UNIVERSIDAD DE LA REPÚBLICA

Facultad de Ciencias Económicas y de Administración

Licenciatura en Estadística

Minimización de pérdida de ingresos al aplicar un plan de simplificación comercial

Miranda, Germán Bizoso, Lucas

Noviembre 2023

Trabajo final de Introducción a la Estadística Computacional

Índice

Introducción

Normalmente, cada cierto período de tiempo (mes, semestre, año, etc.) las empresas tienden a actualizar los productos y/o servicios que ofrecen, ya sea por mejoras en los beneficios que brindan, por innovación tecnológica de los mismos, por adaptarse a lo que ofrece el mercado en ese momento, entre otros.

En este contexto, habitualmente sucede que esos productos y/o servicios tienden a acumularse en el catálogo histótico ya que si inicialmente un cliente contrata uno de los productos, este, por contrato, deberá mantenerse hasta que el mismo cese ya sea por voluntad del cliente o decisión unilateral de la empresa.

A simple vista, esto no siempre puede ser un problema, pero en empresas de gran tamaño y de rubro de servicios si puede serlo, por varias razones:

- En caso de mejoras en los nuevos productos, esto supone que los clientes más antigüos, gocen de menos beneficios que los nuevos por el mismo precio. Esto también podría provocar insatisfacción, generando pérdidas de clientes que busquen otras opciones fuera de la compañía.
- Puede suponer un costo de mantenimiento ya que los servicios más antigüos también necesitan cierta actualización, como por ejemplo de precios, etiquetas de facturación, etc., así como también un costo de infraestructura para almacenar la infomación de esos productos que se van acumulando.

Por estos motivos, es habitual que cada cierto período se realicen maniobras de simplificación para reducir la cantidad de servicios activos en los sistemas y también para brindar estas versiones mejoradas a los cientes que quedan desactualizados.

Problema

Esta maniobra de simplificación puede suponer un riesgo, ya que los clientes contractualmente no pueden abonar más de lo acordado, dejando que la opción sea abonar lo mismo o menos. Dado este problema, se asume que siempre la empresa perderá un cierto margen pero el objetivo en el que ahondaremos en el trabajo es poder minimizar lo más posible la pérdida para que no sea un gran impacto.

Para poder resolver este problema, partimos de la situación inicial definiendo cómo se generan los ingresos y de qué variables dependen.

Podemos ver que los ingresos están definidos como:

$$I = \sum_{n=1}^{i} P_i Q_i$$

Donde P_i es el precio del servicio i-ésimo, y Q_i es la cantidad de clientes suscriptos a ese servicio.

En la siguiente tabla, podemos observar para una empresa ficticia, el listado actual de precios que tienen actualmente, la cantidad de suscriptos y el ingreso que se genera por plan.

Table 1: Tabla inicial de planes, precios y cantidades

Nombre_Producto	Precio_Producto	Suscriptores	Ingreso_Por_Plan
Producto 1	268	3162	847416
Producto 2	287	3471	996177
Producto 3	395	535	211325
Producto 4	450	84	37800
Producto 5	462	106	48972
Producto 6	475	6134	2913650
Producto 7	490	254	124460
Producto 8	521	316	164636
Producto 9	550	143	78650
Producto 10	578	2529	1461762
Producto 11	600	19	11400
Producto 12	615	308	189420
Producto 13	670	12	8040
Producto 14	680	65	44200
Producto 15	766	272	208352
Producto 16	790	1222	965380
Producto 17	850	4381	3723850
Producto 18	874	709	619666
Producto 19	954	1595	1521630
Producto 20	1034	49	50666

Con estos datos, podemos armar un gráfico que nos muestre, para cada plan, cuánto aportan al ingreso final.

En el gráfico, se puede observar que no hay una relación entre los precios de los planes y de los ingresos ya que no dependen solamente de su precio, si no de la cantidad de clientes que estén suscriptos.

