Problema 1 - Simulación de Montecarlo

```
clear;
close all;
% Global settings
warning('off'); % warnings about rank
figure('Renderer', 'painters', 'Position', [10 10 900 600]); % figures size
```

El propósito de este ejercicio es utilizar de manera eficiente ciertos comandos de Matlab tales como matrices, loops, while, plot y subplot. Se aplican diversos conocimientos econométricos y se han desarrollado algoritmos de la forma más eficientes posible. Las funciones trabajadas se pueden acceder aquí:

- xx.m
- plotxx.m
- random.m
- algoritmo.m
- algoritmo2.m

El modelo de estudio está dado por:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_3 + \beta_4 \cdot x_4 + \beta_5 \cdot x_5 + \beta_5 \cdot x_5 + \beta_6 \cdot x_6 + \epsilon \qquad (1)$$

(a) Distribución de variables

0. Parámetros

```
r = 1000; % Muestra de tamaño 1000
c = 1;
x = rand(r,c);
x_i = randn(r,c);
```

1. $x_1 \sim U(1,8)$

```
x1 = x*(8-1)+1;
```

2. $x_2 \sim N(3,4)$

Se debe construir con una matriz de números aleatorios con distribución normal, con media 3 y desvío 4 (con randn)

```
x2 = x_i*4 + 3;
```

3. $x_3 \sim X_{100}^2$

Chi cuadrado de 100 gl usando randn que genere una normal de media 0 y varianza 1

```
x3 = 100*(x_i.^2);
```

4. $x_4 \sim (t \text{ student})_2$

t student 2 gl

```
x4a = 0;
for i = 1:2
    x4a = x4a + randn(r,c).^2;
    i = i+1;
end

x4 = x_i./sqrt(x4a/2);
```

5. $x_5 \sim$ una mixtura tal que el 50% de las veces se comporta como x_2 y el 50% de las veces como x_3

Para generar esta distribución utilizaré mi variable aleatoria adicional (x) y a partir de ella seleccionaré una muestra donde con un 50% de probabilidad se recupere el valor de x2 o en caso contrario x3

```
x5 = zeros(r,c);

for i = 1:r
    if x (i,1) < 0.5
        x5(i,1) = x2(i, 1);
    else
        x5(i) = x3(i,1);
    end
    i = i+1;
end
```

6. x_6 es un promedio entre x_4 y x_5 más un error normal con media 0 y desvío 0.1

X6: promedio de X4 y X5 mas un error normal con media cero y desvio 0.0001.

```
error = x_i* 0.1;
x6 = (x4+x5)/2 + error;
```

7. $y \sim \exp(0.6)$

```
% --> 1-rand = exp(-lamda*x) --> log(1-rand) = (-lamda*x) -->
% log(1-rand)/-lamda = x
y = log(1-x)/(-0.6);
```

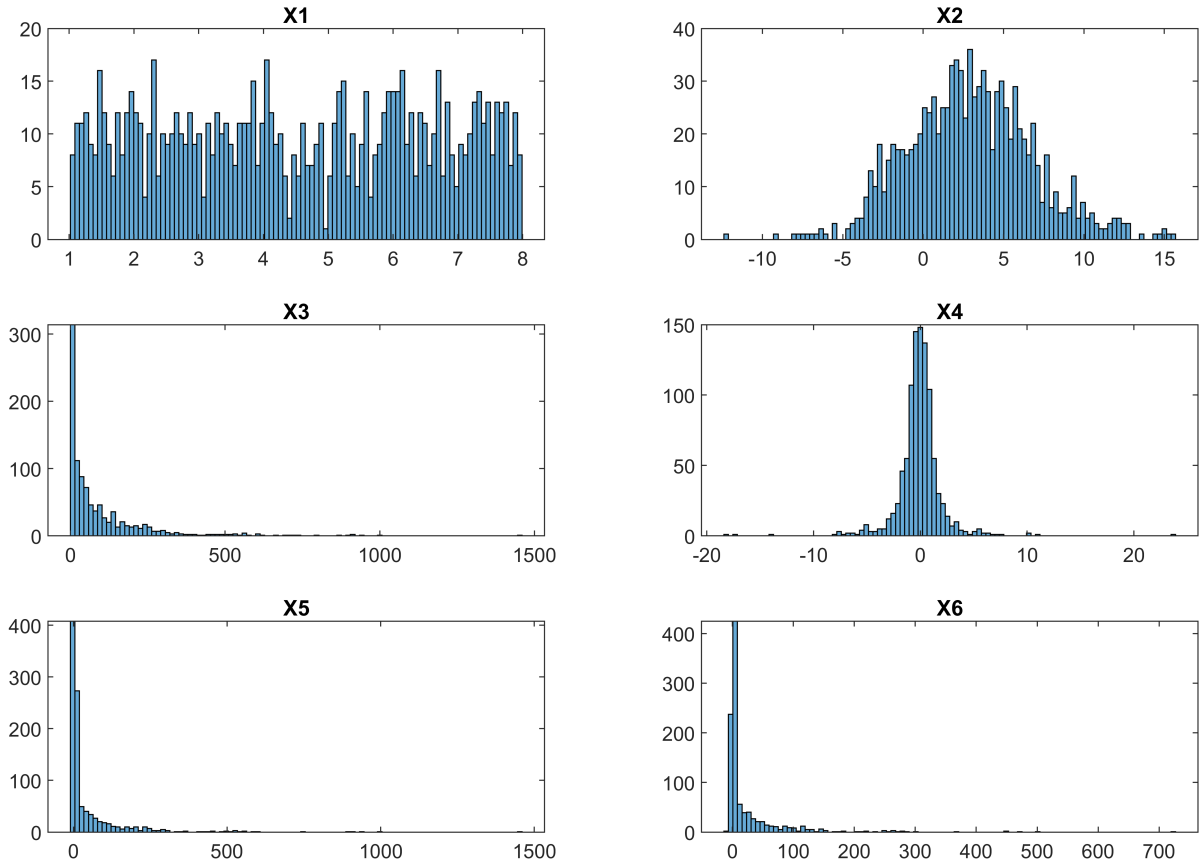
8. $\epsilon \sim N(0, 1)$

```
epsilon = x_i;
```

Se ha creado una función xx.m que construye las distribuciones de las variables indicadas, y que recibe como input el número de observaciones. Con la función plotxx.m graficaremos estas distribuciones en un histograma

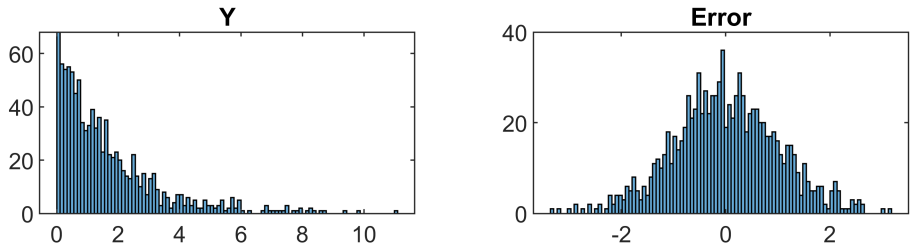
```
mm = xx(1000,1);
```

```
plotxx(mm, 1000,1)
```



Además también se muestra la distribución de **y** y el **error**

```
figure(2)
subplot(3,2,1)
histogram(y,100)
title('Y')
subplot(3,2,2)
histogram(epsilon,100)
title('Error')
hold off
```



