Agent-based model simulator

Necantis

2018-08-11

## Theoretical model, version 01

In this initial model, we have 3 phases  
# Phase 01: Idea generation We assign a random number to each agent

target <- 1e9 # we set the target at 1'000'000'000  
totAgents <- 1e3  
clusters <- 10  
  
set.seed(1) #Setting a seed to allow comparable results  
ideas <- runif(totAgents, min=0, max=target) # Generating random numbers in a uniform distribution

We cluster the results

ideaClusters <- 1+as.integer(ideas/(target/clusters)) #Extracting the cluster by dividing each number to the number of clusters  
agentsId <- 1:totAgents # Adding a UID  
agentsReward\_01 <- rep(0,totAgents) # A variable associated to phase 01  
agentsReward\_02 <- rep(0,totAgents) # A variable associated to phase 02  
agents <- data.frame(agentsId, ideas, ideaClusters,agentsReward\_01,agentsReward\_02) #Creating a dataframe with ID, value and cluster  
  
# hist(agents$ideas)  
# hist(agents$ideaClusters)

We assign prizes

idClusters <- seq(1:clusters)  
idWinners <- rep(NA,clusters)  
valueWinners <- rep(NA,clusters)  
listWinners <- data.frame(idClusters,idWinners, valueWinners) #Creating a dataframe with Cluster ID, Winner ID and value  
  
for(i in 1:totAgents){  
 j <- agents[i,3] # Check the cluster of the agent  
 if(is.na(listWinners[j,2])){ # If the cluster of the agent does not have a winner (it's value is NA) ...  
 listWinners[j,2] <-agents[i,1] # ... use the agent's ID  
 listWinners[j,3] <-agents[i,2] # ... use the agent's value   
 agents[i,4] <- 1 # rewarding the selected agent  
 }  
}  
  
listWinners

## idClusters idWinners valueWinners  
## 1 1 10 61786270  
## 2 2 12 176556753  
## 3 3 1 265508663  
## 4 4 2 372123900  
## 5 5 16 497699242  
## 6 6 3 572853363  
## 7 7 8 660797792  
## 8 8 15 769841420  
## 9 9 6 898389685  
## 10 10 4 908207790

# Phase 02: Idea pooling

We use [lpSOlve](https://www.kdnuggets.com/2018/05/optimization-using-r.html) to define the objective function

library(lpSolve)  
objective.in <- listWinners[,3] # Pooling the retained ideas by looking for the best combination

We define the constraints of the function

mat <- matrix(listWinners[,3], nrow=1, byrow=TRUE) # The sum of the pooled ideas ...  
dir <- "<=" # ... should be below ...  
rhs <- target # ... the target

We solve the Linear programming function and we reward the owners of the pooled ideas

optimum <-lp(direction="max", objective.in, mat, dir, rhs, all.bin = TRUE) # Which is the best combination of pooled ideas?  
optimum$solution # The selected ideas

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

for(i in 1:clusters){  
 if(optimum$solution[i] >0){ # if an idea is pooled ...  
 j <- listWinners[i,2] # ... select the owner of the pooled idea  
 agents[j,5] <- 1 # ... and reward the agent another time  
 }  
}

# Phase 03: Analyze results

[Filter](http://r4ds.had.co.nz/transform.html#filter-rows-with-filter) the winners

totWinners <- filter(agents, agentsReward\_01 >0) # This could reduce computational effort

[Gather](http://r4ds.had.co.nz/tidy-data.html#spreading-and-gathering) the two last columns of agents into one

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.4.4

## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.0.0 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.6  
## v tidyr 0.8.1 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.4

## Warning: package 'readr' was built under R version 3.4.4

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'stringr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.4

## -- Conflicts ------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

agentsRewards <- agents %>%   
 gather('agentsReward\_01', 'agentsReward\_02', key = "Phase", value = "Rewards")

[Summarize](http://r4ds.had.co.nz/transform.html#grouped-summaries-with-summarise) by agent

summarise(group\_by(agentsRewards, agentsId), Rewards = sum(Rewards, na.rm = TRUE))

## # A tibble: 1,000 x 2  
## agentsId Rewards  
## <int> <dbl>  
## 1 1 2  
## 2 2 1  
## 3 3 1  
## 4 4 1  
## 5 5 0  
## 6 6 1  
## 7 7 0  
## 8 8 2  
## 9 9 0  
## 10 10 2  
## # ... with 990 more rows