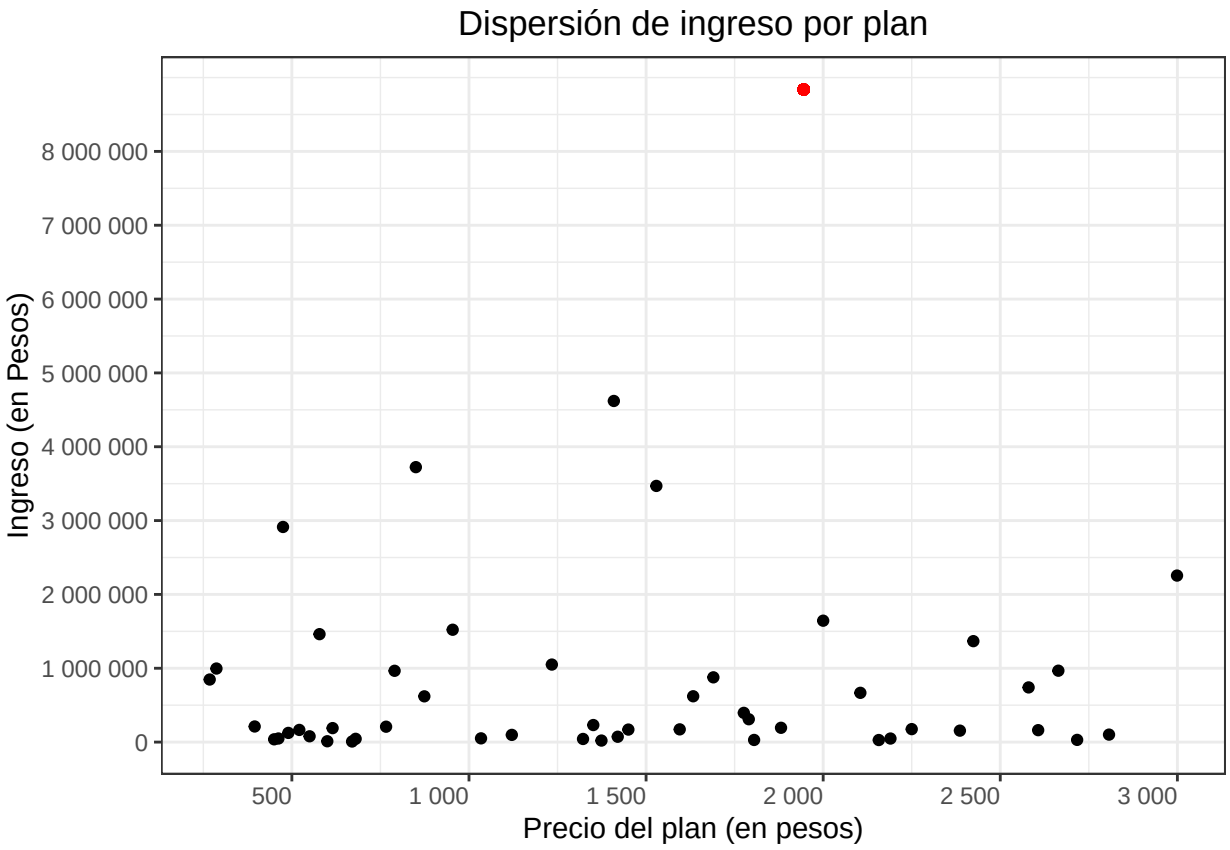

Figure 1: Dispersión de ingresos por plan comercial

Simulación para un caso puntual

(CONTINUAR ACÁ)

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=200,max=3000) {
    # n_final cantidad de planes
    # 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 = sort(rdunif(n_final-2,min,max))
    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 distribución 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,
                porc_dif = (diferencia)/old_ingresos)
    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)
```

Simulación para varios casos

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)
}



new_p
old_p

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

View(res_sim)

summary(res_sim)

res_sim

max(res_sim[,21])

plot(res_sim[,21])
```

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)
}

mat <- simu_cant(10,30,200,3000,p_actuales,q_actuales,1000)
View(mat)
```

```
sq <- c(1:nrow(eurodistmat), 1)

planes_nuv <- function(n_final=20,min=200,max=3000) {
    # 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)
}
set.seed(1)
new_p=planes_nuv(10,200,3000)
new_p
new_cli=rep(0,length(new_p))
old_cli=unlist(db$Suscriptores)
old_p=unlist(db$Precio_Producto)
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]
    }

sum(new_cli)
sum(old_cli)
ingresos(new_p)

sum(old_p*old_cli)


new_p =c(268,  435,  467,  541,  642,  685, 1095 ,1253 ,1436, 1696, 1897, 1938, 1991, 2230, 2499, 2510,


dif_ingresos= function(new_p) {
  i_inicial=43777406
  new_cli=c(7168,190, 6704 ,2999,   77, 8228,  938, 3548, 3393,  515,    0, 4544, 1174,  707,    0,  34
  i=sum(new_p*new_cli)
  i=i_inicial-i
   return(i)}
```

