Estimación del error estándar para la construcción de intervalos de confianza mediante la técnica Boostrap (no paramétrico) a los estadísticos de la media, mediana, varianza y desviación estándar. Un enfoque computacional

Miranda Belmonte Hairo Ulises,[1*]

[1] Centro de Investigación en Matemáticas, A.C.

*E-mail: hairo.miranda@cimat.mx.

Abstract

Resumen

Se realiza la estimación del error estándar para la construcción de intervalos de confianza mediante la técnica "Bootstrap" para los estadísticos; media, mediana, varianza y desviación estándar. La estimación se realiza mediante simulación en C y exportación a R, haciendo uso de la librería "Rcpp". Se contrastan resultados en tiempo de estimación respecto a la librería "boot", la cual puede ser implementada en R. El trabajo muestra que se es más eficiente implementar el código en C que utilizar la librería "boot" en R, permitiendo realizar mayor número de simulaciones en menor tiempo.

Introducción

Tener una medida de precisión en la estimación de un estadístico sirve de referencia sobre la variabilidad de la estimación. Esta medida de variación se puede representar mediante desviaciones estándar. No obstante, obtenerlo no siempre se hace de una forma sencilla. Una vez estimado, se podría hacer la construcción de un intervalo de confianza para el estadístico deseado.

La técnica de remuestreo Bootstrap, desarrollada por Bradley Efron en 1960, pretende abarcar la problemática de la estimación del error estándar, y a su vez, permite la construcción de diversos intervalos de confianza. Asimismo, esta técnica es una generalización de la técnica de remuestreo Jackknife.

Para realizar una estimación consistente, mediante la técnica bootstrap, en necesario un tamaño de muestra relativamente grande. Dicha estimación se realiza por medio de simulación, la cual al tener un tamaño de muestra grande, una simulación de un tamaño considerable hará que bajo ciertas circunstancias el estadístico del error estándar converja en probabilidad al parámetro muestral.

La técnica Bootsrap se encuentra implementada en diversas programas estadísticos. Entre estos programas R se ha convertido el referente principal para el análisis estadístico. El problema de R es que no está pensado como lenguaje de programación. Al ser el bootstrap una técnica computacional, se puede pensar en la problemática de realizar simulaciones exhaustivas en R.

Al no estar pensado R como lenguaje de programación, se ha abordado este problema con la creación de librerías compartidas que ayudan a realizar la ejecución desde otro lenguaje de programación. Rccp es una paquetería en R que ayuda a conectar C++ con R. Asimismo, realiza compilación de funciones escritas en C++ ó C, las cuales se desempeñan mejor que

escribir código en R.

En este documento se pretende demostrar como R presenta problemas cuando se es pensado como lenguaje de programación. De está manera, se escribe la técnica de boostrap en C y se compila en R mediante la paquetería Rcpp con la finalidad de obtener mejores resultados con un número mayor de simulaciones, siempre y cuando el tamaño de muestra sea grande.

Para demostrar la eficiencia entre C y R, se contrasta contra la función *"boot"*, la cual se encuentra escrita en R .Se toman datos pre-cargados del *dataset* que proporciona R con el fin de evaluar la función *boot* y la escrita en C. Una vez lo mencionado, se estima el error estándar de la media, mediana, varianza y desviación estándar. La otra parte del trabajo, una vez obtenido los errores estándar estimados, consiste en construir distinto intervalos de confianza.

El presente trabajo se estructura de la siguiente manera; en la sección 2 se habla sobre la técnica de remuestreo bootstrap. La sección 3, introduce a la librería Rcpp y su conexión con C. En la sección 4, se presentan las funciones utilizadas en R y las escritas en C para la técnica boostrap. La sección número 5, se da una breve descripción a los datos que se utilizaron. la sección 6, realiza las pruebas de eficiencia en tiempo, entre la función de la técnica de remuestreo bootsrap escrita en C, respecto a la función *boot* de R. La sección 7, se presentan los resultados de la estimación del error y la construcción de intervalos de confianza del boostrap para los estadísticos media, mediana, varianza, y desviación . Por último, se realiza una breve conclusión sobre los principales resultados de escribir código en C y compilarlo en R.

Sección 2. Técnica de remuestreo Bootstrap[1]

La técnica Bootstrap es utilizado para la estimación del error estándar y construcción de intervalos de confianza. De esta forma, se tiene $T_n(X) = f(X_1, ..., X_n)$ un estadístico en función de

[1] Sección basada en Wasserman(2005) (2)

la muestra, el cual se desea estimar $V(T_n)$ su variaza , cuya distribución suele no ser conocida, y por ende, en algunos casos, difícil de estimar.

Para obtener la varianza del estadístico es necesario conocer su distribución . No obstante, al no conocerla, se puede aproximar mediante el calculo de la función de distribución estimada. La forma en que se aproxima es sencilla, se ordena la muestra $(x_1, ..., x_n)$ y se ponderando por $1/n$, realizando la suma (acumulada) y obteniendo la función de distribución estimada. Realizar esto es igual a generar un remuestreos con remplazo de la muestra. Dado que tomar una observación de una muestra de tamaño en n tiene como probabilidad $1/n$, y si se toma otra observación (una vez remplazada), la probabilidad seguirá siendo la misma (i.e. uniforme).

El algoritmo por el cual se estima está basado en el que presenta Wasserman(2005) (2) en su libro de texto.

1) Tomar una muestra de tamaño n de $(x_1, ..., x_n)$.

2) Calcular el estadístico con el remuestreo del paso 1 $T_n^* = (X_1^*, ..., X_n^*)$.

3) Repetir el paso 1 y 2 , B veces, y obtener B estadísticos $T_{n,B}^*$.

