Reporte 11:
Frentes de Pareto

Jorge Torres

9 de mayo de 2022

Capítulo 1

Cantidad de Funciones Objetivo

1.1. Objetivo

En optimización multicriterio, a un mismo conjunto de variables se necesita asignarse valores de tal forma que se optimicen dos o más funciones objetivo, que pueden contradecir una a otra — una mejora en una puede corresponder en una empeora en otra. Además hay que respetar potenciales restricciones, si es que las haya.

El objetivo de la actividad consiste en graficar el porcentaje de soluciones de Pareto como función del número de funciones objetivo para $k \in [2, 3, 4, 5]$ con diagramas de violín combinados con diagramas de caja-bigote, verificando que diferencias observadas, cuando las haya, sean estadísticamente significativas.

1.2. Desarrollo

El desarrollo de la actividad se basa en el código implementado por E. Schaeffer, donde selecciona las mejores soluciones de un conjunto aleatorio para dos polinomios generados al azar [1]. La implementación completa se puede encontrar en el repositorio en GitHub de J. Torres [2]. En el código 1.1 se tienen tres funciones. La función `poli` crea una cantidad requerida de expresiones polinomiales aleatoriamente, que después son evaluadas para ciertas soluciones con la función `eval`. La tercera función, `domin.by`, revisa si las soluciones provistas son una mejora o empeora para cada una de las funciones objetivo, seleccionando únicamente las soluciones que son una mejora.

Código 1.1: Creación y Evaluación de Polinomios; Verificación de Soluciones Dominadas

```
poli <- function(maxdeg, varcount, termcount) {
  f <- data.frame(variable=integer(), coef=integer(), degree=integer())
  for (t in 1:termcount) {
    var <- sample(1:varcount, 1)
    deg <- sample(0:maxdeg, 1)
    f <-  rbind(f, c(var, runif(1), deg))
  }
  names(f) <- c("variable", "coef", "degree")
  return(f)
}

eval <- function(pol, vars) {
  value <- 0.0
  terms = dim(pol)[1]
  for (t in 1:terms) {
    term <- pol[t,]
    value <-  value + term$coef * vars[term$variable]^term$degree
  }
  return(value)
}

domin.by <- function(target, challenger) {
  if (sum(challenger < target) > 0) {
    return(FALSE)
  }
  return(sum(challenger > target) > 0)
}
```

Las líneas del código 1.2 establecen los parámetros iniciales de operación, con una cantidad de variables $vc = 4$,

el grado máximo de polinomio $md = 3$, y una cantidad de términos $tc = 5$. También se tienen una cantidad de funciones objetivo $k \in [2, 3, 4, 5]$.

Código 1.2: Parámetros Iniciales

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

Para cada cantidad de funciones objetivo se hacen 20 iteraciones del experimento, lo cual se inicializa en el código 1.3. En estas líneas también se evalúa una cantidad $n = 200$ soluciones creadas aleatoriamente para todas las funciones objetivo.

Código 1.3: Inicialización de Iteraciones y Evaluación de Soluciones

```
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)
    n <- 200
    sol <- matrix(runif(vc * n), nrow=n, ncol=vc)
    val <- matrix(rep(NA, k * n), nrow=n, ncol=k)
    for (i in 1:n) {
      for (j in 1:k) {
        val[i, j] <- eval(obj[[j]], sol[i,])
      }
    }
```

Para decidir cuáles soluciones son las que dominan y cuáles son las dominadas, se implementa el código 1.4, donde se utiliza un lógica de `VERDAD/FALSO` según los resultados de la función `domin.by` del código 1.1. Asímismo, se hace un conteo de las soluciones dominadas y no dominadas para calcular el porcentaje de soluciones que dominan, el cual se utiliza posteriormente para hacer las gráficas tipo violín.

