Práctica 3:
Teoría de colas

Fabiola Vázquez

8 de octubre de 2020

---

1. Experimento

El objetivo de esta práctica [3] es examinar si los tiempos de ejecución de la *tarea* son afectados por los diversos ordenamientos de un vector (ascendente, descendente, pseudoaleatorio, primero los primos, al final los primos) o por las variaciones de los núcleos que se asignan al cluster. Dicho vector, que contiene números primos extraídos de https://primes.utm.edu/lists/small/millions/, contiene el 50 % de números primos. También, se analiza si la proporción de primos en el vector afecta los tiempos de ejecución. En cada experimento se realizan 50 réplicas. Dicho experimento se realiza con el software R versión 3.7.9 [2] en un cuaderno de Jupyter [1].

En la primera parte del experimento, se considera como *tarea* determinar si un número es primo, para ello se utiliza la función descrita en el código 1, elaborado por la Dra. Elisa Schaeffer [4].

Código 1: Función para determinar si un número es primo.

```
primo <- function(n){
    if (n == 1 || n == 2) {
        return(TRUE)
    }
    if (n%% 2 == 0) {
        return(FALSE)
    }
    for (i in seq(3, max(3, ceiling(sqrt(n))), 2)) {
        if ((n%% i) == 0) {
            return(FALSE)
        }
    }
    return(TRUE)
}
```

Cuadro 1: Valores p obtenidos de la prueba de Kruskal-Wallis en el primer experimento.

Datos	valor p
Tiempo $\sim$ Núcleo	0.0076
Tiempo $\sim$ Orden	0.4490

Cuadro 2: Valores p obtenidos de la prueba de Kruskal-Wallis en el experimento de variación del porcentaje de primos en el vector.

Datos	valor p
Tiempo $\sim$ Núcleo	0.0016
Tiempo $\sim$ Porcentaje de Primos	0.2926

En la figura 1 se muestran gráficos de caja en donde se tienen los tiempos de cada uno de los cinco órdenes del vector variando los núcleos. Como se puede apreciar en dicha figura, no parece haber gran diferencia entre los tiempos de ejecución al variar los núcleos utilizados.

Realizamos una prueba de Kruskal-Wallis para verificar si los núcleos o el tipo de orden del vector afectan los tiempos de ejecución. En el cuadro 1 se tienen los valores p obtenidos con dicha prueba, con lo que se concluye que la cantidad de núcleos no afecta los tiempos de ejecución y los órdenes del vector sí afectan.

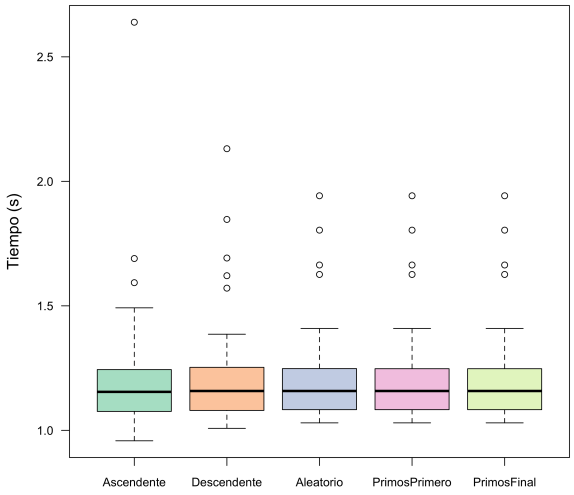
Ahora, se analiza el efecto que tiene, en los tiempos de ejecución, la proporción de números primos que hay en el vector. Para ello, crearemos un vector variando el porcentaje de primos, de 0 % a 100 % en incrementos de 10 %. En la figura 2 tenemos gráficas de caja con los tiempos de ejecución variando la cantidad de núcleos usados. Se nota que entre mayor proporción de primos, mayor es el tiempo de ejecución.

Se realiza, también, una prueba de Kruskal-Wallis para comprobar dicha suposición. En el cuadro 2 se tienen los valores p obtenidos en la prueba, con lo cuál, se concluye que el porcentaje de primos que hay en el vector afecta el tiempo de ejecución de la tarea.

