Magíster en Matemática

UNIVERSIDAD DEL BIO-BIO
Facultad de Ciencias
Concepción - Chile

Magíster en Matemática, Mención Matemática Aplicada o Mención Estadística

Simulación de la probabilidad de Error tipo I de un test de bondad de ajuste para una distribución Poisson Bivariada

Tesis presentada al Programa de *Magíster en Matemática, Mención Matemática Aplicada, o Mención Estadística* del Departamento de Estadística y Departamento de Matemática de la Universidad del Bío-Bío como parte de los requisitos para la obtención del grado de Magíster en Matemática con Mención en Estadística de la Universidad del Bío-Bío.

Claudia Andrea González Agüero

Profesor Director: Dr. Francisco Novoa Muñoz

Concepción, 2020.

Magíster en Matemática

UNIVERSIDAD DEL BIO-BIO
Facultad de Ciencias
Concepción - Chile

Simulación de la probabilidad del Error tipo I de un test de bondad de ajuste para una distribución Poisson Bivariada

Por

Claudia Andrea González Agüero

Disertación presentada al Programa de *Magíster en Matemática, Mención Matemática Aplicada, o Mención Estadística* del Departamento de Estadística y Departamento de Matemática de la Universidad del Bío-Bío, como requisito parcial para la obtención del Grado de Magíster en Matemática con Mención Estadística.

Aprobada por:

_____

Dr. Francisco Novoa Muñoz
Profesor Guía
Departamento de Estadística – Universidad del Bío-Bío, Chile

_____

Dr. Sergio Contreras Espinoza
Profesor Informante
Departamento de Estadística – Universidad del Bío-Bío, Chile

_____

Dr. Rubén Carvajal Schiaffino
Profesor Informante
Departamento de Matemática y Ciencia de la Computación – Universidad de Santiago de Chile

Concepción, 2020.

Resumen

Determinar el modelo probabilístico del cual proviene una determinada muestra de datos, permite hacer predicciones y tomar decisiones con un alto nivel de asertividad, con base en dicha muestra. Una manera de conocer si un conjunto de datos se comporta de acuerdo con la distribución de probabilidad en la que se basa un modelo probabilístico específico, es a través de un test de bondad de ajuste. Este trabajo de investigación tiene por objetivo simular la probabilidad de error tipo I de un test de bondad de ajuste propuesto para una distribucin Poisson Bivariante. Se plantea estudiar los resultados para distintos tamaños muestrales y para muestras provenientes de poblaciones de diversas caractersticas. Con este fin, se buscan alternativas de paralelización en el lenguaje R, entre ellas se selecciona la que permite minimizar los tiempos de cómputo y, con base en esta alternativa, se simulan las probabilidade del error tipo I buscadas. Se prueba satisfactoriamente que el lenguaje R permite la paralelización del proceso en cuestión, al menos de dos maneras distintas y que estas alternativas permiten una disminución significativa de los tiempos de cómputo, en comparación con el mismo procedimiento de manera secuencial. A su vez, el trabajo de investigación permite la simulación con tamaños muestrales y vectores de parámetros no estudiados anteriormente, dejando en evidencia que la probabilidad del error tipo I del test es cercana al valor nominal en la mayoría de los casos.

Abstract

Determining the probabilistic model from which a certain data sample comes, allows making predictions and making decisions with a high level of assertiveness, based on the sample. One way to know if a data set behaves according to the probability distribution on which a specific probabilistic model is based is through a goodness-of-fit test. The objective of this research work is to simulate the probability of type I error of a test of goodness-of-fit proposed for a Bivariate Poisson distribution. It is proposed to study the results for different sample sizes and for samples from populations of different characteristics. In order to achieve this objective, parallelization alternatives are sought in the R language, among them the one that allows minimizing computation times is selected and, based on this alternative, the probabilities of type I error sought are simulated. It is satisfactorily proven that the R language allows the parallelization of the process in question, at least in two different ways and that these alternatives allow a significant decrease in computation times, compared to the same procedure sequentially. In turn, the research work allows simulation with sample sizes and parameter vectors not previously studied, making it clear that the probability of the type I error of the test is close to the nominal value in most cases.

Agradecimientos

En primer lugar agradezco a Dios por todo lo que me ha dado.

Doy las gracias a mi hermana Alejandra por motivarme a estudiar, confiar en mis capacidades y dedicarme tiempo cada vez que se lo pedí.

Agradezco a mi madre por ayudarme a superar este desafío dándome la tranquilidad de que mis hijos estarían en las mejores manos.

Doy las gracias al profesor Dino Risso por su apoyo desinteresado, por facilitar los recursos computacionales que hicieron posible llevar a cabo esta investigación y por dedicar muchas horas de su tiempo en instruirme en el uso de estos.

Agradezco muy especialmente al profesor Francisco Novoa por su dedicación, por su paciencia infinita para aclarar cada nueva duda, por aceptar mi falta de organización y sobre todo le agradezco por no permitirme decaer.

De manera muy especial agradezco a mis hijos Álvaro, Tomás y Emilia, por comprender el interés personal de seguir aprendiendo.

Finalmente agradezco a mi esposo Ricardo por apoyarme y acompañarme en cada nuevo desafío.

A mi familia.

Índice general

Índice de tablas

Introducción

En la actualidad es imprescindible anticiparse a los hechos que puedan ocurrir, para ello es vital contar con modelos que predigan dichos sucesos con la máxima precisión posible. La estadística posee herramientas para hacer predicciones o tomar decisiones basándose en la información contenida en las muestras de datos de las que se disponen. Por lo general, dichas herramientas consisten en modelos basados en distribuciones de probabilidad, pero antes de usar tales modelos es esencial averiguar si los datos muestrales se comportan de acuerdo a la distribución de probabilidad en la que se basa el modelo probabilístico. Esto se logra aplicando un test de bondad de ajuste.

Una parte esencial de la generación de un test bondad de ajuste es el resultado de la simulación bajo la hipótesis nula, lo que permite confirmar si se alcanza el nivel nominal establecido. Los autores Novoa-Muñoz y Jiménez-Gamero (2014) generaron tests de bondad de ajuste y usaron el método bootstrap paramétrico para simular la probabilidad de error tipo I, sin embargo los altos tiempos de cómputo de cada simulación influyó en que hicieran pocas variaciones al vector de parámetros y a los tamaños muestrales considerados.

En esta investigación se busca superar las limitantes que tuvieron los autores del test de modo de conseguir hacer mayor variación de las componentes consideradas al hacer las simulaciones de la probabilidad de error tipo I. Como una manera de conseguir lo anterior es que se propone buscar alternativas de paralelización en el lenguaje R y entre ellas seleccionar la que permita minimizar los tiempo de cómputo de cada simulación. De esta manera se podrá considerar un mayor número de vectores de parámetros así como también se podrá considerar tamaños muestrales mayores y variar los exponentes en la función de peso involucrada en el estadístico utilizado en el test propuesto.

El presente documento se organiza de la siguiente manera:

El Capítulo 1 da a conocer los objetivos de investigación propuestos.

El Capítulo 2 nos presenta una serie de resultados anteriores y definiciones en los que se basa el estudio. Se encuentra aquí la definición de distribución Poisson bivariante junto a sus características. Se dan a conocer los principales test de bondad de ajuste para esta distribución y se detalla el primer test propuesto por Novoa-Muñoz y Jiménez-Gamero (2014). Por otro lado se consideran algunos conceptos básicos relacionados con la paralelización, así como las principales herramientas que ofrece el lenguaje R para implementarla.

A continuación, en el Capítulo 3 se da a conocer la metodología de investigación utilizada, considerando la manera en que se seleccionó la variación que de los componentes y por otro lado qué recursos computacionales se utilizaron y de qué manera. Se presenta en este capítulo las tres etapas de las que contó este estudio.

En el Capítulo 4 se dan a conocer los resultados de las simulaciones realizadas. En primer lugar se muestran los resultados del análisis de las paralelizaciones implementadas, para luego dar a conocer los resultados de las simulaciones al variar vectores de parámetros y tamaños muestrales.

Finalmente, en el Capítulo 5 se presentan las conclusiones que permite obtener este trabajo de investigación as como las sugerencias para trabajos futuros.

Capítulo 1

Objetivos

1.1. Objetivo General

Simular la probabilidad de error tipo I del primer test de bondad de ajuste propuesto por Novoa-Muñoz y Jiménez-Gamero (2014) ampliando el espectro de variación de los componentes estudiados.

1.2. Objetivos Específicos

1. Simular la probabilidad de error tipo I del primer test de bondad de ajuste propuesto por Novoa-Muñoz y Jiménez-Gamero (2014)
   - utilizando tamaños muestrales ya considerados por estos autores ($n = 30, 50, 70$) y otros mayores ($n = 100, 200$).
   - al variar el vector de parámetros, de modo de obtener coeficientes de correlación distintos a los ya considerados por los autores.
2. Seleccionar el método de paralelización, entre las alternativas que ofrece R, que minimice el tiempo de CPU de las simulaciones.

Capítulo 2

Marco teórico

2.1. Distribución Poisson bivariante

Se han dado varias definiciones para la distribución Poisson Bivariante, en adelante DPB (ver por ejemplo, Kocherlakota y Kocherlakota (1992, pp. 87–90) [12], para una revisión detallada). En esta memoria, consideraremos la siguiente, que es la que ha recibido más atención en la literatura estadística (ver por ejemplo, Holgate (1964) [8]; Johnson, Kotz y Balakrishnan (1997) [10]).

Definición 2.1.1 *(Johnson, Kotz y Balakrishnan (1997, pp. 124–125) [10]) Sean*

$$X_1 = Y_1 + Y_3 \quad \text{y} \quad X_2 = Y_2 + Y_3\,,$$

donde Y_1, Y_2 e Y_3 son v.a. Poisson, mutuamente independientes con medias dadas por $\theta_1' = \theta_1 - \theta_3 > 0$, $\theta_2' = \theta_2 - \theta_3 > 0$ y $\theta_3 \geq 0$, respectivamente.

A la distribución conjunta del vector (X_1, X_2) se le denomina **distribución Poisson bivariante** *con parámetro $\theta = (\theta_1, \theta_2, \theta_3)$, lo cual denotaremos mediante $(X_1, X_2) \sim PB(\theta_1, \theta_2, \theta_3)$ o simplemente $(X_1, X_2) \sim PB(\theta)$.*

La función de probabilidad conjunta de X_1 y X_2 está dada por

$$P_\theta(X_1 = x_1, X_2 = x_2) = \exp(\theta_3 - \theta_1 - \theta_2) \sum_{i=0}^{\min\{x_1,x_2\}} \frac{(\theta_1 - \theta_3)^{x_1-i}\,(\theta_2 - \theta_3)^{x_2-i}\,\theta_3^i}{(x_1 - i)!\;(x_2 - i)!\;i!},$$

donde $x_1, x_2 \in \mathbb{N}_0$.

Además, como las v.a. Y_1, Y_2 e Y_3 son mutuamente independientes, entonces, la función generatriz de probabilidad (fgp) conjunta, $g(u;\theta)$, de la DPB se obtiene mediante

$$\begin{aligned} g(u;\theta) &= E_\theta\left(u_1^{X_1}\, u_2^{X_2}\right), \\ &= E_\theta\left(u_1^{Y_1+Y_3}\, u_2^{Y_2+Y_3}\right) = E_{\theta_1'}\left(u_1^{Y_1}\right) E_{\theta_2'}\left(u_2^{Y_2}\right) E_{\theta_3}\left\{(u_1u_2)^{Y_3}\right\} \\ &= \exp\{\theta_1'(u_1-1) + \theta_2'(u_2-1) + \theta_3(u_1u_2-1)\}, \\ &= \exp\{(\theta_1-\theta_3)(u_1-1) + (\theta_2-\theta_3)(u_2-1) + \theta_3(u_1u_2-1)\}, \\ &= \exp\{\theta_1(u_1-1) + \theta_2(u_2-1) + \theta_3(u_1-1)(u_2-1)\}, \end{aligned} \tag{2.1}$$

$\forall\, u = (u_1, u_2) \in \mathbb{R}^2$, $\forall\, \theta \in \Theta$.

Observación 2.1.2 *Si $\theta_3 = 0$, entonces $X_1 = Y_1$ y $X_2 = Y_2$, y por tanto X_1 y X_2 son independientes, pues Y_1 e Y_2 son v.a. Poisson mutuamente independientes.*

2.2. Bootstrap paramétrico

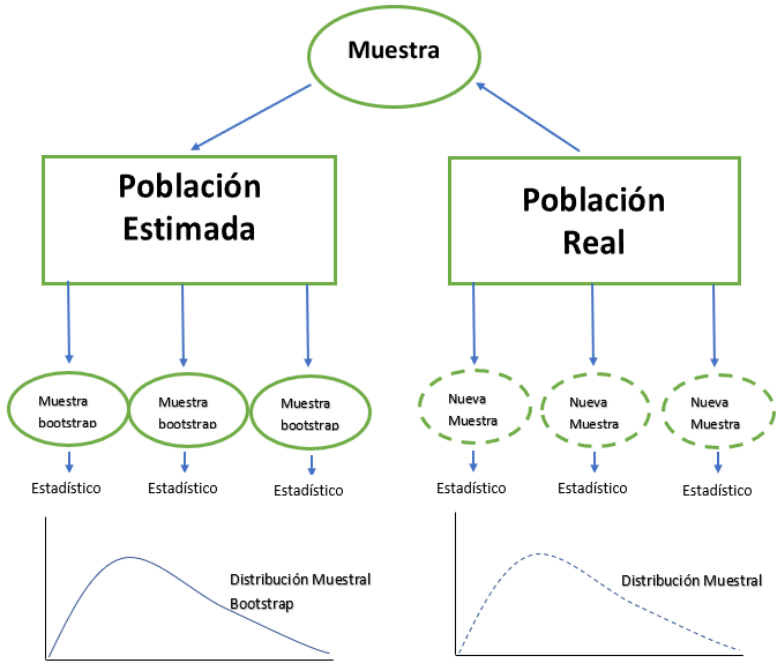
El Bootstrap es una metodología de remuestreo de datos propuesto por Bradley Efron en 1979, como un método que reemplaza las técnicas complejas de análisis estadístico convencional por cálculos intensivos sobre las muestras generadas por simulación de Monte Carlo. La publicación del primer artículo de Bradley Efron sobre métodos bootstrap fue un evento importante en las estadísticas que sintetizaron algunas de las ideas de remuestreo anteriores y establecieron un nuevo marco para el análisis estadístico basado en simulación [7]. El autor introduce la metodología bootstrap para estimar las distribuciones de algunos estadísticos cuando el tamaño muestral es pequeño o las expresiones de dichas distribuciones son analíticamente intratables. Desde entonces numerosos autores han desarrollado métodos bootstrap para diversos procedimientos inferenciales, tales como modelos de regresión, datos censurados, construcción de intervalos de confianza, estimación de parámetros, entre otros [2].

La metodología bootstrap se utiliza, entre otras cosas, para aproximar la distribución de probabilidad del estadístico de interés cuando la hipótesis nula se cumple (distribución nula) en los casos en que este no tiene una distribución teórica conocida. Esta distribución podría ser determinada si se conocieran todas las posibles muestras

de la población en estudio, sin embargo, en la práctica la mayoría de las veces se tiene sólo una muestra observada.

La esencia de la metodología de remuestro Bootstrap es considerar que esta muestra observada es una buena representación de la población, así los nuevos conjuntos de datos (muestras bootstrap) se extraerán de esta muestra original.

Figura 2.1: Esquema de representación del bootstrap paramétrico.



Los pasos del algoritmo bootstrap se ilustran en la figura 2.1. Las cantidades observadas se denotan mediante líneas continuas y las cantidades no observadas mediante líneas discontinuas. El esquema muestra que a lo que se puede llegar a travs de las muestras bootstrap extradas es a estimar la verdadera distribución muestral del estadístico de interés.

Es importante considerar que hay dos situaciones que distinguir, la paramétrica y la no paramétrica. Cuando hay un modelo matemático particular con constantes ajustables o parámetros que determinan completamente la distribución del estadístico, dicho modelo se llama paramétrico y los métodos estadísticos basados en este modelo son métodos paramétricos [7].

Considerando lo anterior, dependiendo del modo en que se forman las muestras Bootstrap, se hablará de Bootstrap paramétrico o no paramétrico. En este trabajo nos enfocaremos en el Bootstrap Paramétrico.

El Bootstrap paramétrico, a diferencia de las otras modalidades de este método, simula las muestras Bootstrap a partir de los parámetros obtenidos en la muestra observada. El algoritmo propuesto se puede resumir en las siguientes etapas:

- Extraer una muestra aleatoria de tamaño n de la población (o bien, simular una muestra independiente identicamente distribuida, en adelante iid, de una determinada distribución).
- Estimar los parámetros de la distribución a partir de la muestra observada.
- Estimar el estadístico de interés a partir de la muestra observada.
- Simular B muestras Bootstrap de tamaño n a partir del modelo paramétrico estimado.
- A partir de cada muestra Bootstrap calcular el estadístico de interés.

Si bien el método bootstrap ha sido de gran utilidad en la inferencia estadística, Andrew (2000) nos muestra que hay situaciones en las que éste no es consistente. En su artículo nos da a conocer ejemplos en los que el bootstrap paramétrico no da buenos resultados y en particular, prueba que el método es inconsistente cuando el verdadero valor del parámetro está en la frontera del espacio paramétrico o muy cercano a ella.

2.3. Test de Bondad de ajuste de una DPB

Cuando se pretende realizar un estudio estadístico a partir de una muestra $X_1, ..., X_n$ iid extraída de una población con cierta distribución F, las propiedades de este modelo serán fundamentales para conseguir buenas inferencias. Sin embargo, en la mayoría de las situaciones, la distribución F es desconocida por lo que surge la necesidad de averiguar si esta coincide con algún modelo F_0 conocido.

Se entiende por bondad de ajuste al grado de cercanía que tiene la distribución de nuestras observaciones con un modelo probabilístico determinado.

Una manera de contrastar la bondad de ajuste de las observaciones dadas con el modelo supuesto es a través de la aplicación de un test de bondad de ajuste, el que basa sus hipótesis nula (H_0) y alternativa (H_1) en la distribución de probabilidad teórica.

$$H_0 : X \sim F_0(\theta), \text{ para algún } \theta \in \Theta$$
$$H_1 : X \not\sim F_0(\theta), \forall \theta \in \Theta$$

La literatura sobre tests de bondad de ajuste para la DPB es escasa. Podemos mencionar el test propuesto por Crockett [6], el test desarrollado por Loukas y Kemp [14], que se basa en una extensión del índice de dispersión univariante, y el test sugerido por Rayner y Best [21], que consiste en una modificación del test dado por Loukas y Kemp [14]. La principal desventaja de estos tests de bondad de ajuste es que no son consistentes.

