Simulación con Estimadores en R

19 de fevereiro de 2014

I Congreso Internacional de Estadística. Trujillo - Perú. 2013

Resumo

R es un ambiente rico para el cálculo estadístico y tiene muchas posibilidades para explorar datos en su biblioteca básica. Además, R contiene una colección de funciones para la simulación y el resumen de las distribuciones de probabilidad de uno o más parámetros. Uno de los objetivos es dar una breve introducción a los comandos básicos para resumir datos y gráficos. Ilustraremos estos comandos en un conjunto de datos. Un segundo objetivo é usar ambientes de programación, creando nuestras propias funciones vía algoritmos, como por ejemplo visualizar si las propriedades de los estimadores como, viés, cuadrado medio del error, consistencia y eficiencia en el límite (cuando el tamaño de la muestra crece) pueden ser superados usando simulaciones de MonteCarlo, o si el verdadero nivel de significación puede ser confirmado en repetidas simulaciones. Un tercero objetivo es verificar si las funciones dentro de bibliotecas implementadas en R, pueden ser comparadas usando simulaciones y retornar resultados satisfactorios (verificando su desempeño), creando categorías para estos ambientes de comparación.

1 Definición de Estimadores

Definiremos algunas propriedades de los estimadores.

1. Parámetro. Verdadero valor de una característica de interes, denominado por θ, que raramente es conocido.

2. Estimativa. Valor numérico obtenido por el estimador, denominado de $\hat{\theta}$ en una muestra.

3. Viés y no viés. Un estimador es no in-sesgado si: $E(\hat{\theta}) = \theta$, onde el viés es dado por: $vies(\hat{\theta}) = E(\theta \hat{-} \theta) = E(\hat{\theta}) - \theta$

4. Cuadrado médio del error (ECM). Es dado por:
   $ECM(\hat{\theta}) = E(\hat{\theta} - \theta)^2 = V(\hat{\theta}) + (vies[\hat{\theta}])^2$

5. Eficiencia de un Estimador: $\hat{\theta}$ es un estimador eficiente de θ si son satisfechas las siguientes condiciones:

   - $E(\hat{\theta}) = \theta$; y
   - $Var(\hat{\theta}) \leq Var(\tilde{\theta})$, donde $\tilde{\theta}$ es cualquier otro estimador de θ.

6. Un estimador es consistente si: $plim(\hat{\theta}) = \theta$; y
   $\lim \longrightarrow \infty ECM(\hat{\theta}) = 0$

7. Las leyes de los grandes números explican por qué el promedio o media de una muestra al azar de una población de gran tamaño tenderá a estar cerca de la media de la población completa.

Ilustrando Propriedades de estimadores usando simulaciones de Monte Carlo como un subproducto de la ley de los grandes números

1.1 Algoritmo del viés

Para ilustrar el viés basta obtener una muestra aleatoria de una determinada distribución, Ejemplo considere 10 muestras de una N(10,4), determine o viés no R.

```
set.seed(20)
vies=mean(rnorm(10,10,4))-10
vies
[1] -1.449323

#Algoritmo con simulación de Monte Carlo:
set.seed(20)
ns= seq(2, 1000,by=2)#inicia em 2 termina  em 1000 a cada 2
vies= numeric(length(ns))  # prepara 500 elementos (vector de ceros)
for (i in 1:length(ns)){   # cadena o lazo
muestra= rnorm(ns[i],10,4) # números aleatorios N(10,4)
vies[i]= mean(muestra)-10  # diferencia de medias (muestra y parámetro)
}
# gráfico del tamaño y la media de la muestra
plot(ns, vies, main="Desempeño del viés\n para varios tamaños de n",
type="s",ylab="viés")
abline(h=0,col=2)
```

Para visualizar desempeño del viés podemos observar o gráfico 1.

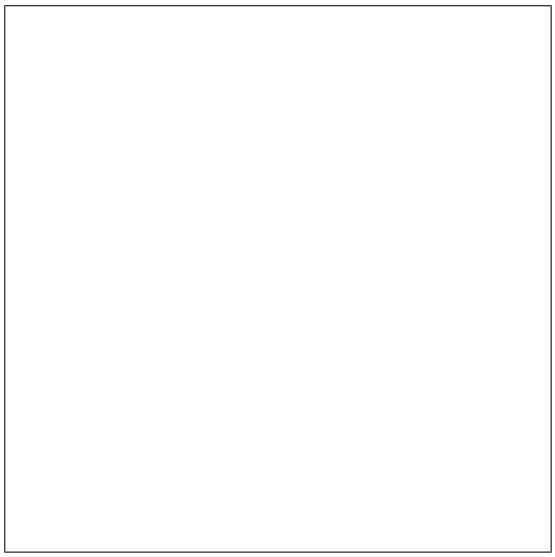


Gráfico 1. Desempeño del viés

Podemos realizar un teste de comparación de medias para verificar que mismo que a lo largo de la cadena las medias son las mismas, sin embargo un teste de comparación de varianzas acusa variabilidad diferente a lo largo de la cadena. Para tanto usamos dos muestras, la primera del elemento 100 hasta 300, y la segunda de 301 hasta 500 del vies.

1.2 Testes de comparación

Usamos el teste t student, para comparar as medias:

```
> t.test(vies[100:300],vies[301:500]) #Teste t

        Welch Two Sample t-test

data:  vies[100:300] and vies[301:500]
t = 1.342, df = 352.752, p-value = 0.1805
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.01109556  0.05876153
sample estimates:
   mean of x    mean of y
 0.019856577 -0.003976409
```

Un algoritmo para la salida gráfica gráfica puede ser construida:

```
x<-seq(-3,3,0.005)
y<-dt(x,352.752)
plot(x,y,type="l", main="Comparando las medias\n vies[100:300]
y vies[301:500]")
polygon(c(x[x>=1.96],1.96),c(y[x>=1.96],y[x==3]),col=2)
polygon(c(x[x<=-1.96],-1.96),c(y[x<=-1.96],y[x==-3]),col=2)
legend("center","95%")
abline(v=1.342)
```

con el siguiente gráfico:

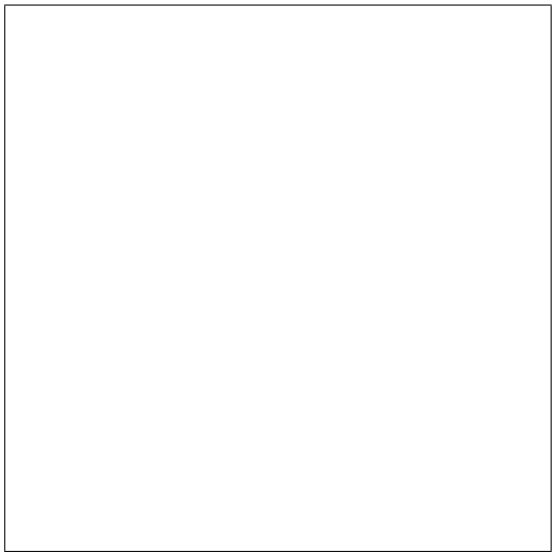


Gráfico 2. Salida gráfica de la comparación de medias.

Usamos el teste F, para comparar varianzas:

```
> var.test(vies[100:300],vies[301:500]) #Teste F

        F test to compare two variances

data:  vies[100:300] and vies[301:500]
F = 2.1583, num df = 200, denom df = 199, p-value = 8.327e-08
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 1.633884 2.850821
sample estimates:
ratio of variances
          2.158325
```

Una salida gráfica con su respectivo algoritmo puede ser construida:

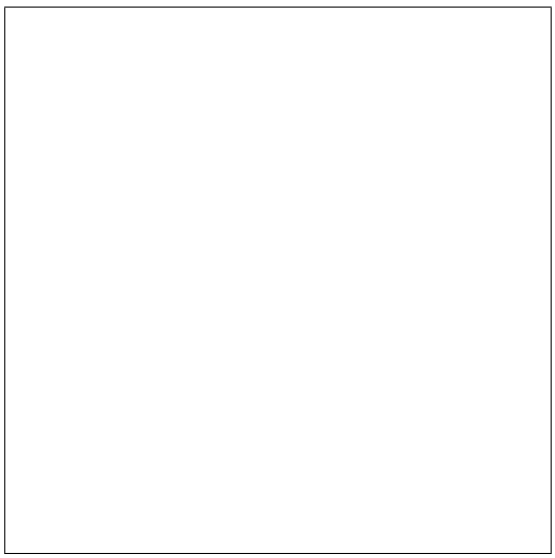


Gráfico 3. Salida gráfica para la comparación de varianzas.

```
x<-seq(0.5,2.5,0.005)
y<-df(x,200,199)
plot(x,y,type="l", main="F(200,199)")
polygon(c(x[x>=qf(0.95,200,199)],qf(0.95,200,199)),
c(y[x>=qf(0.95,200,199)],y[x==2]),col=2)
legend(0.8,0.5,"95%")
abline(v=2.1583)
```

Boxplot para visualizar a comparación del viés.

```
boxplot(vies[100:300],vies[301:500],col="hotpink",
main=" Comparación entre \n vies[100:300] y vies[301:500]")
```

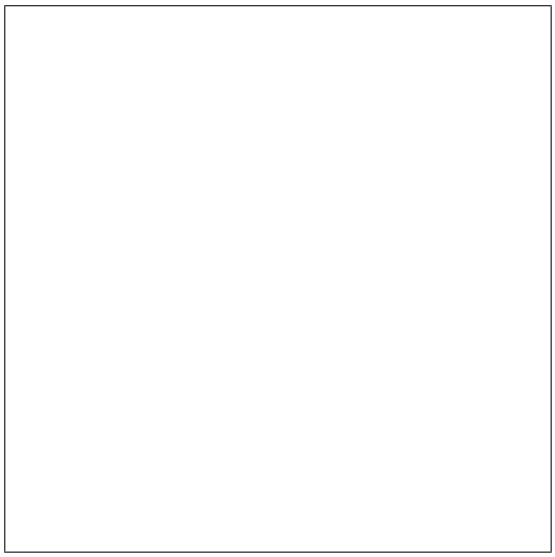


Gráfico 4. medias iguales y Varianzas diferentes.

Tarea: Primero aumente el tamaño de la muestra y verifique el desempeño del viés. Segundo use otra distribución de probabilidad.

1.3 Algoritmo para la Consistencia

Un estimador es consistente cuando su valor se aproxima del verdadero valor del parámetro a medida que aumenta el tamaño de la muestra. Veamos como podemos ilustrar este resultado usando simulación. La idea básica es la siguiente:

1. Escoger una distribución y sus parámetros,
2. Definir el estimador,
3. Definir una secuencia creciente de valores para los tamaños de muestra,
4. Obtener una muestra de cada tamaño,
5. Calcular la estadística para cada muestra,

6. Realizar un gráfico de los valores de las estimativas contra el tamaño de la muestra, indicando en este gráfico el valor verdadero del parámetro.

```
####  Algoritmo #####

#media de la distribución Normal con media 10 y varianza 16
#el estimador es la media de la muestra en diferentes tamaños normales.
ns= c(2, seq(5, 1000, by=5), seq(1010, 5000, by=10))
estim= numeric(length(ns))   #prepara 601 elementos (vector de ceros)
for (i in 1:length(ns)){     #lazo (de 1 a 601 realizar)
muestra= rnorm(ns[i], 10, 4) #números aleatórios N(10,16)
estim[i]=mean(muestra)       #aloca la media de la muestra en estim
}
plot(ns, estim,type="l", main="Consistencia de un estimador
\n N(10,16)") #gráfico del tamaño y la media de la muestra
abline(h=10, col=2) # línea de la media =10
```

Salida gráfica de la consistencia de un estimador

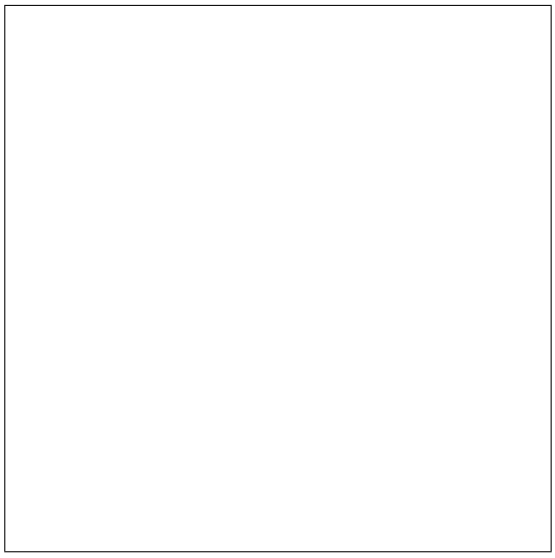


Gráfico 5. Varios tamaños de muestras normales.

