TP1, Modelo Lineal

Gonzalo Barrera Borla y Octavio Martín Duarte

5 de Junio de 2019

Consignas

Escriba y entregue en un *script* un programa de R que haga lo siguiente.

a) Fije la Semilla

i. Para $n = 10$ genere n datos y_i que sigan el modelo lineal $y_i = 4 + 2 \cdot x_{i1} - 3 \cdot x_{i2} + 0,5 \cdot x_{i3} + \varepsilon_i$, $1 \leq i \leq n$. Donde

- $x_{1i} \sim \mathcal{U}(-5, 5), iid.$
- $x_{2i} \sim \mathcal{U}(-5, 5), iid.$
- $x_{3i} \sim \mathcal{U}(-5, 5), iid.$
- $x_{4i} \sim \mathcal{U}(-5, 5), iid.$
- $\varepsilon_{1i} \sim \mathcal{E}(\lambda = 1/2) - 2, iid.$

ii. Ajuste el modelo $y_i = \beta_0 + \beta_1 \cdot x_{i1} - \beta_2 \cdot x_{i2} + \beta_3 \cdot x_{i3} + \beta_4 \cdot x_{i4} + u_i$

iii. Guarde los parámetros estimados.

iv. Construya el intervalo de confianza de nivel 0.90 para el parámetro β_1 y para el parámetro β_4 asumiendo normalidad de los errores. ¿Contienen estos intervalos a los verdaderos parámetros para la muestra simulada? Guarde en un nuevo objeto un uno si lo contiene, y un cero sino, para cada uno de los dos intervalos.

v. Construya el intervalo de confianza de nivel asintótico 0.90 para el parámetro β_1 y para el parámetro β_4. ¿Contienen estos intervalos a los verdaderos parámetros para la muestra simulada? Guarde en un nuevo objeto un 1 si lo contiene, y un cero sino, para cada uno de los dos intervalos.

vi. Repita los items a)i) hasta a)v) B = 1000 veces, de modo de tener una muestra de tamaño B de los estimadores de cada β_j. ¿Diría que la distribución on de los estimadores de β_2 puede aproximarse por la normal? Haga gráficos que le permitan tomar esta decisión. ¿Qué proporción de los B intervalos calculados para β_1 y β_4 basados en una muestra de n observaciones contuvo al verdadero valor del parámetro? Responda para cada tipo de intervalo calculado.

b) Repita a para $n = 25$ y $n = 100$.

c) Repita a y b para el caso de tener errores con distribución $Lognormal(\mu, \sigma^2) - e^{\mu+\sigma^2/2}$, tomando $\mu = 0$ y $\sigma^2 = 1$. Si para alguna de las distribuciones no consigue convencerse de que los $\hat{\beta}$ tienen distribución que puede ser aproximada por una normal, repita para errores generados con esta distribución en el esquenma de simulación anterior pero con $n = 250, 500, 1000, 15000, 2000, 3000$. Exhiba los resultados en una tabla y comente brevemente sus conclusiones.

d) Repita c pero ahora con la distribución de errores $\mathcal{U}(-3; 3)$ y con ${\chi^2}_k - k$ con $k = 3$ y t_k con $k = 3$.

Desarrollo

Método Empleado

Dado que se solicita muchas repeticiones de consignas similares variando ciertos parámetros, comenzamos por elaborar una gran cantidad de simulaciones con todas las distribuciones contempladas para el error (y algunas de nuestra añadidura) con la máxima cantidad de repeticiones y el máximo número de muestras por simulación. Después vamos a ir acudiendo a ellas para tomar subconjuntos. Esto lo hicimos trivialmente y sin remuestrear, tomando los primeros n elementos de cada simulación dado que como estas son efectivamente simulaciones de procesos aleatorios no nos pareció que el remuestreo fuera crítico a nuestros fines.

Dejamos una serie de parámetros que permitirían generalizar aún más la simulación, el modelo del proceso generador de datos `beta_pgd`, el conjunto de `n` adoptados, la cantidad de simulaciones `n_sim`, etc.

Dado que esta tabla y la siguiente involucran cantidades nada despreciables de cálculos, acá adjuntamos una versión del código similar a la usada pero parametrizada para muchas menos repeticiones, al lado aparecen comentados los verdaderos vectores que se usaron.