Novoa Muñoz y Jiménez Gamero [17] proponen dos tests de bondad de ajuste para la DPB, los que cuentan con la consistencia de la que carecen los anteriores. Esto lo logran basándose en la función generatriz de probabilidad empírica.

2.4. Primer test de Bondad de ajuste de Novoa y Jiménez

Este apartado toma como referencia lo expuesto por los autores Novoa y Jiménez (2014), donde detallan los tests de bondad de ajuste que proponen y las justificaciones teóricas de ellos.

Todos ellos consideran que:

$\boldsymbol{X}_1 = (X_{11}, X_{21}), \boldsymbol{X}_2 = (X_{12}, X_{22}), \ldots, \boldsymbol{X}_n = (X_{1n}, X_{2n})$ son vectores aleatorios iid de $\boldsymbol{X} = (X_1, X_2) \in \mathbb{N}_0^2$.

Y tienen por hipótesis nula

$$H_0 : (X_1, X_2) \sim PB(\theta_1, \theta_2, \theta_3), \text{ para algún } (\theta_1, \theta_2, \theta_3) \in \Theta,$$

contra la alternativa

$$H_1 : (X_1, X_2) \nsim PB(\theta_1, \theta_2, \theta_3), \ \forall (\theta_1, \theta_2, \theta_3) \in \Theta.$$

En particular, el primer test de bondad de ajuste propuesto por los investigadores, en adelante, test $R_{n,w}$, es del tipo Cramér Von Misses y se basa en el estadístico:

$$R_{n,w}(\hat{\theta}_n) = \int_0^1 \int_0^1 G_n^2(u;\hat{\theta}_n) w(u) du, \tag{2.2}$$

donde

$$G_n(u;\hat{\theta}_n) = \sqrt{n}\left\{ g_n(u) - g(u;\hat{\theta}_n) \right\},$$

$\hat{\theta}_n = \hat{\theta}_n(\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n) = (\hat{\theta}_{1n}, \hat{\theta}_{2n}, \hat{\theta}_{3n})$ es un estimador consistente de θ, $g_n(u), u \in [0,1]^2$ es la fución generatriz de probabilidad, $g(u;\hat{\theta}_n)$ la función generatriz de probabilidad empírica y $w(u)$ es una función medible de peso tal que $w(u) \geq 0,\ \forall u \in [0,1]^2$, y

$$\int_0^1 \int_0^1 w(u) du < \infty. \tag{2.3}$$

Este último supuesto sobre w asegura que la integral doble en (2.2) es finita para cada n fijo.
La regla de decisión del test considera que, se rechazará la hipótesis nula H_0 para valores "grandes" de $R_{n,w}(\hat{\theta}_n)$.

El test $R_{n,w}$ se basa en que la función generatriz de probabilidad empírica es una buena estimación de la fgp; así , si H_0 es verdadera y $\hat{\theta}_n$ es un estimador consistente de θ, entonces $g(u;\hat{\theta}_n)$ estima consistentemente la función generatriz de probabilidad de los datos.

Como la distribución de $\boldsymbol{X}$ es determinada de forma única por su función generatriz de probabilidad, $g(u)$, $u \in [0,1]^2$, si H_0 es cierta, el estadístico $R_{n,w}(\hat{\theta}_n)$ debiese tener valores "suficientemente" pequeños. Así se rechazará H_0 para valores "grandes" de $R_{n,w}(\hat{\theta}_n)$.

Ahora, para determinar cuáles son los valores grandes, se debe calcular la distribución nula del estadístico o al menos una aproximación de ésta.

Puesto que las distribuciones nulas son desconocidas, se intenta buscar una aproximación. Un modo clásico de estimar la distribución nula es mediante la distribución asintótica nula.

A pesar de que los autores determinan una distribución asintótica nula, estas dependen del verdadero valor del parámetro θ, por lo tanto, en la práctica no proporcionan

una solución útil al problema de estimar la distribución nula del test estadístico. Si bien, este problema se podría sobrellevar al reemplazar θ por $\hat{\theta}_n$, aún la expresión encontrada considera funciones muy complejas, por lo que surge la necesidad de buscar un modo alternativo de aproximación de la distribución nula.

El método utilizado por los autores para estimar la distribución nula del test de bondad de ajuste propuesto, es el bootstrap paramétrico. Siguiéndo los pasos de este método, si se desea realizar el test con una significancia del $p\,\%$ se comienza por estimar el parámetro θ a partir de la muestra observada. Luego, a partir de este valor se calcula el valor de estadístico $R_{n,w}(\hat{\theta}_n)$ de acuerdo a 2.2. Se procede a simular B muestras DPB de parámetro $\hat{\theta}_n$, para luego en cada una de ella estimar nuevamente el parámetro poblacional $\hat{\theta}_n^*$ y en base a éste, el estadístico de interés $R_{n,w}(\hat{\theta}_n^*)$. Conociendo el valor de los B estadísticos se determina el percentil p de su distribución. Luego, si $R_{n,w}(\hat{\theta}_n)$ es menor que el valor obtenido, se rechazará H_0, en caso contrario, no se rechazará.

2.5. Resultados y dificultades del primer test de Bondad de ajuste de Novoa y Jiménez

Una parte fundamental en la generación de un test de bondad de ajuste es aquella en que se analizan los resultados de simulación bajo la hipótesis nula, que permiten confirmar si se alcanza el nivel nominal establecido.

Los autores llevaron a cabo la simulación bajo el lenguaje R y compararon sus resultados con los test ya propuestos por Crockett [6], Loukas y Kemp [14], y Rayner y Best [21].

Para estudiar la bondad de la aproximación bootstrap en muestras de tamaño finito tanto para los tests propuestos, como para las aproximaciones a la distribución nula de los otros tests estadísticos, que están basados en sus distribuciones asintóticas nulas, generaron muestras de tamaño $n = 30, 50, 70$ de la distribución $PB(\theta_1, \theta_2, \theta_3)$, con $\theta_1 = \theta_2 =$1.0 y θ_3 tal que el coeficiente de correlación, $\rho = \theta_3/\sqrt{\theta_1\,\theta_2}$ sea igual a 0.25, 0.50 y 0.75, con el fin de examinar la bondad de las aproximaciones para datos con una correlación baja, media y alta, respectivamente. Luego, se hizo un trabajo similar con los parámetros $\theta_1 = 1.5$, $\theta_2 = 1$.

Para estimar el parámetro θ emplearon el método de máxima verosimilitud como se describe en Kocherlakota y Kocherlakota (1992, pp. 103–105) [12]. Luego, aproximaron

los $p-$valores bootstrap, p^*, de los tests propuestos, para ello se usó la función de peso $w(u; a_1, a_2) = u_1^{a_1} u_2^{a_2}$ para $(a_1, a_2) \in \{(0,0), (1,0)\}$ y generaron $B = 500$ muestras bootstrap. También, calcularon los $p-$valores (asintóticos) asociados a los otros test.

El procedimiento anterior lo repitieron 1000 veces y calcularon la fracción de los $p-$valores estimados que resultaron ser menores o iguales que 0.05 y 0.10, que son las estimaciones de las probabilidades del error tipo I para $\alpha = 0.05$ y 0.10 (f05 y f10 respectivamente). Con las mismas muestras anteriores se les calculó los p-valores (asintóticos) asociados a los tres test estadísticos propuestos previamente.

De acuerdo a lo expresado por los autores, los resultados obtenidos muestran que el bootstrap proporciona una aproximación precisa a la distribución nula de $R_{n,w}(\hat{\theta}_n)$, en todos los casos tratados, superando a las aproximaciones asintóticas de los p-valores de los test estadísticos de comparación.

Adicionalmente, se estudió la potencia de los tres test propuestos, además de los tres tests de comparación, concluyendo que el test $R_{n,w}$ fue el de mayor potencia.

Las simulaciones de la probabilidad de error tipo I realizadas por los autores, les permiten obtener los siguientes resultados:

Tabla 2.1: Resultados simulación probabilidad de error tipo I. Caso $E(X_1) = E(X_2)$, $\rho = 0.25$, por Jimenez y Novoa (2014)

	$n = 30$		$n = 50$		$n = 70$	
θ=(1.0, 1.0, 0.25) $\rho = 0{,}25$	f05	f10	f05	f10	f05	f10
$a_1 = 0$, $a_2 = 0$	0.035	0.091	0.047	0.100	0.041	0.083
$a_1 = 1$, $a_2 = 0$	0.035	0.093	0.050	0.100	0.042	0.086

Tabla 2.2: Resultados simulación probabilidad de error tipo I. Caso $E(X_1) = E(X_2)$, $\rho = 0.50$, por Jimenez y Novoa (2014)

	$n = 30$		$n = 50$		$n = 70$	
θ=(1, 1, 0.5) $\rho = 0.5$	f05	f10	f05	f10	f05	f10
$a_1 = 0$, $a_2 = 0$	0.047	0.091	0.044	0.101	0.048	0.100
$a_1 = 1$, $a_2 = 0$	0.049	0.097	0.039	0.96	0.054	0.097

Tabla 2.3: Resultados simulación probabilidad de error tipo I. Caso $E(X_1) = E(X_2)$, $\rho = 0.75$, por Jimenez y Novoa (2014)

	$n = 30$		$n = 50$		$n = 70$	
θ=(1, 1, 0.75) $\rho = 0.75$	f05	f10	f05	f10	f05	f10
$a_1 = 0$, $a_2 = 0$	0.045	0.097	0.60	0.107	0.056	0.107
$a_1 = 1$, $a_2 = 0$	0.052	0.096	0.059	0.106	0.050	0.109

Tabla 2.4: Resultados simulación probabilidad de error tipo I. Caso $E(X_1) \neq E(X_2)$, $\rho \approx 0.25$, por Jimenez y Novoa (2014)

	$n = 30$		$n = 50$		$n = 70$	
θ=(1.5, 1, 0.31) $\rho = 0.25311$	f05	f10	f05	f10	f05	f10
$a_1 = 0$, $a_2 = 0$	0.048	0.101	0.053	0.110	0.047	0.100
$a_1 = 1$, $a_2 = 0$	0.046	0.097	0.053	0.104	0.046	0.104
$a_1 = 0$, $a_2 = 1$	0.055	0.106	0.061	0.112	0.048	0.090

Tabla 2.5: Resultados simulación probabilidad de error tipo I. Caso $E(X_1) \neq E(X_2)$, $\rho \approx 0.50$, por Jimenez y Novoa (2014)

	$n = 30$		$n = 50$		$n = 70$	
θ=(1.5, 1, 0.62) $\rho = 0.50623$	f05	f10	f05	f10	f05	f10
$a_1 = 0$, $a_2 = 0$	0.050	0.094	0.048	0.093	0.054	0.110
$a_1 = 1$, $a_2 = 0$	0.049	0.093	0.046	0.092	0.050	0.104
$a_1 = 0$, $a_2 = 1$	0.050	0.089	0.043	0.100	0.054	0.100

Tabla 2.6: Resultados simulación probabilidad de error tipo I. Caso $E(X_1) \neq E(X_2)$, $\rho \approx 0.75$, por Jimenez y Novoa (2014)

	$n = 30$		$n = 50$		$n = 70$	
θ=(1.5, 1, 0.92) ρ =0.75118	f05	f10	f05	f10	f05	f10
$a_1 = 0$, $a_2 = 0$	0.059	0.102	0.053	0.094	0.046	0.091
$a_1 = 1$, $a_2 = 0$	0.051	0.096	0.051	0.099	0.050	0.091
$a_1 = 0$, $a_2 = 1$	0.058	0.104	0.049	0.098	0.046	0.089

Si bien los autores pudieron concluir que para las muestras estudiadas el test $R_{n,w}$ era el que entregaba los mejores resultados. No pudieron extender esta conclusión a muestras con vectores de parámetros distintos ni de tamaños muestrales mayores de-

bido a que el tiempo de cómputo que requería cada una de estas simulaciones lo hacía imposible dentro del período comprendido para la investigación.

2.6. Programación en Paralelo

Es sabido que durante las últimas décadas los avances computacionales han sido muy importantes, pero así como aumenta la tecnología, los problemas que surgen también lo hacen. Son muchos los ejemplos que se podrían dar [18].

Una manera en que la ciencia de la computación ha contribuido en resolver estos problemas ha sido mediante la paralelización o paralelismo computacional. La computación paralela es una forma de computación en la cual, para resolver un problema, varias operaciones o procesos se desarrollan simultáneamente [1].

Los avances tecnológicos han permitido que en la actualidad la computación paralela sea de uso común, principalmente por la llegada de los procesadores de varios núcleos en la mayoría de dispositivos computacionales. Por otro lado, los softwares han evolucionado para facilitar el uso en paralelo de los distintos núcleos de los que se dispone, considerando que los programas paralelos son más complejos de escribir que los programas secuenciales, ya que se requieren que haya una comunicación y sincronización entre las tareas que se han paralelizado. Sin embargo, no basta con tener un equipo y software adecuado para poder trabajar en paralelo, ya que es trascendental evaluar la factibilidad de paralelizar la tarea de interés. Una tarea puede ser desde inherentemente secuencial hasta absolutamente paralelizable. Esto dependerá de cuan independiente sean las instrucciones entre sí [11].

Pacheco (2011) indica que hay dos términos estréchamente relacionados con la computación paralela que es importante conocer al estudiar el tema: computación concurrente y computación distribuida. El primero habla de la simultaneidad de las tareas, independientemente de si estas se desarrollan en un mismo núcleo o no. A su vez, la computación distribuida se refiere a la distribución física de las partes del proceso. Considerando que en la literatura no hay claridad frente a estos términos, podríamos considerar que un sistema paralelo es un ejemplo de computción concurrente y podría, dependiendo de su implementación, ser un caso de computación distribuida.

La mayoría de los programas que se han escrito para los sistemas convencionales de un sólo núcleo no pueden aprovechar la presencia de múltiples núcleos.
Para lograr sacar provecho de esta situación, necesitamos reescribir nuestros programas en serie para que sean paralelos, o bien, escribir programas de traducción, es decir, programas que conviertan automáticamente los programas en serie en programas pa-

ralelos. La mala noticia es que los investigadores han tenido un éxito muy limitado escribiendo programas que convierten programas en serie en lenguajes como C y C ++ en programas paralelos [18].

Existen dos enfoques principales para pensar la programación en paralelo: paralelismo de tareas y paralelismo de datos. En el paralelismo de tareas se identifican todas las tareas que se deben llevar a cabo para resolver el problema y estas se distribuyen entre los distintos núcleos. Por otro lado, el paralelismo de datos particiona los datos para distribuirlos entre los distintos núcleos, en cada uno de los cuales se desarrollarán todas las tareas involucradas [18]. En ambos casos, es necesario que exista coordinación y comumnicación entre los núcleos para poder resolver el problema de interés.

Una implementación paralela eficiente de un programa en serie no se encuentra en la paralelización eficiente de cada uno de sus pasos. Por el contrario, la mejor paralelización se obtendrá volviendo atrás e ideando un algoritmo completamente nuevo [18]. Hay muchas maneras de escribir un programa en paralelo, pero todas consideran particionar el trabajo para que este sea realizado entre los distintos núcleos disponibles.

El método de remuestreo Bootstrap es un claro ejemplo de que en ocasiones los trabajos científicos requieren de una gran cantidad de procesos computacionales que suponen una tarea muy compleja y demorosa para un sólo procesador.

2.7. Paralelización en lenguaje R

El lenguaje de programación R ofrece la posibilidad de desarrollar tareas en paralelo, cuando estas lo permiten, a través de una serie de paquetes que facilitan comandos para este tipo de procesamiento. Entre los paquetes que ayudan a realizar esta tarea se encuentran *Snow*, *Multicore*, *Parallel*, entre otros.

Hoy, uno de los paquetes que ha cobrado gran protagonismo en la programación en lenguaje R, es *Parallel* (2018), que rescata lo mejor de dos versiones anteriores *Multicore* y *Snow*, lo que se complementa con la generación de números aleatorios [23] [4]. Este paquete contiene versiones en paralelo para todas las funciones de la familia *Apply*, además de proveernos de los comandos necesarios para averiguar cuántos núcleos lógicos

se tienen a disposición y para crear manualmente un cluster.La familia *Apply* mencionada es la que permite la manipulación directa de cada uno de los datos de vectores, matrices, listas o data frame, evitando el uso de ciclos.
Así, el paquete *parallel* permite, distribuir los datos en los distintos núcleos disponibles y ahí, de manera simultanea, manipularlos según la función indicada, para luego reunir toda la información y retornarla como una lista. [5]

En términos generales podemos distinguir dos maneras de implementar el paralelismo haciendo uso de este paquete: a través de forking o sockets. El primer método, forking se basa en la duplicación completa del proceso maestro con el entorno compartido hacia cada uno de los entornos paralelos, incluidos los objetos o variables definidos antes del inicio de subprocesos paralelos. Este método tiene la ventaja de ser muy rápido, pero no puede ejecutarse bajo Windows. Por otro lado, el método sockets se basa en que cada subproceso se ejecuta por separado sin compartir objetos o variables, que solo se pueden pasar del proceso maestro explícitamente. Como resultado, se ejecuta más lentamente debido a la sobrecarga de comunicación, pero tiene la ventaja de poder se implementado en cualquier sistema operativo. [5]

Así como el paquete *Parallel* ofrece herramientas que permiten la paralelización en R, existe un paquete *Boot* (2019) que incorpora funciones que facilitan la aplicación del método bootstrap, basadas en el libro "Métodos boostrap y sus aplicaciones" de Davison y Hinkley (1997). Este paquete, que tiene como co-autor al creador del paquete *Parallel*, considera la posibilidad de hacer las operaciones en paralelo usando forking o socket.

Al programar en paralelo en lenguaje R, con el objetivo de reducir tiempos de cómputo, es importante tener en cuenta algunas recomendaciones de expertos. En muchos casos, simplemente "bastante rápido" es bastante bueno. La velocidad extra que podríamos alcanzar al pasar de R a C / C ++ no justifica la gran cantidad de tiempo necesario para escribir, depurar y mantener el código a ese nivel [16].

2.8. Evaluación del desempeño de un programa en paralelo

Evidentemente el objetivo principal de programar en paralelo es mejorar el desempeño del programa respecto del programa secuencial equivalente, pero surge la interrogante de cómo evaluar esta mejora. Aquí es cuando aparecen dos nuevos conceptos: velocidad y eficiencia.

Cuando se paraleliza un programa que inicialmente era secuencial, lo habitual es, esperar que el tiempo que tarde, sea igual al cuociente entre el tiempo del programa secuencial y el número de núcleos con el que se trabaje, sin embargo se debe considerar que cada uno de estos núcleos podría tener un trabajo adicional que se traduciría en aumento del tiempo, respecto del esperado.

Si se llama T_s al tiempo de la ejecución secuencial del programa y T_p al tiempo de ejecución en paralelo con p núcleos, se define la velocidad de un programa en paralelo como $S = \frac{T_s}{T_p}$. Si $S = p$, diremos que el programa tiene velocidad lineal [18].