(b) Genere 1000 muestras de tamaño n = 100 para todas la variables x

```
x_i = randn(100,1000)
```

```
x_i = 100×1000
   -0.2720    0.1553   -0.3770    1.0696   -0.5956   -0.1079    0.3474   -0.6334 ···
   -0.9596    0.5499    0.3887   -0.5196   -0.2508    0.4376    0.4582   -0.6959
   -1.5208    0.6736   -1.0119   -0.1829   -2.0685   -2.0307   -0.5910   -0.5256
    0.1852   -1.0349   -0.9492   -0.4233    0.7865   -0.4649    0.7339    1.5388
    1.1726   -0.3060    0.0071    0.1174    0.6798    0.3386   -1.0877    0.0605
   -0.2241   -0.8126   -0.3462   -0.3816    0.3767   -1.1032    1.1843    0.1701
   -0.0860    1.2470    0.3322   -0.5007    1.3763   -0.6115    0.6706    0.2726
    1.2747    0.6493    0.1550    0.0374   -0.4622   -0.7172   -0.4497    0.7486
    0.7079   -0.1880   -0.6242   -0.7194   -0.2868    3.5675   -0.0819   -1.3496
    0.3679    1.0939   -0.2602   -0.8267   -1.3888   -0.3930    1.2699    0.9138
      ⋮
```

```
x = rand(100,1000)
```

```
x = 100×1000
    0.2078    0.9048    0.3286    0.6572    0.8834    0.9197    0.9779    0.9924 ···
    0.9898    0.1602    0.8020    0.0652    0.8384    0.1964    0.2477    0.1391
    0.4238    0.5884    0.6714    0.6296    0.4409    0.9569    0.6822    0.5366
    0.4454    0.3916    0.3526    0.3638    0.0378    0.8326    0.9518    0.4757
    0.2470    0.5432    0.8325    0.3899    0.7834    0.3060    0.1718    0.6726
    0.1920    0.0787    0.6638    0.2432    0.9349    0.9174    0.4710    0.5426
    0.7278    0.8289    0.3817    0.6866    0.5631    0.5331    0.0569    0.3914
    0.6388    0.2568    0.2956    0.2729    0.6392    0.9221    0.2973    0.8296
    0.1987    0.5360    0.0640    0.2369    0.3808    0.2521    0.2520    0.6907
```

```
    0.5544    0.3000    0.7020    0.8916    0.4196    0.9728    0.0785    0.4538
     ⋮
```

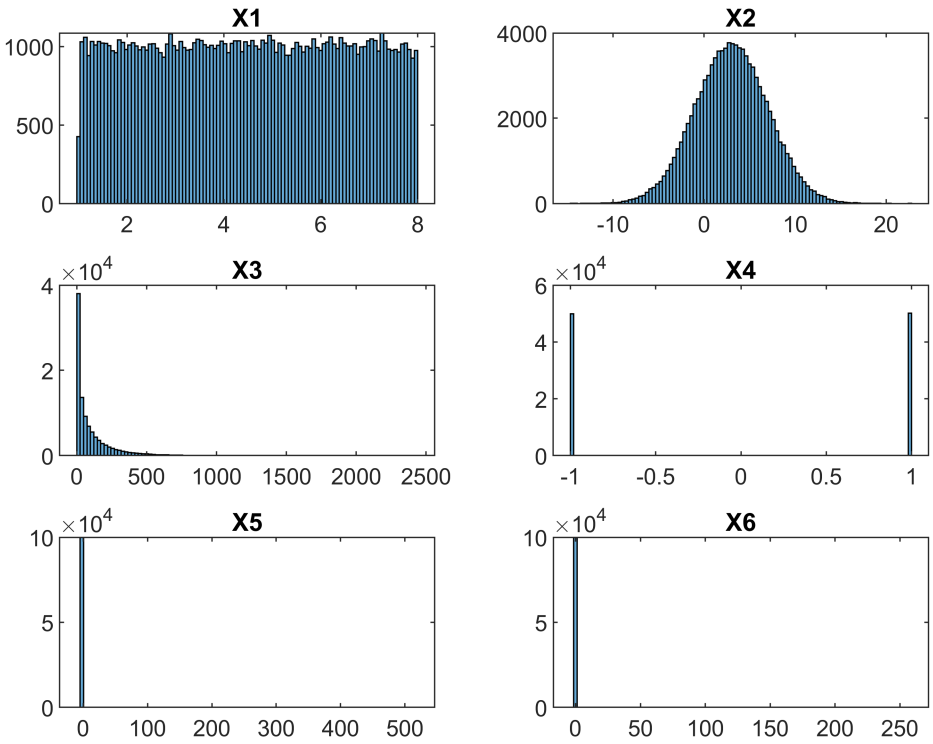
(c) Crear función

Cree la funcion random que recibe como inputs el número de muestras y el tamaño de cada muestra. Además, debe entregar como output una matriz que compile todas las variables x(1−6) junto con entregar un subplot que indique como distribuye cada una de las variables (histograma).

```
random(100,1000)
```

```
ans = 100×6000
10^3 ×
    0.0013    0.0013    0.0045    0.0047    0.0055    0.0047    0.0021    0.0072 ···
    0.0026    0.0069    0.0031    0.0060    0.0033    0.0018    0.0022    0.0015
    0.0057    0.0044    0.0042    0.0046    0.0046    0.0064    0.0046    0.0058
    0.0048    0.0028    0.0077    0.0059    0.0041    0.0043    0.0068    0.0027
    0.0057    0.0066    0.0066    0.0059    0.0039    0.0033    0.0033    0.0051
    0.0077    0.0018    0.0080    0.0061    0.0075    0.0047    0.0051    0.0029
    0.0022    0.0048    0.0055    0.0019    0.0037    0.0048    0.0042    0.0076
    0.0050    0.0041    0.0057    0.0059    0.0063    0.0039    0.0035    0.0065
    0.0048    0.0057    0.0010    0.0016    0.0048    0.0069    0.0072    0.0025
    0.0048    0.0020    0.0047    0.0044    0.0030    0.0077    0.0041    0.0016
     ⋮
```

```
X = random(100, 1000);
```



(d) Compute la media, mediana, mínimo, máximo, varianza y percentiles 25 y 75 de las muestras obtenidas.

La tabla se lee como que cada fila corresponde a los estadísticos descriptivos de cada x_n

```
statistics = [ mean(X(:,1:1000:6000))' std(X(:,1:1000:6000))'  min(X(:,1:1000:6000))' max(X(:,1
```

```
statistics = 6×7
    4.7001    2.0812    1.1620    7.9631    4.8678    2.9692    6.5360
    2.7870    3.9903   -6.1136   11.1098    2.7745    0.0640    5.6551
   98.8051  119.6668    0.0002  519.1137   49.1089    7.0975  159.9057
   -0.0600    1.0032   -1.0000    1.0000   -1.0000   -1.0000    1.0000
   54.8542  100.3070   -5.3336  519.1137    6.3363    2.3476   51.2967
   27.3918   50.1399   -3.3751  258.8290    3.7516    0.6585   26.2200
```

OLS

Ahora, dado que se conoce la distribución de cada una de las variables x(1−6) y el vector y, aplicando la metodología OLS, estimar a los parámetros β1−6.

Sabemos que para poder calcular los coeficientes de regresión debemos utilizar la matrix de los X_n para luego calcular $\widehat{\beta} = (X'X)^{-1}X'\,y$.

