

# Un algoritmo aleatorizado para calcular la mediana

E. Talla Chumpitaz<sup>1</sup>, C. Aznarán Laos<sup>2</sup>, M. Silva Menejes<sup>2</sup> y J. Jáuregui Alvarado<sup>2</sup>.

<sup>1</sup>Escuela Profesional de Ciencia de la Computación, Universidad Nacional de Ingeniería, Perú.

<sup>2</sup>Escuela Profesional de Matemática, Universidad Nacional de Ingeniería, Perú.

## Objetivos

El objetivo de este trabajo es comparar los diferentes algoritmos de ordenación con respecto al tiempo de ejecución medio que tarda en calcular la mediana de una data.

## Introducción

En el mundo de la estadística existen distintas *medidas de tendencia central*, entre las más comunes se encuentran la *media*, la *mediana* y la *moda*, en este reporte nos enfocaremos en la *mediana*, repasando desde su origen, su importancia y sus aplicaciones prácticas mediante un ejemplo con datos estadísticos.

## Metodología

Para hallar la mediana de un conjunto de datos, se debe de utilizar un algoritmo de *ordenamiento* y de *búsqueda*, ya que no se sabe si el conjunto de datos está ordenado. Para ello se analiza primero los algoritmos de *búsqueda*.

Aplicado a nuestro objetivo sería ineficiente buscar secuencialmente la *mediana*, ya que no necesitamos recorrer toda la cadena para hallarlo, aún peor cuando no está ordenado.

## Conclusiones

- La aleatorización del arreglo ayudó al algoritmo *quickselect* a tener una complejidad  $\mathcal{O}(n)$ .
- En el lenguaje R, el algoritmo más eficiente es *Radix*, pero implementado en el lenguaje C, el algoritmo más eficiente es *Random Selection*.

## Referencias bibliográficas


- [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.
- [2] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.
- [3] C. A. R. Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.
- [4] R. V. Hogg and A. T. Craig. *Introduction to mathematical statistics*. Upper Saddle River, New Jersey: Prentice Hall, 1995.
- [5] D. E. Knuth. *The art of computer programming: sorting and searching*, volume 3. Pearson Education, 1997.
- [6] D. Podgorelec and G. Klajnšek. Acceleration of sweep-line technique by employing smart quicksort. *Information Sciences*, 169(3):383 – 408, 2005. ISSN 0020-0255. doi: 10.1016/j.ins.2004.07.002.

## Agradecimientos

Van a faltar palabras para agradecer a las personas que se han involucrado en la realización de este trabajo, sin embargo, el profesor Cesar Lara Ávila merece un reconocimiento especial ya que gracias a sus consejos y correcciones hoy pudimos culminar este trabajo.

## Información del contacto

- Autor principal: erwinleo\_98@hotmail.com
- Autor secundario: caznarani@uni.pe
- Autor secundario: miller\_silva\_96@hotmail.com
- Autor secundario: jjaureguia@uni.pe
- Profesor investigador: cesarlaraavila@gmail.com

• Disponible en  [github.com/carlosal1015/A-randomized-algorithm-to-calculate-the-median](https://github.com/carlosal1015/A-randomized-algorithm-to-calculate-the-median)

## Resumen

En el ámbito de la estadística y probabilidad existe una controversia sobre cuándo emplear la *mediana* como medida de tendencia central. El objetivo de este artículo es valorar mediante un programa en R su utilidad. Para poder llevar a cabo esta investigación, se han revisado artículos científicos similares y consultado la base de datos de World Bank Open Data, ScienceDirect de donde se extrajo la muestra para la experimentación.

Después de probar diferentes métodos para el cálculo aleatorizado de la mediana, llamamos “random quickselect” a nuestro programa más eficiente.

## Resultados y discusiones

La mediana es un estadístico robusto que incluso aunque los extremos de los datos se vean alterados, la mediana permanece invariable muy útil cuando se trabaja con distribuciones sesgadas.