Es esperable que en la medida que aumenten los núcleos utilizados haya más trabajo (y tiempo) adicional respecto del esperado, es decir, que la razón entre S y p disminuya su valor. Así, se define la eficiencia de un programa en paralelo como $E = \frac{S}{p}$ [18].

Otro concepto que cobra relevancia al evaluar el desempeño de un programa en paralelo es el de escalabilidad. Este concepto utilizado ampliamente en tecnología, en programación es el caracteriza a un programa para el cual se puede encontrar una determinada taza de crecimiento del problema.

Capítulo 3

Metodología

Para cumplir con los objetvos propuestos en esta investigación, se dividió el trabajo en tres partes principales: preparación de los programas, simulación con los mismos componentes que los utilizados por Jiménez y Novoa (2014) y simulaciones con variación de componentes. A continuación se detalla cada una de las etapas efectuadas.

- Preparación de los programas.
  En primer lugar, se analizó el proceso de simulación de la probabilidad del error tipo I utilizado por los autores del test R, de modo de identificar aquellas partes del programa que podían modificarse para reducir tiempos de cómputo. Por otro lado se analizó la factibilidad de paralelizar el código con lo que se identificó aquellas partes del proceso que permitían este tipo de trabajo y aquellas que no. Habiendo identificado las partes del proceso posibles de paralelizar, se seleccionó dos comandos del lenguaje R que permitían llevarlo a cabo en este caso en particular: parRapply del paquete *Parallel* y boot del paquete *Boot*.
  En base a lo anterior se desarrollaron nuevos códigos especialmente diseñados para paralelizar el proceso de simulación haciendo uso de los dos comandos seleccionados respectivamente.

- Simulación según componentes utilizados por Jiménez y Novoa (2014).
  Para cumplir con el primer objetivo específico planteado en esta investigación es que se replicaron las simulaciones efectuadas por los autores del test R, considerando además de la ejecución secuencial del código, la ejecución en paralelo, haciendo variar el número de núcleos p utilizados desde 2 a 12.

De cada simulación efectuada se registró además de la probabilidad de error tipo I resultante, el tiempo que esta tardó.
Se estudiaron los resultados de las probabilidades de error tipo I simuladas en paralelo.
Por otro lado, a partir de los tiempos obtenidos, se analizó la aceleración y eficiencia de los programas.

- Simulaciones con variación de componentes.
  En esta etapa de la investigación se comenzó por utilizar las mismas muestras de la etapa anterior, pero en esta ocasión se simuló para la aplicación del test considerando $(a_1,a_2) \in \{(1,0),(1,1)\}$ en todos los casos. Las simulaciones involucradas se efectuaron haciendo trabajar desde 1 hasta 12 núcleos en paralelo (parRapply y boot).
  A continuación se consideró variar el tamaño muestral $n \in \{100, 150, 200, 250, 300, 500\}$ de modo de conocer la eficiencia del test para muestras mayores a las ya estudiadas, pero sin considerar variación de los parámetros poblacionales, respecto de los anteriores. Estas simulaciones se efectuaron únicamente haciendo uso de la paralelización del proceso en 10 núcleos (parRapply).
  Para continuar con la variación de componentes de las simulaciones, y en particular, para conocer la eficiencia del test para vectores de parámetros que den cuenta de un coeficiente de correlación $\rho = \theta_3/\sqrt{\theta_1\,\theta_2}$ menor a 0.25 o mayor a 0.75, es que efectuaron las simulaciones considerado
  $(\theta_1, \theta_2, \theta_3) \in$\{ (1, 1, 0.1), (1, 1, 0.2), (1.5, 1, 0.12), (1.5, 1, 0.25), (1, 1, 0.8), (1, 1, 0.9), (1.5, 1, 0.98), (??) \},
  $n \in \{30, 50, 70, 100, 150, 200, 250, 300, 500\}$
  y $(a_1, a_2) \in \{(0,0),(0,1),(1,0),(1,1)\}$
  todo esto haciendo uso de la paralelización en 10 núcleos proporcionada por el comando parRapply.

Es importante señalar que las simulaciones descritas en la segunda y tercera etapa de esta investigación se realizaron en un cluster que consta de 12 procesadores Intel modelo:

Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz

Con las siguientes características:

cpu MHz : 1199.951 cache size : 15360 KB RAM : 32GB

El sistema operativo que utiliza es GNU/Linux, distribución Centos 7.

Capítulo 4

Resultados numéricos

4.1. Resultados de simulación de la probabilidad de Error Tipo I del primer test de Novoa - Jiménez, considerando los mismos componentes utilizados por los investigadores.

La primera parte del proceso de simulación de esta investigación, consideró los mismos componentes estudiados por Novoa y Jiménez (2014), de lo que se obtiene, en primer lugar, los valores de la probabilidad de error simulada para cada uno de los casos estudiados. Al no haber variación significativa de estos valores (ver anexos), respecto de los valores nominales, se procede a analizar los tiempos de cómputo, así como la aceleración y eficiencia de los programas implementados en paralelo, tanto con el uso del comando *parRapply* como con el comado *boot*.

A continuación se muestran las tablas de los tiempos de simulación obtenidos para los distintos vectores de parámetros, para los cuales en primer lugar se presentan los resultados al utilizar el comando *parRapply* y luego los resultados al aplicar el comando *boot*. Desde la tabla 4.1 hasta la tabla 4.6 se muestran los tiempos para los casos en que $E(X_1) \neq E(X_2)$ y luego, desde la tabla 4.7 hasta la tabla 4.12 se muestran los tiempos de simulación para los casos en que $E(X_1) = E(X_2)$.

Tabla 4.1: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **parRapply**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1.5, 1, 0.31)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2) utilizados, para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8209	8307	8175	8146
2	4049	4058	4063	4088
3	2764	2762	2801	2793
4	2118	2100	2092	2112
5	1726	1761	1713	1724
6	1445	1457	1444	1461
7	1242	1246	1257	1263
8	1102	1125	1123	1090
9	1007	1007	1012	1012
10	919	916	914	918
11	841	840	835	845
12	764	765	764	776

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8907	8925	8913	8914
2	4402	4428	4413	4440
3	2959	2979	2970	3006
4	2261	2259	2255	2248
5	1868	1888	1874	1870
6	1575	1551	1601	1567
7	1365	1350	1361	1356
8	1203	1193	1191	1196
9	1105	1099	1112	1097
10	988	981	976	986
11	905	931	913	933
12	823	820	820	823

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	9427	9451	9398	9400
2	4657	4662	4663	4674
3	3164	3142	3164	3162
4	2406	2399	2389	2404
5	2004	1970	1988	1969
6	1679	1658	1640	1655
7	1432	1457	1420	1447
8	1288	1261	1274	1264
9	1158	1162	1160	1171
10	1044	1039	1055	1037
11	963	975	960	981
12	869	875	882	877

Tabla 4.2: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **boot**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1.5, 1, 0.31)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2) utilizados, para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8313	8335	8326	8343
2	4359	4377	4338	4351
3	3089	3078	3078	3079
4	2368	2368	2367	2380
5	2003	2010	1998	2010
6	1701	1707	1710	1706
7	1543	1533	1532	1530
8	1366	1360	1362	1362
9	1280	1270	1281	1280
10	1161	1177	1162	1169
11	1098	1088	1095	1099
12	1025	1020	1022	1027

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	9028	9138	8991	9029
2	4645	4660	4672	4650
3	3271	3280	3269	3258
4	2496	2489	2489	2478
5	2110	2114	2098	2107
6	1775	1772	1783	1771
7	1598	1602	1600	1598
8	1395	1400	1403	1393
9	1322	1322	1328	1328
10	1194	1193	1203	1196
11	1126	1125	1119	1126
12	1027	1029	1040	1034

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	9653	9553	9573	9587
2	4936	4943	4941	4948
3	3453	3454	3465	3443
4	2631	2626	2626	2605
5	2229	2225	2219	2209
6	1875	1852	1861	1863
7	1670	1670	1666	1666
8	1465	1471	1465	1476
9	1376	1379	1377	1382
10	1260	1255	1261	1254
11	1173	1175	1170	1178
12	1074	1072	1058	1071

Tabla 4.3: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **parRapply**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1.5, 1, 0.62)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2) utilizados, para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7937	7895	7969	7878
2	3897	3879	3969	3890
3	2658	2632	2626	2646
4	2037	2011	2015	2005
5	1669	1638	1672	1673
6	1391	1392	1375	1392
7	1200	1195	1205	1201
8	1064	1046	1053	1001
9	974	976	970	970
10	867	873	877	874
11	819	806	807	815
12	728	737	730	738

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8630	8624	8495	8584
2	4257	4316	4291	4261
3	2889	2886	2915	2909
4	2178	2208	2195	2186
5	1797	1819	1823	1809
6	1495	1514	1524	1516
7	1311	1307	1317	1326
8	1149	1156	1149	1151
9	1058	1066	1059	1071
10	973	948	972	950
11	873	886	876	886
12	820	807	806	810

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	9174	9097	9382	9210
2	4529	4544	4530	4564
3	3107	3109	3091	3093
4	2330	2322	2343	2344
5	1925	1934	1949	1927
6	1609	1610	1610	1615
7	1388	1396	1380	1399
8	1218	1236	1212	1224
9	1117	1135	1124	1131
10	1021	1004	1016	1023
11	929	959	959	948
12	859	861	853	858

Tabla 4.4: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **boot**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1.5, 1, 0.62)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2) utilizados, para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7977	7926	7933	7952
2	4133	4111	4110	4110
3	2883	2874	2891	2875
4	2184	2184	2188	2181
5	1857	1854	1853	1854
6	1555	1559	1554	1553
7	1403	1397	1401	1399
8	1223	1226	1223	1225
9	1147	1144	1145	1145
10	1037	1036	1036	1037
11	972	973	968	972
12	892	893	895	898

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8823	8789	8768	8782
2	4542	4552	4529	4530
3	3169	3175	3162	3160
4	2394	2396	2388	2394
5	2034	2031	2027	2033
6	1697	1696	1697	1696
7	1528	1526	1524	1525
8	1335	1333	1335	1333
9	1259	1258	1256	1256
10	1132	1131	1132	1132
11	1058	1056	1055	1056
12	973	971	962	973

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	9415	9403	9500	9391
2	4840	4836	4844	4833
3	3381	3372	3395	3373
4	2549	2547	2564	2550
5	2163	2155	2174	2158
6	1810	1798	1810	1801
7	1621	1619	1622	1622
8	1420	1411	1419	1410
9	1340	1338	1341	1339
10	1211	1208	1212	1209
11	1133	1132	1133	1129
12	1035	1024	1032	1024

Tabla 4.5: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **parRapply**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1.5, 1, 0.92)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7995	7976	8051	8004
2	3970	3971	4000	4004
3	2740	2713	2701	2708
4	2064	2049	2028	2042
5	1696	1677	1668	1686
6	1403	1411	1399	1438
7	1239	1221	1235	1224
8	1087	1059	1079	1076
9	981	986	998	989
10	896	881	886	882
11	832	819	841	820
12	751	742	753	751

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8804	8886	8819	8713
2	4351	4331	4355	4326
3	2956	2963	2969	2947
4	2250	2251	2253	2251
5	1841	1865	1855	1845
6	1551	1549	1538	1537
7	1344	1339	1329	1328
8	1191	1172	1174	1192
9	1086	1080	1083	1086
10	983	975	972	989
11	906	893	884	912
12	820	823	825	823

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	9398	9287	9270	9446
2	4639	4619	4654	4599
3	3130	3178	3106	3130
4	2369	2391	2349	2354
5	1978	1962	1959	1973
6	1645	1630	1620	1617
7	1436	1414	1419	1409
8	1243	1238	1234	1234
9	1162	1144	1139	1138
10	1034	1037	1027	1030
11	948	943	944	948
12	866	868	873	871

Tabla 4.6: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **boot**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1.5, 1, 0.92)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7910	7865	7885	7889
2	4085	4083	4083	4082
3	2860	2850	2847	2856
4	2155	2159	2169	2155
5	1836	1838	1837	1835
6	1534	1530	1536	1527
7	1381	1382	1380	1381
8	1202	1203	1203	1206
9	1121	1125	1122	1126
10	1016	1018	1018	1018
11	956	955	955	954
12	876	881	886	877

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8765	8812	8776	8817
2	8828	4551	4539	4535
3	3185	3175	3165	3166
4	2407	2396	2391	2401
5	2044	2034	2032	2035
6	1704	1700	1698	1698
7	1533	1531	1526	1528
8	1339	1337	1333	1333
9	1264	1264	1262	1260
10	1133	1132	1133	1135
11	1060	539	1058	1058
12	975	976	965	966

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	9441	9362	9393	9492
2	4836	4839	4839	4830
3	3374	3372	3396	3376
4	2556	2543	2547	2547
5	2166	2156	2164	2164
6	1804	1806	1808	1811
7	1622	1622	1625	1626
8	1414	1417	1418	1413
9	1341	1341	1342	1339
10	1210	1210	1214	1208
11	1136	1134	1135	1134
12	1028	1028	1027	1027

Tabla 4.7: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **parRapply**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1, 1, 0.5)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	6923	7114	7006	6985
2	3457	3439	3509	3475
3	2341	2330	2326	2350
4	1782	1769	1769	1758
5	1466	1479	1468	1491
6	1222	1228	1222	1232
7	1070	1075	1063	1061
8	941	938	942	931
9	848	849	849	858
10	777	808	781	774
11	715	703	725	723
12	656	654	655	660

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7554	7556	7513	7511
2	3727	3759	3767	3771
3	2498	2522	2539	2517
4	1929	1931	1914	1930
5	1591	1595	1602	1590
6	1326	1335	1325	1329
7	1150	1156	1138	1146
8	1003	1004	1015	1013
9	936	924	923	930
10	839	834	835	839
11	773	768	773	774
12	709	706	710	712

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8008	8086	8055	8109
2	3960	3978	3952	3979
3	2679	2721	2676	2704
4	2053	2037	2031	2034
5	1679	1673	1690	1680
6	1415	1399	1413	1409
7	1229	1211	1210	1213
8	1075	1070	1072	1074
9	976	1011	987	1016
10	888	877	877	899
11	813	821	823	823
12	755	761	743	749

Tabla 4.8: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **boot**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1, 1, 0.5)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7073	7042	7070	7059
2	3665	3673	3670	3682
3	2577	2572	2589	2586
4	1955	1947	1959	1961
5	1668	1663	1662	1667
6	1399	1400	1394	1399
7	1263	1260	1262	1256
8	1091	1092	1093	1092
9	1014	1014	1019	1015
10	938	935	933	932
11	875	878	880	877
12	809	810	809	810

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7752	7687	7678	7690
2	4000	3983	3988	3992
3	2792	2786	2799	2806
4	2109	2103	2113	2110
5	1794	1790	1797	1795
6	1497	1498	1497	1498
7	1351	1353	1355	1352
8	1175	1170	1172	1174
9	1093	1091	1095	1095
10	989	989	988	989
11	931	929	929	930
12	859	862	861	860

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8256	8239	8267	8237
2	4249	4255	4238	4241
3	2971	2984	2966	2967
4	2233	2247	2229	2235
5	1901	1908	1898	1903
6	1584	1591	1590	1594
7	1431	1432	1431	1433
8	1254	1250	1250	1252
9	1172	1172	1175	1169
10	1049	1048	1052	1050
11	985	986	988	986
12	904	904	913	911

Tabla 4.9: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **parRapply**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1, 1, 0.25)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7451	7257	7247	7317
2	3589	3605	3583	3587
3	2436	2452	2442	2458
4	1844	1851	1859	1879
5	1528	1563	1545	1537
6	1280	1296	1277	1289
7	1125	1114	1111	1136
8	988	972	980	974
9	890	892	897	892
10	811	812	810	799
11	748	746	751	742
12	689	678	687	685

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7856	7795	7854	7778
2	3925	3975	3867	3904
3	2660	2628	2630	2667
4	1998	1991	2018	1993
5	1667	1669	1652	1648
6	1375	1384	1382	1379
7	1200	1194	1203	1223
8	1062	1043	1049	1059
9	960	957	968	958
10	874	880	870	872
11	797	821	799	794
12	745	736	738	739

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8264	8359	8270	8220
2	4113	4063	4154	4105
3	2792	2837	2796	2803
4	2105	2103	2139	2106
5	1760	1762	1748	1745
6	1445	1468	1449	1465
7	1260	1267	1253	1260
8	1108	1100	1109	1111
9	1054	1014	1038	1017
10	914	925	910	921
11	846	850	855	846
12	771	786	773	779

Tabla 4.10: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **boot**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1, 1, 0.25)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7526	7536	7472	7485
2	3968	3950	3972	3954
3	2830	2809	2812	2813
4	2175	2166	2169	2164
5	1848	1845	1853	1856
6	1586	1570	1570	1580
7	1416	1421	1419	1424
8	1255	1251	1252	1256
9	1171	1180	1181	1178
10	1082	1086	1087	1084
11	1028	1020	1019	1025
12	954	952	959	952

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7905	7960	8013	7993
2	4136	4150	4142	4141
3	2923	2913	2922	2907
4	2215	2222	2229	2222
5	1892	1892	1893	1888
6	1590	1590	1595	1586
7	1430	1438	1443	1434
8	1255	1261	1260	1255
9	1169	1174	1176	1176
10	1069	1070	1068	1074
11	1003	1008	1007	1002
12	931	935	925	927

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8468	8380	8429	8385
2	4368	4354	4384	4386
3	3069	3086	3058	3065
4	2319	2328	2333	2323
5	1972	1980	1972	1972
6	1663	1659	1651	1664
7	1487	1484	1480	1483
8	1303	1299	1302	1310
9	1222	1224	1226	1219
10	1100	1096	1099	1102
11	1028	1031	1032	1027
12	950	952	946	953

Tabla 4.11: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **parRapply**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1, 1, 0.75)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	6851	6881	6892	6912
2	3383	3426	3400	3418
3	2321	2293	2321	2296
4	1742	1755	1752	1747
5	1478	1451	1462	1444
6	1204	1205	1214	1217
7	1053	1040	1036	1047
8	907	927	916	927
9	850	844	847	852
10	755	767	762	767
11	701	708	713	699
12	643	647	648	645

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7558	7432	7657	7461
2	3731	3716	3725	3733
3	2511	2543	2525	2566
4	1899	1910	1931	1918
5	1610	1585	1599	1603
6	1320	1315	1338	1322
7	1150	1146	1145	1147
8	1002	1009	1009	1004
9	931	921	920	920
10	831	831	843	842
11	775	762	770	772
12	711	713	700	707

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8108	8054	7952	8019
2	3946	3970	3981	3963
3	2697	2690	2705	2683
4	2064	2021	2031	2063
5	1701	1693	1702	1700
6	1410	1405	1416	1408
7	1238	1234	1225	1227
8	1077	1083	1090	1062
9	974	988	978	992
10	892	897	894	885
11	810	801	805	812
12	759	750	759	753

Tabla 4.12: Tiempo (seg) de cómputo del programa en paralelo, haciendo uso del comando **boot**, al aplicarlo a muestras provenientes de una población con vector de parámetros $\theta =$ **(1, 1, 0.75)**. Se muestran los resultados según el número de núcleos (p) y el vector de pesos (a_1, a_2), para distintos tamaños muestrales (n).