Ahora bien, en **Matlab** es más rápido resolver como $\widehat{\beta} = (\mathrm{Xy})'$

(e) Compute el vector3 de estimadores $\beta_{7\mathrm{x}1}$ para una muestra arbitraria.

```
mm = [ones(1000,1) mm]
```

```
mm = 1000×7
10^3 ×
    0.0010    0.0011   -0.0021    0.1631   -0.0036   -0.0021   -0.0030
    0.0010    0.0023    0.0058    0.0497    0.0010    0.0058    0.0035
    0.0010    0.0053    0.0111    0.4086    0.0019    0.4086    0.2054
    0.0010    0.0067   -0.0007    0.0841   -0.0013    0.0841    0.0413
    0.0010    0.0015    0.0022    0.0043   -0.0004    0.0022    0.0009
    0.0010    0.0069   -0.0054    0.4362   -0.0024    0.4362    0.2167
    0.0010    0.0038    0.0023    0.0029   -0.0010    0.0023    0.0006
    0.0010    0.0020    0.0006    0.0369   -0.0007    0.0006   -0.0001
    0.0010    0.0048    0.0033    0.0005    0.0001    0.0005    0.0003
    0.0010    0.0048    0.0051    0.0265    0.0005    0.0265    0.0136
     ⋮
```

```
%beta_1 = inv(mm'*mm)*mm'*y
beta_1 = (mm\y)'
```

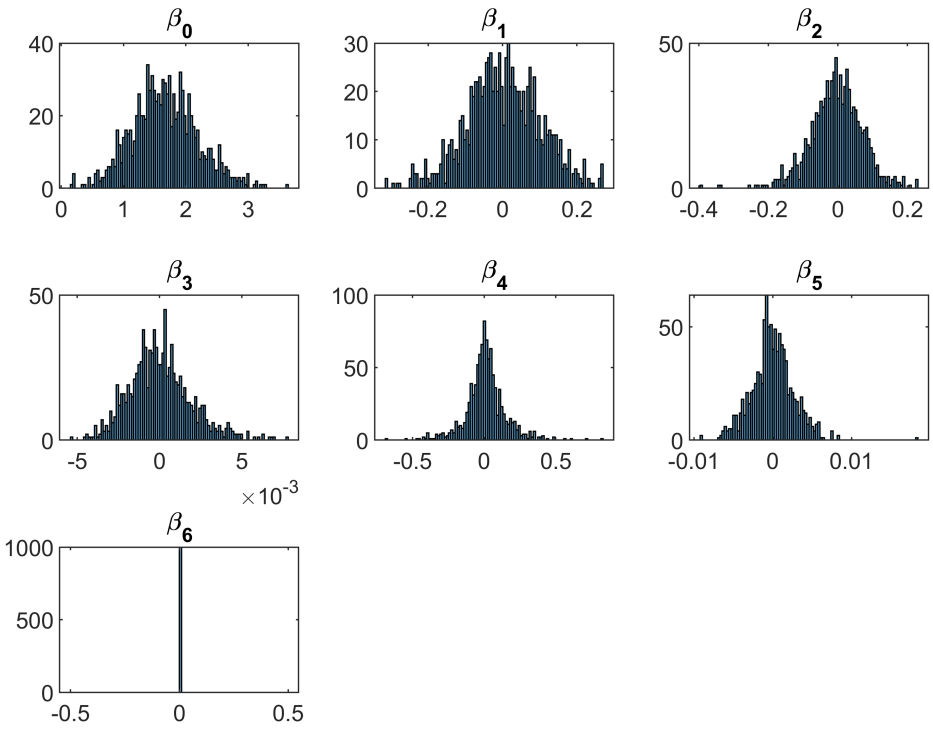
```
beta_1 = 1×7
    1.6549   -0.0124    0.0081    0.0003    0.0182    0.0000         0
```

(f) Compute una matriz que contenga los vectores de estimadores β_{ols} para 1000 muestras

```
for i=1:1000
    X = [(ones(100,1)) xx(100,1)];
    y = log(1-rand(100,1))/(-0.6);
    beta_2(i,:) = (X\y)';
end
```

Podemos representar los coeficientes de regresión de la siguiente manera

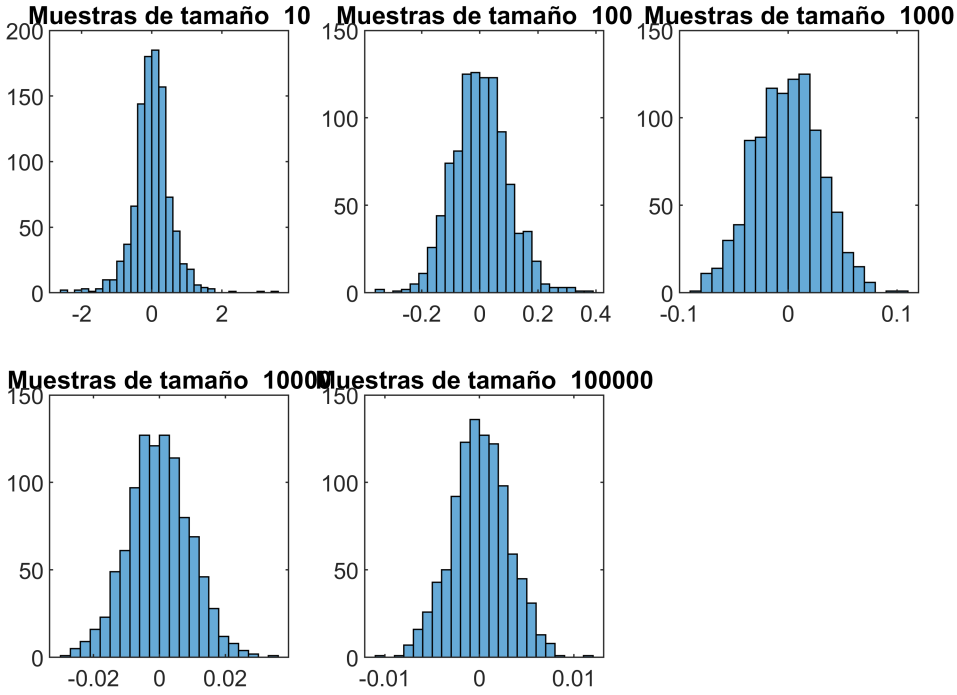
```
figure(5)
for j=1:7
    subplot(3,3,j);
    histogram(beta_2(:,j),100);
    title(['\beta_',num2str(j-1)])
end
```



