Minimización de pérdida al aplicar un plan de simplificación

Trabajo Final de Introducción a la Estadística Computacional

Lucas Bizoso Germán Miranda

2023-11-16

Índice

1 Introducción

(incluir intro al problema)

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## here() starts at /Users/germanmiranda/Desktop/Proyectos R/Intro Est Comp/Intro_Est_Comp
```

2 Planteo del problema

Leemos los datos y graficamos para ver si hay algún tipo de correlación entre los precios y el ingreso que generan.

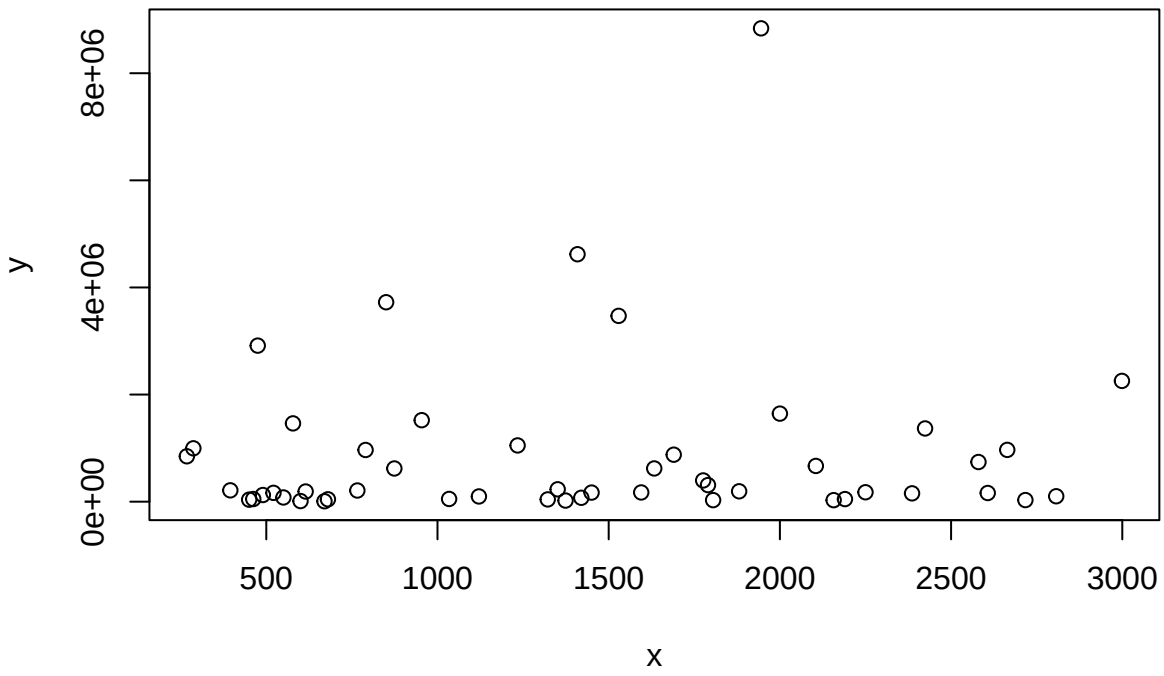
```
db = read_xlsx(here("Entrega Final","Base Productos.xlsx"))

db = db %>% mutate (Ingreso_Por_Plan = Suscriptores*Precio_Producto)   %>%
          arrange(Precio_Producto)

x=unlist(db$Precio_Producto)

y=unlist(db$Ingreso_Por_Plan)

plot(x,y)
```


y
0e+00
4e+06
8e+06
500
1000
1500
2000
2500
3000
x

3 Simulación para un caso puntual

Como primer paso, creamos una función que genera precios de planes aleatorios en base a tres parámetros:

- Cantidad de planes a crear
- Precio del plan máximo a crear que se incluirá también en el listado
- Precio del plan mínimo a crear que se incluirá también en el listado

Creamos dos funciones que en base a la cantidad nueva de planes que queremos simular, define la cantidad de suscriptores que entran en ese nuevo plan.

```
p_actuales=unlist(db$Precio_Producto)
q_actuales=unlist(db$Suscriptores)


planes_nuv <- function(n_final,min,max) {
    # n_final cantidad de planes menos dos
    # min precio para crear planes y el primer plan
    # max precio para crear planes y el último plan
    # devuelve n planes nuevos generados por runif
    p_nuevos_L = as.integer(runif(n_final-2,min+1,max-1))
    p_nuevos_L = p_nuevos_L[order(p_nuevos_L)]
    p_nuevos_L = c(min,p_nuevos_L,max)
    return(p_nuevos_L)
}

ganancia_nuv <- function(new_p,old_p,old_cli){
    # new_p vector planes nuevos
    # old_p vector planes viejos
    # old_cli vector actuales
    # devuelve: - distribucion nueva de los clientes
    #           - cuanto es la ganancia del plan nuevo
    #           - la ganancia del plan viejo
    #           - la diferencia entre ambas

