TAREA # 11
Frentes de Pareto

Natalia Berenice Pérez López

10 de noviembre de 2021

1. Objetivo

El objetivo de esta práctica es graficar el porcentaje de soluciones de Pareto (ojo, no es lo mismo que se grafica en el código ejemplo) como función del número de funciones objetivo con diagramas de violín combinados con diagramas de caja-bigote, verificando que diferencias observadas, cuando las haya, sean estadísticamente significativas. Razona en escrito a qué se debe el comportamiento observado.

2. Desarrollo

Para generar el código de esta práctica se realizaron algunas ideas y pruebas iniciales, las cuales se encuentran en mi repositorio en GitHub. Se inició tomando como base el código para obtener el frente de Pareto [1] y el código para realizar gráficas de violín [2], ambos revisados en clase. Las modificaciones que se le realizaron al código fueron: establecer un vector con las cantidades de funciones objetivo, agregar un ciclo `for` para variar dichas cantidades y otro para hacer 20 réplicas de cada cantidad de objetivos, también se agregó el código para obtener el porcentaje de soluciones Pareto y un `data.frame` para almacenar el valor de k, el número de réplica y el porcentaje correspondiente.

A continuación se muestra el fragmeto modificado en el código:

```
df = data.frame()
vc <- 4
md <- 3
tc <- 5
funciones <- c(2, 3, 4, 5) # cuantas funciones objetivo
obj <- list()
k = 0

for (j in funciones){
  k = j
  for (replica in 1:20){
    for (i in 1:k) {
      obj[[i]] <- poli(md, vc, tc)
    }
    minim <- (runif(k) > 0.5)
    sign <- (1 + -2 * minim) # neg -> min, pos -> max
    n <- 200 # cuantas soluciones aleatorias
    sol <- matrix(runif(vc * n), nrow=n, ncol=vc)
    val <- matrix(rep(NA, k * n), nrow=n, ncol=k)
    for (i in 1:n) { # evaluamos las soluciones
      for (j in 1:k) { # para todos los objetivos
        val[i, j] <- eval(obj[[j]], sol[i,])
      }
    }
    mejor1 <- which.max(sign[1] * val[,1])
    mejor2 <- which.max(sign[2] * val[,2])
    cual <- c("max", "min")
    no.dom <- logical() # TRUE/FALSE segun si nadie lo domina
    dominadores <- integer()
    for (i in 1:n) { # para cada asignacion
      d <- logical() # quienes le dominan (si / no)
```

```
      for (j in 1:n) { # para todos los demas
        d <- c(d, domin.by(sign * val[i,], sign * val[j,]))
      }
      cuantos <- sum(d)
      dominadores <- c(dominadores, cuantos)
      no.dom <- c(no.dom, sum(d) == 0) # nadie le domina (puros FALSE)
    }
    frente <- subset(val, no.dom) # solamente las no dominadas
    porcentaje = (length(frente[,1])/n)*100
    resultado = c(k, replica, porcentaje)
    df = rbind(df, resultado)
    names(df) = c("k", "Replica", "Porcentaje")
  }
}
```

Listing 1: Fragmento del código modificado.

Utilizando el `data.frame` obtenido con el código anterior se realizó el diagrama caja-bigote combinado con diagramas de violín para analizar como varía el porcentaje de soluciones Pareto al incrementar el número de funciones objetivo (Ver figura 1).

