UNTREF UNIVERSIDAD NACIONAL DE TRES DE FEBRERO

Ingeniería en Computación

Probabilidad y Estadística

Trabajo Práctico
"Simulación de variables aleatorias"

Alumnos:

Araneda, Alejandro – eloscurodeefeso@hotmail.com
Quinteros, Fernando - lordfers@gmail.com
Speciale, Gastón - gasticai@hotmail.com

Práctica entregada:

2do. Cuatrimestre 2020
Jueves, 3 de Diciembre

Docentes:

Dr. Ing. Néstor Rubén Barraza
Dr. Lic. Verónica Moreno
Ing. Gabriel Pena

Resumen

En el presente trabajo tiene el objetivo de servir de breve introducción a los métodos de generación de muestras de variables aleatorias que siguen determinada distribución de probabilidad.

Introducción

La inferencia de característica de sistemas complejos, así como su modelado a través de técnicas numéricas, se ve beneficiado si su verisimilitud es puesta a prueba mediante estímulos de naturaleza aleatoria[1]. Siendo muchas veces necesario conducir ese azar dentro del marco conceptual de los estudios de probabilidad y la estadística. De allí la búsqueda de métodos generadores de variables aleatorias con cierta distribución.

Uno de los métodos es la genaración da variables aleatorias por simulación directa. Este es caso para los ensayos Bernoulli, cuyos resultados pueden ser sólo éxito o fracaso con cierta probabilidad. Sea U una variable aleatoria con distribución uniforme en el intervalo $[0,1]$, entonces la simulación directa de un ensayo Bernoulli con p probabilidad de éxito viene dada por la función:

$$Ber(p) = \begin{cases} 1 & U <= p \\ 0 & \text{en otro caso} \end{cases}$$

Con el mismo criterio podemos tambié generar una muestra de variables aleatorias que respeten una distribución binomial, define por la cantidad de éxitos de n ensayos Bernoulli con p probabilidad de éxito, mediante una suma directa.

Otro método es el uso de la inversa de la función densidad de probabilidad acumulada. Es posible probar que si $F(x)$ es la función densidad de probabilidad acumulada para algún valor x de una variable aleatoria X y U es una variable aleatoria uniformemente distribuida en el entorno semiabierto $[0,1)$, entonces la función inversa $F^{-1}(U)$ nos permitirá obtener una muestra de valores de X que siguen su misma distribución de probabilidad.

Así para la distribución exponencial con parámetro λ igual a uno, su función densidad de probabilidad acumulada y su inversa son:

$$F(x) = 1 - e^{-x} \text{ con } x \geq 0$$
$$F^{-1}(u) = -ln(1-u) = x$$

En el caso de la distribución normal, el cálculo analítico de su inversa se ve imposibilitado. Es por ello que se emplea comúnmente un método que aproxima su inversa denominado Box-Müller por los nombres de sus desarrolladores. La técnica parte de suponer que U_1 y U_2 son dos variables aleatorias independientes que están uniformemente distribuidas en el intervalo semiabierto $(0,1]$. Entonces Z_0 y Z_1 son variables aleatorias independentes con una distribución normal con desviación estandar igual a 1 y sus fórmulas son:

$$Z_0 = Rcos(\theta) = \sqrt{-2ln(U_1)}cos(2\pi U_2)$$
$$Z_1 = Rsin(\theta) = \sqrt{-2ln(U_1)}sin(2\pi U_2)$$

Descripción de la práctica

Enunciado

Parte 1: Simulación

En esta primera parte, construiremos varios generadores de números aleatorios que usaremos para obtener muestras con distribu- ción conocida sobre las que vamos a trabajar posteriormente.

1. Utilizando únicamente la función random de su lenguaje (la función que genera un número aleatorio uniforme entre 0 y 1), implemente una función que genere un número distribuido Bernoulli con probabilidad p.
2. Utilizando la función del punto anterior, implemente otra que genere un número binomial con los parámetros n,p.
3. Utilizando el procedimiento descrito en el capítulo 6 del Dekking (método de la función inversa o de Monte Carlo), implementar una función que permita generar un número aleatorio con distribución Exp(λ).
4. Investigar como generar números aleatorios con distribución normal. Implementarlo.