$n = 30$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	6885	6868	6907	6962
2	3582	3601	3601	3596
3	2533	2524	2539	2532
4	1904	1900	1905	1900
5	1625	1623	1621	1622
6	1358	1356	1353	1351
7	1220	1219	1218	1218
8	1053	1050	1049	1052
9	976	978	978	979
10	895	895	894	896
11	843	844	843	846
12	782	777	775	784

$n = 50$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	7720	7688	7658	7691
2	3957	3978	3983	3966
3	2789	2790	2795	2783
4	2105	2099	2095	2088
5	1791	1791	1789	1785
6	1486	1492	1491	1484
7	1346	1347	1345	1347
8	1169	1166	1166	1161
9	1086	1087	1087	1084
10	983	984	985	982
11	927	925	925	924
12	856	849	860	857

$n = 70$				
	(a_1, a_2)			
p	(0, 0)	(0, 1)	(1, 0)	(1, 1)
1	8207	8164	8231	8243
2	4226	4631	4244	4254
3	2963	2957	2966	2972
4	2228	2335	2229	2241
5	1901	1904	1898	1897
6	1588	1589	1585	1585
7	1428	1432	1427	1427
8	1245	1249	1249	1247
9	1169	1170	1170	1169
10	1048	1050	1049	1051
11	984	986	986	985
12	903	911	914	906

Como era de esperar, los tiempos disminuyen en la medida que aumenta el número de núcleos utilizados. A su vez, los tiempos aumentan en la medida que aumenta el tamaño muestral.

Considerando que los resultados de las simulaciones con los distintos vectores de pesos no presentan diferencias significativas entre sí, es que a continuación se muestran los gráficos de aceleración y eficiencia de los programas en paralelo al utilizar $a_1 = 0$ y $a_2 = 0$ en la función de peso.

Es preciso recordar que la aceleración del programa en paralelo se mide en base al tiempo de cómputo del programa secuencial, de esta manera, la aceleración de una ejecución particular del programa indicará cuántas veces más rápido es que la ejecución secuencial.

Desde la figura 4.1 hasta la 4.3 se presentan las gráficas de aceleración de los programas para el caso $E(X_1) \neq E(X_2)$. Cada figura muestra dos gráficas, de las cuales, una corresponde a la aceleración usando el comando *parRapply* y la otra, a la aceleración al usar el comando *boot*. De la misma manera, desde la figura 4.4 hasta la figura 4.6 se muestran las gráficas correspondientes a los casos en que $E(X_1) = E(X_2)$.

Figura 4.1: Aceleración de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vector de parámetros θ =(1.5, 1, 0,31), $\rho \approx 0.25$

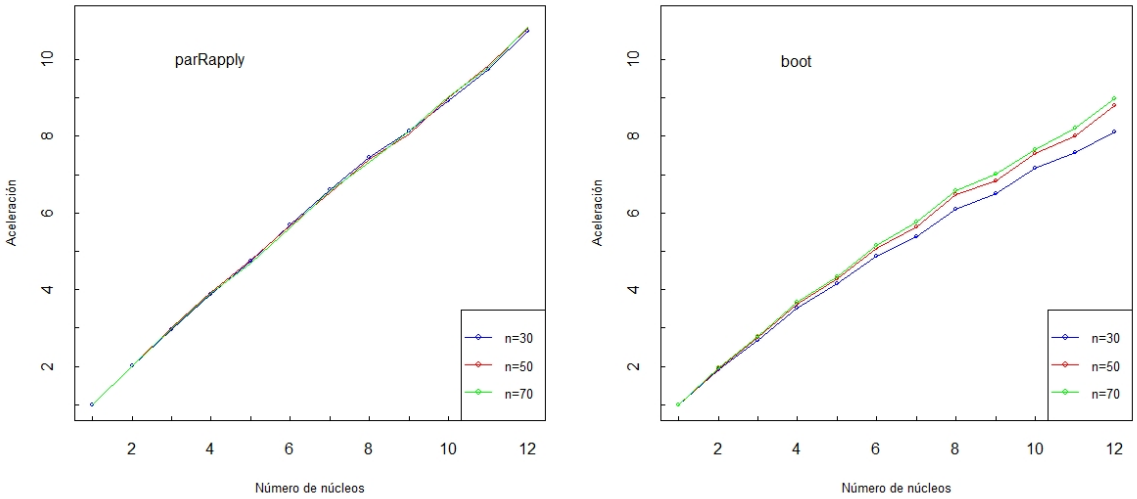


Figura 4.2: Aceleración de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vector de parámetros θ =(1.5, 1, 0,62), $\rho \approx 0.5$.

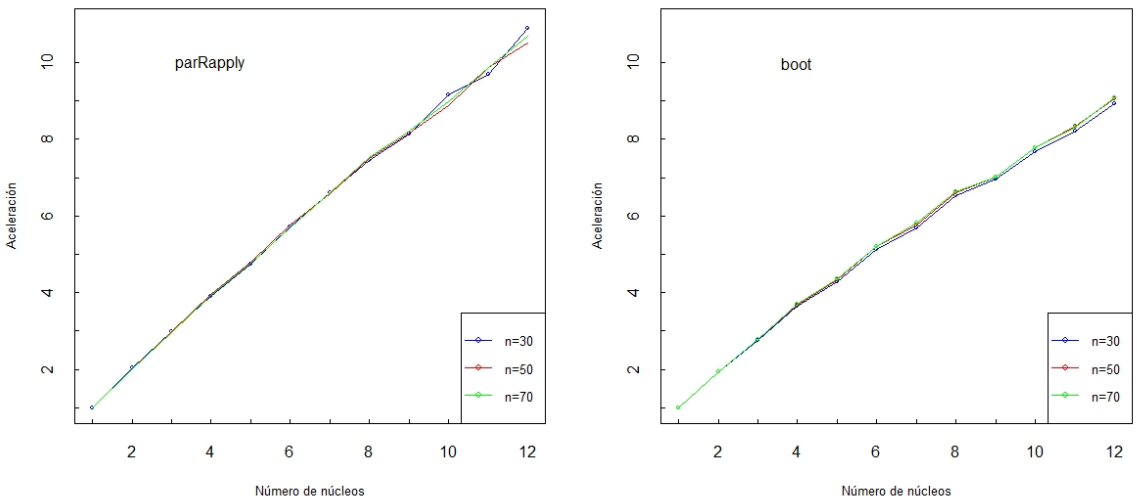


Figura 4.3: Aceleración de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vector de parámetros θ =(1.5, 1, 0,92), $\rho \approx 0.75$.

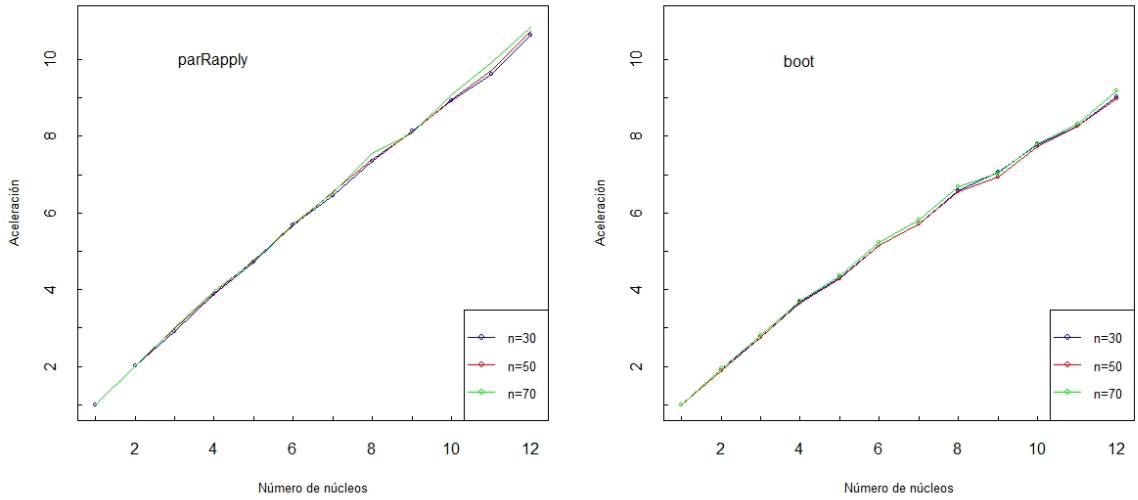


Figura 4.4: Aceleración de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vector de parámetros θ =(1, 1, 0,5), $\rho = 0.5$.

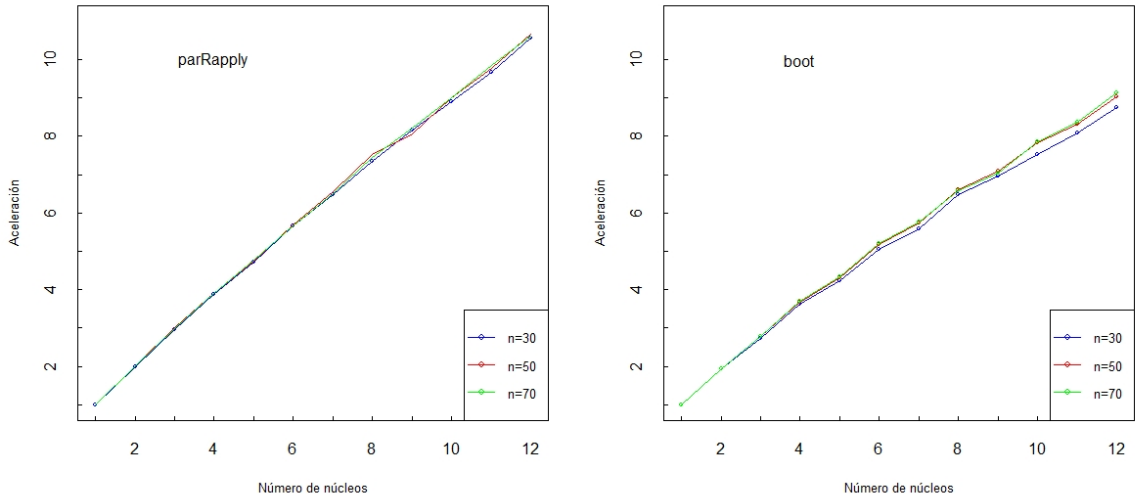


Figura 4.5: Aceleración de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vector de parámetros θ =(1, 1, 0,25), $\rho = 0.25$.

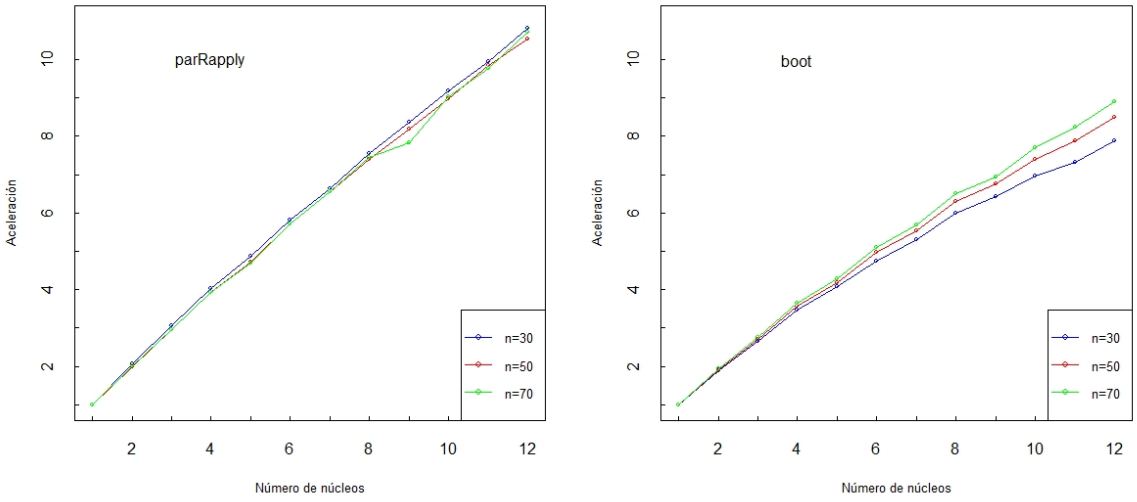
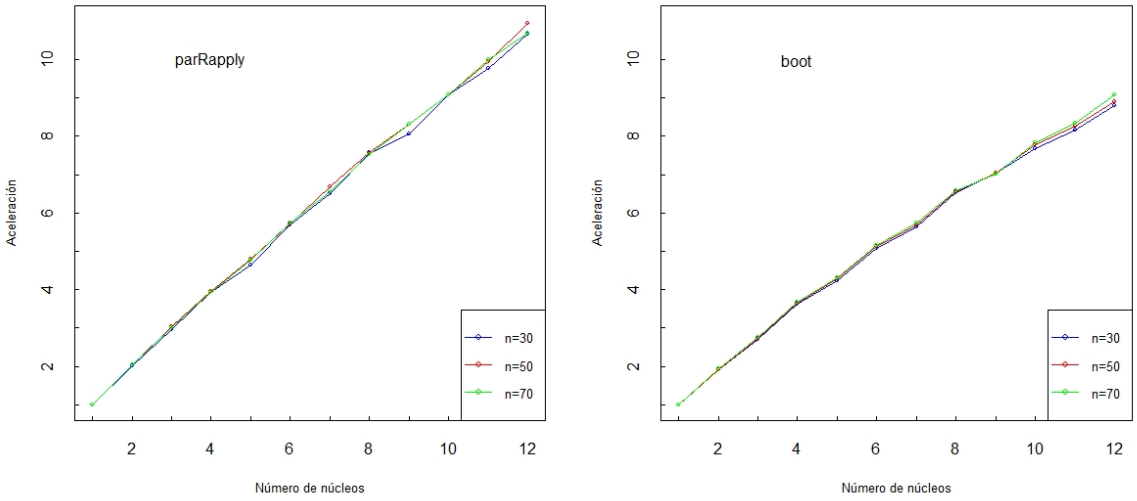


Figura 4.6: Aceleración de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vector de parámetros θ =(1, 1, 0,75), $\rho = 0.75$.



De las figuras 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6 se observa que al hacer uso del comando *parRapply* la aceleración del programa tiene un comportamiento casi lineal, donde independientemente del vector de parámetros utilizado, al hacer uso de los 12 núcleos, se logra un tiempo al menos 10 veces menor que el secuencial.

Por otro lado, se puede apreciar que al utilizar el comando *parRapply* la aceleración es mayor que al utilizar el comando *boot* para todos los tamaños muetrales (n) y vectores de parámetros, a partir del uso de 4 núcleos en adelante.

A continuación se presentan los gráficos de eficiencia de los programas. Desde la figura 4.7 hasta la 4.9 se presentan las gráficas de eficiencia de los programas para el caso $E(X_1) \neq E(X_2)$. Cada figura muestra dos gráficas, de las cuales, una corresponde a la eficiencia usando el comando *parRapply* y la otra, a la eficiencia al usar el comando *boot*. De la misma manera, desde la figura 4.10 hasta la figura 4.12 se muestran las gráficas correspondientes a los casos en que $E(X_1) = E(X_2)$.

Figura 4.7: Eficiencia de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vectores de parámetros θ =(1.5, 1, 0,31), $\rho \approx 0.25$.

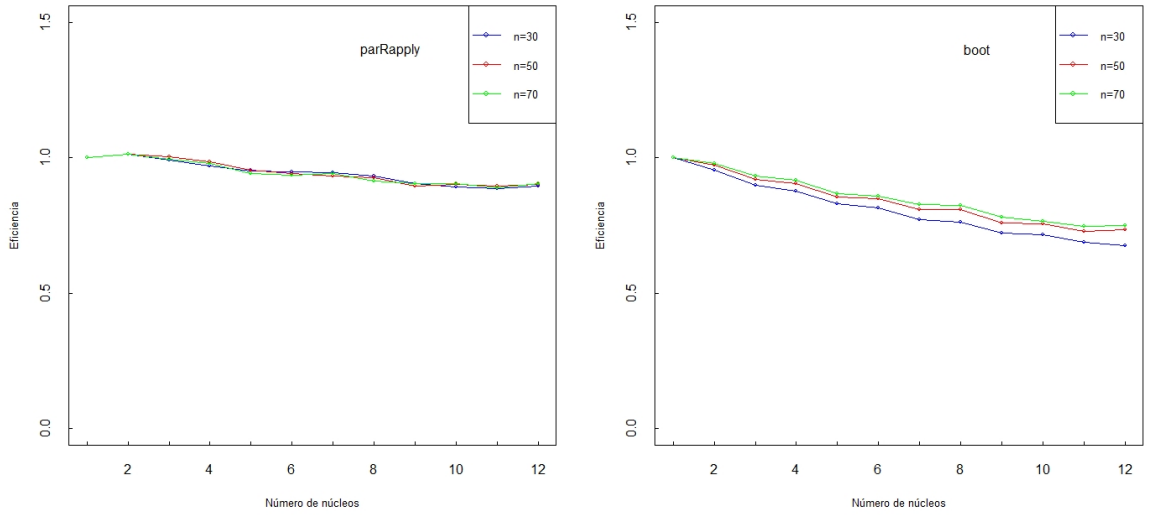


Figura 4.8: Eficiencia de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vectores de parámetros θ =(1.5, 1, 0,62), $\rho \approx 0.5$.

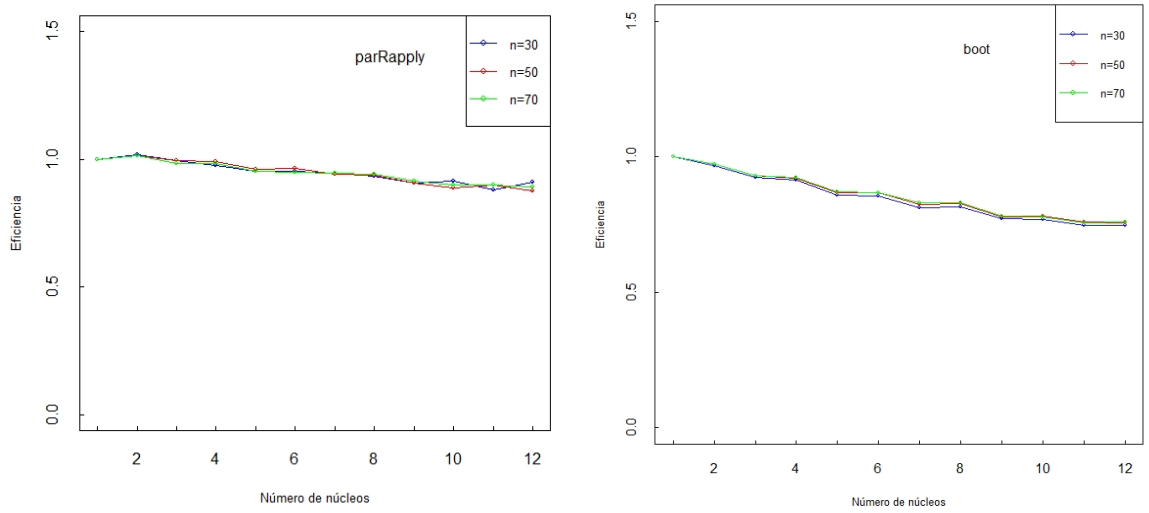


Figura 4.9: Eficiencia de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vectores de parámetros θ =(1.5, 1, 0,92), $\rho \approx 0.75$.

