Rendu Mega\_Sae Méthodes Numériques

Adrien - Antonin - Vladimir - Tristan - Titouan (groupe B2)

13/05/24

Utiliser des libraries R selon votre problèeme par exemple : ompr, boot, . . .

# Fonction linéaire sous contraintes linéaires

## Introduction

En rapport avec notre sujet de MEGA - SAE2.01234 : Conception et développement d’un outil d’aide et de pilotage à destination d’organisateurs de mariage (entreprise professionnel), nous allons chercher à optimiser le nombre de mariages de type A et B qu’un weeding planner peut organiser sur une année afin de maximiser le profit, tout en respectant certaines contraintes sur les ressources disponibles.

Nous allons ici, procéder à l’optimisation mathématique sous contraintes, mettant en oeuvre plusieurs fonctions linéaire. En effet, la fonction à maximiser (le profit total) ainsi que les contraintes définies (nombre de jours, nombre de logements, nombre d’intervenants) sont toutes des fonctions linéaires des variables de décision xA et xB (relatif aux mariages de type A et B).

Pour répondre le plus possible à la consigne, ce document est divisé en 3 partie : - Partie modélisation : Définition de la fonction à optimiser ainsi que les variables - Partie modélisation : Définition des contraintes sur les variables - Partie résolution : Résolution de notre problème (partie mathématiques et implémentation informatique)

Enfin, nous commenterons et interpréterons la solution obtenue afin de valider la pertinence de notre approche pour répondre à la problématique initiale.

## Modélisation - Définition de la fonction à optimiser

Imaginons que nous avons deux types de fêtes de mariage, A et B, avec différents coûts et exigences en ressources. Nous cherchons à maximiser le profit total en organisant un certain nombre de fêtes de type A et B. Nous définissons les variables suivantes : - xA : nombre de fêtes de type A à organiser - xB : nombre de fêtes de type B à organiser

Chaque mariage doit respecter les conditions suivantes :

Un mariage c’est environ 135€ par invité. Un mariage de type A coûterait 6750€, tandis qu’un mariage de type B coûterait environ 14000€. L’objectif est d’optimiser le profit.

La fonction à maximiser serait donc la suivante :

## Modélisation - Définition des contraintes

* On note Xa et Xb le nombre de mariages de type A et B à organiser respectivement.
* Notre objectif est de maximiser la fonction
* Les contraintes sont les suivantes :

La première contrainte est le nombre de jour travaillé par an pour l’utilisateur de notre outil. Il ne peut pas travailler plus de 290 jours par an.

La deuxième contrainte est le nombre de personnes à gérer par an. Il ne peut pas gérer plus de 1125 personnes par an.

La troisième contrainte est le nombre d’intervenants à gérer par an. Il ne peut pas gérer plus de 120 intervenants par an.

La quatrième et cinquième contrainte sont les contraintes de positivité des variables de décision.

Au niveau de la justification de nos contraintes :

### Nombre de jours travaillés par an :

Un salarié travaille environ 230 jours par an (on enlève les jours fériés, 25 jours de congés, 52 samedis et 52 dimanche.

Partons du principe que notre wedding planner est auto-entrepreneur et qu’il souhaite se lancer. Il veut également maximiser sa part de bénéfices et il mettra ses jours fériés à profit pour travailler. En effet, ses journées de travail ne sont pas très prenante puisqu’il utilise une application qui permet d’optimiser toutes ses tâches.

Notre wedding planner travaillera alors environ 290 jours par an.

### Temps de préparation d’un mariage :

* Un mariage de type A coûte moins cher puisqu’il y a moins de choses à préparer. Comme il y a moins préparatifs, le temps de préparation du mariage est 20 jours.
* Un mariage de type B coûte plus cher qu’un mariage de type A, puisqu’il y a plus de préparatifs, plus de personnes a accueillir… Ainsi, le temps de préparation d’un mariage de type B est 40 jours.

### Nombre de personnes à gérer par an :

Un mariage de type A accueille 50 personnes. Un mariage de type B accueille 120 personnes. Notre wedding planner ne peut pas gérer plus de 1125 personnes par an par soucis de stockage dans sa base de données.

### Nombre de logements à gérer par an :

Un mariage de type A nécessite 10 logements. Un mariage de type B nécessite 22 logements. Notre wedding planner ne peut pas gérer plus de 10 logements par an par soucis d’organisation.

# Fonction pour résoudre le modèle avec des coefficients légèrement perturbés  
solve\_with\_perturbation <- function(perturb\_factor) {  
 # Perturber les coefficients des contraintes  
 coef1 <- 10 + rnorm(1, 0, perturb\_factor)  
 coef2 <- 20 + rnorm(1, 0, perturb\_factor)  
 coef3 <- 20 + rnorm(1, 0, perturb\_factor)  
 coef4 <- 10 + rnorm(1, 0, perturb\_factor)  
 coef5 <- 10 + rnorm(1, 0, perturb\_factor)  
 coef6 <- 15 + rnorm(1, 0, perturb\_factor)  
  
 # Résoudre le modèle avec les coefficients perturbés  
 result <- MIPModel() |>  
 add\_variable(xA, type = "integer", lb = 0) |>  
 add\_variable(xB, type = "integer", lb = 0) |>  
 set\_objective(9500 \* xA + 14000 \* xB, "max") |>  
 add\_constraint(coef1 \* xA + coef2 \* xB <= 290) |>  
 add\_constraint(coef3 \* xA + coef4 \* xB <= 450) |>  
 add\_constraint(coef5 \* xA + coef6 \* xB <= 350) |>  
 solve\_model(with\_ROI(solver = "glpk"))  
  
 # Extraire les solutions  
 xA\_value <- ifelse(!is.na(get\_solution(result, xA)), get\_solution(result, xA), NA)  
 xB\_value <- ifelse(!is.na(get\_solution(result, xB)), get\_solution(result, xB), NA)  
   
 return(c(xA = xA\_value, xB = xB\_value))  
}  
  
# Nombre de simulations bootstrap  
B <- 1000  
# Facteur de perturbation (écart-type de la perturbation)  
perturb\_factor <- 1  
  
# Exécuter les simulations bootstrap  
bootstrap\_results <- replicate(B, solve\_with\_perturbation(perturb\_factor))  
  
# Convertir les résultats en data frame pour analyse  
bootstrap\_df <- as.data.frame(t(bootstrap\_results))  
colnames(bootstrap\_df) <- c("xA", "xB")  
  
# Afficher un résumé des solutions bootstrap  
summary(bootstrap\_df)

## xA xB   
## Min. :15.00 Min. :0.000   
## 1st Qu.:18.00 1st Qu.:4.000   
## Median :19.00 Median :5.000   
## Mean :19.52 Mean :4.455   
## 3rd Qu.:21.00 3rd Qu.:6.000   
## Max. :25.00 Max. :8.000

library(ROI.plugin.glpk)  
library(ompr)  
library(ompr.roi)  
  
result <- MIPModel() |>  
 add\_variable(xA, type = "integer", lb = 0) |>  
 add\_variable(xB, type = "integer", lb = 0) |>  
 set\_objective(9500 \* xA + 14000 \* xB, "max") |>  
 add\_constraint(10 \* xA + 20 \* xB <= 290) |>  
 add\_constraint(20 \* xA + 10 \* xB <= 450) |>  
 add\_constraint(10 \* xA + 15 \* xB <= 350) |>  
 solve\_model(with\_ROI(solver = "glpk"))  
  
get\_solution(result, xA)

## xA   
## 19

get\_solution(result, xB)

## xB   
## 5