4) Calcular la varianza de la siguiente forma:

$$v_{bootstrap} = \frac{1}{B}\sum_{b=1}^{B}(T_{n,b}^* - \frac{1}{B}\sum_{r=1}^{B} T_{n,r}^*)$$

Una vez estimada la varianza del estadístico la construcción de intervalo de confianza es el siguiente:

Intervalo Normal:

$$\hat{se}_{bootstrap} = \sqrt{v_{bootstrap}} \tag{1}$$

$$T_n \pm z_{\alpha/2}\hat{se}_{bootstrap} \tag{2}$$

Dicho intervalo presenta la limitante de que si la muestra no proviene de una población con distribución normal, entonces no se le considera un buen intervalo. Es por eso, que se presentar dos alternativas distintas al ya mencionado.

Intervalo Pivote:

$$C_n = (2\hat{\theta} - \hat{\theta}^*_{1-\alpha/2}, 2\hat{\theta} - \hat{\theta}^*_{\alpha/2}) \tag{3}$$

$$\hat{\theta} = T_n$$

$$\hat{\theta}^* = T^*_n$$

Intervalo Percentil:

$$C_n = (2\hat{\theta}_{1-\alpha/2}, 2\hat{\theta}_{\alpha/2}) \tag{4}$$

Donde $\hat{\theta}_{1-\alpha/2}$ son los cuantiles de la $n - esima$ replica del estadístico con el remuestreo al α nivel de significancia.

El Boostrap, de esta manera, se presenta como una generalización de la técnica Jackknife. Ambas técnicas realizan un remuestreo sobre la muestra realizando remplazo(generan muestras más grandes). La técnica Bootstrap, sin embargo, suele tener condiciones menos restrictivas que el Jackknife, estimando cosas no tan suaves como cuantiles. Una de sus funcionalidades es el estimar la función de distribución acumulada. No obstante, entre sus limitantes, se presenta que a tamaño de muestra (n) pequeños, el estimador no es consintiente (i.e, no converge en probabilidad), inclusive si se tiene un tamaño de simulación muy grade. Es importante mencionar que tener una muestra pequeña y una simulación considerablemente grande, lo único que sucedería es que la simulación estaría aproximando al estimador, y no al parámetro deseado.

Sección 3. Rcpp y C

De acuerdo con Wickham (2015) (3) programar en R no es lo más eficiente. De esta forma, escribir el código en algún otro lenguaje para utilizarlo en R puede ser más eficiente. Una de las ventajas de realizar el código en R, es que su interprete y su mecanismo de extensión se encuentran implementados en C.

Eddelbullet, D.(2013) (1) recalca que la ventaja de programar en C, es por la fácil comunicación entre otros programas por medio de librerías compartidas. Es aquí donde R, por medio de la paquetería Rcpp, puede sacar provecho a lo ya mencionado. La contribución de Rcpp es de Dirk Eddelbullet y Ramain Francois, escribiendo la paqueteria que permite conectar R con C++, o inclusive de C++ a R, con la finalidad de de ejecutar el código con mayor precisión y eficiencia[2].

La librería Rcpp, realiza el interfaz entre C++ o C y R por medio de una compilación compartida (Eddelbullet, 2013) (1). De esta forma, existen dos maneras básicas de realizar la compilación en R; la primera, de forma ***inline*** cuya compilación se hace directamente al escribir el código en la sesión de R, por medio de la función *"cppFunction"*. La segunda, es de la forma ***outline***[3]. Esta versión incluye la función *cppFunction* y *sourceCpp*, la cual permite compilar de manera directa en un compilador de C o C++ y exportar lo realizado a R mediante la función *sourceCpp*, es importante recalcar que se es necesario programar en C o C++ con algunas expresiones que Rcpp requiere para su interpretación .

Existe una tercer manera de realizar la conexión, el cual consiste en programar directamente desde C, debido a que muchas funciones básicas de R se encuentran programadas en C. El escribir código directo de interfaz puede ser algo complicado dado a que la lectura de dicho código puede ser difícil.

En este documento se hace uso de la la paqueteria Rcpp (Versión 0.12.16), y de la función *soruceCpp*, permitiendo trabajar directo desde C++ o C, e implementar las funciones elaboradas en R. Lo cual facilita escribir el código sobre un compilador ya conocido en C++ o C , y poder percatarse de los errores a la hora de hacer la compilación.

[2]Para saber más cobre el tema vea: http://dirk.eddelbuettel.com/code/rcpp.html

[3]aproximación se encuentra disponible con la versión 0.10.0 de la paquetería Rcpp

Sección 4. Funciones a Utilizar

En esta parte del documento hablaremos sobre las funciones que se han sido implementadas en este trabajo. Primero, comenzaremos describiendo paqueterías y librerías utilizadas en R. Enseguida de eso, se presenta con detalle las funciones que se realizan en C y se exportan a R para su compilación.

Si uno quisiera estimar el error estándar y construir intervalos de confianza por medio de bootstrap, puede encontrar variedad de paqueterías en R que lo hacen. Sin embargo, entre la variedad que se presenta buscamos alguna que sólo realice lo más básico, i.e., estimación del error estándar del estadístico y construcción de distintos intervalos de confianza, y sobre todo, que se encuentre escrita en R. Esto último con el fin de comprobar que programar en R no es lo más eficiente. Es por eso, que de entre las que se encontraron se seleccionó la paquetería *"boot"*, cuya función con el mismo nombre realiza lo ya mencionado.

A continuación se presenta a manera general las funciones utilizadas de R (sólo los inputs necesarios para la estimación).

Remuestreo Bootstrap(**boot** [4])

Descripción : Genera n replicas bootstrap de algún estadístico. Se puede realizar un bootstrap parámetrico ó no parámetrico. **Está función está realizada en R.**

Uso[5] **:** boot(datos, estadístico, Número de simulaciones, Otros)

[4]Ver: https://cran.r-project.org/web/packages/boot/boot.pdf

[5]Para más inputs vea el apartado usage de : https://cran.r-project.org/web/packages/boot/boot.pdf

Intervalos de confianza no parámetricos Bootstrap (**boot.ci** [6])

Descripción : Genera 5 tipos diferentes de intervalos de confianza. Normal, el básico bootstap, studentizado, percentil ajustado. **Está función está realizada en R.**

Uso[7] **:** boot.ci(Salidos de boot, Nivel de confianza = 0.95, tipo de IC = "all") Ahora, se presentan las funciones realizadas en C.