Check the reminder for another idea challenge

remainder <- target - sum(agents$ideas\*agents$agentsReward\_02) #Comparing the initial target with the sum of the pooled ideas

## Theoretical model, version 02

Setting up a function to generate ideas

phase00 <- function(target, totAgents,totIdeas,clusters){  
   
 set.seed(1) #Setting a seed to a fix number to allow comparable results  
 agentsId <- as.integer(rnorm(totIdeas, totAgents/2, totAgents/10)) # Assigning an owner to each idea (this allows multiple ideas to one owner)  
   
 set.seed(1) #Setting a seed again  
 idea <- as.integer(runif(totIdeas, min = 0, max = target)) # Generating random numbers in a uniform distribution  
   
 ideaClusters <- 1+as.integer(idea/(target/clusters)) #Extracting the cluster by dividing each number to the number of clusters  
 agentsReward\_01 <- rep(0,totIdeas) # A variable associated to phase 01  
 agentsReward\_02 <- rep(0,totIdeas) # A variable associated to phase 02  
 ideas <- data.frame(agentsId, idea, ideaClusters,agentsReward\_01,agentsReward\_02) #Creating a dataframe with ID, value and cluster  
  
 return(ideas)  
}

Setting up a function to select ideas

phase01 <- function(clusters, ideas){  
  
 #Creating a dataframe with Cluster ID, Winner ID and value  
 idClusters <- seq(1:clusters)  
 idWinners <- rep(NA, clusters)  
 valueWinners <- rep(NA, clusters)  
 listWinners <- data.frame(idClusters, idWinners, valueWinners)   
   
 #Reward the owners of the selected ideas  
 lenIdeas <- dim(ideas)[1]  
 for (i in 1:lenIdeas) {  
 j <- ideas[i, 3] # Check the cluster of the idea  
 if (is.na(listWinners[j, 2])) {  
 # If the cluster of the agent does not have a winner (it's value is NA) ...  
 listWinners[j, 2] <- ideas[i, 1] # ... use the agent's ID  
 listWinners[j, 3] <- ideas[i, 2] # ... use the idea's value  
 ideas[i, 4] <- 1 +ideas[i, 4] # rewarding the selected agent by increasing the reward to 1 (taking into account multiple challenges)  
 }  
 }  
 return(ideas)  
}

Setting up a function to pool ideas

phase02 <- function (selectedIdeas, target) {  
  
 library(lpSolve)  
 ideas <- selectedIdeas  
 listWinners <- filter(ideas, agentsReward\_01 >0) #Filtering the list of winners before performing a left join with the pooled ideas  
  
 #Linear programming  
 objective.in <- listWinners[, 2] # Trying to pool the retained ideas by looking for the best combination of coefficients  
 mat <- matrix(listWinners[, 2], nrow = 1, byrow = TRUE) # The sum of the pooled ideas ...  
 dir <- "<=" # ... should be below ...  
 rhs <- target # ... the target  
 optimum <- lp(direction = "max", objective.in, mat, dir, rhs, all.bin = TRUE) # Which is the best combination of pooled ideas?  
 optimum$solution # The selected ideas  
   
 listWinners[,5] <- optimum$solution #Assign prizes  
 polledIdeas <- ideas %>%  
 left\_join(listWinners, by = c("agentsId","idea","ideaClusters","agentsReward\_01")) #Left join  
 agentsReward\_02 <- polledIdeas$agentsReward\_02.x+ polledIdeas$agentsReward\_02.y # Taking into account previous idea challenges  
 polledIdeas <- data.frame(polledIdeas,agentsReward\_02) #add the new column  
  
 polledIdeas[5] <- NULL # Removing the redundant columns  
 polledIdeas[5] <- NULL # Removing the redundant columns  
  
 return(polledIdeas)  
}

Setting up a function to summarize the rewards

phase03 <- function(ideas, target){  
 library(tidyverse)  
 agentsRewards <- ideas %>%   
 gather('agentsReward\_01', 'agentsReward\_02', key = "Phase", value = "Rewards")  
  
 winners <- summarise(group\_by(agentsRewards, agentsId, idea), Rewards = sum(Rewards, na.rm = TRUE))  
 winners <- arrange(winners, idea) #Sorting ideas  
 problemSolved <- sum(filter(winners, Rewards > 1)[, 2])  
   
