EL COLEGIO DE MÉXICO
CENTRO DE ESTUDIOS ECONÓMICOS

MAESTRÍA EN ECONOMÍA

TRABAJO DE INVESTIGACIÓN PARA OBTENER EL GRADO DE MAESTRO EN ECONOMÍA

EVALUACIÓN DE PRUEBAS ÓMNIBUS DE NORMALIDAD MULTIVARIADA MEDIANTE LA TÉCNICA DE MONTE CARLO

BÁRBARA RUTH TREJO BECERRIL

PROMOCIÓN 1997-1999

ASESOR:

CARLOS MANUEL URZÚA MACÍAS

2006

Agradecimientos

Son muchas las personas a quienes quiero agradecer su ayuda, en primer lugar a mis padres y a mis tres hermanas por todo su apoyo, a mis profesores por todo lo que aprendí de ellos, al Dr. Carlos Urzúa por su tiempo y ayuda para la realización de esta tesis y finalmente a mis compañeros, pero en especial a Luis Fernando, no solo por haber estudiado con él desde el principio y por hacer de mi estancia en el Colegio tan divertida, sino por estar conmigo cuando más lo necesitaba.

A todos ellos, ¡muchas gracias!

ABSTRACT

Mediante la tecnica de Montecarlo se evalua la potencia de las pruebas de normalidad multivariada propuestas por Urzúa (1997) las cuales son: prueba utilizando solo terceros y cuartos momentos puros (ALM_p), prueba ajustada de falta de simetría ($ALM_{1,p}$) y prueba ajustada de kurtosis ($ALM_{2,p}$), para el caso de observaciones, para lo cual se elaboraron subrutinas en el lenguaje de programación GAUSS, se consideraron seis funciones de distribución multivariadas: t-multivariada, normal multivariada, mezcla de dos normales multivariadas, Wishart, Dirichlet y un vector aleatorio cuyas entradas son distribuciones univariadas como: T-student, F, Tukey, mezcla de dos normales y uniforme, ademas se consideraron seis tamaños de muestras (10, 20, 50, 100, 200, 800) y diferentes dimensiones del vector aleatorio (1, 2, 3, 4, 5) y valores de significancia del 90%. Los resultados obtenidos en el trabajo fueron los siguientes: para el caso de la distribución normal multivarada, las mejores pruebas son la $ALM_{1,P}$ para tamaños de muestras pequeñas y la ALM_P para tamaños de muestras grandes en los casos de $p = 2, 3, 5$, mientras que en la distribución mezcla de dos normales multivariadas y la distribución t-multivariada la mejor prueba en casi todos los casos es la $ALM_{2,p}$, en la distribución Dirichlet la mejor prueba fue la $ALM_{1,P}$, para $p = 1, 2, 3, 4$, mientras que para $p = 1$, la mejor resultó ser la prueba $ALM_{2,p}$, para la distribución Wishart las mejores pruebas para $p = 1, 2, 3, 4$ fueron las pruebas $ALM_{2,p}$ ALM_p , mientras que en $P = 1$, la mejor resultó ser la prueba $ALM_{1,p}$, respecto al vector aleatorio todas las pruebas resultaron ser buenas, con una pequeña ventaja de las pruebas $ALM_{1,p}$ y ALM_p . Por todo lo anterior no se puede afirmar que prueba es mejor, pues esto depende de la distribución que se este analizando y en algunos casos tambien de la dimensión del vector y el tamaño de la muestra. Pero lo que si se puede afirmar es que los resultados convergen a la solución, esto es, a medida que el tamaño de la muestra aumenta el resultado refleja mejor la realidad, es decir, la existencia o ausencia de normalidad.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

Capítulo 1

INTRODUCCIÓN

El presente trabajo tiene como objetivo medir la potencia de las pruebas ómnibus de normalidad multivariada propuestas por **Urzúa (1997)**, para el caso de observaciones, el método que se utilizó para medir la potencia de estas pruebas fué la técnica de Monte Carlo que consiste en simular números aleatorios multivariados de diferentes distribuciones multivariadas, aplicarles el estadístico propuesto y medir la efectividad de cada prueba (es decir cuantas veces el estadístico rechazó normalidad multivariada dado que no existía). Por lo que el primer paso de este trabajo consistió en hacer programas computacionales que generaran números aleatorios multivariados, entre las distribuciones que se ocuparon para probar la potencia de estas pruebas se encuentran las siguientes: Wishart, Dirichlet, mezcla de normales multivariadas, un vector aleatorio cuyas entradas son variables aleatorias univariadas con distribuciones como la t-student, F y Tukey, también una distribución t-student multivariada y una normal multivariada, estos dos últimos programas fuéron realizados por Gary King (1999) de la universidad de Harvard. Una vez teniendo estas subrutinas se desarrollo el programa que simula la técnica de Monte Carlo, las pruebas se realizaron con valores de significancia del 10 %.

Cabe mencionar que en la realidad no es fácil encontrar números aleatorios multivariados con una función de densidad normal multivariada (NM), sin embargo es importante tener este tipo de pruebas de normalidad multivariadas pues existen muchos modelos que entre sus supuestos contemplan que las observaciones se distribuyan como una NM, por lo que es necesario antes de utilizar este tipo de modelos verificar que en verdad los datos en estudio se distribuyen NM, pues si esto no ocurre las conclusiones que se puedan generar de la aplicación de estos métodos no son validas, un ejemplo de este tipo son los modelos de tipo estadístico que tratan de calcular el rendimiento de algún instrumento financiero los cuales asumen que el activo se distribuye normal multivariado independiente e identicamente distribuido en el tiempo, como en los modelos multifactores. Entre las variables que existen que se distribuyen de esta manera se encuentran variables de tipo financieras, por ejemplo:

La solución del siguiente sistema de ecuaciones diferenciales estocásticas se distribuye normal multivariada.

$$df = f\mu_f dt + f\sigma_{f,1} dW_1(t) + f\sigma_{f,2} dW_2(t)$$
$$dg = g\mu_g dt + g\sigma_{g,1} dW_1(t) + g\sigma_{g,2} dW_2(t)$$

donde $W_1(t)$, $W_2(t)$ son dos movimientos brownianos estándar independientes. Este sistema se podría ver como si f fuera el precio de una acción el cual depende de

las variaciones en rendimiento de esta acción y de las variaciones en el rendimiento de otra acción g y lo mismo para la acción g.

Este trabajo esta organizado de la siguiente manera: 1. un resumen de las pruebas de normalidad multivariadas propuestas por Urzúa (1997), 2. una breve revisión de los métodos que existen para la generación de números aleatorios univariados y multivariados, 3. Resultados y por último 4. Conclusiones, también existe un apéndice donde se encuentran todos los programas que se utilizaron para la obtención de resultados de la presente tesis.

Capítulo 2

PRUEBAS DE NORMALIDAD MULTIVARIADA PARA OBSERVACIONES.

Como se dijo al principio lo que se pretende en este trabajo es evaluar la potencia de la pruebas ómnibus de normalidad multivariada propuestas por Urzúa (1997), en esta sección se explicara en que consisten dichas pruebas.

2.1. La prueba del multiplicador de lagrange.

Notación: Sea $s(\alpha)$ el gradiente (score) de la función log-verosimilitud, y sea $I.$ la matriz de información. Dada una partición de α como $\left(\theta_1', \theta_2'\right)'$, el score puede ser escrito como $s(\alpha) = \left(s_1', s_2'\right)'$, con $s_j = \partial L(\alpha)/\partial\theta_j$, $j = 1, 2$; mientras que la matriz de información puede ser particionada en cuatro submatrices de la forma

$I_{ij} = E\{-\partial^2 L(\alpha)/\partial\theta_i\partial\theta_j'\}, i, j = 1, 2.$

Sea $\left(\widetilde{\theta}_1, 0\right)$ el estimador de máxima verosimilitud para $\alpha = \left(\theta_1', \theta_2'\right)'$; es decir $\widetilde{\theta}_1$ es el estimador de máxima verosimilitud para θ_1 después de establecer la restricción $\theta_2 = 0$. Sea además $\widetilde{s} = s\left(\widetilde{\theta}_1, 0\right)$ y $\widetilde{I} = I\left(\widetilde{\theta}_1, 0\right)$. Entonces el estadístico LM es definido como $LM = \widetilde{s}' I^{-1}\widetilde{s}/n$, pero como $\widetilde{s}_1 = 0$,

$$LM = \widetilde{s}_2'\left(\widetilde{I}_{22} - \widetilde{I}_{21}\widetilde{I}_{11}^{-1}\widetilde{I}_{12}\right)^{-1}\widetilde{s}_2/n$$

LM se distribuye asintóticamente bajo H_0 como una χ_v^2,Chi-square con v grados de libertad los cuales son iguales a la dimensión del vector θ_2.

2.2. Una prueba de normalidad multivariada para observaciones.

Como se puede observar la complejidad de las pruebas de normalidad multivariada podría haber complicado la obtención del estadístico LM , sin embargo si se asume que la distribución alternativa para la normal multivariada es la exponencial cuártica, la obtención del estadístico LM ya no es una tarea tan compleja.

Asumir que la exponencial cuártica es la distribución alternativa para la normal multivariada es suficiente, pues (i) es la distribución "más apropiada" cuando momentos más alla del cuarto se asume que existen, y (ii) esta es cercana a la normal multivariada, esta puede aproximar a la familia de Pearson multivariada tanto como se necesite. Además de que todos los resultados obtenidos pueden ser extendidos a todas las posibles distribuciones Q-exponenciales.

Notación.

Primero se transforman las observaciones originales de X: Sea $\overline{x}$ y S el vector de medias de una muestra y la matriz de varianzas y covarianzas de la muestra de observaciones $\{x_1, ..., x_n\}$. Sea G la matriz ortogonal cuyas columnas son eigenvectores estandarizados de S, y D la matriz de eigenvalores. Usando $S^{-1/2} = GD^{-1/2}G'$, se transforman las observaciones en:

$$y_t = S^{-1/2}(x_t - \overline{x}). \qquad t = 1, ..., n.$$

después se define

$$Q_{ijk} = \sum_{t=1}^{n} y_{ti}y_{tj}y_{tk}/n \qquad y \qquad R_{ijkq} = \sum_{t=1}^{n} y_{ti}y_{tj}y_{tk}y_{tq}/n \tag{1}$$

El resultado principal del artículo Urzúa (1997) es presentado como sigue:

Proposición 1.

Bajo las condiciones mencionadas, el multiplicador de Lagrange para probar normalidad multivariada de observaciones esta dado por

$$\mathbf{LM}_p = \mathbf{n}\left[\begin{array}{c} \sum_{i=1}^{p} Q_{iii}^2/6 + \sum_{\substack{i,j=1 \\ i\neq j}}^{p} Q_{iij}^2/2 + \sum_{\substack{i,j,k=1 \\ i<j<k}}^{p} Q_{ijk}^2 + \sum_{i=1}^{p}(R_{iiii}-3)^2/24 + \\ \sum_{\substack{i,j=1 \\ i<j}}^{p}(R_{iijj}-1)^2/4 + \sum_{\substack{i,j=1 \\ i\neq j}}^{p} R_{iiij}^2/6 + \sum_{\substack{i,j,k=1 \\ i\neq j, i\neq k, j<k}}^{p} R_{iijk}^2/2 + \\ \sum_{\substack{i,j,k,q=1 \\ i<j<k<q}}^{p} R_{ijkq}^2 \end{array}\right]$$

donde el estadístico LM se distribuye asintóticamente como una χ_v^2, con $v = p(p+1)(p+2)(p+7)/24$.

Esta es una prueba ómnibus que involucra todos los posibles terceros y cuartos momentos (puros y mixtos).

Casos especiales de la prueba LM_p Esta prueba LM incluye varios casos especiales de pruebas para normalidad multivariada que han sido propuestos en la literatura.

1. el caso univariado LM_1 la cual puede ser expresada en términos del tercer y cuarto momento estandarizados de las observaciones originales.

Definiendo $\sqrt{b_1} = m_3/m_2^{3/2}$ y $b_2 = m_4/m_2^2$, donde el i-esimo momento central m_i es igual a $\sum (x_j - \overline{x})^i / n$, por lo que la prueba es la siguiente:

$$LM_1 = n\left[\left(\sqrt{b_1}\right)^2/6 + (b_2 - 3)^2/24\right] \overset{A}{\sim} \chi_2^2.$$

Este estadístico fué propuesto en 1975 por Bowman y Shenton, y también por Jarque y Bera en 1980 y 1987, estos autores asumen que bajo la hipótesis nula la media asintótica de $\sqrt{b_1}$ y b_2 son 0 y 3 respectivamente, mientras que sus varianzas asintóticas son $6/n$ y $24/n$, y su covarianza asintótica es cero.

2.2.1. Ajustando la prueba del multiplicador de Lagrange.

Esta subsección presenta unas pruebas más simples utilizando solo algunos elementos de la prueba LM_p.

Caso univariado. Utilizando el resultado de Fisher (1930), es posible calcular exactamente cuales son las medias y las varianzas de $\sqrt{b_1}$ y b_2 bajo normalidad (hipótesis nula). Como en el caso asintótico, $E\{\sqrt{b_1}\} = 0$, pero la otra media y las otras dos varianzas son exactamente: (Ver Urzúa 1996)

$$E\{b_2\} = 3(n-1)/(n+1)$$
$$var\{\sqrt{b_1}\} = 6(n-2)/(n+1)(n+3)$$

$$var\{b_2\} = 24n(n-2)(n-3)/(n+1)^2(n+3)(n+5) \qquad (2)$$

Entonces como fué sugerido en Urzúa (1997), se sustituyen los valores asintóticos por los valores exactos y así se obtiene un nuevo estadístico LM_1, por lo que la prueba es:

$$ALM_1 = \left(\sqrt{b_1}\right)^2/var\{\sqrt{b_1}\} + (b_2 - E\{b_2\})^2/var\{b_2\} \overset{A}{\sim} \chi_2^2.$$

Como fué visto en Urzúa (1996), esta nueva prueba de normalidad es mejor en los caso de pequeños y medianos tamaños de muestras. Además, como también se probó el poder de este estadístico es más grande que el poder del estadístico Bowman-Shenton. Jarque-Bera.

Dados estos resultados, se consideraron las contrapartes individuales de la prueba ómnibus univariada ALM_1. Entonces usando la misma prueba, la prueba ajustada que mide la falta de simetría se define como:

$ALM_1 = \left(\sqrt{b_1}\right)^2 / var\{\sqrt{b_1}\} \overset{A}{\sim} \chi^2_2.$

y la prueba ajustada que mide la kurtosis se define como:

$ALM_1 = (b_2 - E\{b_2\})^2 / var\{b_2\} \overset{A}{\sim} \chi^2_2.$

Caso multivariado Primero se reducen los grados de libertad presentados en la prueba general dada por la proposición 1, es natural enfocarse solo en los terceros y cuartos momentos puros. Utilizando (1) y (2), la prueba omnibus ajustada para normalidad multivariada de observaciones es definida como:

$$ALM_p = \sum_{i=1}^{p} Q^2_{iii}/var\{\sqrt{b_1}\} + \sum_{i=1}^{p} (R_{iiii} - E\{b_2\})^2 / var\{b_2\} \overset{A}{\sim} \chi^2_{2p}.$$

Además generalizando el caso univariado, se define la prueba ajustada de falta de simetría como:

$$ALM_{1,p} = \sum_{i=1}^{p} Q^2_{iii}/var\{\sqrt{b_1}\} \overset{A}{\sim} \chi^2_p.$$

y la prueba ajustada de kurtosis se define como:

$$ALM_{2,p} = \sum_{i=1}^{p} (R_{iiii} - E\{b_2\}^2/var\{b_2\} \overset{A}{\sim} \chi^2_p.$$

Para cada uno de los tres estadísticos siguientes, la tabla 1 (ver Urzúa 1997) reporta los valores críticos para varios niveles de significancia usados para probar normalidad multivariada en observaciones.