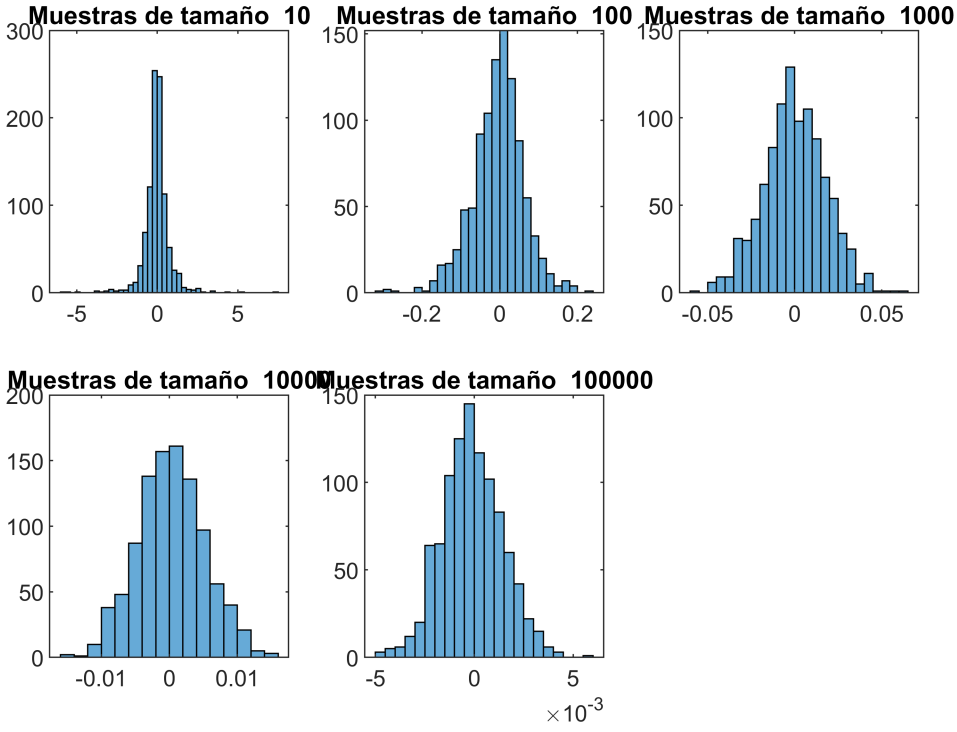
g) Desarrolle un algoritmo que reciba como *input* el **tamaño de cada muestra (m)** y el **número de muestras (i)** y que entregue como *output* un **subplot** donde se observe la distribución de cada estimador β_{ols} para cada figura asociada a cada tamaño de muestra. Utilice el vector $t[10^2\ 10^3\ 10^4\ 10^5]$ para los distintos tamaños de muestras.

```
m = 1000;
t = [10 10^2 10^3 10^4 10^5];
algoritmo(t,m)
```