Media (**media**)

Descripción : Estima la media muestral.

Uso : mean(NumericVector)

x Vector de tamaño n

Tiempo medido con Microbechmark

	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
media C	nanoseconds	854	1280	1476.95	1281	1281	20907	100
mean R	nanoseconds	4267	4694	5043.89	4694	4694	34134	100

Mediana (**mediana**)

[6]Ver: https://cran.r-project.org/web/packages/boot/boot.pdf

[7]Para más inputs vea página 18 de : https://cran.r-project.org/web/packages/boot/boot.pdf

Descripción : Estima la mediana muestral.

Uso : mean(NumericVector)

x Vector de tamaño n

Nota : La eficiencia en el calculo de la mediana depende del algoritmo de ordenamiento por el cual se realice. En una primera instancia se trabaja con un Bubblesor, no obstante, en el mejor casos (datos no ordenados) su complejidad es de orden $\Omega(n)$, y en el peor caso (datos ordenados) $O(n_2)$ cuadrático. Es por eso que el ordenamiento se realiza con un Quick sort, que en el mejor de los casos su ordenamiento es $\Omega(nlog(n))$, pero en el peor de los caso será del mismor orden que el Bubble sort. Por lo regular, en la práctica, los datos no suelen encontrarse ordenado, a menos que sean estadísticos de orden.

Tiempo medido con Microbechmark

	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
media C	microseconds	17.0	17.9	19.8	18.77	19.6	84.4	100
mean R	microseconds	26.0	26.8	31.7	27.7	29.4	153.1	100

Como se puede observar, en el cuadro de arriba, utilizando un Quicksort es suficiente para mejorar la mediana que calcula R. Esto tiene relevancia a la hora de realizar el bootstrap para el estadístico de la mediana, el cual nos ahorrará tiempo en la simulación. En el apéndice del documento se presenta un cuadro sobre la complejidad y orden de los distinto tipos de ordenamiento, en el cual se observa que si se utiliza un ordenamiento tipo Radix, mejoraría bastante la simulación[8].

[8]La estimación con ordenamientos de orden lineal se presenta entre los trabajos posteriores a realizar

Varianza (**varianza**)

Descripción : Técnica Bootstrap basado en:

Descripción : Estima la varianza muestral.

Uso : varianza(NumericVector)

x Vector de tamaño n

Tiempo medido con Microbechmark

	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
media C	microseconds	1.2	1.7	2.1	1.7	2.1	15.7	100
mean R	microseconds	15.3	16.4	18.0	17.0	17.4	110.9	100

Desviación Estándar (**sd**)

Descripción : Estima la desviación estándar muestral.

Uso : sd(NumericVector)

x Vector de tamaño n

Tiempo medido con Microbechmark

	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
media C	microseconds	2.1	2.5	3.7	3.4	3.8	32.8	100
mean R	microseconds	43.9 4	5.2	54.8	46.5	50.1	261.5	100

Quick sort (**quicksort**)

Descripción : Ordenamiento de datos de menor a mayor..

Uso : quicksort(NumericVector, int l, int r)

x	Vector de tamaño n
l	Inicia en cero
r	Tamaño de muestra menos una observación

Mejor caso: $\Omega(nlog(n))$

Promedio: $\theta(nlog(n))$

Peor caso : $O(n_2)$

Tiempo medido con Microbechmark

	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
media C	microseconds	15.7	16.6	17.5	17.0	17.4	56.3	1000
mean R	microseconds	11.5	12.3	14.2	12.8	13.2	206.0	1000

A pesar de que en el calculo de la mediana marca diferencia, el quick sort no le gana (en términos de tiempo) a la función *"order"* en R. Sólo por dato curioso se presenta la tabla de eficiencia en tiempo medido con microbenchmark para el peor caso (datos ya ordenados).

Bootstrap Estadístico: Media (**bootstrapMedia**)

Referencia : Algoritmo basado en Wasserman, Larry. All of statisitcs: A concise course in statistical inference. Ed. Springer, pp. 109, 2005.

Descripción : Realiza la estimación del error estándar por la técnica botstrap

Uso : bootstrapMedia(NumericVector x1 ,NumericVector xx1, B)

Salida : Lista que contiene el error estimado del estadístico, y el $n - esimo$ estadístico bootstrap.

x1	Vector de la muestra (datos).
B	Número de simulaciones que realizará el Bootstrap para estimar el error estándar.

Nota : El $n - esimo$ estadístico bootstrap es de importancia para la construcción de los intervalos de confianza percentil y pivote.

Bootstrap Estadístico: Mediana (**bootstrapMediana**)

Referencia : Algoritmo basado en Wasserman, Larry. All of statisitcs: A concise course in statistical inference. Ed. Springer, pp. 109, 2005.

Descripción : Realiza la estimación del error estándar por la técnica botstrap

Uso : bootstrapMediana(NumericVector x1 ,NumericVector xx1, B)

Salida : Lista que contiene el error estimado del estadístico, y el $n - esimo$ estadístico bootstrap.

x1 Vector de la muestra (datos).

B Número de simulaciones que realizará el Bootstrap para estimar el error estándar.

Nota : El $n - esimo$ estadístico bootstrap es de importancia para la construcción de los intervalos de confianza percentil y pivote.

Bootstrap Estadístico: Varianza (**bootstrapVarianza**)

Referencia : Algoritmo basado en Wasserman, Larry. All of statisitcs: A concise course in statistical inference. Ed. Springer, pp. 109, 2005.

Descripción : Realiza la estimación del error estándar por la técnica botstrap

Uso : bootstrapVarianza(NumericVector x1 ,NumericVector xx1, B)

Salida : Lista que contiene el error estimado del estadístico, y el $n - esimo$ estadístico bootstrap.

x1 Vector de la muestra (datos).

B Número de simulaciones que realizará el Bootstrap para estimar el error estándar.

Nota : El $n - esimo$ estadístico bootstrap es de importancia para la construcción de los intervalos de confianza percentil y pivote.