Capítulo 3

GENERACIÓN DE NÚMEROS ALEATORIOS

Para realizar las pruebas a los estadísticos primero se generaron números aleatorios multivariados continuos, las distribuciones que se generaron fueron la distribución Wishart, Dirichlet, t-student multivariada, normal multivariada, mezcla de normales multivariadas, y un vector aleatorio en el cual cada entrada tiene una distribución aleatoria univariada diferente. La Métodología que se empleo para la generación de estos números se describe a continuación:

Para la generación del vector aleatorio, primero se generaron tres distribuciones aleatorias univariadas las cuales fueron: la distribución F, Tukey y t-student.

3.1. Generación de números aleatorios univariados

Existen varios métodos para generar números univariados los cuales tienen en común el siguiente supuesto.

Existe un generador perfecto de números aleatorios uniformes, es decir, un generador capaz de reproducir una secuencia $U_1, U_2, ..., U_n$ *de variables, con una distribución uniforme en* $(0, 1)$.

Entre los métodos más importantes que existen para generar estos números aleatorios se encuentran:

1. Método de rechazo
2. Método del alias
3. Método de la urna de alias (Alias urn method)
4. Método de aceptación-complemento
5. Método de rechazo de mezclas (Mixture rejection method)
6. Método de aproximación exacta
7. Método de tiempo de espera
8. Método Forsythe
9. Método de proporciones uniformes

3.1.1. Método de rechazo

La condición para poder aplicar este método es que $f(x)$ este acotada, y que x tenga un rango finito i.e. $a \leq x \leq b$. Este método consiste en normalizar primero el rango f mediante un escalar c, tal que $cf(x) \leq 1$. Posteriormente se define X como una función lineal de U, $x = a+(b-a)U$. A continuación se generan pares de números aleatorios uniformes (u_1, u_2). Finalmente, cuando se encuentra un par (u_1, u_2) tal que $u \leq cf(a + (b-a)u_1)$, se aceptará el par y se usará $x = a+(b-a)u_1$, como la variable aleatoria cuya densidad de probabilidad es $f(x)$.

Se ha encontrado que el número esperado de intentos antes de aceptar un par (u_1, u_2) es igual a $1/c$, lo cual implica que el método puede ser ineficiente para ciertas funciones de densidad.

3.1.2. Método del alias

Este método es eficiente en la generación de observaciones de distribuciones discretas con soporte finito. Sin perdida de generalidad, sea X una variable aleatoria que toma los valores $1, 2, ..., n$ con sus respectivas probabilidades $p_1, p_2, ..., p_n$. Para $i = 1, 2, ..., n$, sea a_i, llamado un valor alias, un entero de 1 a n. El método del alias primero selecciona un entero $I = i$ uniformemente de $\{1, 2, ..., n\}$ y entonces regresa $X = i$ con una probabilidad apropiada de lo contrario regresa un valor alias. El método del alias asegura que una observación es regresada en iteración.

El método del alias se implementa en dos etapas. La primer etapa establece los valores alias y las probabilidades de aceptación. Esta etapa esta basada en un teorema de Kronmal y Peterson (1979) el cual dice que cualquier distribución discreta con un soporte finito puede ser expresada como una mezcla de distribuciones y la segunda etapa que es más simple esta dada por el siguiente algoritmo.

Algoritmo:
Paso 1. Generar $U = u$ y $V = v$ de $U(0, 1)$.
Paso 2. Sea $i = [nv] + 1$.
Paso 3. Si $u \leq q_i$ entonces regresar $X = i$, de lo contrario regresar $X = a_i$.

En este método, las probabilidades q_i y los valores alias a_i, $i = 1, 2, ..., n$ son puestos en una tabla. Para generar una observación de la variable aleatoria X, es necesario revisar las q_i y aveces las a_i.

El número esperado de accesos a la tabla es
$1 + \frac{1}{n}\sum_{i=1}^{n}(1 - q_i)$.

3.1.3. Método de la urna de alias (Alias urn method)

Peterson y Kronmal (1982) sugieren la siguiente generalización del método del alias el cual reduce el número de accesos a la tabla: Considere la distribución de

probabilidad de n puntos $1, 2, ..., n$ con sus respectivas probabilidades $p_1, p_2, ..., p_n$ y extiende esta a una distribución de $1, 2, ..., n^*$ (donde $n^* > n$) poniendo $p_i = 0$ para $i = n+1, ..., n^*$.

Algoritmo.
Paso 1. Generar $X = x$ uniformemente en $1, ..., n^*$.
Paso 2. Si $x > n$ entonces regresar $X = a_x$.
Paso 3. Generar $U = u$ uniformemente en $(0, 1)$.
Paso 4. Si u$\leq q$, entonces regresar $X = x$ sino regresar $X = a_x$.

En este método el número de accesos a la tabla es
$1 + \frac{1}{n^*} \sum_{i=1}^{n} (1 - q_i) \leq 1 + \frac{n}{n^*}$.

3.1.4. Método de aceptación-complemento

El método de aceptación-complemento es el análogo al método del alias para una muestra de distribuciones continuas (Kronmal y Peterson, 1981). Suponga que las observaciones que se quieren obtener tienen la función de densidad $f_X(x)$ la cual se puede descomponer como $f_X(x) = p\, f_1(x) + (1-p)\, f_2(x)$, $0 \leq p \leq 1$. Además supone que $f_Y(y)$ es una densidad la cual aproxima $f_1(z)$. El método consiste en generar una observación $Y = y$ de f_y y regresar $X = y$ con probabilidad $pf_1(y) / f_Y(y)$. En otro caso se regresa una observación de f_2 Usando una variable aleatoria uniforme U decidir cual densidad usar con el siguiente algoritmo.

Algoritmo.
Paso 1. Generar $Y = y$ de f_Y y $U = u$ de $U(0, 1)$.
Paso 2. Si $u \leq pf_1(y) / f_Y(y)$, entonces regresar $X = y$.
Paso 3. Generar una observación $W = w$ de $f_2(w)$ y regresar $X = w$.

3.1.5. Método de rechazo de mezclas (Mixture rejection method)

El método de rechazo de mezclas, algunas veces llamado técnica de composición-rechazo. Suponga que la densidad de interés, $f_X(x)$, puede ser escrita como una mezcla

$$f_X(x) = p_1 f_1(x) + p_2 f_2(x) + ... + p_n f_n(x).$$

Entonces X puede ser generada por el método de mezcla, y las observaciones necesarias de $f_i(y)$ pueden ser generadas aplicando la técnica de rechazo a observaciones de $f_{Y_i}(y)$ con las probabilidades de aceptación $a_i(y)$ y $P(A_i)$. Entonces

$$f_i(x) = f_{Y_i}(x)\, a_i(x) / P(A_i), \quad i = 1, ..., n,$$

y

$$f_X(x) = \frac{p_1 f_{Y_1}(x) a_1(x)}{P(A_1)} + \frac{p_2 f_{Y_2}(x) a_2(x)}{P(A_2)} + ... + \frac{p_n f_{Y_n}(x) a_n(x)}{P(A_n)}$$

3.1.6. Método de aproximación exacta

El método básico.

La variable de interés X puede ser representada como $X = F_X^{-1}(U)$ donde $U(0,1)$. En el método de aproximación exacta, el cual fué elaborado por Marsaglia (1984), la transformación F_X^{-1} es remplazada por una función de aproximación g y, U es remplazada por una apropiada variable V cuya distribución dependerá de g. Haciendo el cambio de variable, ahora V tiene la función de densidad

$$f_V(v) = f_X[g(v)]\, g'(v), \quad 0 < v < 1,$$

y $X = g(V)$ tiene la distribución que se desea obtener. El método requiere que f_v este acotado en cero en el intervalo unitario. Y se escoge un valor p que satisfaga

$$0 < p \leq p_{opt} \equiv \inf\{f_V(v) : 0 < v < 1\},$$

y la densidad de V es descompuesta en una mezcla de densidades uniformes y una densidad residual:

$$f_V(v) = p + (1-p)\{f_X[g(v)]\, g'(v) - p\}/(1-p).$$

La elección óptima de p es p_{opt}.

Este método asume que se pueden generar observaciones de una densidad residual.

Algoritmo:

Paso 1. Generar una variable aleatoria U.

Paso 2. Si $U \leq p$ entonces regresar $X = g(U/p)$.

Paso 3. Generar V de una densidad residual, y regresar $X = g(V)$.

3.1.7. Método de tiempo de espera

El método de rechazo consiste en generar variables aleatorias $Y_1, Y_2, ..., Y_n$ hasta que se obtiene un valor aceptable; entonces $X = Y_n$ es regresado. El método de tiempo de espera es una generalización donde la muestra continua hasta que algún evento ocurre, y entonces $X = g(Y_1, Y_2, ..., Y_n)$ es regresado donde g es una conveniente transformación. *Ejemplo: Distribución exponencial*

Algoritmo:

Paso1. Inicializar $y = 0$.

Paso 2. Generar $X = x$ de una distribución Poisson estándar.

Paso 3. Si$x = 0$, incrementar y e ir al paso 2.

Paso 4. Generar $U_1 = u_1, ...U_x = u_x$ de $U(0,1)$ y establecer $z = min\{u_1, ..., u_x\}$.

Paso 5. Regresar $X = y + z$.

3.1.8. Método Forsythe

Forsythe (1942) creo un método para generar realizaciones de variables aleatorias cuya función de densidad tenga la forma $f_X(x) = ce^{-g(x)}$ donde $a < x < b$ y $0 \leq g(x) \leq 1$. Donde a y b son finitos, el método es una técnica de rechazo que usa una variable de tiempo de espera para decidir si aceptar o rechazar.

Sean $U_0, U_1, U_2, ...$ una secuencia de variables $U(0,1)$. Entonces la variable de tiempo de espera es el primer subíndice k para la cual la monotonicidad es violada en la siguiente secuencia: $g(a + (b-a)U_0) \geq U_1 \geq U_2 \geq ...$

Algoritmo:
Paso 1. Generar $W = w$ de $U(0,1)$ e inicializar $k = 1$.
Paso 2. Establecer $x = a + (b-a)w$ y calcular $g = g(x)$.
Paso 3. Generar $U = u$ de $U(0,1)$.
Paso 4. Si $g \geq u$ entonces poner $g = u$, incrementar k, e ir al paso 3.
Paso 5. Si k es impar, entonces regresar $X = x$.
Paso 6. Ir al paso 1.

La distribución de X es $f_X(x) = e^{-g(x)}/P(A)$, $a \leq x < b$, donde $P(A) = \int_a^b e^{-g(x)}dx$.

3.1.9. Método de proporciones uniformes

La idea detrás de este método hecho por Kinderman y Mohan (1977), es generar la variable no negativa de interés X por la relación

$$X = V/U$$

donde U y V son las coordenadas de un punto escogido aleatoriamente (i.e. uniformemente) de una región plana R. El área de R debe ser finita, si R es una región con frontera, entonces el punto (U,V) es fácil de generar encerrando la región R en un rectángulo, usando un generador uniforme aleatoriamente seleccionar un punto del rectángulo, y aceptar el punto si este esta en R o rechazarlo en otro caso. algunos otros métodos de generación de puntos pueden ser utilizados cuando la región no tiene frontera o límites.

Supongamos que la región tiene la forma

$$R = \{(u,v) : 0 < u, 0 < v, \quad y \quad u < g(v/u)\},$$

donde $g(x), x > 0$, es una función especificada no negativa (la región es finita y se puede calcular su área de forma exacta). Sea A el área de R, la densidad de (U,V) es $f_{U,V}(u,v) = 1/A$ si $(u,v) \in R$ y 0 en otro caso. Haciendo un cambio de

variables,$x = v/u$ y $y = u$, tenemos que la función de densidad de X es $f_{X;Y}(x,y) = y/A$ en $\{0 < x, 0 < y, \ \ y \leq g(x)\}$.

La densidad marginal de X es $f_X(x) = [g(x)]^2/2A$ y $A = 0{,}5\int [g(x)]^2 dx$. Para generar observaciones dada una función de densidad en el eje real positivo, se puede tomar $g(x)$ proporcional a la raíz cuadrada de la densidad. La *Ejemplo: distribución uniforme (a,b)*

Esta función de densidad puede ser escrita en términos de la función indicadora I del intervalo (a,b) como

$f_X(x) = \frac{1}{b-a} I(x)$

La función g se escogió proporcional a la raíz cuadrada de f_X

$g(x) = I^{1/2}(x) \equiv I(x)$.

La región R es

$R = \{(u,v) : 0 < u, 0 < v, \ y \ u < I(v/u)\}$,
$= \{(u,v) : 0 < u < 1, a < v/u < b\}$,

el cual es un triángulo formado por las líneas $u = 1$, $v = au$ y $v = bu$. Como R se puede encerrar el rectángulo $\{(u,v) : 0 < u < 1, 0 < v < b\}$, por lo que el algoritmo para generar números aleatorios que se distribuyan $U(a,b)$ es

Algoritmo:
Paso 1. Generar $U = u$ y $V = v$ donde $U(0,1)$ y $U(0,b)$, respectivamente.
Paso2. Calcular $x = v/u$.
Paso 3. Si $a < x < b$, regresar $X = x$.
Paso 4. Ir al paso 1.

3.1.10. Funciones de densidad Univariadas

Distribución T-student Esta distribución resulta de dividir una variable aleatoria normal estándar, sobre la raíz cuadrada de una chi-cuadrada previamente dividida por sus grados de libertad. Esto es, si z tiene una distribución normal estándar, y u tiene una distribución chi-cuadrada con k grados de libertad, y si además ambas son independientes, entonces

$x = \frac{z}{\sqrt{(u/k)}}$

tiene una distribución t-student con k grados de libertad. Su función se define como:

$f(x) = \frac{\Gamma[(k+1)/2]}{\Gamma(k/2)\sqrt{k\pi}(1+x_2/k)(k+1)/2}$

Como caso especial de esta distribución tenemos a la Cauchy, la cual resulta cuando $k = 1$, y a la función F, la cual resulta de la raíz cuadrada de una variable t. Cuando k es mayor que 30, se usa directamente la distribución normal.

Esta distribución es de mayor semejanza con la normal estándar, ya que también es simétrica con respecto a cero, y tiene por lo tanto el mismo valor esperado que la normal. Solo varia en una mayor varianza y en una menor kurtosis, diferencias que se atenúan cuando k crece.

La subrutina de Gauss que genera observaciones de dicha variables es:

```
proc rndts(m,n,s);
 local t, x, y, u;
 u=rndns(n,m+1,s);
 x=u[ . ,1];
 y=u[ . ,2:(m+1)];
 t=x./sqrt(sumc((y^2)')/m);
 retp(t);
endp;
```

Distribución F La distribución F resulta del cociente de dos variables chi-cuadradas independientes, dividida cada una por sus grados de libertad. Esto es, si v y u son dos variables independientes con distribución chi-cuadrada, con m y n grados de libertad respectivamente, entonces la variable X, definida como:

$$x = \frac{V/m}{U/n}$$

Tiene una distribución F con m grados de libertad en el numerador y n en el denominador, o sea, $F(m, n)$.

Su función de densidad se expresa

$$f\,(x) = \frac{\Gamma[(m+n)/2]}{\Gamma(m/2)\Gamma(n/2)}\,(m/n)\,m/2\frac{x(m-2)/2}{[1+(m/n)x](m+n)/2}$$

Esta distribución se puede generar a partir de variables normales (el cociente del cuadrado de ellas, dividida cada una por sus grados de libertad).