Tarea:

1. realizar el algoritmo para muestras mayores con normales, N(20,5)
2. realizar el algoritmo para muestras mayores con exponenciales, exp(.2)

1.4 algoritmo de la media muestral

En la inferencia estadística es necesario conocer la distribución de la muestra de los estimadores. Para clarificar este fato podemos ilustrarlo usando algoritmos alocando matrices, considere como caso particular del la distribución muestral de estimadores a media muestral).

Ejemplo: considere Y una V.A. con distribución normal N(μ, σ^2) y sea um parâmetro de interes T, onde o estimador $\hat{T} = \hat{\mu}/\hat{\sigma}$. Para obtener por simulacíon la esperanza y varianza del estimador seguimos os passos:

1. Escojer una distribución y sus parámetros, caso particular escojemos una N(180,64)
2. Definir un tamaño de muestra, asi escojemos n=20,
3. Obtener por simulación un número N de repeticiones, vamos usar N=1000,
4. Calcular la estadística de interes para cada repetición
5. Obtener las estimativas para E[T] y V[T] dadas por $E[\hat{T}]$ y $V[\hat{T}]$.

O algoritmo no R:

```
#####  1000 simulaciones de tamaño 100 ######
set.seed(20)
muestras<- matrix(rnorm(100*1000, mean=180, sd=10), nc=1000)
Tmuestra<- apply(muestras, 2, function(x) {mean(x)/sd(x)})
estET <- mean(Tmuestra)
estVarT<- var(Tmuestra)
cbind(estET,estVarT)
        estET  estVarT
[1,] 18.10165 1.596866

#####  1000 simulaciones de tamaño 10000 ######
set.seed(20)
muestras2<- matrix(rnorm(10000*1000, mean=180, sd=10), nc=1000)
Tmuestra2<- apply(muestras2, 2, function(x) {mean(x)/sd(x)})
estET2 <- mean(Tmuestra2)
estVarT2<- var(Tmuestra2)
cbind(estET2,estVarT2)
     estET2   estVarT2
[1,] 18.00026 0.01549134
```

En el algoritmo obtenemos 1000 repeticiones de tamaño 100 e 10000 (tamaño de cada muestra) que almacenamos en una matriz com 1000 colunas, donde cada columna es una muestra por simulación. Seguidamente usamos la función **apply** para calcular las cantidades deseadas que definimos con la función: function(x) mean(x)/sd(x). En el ejemplo fue obtenido o valor esperado y la varianza, presentando medias próximas y variabilidad proporcional al tamaño de la muestra, una presentación visual es dado por la función en el R y el gráfico de Box plot:

```
boxplot(Tmuestra,Tmuestra2,main ="Box Plot de 1000 Simulaciones
de la Media muestral \n para n=100 y n=1000")
```

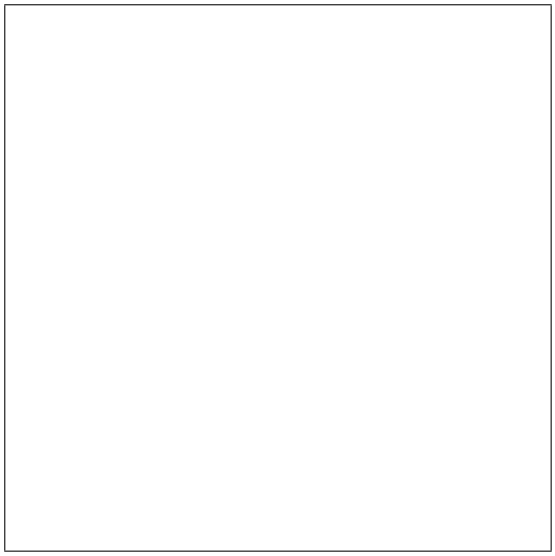


Gráfico 6. Desempeño de la media muestral.

Para comparar medias e varianzas usando os testes t e F, respectivamente temos:

```
> t.test(Tmuestra,Tmuestra2)

        Welch Two Sample t-test

data:  Tmuestra and Tmuestra2
t = 2.5251, df = 1018.381, p-value = 0.01172
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.02259856 0.18018730
sample estimates:
mean of x mean of y
 18.10165  18.00026

> var.test(Tmuestra,Tmuestra2)

        F test to compare two variances

data:  Tmuestra and Tmuestra2
F = 103.0812, num df = 999, denom df = 999, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
  91.05149 116.70027
sample estimates:
ratio of variances
          103.0812
```

Una alternativa visual del desempeño de la media muestral es dada por el gráfico de plot:

```
par(mfrow=c(2,1))
plot(Tmuestra,type="l", main="1000 Simulaciones de la Media muestral
\n para n=100 y n=1000")
plot(Tmuestra2, type="l", ylim=c(16,22))
```

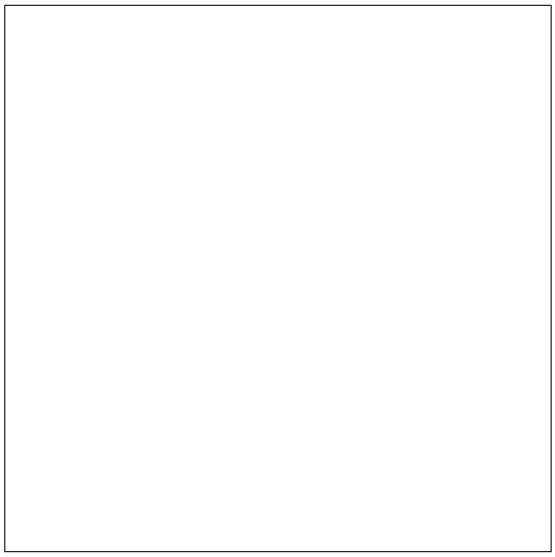


Gráfico 7. Desempeño de la media muestral.