Bootstrap Estadístico: Desviación Estándar (**bootstrapSD**)

Referencia : Algoritmo basado en Wasserman, Larry. All of statisitcs: A concise course in statistical inference. Ed. Springer, pp. 109, 2005.

Descripción : Realiza la estimación del error estándar por la técnica botstrap

Uso : bootstrapSD(NumericVector x1 ,NumericVector xx1, B)

Salida : Lista que contiene el error estimado del estadístico, y el $n - esimo$ estadístico bootstrap.

x1	Vector de la muestra (datos).
B	Número de simulaciones que realizará el Bootstrap para estimar el error estándar.

Nota : El $n - esimo$ estadístico bootstrap es de importancia para la construcción de los intervalos de confianza percentil y pivote.

Sección 5. Descripción a los datos

Los datos que se utilizan se encuentran pre-cargados en R. El conjunto de datos iris son bien conocidos por los usuarios que se encuentran familiarizándose con el programa. La información fue aportada Fisher y Anderson, los cuales tomaron una muestra de la longitud del sépal y ancho del pétalo (centímetros) de 50 flores de 3 especies distintas. Estas especies son Iris setosa, versicolor, and virginica.

Para propósitos del documento, se toma una serie solamente del conjunto de datos. Se selecciona la longitud de sépal. A continuación se muestra a manera general los descriptivos de la muestra.

Tiempo medido con Microbechmark

Muestra	Min	Max	Media	Mediana
150	4.3	5.843	5.8	7.9

Para construir intervalos de confianza normales adecuados, es necesario que la distribución

de la muestra se aproxime a la distribución normal. Con la finalidad de tener argumentos para la construcción de estos intervalos, se realiza pruebas para probar normalidad en la distribución de los datos. Primero, se gráfica un QQ-plot[9] con un intervalo de confianza al 95

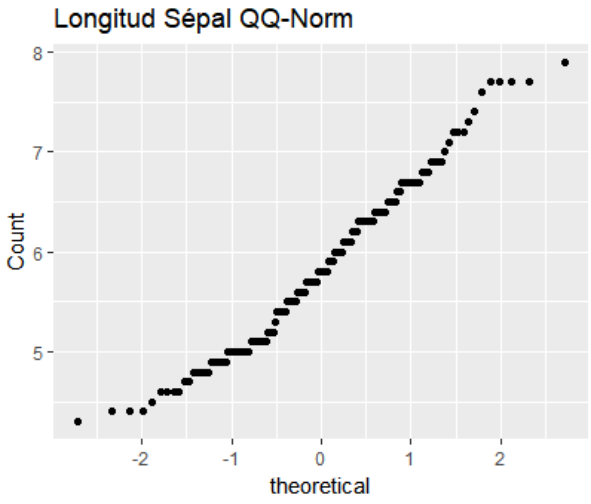

Figure 1: QQ-plot Longitud Sépal

Como se observa en la figura 1, la distribución empírica de los cuantiles de la muestra se aproxima a la distribución normal teórica. Sin embargo, se puede observar colas ligeramente pesadas, y algo de sesgo, cargando a la distribución empírica a la derecha. En la figura 2, se muestra el histograma de los datos (longitud del sépal en centímetros), que al igual que en el QQ-plot, se puede observar lo ya mencionado sobre la distribución muestral.

[9]Se utiliza la librería *"fBasic"*.

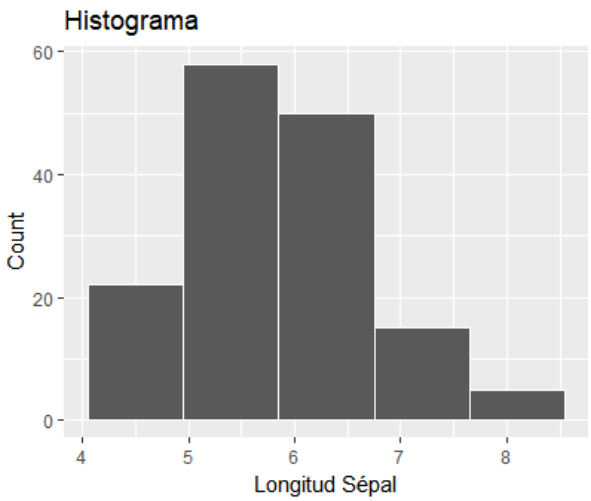

Figure 2: Histograma Longitud Sépal

Por último, se realiza las pruebas Jarquer-Bera[10] y Shapiro-Wilk[11] , ambas bajo la hipótesis nula de normalidad, en este caso, si es que la distribución de la muestra, bajó la hipótesis nula, se distribuye como una población normal.

	Estadístico	P-valor
W	0.9760903	0.010
χ^2	4.4859	0.106

Table 1: Prueba de Normalidad al 95% de significancia

Como se observa en la tabla 1, en el caso de la prueba de Shapiro-Wilk, se tiene un estadístico cercano a uno y un p-valor de 0.01, y recordando que un valor del estadístico cercano a uno nos permite concluir, en el caso del estadístico W, que se tiene la evidencia suficiente para no rechazar la hipótesis nula de normalidad (Wasserman, 2005) (2) . El p-valor y el estadístico de la prueba Jarquer-Bera, al igual que Shaphito-Wilk, apoya a favor sobre la normalidad en la

[10]Se utiliza la librería *"fBasic"*.
[11]Se utiliza la librería *"fBasic"*.

distribución de los datos de la longitud del sépal.

Una vez realizado lo anterior, asumiremos supuesto de normalidad en la muestra. Dicho supuesto tendrá relevancia a la hora de realizar la construcción de los intervalos de confianza Normales.

Sección 6. Eficiencia en Tiempos

En está sección, se toma la muestra de la longitud del sépal y se mide la eficiencia en tiempo de la estimación del error estándar por la técnica bootstrap - el cual es programado en C y exportado a R- para los estadísticos de la media, mediana, varianza y desviación estándar. La comparación se realiza respecto función *"boot"*, la cual se encuentra escrita en R, y cuya librería cuenta con el mismo nombre. El tiempo entre expresiones evaluado las n veces se cuantifica mediante el uso de "microbenchmark"[12].