La subrutina de Gauss que genera observaciones de dicha variables es:

```
proc rndfs(m,k,n,s);
 local rn,x,y,f;
 rn=rndns(m+k,n,s);
 x=sumc(rn[1:m, . ]^2)/m;
 y=sumc(rn[m+1:m+k, . ]^2)/k;
 f=x./y;
 retp(f);
endp;
```

Distribución Tukey Esta distribución es muy importante considerarla, pues con ciertos valores de los parámetros, se asemeja mucho a la normal. La función de distribución original es

$$F^{-1}(U) = \frac{1}{\sigma}(U\sigma - (1-U)\sigma)$$

Donde σ es el parámetro de forma y U una variable $uniforme(0,1)$. Esta distribución ha sido generalizada de manera tal que se incluye un parámetro de escala. Esta nueva distribución se define

$$F^{-1}(U) = \sigma_1 + 1/\sigma_2 (U^{\sigma_3} - (1-U)^{\sigma_4})$$

Donde σ_1 es el parámetro de localización y σ_2 es el parámetro de escala. El mérito de esta familia es su versatilidad en la modelación de datos, y su fácil generación. Las formas más comunes que puede tomar esta distribución son:

- Cuando $\sigma_3 = \sigma_4 = 0$, la densidad de la distribución tiende a una logística.
- Cuando $\sigma_1 = \sigma_3$ y $\sigma_2 = \sigma_4 \longrightarrow 0$, la función de densidad tiende a una densidad exponencial.
- Cuando $\sigma_1 = 0$, $\sigma_2 = 0{,}1975$, $\sigma_3 = 0{,}1349$, la función así obtenida difiere de la normal en solo 0,002.

La función de densidad no es conocida en su forma cerrada.

La subrutina de Gauss que genera observaciones de dicha variables es:

```
proc rndtuks(l1,l2,l3,l4,n,s);
 local u,z;
 u=rndus(n,1,s);
 z=l1+(1/l2)*((u)^l3-(1-u)^l4);
 retp(z);
endp;
```

La subrutina de Gauss que se utilizó para generar las observaciones del vector aleatorio fué la siguiente:

```
proc vector5(m,k,n,s); /*m= grados de libertad de la F y T*/
      local rn, x1, y1, f, t,x2,y2,u2,u,z,un,nm,vector;

      /* distribución F*/
      rn=rndns(m+k,n,s);
     x1=sumc(rn[1:m, . ]^2)/m;
     y1=sumc(rn[m+1:m+k, . ]^2)/k;
     f=x1./y1;

     /* distribución t-student*/
     u2=rndns(n,m+1,s);
```

```
    x2=u2[ . ,1];
     y2=u2[ . ,2:(m+1)];
    t=x2./sqrt(sumc(((y2)^2)')/m);
    /* distribución tukey*/
     u=rndus(n,1,s);
    z=(1/0.1975)*((u)^(0.1349)-(1-u)^(0.1349));
     /* distribución uniforme*/
     un=rndus(n,1,s);
     /* distribución normal*/
     nm=rndns(n,1,s);
    vector = f~t~z~un~nm;
    retp(vector);
endp;
```

3.2. Generación de números aleatorios multivariados.

La generación computacional de números aleatorios multivariados es mucho más compleja que la generación de números aleatorios univariados, esto porque las estructuras de dependencias forzan a generar varios componentes colectivamente en vez de solo uno. En general la complejidad se incrementa. Existen varios métodos para generar números aleatorios multivariados pero los tres más importantes son:

1. El método de distribución condicional
2. El método de rechazo y
3. El método de transformación

3.2.1. El método de distribución condicional.

Una distribución de k-dimensiónes puede ser representada como el producto de k distribuciones condicionales univariadas. Correspondientemente, la generación de variables aleatorias para distribuciones multivariadas puede ser realizada generando en secuencia una observación para cada distribución condicional univariada. Sea X el vector aleatorio cuya densidad se puede factorizar como:

$$f(x_1, x_2, ..., x_K) = f_1(x_1) f_2(x_2/x_1) \cdots f_k(x_k/x_1, ..., x_{k-1})$$

El método de generación es descrito en el siguiente algoritmo.

Algoritmo:
Paso 1. Generar $X_1 = x_1$ de la densidad marginal $f_1(x_1)$

Paso 2. Establecer $i = 2$.
Paso 3. Generar $X_i = x_i$ de la densidad condicional univariada
$f_k(x_k/x_1, ..., x_{k-1})$
Paso 4. Sí $i = k$, entonces $X = (x_1, ..., x_k)'$
Paso 5. Incrementar i a $i + 1$
Paso 6. Ir al paso 3.

Ejemplo: La distribución normal multivariada

Supongamos que X sigue la distribución normal multivariada no-singular con vector de medias $\mu = (\mu_1, ..., \mu_k)'$ y matriz de covarianzas (positiva definida) $\Sigma = (\sigma_{ij})$. Además las entradas de la diagonal de Σ son $\sigma_i^2 = \sigma ii$. Para i = 1, . . . , k, definimos $X(i) = (X1, ..., Xi)'$ como el vector de las primeras i componentes de X. El vector media de $X(i)$ es $\mu_{(i)} = (\mu_1, ..., \mu_i)'$ y la matriz de covarianzas de $X_{(i)}$, esta dada por

$$\sum_{ii} = \begin{pmatrix} \sigma_{11} & . & . & . & \sigma_{1i} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ \sigma_{i1} & . & . & . & \sigma_{ii} \end{pmatrix}$$

que es una submatriz de Σ. El vector de covarianzas de X_{i+1} con $X_1, ..., X_i$ esta dado por $\sigma_{(i)} = (\sigma_{1,i+1}, ..., \sigma_{i,i+1})'$.

La distribución marginal de $X_|$ es una normal univariada con media μ_1 y varianza σ_1^2; mientras que para $i = 1, ..., k - 1$, la distribución condicional de X_{i+1}, dadas $X_1, X_2, ..., X_i$,es una normal univariada con media

$$\overline{\mu}_{i+1} = \mu_{i+1} + \sigma'_{(i)} \Sigma_{ii}^{-1} \left(X_{(i)} - \mu_{(i)} \right)$$

y varianza

$$\overline{\sigma}_{i+1}^2 = \sigma_{i+1}^2 - \sigma'_{(i)} \Sigma_{ii}^{-1} \sigma_{(i)}$$

Esto nos da el siguiente algoritmo para generar la normal multivariada de k-dimensiones con media μ y matriz de dispersión Σ.

Algoritmo Normal Multivariada¨
Paso 1. Fijar $i = 1$, $\mu = \mu_1$, y $\sigma^2 = \sigma_1^2$.
Paso 2. Generar $X_i = x_i$ de $N(\mu, \sigma^2)$.
Paso 3. Si $i = k$, entonces regresar $X = (x_1, x_2, ..., x_k)'$.
Paso 4. Incrementar i.
Paso 5. Calcular $\mu = \overline{\mu}_i$ y $\sigma^2 = \overline{\sigma}_i^2$.

Paso 6. Ir al paso 2.

Los obstáculos para implementar este algoritmo son: la determinación de la distribución condicional, y la identificación de una conveniente técnica para generar cada una de las distribuciones condicionales.

3.2.2. El método de rechazo.

Este método es poderoso en la generación de variables aleatorias univariadas, además de que es claro que este método no esta restringido a una sola dimensión. En principio, el método de rechazo multivariado es una fuerte generalización del método de rechazo univariado, el cual será presentado en el siguiente algoritmo. El vector aleatorio de interés es X con una densidad multivariada f, y Y con una densidad g es el vector aleatorio cuyas realizaciones son aceptadas o rechazadas para producir las realizaciones de X.

Algoritmo:
Paso 1. Calcular $m = sup_x[f(x)/g(x)]$.
Paso 2. Generar una observación y de g.
Paso 3. Generar $U = u$ para $U(0,1)$.
Paso 4. Sí $u < f(y)/\{mg(y)\}$, entonces $X = y$.
Paso 5. Ir al paso 2.

Una de las principales dificultades de este método es encontrar la función de densidad g que sea lo suficientemente cercana a la densidad multivariada de interés. Para simplificar la generación de la función de densidad de g en el paso 2, se escoge la distribución Y de manera que sus componentes estén independientemente distribuidos. Una posible solución es que Y tenga componentes independientes Y_i con la misma distribución que sus correspondientes componentes en X. Pero ni siquiera esta solución nos quita las dificultades de este método, pues, primero con esta solución no se asegura que la constante m que aparece en el primer paso sea finita, además, con componentes independientes, el respaldo de Y es un hipercubo en R^k y puede ser mucho más grande que el respaldo de X, creando significantes ineficiencias en el algoritmo. Finalmente si la dependencia de los componentes de X es muy fuerte, no puede esperarse que su función de densidad este bien aproximada por una densidad con marginales independientes.

3.2.3. El método de transformación

Si X y Y son vectores aleatorios que satisfacen la relación funcional $Y = h(X)$, entonces las observaciones de Y pueden ser generadas primero generando las observaciones de X y entonces aplicar la transformación $Y = h(X)$.

Una importante aplicación de las transformaciones lineales es el generar observaciones multivariadas con vector media μ y matriz de dispersión Σ especificados, pero con una distribución desconocida o no especificada.

Supongamos, entonces, que el vector aleatorio Y de k-dimensiones es especificado con media μ y matriz de dispersión Σ. Por simplicidad, tomamos Σ como positiva definida. De acuerdo con la descomposición de Cholesky, Σ puede ser escrito como

$$\Sigma = HH',$$

Donde H es una matriz inferior k por k, ver George y Liu (1981) o Golub (1969). Ahora sea X cualquier vector aleatorio k-dimensional con componentes independientes con medias cero y varianzas uno. Entonces $E[X] = 0$ y $E[XX\prime] = I$, y con la transformación

$$Y = HX + \mu,$$

Se genera Y con los momentos deseados, Por supuesto, X es relativamente fácil de generar pues sus componentes son independientes. Hull (1977) describe un método que involucra transformaciones no lineales en el cual genera vectores aleatorios con distribuciones marginales prescritas.

Si recordamos que la familia de distribuciones normales multivariadas es mapeada en ella misma por transformaciones afines, entonces la anterior transformación nos da un método alternativo para generar observaciones normales multivariadas: simplemente tomando los componentes de X como una normal estándar. En realidad un estudio cercano de la descomposición de Cholesky revela que el método de generación de normales multivariadas no es fundamentalmente diferente del método que utiliza distribuciones condicionales. Ver Chung, Kailath y Lev-Ari (1987) y Cheng (1985)

3.2.4. Funciones de densidad multivariadas

La distribución normal multivariada.(no estandarizada) La subrutina de Gauss que genera observaciones de dicha variables es:

```
/*
**
** random (unstandardized) normal multivariada no estandarizada
**
** y = rndmnorm(mu,vc,n);
**
** inputs: mu = kx1 media
**         vc = kxk matriz de varianzas y covarianzas
```

```
**          n = número de simulaciones
**
** output: y = nxk matriz de variables aleatorias multivariada dependientes
**               cada renglón de y es una simulación 1xk
**
*/
proc rndmnorm (mu, vc, n);
      local k, c, r, i, t, vcd, ad, a, res;
      k=rows(mu);
      c=cols(mu);
      r=rows(vc);

      if (( r/=k ) .or (cols (vc) /=k)) .and (r/=1);
           errorlog "rndmn: mu debe ser de dimensión k*1, y vc k*k o un escalar";
           end;
      endif;

      if n<1;
           errorlog "rndmn: el número de simulaciones debe ser >=1 ";
           end;
      endif;

      if vc==0;
      retp(mu' .*ones (n,1));
      endif;
      i=sumc(dotfeq(vc, 0)) .==r;

      if sumc(i) == 0;
           a=chol(vc) ' ;
      else;
           t=delif( eye(r), i ) ;
           vcd=t*vc*t' ;
           ad=chol(vcd) ;
           a=t' ad*t ;
      endif;

      res= (mu+a*rndn (k, n)) ' ;
      retp(res);
endp;
```

La distribución t. La subrutina de Gauss que genera observaciones de dicha variables es:

```
/*
**variable aleatoria t-multivariada
** y = rndmt(mu,vc,df,n);
**
** inputs: mu = kx1 media
**         vc = kxk matriz de varianzas
**         df = escalar de grados de libertad
**          n = número de simulaciones
**
** output: y = nxk matriz de Variables aleatorias T multivariadas
** cada renglón de y es una simulación 1xk
**
** El método que se utilizó para generar las variables aleatorias fué la descomposición cholsky
*/
proc rndmt(mu,vc,df,n);
    local k,c,r,i,t,vcd,ad,a,res;
    k=rows(mu);
    c=cols(mu);
    r=rows(vc);
    if ((r/=k).or(cols(vc)/=k)).and(r/=1);
        errorlog "rndmt: mu debe ser kx1, y vc kxk o un escalar";
        stop;
    endif;
    if n<1;
        errorlog "rndmt: el número de simulaciones debe ser mayor o igual a 1

        stop;
    endif;
    if c/=1 and c/=n;
        errorlog "rndmt: mu debe ser kxn o kx1";
        end;
    endif;
    if vc==0;
        retp(mu'.*ones(n,1));
    endif;
    i=sumc(dotfeq(vc,0)).==r;
```

```
    if sumc(i)==0;
        a=chol(vc)';
    else;
        t=delif(eye(r),i);
        vcd=t*vc*t';
        ad=chol(vcd);
        a=t'ad*t;
    endif;
    res=(mu+a*(rndn(k,n).*sqrt(df./rndchi(1,n,df))))';
    retp(res);
endp;
```

La distribución Wishart Si X sigue una distribución normal k-variable, $N_k(0, \Sigma)$, y si $X_1, ..., X_n$ son realizaciones independientes de X entonces la matriz aleatoria k por k

$W = \sum_{i=1}^{n} X_i X_i'$

Se dice que tiene una distribución Wishart $W_k(n, \Sigma)$ con parámetros n y Σ. La generación de una sola observación de $W_k(n, \Sigma)$ es realizada aplicando las trasformaciones cuadráticas de n observaciones independientes $X_1, ..., X_n$ de $N_k(0, \Sigma)$.