Si por acaso no usar la función **set.seed()**, obtendrá resultados un poco diferentes (mas no mucho!) pues nuestras muestras da distribución escogerán una semilla de generación diferente.

1.5 Algoritmo del CME

El cuadrado medio del error definida por: $CME = var(\hat{\theta}) + vies(\hat{\theta})^2$, tiende a cero cuando n es grande.

1.5.1 En poblaciones Normales

Algoritmo con simulación de Monte Carlo del Cuadrado medio del error para varios tamaños de n (grande):

```
ns= seq(6, 1000,by=2) #inicia en 2 termina  en 1000 a cada 2
eqm= numeric(length(ns))
var= numeric(length(ns))
vies= numeric(length(ns))  # prepara 498 elementos (vector de ceros)
for (i in 1:length(ns)){   # lazo
muestra= rnorm(ns[i],10,4) # números aleatórios N(10,4)
vies[i]= mean(muestra)-10  # diferencia: media de la muestra con parámetro
var[i]= var(muestra)/ns[i] # Varianza de la media muestral
```

```
eqm[i]=var[i]+(vies[i])^2  # ECM (erro Cuadrático médio)
}
plot(ns, eqm,type="l", main="Convergencia do CME cuando n crece")
abline(h=0,col=2)
```

La salida gráfice es:

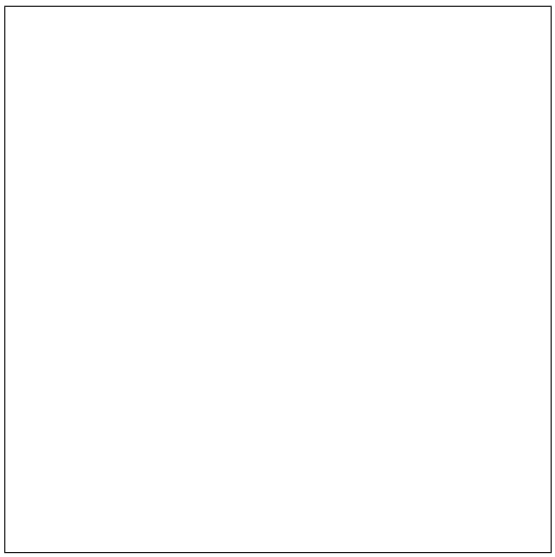


Gráfico 8.1. Desempeño del CME en poblaciones normales.

1.5.2 En poblaciones Exponenciales

```
ns= seq(6, 500,by=2)  #inicia em 2 termina  em 1000 a cada 2
eqme= numeric(length(ns))
vare= numeric(length(ns))
viese= numeric(length(ns))  #prepara 498 elementos (vetor de zeros)
for (i in 1:length(ns)){      #laço
amostrae= rexp(ns[i],.10) #números aleatórios N(10,4)
viese[i]= mean(amostrae)-10  #diferença: média da amostra com parâmetro
vare[i]= var(amostrae)/ns[i]       #Variancia da média da amostra
eqme[i]=vare[i]+(viese[i])^2     #EQM (erro quadratico médio)
 }

plot(ns, eqme,type="l", main="Convergencia do CME cuando n crece")
abline(h=0,col=2)
```

1.5.3 Comparando o CME

Usaremos a construcción de los dos algoritmos para comparar el desempeño de CME.

```
#Comparando normal e Exponencial graficamente
par(mfrow=c(2,2))
plot(ns, eqme, type="l",ylim=c(-0.1,10),main="Erro Quadrático Médio")
#gráfico do tamanho e EQM
lines(ns,eqm,col=2)
abline(h=0,col="lightblue")
legend("topright",legend=c("Normal", "exponencial"),col=c(2,1),lty=1:1)
plot(ns,viese,type="l", main="viés")
lines(ns,vies,col=2)
legend("topright",legend=c("Normal", "exponencial"),col=c(2,1),lty=1:1)
plot(ns,vare,type="l", main="variância do estimador")
lines(ns,var,col=2)
legend("topright",legend=c("Normal", "exponencial"),col=c(2,1),lty=1:1)
plot(density(vies),xlim=c(-2,10),main="Densidade do viés \n para normal e exponenc
lines(density(viese))
legend("topright",legend=c("Normal", "exponencial"),col=c(2,1),lty=1:1)
```

Con la siguiente salida gráfica:

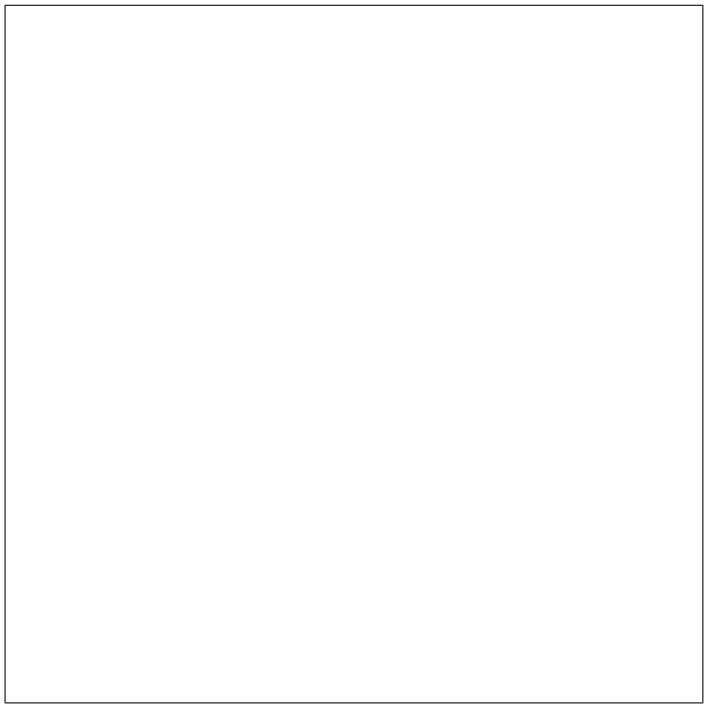


Gráfico 8.2. Compara CME en poblaciones normales y exponenciales.