 #Print results for comparison  
 print("filter(winners, Rewards >0)")  
 print(filter(winners, Rewards > 0)) # Test: selected ideas  
 print("Mean(winners$Rewards)")  
 print(mean(winners$Rewards))  
 print("Problem Solved")  
 print(problemSolved)  
 print("Remainder")  
 print(target-problemSolved)  
   
 return(agentsRewards)  
}

Setting up a function to run an idea Challenge

ideaChallenge <- function(target, totAgents, totIdeas, clusters, sortedIdea) {  
   
 generatedIdeas <- as.data.frame(seq(1:totIdeas))  
 generatedIdeas <- phase00(target, totAgents, totIdeas, clusters) # Generating ideas  
  
 if(sortedIdea) generatedIdeas <-arrange(generatedIdeas,desc(idea)) # Sorting ideas by experts  
  
 selectedIdeas <- phase01(clusters, generatedIdeas) # Selecting ideas  
 polledIdeas <- phase02(selectedIdeas, target) # Polling ideas   
 agentsRewards <- phase03(polledIdeas, target)  
 return(agentsRewards)  
}

# Run the simulation

## Standard idea challenge Benchmark

target <- 1e9 # we set the target at 1'000'000'000  
totAgents <- 1e3  
totIdeas <- 1e4  
clusters <- 1  
  
generatedIdeas0 <- as.data.frame(seq(1:totIdeas))  
generatedIdeas0 <- phase00(target, totAgents, totIdeas, clusters) # Generating ideas  
  
generatedIdeas0 <-arrange(generatedIdeas0,desc(idea)) #Sorting ideas by experts

## Warning: package 'bindrcpp' was built under R version 3.4.4

selectedIdeas0 <- phase01(clusters, generatedIdeas0) # Selecting ideas  
  
agentsRewards0 <- selectedIdeas0 %>%   
 gather('agentsReward\_01', 'agentsReward\_02', key = "Phase", value = "Rewards")  
  
winners <- dim(filter(agentsRewards0, Rewards>0))[1]  
quality <-filter(agentsRewards0, Rewards==1)[,2]  
prize <- mean(agentsRewards0$Rewards) \* totAgents  
challengeID <- "Single Winner"  
  
performance <- data.frame(challengeID, winners, quality, prize)  
  
performance$winners # Winners

## [1] 1

performance$quality # Sum of the pooled ideas

## [1] 999930593

performance$prize # Cost of prizes

## [1] 0.05

## First idea challenge

MODEL A: Selecting the first idea for each cluster

target <- 1e9 # we set the target at 1'000'000'000  
totAgents <- 1e3  
totIdeas <- 1e4  
clusters <- 10  
  
agentsRewards1\_1 <- ideaChallenge(target, totAgents, totIdeas, clusters, FALSE)

## [1] "filter(winners, Rewards >0)"  
## # A tibble: 10 x 3  
## # Groups: agentsId [10]  
## agentsId idea Rewards  
## <int> <int> <dbl>  
## 1 469 61786270 1  
## 2 538 176556752 2  
## 3 437 265508663 1  
## 4 518 372123899 1  
## 5 495 497699242 1  
## 6 416 572853363 1  
## 7 573 660797792 1  
## 8 612 769841419 2  
## 9 417 898389684 1  
## 10 659 908207789 1  
## [1] "Mean(winners$Rewards)"  
## [1] 0.0012  
## [1] "Problem Solved"  
## [1] 946398171  
## [1] "Remainder"  
## [1] 53601829

Model B: Selecting the idea for each cluster with the highest score

agentsRewards2 <- ideaChallenge(target, totAgents, totIdeas, clusters, TRUE) #using sorted ideas

## [1] "filter(winners, Rewards >0)"  
## # A tibble: 10 x 3  
## # Groups: agentsId [10]  
## agentsId idea Rewards  
## <int> <int> <dbl>  
## 1 359 99989993 2  
## 2 508 199977182 1  
## 3 486 299951496 1  
## 4 617 399994368 2  
## 5 474 499982866 2  
## 6 552 599919136 1  
## 7 469 699974252 1  
## 8 468 799901459 1  
## 9 544 899746668 1  
## 10 578 999930593 1  
## [1] "Mean(winners$Rewards)"  
## [1] 0.0013  
## [1] "Problem Solved"  
## [1] 999967227  
## [1] "Remainder"  
## [1] 32773

## New idea challenge starting from the remainder of the previous idea challenge

# Setting up the remainder as new target  
 winners1\_1 <- summarise(group\_by(agentsRewards1\_1, agentsId, idea), Rewards = sum(Rewards, na.rm = TRUE))  
 problemSolved1\_1 <- sum(filter(winners1\_1, Rewards > 1)[, 2])   
 remainder1\_1 <- target - problemSolved1\_1  
  
agentsRewards1\_2 <- ideaChallenge(remainder1\_1, totAgents, totIdeas, clusters, FALSE)