Microbenchmark trata de evaluar con precisión el tiempo que le toma evaluar una expresión n número de veces (100 de manera automática). De esta forma, la función microbenchmarck, compara operaciones que por lo regular la diferencia entre ellas es mínima. El tiempo que mide puede ser expresado en microsegundos o nanosegundos. La función, a su vez, tiene como salida los siguientes elementos: Unidad de medición, mínimo, cuantil inferior, cuantil superior, mediana y media. Estos últimos en términos de evaluar una expresión en las n veces seleccionadas. Es importante saber que todas las expresiones evaluadas están hechas en C.

A continuación, se presentan cuatro tablas, en las cuales se observan la eficiencia (medida por la función Microbenchmark) del bootstrap realizado en C, y la función *"boot"* de la librería en R. Se recuerda al lector que se realizan cuatro funciones de bootstrap, cuyo fin es estimar el error estándar de la media, mediana, varianza y desviación estándar muestral. Asimismo, se

[12]Para más información ver: https://cran.r-project.org/web/packages/microbenchmark/microbenchmark.pdf

realiza la evaluación para un distinto número de simulaciones[13].

En la tabla 2, se presenta el tiempo(máquina) que le toma evaluar un cierto número de veces y con un número distinto de simulaciones, la función bootstrap para la estimación del error estándar de la media programada en C, respecto a la función *boot* que implementa y se escribe en R[14]. En las tablas 3, 4 y 5, se presentan a manera similar, los resultados para la estimación del error estándar de la mediana, varianza y desviación estándar.

En el caso del bootstrap para la estimación del error estándar de la media, se observa, que la función realizada en C es a lo menos, 11 veces más rápida que la que realiza la librería boot. Al realizar la evaluación en 10^6, a pesar que se realiza solamente una evaluación, se puede observar que la diferencia en segundos sigue siendo aproximadamente la misma que con un número menor de simulaciones.

Sim = 10^4	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	milliseconds	28.3	29.0	29.8	29.6	30.2	37.3	100
Bootstrap R	milliseconds	301.0	332.9	345.4	340.9	350.2	562.7	100

Sim = 10^5	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	milliseconds	287.1	291.2	298.7	293.4	298.8	460.6	100
Bootstrap R	milliseconds	3177.2	3353.0	3503.5	3480.0	3588.4	4259.9	100

[13] Simulaciones se encuentran realizadas en Rstudio versión 1.1456. El código en C se implementa en el compilador Dev C++ versión 5.11. El computador en el cual se realiza lo anterior tiene las siguiente descripción: Procesador: Intel(R) Core(TM) i5-6200U CPU. Memoria RAM: 4.00GB, Nucleos 2

[14] Es importante observar que la simulación, para todos los casos, de 10^6, se le indica a la función microbenchmark que sólo la evalué una vez, esto por limitaciones del computador en la que se realiza esté documento.

Sim = 10^6	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	seconds	3.1	3.1	3.1	3.1	3.1	3.1	1
Bootstrap R	seconds	40.4	40.4	40.4	40.4	40.4	40.4	1

Table 2: Estimación por Bootstrap del Error Estándar de la Media

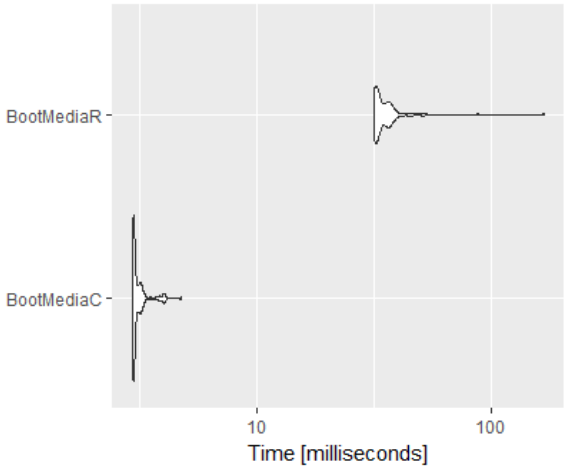

Figure 3: Micrbenchmark Plot: Simulación 10^3

Para tener una perspectiva más didáctica, realizamos un *"autoplot"*[15]. El cual permite realizar una mejor interpretación. Como se puede observar en la figura 3, la estimación del error estándar de la media por la función realizada en C (BootMediaC), es unos mili segundos más rápido que R (BootMediaR). En ambos casos, el cluster que se forma al inicio nos dice que ambas funciones suelen terminar de manera rápida.

En el caso del bootstrap para la estimación del error estándar de la mediana, solamente la función escrita en C es cuatro veces más rápida (milli segundos) en promedio. Al realizar 10^6 simulaciones, en promedio se tiene la misma diferencia pero en términos de segundo. De esta manera, podemos notar que al aumentar el número de simulaciones, el tiempo incrementa en

[15]Función de la paquetería *"ggplot2"*.

un factor constante que solamente cambia las unidades.

La estimación del error estándar para el estadístico de la mediana, como se mencionó a la hora de describir la función de la mediana (vea sección Funciones a Utilizar), va a depender del algoritmo de ordenamiento que se utilice. En este caso, el Quick sort bastó para mejorar la simulación que realiza la técnica bootstrap.