Parte 2: Estadística descriptiva

Ahora vamos a aplicar las técnicas vistas en la materia al estudio de algunas muestras de datos.

1. Generar tres muestras de números aleatorios Exp(0,5) de tamaño n = 10, n = 30 y n = 200. Para cada una, computar la media y varianza muestral. ¿Qué observa?
2. Para las tres muestras anteriores, graficar los histogramas de frecuencias relativas con anchos de banda 0,4, 0,2 y 0,1; es decir, un total de 9 histogramas. ¿Qué conclusiones puede obtener?
3. Generar una muestra de números Bin(10,0,3) de tamaño n = 50. Construir la función de distribución empírica de dicha muestra.
4. A partir de la función de distribución empírica del punto anterior, generar una nueva muestra de números aleatorios utilizando el método de simulación de la primera parte. Computar la media y varianza muestral y graficar el histograma.
5. Repetir el experimento de los dos puntos anteriores con dos muestras aleatorias más generadas con los mismos parámetros. ¿Qué conclusión saca?

Parte 3: Convergencia

El propósito de esta sección es ver en forma práctica los resultados de los teoremas de convergencia.

1. Generar cuatro muestras de números aleatorios de tamaño 100, todas con distribución binomial con p = 0,40 y n = 10, n = 20, n = 50 y n = 100 respectivamente. Graficar sus histogramas. ¿Qué observa?
2. Elija la muestra de tamaño 200 y calcule la media y desviación estándar muestral. Luego, normalice cada dato de la muestra y grafique el histograma de la muestra normalizada. Justifique lo que observa.
3. Para cada una de las muestras anteriores, calcule la media muestral. Justifique lo que observa.

Parte 4: Estadística inferencial

Para terminar, vamos a hacer inferencia con las muestras que generamos y obtener así información sobre sus distribuciones.

1. Generar dos muestras N(100,5), una de tamaño n = 10 y otra de tamaño n = 30. Obtener estimaciones puntuales de su media y varianza.
2. Suponga que ya conoce el dato de que la distribución tiene varianza 5. Obtener intervalos de confianza del 95
3. Repita el punto anterior pero usando la varianza estimada s2, para la muestra de tamaño adecuado.
4. Probar a nivel 0,99 la hipótesis de que la varianza sea $\sigma^2 > 5$. Calcular la probabilidad de cometer error tipo II para la hipótesis alternativa $\sigma^2 = 6$.
5. Agrupando los datos en subgrupos de longitud 0,5, probar a nivel 0,99 la hipótesis de que la muestra proviene de una distribución normal.

Entorno de Desarrollo

Siguiendo las indicaciones de la cátedra, implementamos los generadores de muestras aleatorias con el lenguaje Python. Nuestro desarrollo tiene dependencia de librerías del proyecto *SciPy*: los ejercicios son resueltos en el entorno interactivo *Jupyter Notebook*[3], las gráficas generadas con *Matplotlib*[4] y las inversas de las tablas de la normal estandariza, la distribució T de Studente y la χ^2 son tomadas de la librería *Scipy*[5].

Para documentar el código fuente nos adherimos a las convenciones del lenguaje utilizando *Docstring*[9]. De esta forma, su presentación la automatizamos mediante el paquete provisto por el proyecto *Sphinx*[6]. Para la presetación final del trabajo, las *notebooks* se unifican mediante el paquete *nbmerge*[7] he integradas manualmente a este informe para ajustes estéticos.

Las fuciones principales de cada módulo y que nos sirven en la resolución de los ejercicios propuestos, han sido verificadas en su funcionamiento previamente mediante el uso de la librería *Unittest*[2].

La organización del aspecto colaborativo entre los participantes de la práctica es resuelto mediante la utilización de un repositorio de cotrol de versiones alojado en el sitio *GitHub*[8].

Parte 1: Simulación

Todo el código desarrollado se encuentra profusamente documentado en un manual anexo. Sin embargo, utilizaremos la presente sección para describir el contenido de los archivos de la práctica que son entregados juntamente con este informe.

El uso del entorno interactivo correspondiente al proyecto *Jupiter Notebook*, nos decidió por una arquitectura simple con módulos de funciones estáticas separados por incumbencias.