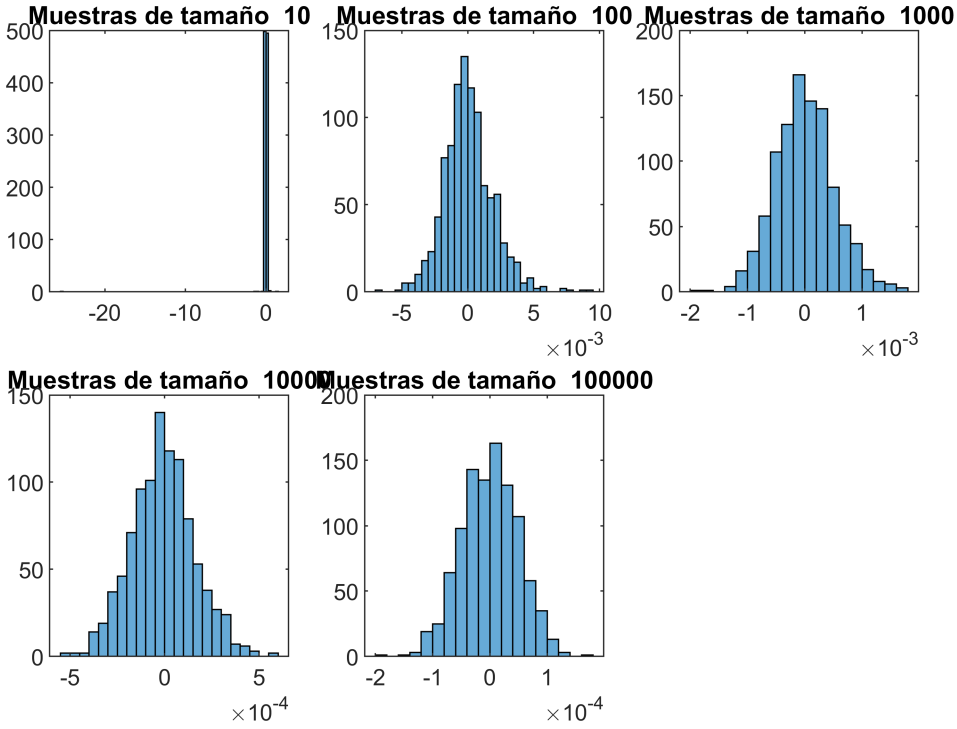
Distribución β_1



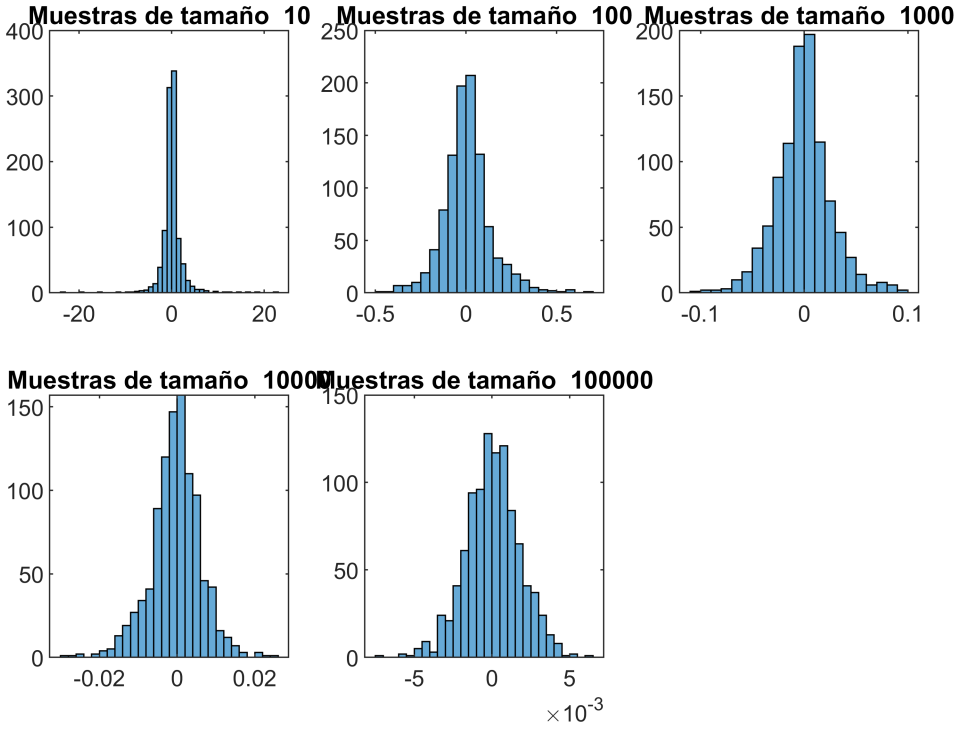
Distribución β_2



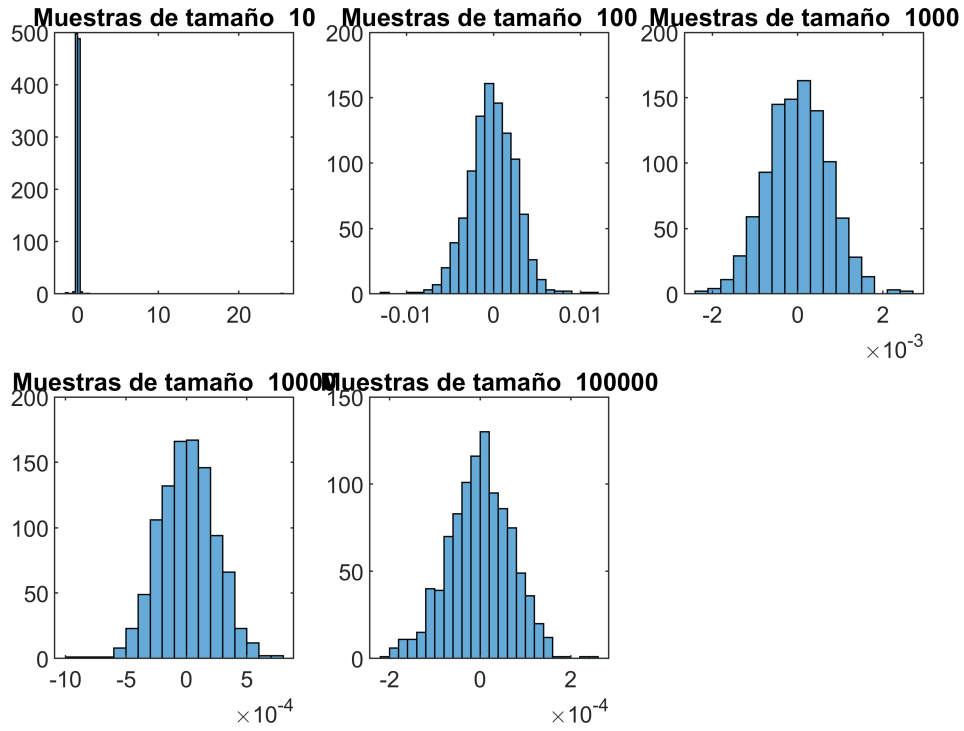
Distribución β_3



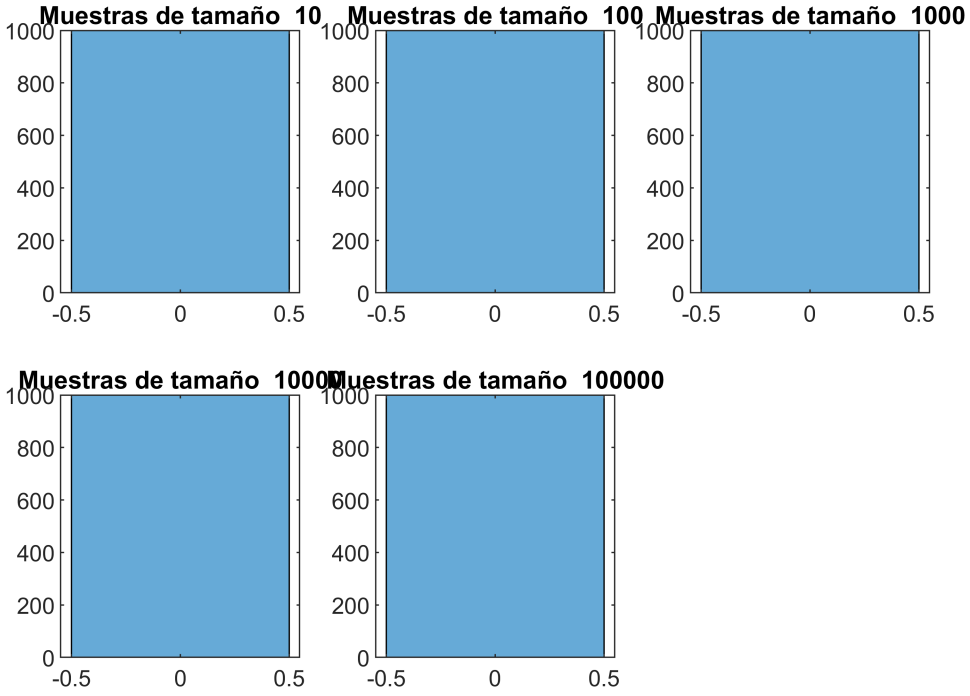
Distribución β_4



Distribución β_5



Distribución β_6




```
ans = struct with fields:
        ols_10: [1000×7 double]
       ols_100: [1000×7 double]
      ols_1000: [1000×7 double]
```

```
 ols_10000: [1000×7 double]
ols_100000: [1000×7 double]
```

Responda: ¿Cómo distribuyen los estimadores? ¿En torno a que valor están centradas las distribuciones? ¿Por qué? Interprete y explique brevemente

En el caso de los β_o, β_2 y β_4 podemos notar que las distribuciones están centradas en el cero, y a medida que aumenta el tamaño de la muestra su distribución es cada vez más simétrica y con menor curtosis. En estos dos casos es notable como contrasta esto con muestras de tamaño 10, donde son mucho más asimétricas y hay presencia de outliers en valores negativos en caso del intercepto, y en valores positivos en caso de el coeficiente 2.

En el caso de β_1 esta distribución no cambia mucho con el aumento del tamaño de la muestra. Mientras que en β_3 y β_5 las muestras de tamaño más pequeño tienen una curtosis notable que se va suavizando a medida que aumenta el tamaño de la

El caso de β_6 tenemos un valor también centrado en el cero, pero casi único. Recordemos que esta variable se calculada con el promedio entre x_4 y x_5 y estas dos tienen una distribución "espejo" donde su único valor que intersecta es el 0 (ver figuras iniciales).

En términos generales, gracias a estas gráficas podemos notificar el poder que tienen las simulaciones de Montecarlo. Montecarlo es una metodología de estudio en la cual, con la generación de una gran cantidad de números aleatorios, podemos determinar el comportamiento de un sistema y/o el valor de distintas variables.

La idea básica detrás de usar Montecarlo es ejecutar simulaciones una y otra vez para obtener la distribución probabilística de una entidad pobabilística desconocida. Los métodos numéricos, como Montecarlo, muchas veces son útiles cuando los métodos analíticos son muy difiíciles de resolver o la solución no existe. La razón: nos permite rescatar las **propiedades asintóticas** de la estimación de las regresiones. Y eso mismo estamos haciendo en estos ejercicios.

Estimación de coeficientes de regresión

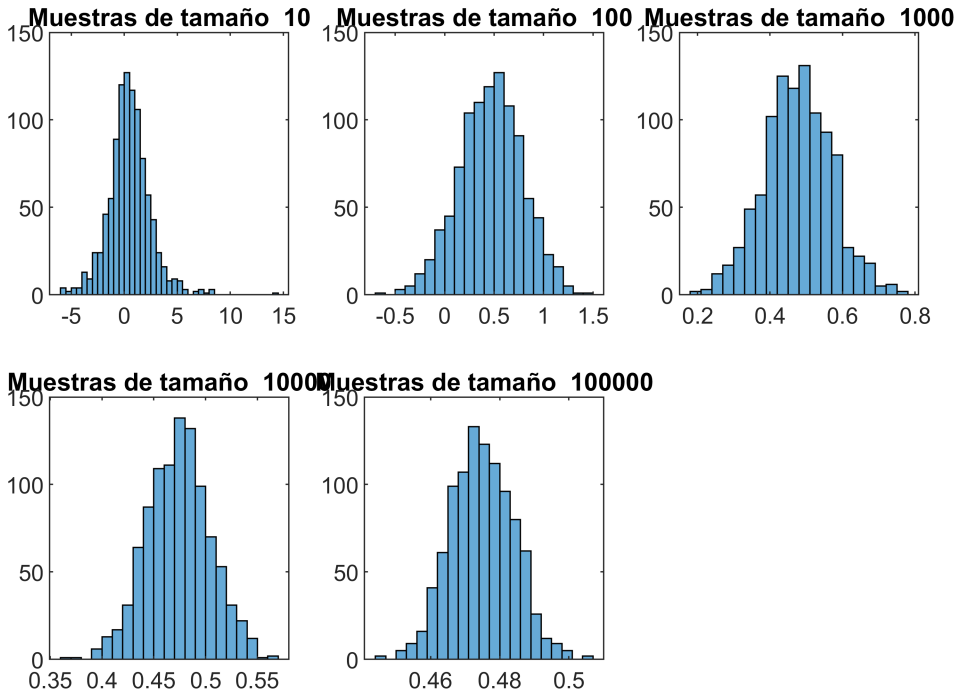
A partir de la última pregunta estimaremos el vector **y** haciendo uso del vector $\hat{\beta}_{\text{ols}}$ [1 2 3 4 5 6 7]' y del vector de residuos $\epsilon \sim N(0, 1)$

