TAREA # 7
Búsqueda local

Natalia Berenice Pérez López

13 de octubre de 2021

1. Objetivo

El objetivo de esta práctica es crear una visualización (animada) de cómo proceden por lo menos 5 réplicas simultáneas de la búsqueda encima de una gráfica de proyección plana, y después estudiar estadísticamente el efecto que tiene el largo de paso máximo en la cantidad de iteraciones que se requiere para llegar por primera vez al óptimo de la zona de estudio.

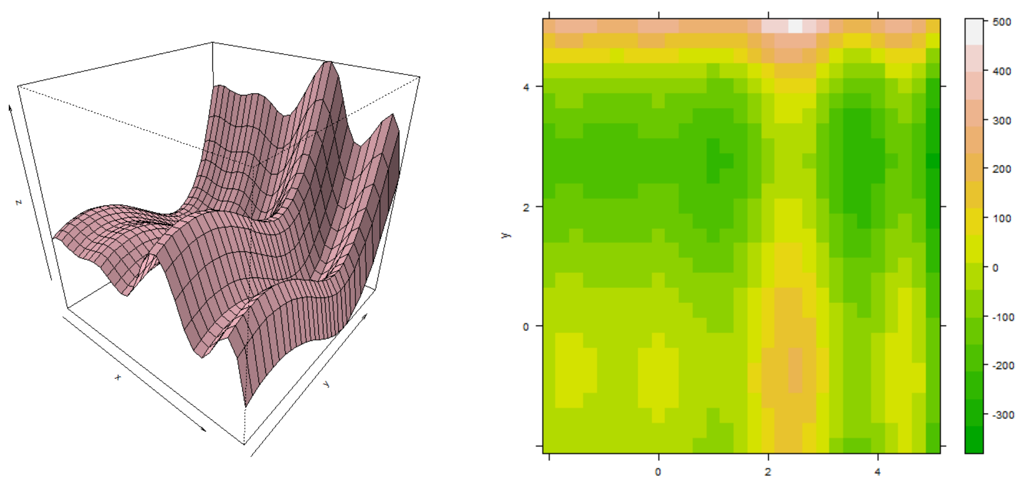
2. Desarrollo

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 las réplicas en ejecución paralela en `Rstudio` [4]. Las modificaciones que se le realizaron al código fueron: variar la función ejemplo y hacerla bidimensional, restringir la función en un intervalo máximo para x, y, con la técnica del ejemplo unidimensional hacer que los puntos rojos busquen el máximo global de la función en lugar del mínimo, para esto se utilizó apoyo del repositorio de Montemayor [1], también se agregó un ciclo `for` para variar el *paso* con el que se mueven los puntos rojos y otro ciclo `for` para hacer repeticiones del código y así poder analizar el efecto del *paso* en un diagrama caja-bigote.

La variante de la función ejemplo bidimensional se muestra en la función (1).

$$f(x, y) = sin(2x) * ((x + 1)^4 - 30x^2 - 50x) + (y + 1)^4 - 30y^2 - 50y \tag{1}$$

Las restricciones son $-2 \leq x, y \leq 5$. En la figura 1 (a) muestra la función (1) graficada en tres dimensiones y (b) muestra una proyección del plano xy donde el valor z se representa con colores.



(a) Función graficada en 3D.

(b) Proyección del plano xy.

Figura 1: Visualización gráfica de la función (1).

A continuación se muestra el código para obtener la función bidimensional graficada en 3D:

```
g <- function(x,y) {
  return((sin(x*2) * ((x + 1)^4 - 30 * x^2 -50 * x) + (y + 1)^4 - 30 * y^2 - 50 * y))
}
x <- seq(-2, 5, 0.25)
y <- x
z <- outer(x, y, g)
png("p7_2d.png", width = 700, height = 700)
persp(x, y, z, shade = 0.2, col = 'pink', theta = 40, phi = 30)
graphics.off()
```

Listing 1: Código graficar la función (1) en 3D.