Código 1.4: Decisión y Conteo de Soluciones Dominadas y No Dominadas

```
    mejor1 <- which.max(sign[1] * val[,1])
    mejor2 <- which.max(sign[2] * val[,2])
    cual <- c("max", "min")
    no.dom <- logical()
    dominadores <- integer()
    for (i in 1:n) {
      d <- logical()
      for (j in 1:n) {
        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)
    }
    frente <- subset(val, no.dom)
    porcentaje = (length(frente[,1])/n)*100
    resultado = c(k, replica, porcentaje)
    df = rbind(df, resultado)
    names(df) = c("k", "Replica", "Porcentaje")
  }
}
```

1.3. Resultado

La figura 1.1 muestra un ejemplo de cómo se ve el frente de soluciones de Pareto con dos funciones objetivo. Resulta bastante difícil graficar ejemplos para sistemas con más de dos funciones, además de que el frente no se podría apreciar tan fácilmente, por lo que no se incluyen de manera visual. Sin embargo, estos sistemas sí se toman

en cuenta al realizar los diagramas de tipo violín, mostrando las distribuciones de los porcentajes de soluciones dominantes, lo cual se observa en la figura 1.2.

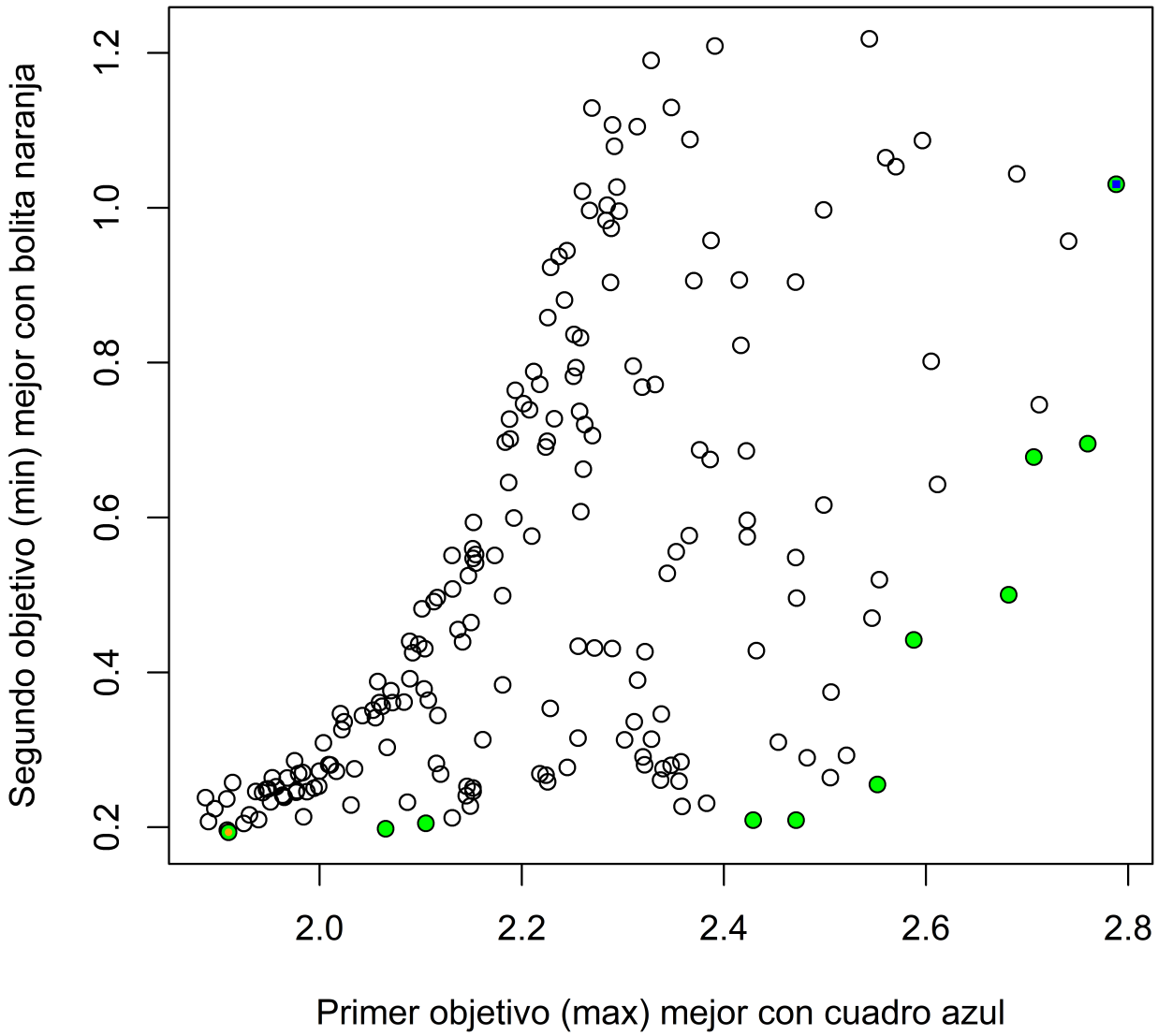


Figura 1.1: Frente de Pareto para un sistema de dos funciones objetivo. Los puntos verdes representan aquellas soluciones que no fueron dominadas por ninguna otra.