$$y = X\beta + \epsilon$$

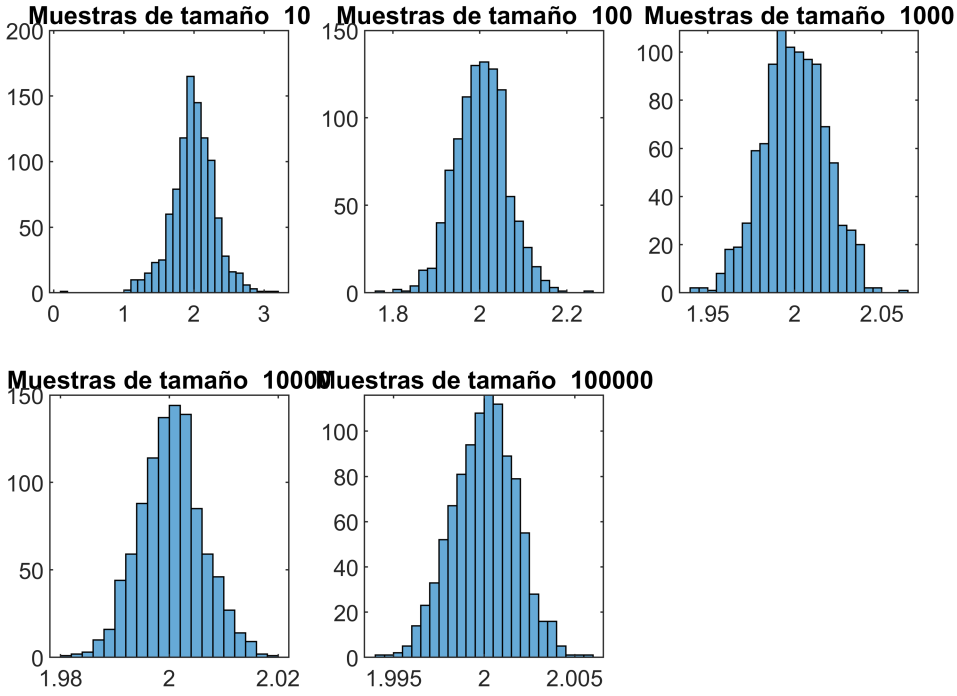
(h) De forma similar a lo desarrollado eb el ítem anterior, cree un algoritmo que entregue como output un subplot donde se observe la distribución de cada $\hat{\beta}_{\text{ols}}$ para cada tamaño de muestra distinta

```
algoritmo2(t,m)
```