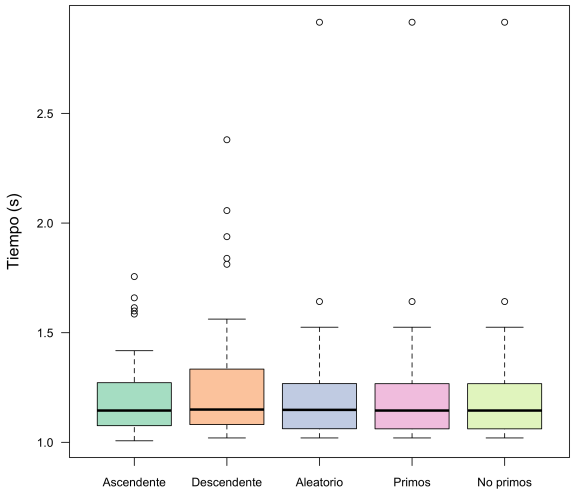
2. Reto 1

El primer reto, es modificar la *tarea* para que encuentre todos los divisores de un número n. Para ello, se utiliza la función dada en el código 2. En la figura 3 tenemos gráficos de caja donde tenemos los tiempos de ejecución de la tarea, variando los núcleos utilizados.

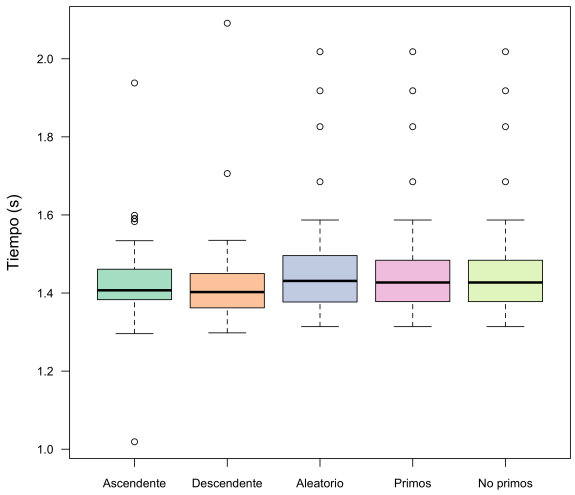
En el cuadro 3 se tienen los valores p obtenidos al realizar una prueba de Kruskal-Wallis a los datos, para verificar si la cantidad de núcleos utilizados o el orden afectan a los tiempos. En este caso, también lo único que afecta es el orden del vector.



(a) Usando tres núcleos.

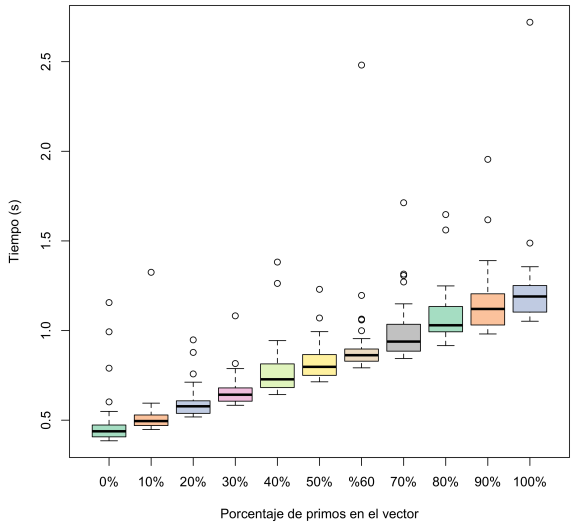


(b) Usando dos núcleos.

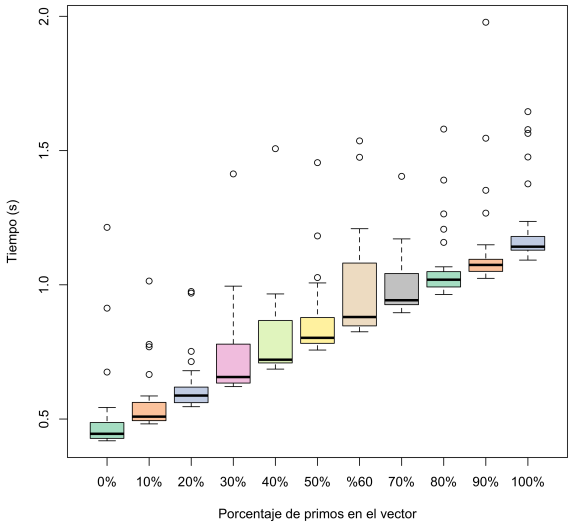


(c) Usando un núcleo.