Parámetros

```
library('tidyverse')
library('stats')
library('future')
library('furrr')
```

```r
library('knitr')
set.seed(42)

# Coeficientes "platonicos" (i.e., del proceso generador de datos)
beta_pgd <- c(4, 2, -3, 0.5, 0)

metodos_intervalo <- c("asintotico", "exacto")
alfa <- 0.1

# Funciones generadoras de x_i
generadores_x <- list(
    "x1" = function(n) { runif(n, min=-5, max=5) },
    "x2" = function(n) { runif(n, min=-5, max=5) },
    "x3" = function(n) { runif(n, min=-5, max=5) },
    "x4" = function(n) { runif(n, min=-5, max=5) }
)

generadores_eps <- list(
  "normal" = function(n) { rnorm(n) },
  "exponencial" = function(n) { rexp(n, rate = 1/2) - 2 },
  "lognormal" = function(n) { exp(rnorm(n) - exp(0.5))  },
  "uniforme" = function(n) { runif(n, -3, 3) },
  "chi_cuadrado" = function(n) { rchisq(n, 3) - 3 },
  "student1" = function(n) { rt(n, 1) },
  "student3" = function(n) { rt(n, 3) }
)

funciones_a <- list(
  beta1 = c(0, 1, 0, 0, 0),
  beta4 = c(0, 0, 0, 0, 1)
)

generador_y <- function(x1, x2, x3, x4, beta_pgd, eps, ...)   {
  c(1, x1, x2, x3, x4) %*% beta_pgd + eps
}
```

Obtención de la muestra.

Para el vector con todos los valores de n, el conjunto de muestras pesa 3GB. Por esta razón, optamos por trabajar con otra tabla que acude a la tabla con las muestras cuando son necesarias y toma los datos pedidos para hallar los intervalos. Pusimos la salida de ambos tibble mostrando su forma, `muestras_maestras` es el conjunto de las muestras y `muestras_puntuales` conserva sólo la información necesaria para buscar en la tabla más grande.

```r
generar_muestra <- function(n, generadores_x, generador_eps, beta_pgd) {
  # Tibble vacio
  df <- tibble(.rows = n)
  # Genero variables regresoras y errores
  for (nombre in names(generadores_x)) {
    if (nombre != "y") {
      df[nombre] <- generadores_x[[nombre]](n)
    }
  df$eps <- generador_eps(n)
  }
```

```
  # Genero y
  df["y"] <- pmap_dbl(df, generador_y, beta_pgd=beta_pgd)

  return(df)
}

ayudante_generar_muestra <- function(distr_eps, generadores_x, beta_pgd, n) {
  generar_muestra(n,generadores_x, generadores_eps[[distr_eps]],beta_pgd=beta_pgd)
}


n_muestrales <- c(10, 25)
#n_muestrales <- c(10, 25, 100, 250, 500, 1000, 1500, 2000, 3000)

max_n_muestral <- max(n_muestrales)
n_sims <- 1000
muestras_maestras <- crossing(
  n_sim = seq(max_n_muestral),
  distr_eps = names(generadores_eps)) %>%
  mutate(
    muestra = future_map(.progress=TRUE,
                  distr_eps,
                  ayudante_generar_muestra,
                  generadores_x = generadores_x,
                  beta_pgd = beta_pgd,
                  n = max_n_muestral)
  )

muestras_maestras
```

```
## # A tibble: 175 x 3
##    n_sim distr_eps    muestra
##    <int> <chr>        <list>
##  1     1 chi_cuadrado <tibble [25 x 6]>
##  2     1 exponencial  <tibble [25 x 6]>
##  3     1 lognormal    <tibble [25 x 6]>
##  4     1 normal       <tibble [25 x 6]>
##  5     1 student1     <tibble [25 x 6]>
##  6     1 student3     <tibble [25 x 6]>
##  7     1 uniforme     <tibble [25 x 6]>
##  8     2 chi_cuadrado <tibble [25 x 6]>
##  9     2 exponencial  <tibble [25 x 6]>
## 10     2 lognormal    <tibble [25 x 6]>
## # ... with 165 more rows
```

```
#El '-3' es poco legible, buscar cómo sustraer una columna por nombre.

muestras_puntuales <- muestras_maestras[-3] %>%
  crossing(
    n = n_muestrales
  )

muestras_puntuales
```

```
## # A tibble: 350 x 3
```

```
##    n_sim distr_eps        n
##    <int> <chr>        <dbl>
##  1     1 chi_cuadrado    10
##  2     1 chi_cuadrado    25
##  3     1 exponencial     10
##  4     1 exponencial     25
##  5     1 lognormal       10
##  6     1 lognormal       25
##  7     1 normal          10
##  8     1 normal          25
##  9     1 student1        10
## 10     1 student1        25
## # ... with 340 more rows
```