Cuando se hacen las comparaciones, el programa implementado en el lenguaje C, el *random quickselect* es más rápido que el algoritmo de ordenación *Radix*, pero cuando se implementa en el lenguaje R, *Radix* obtiene la complejidad  $\mathcal{O}(1)$ , es decir, ya no depende del tamaño de datos  $n$ .

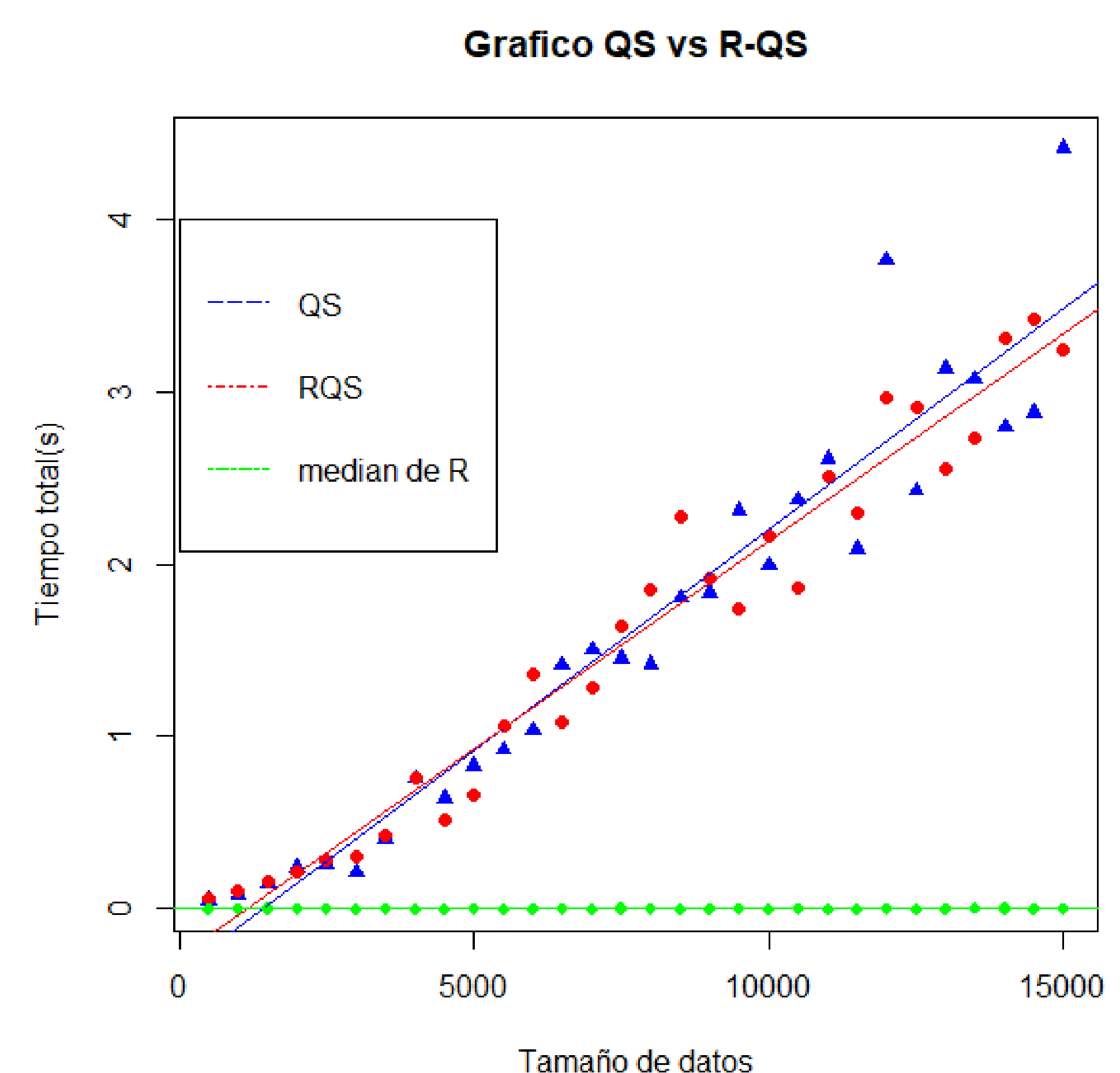


Figura 1: Quickselect vs Random quickselect programado en R.