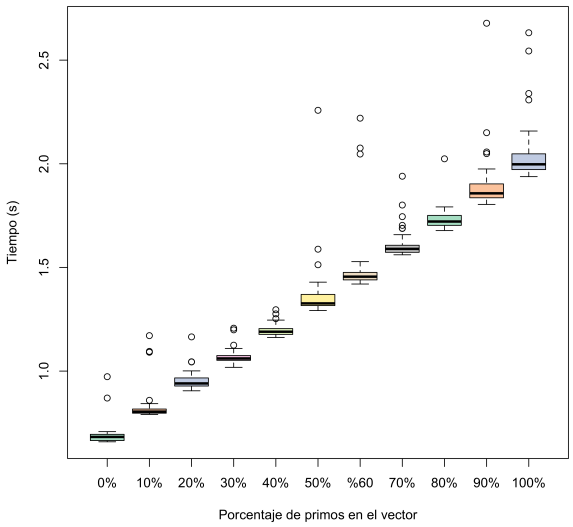
Figura 1: Gráfica de cajas variando los núcleos usados en el experimento.



(a) Usando tres núcleos.



(b) Usando dos núcleos.



(c) Usando un núcleo.

Figura 2: Gráfica de cajas variando los núcleos usados en el experimento.

Cuadro 3: Valores p obtenidos de la prueba de Kruskal-Wallis en el experimento usando la *tarea* de encontrar los divisores.

Datos	valor p
Tiempo $\sim$ Núcleo	0.0018
Tiempo $\sim$ Orden	0.6029

Código 2: Función para encontrar los divisores de un número.

```
divisores<-function(n) {
    div<-numeric()
    for (i in 1:ceiling(sqrt(n))) {
        if ((n%% i) == 0) {
            div<-c(div,i)
            div<-c(div,n/i)
        }
    }
    return(sort(unique(div)))
}
```

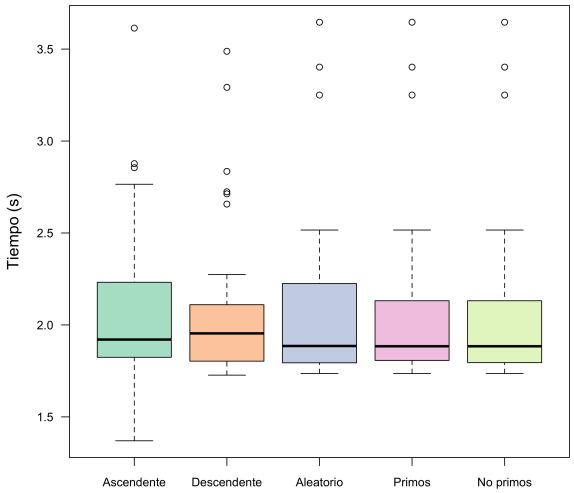
3. Reto 2

En el segundo reto, se modifica la *tarea* para factorizar en primos el número n, en este caso se utiliza la función dada en el código 3. En la figura 4 se tienen los gráficos de caja con los tiempos de ejecución variando los núcleos y en el cuadro 4 se tiene los valores p que se obtienen al realizar una prueba de Kruskal-Wallis. Nuevamente, en este caso lo que afecta es el orden del vector.

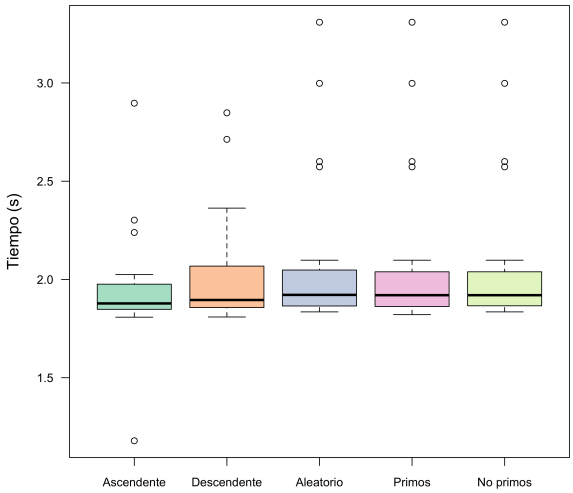
En ninguno de los dos retos, cambiamos las conclusiones obtenidas en el primer experimento.

Código 3: Función para factorizar en primos un número.

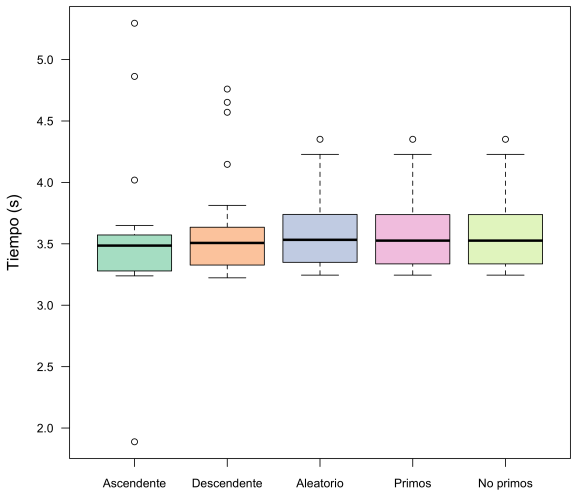
```
fact_primos<-function(n) {
    div<-numeric()
    while(n%% 2 == 0) {
    div<-c(div,2)
    n = n/2
    }
    for(i in seq(3, max(3, ceiling(sqrt(n))), 2)) {
        while(n%% i==0) {
            div<-c(div,i)
            n=n/i
        }
    }
    if(n>2){ div<-c(div,n) }
    return(table(div))
}
```



(a) Usando tres núcleos.