Intervalos

Para obtener los intervalos, usamos los parámetros que tiene guardada cada fila de `muestras_puntuales` y ejecutamos una función que lee en `muestras_maestras` de acuerdo a estos.

```
intervalo_conf <- function(a_vec, llamada_lm, alfa, metodo = "exacto") {

  betahat <- llamada_lm$coefficients
  # Matriz de covarianza estimada para los coeficientes
  Sigmahat <- vcov(llamada_lm)

  n_muestra <- nrow(llamada_lm$model)
  r <- llamada_lm$rank
  # Cualculo cuantil t o z, segun corresponda
  if (metodo == "exacto") {
    cuantil <- qt(p = 1 - alfa/2, df = n_muestra - r)
  } else if (metodo == "asintotico") {
    cuantil <- qnorm(p = 1 - alfa/2)
  } else {
    stop("Los unicos metodos soportados son 'exacto' y 'asintotico'")
  }

  centro <- t(a_vec)%*%betahat
  delta <- cuantil * sqrt(t(a_vec) %*% Sigmahat %*% a_vec)
  return(c(centro - delta, centro + delta))
}

cubre <- function(intervalo, valor) { intervalo[1] <= valor & intervalo[2] >= valor}

ayudante_intervalo_conf <- function(n_simulacion, distr_epsilon, n, fun_a, met_int, alfa) {
  muestra_a_evaluar <- (muestras_maestras %>% filter(n_sim==n_simulacion,distr_eps==distr_epsilon))[[1,
  modelo <- lm(y ~ x1 + x2 + x3 +x4,data=muestra_a_evaluar)
  intervalo_conf(a_vec = funciones_a[[fun_a]], llamada_lm=modelo, alfa=alfa, metodo = met_int)
}

intervalos <- muestras_puntuales %>%
  crossing(
    fun_a = names(funciones_a),
    met_int = metodos_intervalo) %>%
  mutate(
```

```
    #atbeta es el valor del parámetro en el PGD.
    atbeta = map_dbl(fun_a, function(i) funciones_a[[i]] %*% beta_pgd),
    ic = future_pmap( .progress = TRUE,
      list(n_sim, distr_eps, n, fun_a, met_int),
      ayudante_intervalo_conf,
      alfa = alfa),
    cubre = map2_lgl(ic, atbeta, cubre),
    ic_low = map_dbl(ic, 1),
    ic_upp = map_dbl(ic, 2)
    )
```

Respuestas

```
intervalos <- read_rds("simulacion.Rds") %>%
  mutate(
    estimador = (ic_upp+ic_low)/2
  )
```

¿Los intervalos Cubren a los Parámetros?

Respondemos directamente para todas las distribuciones estudiadas, con muestras de diez iteraciones.

Para $B = 1000$ **y** $n = 10$

```
sintesis <- intervalos %>%
  filter(n_sim<=1000) %>%
  group_by(distr_eps, n, met_int, fun_a) %>%
  summarise(prop_cubre = mean(cubre))

sintesis %>%
  filter(n==10)  %>%
kable()
```

distr_eps	n	met_int	fun_a	prop_cubre
chi_cuadrado	10	asintotico	beta1	0.841
chi_cuadrado	10	asintotico	beta4	0.848
chi_cuadrado	10	exacto	beta1	0.905
chi_cuadrado	10	exacto	beta4	0.910
exponencial	10	asintotico	beta1	0.860
exponencial	10	asintotico	beta4	0.865
exponencial	10	exacto	beta1	0.917
exponencial	10	exacto	beta4	0.915
lognormal	10	asintotico	beta1	0.842
lognormal	10	asintotico	beta4	0.825
lognormal	10	exacto	beta1	0.893
lognormal	10	exacto	beta4	0.890
normal	10	asintotico	beta1	0.841
normal	10	asintotico	beta4	0.852
normal	10	exacto	beta1	0.899
normal	10	exacto	beta4	0.910
student1	10	asintotico	beta1	0.858
student1	10	asintotico	beta4	0.856

distr_eps	n	met_int	fun_a	prop_cubre
student1	10	exacto	beta1	0.923
student1	10	exacto	beta4	0.913
student3	10	asintotico	beta1	0.853
student3	10	asintotico	beta4	0.856
student3	10	exacto	beta1	0.909
student3	10	exacto	beta4	0.909
uniforme	10	asintotico	beta1	0.858
uniforme	10	asintotico	beta4	0.823
uniforme	10	exacto	beta1	0.903
uniforme	10	exacto	beta4	0.900