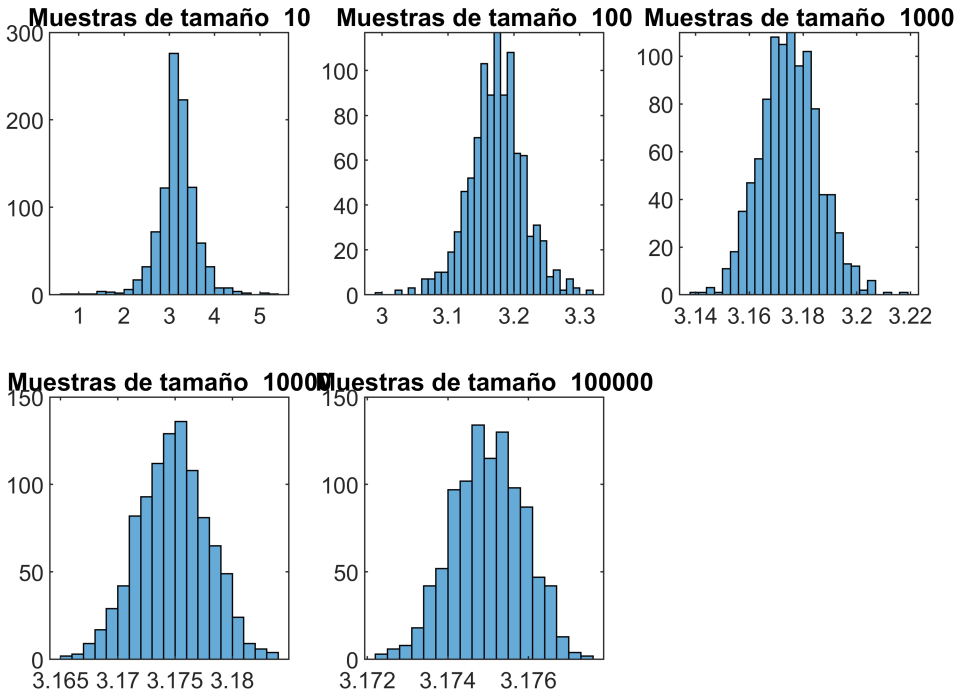
Distribución β_0



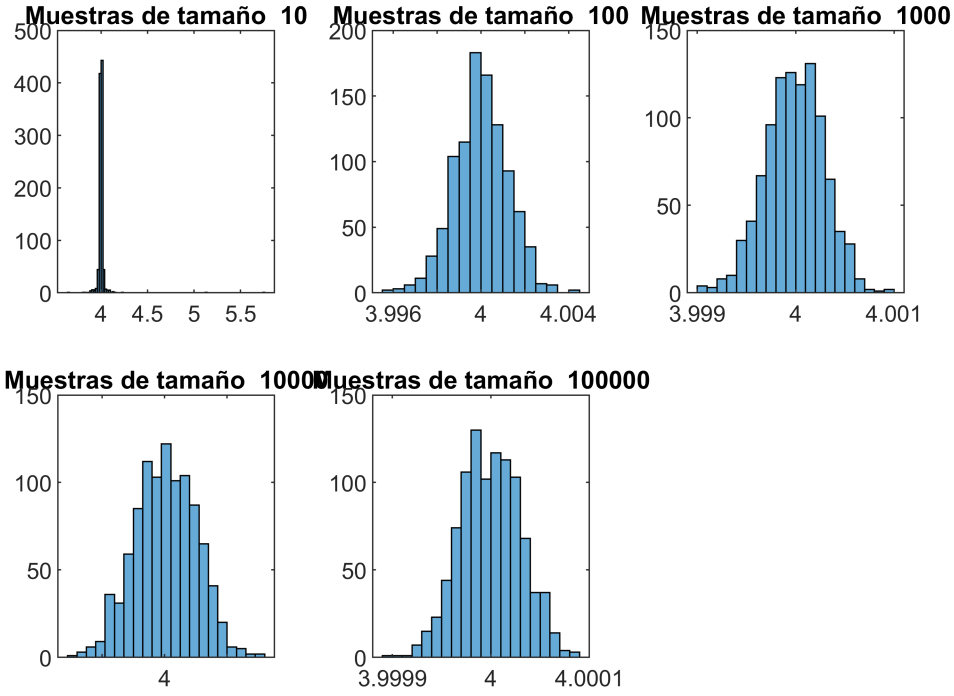
Distribución β_1



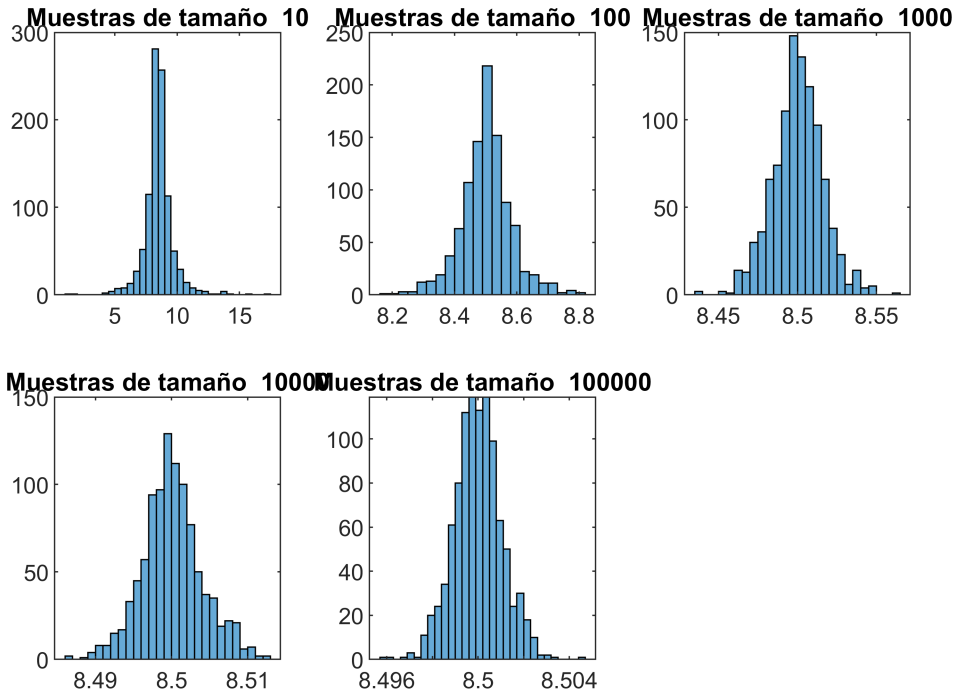
Distribución β_2



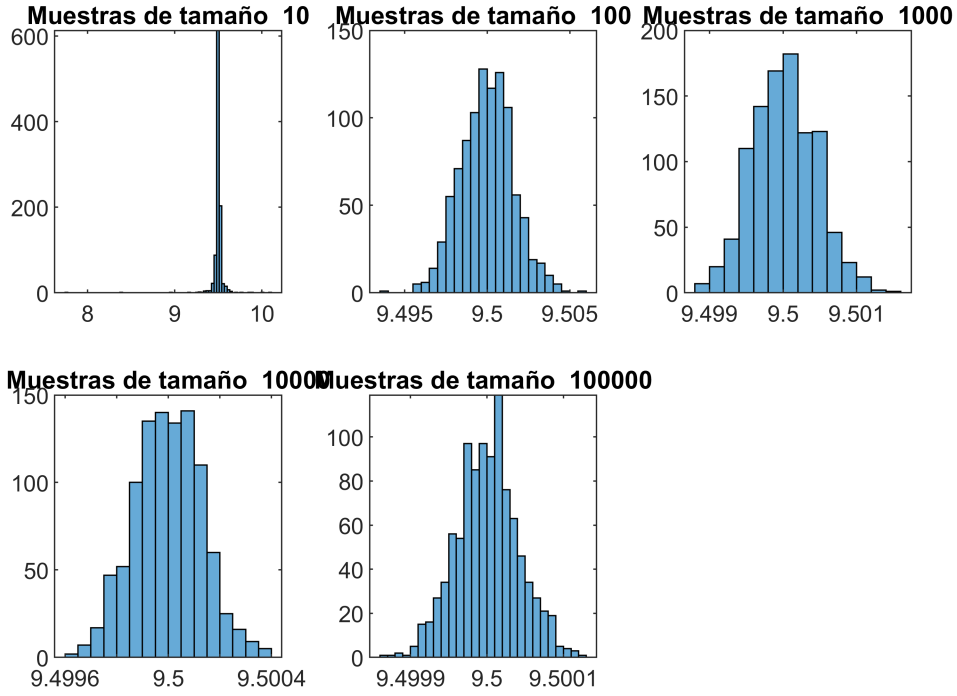
Distribución β_3



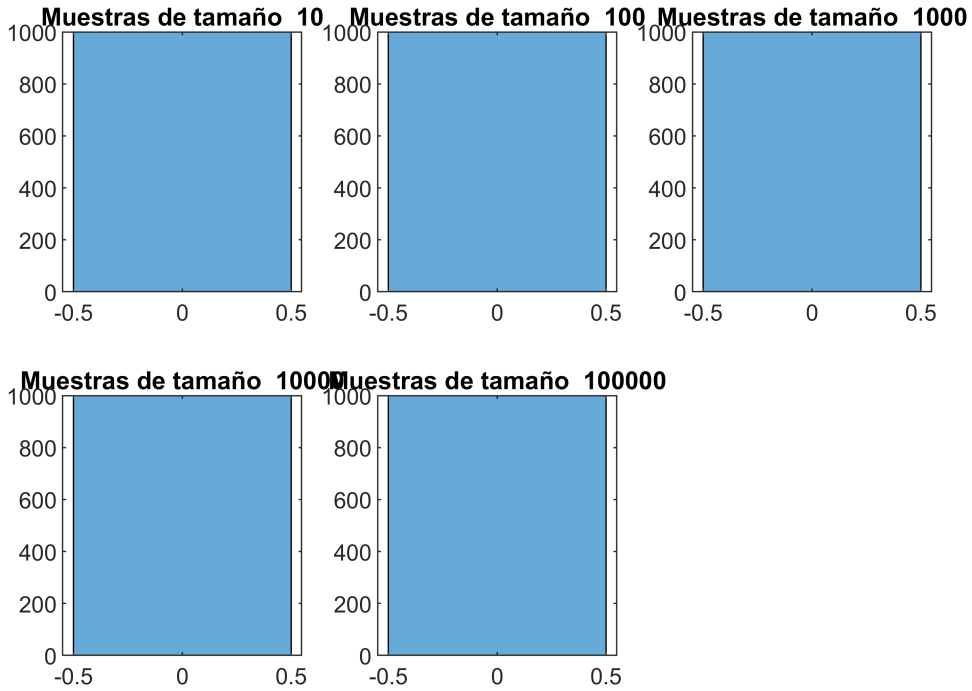
Distribución β_4



Distribución β_5



Distribución β_6




```
ans = struct with fields:
        ols_10: [1000×7 double]
       ols_100: [1000×7 double]
      ols_1000: [1000×7 double]
     ols_10000: [1000×7 double]
    ols_100000: [1000×7 double]
```

¿En torno a que valores están centradas las distribuciones? ¿Porqué? Explique las diferencias respecto al inciso anterior

A diferencia del inciso anterior, las distribuciones no están centradas en el cero, sino que sus estimaciones están desviadas positivamente. La razón es que ahora, para calcular los coeficientes de regresión no utilizamos directamente los valores de la matriz de vectores de variables aleatorias, y su relación con el **y.**
Si no que partimos prediciendo el **y** con una perturbación normal ($\epsilon \sim N(0,1)$) y en donde, entonces tendremoa que nuestra variable real estará estimada con un error. Formalmente tenemos que :

$\epsilon = y - y^*$ donde y^* es la variable real

Luego tenemos que si bien el modelo original es

$$y^* = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_3 + \beta_4 \cdot x_4 + \beta_5 \cdot x_5 + \beta_5 \cdot x_5 + \beta_6 \cdot x_6 + u \qquad (1)$$

Nosotros estimamos un

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_3 + \beta_4 \cdot x_4 + \beta_5 \cdot x_5 + \beta_5 \cdot x_5 + \beta_6 \cdot x_6 + u + \epsilon \qquad (1)$$

Esto implica que tendremos ahora un error compuesto: uno conformado por el "error" que teóricamente tiene el modelo estimado y otro que le agregamos a la estimación.