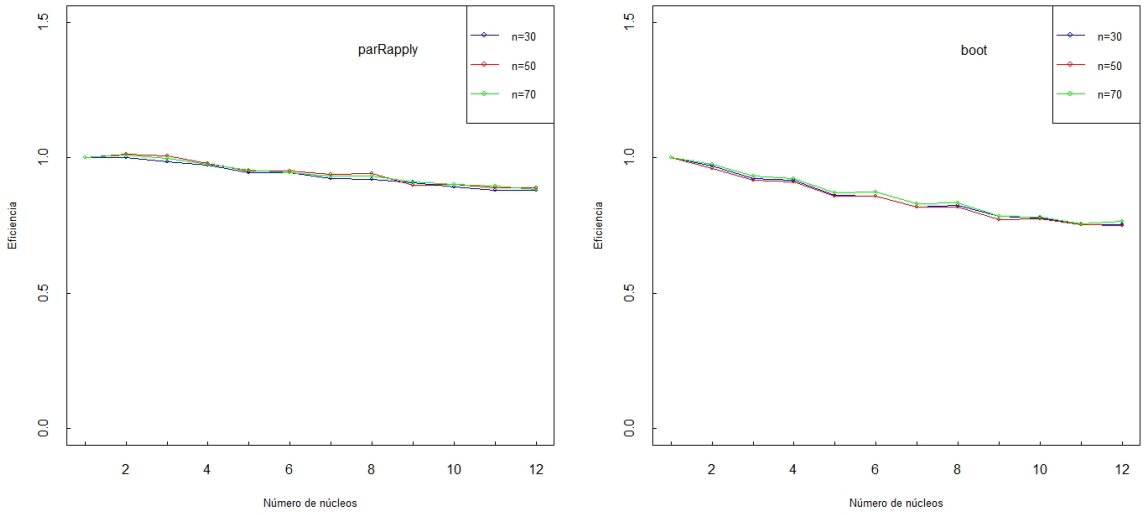


Figura 4.10: Eficiencia de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vectores de parámetros θ =(1, 1, 0,5), $\rho = 0.5$.

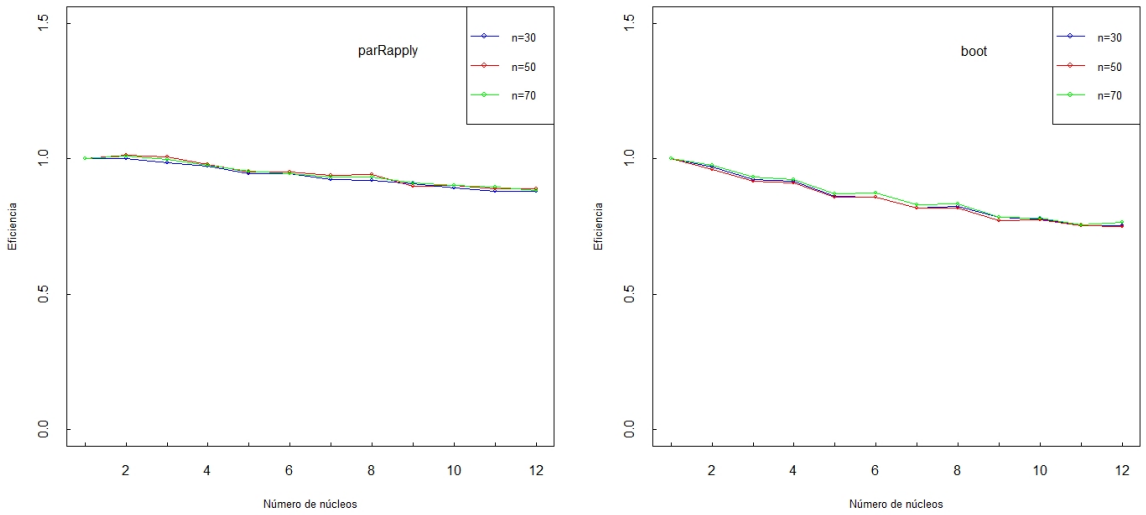


Figura 4.11: Eficiencia de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vectores de parámetros θ =(1, 1, 0,25), $\rho = 0.25$..

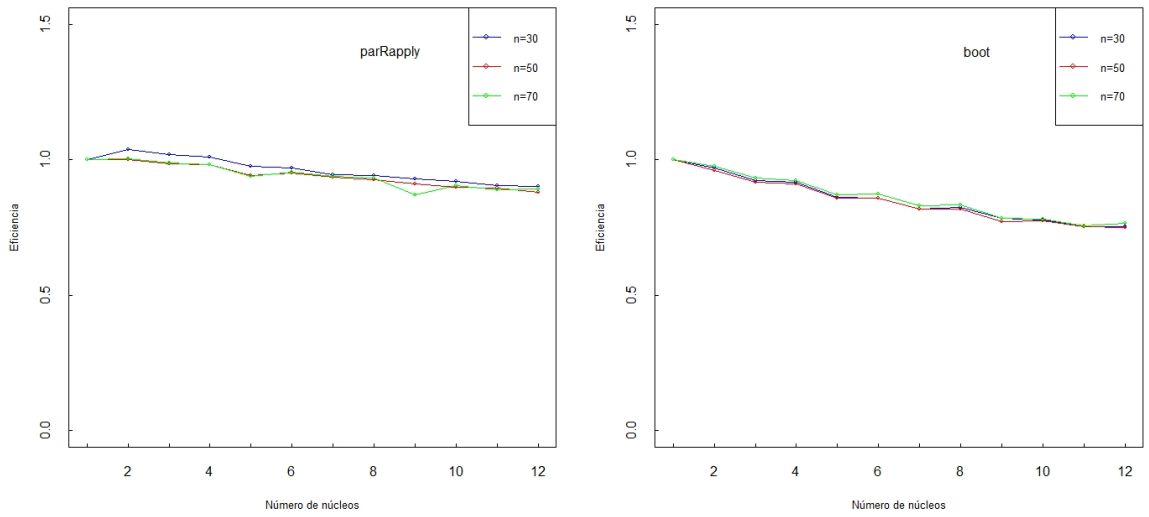
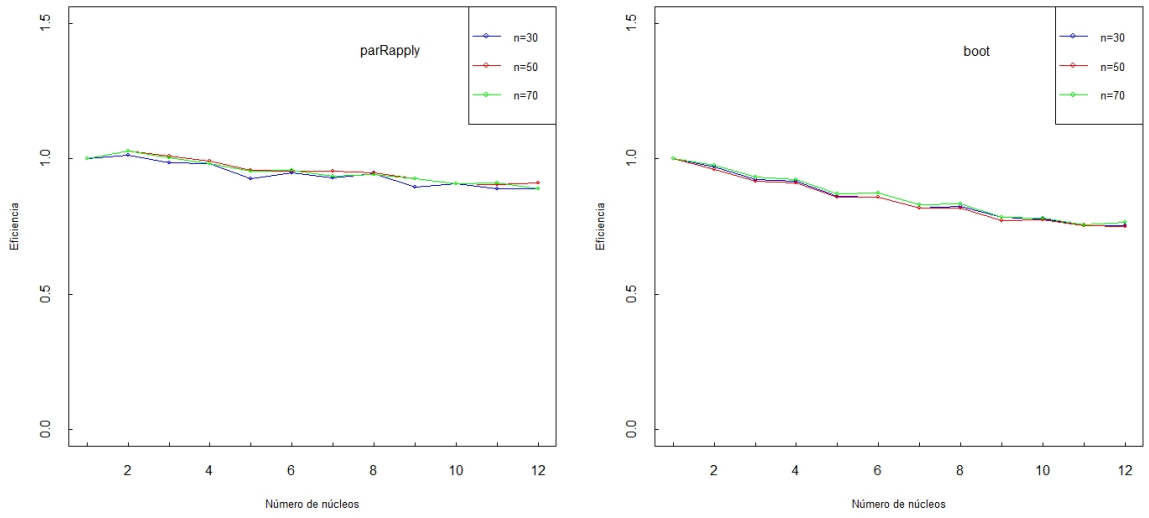


Figura 4.12: Eficiencia de los programas en paralelo según el número de núcleos utilizados, para distintos tamaños muestrales (n) y vectores de parámetros θ =(1, 1, 0,75), $\rho = 0.75$.



Se puede observar en las figuras 4.7, 4.8, 4.9, 4.10, 4.11 y 4.12 que la eficiencia del programa en paralelo disminuye en la medida que aumenta el número de núcleos utilizados, tanto con el uso del comando *boot* como con el uso del comando *parRapply*. Sin embargo, la disminución es lenta y se mantiene cercana al valor 1, que es el máximo valor que puede tomar la eficiencia de un programa.

No obstante lo anterior, para todos los vectores de parámetros se observa que la eficiencia es mayor haciendo uso del comando *parRapply*, en comparación con los programas que utilizan el comando *boot*.

4.2. Resultados de simulación de la probabilidad de Error Tipo I del primer test de Novoa - Jiménez, considerando tamaños muestrales mayores que los estudiados por los investigadores.

Conociendo la imposibilidad de los autores del test R de simular la probabilidad de error tipo I para tamaños muestrales mayores a $n = 70$, es que luego de haber evaluado las alternativas de paralelización se procede a simular haciendo uso del comando *parRapply* con 10 núcleos.

En primer lugar se obtienen las probabilidades de error tipo I simuladas para los distintos tamaños muestrales estudiados. Se dan a conocer también los tiempos, medidos en segundos, que tardaron cada una de estas simulaciones.

Tabla 4.13: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.31)$, $\rho \approx 0.25$.

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
100	0.010	0.044	0.099	0.014	0.053	0.104	0.008	0.049	0.089	0.009	0.048	0.104
150	0.010	0.050	0.098	0.006	0.053	0.097	0.013	0.057	0.100	0.004	0.048	0.098
200	0.009	0.036	0.094	0.008	0.043	0.090	0.007	0.051	0.099	0.008	0.043	0.090
250	0.017	0.066	0.112	0.019	0.064	0.121	0.015	0.056	0.105	0.018	0.062	0.114
300	0.014	0.055	0.091	0.014	0.046	0.088	0.014	0.060	0.112	0.015	0.048	0.094
500	0.016	0.054	0.108	0.014	0.058	0.112	0.012	0.049	0.107	0.015	0.058	0.105

Tabla 4.14: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.62)$, $\rho \approx 0.5$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
100	0.010	0.060	0.116	0.012	0.054	0.115	0.010	0.061	0.108	0.009	0.054	0.111
150	0.012	0.063	0.129	0.015	0.068	0.130	0.014	0.067	0.122	0.011	0.067	0.129
200	0.007	0.038	0.080	0.009	0.039	0.093	0.004	0.037	0.094	0.006	0.040	0.085
250	0.008	0.046	0.094	0.009	0.049	0.103	0.007	0.042	0.089	0.009	0.041	0.094
300	0.016	0.058	0.103	0.014	0.059	0.108	0.013	0.055	0.100	0.010	0.055	0.101
500	0.008	0.035	0.079	0.012	0.036	0.081	0.006	0.043	0.077	0.010	0.035	0.079

Tabla 4.15: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.92)$, $\rho \approx 0.75$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
100	0.007	0.039	0.088	0.008	0.037	0.090	0.009	0.043	0.094	0.006	0.035	0.091
150	0.011	0.066	0.121	0.012	0.054	0.131	0.011	0.067	0.123	0.011	0.056	0.118
200	0.017	0.060	0.110	0.021	0.052	0.101	0.016	0.057	0.110	0.018	0.057	0.106
250	0.019	0.053	0.111	0.021	0.059	0.111	0.011	0.054	0.098	0.022	0.055	0.098
300	0.009	0.054	0.111	0.013	0.053	0.110	0.011	0.060	0.113	0.012	0.062	0.111
500	0.013	0.056	0.099	0.017	0.055	0.107	0.015	0.053	0.111	0.016	0.051	0.110

Tabla 4.16: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.5)$, $\rho = 0.5$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
100	0.009	0.054	0.092	0.009	0.053	0.099	0.015	0.048	0.087	0.009	0.052	0.085
150	0.017	0.051	0.109	0.009	0.055	0.107	0.017	0.061	0.108	0.012	0.054	0.096
200	0.013	0.068	0.116	0.011	0.055	0.115	0.018	0.070	0.114	0.013	0.064	0.114
250	0.009	0.047	0.095	0.009	0.050	0.093	0.014	0.038	0.096	0.010	0.050	0.100
300	0.018	0.048	0.098	0.012	0.048	0.099	0.022	0.049	0.095	0.015	0.050	0.092
500	0.018	0.053	0.102	0.014	0.056	0.100	0.015	0.046	0.098	0.013	0.056	0.100

Tabla 4.17: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.25)$, $\rho = 0.25$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
100	0.006	0.052	0.100	0.008	0.051	0.109	0.011	0.042	0.105	0.009	0.046	0.096
150	0.011	0.047	0.097	0.012	0.052	0.101	0.007	0.047	0.099	0.008	0.043	0.093
200	0.017	0.057	0.090	0.011	0.044	0.091	0.018	0.065	0.110	0.018	0.053	0.102
250	0.013	0.056	0.108	0.009	0.051	0.097	0.013	0.068	0.124	0.012	0.056	0.115
300	0.010	0.057	0.113	0.013	0.066	0.118	0.011	0.049	0.106	0.011	0.057	0.111
500	0.009	0.049	0.100	0.011	0.056	0.106	0.005	0.046	0.098	0.010	0.048	0.100

Tabla 4.18: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.75)$, $\rho = 0.75$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
100	0.009	0.039	0.086	0.013	0.042	0.088	0.007	0.044	0.102	0.010	0.040	0.093
150	0.013	0.054	0.111	0.009	0.048	0.107	0.010	0.054	0.109	0.010	0.052	0.114
200	0.013	0.053	0.112	0.014	0.058	0.108	0.014	0.056	0.115	0.015	0.060	0.113
250	0.016	0.060	0.109	0.014	0.065	0.112	0.014	0.057	0.115	0.014	0.060	0.108
300	0.013	0.051	0.101	0.013	0.050	0.089	0.014	0.052	0.100	0.011	0.048	0.101
500	0.010	0.045	0.092	0.008	0.050	0.096	0.011	0.046	0.097	0.010	0.047	0.090

Tabla 4.19: Tiempo de simulación de probabilidad de error tipo I, para vector de parámetros $\theta = \mathbf{(1.5,\ 1,\ 0.31)}$, $\rho \approx 0.25$.

n	$a_1 = 0,\ a_2 = 0$	$a_1 = 0,\ a_2 = 1$	$a_1 = 1,\ a_2 = 0$	$a_1 = 1,\ a_2 = 1$
100	1103	1118	1108	1112
150	1212	1192	1191	1197
200	1281	1272	1288	1275
250	1360	1353	1372	1354
300	1422	1417	1431	1411
500	1665	1666	1669	1685

Tabla 4.20: Tiempo de simulación de probabilidad de error tipo I, para vector de parámetros $\theta = \mathbf{(1.5,\ 1,\ 0.62)}$, $\rho \approx 0.5$.

n	$a_1 = 0,\ a_2 = 0$	$a_1 = 0,\ a_2 = 1$	$a_1 = 1,\ a_2 = 0$	$a_1 = 1,\ a_2 = 1$
100	1108	1073	1077	1090
150	1164	1175	1189	1182
200	1276	1276	1253	1256
250	1330	1325	1327	1341
300	1414	1396	1388	1399
500	1648	1628	1633	1633

Tabla 4.21: Tiempo de simulación de probabilidad de error tipo I, para vector de parámetros $\theta = \mathbf{(1.5,\ 1,\ 0.92)}$, $\rho \approx 0.75$.

n	$a_1 = 0,\ a_2 = 0$	$a_1 = 0,\ a_2 = 1$	$a_1 = 1,\ a_2 = 0$	$a_1 = 1,\ a_2 = 1$
100	1100	1094	1093	1099
150	1182	1203	1198	1174
200	1260	1268	1265	1262
250	1343	1329	1339	1341
300	1408	1398	1411	1409
500	1645	1633	1643	1643

Tabla 4.22: Tiempo de simulación de probabilidad de error tipo I, para vector de parámetros $\theta =$ **(1, 1, 0.5)**, $\rho = 0.5$.

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
100	935	936	949	947
150	1022	1007	1015	1013
200	1083	1068	1083	1078
250	1141	1143	1140	1147
300	1210	1197	1191	1198
500	1394	1414	1405	1413

Tabla 4.23: Tiempo de simulación de probabilidad de error tipo I, para vector de parámetros $\theta =$ **(1, 1, 0.25)**, $\rho = 0.25$.

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
100	965	976	967	974
150	1039	1054	1049	1048
200	1113	1115	1125	1130
250	1189	1182	1183	1192
300	1250	1232	1240	1250
500	1448	1436	1460	1463

Tabla 4.24: Tiempo de simulación de probabilidad de error tipo I, para vector de parámetros $\theta =$ **(1, 1, 0.75)**.

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
100	934	942	932	948
150	1015	1024	1016	1021
200	1078	1093	1102	1084
250	1157	1152	1155	1163
300	1220	1211	1221	1209
500	1424	1427	1419	1422

4.3. Resultados de simulación de la probabilidad de Error Tipo I del primer test de Novoa - Jiménez, considerando vectores de parámetros distintos a los estudiados por los investigadores.

La simulación del error tipo I implementada en paralelo permite evaluar el desempeño del test R al trabajar con vectores de parámetros distintos a los ya estudiados por los investigadores. En particular permite simular la probabilidad de error tipo I para los casos en los que el coeficiente de correlación ρ toma valores muy pequeños (0.1 y 0.2) o muy grandes (0.8 y 0.9). El primero de estos casos cobra especial relevancia ya que nos habla de parámetros que se encuentran muy cerca de la frontera del espacio paramétrico, lo que de acuerdo a la teoría [3] podría llevar a inconsistencias a la vez que eleva los tiempos de cómputo.

Se presentan los resultados de simulación de error tipo I para los vectores de parámetros que cumplen con las características ya señaladas, considerando nuevamente el caso $E(X_1) \neq E(X_2)$ y $E(X_1) = E(X_2)$. Se muestran estos resultados para tamaño muestral $n = 30$, 50, 70, 100, 150 y 200.

