the list of potential assignments for each resource

- $ss\$users$ - vector of length I showing the resources to be assigned
- $ss\$perform$ - $I \times K$ matrix showing the performance by each resource on each task
- $ss\$costs$ - $I \times J$ matrix showing the cost of setting each resource i to assignment j
- $ss\$P$ - $J \times K$ matrix showing the proportion of assignment j spent on task k
- $ss\$demand$ - vector of length K showing the required work to be done in each task
- $ss\$otcost$ - scalar showing the relative cost of overtime(backlog) cost to normal-time
- $ss\$otperform$ - vector of length K showing the rate at which overtime(backlog) gets done for each task
- $ss\$solution$ - vector of length I showing which assignment each user is given (Initially NULL)
- $ss\$backlog$ - vector of length K showing the amount of extra resources needed to complete each task (Initially NULL)
- $ss\$istar$ - together with $ss\$jstar$ shows the assignment order (Initially NULL). $(ss\$istar[n], ss\$jstar[n])$ shows that on the n^{th} iteration of the algorithm, $ss\$istar[n]$ is assigned to $ss\$jstar[n]$. Backlog resources are denoted by $ss\$istar = 0$
- $ss\$jstar$ - Initially NULL

2.2 iterate_dual_process

Input and output is an object (lpo) containing:

- $lpo\$n$ - vector showing I, J, K from dual_assignment function
- $lpo\$obj$ - objective function for dual linear program
- $lpo\$rhs$ - rhs of inequality constraint for dual linear program

- $lpo\$mat$ - matrix for inequality constraint in dual linear program
- $lpo\$dir$ - direction of inequality constraint
- $lpo\$ass$ - current assignments
- $lpo\$bl$ - curerent backlogs/overtime
- $lpo\$istar$ - most recent assignment
- $lpo\$jstar$ - most recent assignment
- $lpo\$cost$ - scalar showing total running cost of assignment, initialized to 0
- $lpo\$exit$ - scalar showing exit conditions
- $lpo\$count$ - scalar showing number of assignments made so far, intialized to 0

2.3 Examples

2.4 Example 1

In this example we take a simple situation in which five identical resources must be assigned to three tasks. Each assignment is perfectly matched with the task, reflected by $ss\$P$ being the identity matrix. The ability to perform each task is given by $c(3, 3, 1)$ while backlogged work costs 150% of the regular work and is done at a lower rate $c(2, 2, 0.4)$. The amount of work required to be completed is $c(10, 15, 3)$.

```
library(dualassign)
ss=list(
  tasks=c(1,2,3),
  assigns=c(1,2,3),
  users=c(1,2,3,4,5),
  perform=matrix(c(rep(c(3,3,1),each=5)),nrow=5),
  P=diag(c(1,1,1)),
  demand=c(10,15,3),
  otcost=1.5,
  otperform=c(2,2,0.4),
```

```
  solution=NULL,
  backlog=NULL,
  istar=NULL,
  jstar=NULL,
  status=c("incomplete")
)
ss2=dual_assignment(ss)

ss2$solution
[1] 3 1 2 1 3
ss2$backlog
[1] 2 6 3
ss2$istar
 [1] 1 2 3 4 5 0 0 0 0 0 0 0 0 0 0 0
ss2$jstar
 [1] 3 1 2 1 3 2 1 1 2 3 3 3 2 2 2 2
```

The solution vector shows how the resources are assigned: the first resource is assigned to task 3, the second to task 1 and so on.

The backlog vector shows how many additional backlog resources are required for each task. Here we see that a total of 11 extra resources are required, 6 of which are for task 2.

$ss\$istar$ and $ss\$jstar$ together show the order in which resources are assigned, with resource $ss\$istar$ assigned to task $ss\$jstar$. First, resource 1 is assigned to task 3, then resource 2 is assigned to task 1. Once all the resources have been assigned, $ss\$istar$ takes on the value of 0 showing that a backlog resource is assigned here.

3 Theory

The problem statement:

- $i \in I$ collection of resources to distribute to assignments $j \in J$
- $k \in K$ collection of tasks done by each assignment j in proportion p_{jk}
- i performs k at rate r_{ik} and can only be assigned to a single j at cost w_{ij}

- D_k is the required amount of task k to be completed.

- Any tasks not completed by I can be sent to a backlog who completes tasks at rate R_k but increased cost W_k. The backlog can take any amount of work.

If we set x_{ij} to be the assignment decision variable from resource i to task j, relax the integral constraint on x_{ij} from $x_{ij} \in \{0,1\} \to x_{ij} \in [0,1]$ then we can minimize the amount of resources used to meet demand by considering the linear program:

$$
\begin{aligned}
\min_x \sum_{i\in I}\sum_{j\in J} w_{ij}x_{ij} + \sum_{k\in K} W_k y_k & & \text{s.t.} \\
x_{ij} &\geq 0 & (i,j) \in I \times J \\
y_k &\geq 0 & k \in K \\
-\sum_{j\in J} x_{ij} &\geq -1 & i \in I \\
\sum_{i\in I}\sum_{j\in J} p_{jk} r_{ik} x_{ij} + R_k y_k &\geq D_k & k \in K
\end{aligned}
$$

which has dual

$$
\begin{aligned}
\max_{\theta,\eta} \sum_{k\in K} D_k \eta_k - \sum_{i\in I} \theta_i & & \text{s.t.} \\
\theta_i &\geq 0 & i \in I \\
\eta_k &\geq 0 & k \in K \\
\sum_{k\in K} p_{jk} r_{ik} \eta_k - \theta_i &\leq w_{ij} & (i,j) \in I \times J \\
R_k \eta_k &\leq W_k & k \in K
\end{aligned}
$$

The dual variables θ_i and η_k have the meaningful interpretations of θ_i being the value of duplicating resource i and η_k is proportional to the additional resources that an additional task k would cost.

The selection algorithm is as follows: Loop until all remaining demand is non-positive

1. run the dual LP with updated $\{D_k\}$, determine dual variables θ_i, η_k
2. find the maximal element of $\left\{\sum_{k \in K} p_{jk} r_{ik} \eta_k / w_{ij}, R_k \eta_k / W_k\right\}$,
    - if it is some $\sum_{k \in K} p_{jk} r_{ik} \eta_k / w_{ij}$ then assign i to j
    - if it is some $R_k \eta_k / W_k$ then assign a backlog resource to k
3. in the case a resource from I is assigned, remove that resource from the available pool.
4. calculate the work done by the assigned resource and deduct it from $(D_1, ..., D_K)$
5. update demand and list of remaining resources and go to 1.