2 Criando estadisticas

Considere un teste t para dos muestras independientes con varianzas iguales (homogéneas) mas desconocidas

```
testet=function(x,y)
{
options(digits=3)
n1=length(x)
n2=length(y)
dm=sqrt(((n1-1)*sd(x)^2+(n2-1)*sd(y)^2)/(n1+n2-2))
est2=(mean(x)-mean(y))/(dm*sqrt(1/n1+1/n2))
return(est2)
}
```

Un otro teste para comparación de médias es para dos muestras independientes con varianzas iguales y conocidas.

```
testet1=function(x,y)
{
options(digits=3)
n1=length(x)
n2=length(y)
dm=sqrt((sd(x)^2/n1+sd(y)^2/n2))
est1=(mean(x)-mean(y))/(dm)
return(matrix(c("valor observado","valor tabular",est1,qt(0.975,n1+n2)),2,2))

}
```

O R tiene como función `t.test()` de la biblioteca `stats` para comparación de medias, definida para dos muestras independientes con varianzas iguales (homogéneas) mas desconocidas como estándar. Para verificar el calculo da estadística en 100 muestras Normales, `N(0,1)` y comparadas con 100 exponenciales con parámetro 1, `exp(1)`, con semilla de generación 100, realizamos:

```
set.seed(100)
testet(rnorm(100),rexp(100,1))
[1] -6.25
set.seed(100)
t.test(rnorm(100),rexp(100,1))$statistic
    t
-6.25
```

Si comparada con la distribución t con $n1 + n2$ grados de libertad, podemos obtener el valor crítico con $\alpha = 0.05$ en un teste bicaudal.

```
qt(0.025,198)
```

```
[1] -1.97
```

Como $-6.25 < -1.97$, rechazamos la hipótesis que nula de igualdad de medias.

Para testar la hipótesis nula de igualdad de Varianzas, criamos el siguiente algoritmo:

```
testevarianza=function(x,y)
+ {
+ options(digits=3)
+ n1=length(x)
+ n2=length(y)
+ vm=(var(x)/n1)/(var(y)/n2)
+ return(vm)
+
+ }
```

Para Comparar las varianzas del ejemplo de Normales y Exponenciales, usamos la funcion `var.test()` de la biblioteca `stats`:

```
set.seed(100)
testevarianza(rnorm(100),rexp(100,1))
[1] 1.67
set.seed(100)
var.test(rnorm(100),rexp(100,1))$statistic
   F
1.67
```

Si comparada con la distribución F con $n1$ y $n2$ grados de libertad, podemos obtener el valor crítico con $\alpha = 0.05$ en un teste unicaudal.

```
 qf(0.95,99,99)

[1] 1.39
```

Conclusión: comparando 100 muestras `N(0,1)`, y 100 muestras `Exp(1)`, podemos afirmar que no existen evidencias para aceitar que medias y varianzas son iguales con niveles significantes de 5% y con semilla de generación igual a 100. Para visualizar esta comparación, presentamos la función y la salida gráfica:

```
boxplot(rnorm(100),rexp(100,1),col=8,
main="Comparando Normales y Exponenciales")
```

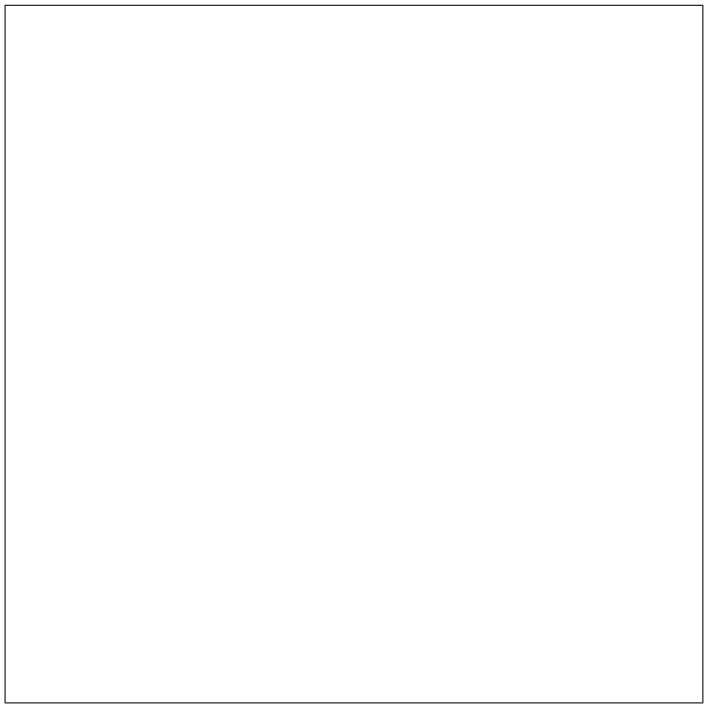


Gráfico 8.3. Boxplot de N(0,1) y Exp(1) respectivamente.

3 Teorema Central del Limite

Asumir que los datos tienen una distribución normal es altamente conveniente tanto del punto de vista teórico como del ponto de vista computacional. Gauss trabajo con este problema mucho tiempo, mas fue Laplace que difundio. Laplace llamo este resultado de Teorema Central del Limite, que dice que sobre la hipótesis de muestreo aleatorio, cuando el tamaño de la muestra aumenta, la distribución muestral de la media muestral se aproxima de una distribución normal, independiente de que distribución proviene la muestra. Asi, si la muestra es uniforme, binomial, poison, etc, se el tamaño muestral es suficientemente grande, podemos asumir que la media muestral tiene una distribución normal.