La subrutina de Gauss que genera observaciones de dicha variables es:

```
proc rndwis (vc, n, m); /* vc= matriz de var-cov,
                           n= grados de libertad
                           m= número de simulaciones*/
    local k, c, r, i, t, vcd, ad, a, res, wis, j, mu,q,wis1;
    q=1;
    j= 1;

    r=rows(vc);
    mu = zeros(r,1);
    k=rows(mu);
    c=cols(mu);
    if (( r/=k ) .or (cols (vc) /=k)) .and (r/=1);
       errorlog "rndmn: mu debe ser de dimensión k*1, y vc k*k o un escalar";
       end;
    endif;
    if n<1;
       errorlog "rndmn: el número de simulaciones debe ser >=1 ";
       end;
     endif;
     wis = zeros (r, r);
```

```
    do while j<=n;
       i=sumc(dotfeq(vc, 0)) .==r;
        if sumc(i) == 0;
           a=chol(vc) ' ;
        else;
              t=delif( eye(r), i ) ;
              vcd=t*vc*t' ;
              ad=chol(vcd) ;
              a=t' ad*t ;
         endif;
         res= (mu+a*rndn (k, 1)) ' ;
         wis = wis + (res' * res);
        j = j+1;
     endo;
     do while q <= (m-1);
          wis1 = zeros (r, r);
          i=sumc(dotfeq(vc, 0)) .==r;
           if sumc(i) == 0;
             a=chol(vc) ' ;
           else;
                t=delif( eye(r), i ) ;
                vcd=t*vc*t' ;
               ad=chol(vcd) ;
                a=t' ad*t ;
           endif;
           res= (mu+a*rndn (k, 1)) ' ;
           wis1 = wis1 + (res' * res);
           wis=wis | wis1;
           q = q+1;
       endo;
    retp(wis);
endp;
;
```

La distribución Dirichlet Suponga que $X_1, ..., X_{k+1}$ son variables aleatorias independientes de una distribución gamma, $X_i \sim \Gamma(\alpha_i, 1)$, con parámetros α_1, $i = 1, ..., k+1$. Definimos $X = \sum_{i=1}^{k+1} X_i$. Entonces el vector aleatorio $Y = (Y_1, ..., Y_k)'$ con $Y_i = X_i/X$, $i = 1, ...k$, sigue una distribución Dirichlet k-dimensional con parámetros $\alpha_1, ..., \alpha_{k+1}$. Note que la distribución beta univariada es un caso especial cuando $k = 1$.

```
proc rndds(a,n,k); /*a= parámetro de las gamas
```

```
                    n=dimensión del vector
                    k= número de simulaciones*/
    local g,u,m,d,ad,u1,g1,i,dir,sum;
   i=1;
    m=ones(1,n);
    u=rndu (n, 1);
    g=-ln(prodc(u'))/a;      /* n gamas con parámetros: 1, a*/
    u1=rndu (1, 1);
    g1=-ln(prodc(u1'))/a;       /* gama con parámetros: 1, a*/
    ad= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1, a*/
    dir=(1/ad)*g;
    dir=(dir)';
    do while i<= (k-1);
       m=ones(1,n);
        u=rndu(n, 1);
        g=-ln(prodc(u'))/a; /* n gamas con parámetros: 1, a*/
        u1=rndu(1, 1);
        g1=-ln(prodc(u1'))/a; /* gama con parámetros: 1, a*/
        sum= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1, a*/
        d=(1/sum)*g;
        d=d';
        dir= dir | d;
        i=i+1;
    endo;
    retp(dir);
endp;
```

La distribución mezcla de dos normales multivariadas Esta distribución resulta de una combinación lineal de dos variables aleatorias normales multivariadas. Esta distribución tiene dos vectores de medias, cada vector corresponde al valor de la media de cada normal multivariada y lo mismo sucede con las matrices de varianzas y covarianzas.

```
/*
**
** random (unstandardized) mezcla de normales multivariadas no estandarizadas
**
** y = rndmn(mu,vc,mu1,vc1,a,n);
**
** inputs: mu = kx1 media
** vc = kxk matriz de varianzas y covarianzas
```

```
** mu1=kx1 media de la segunda normal
** vc1=kxk matriz de varianzas y covarianzas de la segunda normal
** a= factor de la mezcla
** n = número de simulaciones
**
** output: y = nxk matriz de variables aleatorias multivariada dependientes
** de mezcla de normales, cada renglón de y es una simulación 1xk
**
*/
proc rndmmn(mu,vc,mu1,vc1,b,n);
      local k,k1,c,c1,r,r1,i,i1,t,t1,vcd,vcd1,ad,ad1,a,a1,res,res1,mez;
      k=rows(mu);
      k1=rows(mu1);
      c=cols(mu);
      c1=cols(mu1);
      r=rows(vc);
      r1=rows(vc1);

      if c/=c1 and r/=r1 and k/=k1;
            errorlog "rndmn: mu y vc deben ser de la misma dimensión de mu1 y
vc1";
            end;
      endif;

      if b<0 or b>1;
            errorlog "b debe pertenecer al intervalo [0, 1]";
            end;
      endif;

      if ((r/=k).or(cols(vc)/=k)).and(r/=1);
            errorlog "rndmn: mu debe ser kx1, y vc kxk o un escalar";
            end;
      endif;

      if ((r1/=k1).or(cols(vc1)/=k1)).and(r1/=1);
            errorlog "rndmn: mu1 debe ser kx1, y vc1 kxk o un escalar";
            end;
      endif;

      if n<1;
            errorlog "rndmn: el número de simulaciones debe ser >=1 ";
            end;
      endif;

      if c/=1 and c/=n;
```

```
        errorlog "rndmn: mu debe ser kxn o kx1";
        end;
    endif;

    if c1/=1 and c1/=n;
        errorlog "rndmn: mu1 debe ser kxn o kx1";
        end;
    endif;

    if vc==0;
        retp(mu'.*ones(n,1));
    endif;

    if vc1==0;
        retp(mu1'.*ones(n,1));
    endif;

/* generando la primer normal multivariada*/
    i=sumc(dotfeq(vc,0)).==r;
    if sumc(i)==0;
        a=chol(vc)';
    else;
        t=delif(eye(r),i);
        vcd=t*vc*t';
        ad=chol(vcd);
        a=t'ad*t;
    endif;
    res=(mu+a*rndn(k,n))';
/* generando la segunda normal multivariada*/
    i1=sumc(dotfeq(vc1,0)).==r1;
    if sumc(i1)==0;
        a1=chol(vc1)';
    else;
        t1=delif(eye(r1),i1);
        vcd1=t1*vc1*t1';
        ad1=chol(vcd1);
        a1=t1'ad1*t1;
    endif;
    res1=(mu1+a1*rndn(k1,n))';

    mez=b*res+(1-b)*res1;
    retp(mez);
endp;
```

Capítulo 4

RESULTADOS

En esta sección se presentan los resultados que se obtuvieron del estudio de Monte Carlo sobre el poder de las pruebas de normalidad multivariadas, las pruebas que se analizaron como se dijo en la sección dos son:

1. Prueba utilizando solo terceros y cuartos momentos puros

$ALM_p = \sum_{i=1}^{p} Q_{iii}^2/var\{\sqrt{b_1}\} + \sum_{i=1}^{p} (R_{iiii} - E\{b_2\})^2 /var\{b_2\} \tilde{}^{A} \chi_{2p}^2.$

2. Prueba ajustada de falta de simetría

$ALM_{1,p} = \sum_{i=1}^{p} Q_{iii}^2/var\{\sqrt{b_1}\} \tilde{}^{A} \chi_p^2.$

y

3. Prueba ajustada de kurtosis

$ALM_{2,p} = \sum_{i=1}^{p} (R_{iiii} - E\{b_2\}^2/var\{b_2\} \tilde{}^{A} \chi_p^2.$

Se utilizaron seis distribuciones diferentes: normal multivariada, t-multivariada, mezcla de dos normales multivariadas, Wishart, Dirichlet y un vector aleatorio, cuyas entradas tienen una distribución univariada, como la distribución T-student, F, Tukey, mezcla de dos normales y la distribución uniforme, para el caso de las distribuciones en las que fué necesario especificar la matriz de varianzas y covarianzas, esta se tomo como la identidad para garantizar que la matriz fuera positiva semidefinida.

Las pruebas se realizaron sobre seis tamaños de muestras diferentes (10, 20, 50, 100, 200, 800), con distintas dimensiones del vector aleatorio (1, 2, 3, 4, 5) y con valores de significancia del 90 %, estos valores como se menciono se tomaron de la tabla 1 Urzúa (1997).

En este estudio la hipótesis nula H_0 representa normalidad, por lo que el poder estimado de cada prueba se obtuvo dividiendo el número de veces que H_0 fué rechazada, sobre el número de veces que se repitió el experimento, en este caso 10000.

En todas las distribuciones no normales multivariadas la prueba de Tabla 4.6 mayor poder estimado fué aquella que rechazo más veces la hipótesis nula, mientras que para la distribución normal multivariada la mejor prueba fué aquella que acepto más veces la hipótesis nula.

En la tabla 1. Se muestran los resultados que se obtuvieron para el caso de la distribución normal multivarada, como puede verse las mejores pruebas son la $ALM_{1,p}$

para tamaños de muestras pequeñas y la ALM_P para tamaños de muestras grandes en los casos de $p = 2, 3, 5$, mientras que en los casos de la distribución mezcla de dos normales multivariadas y la distribución t-multivariada la mejor prueba en casi todos los casos es la $ALM_{2,p}$.(ver tablas 2 y 3), en la distribución Dirichlet la mejor prueba fué la $ALM_{1,P}$, para los casos de $p = 1, 2, 3, 4$, mientras que para el caso de $p = 1$, la mejor resultó ser la prueba $ALM_{2,p}$ (ver tabla 4), para la distribución Wishart las mejores pruebas para $p = 1, 2, 3, 4$ fueron las pruebas $ALM_{2,p}$ ALM_p , mientras que en $P = 1$, la mejor resultó ser la prueba $ALM_{1,p}$ (ver tabla 5), para el caso del vector aleatorio todas las pruebas resultaron ser buenas, con una pequeña ventaja de las pruebas $ALM_{1,p}$ y ALM_p (ver tabla 6).

p	n	$ALM_{1,P}$	$ALM_{2,p}$	ALM_p
1	10	0.073700000	0.10120000	0.093300000
	20	0.086500000	0.099900000	0.095400000
	50	0.094300000	0.10230000	0.10390000
	100	0.097700000	0.11070000	0.10000000
	200	0.10520000	0.10110000	0.10560000
	800	0.10240000	0.11400000	0.10550000
2	10	0.062800000	0.10040000	0.087200000
	20	0.075300000	0.10210000	0.087800000
	50	0.092600000	0.10080000	0.094600000
	100	0.096400000	0.095400000	0.093000000
	200	0.097200000	0.096100000	0.098100000
	800	0.097400000	0.10060000	0.092200000
3	10	0.059500000	0.097400000	0.084300000
	20	0.074600000	0.097800000	0.094700000
	50	0.094100000	0.10390000	0.10170000
	100	0.099800000	0.099600000	0.098000000
	200	0.10210000	0.10040000	0.096400000
	800	0.10290000	0.097300000	0.095900000
4	10	0.058900000	0.10220000	0.083300000
	20	0.072800000	0.095100000	0.090100000
	50	0.088800000	0.10190000	0.098000000
	100	0.093100000	0.10130000	0.091000000
	200	0.094700000	0.10010000	0.10270000
	800	0.10060000	0.10110000	0.10390000
5	10	0.050600000	0.096700000	0.077300000
	20	0.078600000	0.10550000	0.098000000
	50	0.089400000	0.099700000	0.093600000
	100	0.092100000	0.098200000	0.094500000
	200	0.10060000	0.099800000	0.097600000
	800	0.10160000	0.10520000	0.10090000

CUADRO 4.1. Distribución normal multivariada, con vector de medias =1 y matriz de varianzas y covarianzas igual a la identidad (esto para garantizar que la matriz sea positiva semidefinida).

p	n	$ALM_{1,P}$	$ALM_{2,P}$	ALM_P
1	10	0.072800000	0.093200000	0.089200000
	20	0.080600000	0.10040000	0.093000000
	50	0.097100000	0.10510000	0.10210000
	100	0.10080000	0.10360000	0.097900000
	200	0.10490000	0.10710000	0.10700000
	800	0.098300000	0.10110000	0.094500000
2	10	0.066600000	0.10620000	0.089900000
	20	0.079700000	0.094500000	0.089600000
	50	0.089300000	0.098700000	0.097400000
	100	0.096800000	0.10340000	0.093900000
	200	0.092000000	0.095900000	0.092900000
	800	0.10280000	0.10180000	0.098300000
3	10	0.056700000	0.094900000	0.082500000
	20	0.079200000	0.099600000	0.091300000
	50	0.094600000	0.10110000	0.099600000
	100	0.094000000	0.098100000	0.098100000
	200	0.098300000	0.095700000	0.097900000
	800	0.10340000	0.093800000	0.097700000
4	10	0.054900000	0.10000000	0.082300000
	20	0.084400000	0.099300000	0.096100000
	50	0.090200000	0.10750000	0.10100000
	100	0.091300000	0.094400000	0.091400000
	200	0.094600000	0.097100000	0.10240000
	800	0.10040000	0.10720000	0.10620000
5	10	0.047300000	0.097600000	0.077800000
	20	0.080000000	0.10590000	0.098100000
	50	0.094200000	0.10790000	0.10080000
	100	0.089200000	0.094800000	0.091900000
	200	0.10290000	0.094100000	0.097000000
	800	0.094200000	0.097700000	0.095200000

CUADRO 4.2. Distribución mezcla de dos normales multivariadas, con vector de medias aleatoria.

p	n	$ALM_{1,P}$	$ALM_{2,P}$	ALM_P
1	10	0.25500000	0.30540000	0.29870000
	20	0.40480000	0.48620000	0.48710000
	50	0.58560000	0.78840000	0.78180000
	100	0.58560000	0.94950000	0.94180000
	200	0.77300000	0.99790000	0.99650000
	800	0.88620000	0.99970000	0.99970000
2	10	0.23790000	0.31740000	0.28870000
	20	0.48370000	0.57700000	0.56540000
	50	0.73070000	0.89800000	0.88580000
	100	0.84870000	0.99100000	0.98890000
	200	0.92320000	0.99960000	0.99950000
	800	0.97710000	1.0000000	1.00000000
3	10	0.19740000	0.26430000	0.24280000
	20	0.50680000	0.60030000	0.59440000
	50	0.80590000	0.94060000	0.93850000
	100	0.91680000	0.99770000	0.99730000
	200	0.96480000	0.99980000	0.99980000
	800	0.99440000	0.99940000	1.000000
4	10	0.16060000	0.23410000	0.20920000
	20	0.48920000	0.56660000	0.56770000
	50	0.85150000	0.95920000	0.95360000
	100	0.94990000	0.99840000	0.99850000
	200	0.98750000	0.99990000	0.99990000
	800	0.99890000	1.0000000	1.000000
5	10	0.11240000	0.18250000	0.16180000
	20	0.47020000	0.55030000	0.54860000
	50	0.87580000	0.96830000	0.96500000
	100	0.96630000	0.99980000	0.99960000
	200	0.99300000	1.0000000	1.0000000
	800	0.99940000	1.0000000	1.0000000

CUADRO 4.3. Distribución t-multivariada, con vector de medias aleatorio.

p	n	$ALM_{1,P}$	$ALM_{2,P}$	ALM_P
1	10	0.031000000	0.072400000	0.027500000
	20	0.016900000	0.22740000	0.006100000
	50	0.0095000000	0.85000000	0.31650000
	100	0.0087000000	0.99830000	0.97100000
	200	0.0051000000	1.0000000	1.0000000
	800	0.0052000000	1.0000000	1.0000000
2	10	0.074200000	0.098800000	0.090600000
	20	0.11010000	0.049200000	0.071400000
	50	0.30320000	0.046500000	0.15180000
	100	0.67880000	0.26780000	0.72190000
	200	0.96770000	0.79950000	0.999500000
	800	1.0000000	1.0000000	1.0000000
3	10	0.17190000	0.19160000	0.18850000
	20	0.40490000	0.20320000	0.29150000
	50	0.88720000	0.17310000	0.69420000
	100	0.99890000	0.13650000	0.99170000
	200	1.0000000	0.10260000	1.0000000
	800	1.0000000	0.085600000	1.0000000
4	10	0.29240000	0.30920000	0.30880000
	20	0.68100000	0.36380000	0.51660000
	50	0.99450000	0.50060000	0.95670000
	100	1.0000000	0.61130000	1.0000000
	200	1.0000000	0.78380000	1.0000000
	800	1.0000000	0.99750000	1.0000000
5	10	0.36400000	0.37940000	0.38520000
	20	0.84710000	0.54220000	0.71130000
	50	0.99970000	0.75250000	0.99670000
	100	1.0000000	0.91140000	1.0000000
	200	1.0000000	0.98950000	1.0000000
	800	1.0000000	1.0000000	1.0000000

CUADRO 4.4. Distribución dirichlet, con parámetro aleatorio.

p	n	$ALM_{1,P}$	$ALM_{2,P}$	ALM_P
1	10	0.62680000	0.46700000	0.55920000
	20	0.93670000	0.69600000	0.88360000
	50	0.99970000	0.94760000	0.99990000
	100	1.0000000	0.99720000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
2	10	0.72440000	0.75240000	0.76010000
	20	0.93580000	0.95850000	0.97090000
	50	0.99970000	0.99980000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
3	10	0.82680000	0.88780000	0.88730000
	20	0.97550000	0.99470000	0.99580000
	50	0.99990000	1.0000000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
4	10	0.88870000	0.95690000	0.95340000
	20	0.98840000	0.99950000	0.99960000
	50	0.99990000	1.0000000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
5	10	0.92980000	0.98550000	0.98050000
	20	0.99360000	1.0000000	1.0000000
	50	1.0000000	1.0000000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000

CUADRO 4.5. Distribución Wishart con grados de libertad aleatorios.

p	n	$ALM_{1,P}$	$ALM_{2,P}$	ALM_P
1	10	0.79820000	0.64440000	0.74300000
	20	0.98710000	0.89460000	0.97460000
	50	1.0000000	0.99730000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
2	10	0.84430000	0.81090000	0.83340000
	20	0.99200000	0.98140000	0.99230000
	50	1.0000000	1.0000000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
3	10	0.79280000	0.77980000	0.78670000
	20	0.98570000	0.97510000	0.98800000
	50	1.0000000	0.99990000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
4	10	0.75180000	0.76560000	0.76460000
	20	0.97950000	0.76560000	0.98200000
	50	1.0000000	1.0000000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000
5	10	0.71300000	0.74100000	0.73660000
	20	0.97390000	0.96750000	0.97680000
	50	1.0000000	0.99980000	1.0000000
	100	1.0000000	1.0000000	1.0000000
	200	1.0000000	1.0000000	1.0000000
	800	1.0000000	1.0000000	1.0000000

CUADRO 4.6. Vector aleatorio con grados de libertad aleatorios, para el caso de la distribución F y la t-student.