```
res <- optim(new_p, fn=dif_ingresos,method = 'SANN',
             control = list(maxit = 3600,  trace = TRUE,
                            REPORT = 500))


res
length(new_p)


sum(res$par*new_cli)
sum(new_cli*new_p)



vectorb <- new_p



genseq <- function(sq) {  # Generate new candidate sequence
  idx <- seq(2, NROW(eurodistmat)-1)
  changepoints <- sample(idx, size = 2, replace = FALSE)
  tmp <- sq[changepoints[1]]
  sq[changepoints[1]] <- sq[changepoints[2]]
  sq[changepoints[2]] <- tmp
  sq
}
```

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
```

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)
```

Anexo

Paquetes utilizados

```
## To cite package 'tidyverse' in publications use:
##
##   Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
##   Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller
##   E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V,
##   Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to
##   the tidyverse." _Journal of Open Source Software_, *4*(43), 1686.
##   doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {Welcome to the {tidyverse}},
##     author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang and Lucy D'Agosti
##     year = {2019},
##     journal = {Journal of Open Source Software},
##     volume = {4},
##     number = {43},
##     pages = {1686},
##     doi = {10.21105/joss.01686},
##   }

## To cite package 'here' in publications use:
##
##   Müller K (2020). _here: A Simpler Way to Find Your Files_. R package
##   version 1.0.1, <https://CRAN.R-project.org/package=here>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {here: A Simpler Way to Find Your Files},
##     author = {Kirill Müller},
##     year = {2020},
##     note = {R package version 1.0.1},
##     url = {https://CRAN.R-project.org/package=here},
##   }

## To cite package 'readxl' in publications use:
##
##   Wickham H, Bryan J (2023). _readxl: Read Excel Files_. R package
##   version 1.4.3, <https://CRAN.R-project.org/package=readxl>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {readxl: Read Excel Files},
##     author = {Hadley Wickham and Jennifer Bryan},
##     year = {2023},
##     note = {R package version 1.4.3},
```

```
##     url = {https://CRAN.R-project.org/package=readxl},
##   }

## To cite package 'dplyr' in publications use:
##
##   Wickham H, François R, Henry L, Müller K, Vaughan D (2023). _dplyr: A
##   Grammar of Data Manipulation_. R package version 1.1.4,
##   <https://CRAN.R-project.org/package=dplyr>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {dplyr: A Grammar of Data Manipulation},
##     author = {Hadley Wickham and Romain François and Lionel Henry and Kirill Müller and Davis Vaughan
##     year = {2023},
##     note = {R package version 1.1.4},
##     url = {https://CRAN.R-project.org/package=dplyr},
##   }

## To cite ggplot2 in publications, please use
##
##   H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
##   Springer-Verlag New York, 2016.
##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     author = {Hadley Wickham},
##     title = {ggplot2: Elegant Graphics for Data Analysis},
##     publisher = {Springer-Verlag New York},
##     year = {2016},
##     isbn = {978-3-319-24277-4},
##     url = {https://ggplot2.tidyverse.org},
##   }

## To cite package 'scales' in publications use:
##
##   Wickham H, Seidel D (2022). _scales: Scale Functions for
##   Visualization_. R package version 1.2.1,
##   <https://CRAN.R-project.org/package=scales>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {scales: Scale Functions for Visualization},
##     author = {Hadley Wickham and Dana Seidel},
##     year = {2022},
##     note = {R package version 1.2.1},
##     url = {https://CRAN.R-project.org/package=scales},
##   }

## To cite package 'ggeasy' in publications use:
##
```

```
##   Carroll J, Schep A, Sidi J (2023). _ggeasy: Easy Access to 'ggplot2'
##   Commands_. R package version 0.1.4,
##   <https://CRAN.R-project.org/package=ggeasy>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {ggeasy: Easy Access to 'ggplot2' Commands},
##     author = {Jonathan Carroll and Alicia Schep and Jonathan Sidi},
##     year = {2023},
##     note = {R package version 0.1.4},
##     url = {https://CRAN.R-project.org/package=ggeasy},
##   }
```