Tabla 4.25: Resultados de simulación de Error tipo I. Caso $\theta = (1.5, 1, 0.12)$, $\rho \approx 0.1$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
30	0.015	0.091	0.161	0.01	0.072	0.141	0.02	0.08	0.146	0.011	0.086	0.145
50	0.051	0.142	0.219	0.028	0.109	0.175	0.042	0.139	0.219	0.043	0.14	0.211
70	0.069	0.172	0.24	0.035	0.118	0.2	0.052	0.161	0.227	0.059	0.157	0.237
100	0.095	0.183	0.255	0.051	0.135	0.213	0.088	0.184	0.251	0.081	0.177	0.242
150	0.134	0.233	0.279	0.072	0.179	0.26	0.123	0.228	0.281	0.116	0.221	0.278
200	0.192	0.259	0.299	0.129	0.22	0.273	0.169	0.255	0.295	0.168	0.250	0.290

Tabla 4.26: Resultados de simulación de Error tipo I. Caso $\theta = (1.5, 1, 0.25)$, $\rho \approx 0.2$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
30	0.020	0.062	0.125	0.015	0.067	0.129	0.015	0.065	0.121	0.013	0.059	0.115
50	0.011	0.054	0.099	0.007	0.051	0.097	0.01	0.043	0.091	0.008	0.048	0.103
70	0.02	0.066	0.113	0.013	0.058	0.116	0.017	0.06	0.097	0.016	0.06	0.114
100	0.024	0.059	0.101	0.015	0.054	0.099	0.029	0.064	0.097	0.02	0.056	0.105
150	0.022	0.062	0.098	0.019	0.052	0.081	0.02	0.062	0.105	0.018	0.056	0.093
200	0.013	0.048	0.087	0.012	0.050	0.09	0.013	0.039	0.089	0.013	0.041	0.092

Tabla 4.27: Resultados de simulación de Error tipo I. Caso $\theta = (1.5, 1, 0.98)$, $\rho \approx 0.8$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
30	0.017	0.07	0.138	0.016	0.073	0.130	0.021	0.068	0.129	0.018	0.072	0.127
50	0.015	0.062	0.119	0.016	0.063	0.119	0.014	0.060	0.108	0.013	0.06	0.115
70	0.012	0.043	0.091	0.013	0.043	0.09	0.01	0.049	0.092	0.01	0.049	0.097
100	0.005	0.05	0.102	0.008	0.056	0.107	0.006	0.041	0.102	0.006	0.046	0.098
150	0.009	0.049	0.106	0.011	0.052	0.098	0.012	0.05	0.096	0.009	0.047	0.102
200	0.010	0.047	0.094	0.007	0.041	0.101	0.009	0.048	0.096	0.008	0.043	0.093

Tabla 4.28: Resultados de simulación de Error tipo I. Caso $\theta = (1, 1, 0.1)$, $\rho = 0.1$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
30	0.052	0.141	0.210	0.038	0.117	0.179	0.032	0.115	0.198	0.037	0.121	0.201
50	0.079	0.180	0.263	0.053	0.143	0.222	0.057	0.158	0.225	0.060	0.160	0.243
70	0.117	0.235	0.305	0.082	0.192	0.269	0.083	0.181	0.270	0.087	0.199	0.276
100	0.175	0.281	0.334	0.128	0.243	0.306	0.127	0.249	0.307	0.141	0.259	0.309
150	0.217	0.287	0.317	0.166	0.261	0.306	0.168	0.270	0.305	0.188	0.272	0.309
200	0.258	0.290	0.306	0.214	0.276	0.303	0.219	0.277	0.305	0.231	0.277	0.305

Tabla 4.29: Resultados de simulación de Error tipo I. Caso $\theta = (1, 1, 0.2)$, $\rho = 0.2$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
30	0.014	0.045	0.081	0.009	0.044	0.068	0.013	0.044	0.081	0.011	0.039	0.076
50	0.009	0.041	0.086	0.01	0.039	0.074	0.013	0.045	0.077	0.009	0.039	0.074
70	0.029	0.063	0.101	0.017	0.061	0.107	0.016	0.057	0.102	0.016	0.062	0.100
100	0.018	0.054	0.083	0.015	0.042	0.085	0.014	0.049	0.087	0.016	0.044	0.082
150	0.031	0.058	0.085	0.021	0.056	0.095	0.02	0.052	0.086	0.021	0.054	0.091
200	0.018	0.049	0.077	0.020	0.046	0.085	0.016	0.043	0.075	0.017	0.046	0.075

Tabla 4.30: Resultados de simulación de Error tipo I. Caso $\theta = (1, 1, 0.8)$, $\rho = 0.8$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
30	0.011	0.052	0.114	0.009	0.052	0.1	0.008	0.056	0.112	0.009	0.048	0.113
50	0.012	0.058	0.109	0.012	0.061	0.114	0.014	0.059	0.105	0.008	0.061	0.109
70	0.010	0.055	0.108	0.013	0.055	0.114	0.008	0.050	0.110	0.008	0.053	0.109
100	0.050	0.051	0.122	0.012	0.058	0.118	0.018	0.052	0.112	0.014	0.051	0.119
150	0.006	0.050	0.108	0.008	0.051	0.108	0.008	0.056	0.105	0.008	0.056	0.104
200	0.014	0.041	0.098	0.013	0.043	0.099	0.014	0.045	0.096	0.014	0.045	0.097

Tabla 4.31: Resultados de simulación de Error tipo I. Caso $\theta = (1, 1, 0.9)$, $\rho = 0.9$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
n	0.1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
30	0.018	0.063	0.114	0.02	0.068	0.121	0.021	0.065	0.120	0.020	0.067	0.122
50	0.009	0.054	0.098	0.011	0.056	0.096	0.011	0.056	0.109	0.008	0.056	0.103
70	0.013	0.059	0.114	0.017	0.062	0.107	0.014	0.055	0.112	0.016	0.059	0.117
100	0.014	0.049	0.107	0.015	0.048	0.108	0.013	0.054	0.109	0.012	0.055	0.109
150	0.014	0.058	0.112	0.016	0.059	0.109	0.012	0.056	0.107	0.013	0.059	0.104
200	0.009	0.042	0.089	0.009	0.039	0.089	0.012	0.045	0.091	0.009	0.041	0.091

Se puede apreciar de las tablas anteriores que las probabilidades de error tipo I simuladas son muy cercanas a los valores nominales, en todos los casos en que el coeficiente de correlación $\rho > 0.1$. Sin embargo, tal como se podía esperar de acuerdo a lo planteado por Andrew (2000), en los casos en que $\rho = 0.1$ y $\rho \approx 0.1$ ($\rho = 0.098$) las probabilidades simuladas se encuentran muy alejadas de los valores nominales.

Se estudian también los tiempos, medidos en segundos, que tardaron las simulaciones haciendo uso de 10 núcleos con el comando *parRapply* para los distintos tamaños muestrales n y vectores de pesos (a_1, a_2) estudiados, se presentan a continuación.

Tabla 4.32: Tiempo de ejecución del programa. Caso $\theta = (1.5, 1, 0.12)$, $\rho \approx 0.1$

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
30	1599	1619	1611	1606
50	1735	1737	1738	1739
70	1956	1963	1951	1981
100	2074	2087	2061	2082
150	2382	2380	2359	2411
200	2509	2507	2494	2513

Tabla 4.33: Tiempo de ejecución del programa. Caso $\theta = (1.5, 1, 0.25)$, $\rho \approx 0.2$

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
30	1436	1435	1433	1453
50	1726	1719	1706	1724
70	1795	1774	1772	1785
100	1801	1812	1851	1803
150	1873	1856	1846	1837
200	1898	1889	1880	1892

Tabla 4.34: Tiempo de ejecución del programa. Caso $\theta = (1.5, 1, 0.98)$, $\rho \approx 0{,}8$

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
30	911	911	907	911
50	985	990	984	1001
70	1052	1079	1066	1075
100	1130	1114	1133	1139
150	1898	1889	1880	1892
200	1288	1319	1295	1297

Tabla 4.35: Tiempo de ejecución del programa. Caso $\theta = (1, 1, 0.1)$, $\rho = 0.1$

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
30	1261	1266	1272	1280
50	1533	1537	1529	1521
70	1577	1560	1569	1581
100	1767	1783	1757	1760
150	2045	2046	2072	2043
200	2118	2134	2140	2166

Tabla 4.36: Tiempo de ejecución del programa. Caso $\theta = (1, 1, 0.2)$, $\rho = 0.2$

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
30	1341	1355	1336	1362
50	1568	1533	1540	1563
70	1607	1575	1587	1580
100	1641	1665	1651	1627
150	1653	1669	1662	1652
200	1671	1669	1686	1660

Tabla 4.37: Tiempo de ejecución del programa. Caso $\theta = (1, 1, 0.8)$, $\rho = 0.8$

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
30	754	762	767	754
50	829	821	824	823
70	877	875	880	882
100	941	932	947	934
150	1022	1033	1026	1014
200	1082	1082	1085	1088

Tabla 4.38: Tiempo de ejecución del programa. Caso $\theta = (1, 1, 0.9)$, $\rho = 0.9$

n	$a_1 = 0$, $a_2 = 0$	$a_1 = 0$, $a_2 = 1$	$a_1 = 1$, $a_2 = 0$	$a_1 = 1$, $a_2 = 1$
30	765	786	778	782
50	808	820	809	827
70	873	865	867	871
100	917	923	920	928
150	997	1007	997	1009
200	1075	1068	1067	1058

Se puede apreciar que las simulaciones para aquellos vectores de parámetros en que el coeficiente de correlación ρ es bajo ($\rho \approx 0.1$ y $\rho \approx 0.2$) tardan tiempos significativamente mayores que aquellas en que el coeficiente de correlación ρ es alto ($\rho \approx 0.8$ y $\rho \approx 0.9$).

Capítulo 5

Conclusiones y trabajos futuros

Este trabajo de investigación surge de la necesidad de simular la probabilidad de error tipo I del test $R_{n,w}$ (Novoa y Jiménez, 2014) al variar las componentes respecto de los ya estudiadas por los autores. Lo anterior, debido a la imposibilidad de los autores de hacerlo debido a la gran cantidad de tiempo que significa cada una de estas simulaciones. Luego de haber estudiado distintos métodos de paralelización de la simulación de la probabilidad de error tipo I, se puede concluir que:

- El algoritmo de simulación de la probabilidad de error tipo I del test de bondad de ajuste es paralelizable y permite ser paralelizado al menos de dos maneras distintas en el lenguaje R.
- La paralelización en lenguaje R es una buena alternativa para disminuir los tiempos en la simulación de la probabilidad de error tipo I de este test.
- La probabilidad de error tipo I del test $R_{n,w}$ tiene un valor cercano al nominal tanto para tamaños muestrales bajos (30, 50, 70) como para tamaños muestrales altos (100, 150, 200, 250, 300, 500), siempre y cuando el coeficiente de correlación entre las dos distribuciones Poisson univariadas involucradas es mayor a 0.1.
- La probabilidad de error tipo I del test $R_{n,w}$ toma un valor muy alejado del nominal en los casos estudiados en que las muestras provienen de una población en que el coeficiente de correlación entre las dos distribuciones Poisson univariadas involucradas es menor o igual a 0.1.
- La simulación de la probabilidad de error tipo I del test $R_{n,w}$ implementada en

paralelo toma tiempos significativamente mayores en los casos en que el coeficiente de correlación ρ es bajo ($\rho \approx 0.1$ y $\rho \approx 0.2$), en comparación con aquellos en que el coeficiente de correlación ρ es alto ($\rho \approx 0.8$ y $\rho \approx 0.9$).

En instancias futuras se recomienda:

- Implementar el programa de simulación de la probabilidad de error tipo I del test $R_{n,w}$ en un cluster que se componga de más de 12 procesadores, con el fin de conocer la cantidad de procesadores que logra la aceleración óptima del programa.
- Elaborar un paquete en lenguaje R que incorpore el test de bondad de ajuste propuesto por Novoa y Jiménez (2014).
- Implementar el programa de simulación de la probabilidad de error tipo I del test $R_{n,w}$ en lenguaje de programación C.
- Profundizar la investigación sobre la eficiencia del test $R_{n,w}$ en los casos en que el coeficiente de correlación entre las dos distribuciones Poisson univariadas involucradas son menores a 0.2.

Bibliografía

[1] Almasi, G. y Gottlieb, A. (1989). *Highly parallel computing*, Benjamin-Cummings Publishing Co.

[2] Alvarez, J.y Rubio, P. (2002). Avances recientes en métodos bootstrap para procesos ARCH. Una aplicación en el mercado español de valores. *Estudios de Economía Aplicada*, **20** (2), 487-498.

[3] Andrew, D. (2000). Inconsistency of the bootstrap when a parameter is on the boundary of the parameter space, *Econométrica*, **68**, 399 400.

Baringhaus, L., Gürtler, N. y Henze, N. (2000). Weighted integral test statistics and components of smooth tests of fit, *Australian & New Zealand Journal of Statistics*, **42**, 179–192.

[4] Canty, A., Ripley, B.(1999), "boot: Bootstrap Functions", URL https://cran.r-project.org/package=boot.

[5] Chapple, S., Eilidh T., Thorsten F., Terence S. (2016). *Mastering Parallel Programming with R.* Packt Publishing Ltd.

[6] Crockett, N., (1979). A quick test of fit of a bivariate distribution. In D. McNeil (ed.), *Interactive Statistics*, 185–191. Amsterdam: North-Holland.

[7] Davison, A. y Hinkley, D. (1997) *Bootstrap Methods and Their Application*, Chapter 5. Cambridge University Press.

[8] Holgate, P. (1964). Estimation for the bivariate Poisson distribution, *Biometrika*, **51**, 241–245.

[9] Johnson, N. y Kotz, S. (1969). *Distributions in Statistics: Discrete Distributions*, New York: John Wiley & Sons.

[10] Johnson, N., Kotz, S. y Balakrishnan, N. (1997). *Discrete Multivariate Distributions*, New York: John Wiley & Sons.

[11] Kai Hwang. (1993) *Advanced computer architecture: Parallelism, scalability, programmability*, McGraw-Hill.

[12] Kocherlakota, S. y Kocherlakota, K. (1992). *Bivariate Discrete Distributions*, New York: Marcel Dekker.

[13] Kundu, S., Majumdar, S. y Mukherjee, K. (2000). Central limits theorems revisited, *Statistics & Probability Letters*, **47**, 265–275.

[14] Loukas, S. y Kemp, C. (1986). The Index of Dispersion Test for the Bivariate Poisson Distribution, *Biometrics*, **42**, 941–948.

[15] Loukas, S., Kemp, C. y Papageorgiou, H. (1986). Even point estimation for the bivariate Poisson distribution, *Biometrika*, **73**, 222-223.

[16] Matloff, N. (2016). Parallel computing for data science, *A chapman & hall book*

[17] Novoa, F. y Jiménez, M. (2014). Testing for the bivariate Poisson distribution, *Metrika*, **77(6)**, 771-793.

[18] Pacheco, P., (2011). *An introduction to Parallel Programming*, Morgan Kaufmann Publishers.

[19] Papageorgiou, H. y Loukas, S. (1988). Conditional even point estimation for bivariate discrete distributions, *Communications in Statistics–Theory and Methods*, **17**, 3403–3412.

[20] R Development Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, 2009, URL http://www.R-project.org.

[21] Rayner, J. y Best, D. (1995). Smooth Tests for the Bivariate Poisson Distribution, *Australian & New Zealand Journal of Statistics*, **37**, 233–245.

[22] Román, J., (2018). Ejercicios de programación paralela con openMP y MPI, *Editorial Universitat politécnica de Valencia.*

[23] Urbanek, S.(2014), "multicore: Parallel processing of R code on machines with multiple cores or CPUs", URL https://cran.r-project.org/package=multicore.

Anexos

1. Simulación de Error Tipo I del test $R_{n,w}$, mediante paralelización en lenguaje R con el comando *parRapply*, considerando los mismos componentes utilizados por Novoa Muñoz y Jiménez Gamero (2014).