Observamos que para muestras pequeñas, con $n = 10$, es usual que los intervalos asintóticos no lleguen a cubrir los parámetros. En varios casos incluido el exponencial nuestra media de aciertos está por debajo del α establecido. Esto no mejora incrementando las repeticiones hasta 3.000 .

Para $B = 3000$ **y** $n = 10$

```
sintesis <- intervalos %>%
  group_by(distr_eps, n, met_int, fun_a) %>%
  summarise(prop_cubre = mean(cubre))

sintesis %>%
  filter(n==10)  %>%
kable()
```

distr_eps	n	met_int	fun_a	prop_cubre
chi_cuadrado	10	asintotico	beta1	0.8400000
chi_cuadrado	10	asintotico	beta4	0.8463333
chi_cuadrado	10	exacto	beta1	0.9020000
chi_cuadrado	10	exacto	beta4	0.9056667
exponencial	10	asintotico	beta1	0.8526667
exponencial	10	asintotico	beta4	0.8503333
exponencial	10	exacto	beta1	0.9086667
exponencial	10	exacto	beta4	0.9070000
lognormal	10	asintotico	beta1	0.8376667
lognormal	10	asintotico	beta4	0.8353333
lognormal	10	exacto	beta1	0.8970000
lognormal	10	exacto	beta4	0.8986667
normal	10	asintotico	beta1	0.8473333
normal	10	asintotico	beta4	0.8526667
normal	10	exacto	beta1	0.9070000
normal	10	exacto	beta4	0.9083333
student1	10	asintotico	beta1	0.8390000
student1	10	asintotico	beta4	0.8533333
student1	10	exacto	beta1	0.9110000
student1	10	exacto	beta4	0.9150000
student3	10	asintotico	beta1	0.8373333
student3	10	asintotico	beta4	0.8446667
student3	10	exacto	beta1	0.8973333
student3	10	exacto	beta4	0.9000000
uniforme	10	asintotico	beta1	0.8400000

distr_eps	n	met_int	fun_a	prop_cubre
uniforme	10	asintotico	beta4	0.8280000
uniforme	10	exacto	beta1	0.8993333
uniforme	10	exacto	beta4	0.8993333

Veamos en qué casos no estamos logrando cubrir el valor real del parámetro.

```
sintesis %>%
  filter(n==10,prop_cubre<0.9) %>%
  kable()
```

distr_eps	n	met_int	fun_a	prop_cubre
chi_cuadrado	10	asintotico	beta1	0.8400000
chi_cuadrado	10	asintotico	beta4	0.8463333
exponencial	10	asintotico	beta1	0.8526667
exponencial	10	asintotico	beta4	0.8503333
lognormal	10	asintotico	beta1	0.8376667
lognormal	10	asintotico	beta4	0.8353333
lognormal	10	exacto	beta1	0.8970000
lognormal	10	exacto	beta4	0.8986667
normal	10	asintotico	beta1	0.8473333
normal	10	asintotico	beta4	0.8526667
student1	10	asintotico	beta1	0.8390000
student1	10	asintotico	beta4	0.8533333
student3	10	asintotico	beta1	0.8373333
student3	10	asintotico	beta4	0.8446667
student3	10	exacto	beta1	0.8973333
uniforme	10	asintotico	beta1	0.8400000
uniforme	10	asintotico	beta4	0.8280000
uniforme	10	exacto	beta1	0.8993333
uniforme	10	exacto	beta4	0.8993333