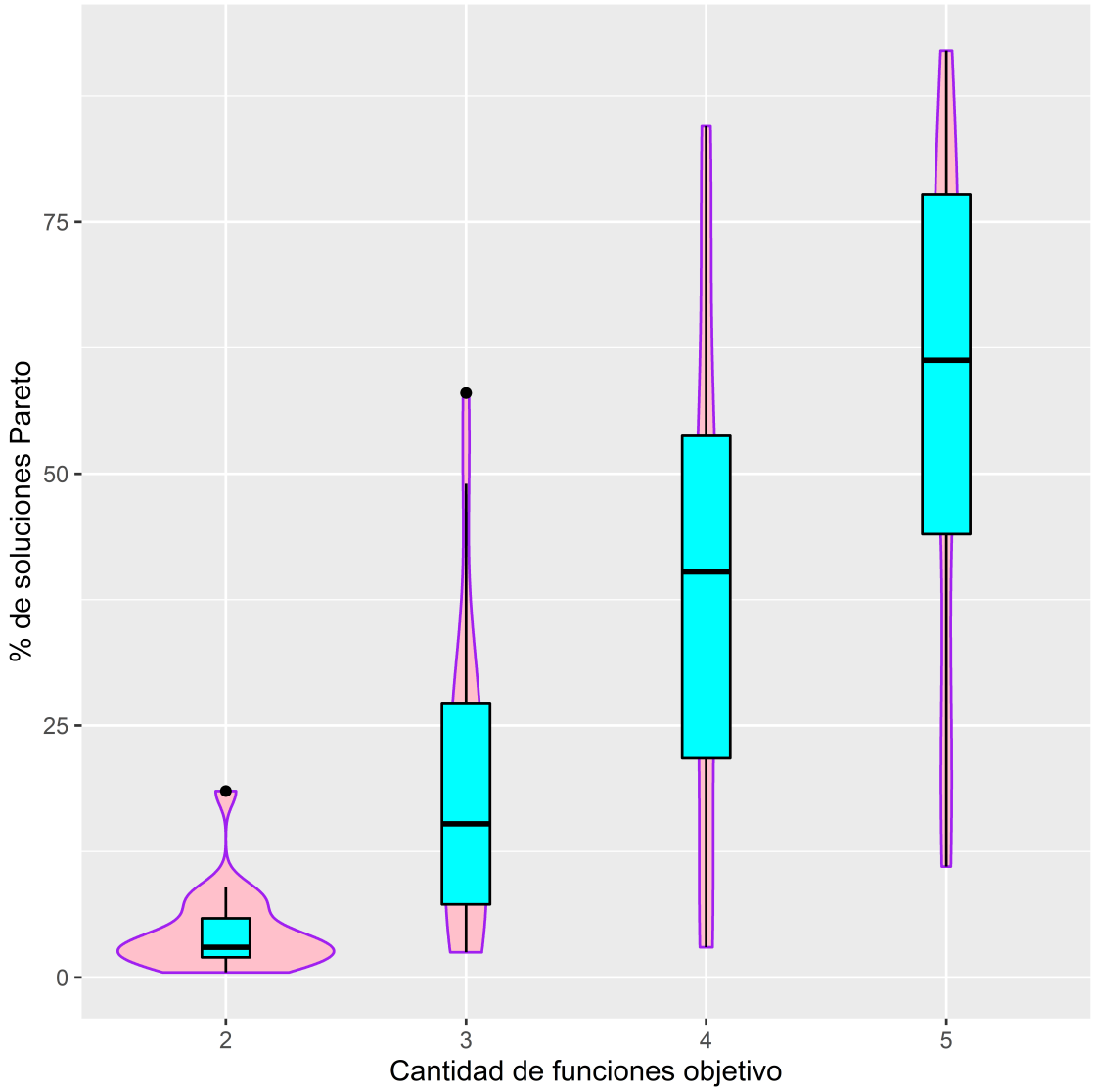


Figura 1: Porcentaje de soluciones de Pareto que se obtienen con cada cantidad de funciones objetivo k.

En la figura 1 se puede observar que con mayores cantidades de funciones objetivo se tienen más soluciones en el frente de Pareto. Para analizar si existe una relación entre la variación del número de funciones objetivo k y el porcentaje de soluciones en el frente de Pareto se realizó una prueba estadística. Se eligió realizar la prueba estadística `Kruskal Wallis` debido a que los datos no presentan normalidad.

En el cuadro 1 se resumen los resultados de la revisión de los supuestos para poder aplicar la prueba estadística. El supuesto outliers se refiere a la cantidad de valores atípicos que existen en los grupos, la normalidad por grupos se obtuvo con la prueba de `Shapiro Wilk` y la homogeneidad de varianza se obtuvo con la prueba de `Levene`.

Cuadro 1: Resultados del los supuestos para aplicar la prueba estadística.

Outliers	2
Normalidad por grupo	2: $p = 0{,}0003$ 3: $p = 0{,}0104$ 4: $p = 0{,}1990$ 5: $p = 0{,}0383$
Homogeneidad de varianza	$p = 1{,}9 \times 10^{-5}$

En los resultados se observa que para la normalidad en tres de los cuatro grupos p es menor a 0,05, por lo tanto no se tiene normalidad.

Al realizar la prueba `Kruskal Wallis` se obtienen los resultados mostrados en el cuadro 2.

Cuadro 2: Resultados al aplicar la prueba estadística `Kruskal Wallis`.

Chi cuadrada	Valor de p
45,369	$7{,}72 \times 10^{-10}$

Hipótesis nula : Las medias son iguales en todos los grupos.

Hipótesis alternativa: Debido a que $p < 0{,}05$ se rechaza la hipótesis nula, es decir que si existen diferencias significativas entre las medias de los grupos.