Sim = 10^4	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	milliseconds	138.6	149.5	159.9	152.0	154.5	345.9	100
Bootstrap R	milliseconds	565.0	606.2	646.0	625.5	652.5	1131.6	100

Sim = 10^5	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	seconds	1.4	1.5	1.6	1.5	1.6	2.1	100
Bootstrap R	seconds	5.7	6.1	6.6	6.3	6.9	9.2	100

Sim = 10^6	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	seconds	15.4	15.4	15.4	15.4	15.4	15.4	1
Bootstrap R	seconds	70.2	70.2	70.2	70.2	70.2	70.2	1

Table 3: Estimación por Bootstrap del Error Estándar de la Mediana

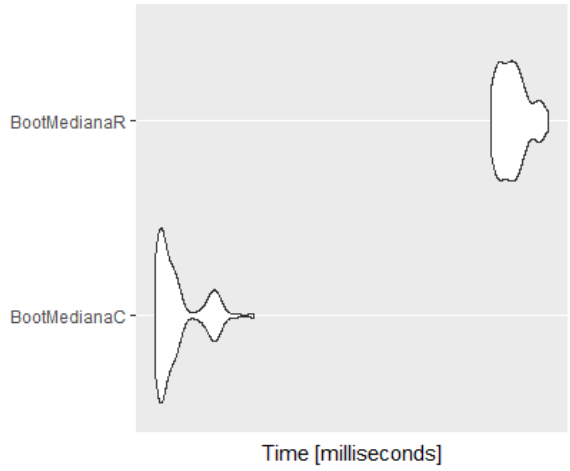

Figure 4: Micrbenchmark Plot: Simulación 10^3

La figura 4, presenta cosas interesantes en el tiempo de evaluación de las funciones . La función escrita en R, por lo general, suele ser lenta para terminar, lo cual se ve representado por lo ancho en periodos medios. Por otra parte, la función escrita en C, suele tener tiempos mínimos con más frecuencia, con valores medios muy cercanos a su media, y pocos valores máximos. El bootstrap para la estimación del error estándar de la varianza, es más rápido si se realiza el código en C que en R, aproximadamente 15 veces más.

Sim = 10^4	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	milliseconds	31.4	32.2	32.6	32.4	32.6	41.3	100
Bootstrap R	milliseconds	452.1	471.3	489.6	481.7	499.4	647.2	100

Sim = 10^5	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	milliseconds	316.0	322.8	343.2	327.2	347.8	455.3	100
Bootstrap R	milliseconds	4569.6	4779.1	5036.5	4860.9	5130.5	7431.0	100

Sim = 10^6	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	seconds	3.3	3.3	3.3	3.3	3.3	3.3	1
Bootstrap R	seconds	57.8	57.8	57.8	57.81	57.8	57.8	1

Table 4: Estimación por Bootstrap del Error Estándar de la Varianza

Para tener una perspectiva más didáctica, realizamos un *"autoplot"*[16]. El cual permite realizar una mejor interpretación.

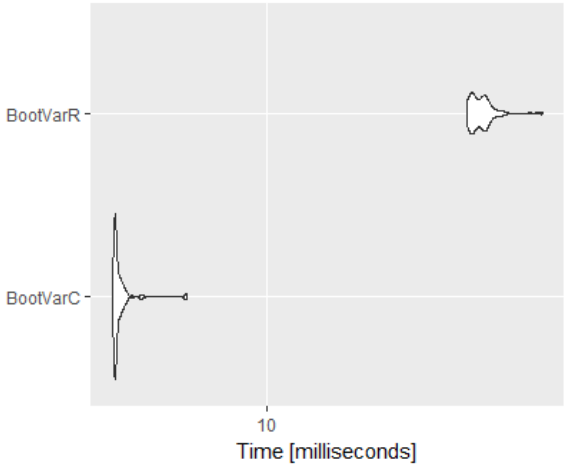

Figure 5: Micrbenchmark Plot: Simulación 10^3

Lo que se puede destacar de la función en C, es que realiza de forma muy rápida la expresión. En la figura 5, se puede observar lo anterior en lo extenso de la cola del gráfico del BootVarC. Al realizar la estimación de error estándar para el estadístico de la desviación estándar, sucede algo interesante. En el texto de Wickham(2015) (3) realiza un ejemplo de como los tiempos en evaluación al realizar una raíz cuadrada, medidos por la función microbenchmark, puede ser distintos. De este modo, otra de las razones del porque la función es 20 veces más rápida -en promedio- que la función escrita en R, se debe a la función *"sqrt "*.

[16]Función de la paquetería *"ggplot2"*.

Sim = 10^4	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	milliseconds	30.6	32.1	37.2	34.1	36.1	159.6	100
Bootstrap R	milliseconds	631.5	694.2	846.4	793.6	835.8	7888.0	100

Sim = 10^5	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	milliseconds	306.9	313.0	321.1	316.5	321.7	443.8	100
Bootstrap R	milliseconds	6145.3	6375.2	6606.4	6492.6	6718.1	8076.8	100

Sim = 10^6	Unidad	Min	Low Q	Media	Mediana	Upper Q	Max	Evaluación
Bootstrap C	seconds	3.2	3.2	3.2	3.2	3.2	3.2	1
Bootstrap R	seconds	73.2	73.2	73.2	73.2	73.2	73.2	1

Table 5: Estimación por Bootstrap del Error Estándar de la Desviación Estándar

Para tener una perspectiva más didáctica, realizamos un *"autoplot"*[17]. El cual permite realizar una mejor interpretación.

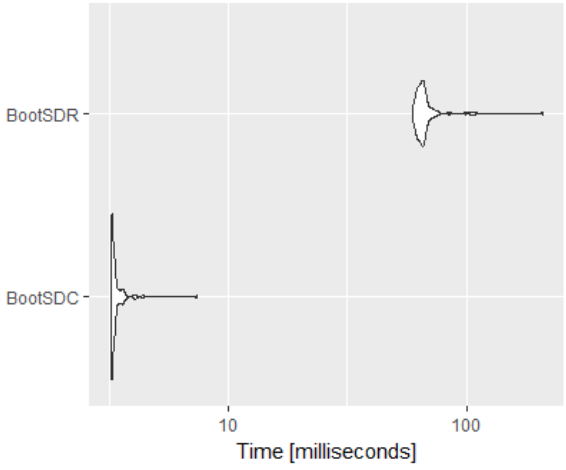

Figure 6: Micrbenchmark Plot: Simulación 10^3

[17]Función de la paquetería *"ggplot2"*.

Sección 7. Aplicación del la técnica Bootstrap

Los resultados de la estimación del error estándar de los estadísticos de la media, mediana, varianza, y desviación estándar, para los datos de la longitud sel Sépal, se contrastan respecto a la función *"boot"*, la cual se habló de ella en secciones anteriores. A su vez, se compara la construcción de algunos intervalos de confianza que dicha función da como salida.