## [1] "filter(winners, Rewards >0)"  
## # A tibble: 10 x 3  
## # Groups: agentsId [10]  
## agentsId idea Rewards  
## <int> <int> <dbl>  
## 1 469 3311857 1  
## 2 538 9463764 2  
## 3 437 14231749 1  
## 4 518 19946521 1  
## 5 495 26677589 1  
## 6 416 30705988 1  
## 7 573 35419970 1  
## 8 612 41264908 2  
## 9 417 48155330 1  
## 10 659 48681598 1  
## [1] "Mean(winners$Rewards)"  
## [1] 0.0012  
## [1] "Problem Solved"  
## [1] 50728672  
## [1] "Remainder"  
## [1] 2873157

## New idea challenge: Cheatstorming with the remainder

# agentsRewards1\_3 <- ideaChallenge\_Cheatstorming(remainder1\_1, totAgents, totIdeas, clusters, agentsRewards1\_1) # Using old ideas  
  
 target <- remainder1\_1 # New target  
  
 generatedIdeas<- agentsRewards1\_1 %>%  
 spread(key = Phase, value = Rewards) # Putting the table back in shape  
   
 generatedIdeas<-filter(generatedIdeas,idea<target) #Removing the ideas that are above the target   
 generatedIdeas$ideaClusters <- 1+as.integer(generatedIdeas$idea/(target/clusters))# New clusters  
  
 generatedIdeas$agentsReward\_01 <- 1/dim(generatedIdeas)[1] +generatedIdeas$agentsReward\_01 # Reward all contributions in phase 01  
 if(is.na(generatedIdeas$agentsReward\_02)) {  
 generatedIdeas$agentsReward\_02 <- 0 # Remove NA from column 2  
 }

## Warning in if (is.na(generatedIdeas$agentsReward\_02)) {: la condition a une  
## longueur > 1 et seul le premier élément est utilisé

selectedIdeas <- generatedIdeas  
 pooledIdeas <- phase02(selectedIdeas, target) # Pooling ideas  
 agentsRewards <- phase03(pooledIdeas, target)

## [1] "filter(winners, Rewards >0)"  
## # A tibble: 564 x 3  
## # Groups: agentsId [301]  
## agentsId idea Rewards  
## <int> <int> <dbl>  
## 1 465 106433 0.00177  
## 2 546 120655 0.00177  
## 3 342 200380 1.00   
## 4 569 570522 0.00177  
## 5 405 605266 1.00   
## 6 573 656347 0.00177  
## 7 510 668174 0.00177  
## 8 585 682127 0.00177  
## 9 320 918506 1.00   
## 10 309 1026818 0.00177  
## # ... with 554 more rows  
## [1] "Mean(winners$Rewards)"  
## [1] 0.0141844  
## [1] "Problem Solved"  
## [1] 53601829  
## [1] "Remainder"  
## [1] 0

filter(agentsRewards, Rewards>=1)

## agentsId idea ideaClusters Phase Rewards  
## 1 260 6082597 2 agentsReward\_02 1  
## 2 320 918506 1 agentsReward\_02 1  
## 3 342 200380 1 agentsReward\_02 1  
## 4 395 1836858 1 agentsReward\_02 1  
## 5 405 605266 1 agentsReward\_02 1  
## 6 545 1363479 1 agentsReward\_02 1  
## 7 776 42594743 8 agentsReward\_02 1

# Third idea challenge: more clusters

clusters <- 50 # Increased the number of clusters  
  
agentsRewards3 <- ideaChallenge(target, totAgents, totIdeas, clusters, FALSE)

## [1] "filter(winners, Rewards >0)"  
## # A tibble: 50 x 3  
## # Groups: agentsId [46]  
## agentsId idea Rewards  
## <int> <int> <dbl>  
## 1 484 717746 1  
## 2 536 1250595 1  
## 3 620 3158990 1  
## 4 469 3311857 1  
## 5 698 5331568 1  
## 6 494 5785975 1  
## 7 301 6729982 2  
## 8 526 7681376 1  
## 9 538 9463764 1  
## 10 494 9981604 1  
## # ... with 40 more rows  
## [1] "Mean(winners$Rewards)"  
## [1] 0.0054  
## [1] "Problem Solved"  
## [1] 53600792  
## [1] "Remainder"  
## [1] 1037