Archivo	Descripción
analisis.py	Módulo python la implementación de las funciones que calculan las medias y varianzas muestrales
build\mytitle.sty	Paquete de latex con la carátula del manual
build\manual.pdf	El manual con la documentación autogenerada de todo el código
graficas.py	Módulo python auxiliar para la generación de gráficas y tablas de la práctica
informe.tex	Fuente latex del presente informe
informe.pdf	Este informe
logo.png	Logo de la Universidad Nacional de Tres de Febrero
parte2.ipynb	Parte 2 de la práctica desarrollada en una Jupyter Notebook
parte3.ipynb	Parte 3 de la práctica desarrollada en una Jupyter Notebook
parte4.ipynb	Parte 4 de la práctica desarrollada en una Jupyter Notebook
simulador.py	Módulo python con la implemetación de los generadores de muestras aleatorias
source\conf.py	Archivo de configuración para extracción de la documentación del código fuente
source\index.rst	Archivo de configuración para la extracción de la documentación del código fuente

Parte 2: Estadística descriptiva

Primeramente es necesario importar las funciones auxiliares que hemos implementado para simular las muestras de variables aleatorias:

[1]:
```
import sys; sys.path.insert(0, '..')
from simulador import *
```

Luego obtenemos tres muestras de números aleatorios con distribución exponencial y parámetro lambda (la inversa de la media) igual a 0,5 pero de diferentes tamaños: 10, 30 y 200 datos cada una. Las llamaremos acordemente `exp10`, `exp30` y `exp200` respectivamente:

[2]:
```
exp10 = exponencial(10, 2)
exp30 = exponencial(30, 2)
exp200 = exponencial(200, 2)
```

Ahora importaremos las funciones `media_muestral` y `varianza_muestral` del módulo `analisis` para completar los datos de la próxima tabla.

[3]:
```
from analisis import media_muestral as mm
from analisis import varianza_muestral as vm
```

[4]:
```
from IPython.display import display, Markdown
display(Markdown('''
| Tamaño | Media | Varianza |
|--------|-------|----------|
```

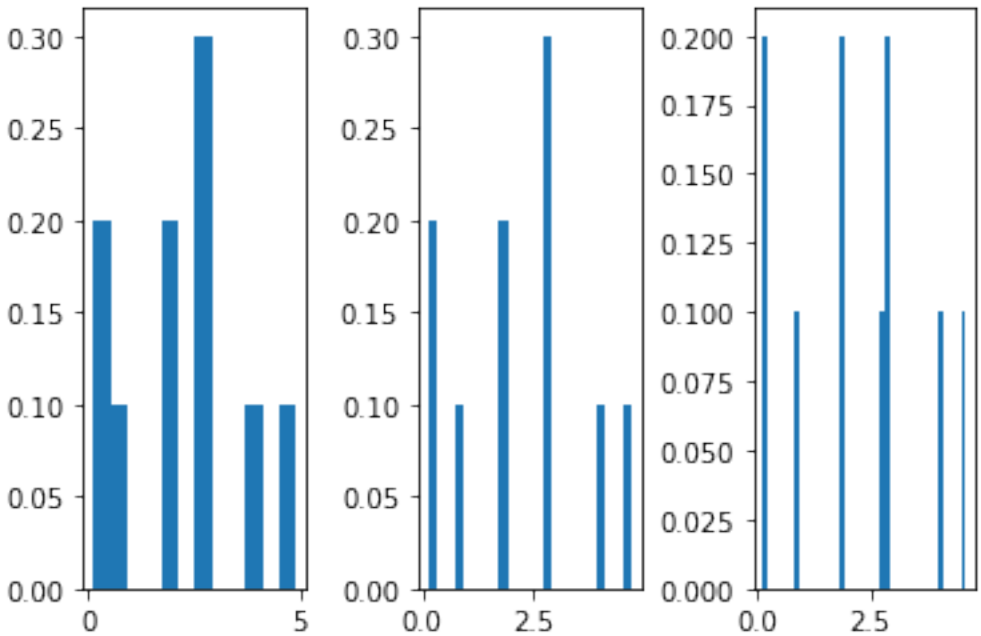
```
| %d      | %.2f   | %.2f      |
| %d      | %.2f   | %.2f      |
| %d      | %.2f   | %.2f      |
''' % (10, mm(exp10), vm(exp10),
       30, mm(exp30), vm(exp30),
       200, mm(exp200), vm(exp200))))
```

Tamaño	Media	Varianza
10	2.20	2.32
30	2.84	9.56
200	1.80	3.03

Lo que podemos concluir al observar la repetición de la experiencia es que si bien como es esperable los parámetros de la muestra se aproximan a los de la distribución cuando el tamaño de la muestra crece, la media muestral lo hace con respecto a la media de la distribución con mayor estabilidad que la varianza muestral con respecto a la varianza de la distribución.