Capítulo 5

CONCLUSIONES

Después evaluar el poder de las pruebas ALM_P, $ALM_{1,P}$ y $ALM_{2,P}$ con valores de significancia de $\alpha = 10\,\%$ no se puede afirmar cual de las tres pruebas es la mejor, pues como se analizó en los resultados a veces la prueba $ALMp$ es la mejor como sucede en las distribuciones normal multivariada, Wishart y el vector aleatorio o como la prueba $ALM_{1,P}$ que es buena en los casos de las distribuciones normal multivariada Dirichlet y el vector aleatorio multivariado, y por último la prueba $ALM_{2,P}$ que resulto ser la mejor para las distribuciones mezcla de dos normales multivariadas, t-multivariada, Wishart y el vector aleatorio (ver tablas 1-6), analizando otros factores como el tamaño de las muestras y la dimensión del vector aleatorio, los resultados son los siguientes:

Respecto al tamaño de las muestras las pruebas que son más eficientes para muestras pequeñas son la $ALM_{1,P}$ y la $ALM_{2,P}$. (Ver tablas 1-6).

Respecto al tamaño del vector no se puede decir algo concluyente, pues los resultados dependen del tipo de distribución que se analice, por ejemplo cuando $p = 1$, las mejores pruebas son la $ALM_{1,p}$ para la distribución Wishart y el vector aleatorio, mientras que la $ALM_{2,p}$ es buena en los casos de la mezcla de dos normales multivariadas, t-multivariada y Dirichlet.

Para $p = 2$, la prueba $ALM_{1,p}$ es buena para la distribución Dirichlet y el vector aleatorio, la prueba $ALM_{2,p}$ en el caso de las distribuciones mezcla de dos normales multivariadas y t-multivariada y la ALM_p en el caso de la distribución Wishart y el vector aleatorio.

Para $p = 3$, la prueba $ALM_{1,p}$ es buena en el caso de la distribución Dirichlet y el vector aleatorio, $ALM_{2,p}$ para la distribución t-multivariada y la distribución Wishart, y la ALM_p para las distribuciones Wishart y el vector aleatorio.

Para $p = 4$, los resultados son como sigue: $ALM_{1,p}$ para las distribuciones normal multivariada y Dirichlet, $ALM_{2,p}$ para las distribuciones t-multivariada, mezcla de normales, Wishart y el vector aleatorio, y ALM_p para las distribuciones Wishart y el vector aleatorio.

Y por último para p=5, la prueba $ALM_{1,p}$ es eficiente para los casos de distribuciones como Dirichlet y Wishart, $ALM_{2,p}$ para las distribuciones mezcla de normales,

t-multivariada y Wishart, y para ALM_p la distribución Wishart y el vector aleatorio. (Ver tablas 1-6)

Por todo lo anterior como se dijo al principio no se puede afirmar que prueba es mejor, pues esto como se mostró depende de la distribución que se este analizando y en algunos casos también de la dimensión del vector y el tamaño de la muestra. Pero lo que si se puede afirmar es que los resultados convergen a la solución, esto es, a medida que el tamaño de la muestra aumenta el resultado refleja mejor la realidad. (i.e. la existencia o no existencia de normalidad)

Apéndice A

PROGRAMAS

A.1. Pruebas ALM1

A.1.1. Normal multivariada

```
/* para p=5, normal multivariada*/
/*prueba ALM1,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\nm15.doc reset;
let nsize=10 20 50 100 200 800;
let sig=9.83, 9.57, 9.45, 9.44, 9.31, 9.27;
jj=1;
mu={1,1,1,1,1};
vc=eye(5);
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
       j=1;
      do while j<=10000;
              x=rndmn(mu,vc,nsize[jj]);
              a[j,1]=alm15(x);
              if a[j,1] >= sig[jj,1];
                 b[jj,1]=b[jj,1]+1;
              endif;
              j=j+1;
       endo;
       b[jj,1]=b[jj,1]/10000;
       output on;
       print "PARA N=";;nsize[jj];
       print "FRACCION =";;b[jj];
       output off;
jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm15(x);
        local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm15;
         p=5;
```

```
        n=rows(x);
        vb1=(6*n-2)/((n+1)*(n+3));
        vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
        eb2=3*(n-1)/(n+1);
        s=moment(x-meanc(x)',0)/rows(x);
        {va,ve}=eighv(s);
        g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
        y=(x-meanc(x)')*g;
        alm15=sumc(meanc(y^3)^2)/vb1;
        retp(alm15);
endp;

proc rndmn(mu,vc,n);
        local k,c,r,i,t,vcd,ad,a,res;
        k=rows(mu);
        c=cols(mu);
        r=rows(vc);
        if ((r/=k).or(cols(vc)/=k)).and(r/=1);
           errorlog "rndmn: mu must be kx1, and vc kxk or scalar";
           end;
        endif;
        if n<1;
          errorlog "rndmn: number of simulations must be >=1 ";
          end;
         endif;
        if c/=1 and c/=n;
          errorlog "rndmn: mu must be kxn or kx1";
           end;
         endif;
         if vc==0;
           retp(mu'.*ones(n,1));
         endif;
         i=sumc(dotfeq(vc,0)).==r;
         if sumc(i)==0;
           a=chol(vc)';
         else;
           t=delif(eye(r),i);
           vcd=t*vc*t';
           ad=chol(vcd);
           a=t'ad*t;
         endif;
         res=(mu+a*rndn(k,n))';
```

```
        retp(res);
endp;
```

A.1.2. Dirichlet

```
/* para p=5, dirichlet*/
/* prueba ALM1,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\dn15.doc reset;
let nsize=10 20 50 100 200 800;
let sig=9.83, 9.57, 9.45, 9.44, 9.31, 9.27;
jj=1;
s1=1;
aa=rndus(1,1,s1);
nn=5;
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
       j=1;
        do while j<=10000;
                x= rndds(aa,nn,nsize[jj]);
                a[j,1]=alm15(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
              print "PARA N=";;nsize[jj];
              print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm15(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm15;
            p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n--3)*(n+5));
```

```
                eb2=3*(n-1)/(n+1);
                s=moment(x-meanc(x)',0)/rows(x);
                {va,ve}=eighv(s);
                g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
                y=(x-meanc(x)')*g;
                alm15=sumc(meanc(y^3)^2)/vb1;
                retp(alm15);
    endp;
    proc rndds(aa,nn,k); /*aa= parámetro de las gamas
                nn=dimensión del vector
                k= número de simulaciones*/
                local g,u,m,d,ad,u1,g1,i,dir,sum;
               i=1;
                m=ones(1,nn);
                u=rndu (nn, 1);
                g=-ln(prodc(u'))/aa; /* n gamas con parámetros: 1, aa*/
                u1=rndu (1, 1);
                g1=-ln(prodc(u1'))/aa; /* gama con parámetros: 1, aa*/
                ad= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1, aa*/
                dir=(1/ad)*g;
                dir=(dir)';
                do while i<= (k-1);
                        m=ones(1,nn);
                        u=rndu(nn, 1);
                        g=-ln(prodc(u'))/aa; /* n gamas con parámetros: 1, aa*/
                        u1=rndu(1, 1);
                        g1=-ln(prodc(u1'))/aa; /* gama con parámetros: 1, aa*/
                         sum= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1,
aa*/
                        d=(1/sum)*g;
                        d=d';
                        dir= dir | d;
                        i=i+1;
                endo;
                retp(dir);
    endp;
```

A.1.3. Mezcla de dos normales multivariadas

```
/* para p=5, mezcla de dos normales multivariadas */
/*prueba ALM1,p*/
```

```
new ,65520;
output file = c:\bárbara\tesis\resultados\mz15.doc reset;
let nsize=10 20 50 100 200 800;
let sig=9.83, 9.57, 9.45, 9.44, 9.31, 9.27;
jj=1;
s1=1;
s2=2;
s3=3;
mu=rndns(5,1,s1);
vc=eye(5);
mu1=rndns(5,1,s2);
vc1=eye(5);
mz= rndus(1,1,s3);
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndmmn(mu,vc,mu1,vc1,mz,nsize[jj]);
                a[j,1]=alm15(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm15(x);
          local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm15;
           p=5;
           n=rows(x);
           vb1=(6*n-2)/((n+1)*(n+3));
           vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
           eb2=3*(n-1)/(n+1);
           s=moment(x-meanc(x)',0)/rows(x);
```

```
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm15=sumc(meanc(y^3)^2)/vb1;
            retp(alm15);
endp;
/*
** random (unstandardized) mezcla de normales multivariadas
** y = rndmmn(mu,vc,mu1,vc1,mz,n);
** inputs: mu = kx1 media
** vc = kxk matriz de varianzas y covarianzas
** mu1=kx1 media de la segunda normal
** vc1=kxk matriz de varianzas y covarianzas de la segunda normal
** mz= factor de la mezcla
** n = número de simulaciones
** output: y = nxk matriz de variables aleatorias multivariada dependientes
** de mezcla de normales, cada renglon de y es una simulación 1xk
*/
proc rndmmn(mu,vc,mu1,vc1,mz,n);
            local k,k1,c,c1,r,r1,i,i1,t,t1,vcd,vcd1,ad,ad1,a,a1,res,res1,mez;
            k=rows(mu);
            k1=rows(mu1);
            c=cols(mu);
            c1=cols(mu1);
            r=rows(vc);
            r1=rows(vc1);
            if c/=c1 and r/=r1 and k/=k1;
               errorlog "rndmn: mu y vc deben ser de la misma dimensión de mu1 y
vc1";
               end;
            endif;
            if b<0 or b>1;
               errorlog "b debe pertenecer al intervalo [0, 1]";
               end;
            endif;
            if ((r/=k).or(cols(vc)/=k)).and(r/=1);
               errorlog "rndmn: mu debe ser kx1, y vc kxk o un escalar";
               end;
            endif;
            if ((r1/=k1).or(cols(vc1)/=k1)).and(r1/=1);
               errorlog "rndmn: mu1 debe ser kx1, y vc1 kxk o un escalar";
```

```
            end;
        endif;
        if n<1;
            errorlog "rndmn: el número de simulaciones debe ser >=1 ";
            end;
        endif;
        if c/=1 and c/=n;
            errorlog "rndmn: mu debe ser kxn o kx1";
            end;
        endif;
        if c1/=1 and c1/=n;
            errorlog "rndmn: mu1 debe ser kxn o kx1";
            end;
        endif;
        if vc==0;
            retp(mu'.*ones(n,1));
        endif;
        if vc1==0;
            retp(mu1'.*ones(n,1));
        endif;
/* generando la primer normal multivariada*/
        i=sumc(dotfeq(vc,0)).==r;
        if sumc(i)==0;
            a=chol(vc)';
        else;
            t=delif(eye(r),i);
            vcd=t*vc*t';
            ad=chol(vcd);
            a=t'ad*t;
        endif;
        res=(mu+a*rndn(k,n))';
/* generando la segunda normal multivariada*/
        i1=sumc(dotfeq(vc1,0)).==r1;
        if sumc(i1)==0;
            a1=chol(vc1)';
        else;
            t1=delif(eye(r1),i1);
            vcd1=t1*vc1*t1';
            ad1=chol(vcd1);
            a1=t1'ad1*t1;
        endif;
```

```
        res1=(mu1+a1*rndn(k1,n))';
        mez=mz*res+(1-mz)*res1;
        retp(mez);
endp;
```

A.1.4. t-multivariada

```
/* para p=5, t - multivariada*/
/*prueba ALM1,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\tn15.doc reset;
let nsize=10 20 50 100 200 800;
let sig=9.83, 9.57, 9.45, 9.44, 9.31, 9.27;
jj=1;
s1=1;
s2=2;
mu=rndns(5,1,s1);
vc=eye(5);
df=trunc(10*rndus(1,1,s2));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=rndmt(mu,vc,df,nsize[jj]);
                a[j,1]=alm15(x);
                if a[j,1] >= sig[jj,1];
                  b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm15(x);
        local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm15;
```

```
        p=5;
        n=rows(x);
        vb1=(6*n-2)/((n+1)*(n+3));
        vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
        eb2=3*(n-1)/(n+1);
        s=moment(x-meanc(x)',0)/rows(x);
        {va,ve}=eighv(s);
        g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
        y=(x-meanc(x)')*g;
        alm15=sumc(meanc(y^3)^2)/vb1;
        retp(alm15);
endp;
/*
** random multivariate t draws
** y = rndmt(mu,vc,df,n);
** inputs: mu = kx1 means
** vc = kxk variance matrix
** df = scalar degrees of freedom
** n = scalar number of simulations
** output: y = nxk matrix of dependent Multivariate T Random Variables
** each row of y is one 1xk simulation
*/
proc rndmt(mu,vc,df,n);
        local k,c,r,i,t,vcd,ad,a,res;
        k=rows(mu);
        c=cols(mu);
        r=rows(vc);
        if ((r/=k).or(cols(vc)/=k)).and(r/=1);
           errorlog "rndmt: mu must be kx1, and vc kxk or scalar";
           stop;
        endif;
        if n<1;
           errorlog "rndmt: number of simulations must be >=1 ";
           stop;
        endif;
        if c/=1 and c/=n;
           errorlog "rndmt: mu must be kxn or kx1";
           end;
        endif;
        if vc==0;
           retp(mu'.*ones(n,1));
```

```
            endif;
            i=sumc(dotfeq(vc,0)).==r;
            if sumc(i)==0;
               a=chol(vc)';
            else;
               t=delif(eye(r),i);
               vcd=t*vc*t';
               ad=chol(vcd);
               a=t'ad*t;
            endif;
            res=(mu+a*(rndn(k,n).*sqrt(df./rndchi(1,n,df))))';
            retp(res);
endp;

/* RNDCHI - Random numbers from a Chi square Distribution
** Usage: x = rndchi(r,c,n)
** Input: R - scalar of row size returned matrix
** C - scalar of Column size returned matrix
** N - matrix of degrees of freedom for Chi square distribution
** (ExE conformable with RxC matrix)
** Output: X - R X C matrix of random numbers ~Chi(N)
*/
proc rndchi(r,c,n) ;
            if not(n > 0) ;
               errorlog "ERROR: RNDCHI - Degrees of Freedom are not positive" ;
               retp(error(99)) ;
            endif ;
            retp(invcfchi(rndu(r,c),n)) ;
endp ;

/* INVCFCHI - Inverse of the Chi squared Cumulative Distribution function
** with n1 degrees of freedom
** Usage: x = invcfchi(p,n)
** Input: P - matrix of percentage points ( 0 < p < 1 )
** N - matrix of degrees of freedom (N > 0) (conformable with X)
** Output: X - matrix of critical values st Pr(x < X) = P and x ~Chi(n)
*/
proc invcfchi(p,n) ;
            local converge,k,tol,t,d,z,x0,x1,f0,df0 ;
            if not(p > 0 and p < 1) ;
               errorlog "ERROR: INVCFCHI - P not in (0,1) " ;
               retp(error(99)) ;
```

```
                endif ;
                if not(n > 0) ;
                  errorlog "ERROR: INVCFCHI - N is not positive" ;
                  retp(error(99)) ;
                endif ;
                tol = 1e-6.*sqrt(n) ;
                clear converge,k ; /*Use Wilson-Hilferty approximation as initial value*/
                t = sqrt(-2*ln(abs((p.>0.5) - p)));
                z = 2.515517 + t.*(0.802853 + t.*0.010328) ;
                d = 1 + t.*(1.432788 + t.*(0.189269 + t.*0.001308)) ;
                z = t - (z./d) ;
                z = (p.>0.5).*z - (p.<=0.5).*z ;
                d = 2./(9.*n) ;
                x0 = n.*(z.*sqrt(d) + 1 - d)^3 ;
                x0 = x0 + (0.1 - x0).*(x0 .<= 0) ;
                p = 1 - p ;
                do until(converge or k > 50) ;
                        f0 = cdfchic(x0,n) ;
                        df0 = missrv(miss(pdfchi(x0,n),0),tol) ;
                        x1 = (x0 - (p-f0)./df0) ;
                        x1 = x1 + (tol - x1).*(x1.<=0) ;
                        converge = abs(x0 - x1) < tol ;
                        x0 = x1 ;
                        k = k + 1 ;
                endo ;
                if not converge ;
                  errorlog "Warning: INVCDFF has not converged " ;
                endif ;
                ndpclex ;
                retp(x0) ;
endp ;

/* PDFCHI - Chi squared Probability density function with n1 degrees
** of freedom
** Usage: D = pdfchi(x,n)
** Input: X - matrix of chi-squared values (X > 0)
** N - matrix of degrees of freedom (N > 0) (conformable with X)
** Output: D - matrix of density of Chi square(n) at X
*/
proc pdfchi(x,n) ;
                if not(n > 0) ;
                    errorlog "ERROR: PDFCHI - N is not positive" ;
```

```
            retp(error(99)) ;
        endif ;
        if not(x > 0) ;
            errorlog "ERROR: PDFCHI - X is not positive" ;
            retp(error(99)) ;
        endif ;
        if n < 100 ;
            retp(exp(-n.*ln(2)./2-ln(gamma(n./2))+(n./2-1).*ln(x)-x./2)) ;
        else ;
            retp(exp(-n.*ln(2)./2-lnfact(n./2-1)+(n./2-1).*ln(x)-x./2)) ;
        endif ;
endp ;
```