3.1 Caso da Uniforme

Algoritmo que selecciona muestras uniformes de tamaños 2,10,25, y, 100, verificando la distribución posterior en repetidas simulaciones (500).

```
par(mfrow=c(1,1))
plot(0,0,type="n", xlim=c(0,1), ylim=c(0,13.5),
xlab="Densidad Estimada", ylab="f(x)")
m=500;a=0;b=1
t=c(2,10,25,100);abn=c()
for(j in 1:length(t)){
for(i in 1:m) abn[i]=mean(runif(t[j],a,b))
lines(density(abn),lwd=2, col=j)}
legend("topleft",legend=c("n=2","n=10","n=25","n=100"),lty=1:1,col=1:4)
```

La salida gráfica es:

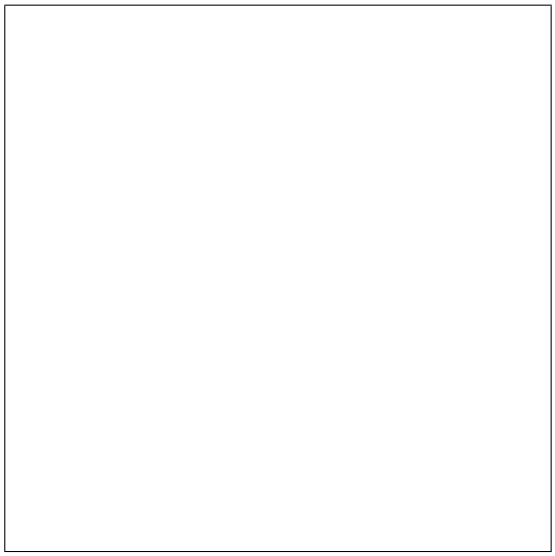


Gráfico 9. Muestras Uniformes simuladas con tamaño 500.

3.2 Caso da Binomial

Una otra distribución bastante usada para aproximar a la normal cuando el tamaño crece es la binomial.

```
par(mfrow=c(2,2))
m=200;p=1/2;
n=c(5,10,25,100)
for(i in 1:length(n))
{
```

```
res=rbinom(m,n[i],p)
hist(res, prob=TRUE,col=gray(0.7),main="aprox. normal pela binomial",xlab=n[i])
curve(dnorm(x,n[i]*p,sqrt(n[i]*p*(1-p))),add=TRUE)
}
```

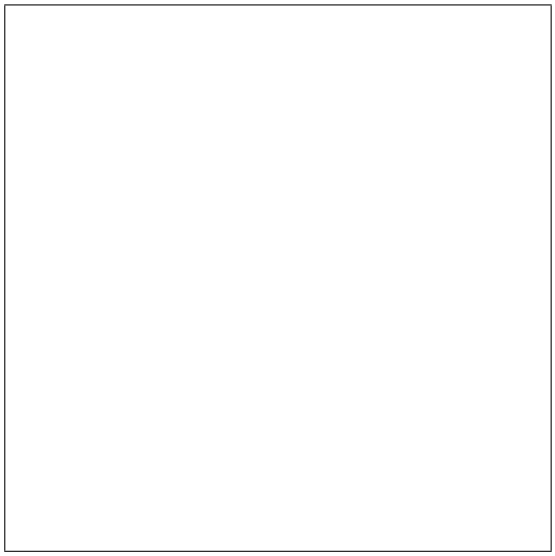


Gráfico 10. Varios tamaños de muestra.

Un algoritmo mas elaborado que muestra paso a paso la transformación de la Binomial a la Normal estándar es:

```
#Algoritmo gráfico para aproximar a distribuição binomial á normal

BinomialAproxNormal<-function(n,p){
cociente=n/trunc(n)
if (n<2 | n>200 | cociente>1)
stop("n debe ser un número natural entre 2 y 200" )
if (p <0.01| p>0.99) stop("p debe variar entre 0.01 y 0.99" )
binom1=dbinom(0:n,n,p)
binom1Red3=round(binom1,3)
matrizBin<-matrix(0:n,ncol= (n+1))
matrizFrec=matrix(binom1Red3,ncol= (n+1))
matrizTabla=rbind(matrizBin, matrizFrec)
TablaTrunc=matrizTabla[,matrizTabla[2,]>=0.001]
fila1=TablaTrunc[1,]
fila2=TablaTrunc[2,]*1000
Xn=rep(fila1,fila2)
hist(Xn,breaks=seq(min(fila1)-0.5, max(fila1)+0.5,by=1),
freq=F,ylab=c("Probabilidad"),main=paste(" "))
#histograma variable X
#Transformando la variable
Yn=Xn-n*p

hist(Yn,breaks=seq(min(fila1-n*p)-0.5,max(fila1-n*p)+0.5,by=1),
freq=F, ylab=c("Probabilidad"),main=paste(" "))
#histograma variable Y=X-n*p
Zn=Yn/sqrt(n*p*(1-p))

inversadesvio=1/(sqrt(n*p*(1-p)))
hist(Zn,freq=F,breaks=seq(min(Zn)-1/2* inversadesvio,
max(Zn)+ 1/2*inversadesvio, by=1/(sqrt(n*p*(1-
p)))),xlim=c(-4,4), ylim=c(0,0.5), ylab=c("Probabilidad"),
main=paste(" ")) #histograma variable Z=X-n*p/
sqrt(n*p*(1-p))
xaleatorios=seq(-3.5,3.5,length=100000)
ynormales=dnorm(xaleatorios, 0,1)
lines(xaleatorios, ynormales,col="black",lwd=3)
}
```

Para ejemplificar considere, n=10, y p=0.4, usamos la función construida `BinomialAproxNormal()`, de la siguiente forma:

```
BinomialAproxNormal(10,0.4)
```

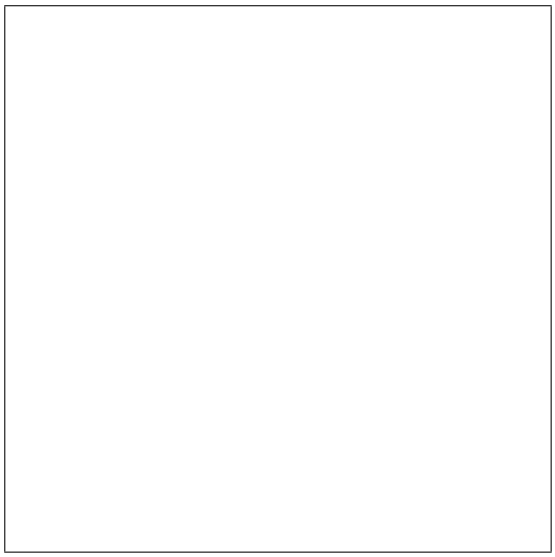


Gráfico 11. Transformación de la Binomial(10,0.4) a la N(0,1).