El código para obtener la proyección del plano xy de la función (1) es el siguiente:

```
g <- function(x,y) {
  return((sin(x*2) * ((x + 1)^4 - 30 * x^2 -50 * x) + (y + 1)^4 - 30 * y^2 - 50 * y))
}
x <- seq(-2, 5, 0.25)
y <- x
z <- outer(x, y, g)
dimnames(z) = list(x, y)
library(reshape2)
d = melt(z)
names(d) = c("x", "y", "z")
library(lattice)
png("p7_flat_2.png", width = 500, height = 500)
levelplot(z ~ x * y, data = d, col.regions = terrain.colors(100))
graphics.off()
```

Listing 2: Código para la proyección del plano xy.

Enseguida se muestra el código objetivo de la práctica:

```
library(lattice)
library(sp)
library(viridisLite)
library(reshape2)
library(ggplot2)
library(tidyverse)
library(ggpubr)
library(car)
library(rstatix)
library(rapportools)
library(readr)
library(gridExtra)

g <- function(x,y) {
  return((sin(x*2) * ((x + 1)^4 - 30 * x^2 -50 * x) + (y + 1)^4 - 30 * y^2 - 50 * y))
}

x <- seq(-2, 5, 0.25)
y <- x
z <- outer(x, y, g)

low <- -2
high <- 5
pasos <- seq(0.25, 2, 0.25) #variar el paso
replicas <- 50 #cantidad de puntos rojos
df<- data.frame()

for (step in pasos) {
  for (repetir in 1:30){ #hacer repeticiones del experimento
    replica <- function(t) {
      curr <- c(runif(1, low, high), runif(1, low, high))
      best <- curr
      for (tiempo in 1:t) {
        delta <- runif(1, 0, step)
        izq <- curr +c(-delta,0)
        der <- curr + c(delta,0)
        arr <- curr + c(0,-delta)
        aba <- curr + c(0,delta)

        coord <- c(izq, der, arr, aba)
```

```r
      for(p in 1:8){
        if(coord[p] < (-2)){
          coord[p] <- coord[p]+5
        }
        if(coord[p] > 5){
          coord[p] <- coord[p]-2
        }
      }

      vx<-c()
      vy<-c()
      for(q in 1:8){
        if(q %% 2 == 0){
          vy <- c(vy,coord[q])
        }else{
          vx <- c(vx,coord[q])
        }
      }

      vg<- c()
      for(k in 1:4){
        vg <- c(vg, g(vx[k], vy[k]) )
      }

      pmax <- which.max(vg)
      curr <- c(vx[pmax], vy[pmax])
      if(g(curr[1],curr[2]) > g(best[1],best[2])){
        best <- curr
      }
    }
    return(best)
  }

  suppressMessages(library(doParallel))
  registerDoParallel(makeCluster(detectCores(logical = FALSE) - 2))

  for (pot in 1:40) { #iteraciones
    tmax <- pot
    resultados <- foreach(i = 1:replicas, .combine=c) %dopar% replica(tmax)

    vx<- c()
    vy<- c()
    aux<-(2*replicas)
    for(q in 1:aux){
      if(q %% 2 == 0){
        vy <- c(vy,resultados[q])
      }else{
        vx <- c(vx,resultados[q])
      }
    }

    val <- c()
    for(k in 1:replicas){
      val <- c(val, g(vx[k], vy[k]))
    }

    maximo <- which.max(val)
    x <- seq(-2, 5, 0.25)
    y <-  x
    z <- outer(x, y, g)
    dimnames(z) <- list(x, y)
    d <- melt(z)
    names(d) <- c("x", "y", "z")

    if (repetir == 1 & step == 0.25){
      png(paste0("t7_", tmax, ".png", sep=""), width=500, height=500)
      plot(levelplot(z ~ x * y, data = d, col.regions = terrain.colors(100)))
      trellis.focus("panel", 1, 1, highlight=FALSE)
      lpoints(vx, vy, pch=20, col="red", cex=2) #puntos rojos
      trellis.unfocus()
      trellis.focus("panel"[1], 1, 1, highlight=FALSE)
      lpoints(vx[maximo], vy[maximo], pch=20, col="blue",cex=3) #raya azul
      trellis.unfocus()
```