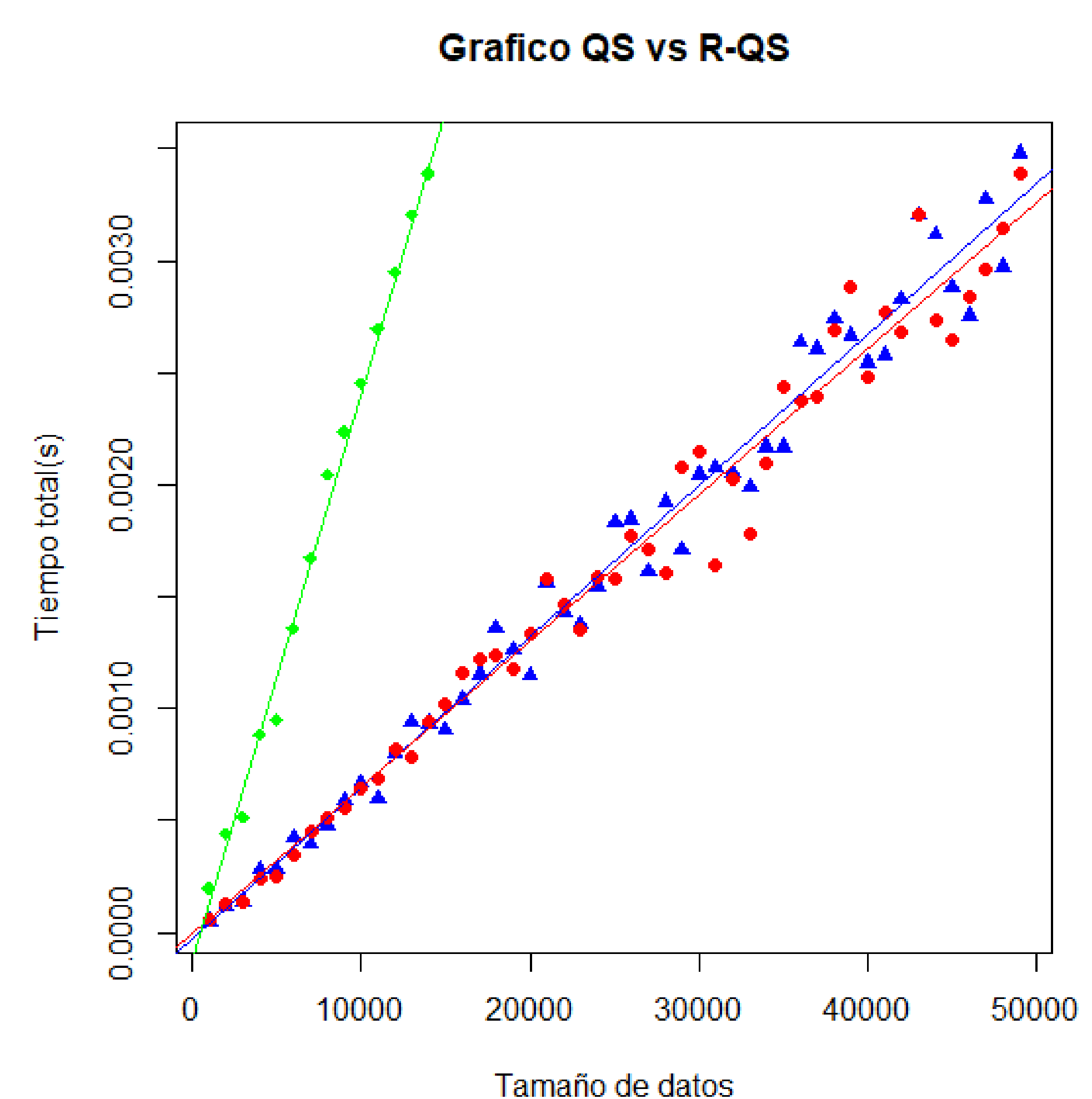


Figura 2: Quickselect vs Random quickselect programado en C.