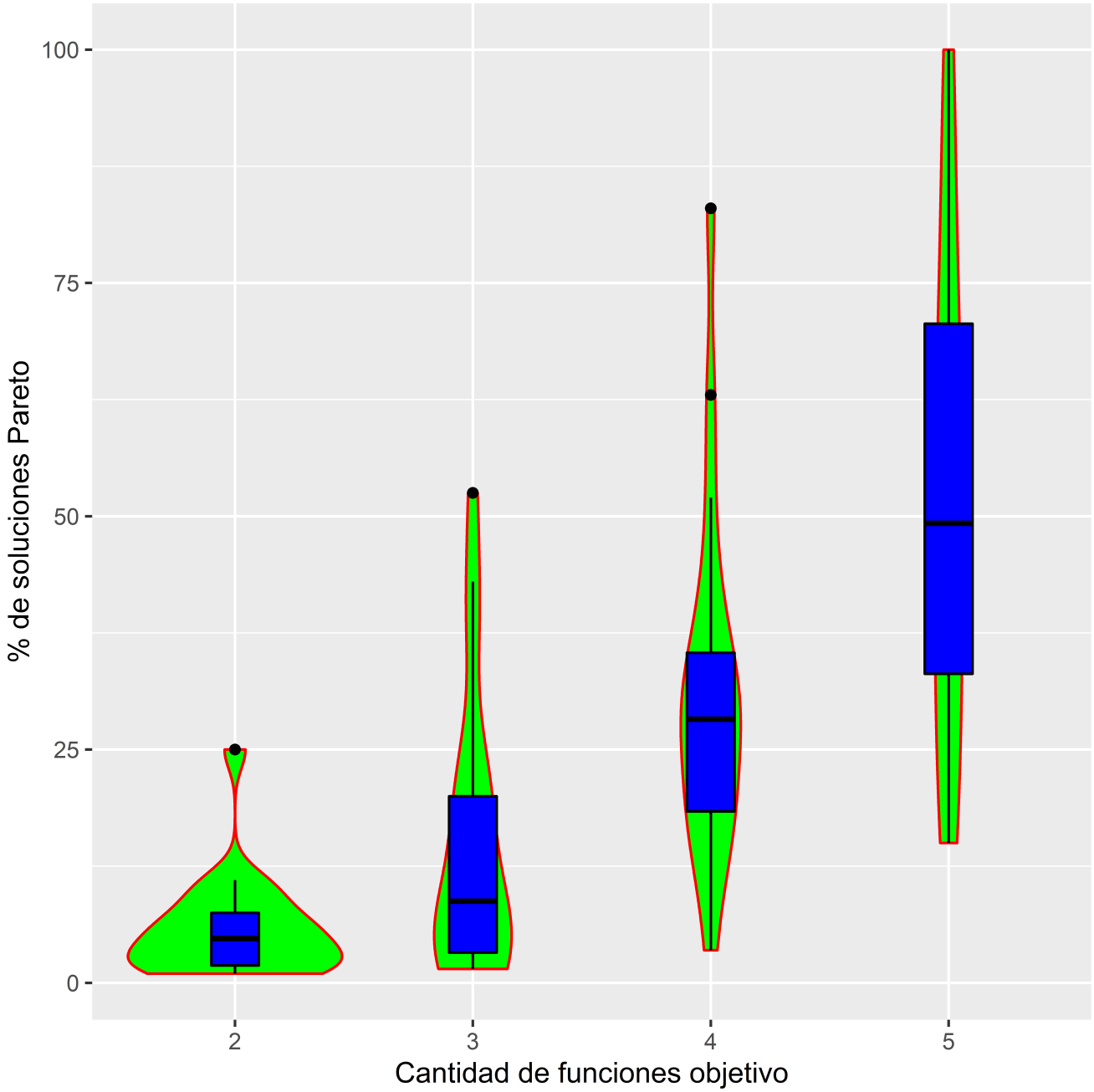


Figura 1.2: Distribuciones de los porcentajes de soluciones dominantes para cada cantidad de funciones objetivo.

Al realizar un análisis de varianza entre los porcentajes arrojados, se encuentra un valor p mucho menor a 0.05, lo cual prueba que, en efecto, existe una diferencia significativa al aumentar la cantidad de funciones objetivo.

1.4. Conclusiones

Como se puede observar del diagrama de la figura 1.2, para sistemas con solamente dos funciones objetivo, la el porcentaje de soluciones que no son dominadas es bastante bajo. En otras palabras, es mucho más encontrar soluciones óptimas para una menor cantidad de funciones. Sin embargo, conforme se aumenta la cantidad de funciones a optimizar, el rango de porcentajes de soluciones no dominadas se vuelve bastante amplio y uniformemente distribuido, teniendo desde $\sim 10\,\%$ de soluciones hasta el $100\,\%$ de ellas. Esto podría deberse a la forma en que se desean optimizar las funciones. Si todas ellas se quieren minimizar o maximizar, es fácil encontrar soluciones que se encuentren muy cerca del óptimo para todas. Pero si algunas de ellas se maximizan mientras otras se minimizan, es más complicado encontrar soluciones que sean dominadas por otras.

Capítulo 2

Reto 1 - Soluciones Diversificadas

2.1. Objetivo

El objetivo del primer reto es el 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.