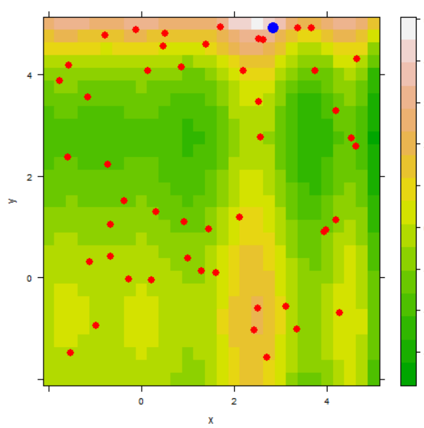
```
        graphics.off()
      }
    }
    ultimo<-min(val)
    df<- rbind(df,c(step,repetir,ultimo))
  }
}
stopImplicitCluster()

#GRAFICAR
names(df) <- c("Paso", "Repeticion", "Minimo")
df$Paso = as.factor(df$Paso)
ggplot(df, aes(x= Paso, y= Minimo, fill= Paso)) +
  geom_boxplot()+
  labs(x = "Paso", y = "Valor minimo")
```

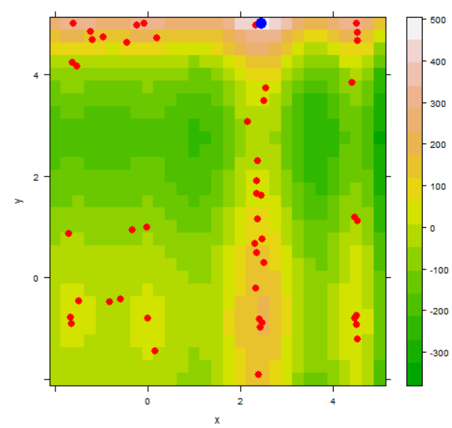
Listing 3: Código objetivo de la práctica.

Con el código anterior los 50 puntos rojos buscan alcanzar el punto máximo de la función bidimensional (1), además se varía el *paso* de los puntos rojos en valores de 0.25, 0.5, 0.75, 1, 1.25, 1.5. 1.75 y 2, para cada *paso* se hacen 30 repeticiones del exprimento lo cual permite analizar con que *paso* los puntos rojos llegan con más éxito al máximo de la función.

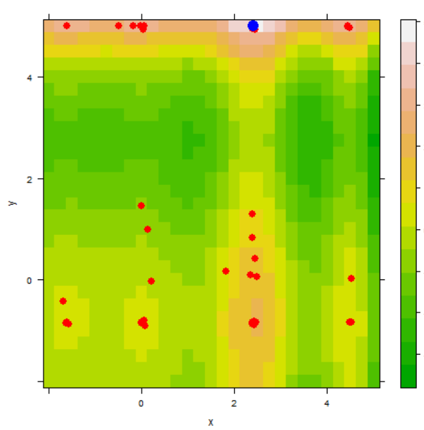
La figura 2 muestra los resultados obtenidos con un $paso = 0{,}25$, se pueden ver los 50 puntos rojos y un punto azul que indica el valor máximo al que han llegado los puntos rojos en las 40 iteraciones.



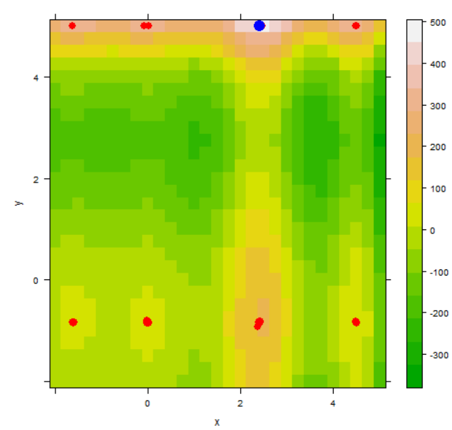
(a) Paso 1.