4 Desempeño de modelo AR e MA

Uno de los objetivos de las series a lo largo del tiempo es identificar el proceso generador de la misma. En situaciones prácticas el proceso generador de una serie puede ser identificado por un caso particular de modelos, con la modela-je ARMA (autorregresivos y medias móviles), siguiendo la metodología Box e Jenkins, entretanto desconocemos el desempeño de estos modelos usando funciones alojadas en la plataforma R. Para identificar este desempeño (el cual fué clasificado) fueron generadas simulaciones para varios parámetros de modelos AR y MA, y comparadas con sus respectivas series teóricas. Este trabajo intenta esclarecer algunas lagunas en cuanto al uso de las funciones: **arima.sim()** y **auto.arima()**, del as bibliotecas **stats** y **forecast** del R.

El interés es determinar el porcentaje de aciertos de las series simuladas con sus correspondientes serie teóricas.

4.1 Clasificación

La clasificación del desempeño de las funciones (en porcentaje), es dado de la siguiente manera:

1. De 0 a 50 del porcentaje de aciertos, desenpeño pobre (F)
2. Mayor de 50 a 70 del porcentaje de aciertos, desenpeño médio (M),
3. Acima de 70 del porcentaje el desenpeño es ideal o bueno (B).

4.2 Las simulaciones

Los ambientes de comparación son varios tamaños de simulación, varios tamaños de la muestra da serie generada, y varios tamaños de parámetros estimados.

Para determinar el desempeño de las series generadas por simulaciones y comparadas con las series teóricas, fué implementado algoritmos que generan series por simulación y retornan la orden de una serie teórica. La primera serie simulada es un $ARIMA(0,0,3)$, o en termos simples un MA(3), con parámetros: $\theta_1 = 0, \theta_2 = 0, \theta_3 = 0.12, 0.24, 0.36, 0.48, 0.6$, lo que representa una serie de medias móviles de orden 3, con 5 parámetros alternativos de simulación para esta orden. La serie teórica es construida a partir de la serie simulada, en termos simples usamos la función del mejor ARIMA disponible en R.

Para 10 y 100 simulaciones y con tamaños de la serie de 50 y 100 observaciones generadas, las frecuencias no revelan la verdadera orden teórica de un MA(3), y si un MA(0), para todos los parámetros de la serie simulada excepto cuando hay 100 simulaciones y tamaño de la serie de 100, considerando los parámetros, $\theta_3 = 0.36, 0.48, 0.6$. Para 1000 simulaciones y con tamaños de la serie de 1000, las observaciones generadas por simulación de un MA(3), tiene como retorno teórico un MA(3) para todos los parámetros simulados. La tabla a seguir identifica el desempeño de la función auto.arima().

4.3 El algoritmo

Algoritmo para un tamaño 1000 de simulación de un MA(3), con tamaños de la serie (muestra) de 1000 es presentado a seguir.

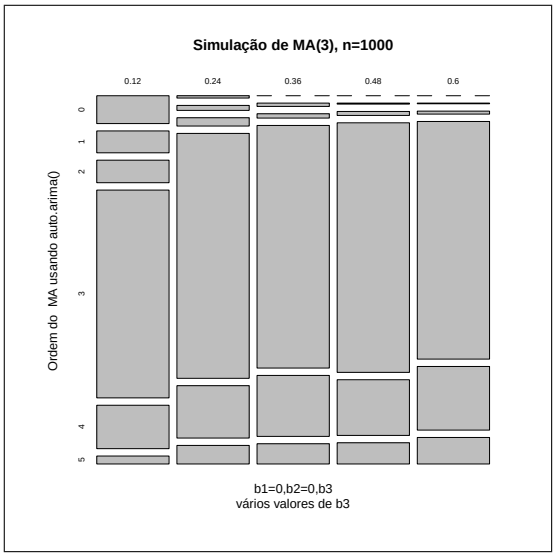
```
set.seed(20)
estMA=matrix(0,nrow=5, ncol=1000)
```

```
for (i in 1:5){ ma3=0.12*i
for (j in 1:1000) {
simts=arima.sim(n=100, model=list(ma=c(0,0,ma3)))
estMA[i,j]=auto.arima(simts,start.q=3)$arma[2]
colnames(estMA)=c(1:1000)
}

}
detectar=table(row(estMA), estMA)
dimnames(detectar)=list(ma3=paste(0.12*(1:5)),
Order=paste(0:(dim(detectar)[2]-1)))
detectar
plot(detectar,main="simulación de MA(3), n=100",
sub="varios valores de b3",
ylab="orden do  MA usando auto.arima()", xlab="b1=0,b2=0,b3")

Order
ma3   0    1  2  3   4    5

0.12 84   66 68 627  131 24
0.24  7   15 25 739  158 56
0.36  0   10 13 732  184 61
0.48  0    3 12 753  168 64
0.60  0    2 09 717  192 80
```

4.4 Os resultados

MA(3) con parámetros 0.12,0.24,0.36,0.48,0.6									
	10 simulaciones			100 simulaciones			1000 simulaciones		
θ_3	50	100	1000	50	100	1000	50	100	1000
0.12	F	F	F	F	F	M	F	F	M
0.24	F	F	F	F	F	B	F	F	B
0.36	F	M	F	F	M	B	F	M	B
0.48	F	M	F	F	M	B	F	M	B
0.60	F	M	F	F	M	B	F	M	B

Figura 41. desempeño de 1000 simulaciones, tamaño 1000 de um MA(3).