A.1.5. Vector aleatorio

```
/* para p=5, vector aleatorio*/
/* prueba ALM1,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\vm15.doc reset;
let nsize=10 20 50 100 200 800;
let sig=9.83, 9.57, 9.45, 9.44, 9.31, 9.27;
jj=1;
s1=1;
s2=2;
m=trunc(10*rndus(1,1,s1));
k=trunc(10*rndus(1,1,s2));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=vector5(m,k,nsize[jj],j+jj);
                a[j,1]=alm15(x);
                if a[j,1] >= sig[jj,1];
                  b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
```

```
            print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm15(x);
        local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm15;
         p=5;
         n=rows(x);
         vb1=(6*n-2)/((n+1)*(n+3));
         vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
         eb2=3*(n-1)/(n+1);
         s=moment(x-meanc(x)',0)/rows(x);
         {va,ve}=eighv(s);
         g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
         y=(x-meanc(x)')*g;
         alm15=sumc(meanc(y^3)^2)/vb1;
         retp(alm15);
endp;

proc vector5(m,k,n,s); /*m= grados de libertad de la F y T*/
        local rn, x1, y1, f, t,x2,y2,u2,u,z,un,nm,vector;
/* distribución F*/
         rn=rndns(m+k,n,s);
         x1=sumc(rn[1:m, . ]^2)/m;
         y1=sumc(rn[m+1:m+k, . ]^2)/k;
         f=x1./y1;
/* distribución t-student*/
        u2=rndns(n,m+1,s);
         x2=u2[ . ,1];
         y2=u2[ . ,2:(m+1)];
         t=x2./sqrt(sumc(((y2)^2)')/m);
/* distribución tukey*/
        u=rndus(n,1,s);
         z=(1/0.1975)*((u)^(0.1349)-(1-u)^(0.1349));
/* distribución uniforme*/
         un=rndus(n,1,s);
/* distribución normal*/
         nm=rndns(n,1,s);
         vector = f~t~z~un~nm;
         retp(vector);
```

```
endp;
```

A.1.6. Wishart

```
/* para p=5, Wishart*/
/* prueba ALM1,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\wn15.doc reset;
let nsize=10 20 50 100 200 800;
let sig=9.83, 9.57, 9.45, 9.44, 9.31, 9.27;
jj=1;
s1=1;
vc=eye(5);
nn=trunc(10*rndus(1,1,s1));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndwis (vc, nn, nsize[jj]);
                a[j,1]=alm15(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;
/* ALMN , PARA P=5*/
proc alm15(x);
          local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm15;
           p=5;
           n=rows(x);
           vb1=(6*n-2)/((n+1)*(n+3));
           vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
           eb2=3*(n-1)/(n+1);
```

```
        s=moment(x-meanc(x)',0)/rows(x);
        {va,ve}=eighv(s);
        g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
        y=(x-meanc(x)')*g;
        alm15=sumc(meanc(y^3)^2)/vb1;
        retp(alm15);
endp;

proc rndwis (vc, nn, m); /* vc= matriz de var-cov,
                            nn= grados de libertad
                            m= número de simulaciones*/
        local k, c, r, i, t, vcd, ad, a, res, wis, j, mu,q,wis1;
        q=1;
        j= 1;
        r=rows(vc);
        mu = zeros(r,1);
        k=rows(mu);
        c=cols(mu);
        if (( r/=k ) .or (cols (vc) /=k)) .and (r/=1);
          errorlog "rndmn: mu debe ser de dimensión k*1, y vc k*k o un escalar";
            end;
        endif;
        if m<1;
            errorlog "rndmn: el número de simulaciones debe ser >=1 ";
            end;
        endif;
        wis = zeros (r, r);
        do while j<=nn;
                i=sumc(dotfeq(vc, 0)) .==r;
                if sumc(i) == 0;
                    a=chol(vc) ' ;
                else;
                    t=delif( eye(r), i ) ;
                    vcd=t*vc*t' ;
                    ad=chol(vcd) ;
                    a=t' ad*t ;
                endif;
                res= (mu+a*rndn (k, 1)) ' ;
                wis = wis + (res' * res);
                j = j+1;
        endo;
        do while q <= (m-1);
```

```
                wis1 = zeros (r, r);
                i=sumc(dotfeq(vc, 0)) .==r;
                if sumc(i) == 0;
                   a=chol(vc) ' ;
                else;
                   t=delif( eye(r), i ) ;
                   vcd=t*vc*t' ;
                   ad=chol(vcd) ;
                   a=t' ad*t ;
                endif;
                res= (mu+a*rndn (k, 1)) ' ;
                wis1 = wis1 + (res' * res);
                wis=wis | wis1;
                q = q+1;
        endo;
        retp(wis);
endp;
```

A.2. Pruebas ALM2

A.2.1. Normal multivariada

```
/* para p=5, normal multivariada*/
/*prueba ALM2,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\nm25.doc reset;
let nsize=10 20 50 100 200 800;
let sig=10.69, 10.16, 9.89, 9.77, 9.53, 9.25;
jj=1;
mu={1,1,1,1,1};
vc=eye(5);
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=rndmn(mu,vc,nsize[jj]);
                a[j,1]=alm25(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
```

```
            j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
            print "PARA N=";;nsize[jj];
            print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm25(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm25;
            p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm25=sumc((meanc(y^4)-eb2)^2)/vb2;
            retp(alm25);
endp;

proc rndmn(mu,vc,n);
            local k,c,r,i,t,vcd,ad,a,res;
            k=rows(mu);
            c=cols(mu);
            r=rows(vc);
            if ((r/=k).or(cols(vc)/=k)).and(r/=1);
               errorlog "rndmn: mu must be kx1, and vc kxk or scalar";
               end;
            endif;
            if n<1;
               errorlog "rndmn: number of simulations must be >=1 ";
               end;
            endif;
            if c/=1 and c/=n;
               errorlog "rndmn: mu must be kxn or kx1";
               end;
```

```
            endif;
            if vc==0;
               retp(mu'.*ones(n,1));
            endif;
            i=sumc(dotfeq(vc,0)).==r;
            if sumc(i)==0;
               a=chol(vc)';
            else;
               t=delif(eye(r),i);
               vcd=t*vc*t';
               ad=chol(vcd);
               a=t'ad*t;
            endif;
            res=(mu+a*rndn(k,n))';
            retp(res);
endp;
```

A.2.2. Dirichlet

```
/* para p=5, Dirichlet*/
/* prueba ALM2,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\dn25.doc reset;
let nsize=10 20 50 100 200 800;
let sig=10.69, 10.16, 9.89, 9.77, 9.53, 9.25;
jj=1;
s1=1;
aa=rndus(1,1,s1);
nn=5;
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndds(aa,nn,nsize[jj]);
                a[j,1]=alm25(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
```

```
        b[jj,1]=b[jj,1]/10000;
        output on;
              print "PARA N=";;nsize[jj];
              print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm25(x);
              local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm25;
              p=5;
              n=rows(x);
              vb1=(6*n-2)/((n+1)*(n+3));
              vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
              eb2=3*(n-1)/(n+1);
              s=moment(x-meanc(x)',0)/rows(x);
              {va,ve}=eighv(s);
              g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
              y=(x-meanc(x)')*g;
              alm25=sumc((meanc(y^4)-eb2)^2)/vb2;
              retp(alm25);
endp;

proc rndds(aa,nn,k); /*aa= parámetro de las gamas
              nn=dimensión del vector
              k= número de simulaciones*/
              local g,u,m,d,ad,u1,g1,i,dir,sum;
             i=1;
              m=ones(1,nn);
              u=rndu (nn, 1);
              g=-ln(prodc(u'))/aa; /* n gamas con parámetros: 1, aa*/
              u1=rndu (1, 1);
              g1=-ln(prodc(u1'))/aa; /* gama con parámetros: 1, aa*/
              ad= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1, aa*/
              dir=(1/ad)*g;
              dir=(dir)';
              do while i<= (k-1);
                    m=ones(1,nn);
                    u=rndu(nn, 1);
                    g=-ln(prodc(u'))/aa; /* n gamas con parámetros: 1, aa*/
                    u1=rndu(1, 1);
```

```
                    g1=-ln(prodc(u1'))/aa; /* gama con parámetros: 1, aa*/
                     sum= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1,
aa*/
                    d=(1/sum)*g;
                    d=d';
                    dir= dir | d;
                    i=i+1;
            endo;
            retp(dir);
    endp;
```

A.2.3. Mezcla de dos normales multivariadas

```
/* para p=5, mezcla de dos normales multivariadas*/
/*prueba ALM2,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\mz25.doc reset;
let nsize=10 20 50 100 200 800;
let sig=10.69, 10.16, 9.89, 9.77, 9.53, 9.25;
jj=1;
s1=1;
s2=2;
s3=3;
mu=rndns(5,1,s1);
vc=eye(5);
mu1=rndns(5,1,s2);
vc1=eye(5);
mz= rndus(1,1,s3);
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndmmn(mu,vc,mu1,vc1,mz,nsize[jj]);
                a[j,1]=alm25(x);
                if a[j,1] >= sig[jj,1];
                  b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
```

```
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm25(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm25;
            p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm25=sumc((meanc(y^4)-eb2)^2)/vb2;
            retp(alm25);
endp;

/*
** random (unstandardized) mezcla de normales multivariadas
** y = rndmmn(mu,vc,mu1,vc1,mz,n);
** inputs: mu = kx1 media
** vc = kxk matriz de varianzas y covarianzas
** mu1=kx1 media de la segunda normal
** vc1=kxk matriz de varianzas y covarianzas de la segunda normal
** mz= factor de la mezcla
** n = number of simulations
** output: y = nxk matriz de variables aleatorias multivariada dependientes
** de mezcla de normales, cada renglon de y es una simulacion 1xk
*/
proc rndmmn(mu,vc,mu1,vc1,mz,n);
            local k,k1,c,c1,r,r1,i,i1,t,t1,vcd,vcd1,ad,ad1,a,a1,res,res1,mez;
            k=rows(mu);
            k1=rows(mu1);
            c=cols(mu);
            c1=cols(mu1);
            r=rows(vc);
```

```
                r1=rows(vc1);
                if c/=c1 and r/=r1 and k/=k1;
                   errorlog "rndmn: mu y vc deben ser de la misma dimensión de mu1 y
vc1";
                   end;
                endif;
                if b<0 or b>1;
                   errorlog "b debe pertenecer al intervalo [0, 1]";
                   end;
                endif;
                if ((r/=k).or(cols(vc)/=k)).and(r/=1);
                   errorlog "rndmn: mu debe ser kx1, y vc kxk o un escalar";
                   end;
                endif;
                if ((r1/=k1).or(cols(vc1)/=k1)).and(r1/=1);
                   errorlog "rndmn: mu1 debe ser kx1, y vc1 kxk o un escalar";
                   end;
                endif;
                if n<1;
                   errorlog "rndmn: el número de simulaciones debe ser >=1 ";
                   end;
                endif;
                if c/=1 and c/=n;
                   errorlog "rndmn: mu debe ser kxn o kx1";
                   end;
                endif;
                if c1/=1 and c1/=n;
                   errorlog "rndmn: mu1 debe ser kxn o kx1";
                   end;
                endif;
                if vc==0;
                   retp(mu'.*ones(n,1));
                endif;
                if vc1==0;
                   retp(mu1'.*ones(n,1));
                endif;
    /* generando la primer normal multivariada*/
                i=sumc(dotfeq(vc,0)).==r;
                if sumc(i)==0;
                   a=chol(vc)';
                else;
```

```
                t=delif(eye(r),i);
                vcd=t*vc*t';
                ad=chol(vcd);
                a=t'ad*t;
            endif;
            res=(mu+a*rndn(k,n))';
/* generando la segunda normal multivariada*/
            i1=sumc(dotfeq(vc1,0)).==r1;
            if sumc(i1)==0;
                a1=chol(vc1)';
            else;
                t1=delif(eye(r1),i1);
                vcd1=t1*vc1*t1';
                ad1=chol(vcd1);
                a1=t1'ad1*t1;
            endif;
            res1=(mu1+a1*rndn(k1,n))';
            mez=mz*res+(1-mz)*res1;
            retp(mez);
endp;
```