    # nueva distrivucion de los clientes
    new_cli=rep(0,length(new_p))

    for(i in length(old_cli):1) {
        j = length(new_p)
        while(old_p[i]<new_p[j] && j!=1){
            j=j-1
        }
        new_cli[j] = new_cli[j] + old_cli[i]
    }
    # nuevas ganacias
    new_ingresos <- sum(new_p*new_cli)
    # viejas ganancias
    old_ingresos <- sum(old_p*old_cli)
    #diferencia (asumiendo old_ingresos > new_ingresos)
    diferencia = new_ingresos- old_ingresos

    res <- list(new_cli = new_cli ,
```

```
                new_ingresos = new_ingresos ,
                old_ingresos = old_ingresos ,
                diferencia = diferencia)
    return(res)
}


n=20
min=min(p_actuales)
max=max(p_actuales)

new_p=planes_nuv(n,min=min,max=max)

ganancia_nuv(new_p = new_p,old_p = p_actuales,old_cli=q_actuales )
```

4 Simulación para varios casos

4.1 Variando solamente precios

Creamos una función "simul_L" que en base a la cantidad de simulaciones que se quieran hacer (por defecto 1000)

```
simul_L <- function(n_final,min,max,old_p,old_cli,tol,cant_sim = 1000){
    # usa planes_nuv y ganancia_nuv
    # tol valor que hay que superar para ser aceptada la nueva distribución
    # cant_sim cantidad de simulaciones

    all <- matrix(0 , nrow = cant_sim , ncol = (n_final+2))
    # primeras n_final+2 columnas son los 20 planes
    # penúltima columna es la diferencia con los ingresos viejos
    # última columna es un T/F según se alcance la tolerancia

    for (i in 1:cant_sim) {
        plan_i <- planes_nuv(n_final,min,max)
        gan_i <- ganancia_nuv(plan_i,old_p,old_cli)
        t_f <- as.numeric(gan_i$diferencia>tol)
        vector <- c(plan_i,gan_i$diferencia,t_f)
        all[i,] <- vector
    }
    return(all)
}


res_sim <- simul_L(n_final=20,min=min,max=max,p_actuales,q_actuales,tol=-600000)

View(res_sim)


max(res_sim[,21])

plot(res_sim[,21])
```

4.2 Variando Precios y Cantidades de planes

```
# Simular con distintas cantidades de planes

simu_cant <- function(n_start,n_end,min,max,old_p,old_cli,tol,cant_sim = 1000) {
    # crea una matris con los datos del summary de la diferencia de matrices de la función simul_L
    # n_start es la cantidad de planes inicial y n_end la final, n_end > n_start1
    mat <- matrix(0, nrow = n_end-n_start+1 , ncol = 8)
    for(n in n_start:n_end){
        datos = simul_L(n,min,max,old_p,old_cli,tol)
        mat[(n-n_start+1),] = c(n,t(as.matrix(summary(datos[,n+1]))),sum(datos[,n]))
    }
    return(mat)
}
```

```r
mat <- simu_cant(18,80,200,3000,p_actuales,q_actuales,1000000)
View(mat)
```

5 SCRIPT AUXILIAR (DESESTIMAR PARA EL FINAL)

```
p_nuevos_tmp=as.integer(runif(n_final,min(p_actuales),max(p_actuales)))

#Concateno los 18 simulados, con el máximo y mínimo

p_nuevos=c(p_nuevos_tmp,min(p_actuales),max(p_actuales))

#Ordeno el vector de menor a mayor
p_nuevos=p_nuevos[order(p_nuevos)]


q_nuevos=c()

#Fijo la cantidad de Q para cada uno de los planes nuevos

for (j in 1:length(p_nuevos)) {
  q=0
  for (i in 1:length(p_actuales)) {
      if (p_actuales[i]>=p_nuevos[j]) {
          q = q + q_actuales[i]
          q_nuevos[j]= q
       }
    }
}

for (i in 1:(length(q_nuevos)-1)) {
  q_nuevos[[i]] = q_nuevos[[i]] - q_nuevos[[i+1]]
}


I_Final=sum(p_nuevos*q_nuevos)

y_final=p_nuevos*q_nuevos

# Porcentaje de pérdida

((I_Final-I_Inicial)/I_Inicial)*100
```

```
qq2=0
for (k in 1:5) {
  if (p_actuales[[k]]>=pp2){
    qq2= qq2 +q_actuales[[k]]
  }
}
qq2

qq3=0
for (k in 1:5) {
  if (p_actuales[[k]]>=pp3){
    qq3= qq3 +q_actuales[[k]]
  }
```

```
}
qq3

qq2= qq2-qq3

qq1 = qq1 -qq2 -qq3
```

5.1 EJEMPLO CON CASOS LIMITADOS

```
p1 = 200
p2 = 350
p3 = 500
p4 = 900
p5 = 1500

p_actuales = c(p1,p2,p3,p4,p5)

q1=5050
q2=700
q3=1000
q4=200
q5=700

q_actuales = c(q1,q2,q3,q4,q5)

sum(q_actuales)
```

```
## [1] 7650
```

```
Ibase = p1*q1 + p2*q2 + p3*q3 + p4*q4 + p5*q5

pn1=200
pn2=400
pn3=900

p_nuevos = c (pn1,pn2,pn3)

p_nuevos=as.integer(runif(20,min(p_actuales),max(p_actuales)))

q_nuevos=c()

for (j in 1:length(p_nuevos)) {
  q=0
  for (i in 1:length(p_actuales)) {
      if (p_actuales[i]>=p_nuevos[j]) {
          q = q + q_actuales[i]
          q_nuevos[j]= q
       }
    }
}
```

```
for (i in 1:(length(q_nuevos)-1)) {
  q_nuevos[[i]] = q_nuevos[[i]] - q_nuevos[[i+1]]
}

Ifinal=sum(p_nuevos*q_nuevos)
```