2.2. Desarrollo

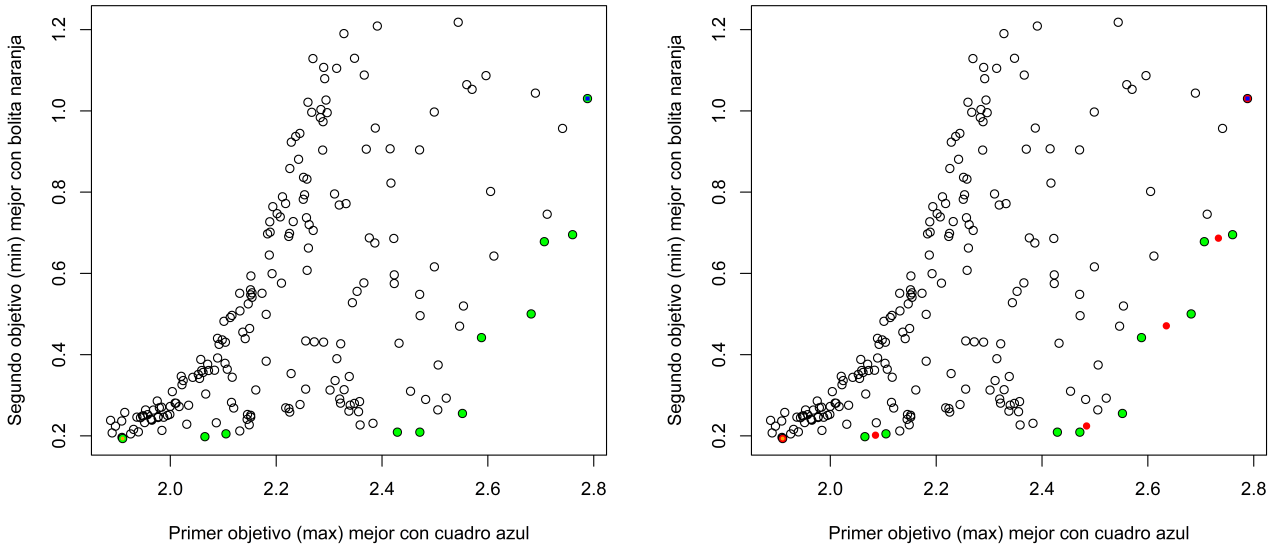
El desarrollo del reto es muy similar a la actividad del capítulo 1. Para simplificación de cálculo y visualización, la cantidad de funciones objetivo se reduce a $k = 2$. De la misma forma vista en los códigos 1.1 al 1.4, se crean funciones polinomiales y soluciones de manera aleatoria, se decide si las funciones se maximizan o minimizan y se seleccionan las funciones que dominen a todas las demás del conjunto. La diferencia radica en que ahora se toma un subconjunto de soluciones cuyo tamaño depende de un porcentaje del frente original. Esto se logra mediante la implementación del código 2.1. Estos nuevos puntos se utilizan posteriormente para graficar las soluciones.

Código 2.1: Selección Diversificada de Nuevas Soluciones

```
porcentaje=50
dispersos = kmeans(frente, round(dim(frente)[1]*porcentaje/100), iter.max = 1000, nstart = 50,
    algorithm = "Lloyd")
dispersos$cluster
dispersos$centers
mejor1 <- which.max((1 + (-2 * minim[1])) * val[,1])
mejor2 <- which.max((1 + (-2 * minim[2])) * val[,2])
```

2.3. Resultados

La figura 2.1a muestra el frente de Pareto para las dos funciones creadas aleatoriamente, donde las soluciones dominantes se resaltan en color verde. La primera función se requiere maximizar, mientras la segunda se desea minimizar. Las mejores soluciones para cada función se resaltan con un cuadro azul y una bola naranja, respectivamente. Por otro lado, la figura 2.1b muestra el subconjunto seleccionado de nuevas soluciones, donde éstas se marcan con bolas rojas.



(a) Frente de soluciones de Pareto para el sistema de dos funciones objetivo.

(b) Subconjunto de soluciones dispersas seleccionadas.

Figura 2.1: Frente de Pareto y subconjunto de nuevas soluciones.

Para visualizar la cantidad y frecuencia de estas soluciones dominantes, se implementa un diagrama de tipo violín en conjunto con uno de caja-bigote, como se puede apreciar en la figura 2.2.

2.4. Conclusiones

El método de las soluciones dispersas es útil para encontrar nuevas soluciones en la optimización de funciones, las cuales no se encuentran estrictamente dentro de un conjunto dado, pero sí muy cerca de las soluciones previamente encontradas.