A.2.4. t-multivariada

```
/* para p=5, t-multivariada*/
/*prueba ALM2,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\tn25.doc reset;
let nsize=10 20 50 100 200 800;
let sig=10.69, 10.16, 9.89, 9.77, 9.53, 9.25;
jj=1;
s1=1;
s2=2;
mu=rndns(5,1,s1);
vc=eye(5);
df=trunc(10*rndus(1,1,s2));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=rndmt(mu,vc,df,nsize[jj]);
```

```
                a[j,1]=alm25(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm25(x);
        local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm25;
        p=5;
        n=rows(x);
        vb1=(6*n-2)/((n+1)*(n+3));
        vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
        eb2=3*(n-1)/(n+1);
        s=moment(x-meanc(x)',0)/rows(x);
        {va,ve}=eighv(s);
        g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
        y=(x-meanc(x)')*g;
        alm25=sumc((meanc(y^4)-eb2)^2)/vb2;
        retp(alm25);
endp;

/*
** random multivariate t draws
** y = rndmt(mu,vc,df,n);
** inputs: mu = kx1 means
** vc = kxk variance matrix
** df = scalar degrees of freedom
** n = scalar number of simulations
** output: y = nxk matrix of dependent Multivariate T Random Variables
** each row of y is one 1xk simulation
*/
proc rndmt(mu,vc,df,n);
        local k,c,r,i,t,vcd,ad,a,res;
```

```
            k=rows(mu);
            c=cols(mu);
           r=rows(vc);
            if ((r/=k).or(cols(vc)/=k)).and(r/=1);
               errorlog "rndmt: mu must be kx1, and vc kxk or scalar";
               stop;
            endif;
            if n<1;
               errorlog "rndmt: number of simulations must be >=1 ";
               stop;
            endif;
            if c/=1 and c/=n;
               errorlog "rndmt: mu must be kxn or kx1";
               end;
            endif;
           if vc==0;
               retp(mu'.*ones(n,1));
            endif;
            i=sumc(dotfeq(vc,0)).==r;
            if sumc(i)==0;
               a=chol(vc)';
            else;
               t=delif(eye(r),i);
               vcd=t*vc*t';
               ad=chol(vcd);
               a=t'ad*t;
            endif;
            res=(mu+a*(rndn(k,n).*sqrt(df./rndchi(1,n,df))))';
            retp(res);
endp;

/* RNDCHI - Random numbers from a Chi square Distribution
** Usage: x = rndchi(r,c,n)
** Input: R - scalar of row size returned matrix
** C - scalar of Column size returned matrix
** N - matrix of degrees of freedom for Chi square distribution
** (ExE conformable with RxC matrix)
** Output: X - R X C matrix of random numbers ~Chi(N)
*/
proc rndchi(r,c,n) ;
            if not(n > 0) ;
               errorlog "ERROR: RNDCHI - Degrees of Freedom are not positive" ;
```

```
            retp(error(99)) ;
        endif ;
        retp(invcfchi(rndu(r,c),n)) ;
endp ;

/* INVCFCHI - Inverse of the Chi squared Cumulative Distribution function
** with n1 degrees of freedom
** Usage: x = invcfchi(p,n)
** Input: P - matrix of percentage points ( 0 < p < 1 )
** N - matrix of degrees of freedom (N > 0) (conformable with X)
** Output: X - matrix of critical values st Pr(x < X) = P and x ~Chi(n)
*/
proc invcfchi(p,n) ;
        local converge,k,tol,t,d,z,x0,x1,f0,df0 ;
        if not(p > 0 and p < 1) ;
          errorlog "ERROR: INVCFCHI - P not in (0,1) " ;
          retp(error(99)) ;
        endif ;
        if not(n > 0) ;
          errorlog "ERROR: INVCFCHI - N is not positive" ;
          retp(error(99)) ;
        endif ;
        tol = 1e-6.*sqrt(n) ;
        clear converge,k ; /*Use Wilson-Hilferty approximation as initial value*/
        t = sqrt(-2*ln(abs((p.>0.5) - p)));
        z = 2.515517 + t.*(0.802853 + t.*0.010328) ;
        d = 1 + t.*(1.432788 + t.*(0.189269 + t.*0.001308)) ;
        z = t - (z./d) ;
        z = (p.>0.5).*z - (p.<=0.5).*z ;
        d = 2./(9.*n) ;
        x0 = n.*(z.*sqrt(d) + 1 - d)^3 ;
        x0 = x0 + (0.1 - x0).*(x0 .<= 0) ;
        p = 1 - p ;
        do until(converge or k > 50) ;
                f0 = cdfchic(x0,n) ;
                df0 = missrv(miss(pdfchi(x0,n),0),tol) ;
                x1 = (x0 - (p-f0)./df0) ;
                x1 = x1 + (tol - x1).*(x1.<=0) ;
                converge = abs(x0 - x1) < tol ;
                x0 = x1 ;
                k = k + 1 ;
        endo ;
```

```
            if not converge ;
              errorlog "Warning: INVCDFF has not converged " ;
            endif ;
            ndpclex ;
            retp(x0) ;
endp ;

/* PDFCHI - Chi squared Probability density function with n1 degrees
** of freedom
** Usage: D = pdfchi(x,n)
** Input: X - matrix of chi-squared values (X > 0)
** N - matrix of degrees of freedom (N > 0) (conformable with X)
** Output: D - matrix of density of Chi square(n) at X
*/
proc pdfchi(x,n) ;
            if not(n > 0) ;
                errorlog "ERROR: PDFCHI - N is not positive" ;
                retp(error(99)) ;
            endif ;
            if not(x > 0) ;
                errorlog "ERROR: PDFCHI - X is not positive" ;
                retp(error(99)) ;
            endif ;
            if n < 100 ;
                retp(exp(-n.*ln(2)./2-ln(gamma(n./2))+(n./2-1).*ln(x)-x./2)) ;
            else ;
                retp(exp(-n.*ln(2)./2-lnfact(n./2-1)+(n./2-1).*ln(x)-x./2)) ;
            endif ;
endp ;
```

A.2.5. Vector aleatorio

```
/* para p=5, vector aleatorio*/
/* prueba ALM2,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\vm25.doc reset;
let nsize=10 20 50 100 200 800;
let sig=10.69, 10.16, 9.89, 9.77, 9.53, 9.25;
jj=1;
s1=1;
s2=2;
m=trunc(10*rndus(1,1,s1));
```

```
k=trunc(10*rndus(1,1,s2));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=vector5(m,k,nsize[jj],j+jj);
                a[j,1]=alm25(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm25(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm25;
            p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm25=sumc((meanc(y^4)-eb2)^2)/vb2;
            retp(alm25);
endp;

proc vector5(m,k,n,s); /*m= grados de libertad de la F y T*/
        local rn, x1, y1, f, t,x2,y2,u2,u,z,un,nm,vector;
/* distribución F*/
        rn=rndns(m+k,n,s);
        x1=sumc(rn[1:m, . ]^2)/m;
```

```
        y1=sumc(rn[m+1:m+k, . ]^2)/k;
        f=x1./y1;
/* distribución t-student*/
        u2=rndns(n,m+1,s);
        x2=u2[ . ,1];
        y2=u2[ . ,2:(m+1)];
        t=x2./sqrt(sumc(((y2)^2)')/m);
/* distribución tukey*/
        u=rndus(n,1,s);
        z=(1/0.1975)*((u)^(0.1349)-(1-u)^(0.1349));
/* distribución uniforme*/
        un=rndus(n,1,s);
/* distribución normal*/
        nm=rndns(n,1,s);
        vector = f~t~z~un~nm;
        retp(vector);
endp;
```

A.2.6. Wishart

```
/* para p=5, Wishart*/
/* prueba ALM2,p*/
new ,65520;
output file = c:\bárbara\tesis\resultados\wn25.doc reset;
let nsize=10 20 50 100 200 800;
let sig=10.69, 10.16, 9.89, 9.77, 9.53, 9.25;
jj=1;
s1=1;
vc=eye(5);
nn=trunc(10*rndus(1,1,s1));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndwis (vc, nn, nsize[jj]);
                a[j,1]=alm25(x);
                if a[j,1] >= sig[jj,1];
                  b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
```

```
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm25(x);
        local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm25;
        p=5;
        n=rows(x);
        vb1=(6*n-2)/((n+1)*(n+3));
        vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
        eb2=3*(n-1)/(n+1);
        s=moment(x-meanc(x)',0)/rows(x);
        {va,ve}=eighv(s);
        g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
        y=(x-meanc(x)')*g;
        alm25=sumc((meanc(y^4)-eb2)^2)/vb2;
        retp(alm25);
endp;

proc rndwis (vc, nn, m); /* vc= matriz de var-cov,
                            nn= grados de libertad
                            m= número de simulaciones*/
        local k, c, r, i, t, vcd, ad, a, res, wis, j, mu,q,wis1;
        q=1;
        j= 1;
        r=rows(vc);
        mu = zeros(r,1);
        k=rows(mu);
        c=cols(mu);
        if (( r/=k ) .or (cols (vc) /=k)) .and (r/=1);
          errorlog "rndmn: mu debe ser de dimensión k*1, y vc k*k o un escalar";
            end;
        endif;
        if m<1;
            errorlog "rndmn: el número de simulaciones debe ser >=1 ";
            end;
```

```
        endif;
        wis = zeros (r, r);
        do while j<=nn;
                i=sumc(dotfeq(vc, 0)) .==r;
                if sumc(i) == 0;
                    a=chol(vc) ' ;
                else;
                    t=delif( eye(r), i ) ;
                    vcd=t*vc*t' ;
                    ad=chol(vcd) ;
                    a=t' ad*t ;
                endif;
                res= (mu+a*rndn (k, 1)) ' ;
                wis = wis + (res' * res);
                j = j+1;
        endo;
        do while q <= (m-1);
                wis1 = zeros (r, r);
                i=sumc(dotfeq(vc, 0)) .==r;
                if sumc(i) == 0;
                   a=chol(vc) ' ;
                else;
                   t=delif( eye(r), i ) ;
                   vcd=t*vc*t' ;
                   ad=chol(vcd) ;
                   a=t' ad*t ;
                endif;
                res= (mu+a*rndn (k, 1)) ' ;
                wis1 = wis1 + (res' * res);
                wis=wis | wis1;
                q = q+1;
        endo;
        retp(wis);
endp;
```

A.3. Pruebas ALM

A.3.1. Normal Multivariada

```
/* para p=5, normal multivariada*/
/*prueba ALMp*/
```

```
new ,65520;
output file = c:\bárbara\tesis\resultados\nm5.doc reset;
let nsize=10 20 50 100 200 800;
let sig=19.98, 18.67, 18.06, 17.6, 17.0, 16.21;
jj=1;
mu={1,1,1,1,1};
vc=eye(5);
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=rndmn(mu,vc,nsize[jj]);
                a[j,1]=alm5(x);
               if a[j,1] >= sig[jj,1];
                  b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
              print "PARA N=";;nsize[jj];
              print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm5(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm5;
           p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm5=sumc(meanc(y^3)^2)/vb1+sumc((meanc(y^4)-eb2)^2)/vb2;
            retp(alm5);
endp;
```

```
proc rndmn(mu,vc,n);
          local k,c,r,i,t,vcd,ad,a,res;
          k=rows(mu);
          c=cols(mu);
          r=rows(vc);
          if ((r/=k).or(cols(vc)/=k)).and(r/=1);
             errorlog "rndmn: mu must be kx1, and vc kxk or scalar";
             end;
          endif;
          if n<1;
             errorlog "rndmn: number of simulations must be >=1 ";
             end;
           endif;
          if c/=1 and c/=n;
            errorlog "rndmn: mu must be kxn or kx1";
             end;
          endif;
          if vc==0;
             retp(mu'.*ones(n,1));
          endif;
          i=sumc(dotfeq(vc,0)).==r;
          if sumc(i)==0;
             a=chol(vc)';
          else;
             t=delif(eye(r),i);
             vcd=t*vc*t';
             ad=chol(vcd);
             a=t'ad*t;
          endif;
          res=(mu+a*rndn(k,n))';
          retp(res);
endp;
```

A.3.2. Dirichlet

```
/* para p=5, dirichlet*/
/* prueba ALMp*/
new ,65520;
output file = c:\bárbara\tesis\resultados\dn5.doc reset;
let nsize=10 20 50 100 200 800;
let sig=19.98, 18.67, 18.06, 17.6, 17.0, 16.21;
```

```
jj=1;
s1=1;
aa=rndus(1,1,s1);
nn=5;
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndds(aa,nn,nsize[jj]);
                a[j,1]=alm5(x);
                if a[j,1] >= sig[jj,1];
                  b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm5(x);
          local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm5;
          p=5;
          n=rows(x);
          vb1=(6*n-2)/((n+1)*(n+3));
          vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
          eb2=3*(n-1)/(n+1);
          s=moment(x-meanc(x)',0)/rows(x);
          {va,ve}=eighv(s);
          g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
          y=(x-meanc(x)')*g;
          alm5=sumc(meanc(y^3)^2)/vb1+sumc((meanc(y^4)-eb2)^2)/vb2;
          retp(alm5);
endp;

proc rndds(aa,nn,k); /*aa= parámetro de las gamas
            nn=dimensión del vector
```

```
                k= número de simulaciones*/
                local g,u,m,d,ad,u1,g1,i,dir,sum;
               i=1;
                m=ones(1,nn);
                u=rndu (nn, 1);
                g=-ln(prodc(u'))/aa; /* n gamas con parámetros: 1, aa*/
                u1=rndu (1, 1);
                g1=-ln(prodc(u1'))/aa; /* gama con parámetros: 1, aa*/
                ad= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1, aa*/
                dir=(1/ad)*g;
                dir=(dir)';
                do while i<= (k-1);
                        m=ones(1,nn);
                        u=rndu(nn, 1);
                        g=-ln(prodc(u'))/aa; /* n gamas con parámetros: 1, aa*/
                        u1=rndu(1, 1);
                        g1=-ln(prodc(u1'))/aa; /* gama con parámetros: 1, aa*/
                         sum= (m*g) + g1; /* suma de las n+1 gamas con parámetros: 1,
aa*/
                        d=(1/sum)*g;
                        d=d';
                        dir= dir | d;
                        i=i+1;
                endo;
                retp(dir);
    endp;
```

A.3.3. Mezcla de dos normales multivariadas

```
    /* para p=5, mezcla de dos normales multivariadas*/
    /*prueba ALMp*/
    new ,65520;
    output file = c:\bárbara\tesis\resultados\mz5.doc reset;
    let nsize=10 20 50 100 200 800;
    let sig=19.98, 18.67, 18.06, 17.6, 17.0, 16.21;
    jj=1;
    s1=1;
    s2=2;
    s3=3;
    mu=rndns(5,1,s1);
    vc=eye(5);
```

```
mu1=rndns(5,1,s2);
vc1=eye(5);
mz= rndus(1,1,s3);
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndmmn(mu,vc,mu1,vc1,mz,nsize[jj]);
                a[j,1]=alm5(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
        endo;

/* ALMN , PARA P=5*/
proc alm5(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm5;
           p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm5=sumc(meanc(y^3)^2)/vb1+sumc((meanc(y^4)-eb2)^2)/vb2;
            retp(alm5);
endp;

/*
** random (unstandardized) mezcla de normales multivariadas
** y = rndmmn(mu,vc,mu1,vc1,mz,n);
```

```
** inputs: mu = kx1 media
** vc = kxk matriz de varianzas y covarianzas
** mu1=kx1 media de la segunda normal
** vc1=kxk matriz de varianzas y covarianzas de la segunda normal
** mz= factor de la mezcla
** n = number of simulations
** output: y = nxk matriz de variables aleatorias multivariada dependientes
** de mezcla de normales, cada renglon de y es una simulacion 1xk
*/
proc rndmmn(mu,vc,mu1,vc1,mz,n);
                local k,k1,c,c1,r,r1,i,i1,t,t1,vcd,vcd1,ad,ad1,a,a1,res,res1,mez;
                k=rows(mu);
                k1=rows(mu1);
                c=cols(mu);
                c1=cols(mu1);
                r=rows(vc);
                r1=rows(vc1);
                if c/=c1 and r/=r1 and k/=k1;
                   errorlog "rndmn: mu y vc deben ser de la misma dimensión de mu1 y
vc1";
                   end;
                endif;
                if b<0 or b>1;
                   errorlog "b debe pertenecer al intervalo [0, 1]";
                   end;
                endif;
                if ((r/=k).or(cols(vc)/=k)).and(r/=1);
                   errorlog "rndmn: mu debe ser kx1, y vc kxk o un escalar";
                   end;
                endif;
                if ((r1/=k1).or(cols(vc1)/=k1)).and(r1/=1);
                   errorlog "rndmn: mu1 debe ser kx1, y vc1 kxk o un escalar";
                   end;
                endif;
                if n<1;
                   errorlog "rndmn: el número de simulaciones debe ser >=1 ";
                   end;
                endif;
                if c/=1 and c/=n;
                   errorlog "rndmn: mu debe ser kxn o kx1";
                   end;
```

```
                endif;
                if c1/=1 and c1/=n;
                    errorlog "rndmn: mu1 debe ser kxn o kx1";
                    end;
                endif;
                if vc==0;
                    retp(mu'.*ones(n,1));
                endif;
                if vc1==0;
                    retp(mu1'.*ones(n,1));
                endif;
/* generando la primer normal multivariada*/
                i=sumc(dotfeq(vc,0)).==r;
                if sumc(i)==0;
                    a=chol(vc)';
                else;
                    t=delif(eye(r),i);
                    vcd=t*vc*t';
                    ad=chol(vcd);
                    a=t'ad*t;
                endif;
                res=(mu+a*rndn(k,n))';
/* generando la segunda normal multivariada*/
                i1=sumc(dotfeq(vc1,0)).==r1;
                if sumc(i1)==0;
                    a1=chol(vc1)';
                else;
                    t1=delif(eye(r1),i1);
                    vcd1=t1*vc1*t1';
                    ad1=chol(vcd1);
                    a1=t1'ad1*t1;
                endif;
                res1=(mu1+a1*rndn(k1,n))';
                mez=mz*res+(1-mz)*res1;
                retp(mez);
endp;
```