Los siguientes resultados se realizan con 10^5 simulaciones, siendo una número considerable y apropiado dado el tamaño de la muestra($n = 150$). Ya que bajo estas condiciones, la técnica Bootsrap realiza estimaciones del error estándar consistentes, y con ello mejores intervalos de confianza.

Media

	Error Estándar	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.067	5.711	5.975
Bootstrap R	0.067	5.711	5.976

Table 6: Estimación Bootstrap del Error Estándar de la Media e Intervalo de confianza Normal al 95$\%$ de significancia

	Intervalo Inferior	Intervalo Superior
Bootstrap C	5.710	5.976
Bootstrap R	5.711	5.975

Table 7: Intervalos de confianza Pivote de la Media al 95% por la técnica Bootstrap

	Intervalo Inferior	Intervalo Superior
Bootstrap C	5.711	5.976
Bootstrap R	5.712	5.975

Table 8: Intervalos de confianza Percentil de la Media al 95% por la técnica Bootstrap

Mediana

	Error Estándar	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.101	5.600	5.999
Bootstrap R	0.101	5.612	6.011

Table 9: Estimación Bootstrap del Error Estándar de la Mediana e Intervalo de confianza Normal al 95% de significancia

	Intervalo Inferior	Intervalo Superior
Bootstrap C	5.6	6.0
Bootstrap R	5.6	6.0

Table 10: Intervalos de confianza Pivote de la Mediana al 95% por la técnica Bootstrap

	Intervalo Inferior	Intervalo Superior
Bootstrap C	5.6	6.0
Bootstrap R	5.6	6.0

Table 11: Intervalos de confianza Percentil de la Mediana al 95% por la técnica Bootstrap

Varianza

	Error Estándar	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.066	0.554	0.816
Bootstrap R	0.066	0.559	0.821

Table 12: Estimación Bootstrap del Error Estándar de la Varianza e Intervalo de confianza Normal al 95% de significancia

	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.554	0.815
Bootstrap R	0.554	0.817

Table 13: Intervalos de confianza Pivote de la Varianza al 95% por la técnica Bootstrap

	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.555	0.816
Bootstrap R	0.554	0.816

Table 14: Intervalos de confianza Percentil de la Varianza al 95% por la técnica Bootstrap

Desviación Estándar

	Error Estándar	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.040	0.749	0.907
Bootstrap R	0.040	0.752	0.911

Table 15: Estimación Bootstrap del Error Estándar de la Varianza e Intervalo de confianza Normal al 95% de significancia

	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.752	0.911
Bootstrap R	0.752	0.911

Table 16: Intervalos de confianza Pivote de la Varianza al 95% por la técnica Bootstrap

	Intervalo Inferior	Intervalo Superior
Bootstrap C	0.744	0.903
Bootstrap R	0.744	0.903

Table 17: Intervalos de confianza Percentil de la Varianza al 95% por la técnica Bootstrap

Como se observa, la estimación de los errores estándar de los diferentes estadísticos,y sus respectivos intervalos de confianza, son similares si es que no idénticos, al utilizar la función programada en C respecto aquella escrita en R. No obstante, recordemos que escribir el código en C y compilarlo en R es más eficiente (térinos de tiempo) para los estadísticos de la media, mediana, varianza y desviación estándar.

Sección 8. Conclusiones

La estimación y construcción de intervalos es muy parecida. Sin embargo, al tener un tamaño de muestra relativamente grande se desea incrementar el número de simulaciones para que el estimador del error estándar sea consistente.

Pensando de esa manera, lo que se desea es un número grande de simulaciones, y como observamos en la sección de ”Eficiencia en Tiempos”, escribir el código en C y después compilarlo desde R, mejoraría la estimación en términos de realización de un número mayor de simulaciones. De está forma, para obtener mayor poder computacional en un programa que

está pensado para análisis estadístico, como lo es R, es preferible hacerlo en otro lenguaje de programación, en este caso C.

Bibliografía

1. Eddelbullet,Dirk, Seamless R and C++ Integration with Rcpp. Ed. Springer, 2013.

2. Wasserman, Larry. All of statisitcs: A concise course in statistical inference. Ed. Springer, 2005.

3. Wickham, Hadley. Advanced R. Ed. CRC Press, 2015.

Apéndice

Algorithm	Time Complexity		
	Best	Average	Worst
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)
Bubble Sort	Ω(n)	θ(n^2)	O(n^2)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)
Heap Sort	Ω(n log(n))	θ(n log(n))	O(n log(n))
Quick Sort	Ω(n log(n))	θ(n log(n))	O(n^2)
Merge Sort	Ω(n log(n))	θ(n log(n))	O(n log(n))
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)
Radix Sort	Ω(nk)	θ(nk)	O(nk)

Figure 7: Algoritmo de ordenamiento

Código en C

Quick Sort

Código.txt

```
/
void quicksortNumericVector xx1, int l, int r
{
        /       double v,i,j,t;

        /       if r > l{
```

```
            v = xx1[r];
            i= l-1;
            j = r;
            for;;{
                        whilexx1[++i] < v;
                whilexx1[--j] > v;

                ifi >= j
            break;
                t=xx1[i];
                xx1[i]=xx1[j];
                xx1[j]=t;
            }/
            t=xx1[i];
            xx1[i]=xx1[r];
            xx1[r] = t;

            quicksortxx1,l,i-1;
            quicksortxx1,i+1,r;

        }/
} // end quicksort
```

Media

Código.txt

```
/

double  mediaNumericVector xx1
{
        /        int i = 0;
```

```
        double auxSuma = 0.0;
        double mediaBostrap = 0.0;

        /        int n1 = xx1.size;

    /        for i = 0; i < n1; i++ {
                 auxSuma = auxSuma + xx1[i];
        }/
         mediaBostrap = auxSuma / n1;

        return mediaBostrap; /
} // end media
c@FancyVerbLin
```