Se entiende entonces que el aumentar la cantidad de funciones objetivo si tiene un efecto significativo en la cantidad de soluciones en el frente de Pareto.

También podemos realizar la prueba de suma de rangos de `Wilcoxon` por pares para observar los resultados de p y determinar si existen diferencias al comparar entre ellos los valores de k (Ver cuadro 3).

Cuadro 3: Resultados al aplicar la prueba `Wilcoxon`.

Valor de p	2	3	4
3	$1{,}5 \times 10^{-4}$	-	-
4	$3{,}7 \times 10^{-5}$	0,03089	-
5	$6{,}3 \times 10^{-7}$	$9{,}7 \times 10^{-5}$	0,04018

En los resultados de la prueba podemos observar que todos los valores de p son menores a 0,05 lo cual indica que entre todos los grupos sí existen diferencias significativas en sus medias.

A continuación se muestra el código utilizado para realizar la figura 1 y la prueba estadística `Kruskal Wallis`:

```
library(ggplot2)
df$k = as.factor(df$k)
png("Figura1.png", width=15, height=15, units="cm", res=1200)
gr <- ggplot(df, aes(x=k, y=Porcentaje)) + geom_violin(fill="pink", color="purple")
gr + geom_boxplot(width=0.2, fill="cyan", color="black", lwd=0.5) +
  labs(x = "Cantidad de funciones objetivo", y = "% de soluciones Pareto")
graphics.off()

```

```r
library(tidyverse)
library(ggpubr)
library(car)
library(rstatix)
library(rapportools)
library(readr)
library(gridExtra)

#PRUEBA ESTADISTICA...
#Estadisticas descriptivas
df %>%
  group_by(k) %>%
  get_summary_stats(Porcentaje, type = "mean_sd")

#SUPUESTOS PARA ANOVA
#1:Outliers
df %>%
  group_by(k) %>%
  identify_outliers(Porcentaje)

#2:Normalidad por Shapiro
df %>%
  group_by(k) %>%
  shapiro_test(Porcentaje)

#3:Homogeneidad de varianza con prueba Levene
df %>%
  levene_test(Porcentaje~k)

#PRUEBA ESTADISTICA KRUSKAL WALLIS
kruskal.test(Porcentaje ~ k, data = df)

#PRUEBA WILCOXON
pairwise.wilcox.test(df$Porcentaje, df$k)
```

Listing 2: Código para graficar y realizar las pruebas estadísticas `Kruskal Wallis` y `Wilcoxon`.

3. Reto 1

El primer reto es seleccionar un subconjunto (cuyo tamaño como un porcentaje del frente original se proporciona como un parámetro) del frente de Pareto de tal forma que la selección esté diversificada, es decir, que no estén agrupados juntos en una sola zona del frente las soluciones seleccionadas. Graficar los resultados de la selección, indicando con un color cuáles se incluyen en el subconjunto diverso.