Tabla 5.1: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.31)$, $\rho = 0.25$, $n = 30$

	$a_1 = 0$, $a_2 = 0$			$a_1 = 0$, $a_2 = 1$			$a_1 = 1$, $a_2 = 0$			$a_1 = 1$, $a_2 = 1$		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.011	0.047	0.086	0.017	0.041	0.089	0.011	0.047	0.094	0.013	0.043	0.082
2	0.012	0.046	0.080	0.014	0.042	0.086	0.007	0.046	0.10	0.014	0.043	0.084
3	0.012	0.042	0.087	0.014	0.043	0.088	0.011	0.048	0.099	0.012	0.044	0.084
4	0.012	0.048	0.082	0.017	0.041	0.092	0.008	0.047	0.096	0.014	0.042	0.080
5	0.012	0.048	0.079	0.017	0.043	0.086	0.010	0.047	0.090	0.013	0.045	0.083
6	0.016	0.044	0.082	0.012	0.042	0.087	0.011	0.047	0.096	0.013	0.041	0.082
7	0.014	0.044	0.082	0.015	0.042	0.083	0.010	0.047	0.090	0.011	0.043	0.085
8	0.010	0.045	0.080	0.014	0.044	0.093	0.009	0.047	0.091	0.012	0.042	0.080
9	0.011	0.045	0.079	0.013	0.039	0.089	0.011	0.048	0.093	0.012	0.045	0.080
10	0.011	0.043	0.083	0.017	0.046	0.085	0.011	0.049	0.095	0.013	0.044	0.084
11	0.011	0.045	0.081	0.015	0.042	0.090	0.008	0.050	0.097	0.014	0.043	0.083
12	0.012	0.041	0.081	0.016	0.042	0.087	0.009	0.044	0.094	0.015	0.040	0.085

Tabla 5.2: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.31)$, $\rho = 0.25$, $n = 50$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.010	0.037	0.079	0.009	0.035	0.081	0.013	0.042	0.075	0.010	0.039	0.082
2	0.008	0.038	0.080	0.011	0.035	0.080	0.014	0.040	0.073	0.009	0.036	0.080
3	0.012	0.040	0.077	0.010	0.036	0.079	0.012	0.042	0.078	0.009	0.036	0.079
4	0.010	0.037	0.077	0.012	0.036	0.083	0.013	0.040	0.074	0.010	0.038	0.077
5	0.008	0.036	0.079	0.012	0.035	0.083	0.011	0.046	0.080	0.011	0.037	0.078
6	0.014	0.037	0.075	0.011	0.038	0.077	0.009	0.039	0.079	0.009	0.036	0.074
7	0.008	0.038	0.081	0.008	0.032	0.083	0.013	0.041	0.080	0.012	0.034	0.077
8	0.010	0.035	0.083	0.011	0.031	0.084	0.009	0.043	0.078	0.013	0.037	0.078
9	0.012	0.037	0.082	0.010	0.036	0.078	0.011	0.041	0.076	0.011	0.037	0.075
10	0.009	0.038	0.080	0.013	0.035	0.082	0.014	0.040	0.076	0.009	0.036	0.076
11	0.011	0.040	0.076	0.013	0.038	0.081	0.016	0.039	0.075	0.006	0.032	0.074
12	0.011	0.040	0.072	0.010	0.034	0.078	0.012	0.046	0.074	0.007	0.036	0.074

Tabla 5.3: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.31)$, $\rho = 0.25$, $n = 70$

	$a1$ =0, $a2$ =0			$a1$ =0, $a2$ =1			$a1$ =1, $a2$ =0			$a1$ =1, $a2$ =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.010	0.044	0.108	0.012	0.050	0.103	0.010	0.059	0.098	0.012	0.043	0.099
2	0.010	0.046	0.101	0.008	0.051	0.103	0.011	0.054	0.098	0.010	0.047	0.099
3	0.011	0.046	0.099	0.011	0.047	0.099	0.010	0.058	0.100	0.009	0.046	0.101
4	0.008	0.046	0.100	0.009	0.052	0.096	0.010	0.052	0.099	0.011	0.043	0.102
5	0.010	0.045	0.098	0.013	0.047	0.094	0.014	0.060	0.097	0.012	0.045	0.101
6	0.010	0.048	0.100	0.011	0.051	0.102	0.007	0.059	0.095	0.012	0.044	0.104
7	0.010	0.050	0.100	0.010	0.047	0.100	0.011	0.058	0.103	0.011	0.049	0.101
8	0.010	0.048	0.097	0.010	0.050	0.098	0.011	0.057	0.099	0.013	0.045	0.104
9	0.012	0.044	0.104	0.013	0.053	0.102	0.012	0.059	0.099	0.014	0.046	0.106
10	0.009	0.045	0.101	0.008	0.046	0.093	0.009	0.054	0.098	0.013	0.048	0.102
11	0.011	0.046	0.101	0.009	0.049	0.099	0.009	0.055	0.094	0.011	0.049	0.104
12	0.009	0.049	0.096	0.010	0.052	0.098	0.011	0.059	0.097	0.011	0.047	0.104

Tabla 5.4: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.62)$, $\rho = 0.5$, $n = 30$

	$a1$ =0, $a2$ =0			$a1$ =0, $a2$ =1			$a1$ =1, $a2$ =0			$a1$ =1, $a2$ =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.009	0.050	0.118	0.008	0.055	0.117	0.013	0.047	0.115	0.009	0.056	0.111
2	0.007	0.054	0.120	0.008	0.056	0.115	0.010	0.047	0.110	0.006	0.056	0.108
3	0.007	0.052	0.119	0.004	0.057	0.116	0.010	0.047	0.117	0.008	0.055	0.116
4	0.007	0.054	0.115	0.008	0.055	0.115	0.007	0.043	0.112	0.008	0.050	0.112
5	0.006	0.055	0.119	0.007	0.058	0.111	0.009	0.051	0.116	0.007	0.054	0.116
6	0.007	0.056	0.121	0.007	0.058	0.115	0.011	0.050	0.113	0.007	0.054	0.117
7	0.007	0.054	0.113	0.006	0.057	0.108	0.009	0.047	0.111	0.006	0.053	0.117
8	0.007	0.055	0.122	0.011	0.055	0.112	0.010	0.053	0.114	0.007	0.055	0.119
9	0.007	0.051	0.121	0.009	0.063	0.115	0.008	0.049	0.115	0.008	0.051	0.114
10	0.008	0.051	0.120	0.006	0.055	0.113	0.010	0.049	0.112	0.009	0.052	0.110
11	0.007	0.052	0.116	0.009	0.058	0.112	0.009	0.047	0.113	0.007	0.053	0.115
12	0.007	0.052	0.117	0.008	0.059	0.116	0.011	0.051	0.112	0.008	0.046	0.111

Tabla 5.5: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.62)$, $\rho \approx 0.5$, $n = 50$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.015	0.049	0.103	0.008	0.052	0.100	0.011	0.054	0.097	0.010	0.051	0.102
2	0.009	0.051	0.101	0.010	0.052	0.103	0.012	0.054	0.091	0.012	0.052	0.098
3	0.009	0.051	0.100	0.012	0.054	0.096	0.010	0.048	0.095	0.011	0.054	0.102
4	0.009	0.050	0.101	0.011	0.050	0.102	0.011	0.050	0.094	0.010	0.052	0.097
5	0.010	0.047	0.101	0.012	0.054	0.100	0.010	0.053	0.095	0.009	0.051	0.100
6	0.010	0.047	0.104	0.012	0.049	0.098	0.011	0.054	0.098	0.013	0.051	0.103
7	0.010	0.051	0.103	0.015	0.053	0.103	0.012	0.051	0.097	0.011	0.049	0.102
8	0.010	0.053	0.102	0.011	0.058	0.100	0.011	0.052	0.097	0.010	0.054	0.105
9	0.010	0.051	0.099	0.011	0.057	0.102	0.011	0.055	0.093	0.010	0.049	0.100
10	0.008	0.052	0.099	0.010	0.053	0.103	0.011	0.048	0.095	0.013	0.052	0.098
11	0.008	0.053	0.104	0.011	0.048	0.101	0.013	0.053	0.094	0.011	0.045	0.102
12	0.009	0.049	0.102	0.012	0.050	0.051	0.013	0.050	0.097	0.014	0.047	0.097

Tabla 5.6: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.62)$, $\rho \approx 0.5$, $n = 70$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.012	0.05	0.107	0.01	0.053	0.109	0.015	0.063	0.115	0.011	0.052	0.102
2	0.013	0.05	0.102	0.011	0.054	0.103	0.012	0.058	0.119	0.012	0.05	0.111
3	0.013	0.053	0.114	0.012	0.053	0.107	0.011	0.06	0.116	0.016	0.049	0.105
4	0.013	0.055	0.112	0.014	0.053	0.104	0.013	0.054	0.113	0.01	0.05	0.104
5	0.011	0.053	0.112	0.007	0.045	0.102	0.011	0.062	0.116	0.014	0.054	0.104
6	0.014	0.053	0.107	0.015	0.052	0.1	0.012	0.06	0.111	0.009	0.051	0.109
7	0.011	0.05	0.115	0.012	0.048	0.104	0.015	0.062	0.114	0.012	0.05	0.103
8	0.013	0.052	0.105	0.014	0.054	0.105	0.014	0.063	0.111	0.011	0.048	0.108
9	0.012	0.055	0.107	0.012	0.053	0.101	0.013	0.061	0.114	0.012	0.05	0.11
10	0.013	0.055	0.106	0.01	0.053	0.104	0.012	0.057	0.117	0.017	0.051	0.102
11	0.011	0.057	0.11	0.012	0.051	0.103	0.012	0.063	0.112	0.013	0.054	0.104
12	0.014	0.048	0.11	0.011	0.052	0.102	0.013	0.063	0.118	0.011	0.054	0.108

Tabla 5.7: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.92)$, $\rho \approx 0.75$, $n = 30$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.019	0.057	0.103	0.014	0.058	0.102	0.015	0.062	0.111	0.020	0.055	0.108
2	0.019	0.056	0.104	0.019	0.056	0.108	0.018	0.057	0.110	0.017	0.056	0.107
3	0.014	0.060	0.106	0.014	0.064	0.109	0.020	0.058	0.108	0.018	0.057	0.106
4	0.017	0.056	0.108	0.013	0.060	0.104	0.019	0.058	0.108	0.018	0.055	0.107
5	0.024	0.057	0.108	0.018	0.058	0.107	0.018	0.056	0.105	0.018	0.056	0.106
6	0.017	0.058	0.107	0.015	0.055	0.109	0.018	0.060	0.112	0.016	0.058	0.105
7	0.019	0.057	0.109	0.017	0.057	0.104	0.020	0.056	0.109	0.020	0.057	0.105
8	0.018	0.055	0.106	0.015	0.056	0.106	0.019	0.056	0.102	0.016	0.060	0.111
9	0.016	0.057	0.109	0.016	0.062	0.109	0.018	0.059	0.106	0.015	0.054	0.106
10	0.017	0.058	0.111	0.014	0.057	0.103	0.020	0.058	0.108	0.021	0.052	0.109
11	0.016	0.060	0.112	0.017	0.059	0.110	0.015	0.062	0.106	0.017	0.059	0.103
12	0.017	0.058	0.107	0.014	0.057	0.109	0.014	0.058	0.107	0.015	0.054	0.108

Tabla 5.8: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.92)$, $\rho \approx 0.75$, $n = 50$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.005	0.036	0.081	0.005	0.031	0.087	0.003	0.031	0.087	0.004	0.032	0.078
2	0.005	0.034	0.079	0.004	0.030	0.085	0.003	0.029	0.085	0.004	0.033	0.076
3	0.003	0.033	0.077	0.006	0.035	0.086	0.005	0.032	0.084	0.005	0.028	0.072
4	0.003	0.034	0.073	0.005	0.031	0.087	0.005	0.029	0.084	0.003	0.035	0.085
5	0.005	0.033	0.074	0.004	0.034	0.083	0.004	0.031	0.083	0.004	0.027	0.085
6	0.004	0.031	0.076	0.005	0.031	0.083	0.007	0.031	0.085	0.004	0.033	0.085
7	0.006	0.033	0.074	0.004	0.036	0.083	0.004	0.030	0.084	0.005	0.027	0.082
8	0.003	0.034	0.074	0.005	0.035	0.081	0.004	0.030	0.081	0.003	0.028	0.083
9	0.006	0.032	0.079	0.005	0.032	0.084	0.003	0.030	0.084	0.004	0.026	0.082
10	0.005	0.030	0.079	0.007	0.032	0.080	0.004	0.033	0.080	0.005	0.027	0.077
11	0.004	0.036	0.076	0.004	0.038	0.080	0.005	0.033	0.083	0.003	0.033	0.082
12	0.007	0.033	0.074	0.007	0.029	0.085	0.003	0.030	0.084	0.005	0.031	0.081

Tabla 5.9: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1.5, 1, 0.92)$, $\rho \approx 0.75$, $n = 70$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.019	0.056	0.098	0.02	0.053	0.105	0.016	0.058	0.111	0.017	0.05	0.104
2	0.019	0.056	0.1	0.014	0.056	0.106	0.015	0.055	0.105	0.019	0.055	0.108
3	0.016	0.057	0.105	0.02	0.057	0.106	0.016	0.056	0.104	0.018	0.051	0.105
4	0.017	0.059	0.105	0.022	0.054	0.109	0.018	0.052	0.102	0.019	0.055	0.106
5	0.019	0.061	0.105	0.015	0.056	0.106	0.017	0.055	0.104	0.015	0.052	0.103
6	0.014	0.058	0.105	0.02	0.054	0.107	0.016	0.059	0.111	0.015	0.054	0.106
7	0.015	0.051	0.104	0.015	0.055	0.108	0.019	0.053	0.106	0.018	0.055	0.106
8	0.017	0.057	0.106	0.018	0.054	0.105	0.016	0.053	0.106	0.019	0.053	0.106
9	0.018	0.06	0.105	0.021	0.057	0.102	0.018	0.057	0.106	0.018	0.053	0.106
10	0.017	0.053	0.103	0.018	0.056	0.105	0.017	0.053	0.102	0.021	0.051	0.106
11	0.016	0.06	0.1	0.02	0.059	0.107	0.018	0.055	0.109	0.018	0.056	0.106
12	0.02	0.058	0.102	0.019	0.06	0.107	0.018	0.056	0.106	0.015	0.048	0.107

Tabla 5.10: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.5)$, $\rho = 0.5$, $n = 30$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.009	0.044	0.098	0.006	0.041	0.089	0.007	0.041	0.093	0.008	0.039	0.086
2	0.009	0.044	0.098	0.007	0.038	0.092	0.007	0.040	0.092	0.006	0.039	0.088
3	0.007	0.045	0.092	0.009	0.044	0.092	0.009	0.044	0.094	0.005	0.036	0.089
4	0.008	0.042	0.091	0.004	0.040	0.090	0.006	0.046	0.086	0.005	0.038	0.086
5	0.007	0.039	0.093	0.005	0.043	0.088	0.007	0.043	0.088	0.005	0.042	0.089
6	0.009	0.046	0.090	0.006	0.041	0.093	0.005	0.047	0.087	0.005	0.035	0.089
7	0.007	0.042	0.090	0.004	0.040	0.093	0.009	0.042	0.091	0.006	0.040	0.091
8	0.006	0.040	0.095	0.007	0.041	0.097	0.008	0.043	0.090	0.006	0.038	0.086
9	0.008	0.044	0.092	0.004	0.044	0.090	0.009	0.043	0.090	0.006	0.042	0.087
10	0.007	0.042	0.093	0.005	0.045	0.091	0.008	0.039	0.090	0.006	0.042	0.087
11	0.005	0.040	0.093	0.007	0.042	0.092	0.010	0.042	0.091	0.005	0.043	0.092
12	0.005	0.041	0.090	0.004	0.039	0.089	0.005	0.042	0.092	0.006	0.045	0.089

Tabla 5.11: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.5)$, $\rho = 0.5$, $n = 50$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.014	0.055	0.121	0.010	0.055	0.115	0.013	0.054	0.118	0.010	0.052	0.110
2	0.012	0.056	0.115	0.009	0.055	0.110	0.014	0.053	0.114	0.012	0.051	0.114
3	0.013	0.054	0.122	0.005	0.055	0.118	0.016	0.054	0.115	0.014	0.052	0.109
4	0.011	0.054	0.118	0.010	0.053	0.112	0.013	0.053	0.117	0.012	0.049	0.117
5	0.014	0.056	0.118	0.010	0.055	0.117	0.011	0.054	0.116	0.012	0.055	0.117
6	0.011	0.058	0.121	0.008	0.054	0.109	0.014	0.051	0.112	0.011	0.052	0.109
7	0.011	0.055	0.116	0.013	0.058	0.115	0.013	0.052	0.118	0.012	0.054	0.109
8	0.012	0.053	0.120	0.007	0.054	0.114	0.012	0.053	0.114	0.011	0.048	0.112
9	0.009	0.057	0.116	0.009	0.056	0.115	0.014	0.055	0.120	0.012	0.051	0.115
10	0.009	0.054	0.114	0.006	0.054	0.113	0.013	0.055	0.114	0.011	0.050	0.112
11	0.013	0.053	0.119	0.008	0.055	0.118	0.013	0.056	0.118	0.012	0.047	0.113
12	0.016	0.053	0.114	0.008	0.057	0.117	0.012	0.057	0.113	0.010	0.051	0.110

Tabla 5.12: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.5)$, $\rho = 0.5$, $n = 70$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.012	0.051	0.112	0.009	0.060	0.114	0.012	0.050	0.101	0.010	0.057	0.114
2	0.011	0.053	0.114	0.012	0.055	0.111	0.011	0.056	0.105	0.012	0.063	0.117
3	0.012	0.062	0.111	0.010	0.057	0.118	0.012	0.060	0.105	0.006	0.058	0.118
4	0.013	0.056	0.116	0.009	0.059	0.116	0.010	0.055	0.107	0.010	0.055	0.113
5	0.012	0.052	0.113	0.007	0.062	0.115	0.013	0.054	0.099	0.013	0.059	0.112
6	0.013	0.055	0.111	0.011	0.053	0.120	0.013	0.051	0.104	0.011	0.058	0.113
7	0.009	0.055	0.109	0.009	0.057	0.117	0.016	0.051	0.110	0.011	0.061	0.113
8	0.015	0.053	0.010	0.010	0.047	0.115	0.010	0.049	0.104	0.010	0.062	0.115
9	0.011	0.055	0.108	0.010	0.060	0.123	0.014	0.054	0.108	0.010	0.062	0.116
10	0.011	0.055	0.110	0.009	0.058	0.116	0.010	0.055	0.103	0.006	0.061	0.118
11	0.009	0.053	0.109	0.010	0.058	0.115	0.009	0.052	0.104	0.012	0.061	0.115
12	0.011	0.055	0.113	0.009	0.063	0.115	0.011	0.050	0.110	0.009	0.063	0.115

Tabla 5.13: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.25)$, $\rho = 0.25$, $n = 30$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.006	0.045	0.084	0.006	0.037	0.092	0.004	0.043	0.089	0.003	0.041	0.083
2	0.001	0.042	0.079	0.004	0.036	0.091	0.002	0.040	0.090	0.005	0.040	0.081
3	0.007	0.043	0.084	0.004	0.034	0.089	0.005	0.040	0.088	0.003	0.040	0.083
4	0.006	0.040	0.086	0.005	0.036	0.091	0.004	0.038	0.088	0.002	0.041	0.081
5	0.004	0.040	0.081	0.004	0.039	0.091	0.002	0.046	0.089	0.003	0.043	0.085
6	0.005	0.043	0.082	0.007	0.035	0.091	0.008	0.039	0.091	0.002	0.041	0.083
7	0.007	0.044	0.081	0.007	0.038	0.092	0.004	0.038	0.088	0.002	0.039	0.085
8	0.007	0.043	0.082	0.009	0.041	0.088	0.002	0.040	0.087	0.001	0.043	0.087
9	0.005	0.046	0.083	0.005	0.034	0.095	0.003	0.043	0.094	0.002	0.039	0.081
10	0.003	0.040	0.081	0.005	0.038	0.091	0.002	0.042	0.088	0.004	0.041	0.085
11	0.003	0.042	0.084	0.008	0.039	0.091	0.004	0.038	0.088	0.002	0.042	0.087
12	0.006	0.042	0.084	0.003	0.040	0.094	0.005	0.044	0.085	0.003	0.042	0.078

Tabla 5.14: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.25)$, $\rho = 0.25$, $n = 50$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.007	0.054	0.092	0.006	0.052	0.092	0.005	0.044	0.100	0.007	0.045	0.101
2	0.005	0.050	0.095	0.011	0.051	0.089	0.004	0.043	0.100	0.005	0.047	0.095
3	0.007	0.050	0.094	0.006	0.051	0.092	0.006	0.042	0.102	0.006	0.047	0.099
4	0.008	0.048	0.100	0.007	0.053	0.093	0.007	0.042	0.097	0.006	0.044	0.093
5	0.008	0.053	0.096	0.005	0.054	0.097	0.007	0.044	0.098	0.003	0.053	0.097
6	0.005	0.051	0.093	0.007	0.056	0.095	0.007	0.044	0.099	0.004	0.040	0.096
7	0.008	0.053	0.096	0.006	0.051	0.092	0.005	0.045	0.104	0.006	0.042	0.092
8	0.007	0.047	0.095	0.008	0.055	0.094	0.007	0.045	0.098	0.005	0.047	0.095
9	0.007	0.052	0.090	0.007	0.056	0.096	0.008	0.049	0.104	0.005	0.048	0.099
10	0.006	0.049	0.095	0.005	0.056	0.094	0.006	0.045	0.100	0.006	0.048	0.095
11	0.007	0.048	0.092	0.004	0.052	0.093	0.006	0.046	0.101	0.009	0.046	0.097
12	0.007	0.050	0.092	0.007	0.053	0.093	0.007	0.042	0.105	0.004	0.047	0.095