(b) Paso 10.



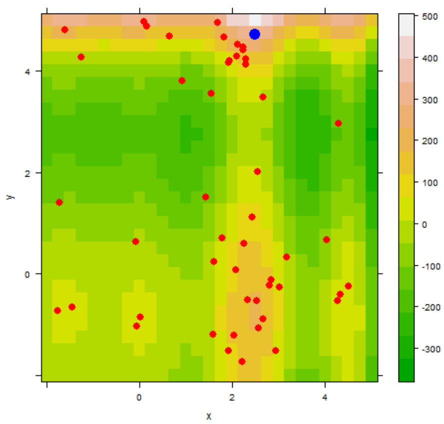
(c) Paso 20.



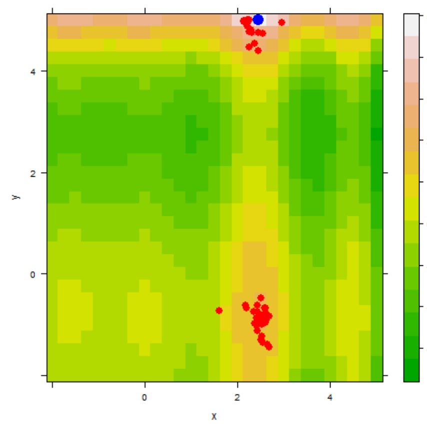
(d) Paso 40.

Figura 2: Búsqueda local de la función (1) con un $paso = 0{,}25$.

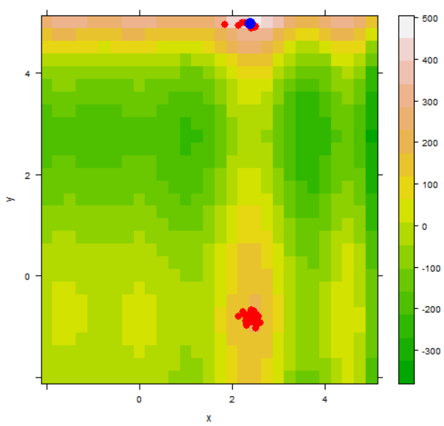
En la figura 3 se muestran los resultados obtenidos con un $paso = 2$.



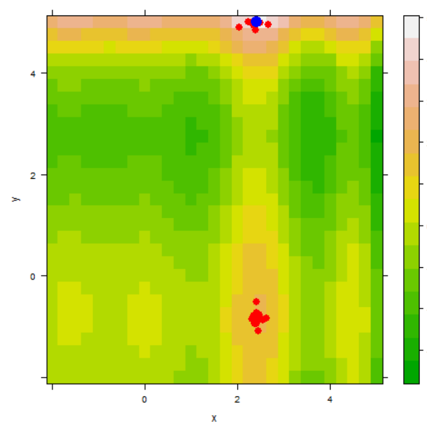
(a) Paso 1.



(b) Paso 10.



(c) Paso 20.



(d) Paso 40.

Figura 3: Búsqueda local de la función (1) con un *paso* = 2.

Los gifs [2] de las búsquedas locales de cada *paso* se encuentran mi repositorio en GitHub.

En la figura 3, la cual tiene un *paso* de 2, se puede ver que una mayor cantidad de puntos rojos lograr alcanzar el punto máximo de la función bidimensional, mientras que con un *paso* de 0,25 (ver figura 2) varios puntos rojos se quedan en zonas más bajas de la función.

Para analizar visualmente el valor mínimo al que llegan los puntos rojos con cada *paso* se realizó un diagrama caja-bigote con las 30 replicas del exprimento para cada *paso* (ver figura 4). En el diagrama podemos observar que con pasos grandes como 1,5, 1,75 y 2 los puntos rojos que no logran llegar al punto máximo de la función (punto azul), si logran llegar a zonas altas.