Notemos un punto importante y es que los estimadores no dejan de ser consistentes dado que el valor esperado del error sigue siendo cero. El problema más bien estará en la varianza condicional del error en relación al X, y que esto se podrá notar claramente en las gráficas. Matemáticamente

$$\text{Sea } v = \epsilon + u$$
$$\text{Var}(v|X) = \text{Var}\,(u + \epsilon\,|X) = \text{Var}(u|X) + \text{Var}(\epsilon\,|X) + 2\text{Cov}(u, \epsilon\,|X)$$

Si decimos que existe una relación entre los errores y el error de medición tendremos que $\text{Cov}(u, \epsilon\,|X) \neq 0$

Si es así, los $\widehat{\beta}$ serán sesgados. Si $\text{Cov}(u, \epsilon\,|X) > 0$ el sesgo de $\widehat{\beta}$ será positivo . Eso implica que los estimadores están desviados respecto del valor real. Además de ello, los estimadores serán más ineficientes. Este resultado es esperable pues construimos sumamos una perturbación que sigue la misma distribución que el error "teórico" del modelo.

Ahora bien notemos dos cosas relevantes:

1. La primera es que la forma de y no va a cambiar. Sigue teniendo una distribución logarítimica, pero desviada en su valor real.
2. Al igual que en el caso del predictor, los estimadores **no cambiarán** su forma de distribución que se asimila a una normal. Las razones son dos. La primera tiene que ver con una propiedad lineal de la normal. Sumar normales da como resultado una normal. La segunda tiene un fuerte respaldo en el ejercicio que hemos hecho en este apartado y es que, por los proceso iterativos con grandes tamaños muestrales y de muestras repetidas nos permiten **asintóticas** decir que pese a las perturbaciones, en los límites se seguirían cumpliendo las propiedades de los estimadores (propiedades asintóticas). No así las de insesgamiento.

Adicional: Comprobando resultados

Adicionalmente, hemos reafirmado este punto mostrando el resultado de la estimación de los coeficientes de regresión. Notemos que su valor esperado está cercano a cero en la mayoría de estos, a excepción del intercepto pero que no tiene una gran confiabilidad si miramos su error estándar. Es decir, con un 95% de confianza podemos decir que los coeficientes no tienen un efecto sustantivo sobre **y** dado que su valor es cero. Y como pudimos ver en el punto h, si agregamos una perturbación, podríamos llegar a resultados equivocados y sesgados respecto de la estimación.

```
regresion = fitlm(X,y)
```

```
regresion =
Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6 + x7

Estimated Coefficients:
                   Estimate        SE         tStat        pValue
                  __________    ________    ________    __________

    (Intercept)        2.1158      0.6147      3.4421    0.00087036
    x1                      0           0         NaN           NaN
    x2                -0.029526     0.11459    -0.25768       0.79723
```

```
x3              -0.043848     0.086832    -0.50497       0.61478
x4              -0.0029352    0.002554    -1.1492        0.25343
x5                0.23899      0.20601     1.1601        0.24901
x6              0.0044493    0.0035467     1.2545        0.21284
x7                      0            0        NaN            NaN


Number of observations: 100, Error degrees of freedom: 94
Root Mean Squared Error: 1.8
R-squared: 0.0362,  Adjusted R-Squared: -0.0151
F-statistic vs. constant model: 0.706, p-value = 0.62
```

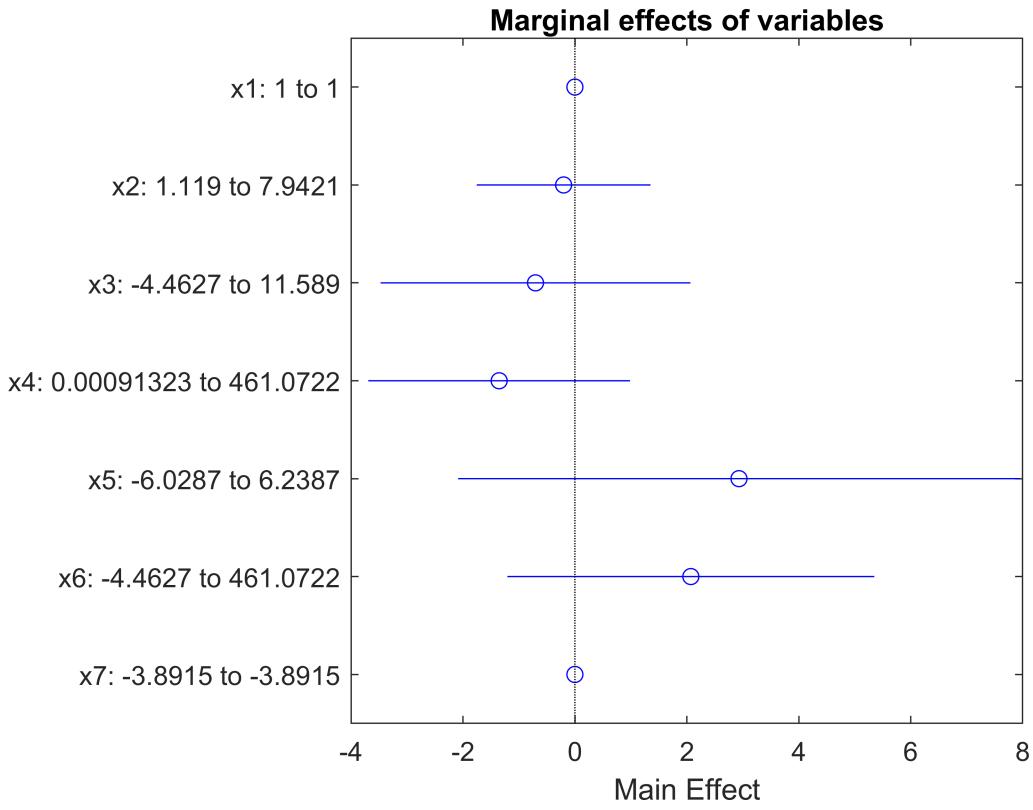
```
regresion.Coefficients
```

ans = 8×4 table

	Estimate	SE	tStat	pValue
1 (Intercept)	2.1158	0.6147	3.4421	0.0009
2 x1	0	0	NaN	NaN
3 x2	-0.0295	0.1146	-0.2577	0.7972
4 x3	-0.0438	0.0868	-0.5050	0.6148
5 x4	-0.0029	0.0026	-1.1492	0.2534
6 x5	0.2390	0.2060	1.1601	0.2490
7 x6	0.0044	0.0035	1.2545	0.2128
8 x7	0	0	NaN	NaN

Presentamos una figura final para mostrar esta ausencia de efecto.

```
figure(30)
plotEffects(regresion)
title("Marginal effects of variables")
hold off
```



Referencias

Greene (2016) Econometric Analysis. Pearson

Russell, K. G. (febrero de 1991). «Estimating the Value of e by Simulation». *The American Statistician* **45** (1): 66-68.

Learning Matlab using OLS