La segunda serie simulada es un ARIMA(1,0,0), que en termos simples es un AR(1), con parámetros: $\phi_1 = 0.12, 0.24, 0.36, 0.48, 0.6$, lo que representa una serie autorregressiva de orden 1, con 5 parámetros alternativos de simulación para esta orden.
Los resultados para 10 e 100 simulaciones y con tamaños de la serie de 50 y 100 observaciones generadas, las frecuencias no revelan la verdadera orden teórica de un AR(1), e si un AR(0), para los parámetros $\phi_1 = 0.12, 0.24, 0.36$ para los otros parámetros de $\phi_1 = 0.48, 0.6$ las simulaciones revelaran las mayores frecuencias como la verdadera orden teórica. Para los tamanhos de la serie de 1000 observaciones generadas por simulación de un AR(1) apenas, $\phi_1 = 0.12$ no tiene las mayores frecuencias como su retorno teórico de un AR(1). Ver tabla a seguir:

AR(1) con parámetros 0.12,0.24,0.36,0.48,0.6									
	10 simulaciones			100 simulaciones			1000 simulaciones		
ϕ_1	50	100	1000	50	100	1000	50	100	1000
0.12	F	F	F	F	F	F	F	F	F
0.24	F	F	F	F	F	M	F	F	M
0.36	F	M	M	M	M	B	F	M	M
0.48	M	F	B	M	B	B	M	B	B
0.60	M	B	B	M	B	M	M	B	B

El algoritmo para 1000 simulaciones con tamaños de la serie de 1000 es presentado a seguir.

```
set.seed(20)
estAR=matrix(0,nrow=5, ncol=1000)
```

```
for (i in 1:5){ ar1=0.12*i
for (j in 1:1000) {
simts=arima.sim(n=1000, model=list(ar=c(ar1,0,0)))
estAR[i,j]=auto.arima(simts,start.p=1)$arma[1]
colnames(estAR)=c(1:1000)
}
}

detectar=table(row(estAR), estAR)
dimnames(detectar)=list(ar1=paste(0.12*(1:5)),
Order=paste(0:(dim(detectar)[2]-1)))
detectar
plot(detectar,main="simulación de AR(1), n=1000",xlab=expression(phi[1]),
ylab="orden do  AR usando auto.arima()", sub="varios valores de um AR(1)")
 detectar
    Order
ar1      0   1   2   3   4   5
  0.12 489 379 109  17   6   0
  0.24 233 612 116  28   8   3
  0.36  69 696 180  47   7   1
  0.48  39 714 198  38   9   2
  0.6    9 742 201  36   8   4
```

El gráfico correspondiente es presentado a seguir:

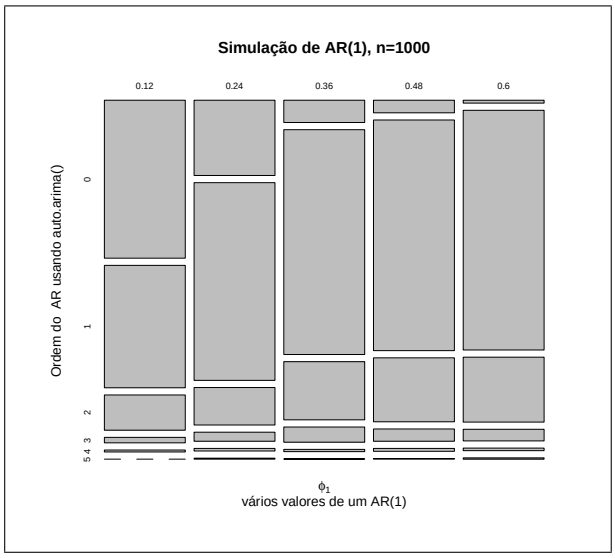


Figura 42. desempeño de 1000 simulaciones de tamaño 1000 de un AR(1).

Referências

[1] Albert, Jim (2009). *Bayesian computation with R.* Editora Springer. Second Edition. New York, USA.

[2] Borde, Arvind (1992). TEX *by example.* A beginner´s guide. Ed. Academic Press Professional. United States of America.

[3] Carneiro, Orlando (1997). *Econometria Básica. Teoria e Aplicações.* Editora Atlas. Segunda Edição. son Paulo.

[4] Conover, W. J. (1999) *Practical nonparametric statistics.* 3.ed. New York.

[5] Cribari Neto, F.; Cabral de Araújo Gois Matheus (2002). *Uma Análise de Monte Carlo do desenpeño de Estimadores de Matrizes de Covarância sob Heterocedasticidade de Forma Desconhecida.* RBE, Rio de Janeiro 56(2), p.309-334.

[6] Cribari Neto, F (2002). *C for Econometricians.* conputational Economics 14. p.135-149.

[7] Dalgaard, Peter(2008). *Introductory Statistics with R.* Editora Springer. Second Edition. New York, USA.

[8] Ehlers,Ricardo.(2007): Análise de series Temporais. Universidade Federal do Paraná.

[9] Frery, Alejandro; Cribari-Neto, Francisco.(2011).*Elementos de Estatística conputacional Usando Plataformas de Software Livre/Gratuito.* Publicações Matemáticas. Editora IMPA.

[10] Goldreich, O. and Micali, S. (1986). *How to construct random functions.* Journal of the Association for conputing Machinery, 33(4), p.792 − 807.

[11] Gujarati, Damodar N (2000). *Econometria Básica.* Makron Books. Terceira Edição. son Paulo.

[12] Knuth, D.E. (1981).*The Art of conputer Programming, Third Edition. Volume 2: Seminumerical. Algorithms*, Addison-Wesley Publishing conpany, Massachusetts.

[13] Korgi, Rodrigo de Castro (2003). *El universo LaTeX*. Editora Facultad de Ciencias. Segunda Edição. Bogota.

[14] Krawczyk, H. *How to Predict Congruential Generators*. In:Journal of Algorithms, V. 13, N. 4, 1992.

[15] L'Ecuyer, P (2001). *Software for uniform random number generation: distinguishing the good and the bad*. In Proceedings of the 2001 Winter Simulation Conference.

[16] Maindonald, John; Braun, W John(2010). *Data analysis and graphics using R : an example-based approach*. Cambridge University Press, New York, USA.