(b) Usando dos núcleos.



(c) Usando un núcleo.

Figura 3: Gráfica de cajas variando los núcleos usados en el experimento.

Cuadro 4: Valores p obtenidos de la prueba de Kruskal-Wallis en el experimento usando la *tarea* de factorizar en primos.

Datos	valor p
Tiempo $\sim$ Núcleo	0.0025
Tiempo $\sim$ Orden	0.5234

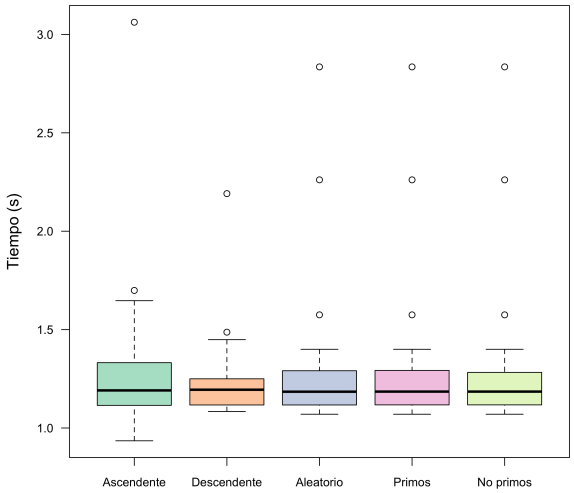
Referencias

[1] Thomas Kluyver, Benjamin Ragan-Kelley, Pérez, et al. Jupyter notebooks—a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas: Proceedings of the 20th International Conference on Electronic Publishing*, page 87. IOS Press, 2016.

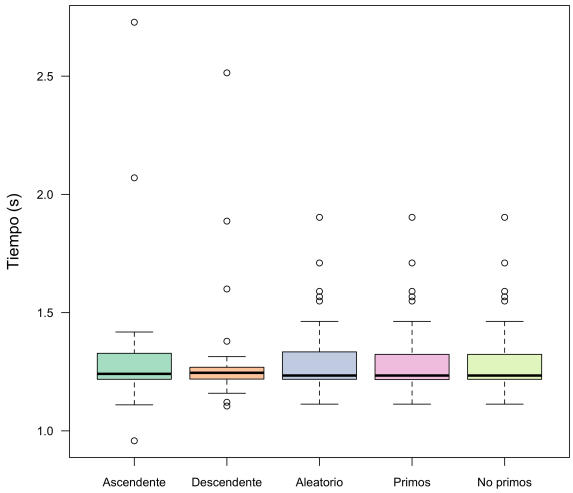
[2] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[3] Elisa Schaeffer. Práctica 3: Teoría de colas. https://elisa.dyndns-web.com/teaching/comp/par/p3.html.

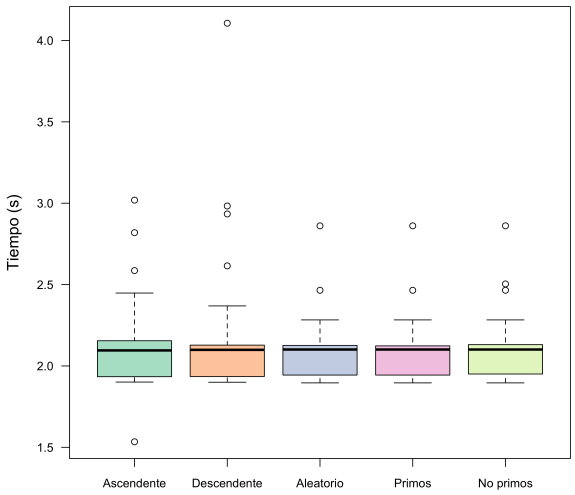
[4] Elisa Schaeffer. Queuing theory. https://github.com/satuelisa/Simulation/blob/master/QueuingTheory/ordering.R.



(a) Usando tres núcleos.



(b) Usando dos núcleos.



(c) Usando un núcleo.

Figura 4: Gráfica de cajas variando los núcleos usados en el experimento.