Tabla 5.15: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.25)$, $\rho = 0.25$, $n = 70$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.011	0.045	0.098	0.011	0.050	0.098	0.011	0.054	0.098	0.010	0.047	0.098
2	0.011	0.044	0.091	0.011	0.043	0.098	0.011	0.051	0.095	0.010	0.045	0.096
3	0.008	0.049	0.096	0.012	0.048	0.104	0.011	0.050	0.098	0.013	0.047	0.102
4	0.008	0.049	0.097	0.011	0.049	0.098	0.008	0.048	0.097	0.009	0.045	0.101
5	0.009	0.044	0.099	0.010	0.048	0.096	0.011	0.049	0.096	0.009	0.042	0.099
6	0.012	0.046	0.097	0.012	0.045	0.096	0.009	0.044	0.100	0.011	0.046	0.099
7	0.010	0.047	0.099	0.011	0.043	0.090	0.010	0.052	0.098	0.010	0.043	0.101
8	0.009	0.046	0.097	0.011	0.048	0.095	0.008	0.047	0.099	0.011	0.045	0.099
9	0.008	0.048	0.100	0.013	0.046	0.103	0.012	0.050	0.097	0.013	0.047	0.100
10	0.008	0.053	0.095	0.010	0.045	0.091	0.012	0.050	0.097	0.012	0.049	0.100
11	0.011	0.046	0.098	0.010	0.045	0.099	0.012	0.049	0.096	0.010	0.044	0.096
12	0.012	0.048	0.094	0.012	0.043	0.100	0.010	0.046	0.093	0.013	0.046	0.099

Tabla 5.16: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.75)$, $\rho = 0.75$, $n = 30$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.010	0.062	0.124	0.010	0.047	0.120	0.008	0.067	0.119	0.009	0.062	0.125
2	0.010	0.063	0.127	0.010	0.052	0.122	0.008	0.072	0.122	0.009	0.062	0.123
3	0.012	0.065	0.117	0.012	0.047	0.128	0.008	0.064	0.123	0.008	0.062	0.118
4	0.009	0.062	0.117	0.011	0.051	0.122	0.007	0.065	0.123	0.009	0.057	0.119
5	0.010	0.068	0.120	0.011	0.046	0.121	0.009	0.068	0.118	0.006	0.064	0.121
6	0.010	0.067	0.119	0.011	0.051	0.125	0.005	0.070	0.120	0.009	0.057	0.120
7	0.010	0.065	0.119	0.010	0.050	0.125	0.006	0.070	0.117	0.001	0.059	0.120
8	0.008	0.068	0.123	0.010	0.047	0.123	0.007	0.063	0.116	0.009	0.059	0.124
9	0.012	0.058	0.118	0.010	0.049	0.122	0.009	0.073	0.117	0.011	0.063	0.122
10	0.011	0.063	0.124	0.009	0.052	0.127	0.008	0.066	0.116	0.009	0.062	0.120
11	0.011	0.065	0.120	0.009	0.054	0.122	0.007	0.066	0.119	0.011	0.063	0.125
12	0.010	0.069	0.122	0.009	0.050	0.122	0.009	0.066	0.118	0.007	0.060	0.123

Tabla 5.17: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.75)$, $\rho = 0.75$, $n = 50$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.013	0.050	0.102	0.012	0.052	0.103	0.011	0.050	0.106	0.011	0.045	0.103
2	0.014	0.048	0.107	0.011	0.051	0.102	0.013	0.051	0.103	0.011	0.050	0.097
3	0.014	0.048	0.105	0.013	0.052	0.104	0.012	0.051	0.103	0.013	0.049	0.105
4	0.015	0.050	0.102	0.011	0.054	0.104	0.012	0.050	0.108	0.011	0.052	0.105
5	0.012	0.050	0.108	0.011	0.053	0.107	0.010	0.049	0.112	0.012	0.047	0.105
6	0.012	0.048	0.109	0.014	0.053	0.104	0.009	0.046	0.106	0.012	0.046	0.104
7	0.013	0.048	0.104	0.012	0.052	0.101	0.011	0.048	0.106	0.013	0.047	0.106
8	0.014	0.050	0.103	0.011	0.054	0.107	0.012	0.050	0.103	0.013	0.047	0.098
9	0.011	0.049	0.110	0.014	0.053	0.103	0.013	0.053	0.108	0.013	0.046	0.105
10	0.012	0.053	0.099	0.012	0.051	0.105	0.010	0.051	0.108	0.012	0.051	0.105
11	0.012	0.056	0.103	0.013	0.052	0.106	0.011	0.050	0.105	0.011	0.042	0.109
12	0.013	0.052	0.104	0.011	0.048	0.099	0.012	0.049	0.104	0.011	0.049	0.100

Tabla 5.18: Resultados simulación probabilidad de error tipo I. Caso $\theta = (1, 1, 0.75)$, $\rho = 0.75$, $n = 70$

	a_1 =0, a_2 =0			a_1 =0, a_2 =1			a_1 =1, a_2 =0			a_1 =1, a_2 =1		
p	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %	1 %	5 %	10 %
1	0.011	0.059	0.116	0.011	0.066	0.125	0.014	0.053	0.112	0.013	0.06	0.114
2	0.007	0.052	0.119	0.011	0.059	0.116	0.013	0.056	0.118	0.012	0.06	0.115
3	0.012	0.063	0.122	0.011	0.062	0.123	0.013	0.051	0.112	0.013	0.059	0.113
4	0.012	0.06	0.115	0.008	0.059	0.122	0.013	0.05	0.114	0.01	0.064	0.116
5	0.012	0.058	0.118	0.008	0.058	0.119	0.014	0.053	0.115	0.011	0.06	0.118
6	0.011	0.057	0.124	0.01	0.06	0.127	0.013	0.052	0.114	0.01	0.06	0.117
7	0.009	0.053	0.12	0.014	0.052	0.119	0.013	0.055	0.112	0.011	0.062	0.11
8	0.009	0.055	0.12	0.009	0.058	0.121	0.014	0.055	0.115	0.009	0.064	0.118
9	0.01	0.056	0.117	0.01	0.057	0.123	0.012	0.051	0.116	0.012	0.06	0.111
10	0.011	0.053	0.12	0.007	0.062	0.119	0.012	0.054	0.114	0.012	0.059	0.119
11	0.011	0.056	0.118	0.012	0.063	0.125	0.011	0.049	0.114	0.012	0.061	0.121
12	0.008	0.06	0.122	0.014	0.06	0.122	0.011	0.051	0.114	0.009	0.063	0.116

2. Código en lenguaje R de la simulación de la probabilidad de error tipo I en paralelo.

```
#####################################################
#Estimacion con metodo de maxima verosimilitud
#####################################################
# Esta funcion estima los parametros de la DPB por el metodo de maxima
# verosimilitud, se utiliza la metodologia empleada por Kocherlakota & Kocherlako
# X es la matriz de orden 2xn que contiene la muestra X=((X_1i,X_2i)), i=1,2,...,
EstimadorMV<-function(X)
{
  n <- dim(X)[2]
  t1 <- mean(X[1,])
  t2 <- mean(X[2,])
  t3 <- min(t1,t2)/2  # valor inicial del parmetro t3

  x_min <- min(X[1,])
  x_max <- max(X[1,])
  y_min <- min(X[2,])
  y_max <- max(X[2,])

  frec <- matrix(0,x_max-x_min+1,y_max-y_min+1)
  for (i in x_min:x_max)
  {
    p <- i-x_min+1
    for (j in y_min:y_max)
    {
      q <- j-y_min+1
      for (k in 1:n)
      {
        if (X[1,k]==i && X[2,k]==j) frec[p,q] <- frec[p,q]+1
      }
    }
  }
  #mtodo de Newton-Raphson
```

```
  #anterior_t3 guarda el anterior valor de t3
  #RMyDerRM_t3 es la funcin que calcula R media y la derivada de R media
  respecto de t3
  num<-1
  difm <- 0.001
  diff<-100
  dif_min <- 100
  t3_min <- t3
  while (diff>=difm && num<=200)
  {
    anterior_t3 <- t3
    RM_Der <- RMyDerRM_t3(n,x_min,x_max,y_min,y_max,frec,t1,t2,t3)
    t3 <- t3-(RM_Der[1]-1)/RM_Der[2]
    diff <- abs(anterior_t3-t3)
    if(is.na(diff))
    {
      diff <- difm/10
      num <- 250
      t3 <- t3_min
    }
    else
    {
      if(diff<dif_min)
      {
        dif_min <- diff
        t3_min <- t3
      }
    }
    num<-num+1
  }
  return(c(t1,t2,t3))
}

#esta funcin calcula la funcin de probabilidad de una PB(t1,t2,t3)
probPB<-function(x,y,t1,t2,t3)
{
```

```
  sal<-0
  if(x>=0 && y>=0)
  {
    m<-min(x,y)
    tt1<-t1-t3
    tt2<-t2-t3
    for (i in 0:m)
    {
      sal<-sal+(tt1^(x-i))*(tt2^(y-i))*(t3^i)/(gamma(x-i+1)*gamma(y-i+1)*
      gamma(i+1))
    }
    sal<-sal*exp(-tt1-tt2-t3)
  }
  return(sal)
}

# esta funcin da la expresin de R media y de la derivada de R media respecto de t3
# la entrada es la matriz de frecuencias, de dim (x_max-x_min+1) por (y_max-y_min+
# y cuyos elementos suman n
RMyDerRM_t3 <- function(n,x_min,x_max,y_min,y_max,frec,t1,t2,t3)
{
  der <- rm <- 0
  for(i in x_min:x_max)
  {
    p <- i-x_min+1
    for (j in y_min:y_max)
    {
      q <- j-y_min+1
      if (frec[p,q]>0)
      {
        prob_ij <- probPB(i,j,t1,t2,t3)
        prob_ijm1 <- probPB(i-1,j-1,t1,t2,t3)
        rm <- rm+frec[p,q]*prob_ijm1/prob_ij
        a1 <- (probPB(i-2,j-2,t1,t2,t3)-probPB(i-2,j-1,t1,t2,t3)-
        probPB(i-1,j-2,t1,t2,t3))/prob_ij
        a2 <- prob_ijm1*(probPB(i-1,j,t1,t2,t3)+probPB(i,j-1,t1,t2,t3)-
```

```
        prob_ijm1)/(prob_ij^2)
        der <- der+frec[p,q]*(a1+a2)
      }
    }
  }
  return(c(rm/n,der/n))
}

f_n <- function(i,j,n,X)
{
  ind1 <- ind2 <- rep(0,n)
  ind1[X[1,]==i] <- 1
  ind2[X[2,]==j] <- 1
  ss <- sum(ind1*ind2)
  return(ss/n)
}


################################################
# Generacin de muestras Poisson bivariante
################################################
# Esta funcin genera muestra poisson bivariante PB(t1,t2,t3) de tamao n
# la salida es el arreglo de orden 2xn: muestraPB=((X_1i,X_2i)), i=1,2,...,n
# (X_1,X_2) sigue una PB(t1,t2,t3)
# t1=E(X_1), t2=E(X_2), t3=cov(X_1,X_2)
# x1 y x2 deben ser enteros no negativos,
# t1,t2 y t3 son los parmetros estimados de la DPB, tales que t1>t3, t2>t3, t3>0
gmpb <- function(n,t1,t2,t3)
{
  if (t1>t3 && t2>t3 && t3>0)
  {
    a <- rpois(n,t1-t3)
    b <- rpois(n,t2-t3)
    c <- rpois(n,t3)
    return(rbind(a+c,b+c))
  }
```

```
  else
  {
    stop("Los parmetros no cumplen los requisitos")
    on.exit
  }
}


#################################
# Clculo del estadstico R_n,a
########################################
# Esta funcin calcula el estadstico R
# n es el tamao de la muestra
# X es la matriz 2xn que contiene la muestra X=(X_1i,X_2i)
# t1,t2 y t3 son los parmetros estimados de la DPB, tal que t1>t3, t2>t3, t3>=0.
# a1>-1, a2>-1 son los parmetros de la funcin de peso w(u)
#Debo acomodar la funcin para que slo dependa de X,t1,t2,t3
R <- function(X,t1,t2,t3)
{
  m <- 10
  inf <- max(max(X[1,]),max(X[2,]))+m

  fp <- Prob <- matrix(0,inf+1,inf+1)
  for(i in 0:inf)
  {
    for(j in 0:inf)
    {
      Prob[i+1,j+1] <- probPB(i,j,t1,t2,t3)
      fp[i+1,j+1] <- f_n(i,j,n,X)-probPB(i,j,t1,t2,t3)
    }
  }

  k <- l <- 0:inf
  f <- function(k,l){1/((ii+k+a1+1)*(jj+l+a2+1))}

  suma <- 0
```

```
  for(ii in 0:inf)
  {
    for(jj in 0:inf)
    {
      suma <- suma+fp[ii+1,jj+1]*sum(fp*outer(k,l,FUN=f))
    }
  }
  return(n*suma)
}


#########################################################
##Defino la funcin para introducir como argumento en parRapply##
##Debe ser funcin de una columna##
funcion_completa<-function(X)
{
  ##le damos la forma de matriz de 2xn
  X<-data.frame(matrix(unlist(X), ncol = 2, byrow = F))
  X<-t(X)

  theta_estim<-EstimadorMV(X)
  t1<-theta_estim[1]
  t2<-theta_estim[2]
  t3<-theta_estim[3]

  ## Calculo R observado como funcin de la muestra (como matriz de 2xn)
  r_obs<-R(X,t1,t2,t3)

  ##Calculos los R boot
  r_boot <- rep(0,B)

  for (b in 1:B)
  {
    sigue <- "no"
    while(sigue=="no")
    { # 2. Se generan B muestras bootstraps Mboot desde la DPB PB(theta_estim)
```

```
      X_PBboot <- gmpb(n,theta_estim[1],theta_estim[2],theta_estim[3])

      # 3. Se calcula el estimador bootstrap de theta
      theta_est_boot <- EstimadorMV(X_PBboot)

      if(theta_est_boot[1]>theta_est_boot[3] && theta_est_boot[2]>theta_est_boot[3
      && theta_est_boot[3]>0) sigue <- "si"
    }
    # 4. Se evalan los estadsticos en cada muestra bootstrap
    r_boot[b] <- R(X_PBboot,theta_est_boot[1],theta_est_boot[2],theta_est_boot[3])
  }

  # 5. Se acumula una aproximacin del valor-p para cada estadstico
  ind_r <- rep(0,B)

  ind_r[r_boot >= r_obs] <- 1
  vp_b <- sum(ind_r)/B
}


###Defino M y##
M<-1000
B<-500

# vector de (a1,a2)
va1<-c(0,0,1,1)
va2<-c(0,1,0,1)



library(parallel)
nc <- detectCores()

tm <- c(30,50,70,100,150,200,250,300,500)   # Tamao muestral
Theta_matriz<-matrix(0,6,3)
Theta_matriz[1,]<-c(1.5,1,0.31)
```

```
Theta_matriz[2,]<-c(1.5,1,0.62)
Theta_matriz[3,]<-c(1.5,1,0.92)
Theta_matriz[4,]<-c(1,1,0.5)
Theta_matriz[5,]<-c(1,1,0.25)
Theta_matriz[6,]<-c(1,1,0.75)


for (i in 1:6) {

theta <- Theta_matriz[i,]

pares = c()                          # Se crea un vector vaco
impares = c()                        # Idem
for(i in 1:(2*M)){                   # Se van a procesar los nmeros de 1 a 2*M
  if(i%%2==0) pares<-c(pares,i)      # Si al dividir por 2 sale 0
  else impares<-c(impares,i)
}

for(t in 1:length(tm))
{ #### Muestras
  n <- tm[t]
  X <- Y <- matrix(0,M,n)


  carpeta <-
  "C:/Users/CLAUDIA GONZALEZ/Desktop/Magister UBB/tesis/muestras/muestras_R"
  arch_L <-
  paste(carpeta,"/Theta_",theta[1],"_",theta[2],"_",theta[3],"_n",n,".txt",
  sep="")

  XY<-matrix(scan(arch_L,sep=","),nrow=2*M,byrow=T)
  X=XY[impares,]
  Y=XY[pares,]

  # Defino una matriz de 2n x M para guardar las muestras
  XY_columnas <- matrix(0,2*n,M)
```

```
for (m in 1:M)
{
  XY_columnas[,m] <- c(X[m,],Y[m,])
}


## paralelo con parRapply con 1 a nc nucleos
carp_arch <-
"C:/Users/CLAUDIA GONZALEZ/Desktop/Magister UBB/tesis/Salidas/"
arch <-
paste(carp_arch,"paralelo_completo",theta[1],"_",theta[2],"_",
theta[3],"_n",n,".txt",sep="")

for (a_12 in 1:4)
{
a1<-va1[a_12]
a2<-va2[a_12]
for(nn in 1:nc)
{
  cl<- parallel::makeCluster(nn)
  clusterExport(cl, "EstimadorMV")
  clusterExport(cl, "RMyDerRM_t3")
  clusterExport(cl, "probPB")
  clusterExport(cl, "f_n")
  clusterExport(cl, "n")
  clusterExport(cl, "a1")
  clusterExport(cl, "a2")
  clusterExport(cl, "B")
  clusterExport(cl, "M")
  clusterExport(cl, "R")
  clusterExport(cl, "gmpb")

  write(paste("Salidas con ",nn," nucleos", "a1=",a1, "a2=",a2, sep=""),
  file=arch,append=TRUE)
```

```
    a<- proc.time()
    res<-parRapply(cl, t(XY_columnas), funcion_completa)
    parallel::stopCluster(cl)
    tiempo_de_ejecucion <- proc.time()-a

    write("",file=arch,append=TRUE)
    write(c("vp : ",res),ncolumns=length(res)+1,file=arch,append=TRUE)

    ind1_r <- ind2_r <- ind3_r <- rep(0,M)
    ind1_r[res <= .01] <- 1
    error01_r <- sum(ind1_r)/M

    ind2_r[res <= .05] <- 1
    error05_r <- sum(ind2_r)/M

    ind3_r[res <= .1] <- 1
    error10_r <- sum(ind3_r)/M
    write("",file=arch,append=TRUE)
    write(c("errores : ",error01_r,error05_r,error10_r),file=arch,ncolumns=4,
    append=TRUE)
    write("",file=arch,append=TRUE)

    write("Tiempo de ejecucion",file=arch,append=TRUE)
    write(tiempo_de_ejecucion,file=arch,append=TRUE)
  }
  }
}
}
```