Para analizar si existe una relación entre la variación del *paso* y qué tan alto pueden llegar los puntos rojos que no logran llegar al punto máximo de la función (1) se realizó una prueba estadística. Primero se elegió utilizar la prueba estadística ANOVA de una vía, pero debido a los resultados obtenidos al revisar la normalidad de los datos, se eligió realizar la prueba estadística `Kruskal Wallis`.

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`.

.

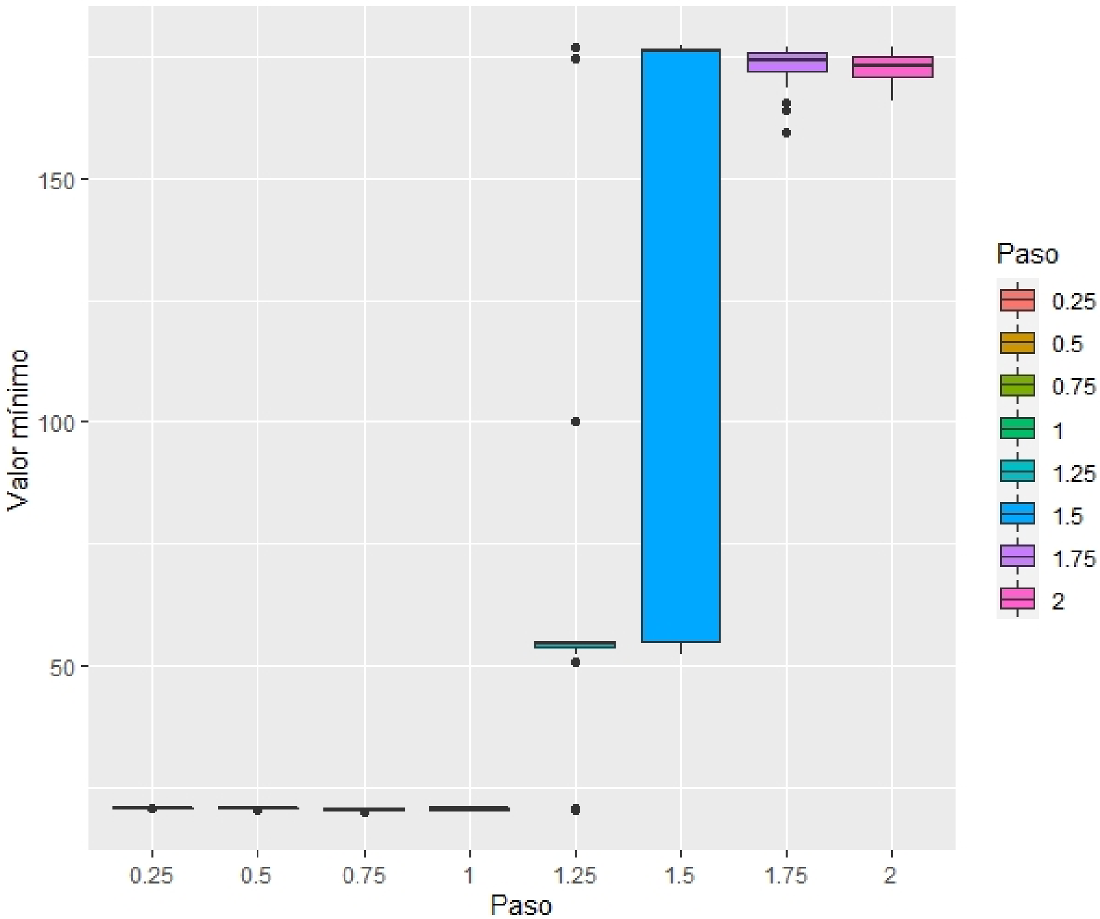


Figura 4: Valor mínimo al que llegan los puntos rojos en cada *paso*.