Mediana

Código.txt

```
/

double  medianaNumericVector xx1
{
        /        int n1 = xx1.size;
        /        quicksortxx1, 0, n1-1;
        /        ifn1%2==0 {
        returnxx1[n1/2] + xx1[n1/2 - 1] / 2.0;
        } else {
        return xx1[n1/2];
        } /
} // end mediana
```

Varianza

Código.txt

```
/

double varianzaNumericVector xx1
{
        /         double mediaBostrap = 0.0; int i = 0;
         double auxVar = 0.0; double varBostrap = 0.0;
         /          int n1 = xx1.size;
         /          mediaBostrap = mediaxx1;

         /          for i = 0; i < n1; i++{
                auxVar = auxVar + powxx1[i] - mediaBostrap, 2;
          }/

         varBostrap = auxVar / doublen1-1;

          return varBostrap; /
} // end varianza
```

Desviación Estándar

Código.txt

```
/

double sdNumericVector xx1
{
         /          return sqrtvarianzaxx1; /
} // end varianza
c@FancyVerbLineea
```

Bootstrap Media

Código.txt

```
  /


List bootstrapMediaNumericVector x1, int B
{
       /        int n1 = 0;  int simulacion = 0; int i = 0; int j = 0; /        double au
       double auxVar = 0.0; double varBostrap = 0.0;
       double sdBostrap = 0.0;
   /        n1 =  x1.size;
       /        simulacion = B;
       /        NumericVector xx1n1;
       /        NumericVector estBootB;

       time_t t;
       srandunsignedtime&t;
       /        for i = 0; i < simulacion; i++{
               forj=0;j< n1;j++{
                       xx1[j] = x1[rand%n1];
        } /
        estBoot[i] = mediaxx1;
       }/         /         for i = 0; i < simulacion; i++{
       auxSuma = auxSuma + estBoot[i];
        }/
       mediaBostrap = auxSuma / simulacion;
        /        for i = 0; i < simulacion; i++{
               auxVar = auxVar + powestBoot[i] - mediaBostrap, 2;
        }/
        /        varBostrap = auxVar / simulacion;
        /
       sdBostrap = sqrtvarBostrap;
```

```
        /        return Rcpp::List::createRcpp::Named"se" = sdBostrap,

}/ }/
```

Bootstrap Mediana

Código.txt

```
/

List bootstrapMedianaNumericVector x1, int B
{
       /        int n1 = 0;  int simulacion = 0; int i = 0; int j = 0;
       double auxSuma = 0.0; double mediaBostrap = 0.0;
       double auxVar = 0.0; double varBostrap = 0.0;
       double sdBostrap = 0.0;
   /        n1 =  x1.size;

       simulacion = B;
       /        NumericVector xx1n1;
       /        NumericVector estBootB;

       time_t t;
       srandunsignedtime&t;
       /        for i = 0; i < simulacion; i++{
               forj=0;j< n1;j++{
               xx1[j] = x1[rand%n1];
       } /
        estBoot[i] = medianaxx1; /        } /
        /         for i = 0; i < simulacion; i++{
               auxSuma = auxSuma + estBoot[i];
        } /
```

```
        mediaBostrap = auxSuma / simulacion;
         /        for i = 0; i < simulacion; i++{
                auxVar = auxVar + powestBoot[i] - mediaBostrap, 2;
         }
         /         varBostrap = auxVar / simulacion;
         /         sdBostrap = sqrtvarBostrap;

        /         return Rcpp::List::createRcpp::Named"se" = sdBostrap,
 } /
```

Bootstrap Varianza

Código.txt

```
/


 List bootstrapVarianzaNumericVector x1, int B
 {
         /        int n1 = 0;  int simulacion = 0; int i = 0; int j = 0;
        double auxSuma = 0.0; double mediaBostrap = 0.0;
        double auxVar = 0.0; double varBostrap = 0.0;
        double sdBostrap = 0;
    /        n1 =  x1.size;
        /        simulacion = B;
        /        NumericVector xx1n1;
        /        NumericVector estBootB;

        time_t t;
        srandunsignedtime&t;

        /        for i = 0; i < simulacion; i++{
```

```
                forj=0;j< n1;j++{
                        xx1[j] = x1[rand%n1];
                 } /                estBoot[i] = varianzaxx1;
        }/
        /         for i = 0; i < simulacion; i++{
                auxSuma = auxSuma + estBoot[i];
         }

        mediaBostrap = auxSuma / simulacion;

        /         for i = 0; i < simulacion; i++{
                auxVar = auxVar + powestBoot[i] - mediaBostrap, 2;
         }

         /         varBostrap = auxVar / simulacion;
         /         sdBostrap = sqrtvarBostrap;
         /          return Rcpp::List::createRcpp::Named"se" = sdBostrap,

 }/ }/
```

Bootstrap Desviación Estándar

Código.txt

```
/

 List bootstrapSDNumericVector x1,  int B
 {
         /         int n1 = 0;  int simulacion = 0; int i = 0; int j = 0;
        double auxSuma = 0.0; double mediaBostrap = 0.0;
        double auxVar = 0.0; double varBostrap = 0.0;
        double sdBostrap = 0.0;
```

```
    /        n1 =  x1.size;
        /         simulacion = B;
        /         NumericVector xx1n1;
        /         NumericVector estBootB;

        time_t t;
        srandunsignedtime&t;
        /         for i = 0; i < simulacion; i++{
                 forj=0;j< n1;j++{
                         xx1[j] = x1[rand%n1];
        } /
        estBoot[i] = sdxx1;
        }/
        /          for i = 0; i < simulacion; i++{
                 auxSuma = auxSuma + estBoot[i];
        }/
        mediaBostrap = auxSuma / simulacion;
        /         for i = 0; i < simulacion; i++{
                 auxVar = auxVar + powestBoot[i] - mediaBostrap, 2;
         }/          /          varBostrap = auxVar / simulacion;
         /         sdBostrap = sqrtvarBostrap;
         /         return Rcpp::List::createRcpp::Named"se" = sdBostrap,

} / } /
```