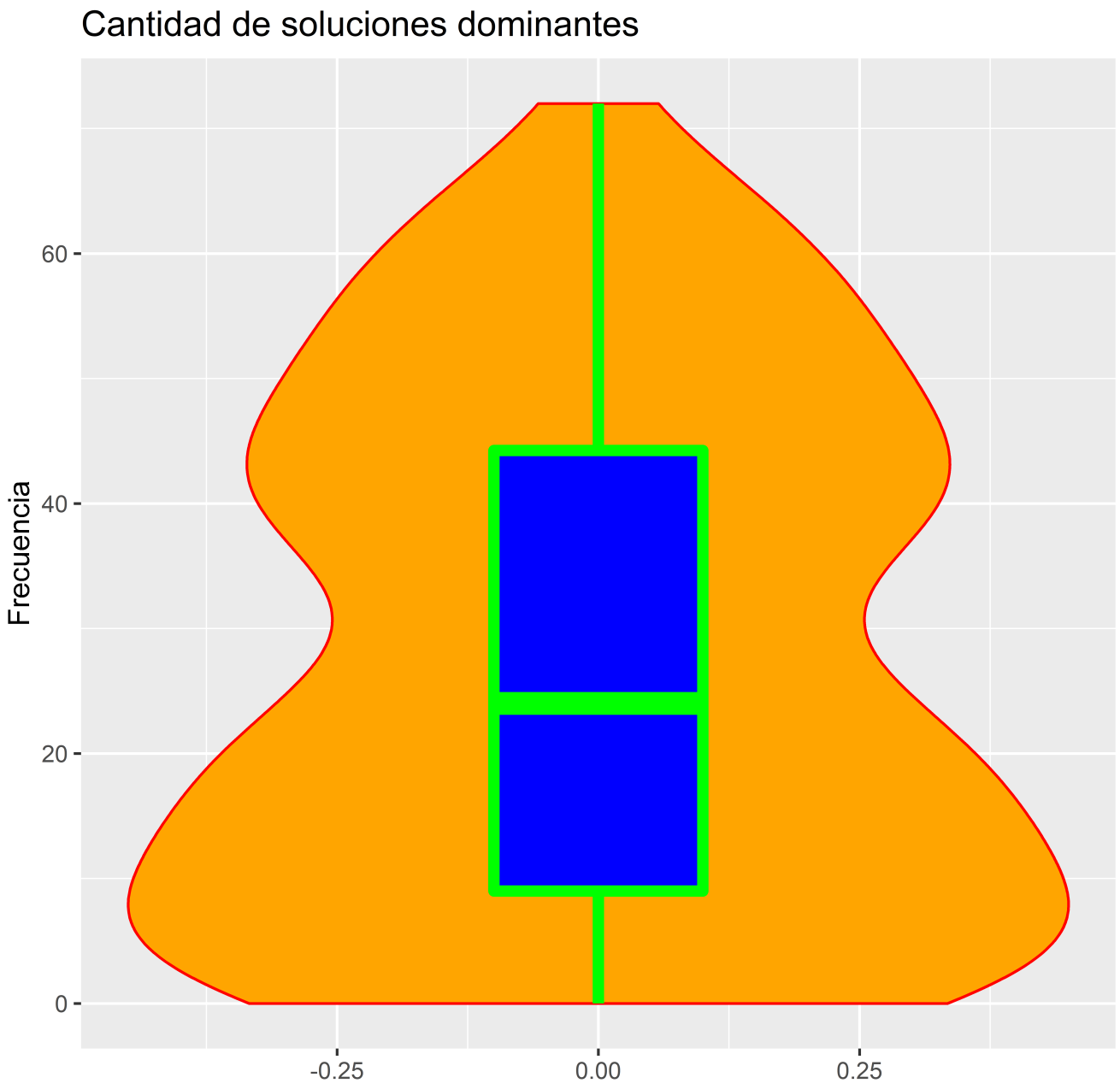


Figura 2.2: Cantidad y frecuencia de soluciones dominantes para el frente de Pareto y el subconjunto de soluciones.

Bibliografía

[1] E. Schaeffer. Paretofronts, 2019. URL https://github.com/satuelisa/Simulation/tree/master/ParetoFronts.

[2] J. Torres. P_11, 2022. URL https://github.com/FeroxDeitas/Simulacion-Nano/tree/main/Tareas/P_11.