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

Outliers	14
Normalidad por grupo	0,25: $p = 0{,}004$ / 0,50: $p = 0{,}106$ / 0,75: $p = 0{,}010$ / 1: $p = 0{,}061$ / 1,25: $p = 1{,}05 \times 10^{-8}$ 1,5: $p = 1{,}68 \times 10^{-7}$ / 1,75: $p = 7{,}72 \times 10^{-5}$ / 2: $p = 0{,}122$
Homogeneidad de varianza	$p = 1{,}18 \times 10^{-11}$

En los resultados se observa que para la normalidad por grupos existen valores de p menores a 0,05, por lo tanto no se tiene normalidad, así que es necesario realizar la prueba estadística `Kruskal Wallis`.

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
197,35	$2{,}2 \times 10^{-16}$

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 la variación del *paso* en el movimiento de los puntos rojos si tiene un efecto significativo en qué tan alto logran llegar en la función bidimensional.

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

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

Valor de p	0,25	0,50	0,75	1,0	1,25	1,50	1,75
0,50	$1{,}1 \times 10^{-5}$	-	-	-	-	-	-
0,75	$4{,}7 \times 10^{-12}$	$1{,}6 \times 10^{-5}$	-	-	-	-	-
1,0	0,0005	0,7487	0,5131	-	-	-	-
1,25	$5{,}5 \times 10^{-9}$	$8{,}2 \times 10^{-11}$	4×10^{-12}	$3{,}5 \times 10^{-11}$	-	-	-
1,50	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$2{,}2 \times 10^{-5}$	-	-
1,75	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$1{,}3 \times 10^{-10}$	1	-
2,0	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$4{,}7 \times 10^{-16}$	$1{,}7 \times 10^{-10}$	1	1

En los resultados de la prueba podemos observar que la mayoría de los valores de p son menores a 0,05, solo existen algunas excepciones en las cuales se tiene una $p > 0{,}05$ como por ejemplo en la $paso = 1$ comparada con la $paso = 0{,}5$, lo cual indica que entre ellas no existen diferencias significativas en sus medias.

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

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

#Estadisticas descriptivas
df %>%
  group_by(Paso) %>%
  get_summary_stats(Minimo, type = "mean_sd")

#SUPUESTOS PARA ANOVA
#1:Outliers
df %>%
  group_by(Paso) %>%
  identify_outliers(Minimo)

#2:Normalidad por Shapiro
df %>%
  group_by(Paso) %>%
  shapiro_test(Minimo)

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

#PRUEBA ESTADISTICA KRUSKAL WALLIS
kruskal.test(Minimo ~ Paso, data = df)

#PRUEBA WILCOXON
pairwise.wilcox.test(df$Minimo, df$Paso)
```

Listing 4: Código para las pruebas estadísticas `Kruskal Wallis` y `Wilcoxon`.

.

3. Conclusión

Con base en las gráficas de las iteraciones de cada *paso*, el diagrama caja-bigote y los resultados obtenidos de las pruebas estadísticas `Kruskal Wallis` y `Wilcoxon` puedo concluir que la variación del *paso* con que se mueven los puntos rojos si afecta significativamente el valor mínimo que alcanzarán los puntos rojos que no llegan al máximo de la función bidimensional, además se oberva que que a mayor *paso* más rápidamente llegan los puntos rojos al máximo de la función (punto azul).

En general en ésta práctica se me dificultó realizar el código y entender el propósito de la prueba estadística, sin embargo con el apoyo de mis compañeros logré realizarla.

.

Referencias

[1] María Montemayor. Búsqueda local, 2021. URL https://github.com/MariaMontemayor/Simul/tree/main/tarea7.

[2] PHOTOGRAMIO. Crear gif animados. URL https://photogramio.com/es/gif-maker#ezgif.

[3] RDocumentation. Wilcox.test: Wilcoxon rank sum and signed rank tests. URL https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/wilcox.test.

[4] Elisa Schaeffer. Localsearch, 2021. URL https://github.com/satuelisa/Simulation/blob/master/LocalSearch/replicas.R.