A.3.4. t-multivariada

```
/* para p=5, t-multivariada*/
/*prueba ALMp*/
```

```
new ,65520;
output file = c:\bárbara\tesis\resultados\tn5.doc reset;
let nsize=10 20 50 100 200 800;
let sig=19.98, 18.67, 18.06, 17.6, 17.0, 16.21;
jj=1;
s1=1;
s2=2;
mu=rndns(5,1,s1);
vc=eye(5);
df=trunc(10*rndus(1,1,s2));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=rndmt(mu,vc,df,nsize[jj]);
                a[j,1]=alm5(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm5(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm5;
           p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
```

```
        alm5=sumc(meanc(y^3)^2)/vb1+sumc((meanc(y^4)-eb2)^2)/vb2;
        retp(alm5);
endp;

/*
** random multivariate t draws
** y = rndmt(mu,vc,df,n);
** inputs: mu = kx1 means
** vc = kxk variance matrix
** df = scalar degrees of freedom
** n = scalar number of simulations
** output: y = nxk matrix of dependent Multivariate T Random Variables
** each row of y is one 1xk simulation
*/
proc rndmt(mu,vc,df,n);
        local k,c,r,i,t,vcd,ad,a,res;
        k=rows(mu);
        c=cols(mu);
        r=rows(vc);
        if ((r/=k).or(cols(vc)/=k)).and(r/=1);
           errorlog "rndmt: mu must be kx1, and vc kxk or scalar";
           stop;
        endif;
        if n<1;
           errorlog "rndmt: number of simulations must be >=1 ";
           stop;
        endif;
        if c/=1 and c/=n;
           errorlog "rndmt: mu must be kxn or kx1";
           end;
        endif;
        if vc==0;
           retp(mu'.*ones(n,1));
        endif;
        i=sumc(dotfeq(vc,0)).==r;
        if sumc(i)==0;
           a=chol(vc)';
        else;
           t=delif(eye(r),i);
           vcd=t*vc*t';
           ad=chol(vcd);
           a=t'ad*t;
```

```
            endif;
            res=(mu+a*(rndn(k,n).*sqrt(df./rndchi(1,n,df))))';
            retp(res);
endp;

/* RNDCHI - Random numbers from a Chi square Distribution
** Usage: x = rndchi(r,c,n)
** Input: R - scalar of row size returned matrix
** C - scalar of Column size returned matrix
** N - matrix of degrees of freedom for Chi square distribution
** (ExE conformable with RxC matrix)
** Output: X - R X C matrix of random numbers ~Chi(N)
*/
proc rndchi(r,c,n) ;
            if not(n > 0) ;
               errorlog "ERROR: RNDCHI - Degrees of Freedom are not positive" ;
               retp(error(99)) ;
            endif ;
            retp(invcfchi(rndu(r,c),n)) ;
endp ;

/* INVCFCHI - Inverse of the Chi squared Cumulative Distribution function
** with n1 degrees of freedom
** Usage: x = invcfchi(p,n)
** Input: P - matrix of percentage points ( 0 < p < 1 )
** N - matrix of degrees of freedom (N > 0) (conformable with X)
** Output: X - matrix of critical values st Pr(x < X) = P and x ~Chi(n)
*/
proc invcfchi(p,n) ;
            local converge,k,tol,t,d,z,x0,x1,f0,df0 ;
            if not(p > 0 and p < 1) ;
               errorlog "ERROR: INVCFCHI - P not in (0,1) " ;
               retp(error(99)) ;
            endif ;
            if not(n > 0) ;
               errorlog "ERROR: INVCFCHI - N is not positive" ;
               retp(error(99)) ;
            endif ;
            tol = 1e-6.*sqrt(n) ;
            clear converge,k ; /*Use Wilson-Hilferty approximation as initial value*/
            t = sqrt(-2*ln(abs((p.>0.5) - p)));
            z = 2.515517 + t.*(0.802853 + t.*0.010328) ;
```

```
                d = 1 + t.*(1.432788 + t.*(0.189269 + t.*0.001308)) ;
                z = t - (z./d) ;
                z = (p.>0.5).*z - (p.<=0.5).*z ;
                d = 2./(9.*n) ;
                x0 = n.*(z.*sqrt(d) + 1 - d)^3 ;
                x0 = x0 + (0.1 - x0).*(x0 .<= 0) ;
                p = 1 - p ;
                do until(converge or k > 50) ;
                        f0 = cdfchic(x0,n) ;
                        df0 = missrv(miss(pdfchi(x0,n),0),tol) ;
                        x1 = (x0 - (p-f0)./df0) ;
                        x1 = x1 + (tol - x1).*(x1.<=0) ;
                        converge = abs(x0 - x1) < tol ;
                        x0 = x1 ;
                        k = k + 1 ;
                endo ;
                if not converge ;
                  errorlog "Warning: INVCDFF has not converged " ;
                endif ;
                ndpclex ;
                retp(x0) ;
endp ;

/* PDFCHI - Chi squared Probability density function with n1 degrees
** of freedom
** Usage: D = pdfchi(x,n)
** Input: X - matrix of chi-squared values (X > 0)
** N - matrix of degrees of freedom (N > 0) (conformable with X)
** Output: D - matrix of density of Chi square(n) at X
*/
proc pdfchi(x,n) ;
                if not(n > 0) ;
                    errorlog "ERROR: PDFCHI - N is not positive" ;
                    retp(error(99)) ;
                endif ;
                if not(x > 0) ;
                    errorlog "ERROR: PDFCHI - X is not positive" ;
                    retp(error(99)) ;
                endif ;
                if n < 100 ;
                    retp(exp(-n.*ln(2)./2-ln(gamma(n./2))+(n./2-1).*ln(x)-x./2)) ;
                else ;
```

```
            retp(exp(-n.*ln(2)./2-lnfact(n./2-1)+(n./2-1).*ln(x)-x./2)) ;
        endif ;
endp ;
```

A.3.5. Vector aleatorio

```
/* para p=5, vector aleatorio*/
/* prueba ALMp*/
new ,65520;
output file = c:\bárbara\tesis\resultados\vm5.doc reset;
let nsize=10 20 50 100 200 800;
let sig=19.98, 18.67, 18.06, 17.60, 17.00, 16.21;
jj=1:
s1=1;
s2=2;
m=trunc(10*rndus(1,1,s1));
k=trunc(10*rndus(1,1,s2));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x=vector5(m,k,nsize[jj],j+jj);
                a[j,1]=alm5(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;
/* ALMN , PARA P=5*/
proc alm5(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm5;
           p=5;
            n=rows(x);
```

```
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm5=sumc(meanc(y^3)^2)/vb1+sumc((meanc(y^4)-eb2)^2)/vb2;
            retp(alm5);
endp;

proc vector5(m,k,n,s); /*m= grados de libertad de la F y T*/
        local rn, x1, y1, f, t,x2,y2,u2,u,z,un,nm,vector;
/* distribución F*/
        rn=rndns(m+k,n,s);
        x1=sumc(rn[1:m, . ]^2)/m;
        y1=sumc(rn[m+1:m+k, . ]^2)/k;
        f=x1./y1;
/* distribución t-student*/
        u2=rndns(n,m+1,s);
        x2=u2[ . ,1];
        y2=u2[ . ,2:(m+1)];
        t=x2./sqrt(sumc(((y2)^2)')/m);
/* distribución tukey*/
        u=rndus(n,1,s);
        z=(1/0.1975)*((u)^(0.1349)-(1-u)^(0.1349));
/* distribución uniforme*/
        un=rndus(n,1,s);
/* distribución normal*/
        nm=rndns(n,1,s);
        vector = f~t~z~un~nm;
        retp(vector);
endp;
```

A.3.6. Wishart

```
/* para p=5, Wishart*/
/* prueba ALMp*/
new ,65520;
output file = c:\bárbara\tesis\resultados\wn5.doc reset;
let nsize=10 20 50 100 200 800;
let sig=19.98, 18.67, 18.06, 17.6, 17.0, 16.21;
```

```
jj=1;
s1=1;
vc=eye(5);
nn=trunc(10*rndus(1,1,s1));
b=zeros(rows(nsize), 1);
do while jj<=rows(nsize);
        a=zeros(10000, 1);
        j=1;
        do while j<=10000;
                x= rndwis (vc, nn, nsize[jj]);
                a[j,1]=alm5(x);
                if a[j,1] >= sig[jj,1];
                   b[jj,1]=b[jj,1]+1;
                endif;
                j=j+1;
        endo;
        b[jj,1]=b[jj,1]/10000;
        output on;
                print "PARA N=";;nsize[jj];
                print "FRACCION =";;b[jj];
        output off;
        jj=jj+1;
endo;

/* ALMN , PARA P=5*/
proc alm5(x);
            local p, vb1, vb2, eb2, g, n, s, y, va, ve, alm5;
           p=5;
            n=rows(x);
            vb1=(6*n-2)/((n+1)*(n+3));
            vb2=(24*n*(n-2)*(n-3))/((n+1)*(n+1)*(n+3)*(n+5));
            eb2=3*(n-1)/(n+1);
            s=moment(x-meanc(x)',0)/rows(x);
            {va,ve}=eighv(s);
            g=ve*diagrv(eye(cols(x)),1/sqrt(va))*ve';
            y=(x-meanc(x)')*g;
            alm5=sumc(meanc(y^3)^2)/vb1+sumc((meanc(y^4)-eb2)^2)/vb2;
            retp(alm5);
endp;

proc rndwis (vc, nn, m); /* vc= matriz de var-cov,
                                nn= grados de libertad
```

```
                    m= número de simulaciones*/
local k, c, r, i, t, vcd, ad, a, res, wis, j, mu,q,wis1;
q=1;
j= 1;
r=rows(vc);
mu = zeros(r,1);
k=rows(mu);
c=cols(mu);
if (( r/=k ) .or (cols (vc) /=k)) .and (r/=1);
  errorlog "rndmn: mu debe ser de dimensión k*1, y vc k*k o un escalar";
    end;
endif;
if m<1;
    errorlog "rndmn: el número de simulaciones debe ser >=1 ";
    end;
endif;
wis = zeros (r, r);
do while j<=nn;
          i=sumc(dotfeq(vc, 0)) .==r;
          if sumc(i) == 0;
              a=chol(vc) ' ;
          else;
              t=delif( eye(r), i ) ;
              vcd=t*vc*t' ;
              ad=chol(vcd) ;
              a=t' ad*t ;
          endif;
          res= (mu+a*rndn (k, 1)) ' ;
          wis = wis + (res' * res);
          j = j+1;
endo;
do while q <= (m-1);
          wis1 = zeros (r, r);
          i=sumc(dotfeq(vc, 0)) .==r;
          if sumc(i) == 0;
              a=chol(vc) ' ;
          else;
              t=delif( eye(r), i ) ;
              vcd=t*vc*t' ;
              ad=chol(vcd) ;
              a=t' ad*t ;
```

```
                endif;
                res= (mu+a*rndn (k, 1)) ' ;
                wis1 = wis1 + (res' * res);
                wis=wis | wis1;
                q = q+1;
        endo;
        retp(wis);
endp;
```

BIBLIOGRAFÍA

[1] BOWMAN, K. O. and SHENTON L. R. (1975), Omnibus Contours for Departures from Normality Based on $\sqrt{b_1}$ y $\sqrt{b_2}$. *Biometrica* 62, 243-250.

[2] CAMPBELL, LO and MACKINLAY (1997), *The econometrics of financial markets*, Princeton University Press

[3] CHENG, R. C. H. (1985), Generation of multivariate normal samples with given sample mean and covariance matrix. *J. Statist. Comput Simulation* 21, 39-50.

[4] CHUNG, J., T. KAILATH and H. LEV-ARI (1987), Fast parallel algorithms for QR and triangular factorization, *SIAM J. Sci. Statist. Comput.* 8, 899-913.

[5] FISHER, R. A. (1930), The Moments of the Distribution for Normal Samples of Measures of Departure from Normality, *Procedings of the Royal Statistical Society*, Series A 130, 16-28

[6] FORSYTHE, G. E. (1942), Von Neumann's comparision method for random sampling from the normal and other distributions. *Math. Comp.* 26, 817-826.

[7] GEORGE, A. and J. W. H. LIU (1981), *Computer Solution of Large Sparce Positive Definitive Systems.* Prentice-Hall, Englewood Cliffs, NJ.

[8] GOLUB, G. H. (1969), Matrix decompositions and statistical calculations, In: R. C. Milton and J. A. Nelder, eds., *Statistical Computation.* Academic Press, New York.

[9] HULL J. C. (1977), Dealing with dependence in risk simulations. *Oper. Res. Quart.* 28, 201-213.

[10] HULL J. C. (2000), *Options, futures & other derivatives*, 4 Ed. Prentice Hall.

[11] JARQUE C. M. and BERA A. K. (1980), Efficient Test for Normality, Homoscedasticity and Serial Independence of Residuals. *Economics Letters* 6, 255-259

[12] JARQUE C. M. and BERA A. K. (1987), A Test for Normality of Oservations and Regression Residuals, *International Statistical Review* 55, 163-172

[13] KINDERMAN, A. J. and J. F. MOHAN (1977), Computer generation of random variables using the ratio of uniform deviates. *ACM Trans. Math. Software* 3, 257-260.

[14] KING G. (1999), *Random multivariate t draws and random multivariate normal*, Harvard University, http://GKing.Harvard.Edu, King@Harvard.Edu

[15] KRONMAL, R. A. and A. V. PETERSON (1981), A variant of the acceptance-rejection method for computer generation of random variables. *J. Amer. Statist. Assoc.* 76, 446-451.

[16] MARSAGLIA, G (1984), The exact-approximation method for generating random variables in a computer. *J. Amer. Statist. Assoc.* 79, 218-221.

[17] MOOD A. D., GRAYBILL F. A., BOES D. C. (1983), *Introduction to the theory of statistics.* McGraw Hill

[18] NIEDERREITER HAROLD (1992), *Random number generation and Quasi-Monte Carlo methods*, Austrian Academy of Sciences. Society for industrial and applied mathematics.

[19] PETERSON, A. V. and R. A. KRONMAL (1982), On mixture methods for the computer generation of random variables. *Amer. Statistical.* 36, 184-191.

[20] RAO C. R. (1993), *Handbook of Statistics*, Vol. 9, Computational Statistics, Elsevier Science Publishers B.V.

[21] REUNEV and RUBISTEIN (1981), *Simulation and the Monte Carlo Method*, Wiley series in probability and mathematical statistics, John Wiley & Sons. Inc.

[22] SOBOL. I. M. (1983), *Método de Montecarlo*, Ed. MIR.

[23] URZÚA C. M. (1996), On the correct use of Omnibus test for normality, *Economics letters* 53, 247-251

[24] URZÚA C. M. (1997), Omnibus test for multivariate normality based on a class of maximum entropy distributions, *Advances in Econometrics* 12, 341-358