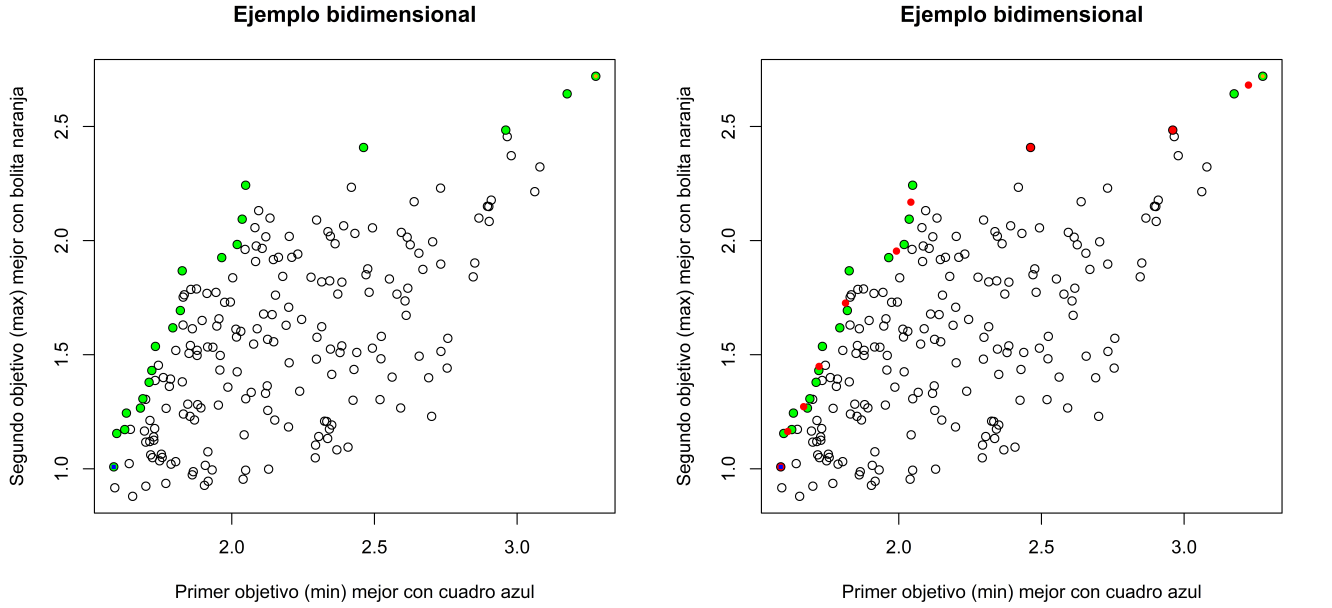
Se comenzó por revisar la explicación del reto 1 [3], posteriormente con apoyo de mis compañeros se revisaron algunas ideas de código y con apoyo del repositorio de Fabiola Vázquez [5] y revisando documentación de la función *kmeans* [4] se generó el código mostrado a continuación:

```
porcentaje=50
dispersos = kmeans(frente, round(dim(frente)[1]*porcentaje/100), iter.max = 1000, nstart = 50,
    algorithm = "Lloyd")
dispersos$cluster
dispersos$centers

png("p11_fdispersos.png", width=15, height=15, units="cm", res=1200)
plot(val[,1], val[,2], xlab=paste(xl, "mejor con cuadro azul"),
     ylab=paste(yl,"mejor con bolita naranja"),
     main="Ejemplo bidimensional")
points(frente[,1], frente[,2], col="green", pch=16, cex=0.9)
points(dispersos$centers[,1], dispersos$centers[,2], col="red", pch=16, cex=0.6)
mejor1 <- which.max((1 + (-2 * minim[1])) * val[,1])
mejor2 <- which.max((1 + (-2 * minim[2])) * val[,2])
points(val[mejor1, 1], val[mejor1, 2], col="blue", pch=15, cex=0.5)
points(val[mejor2, 1], val[mejor2, 2], col="orange", pch=16, cex=0.5)
graphics.off()
```

Listing 3: Código para diversificar las soluciones del frente de Pareto.

En la figura 2 inciso *a*) podemos ver el frente de Pareto original y en el inciso *b*) podemos ver la selección del 50 % de las soluciones del frente de Pareto diversificadas. En esta última figura se puede observar que algunos puntos están desfasados del punto original, esto se debe a que la función *kmeans* realiza los puntos como centroides de las soluciones más cercanas.



(a) Frente de Pareto original.

(b) Diversificación del 50 % de las soluciones del Frente de Pareto (círculos rojos).

Figura 2: Frente de Pareto para dos funciones objetivo.

4. Conclusión

Con base en el diagrama caja-bigote combinado con los diagramas de violín y con los resultados de la prueba estadística puedo concluir que el porcentaje de soluciones de Pareto aumenta cuando se tienen más funciones objetivo, podemos observar que con solo dos funciones objetivo el frente de Pareto se compone de menos del 25 % de las soluciones totales, sin embargo con cinco funciones objetivo el frente de Pareto puede considerarse casi la totalidad de las soluciones, llegando a tener más del 75 % de las soluciones, esto se debe principalmente a que con más funciones objetivo existen más combinaciones que pueden considerarse en el frente de Pareto aunque solo cuenten con algunos de los objetivos óptimizados.

En general ésta práctica me pareció sencilla de comprender y con muchas aplicaciones en la realidad.

Referencias

[1] Elisa Schaeffer. Pareto fronts, 2021. URL https://github.com/satuelisa/Simulation/blob/master/ParetoFronts/front.R.

[2] Elisa Schaeffer. Pareto fronts, 2021. URL https://github.com/satuelisa/Simulation/blob/master/ParetoFronts/violin.R.

[3] Elisa Schaeffer. P11: frentes de pareto (semana 12 de febrero-junio 2021), 2021. URL https://www.youtube.com/watch?v=Blek9NZsbU4&ab_channel=ElisaSchaeffer.

[4] Uniovideo. El algoritmo k-means aplicado a clasificación y procesamiento de imágenes., 2021. URL https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html.

[5] Fabiola Vázquez. Tarea 11, 2020. URL https://github.com/fvzqa/Simulacion/blob/master/Tarea11/Tarea11.pdf.