Para $B = 3000$ **y** $n = 25$

```
sintesis %>%
  filter(n==25)  %>%
kable()
```

distr_eps	n	met_int	fun_a	prop_cubre
chi_cuadrado	25	asintotico	beta1	0.8903333
chi_cuadrado	25	asintotico	beta4	0.8853333
chi_cuadrado	25	exacto	beta1	0.9073333
chi_cuadrado	25	exacto	beta4	0.9026667
exponencial	25	asintotico	beta1	0.8716667
exponencial	25	asintotico	beta4	0.8906667
exponencial	25	exacto	beta1	0.8920000
exponencial	25	exacto	beta4	0.9053333
lognormal	25	asintotico	beta1	0.8940000
lognormal	25	asintotico	beta4	0.8823333
lognormal	25	exacto	beta1	0.9070000
lognormal	25	exacto	beta4	0.9000000

distr_eps	n	met_int	fun_a	prop_cubre
normal	25	asintotico	beta1	0.8883333
normal	25	asintotico	beta4	0.8883333
normal	25	exacto	beta1	0.9013333
normal	25	exacto	beta4	0.9023333
student1	25	asintotico	beta1	0.8893333
student1	25	asintotico	beta4	0.8820000
student1	25	exacto	beta1	0.9056667
student1	25	exacto	beta4	0.9003333
student3	25	asintotico	beta1	0.8913333
student3	25	asintotico	beta4	0.8743333
student3	25	exacto	beta1	0.9100000
student3	25	exacto	beta4	0.8930000
uniforme	25	asintotico	beta1	0.8850000
uniforme	25	asintotico	beta4	0.8833333
uniforme	25	exacto	beta1	0.9000000
uniforme	25	exacto	beta4	0.8960000

Para $B = 3000$ **y** $n = 100$

```
sintesis %>%
  filter(n==100)  %>%
kable()
```

distr_eps	n	met_int	fun_a	prop_cubre
chi_cuadrado	100	asintotico	beta1	0.9013333
chi_cuadrado	100	asintotico	beta4	0.8956667
chi_cuadrado	100	exacto	beta1	0.9026667
chi_cuadrado	100	exacto	beta4	0.8983333
exponencial	100	asintotico	beta1	0.8956667
exponencial	100	asintotico	beta4	0.9010000
exponencial	100	exacto	beta1	0.9003333
exponencial	100	exacto	beta4	0.9053333
lognormal	100	asintotico	beta1	0.9000000
lognormal	100	asintotico	beta4	0.9113333
lognormal	100	exacto	beta1	0.9026667
lognormal	100	exacto	beta4	0.9136667
normal	100	asintotico	beta1	0.8970000
normal	100	asintotico	beta4	0.8983333
normal	100	exacto	beta1	0.9010000
normal	100	exacto	beta4	0.9020000
student1	100	asintotico	beta1	0.9033333
student1	100	asintotico	beta4	0.9043333
student1	100	exacto	beta1	0.9066667
student1	100	exacto	beta4	0.9076667
student3	100	asintotico	beta1	0.8983333
student3	100	asintotico	beta4	0.8980000
student3	100	exacto	beta1	0.9016667
student3	100	exacto	beta4	0.9013333
uniforme	100	asintotico	beta1	0.8993333
uniforme	100	asintotico	beta4	0.8946667
uniforme	100	exacto	beta1	0.9030000

distr_eps	n	met_int	fun_a	prop_cubre
uniforme	100	exacto	beta4	0.8990000

Veamos en qué casos no estamos logrando cubrir el valor real del parámetro. Vamos además a hacer un gráfico que compare cómo evoluciona nuestra media de intervalos que cubren en cada distribución del error de acuerdo al parámetro n.

```
sintesis %>%
  filter(n==100,prop_cubre<0.9)  %>%
kable()
```

distr_eps	n	met_int	fun_a	prop_cubre
chi_cuadrado	100	asintotico	beta4	0.8956667
chi_cuadrado	100	exacto	beta4	0.8983333
exponencial	100	asintotico	beta1	0.8956667
normal	100	asintotico	beta1	0.8970000
normal	100	asintotico	beta4	0.8983333
student3	100	asintotico	beta1	0.8983333
student3	100	asintotico	beta4	0.8980000
uniforme	100	asintotico	beta1	0.8993333
uniforme	100	asintotico	beta4	0.8946667
uniforme	100	exacto	beta4	0.8990000

Se observa un incremento significativo en todas las tendencias a cubrir el valor real. Ninguno de `prop_cubre` está ahora drásticamente lejos del nivel deseado y cabe esperar lograr este con un nuevo incremento de n.

```
intervalos %>%
  filter(distr_eps=='exponencial',cubre==FALSE, n<=3000, n_sim <= 3000) %>%
  ggplot(aes(x = met_int, y=..count../sum(..count..) , fill=n, color = met_int)) +
    geom_bar() +
    facet_grid(fun_a~n)
```