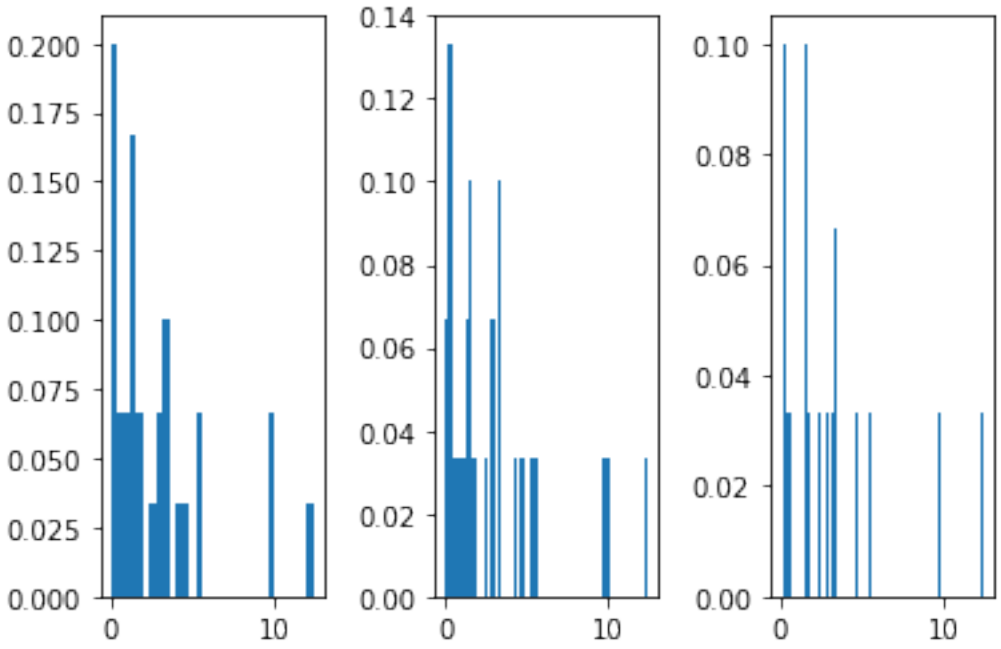
Observaremos a continuación los histogramas de las muestras con diferente anchos de banda, para lo cual importamos una función auxiliar que hace uso de la librería gráfica elegida. Presentamos a continuación los histogramas con anchos de banda de 0,4 unidades (izquierda), de 0,2 unidades (centro) y de 0,1 unidades (derecha) para nuestra muestra de 10 datos.

[5]:
```
%matplotlib inline
from graficas import histograma
histograma(exp10, [0.4,0.2,0.1], relative=True)
```



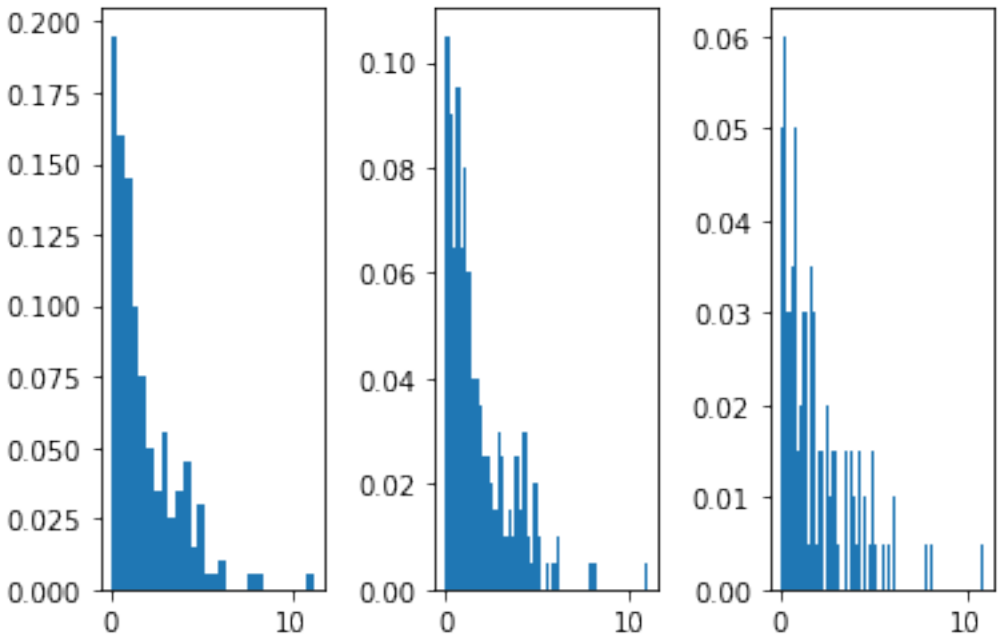
Reiteramos el procedimiento con la muestra de 30 datos:

[6]:
```
histograma(exp30, [0.4,0.2,0.1], relative=True)
```



Y finalmente, obtenemos los histogramas para nuestra muestra de 200 variables aleatorias en base a una distribución exponencial con los mismos parámetros:

[7]:
```
histograma(exp200, [0.4,0.2,0.1], relative=True)
```

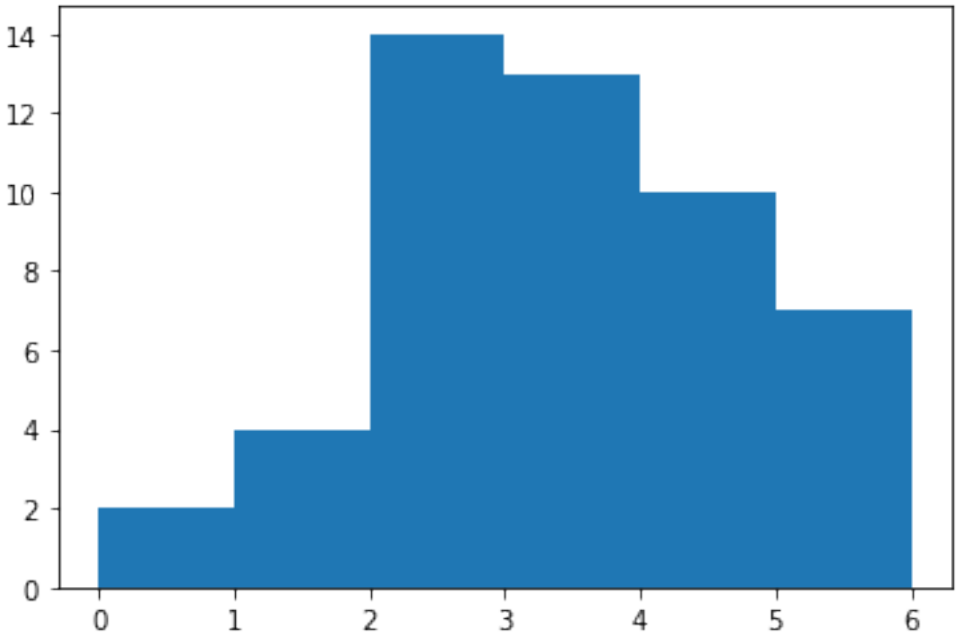


El análisis de los histogramas nos permite verificar que nuestras muestras efectivamente se aproximan a la distribución exponencial cuando mayor es el tamaño de la muestra. También podemos notar que no todos los anchos de banda nos brindan la misma información: para una muestra de tamaño discreto, un ancho de banda muy pequeño podría ubicar quizá sólo una aparición por clase, de manera que las características de nuestra distribución se hallen oscurecidas.

Lo siguiente que haremos será crear una muestra de 50 números con base en una distribución binomial con parámetros correspondientes a 10 experimentos Bernoulli y una probabilidad de éxito de 0,3. También graficaremos su histograma como referencia a la distribución empírica que producirá e informaremos sus estadísticos con la función auxiliar `estadist` importada del módulo `analisis.py`.

[8]:
```
from analisis import estadist
bin50 = binomial(50, 10, 0.3)
estadist(bin50)
histograma(bin50, [1])
```

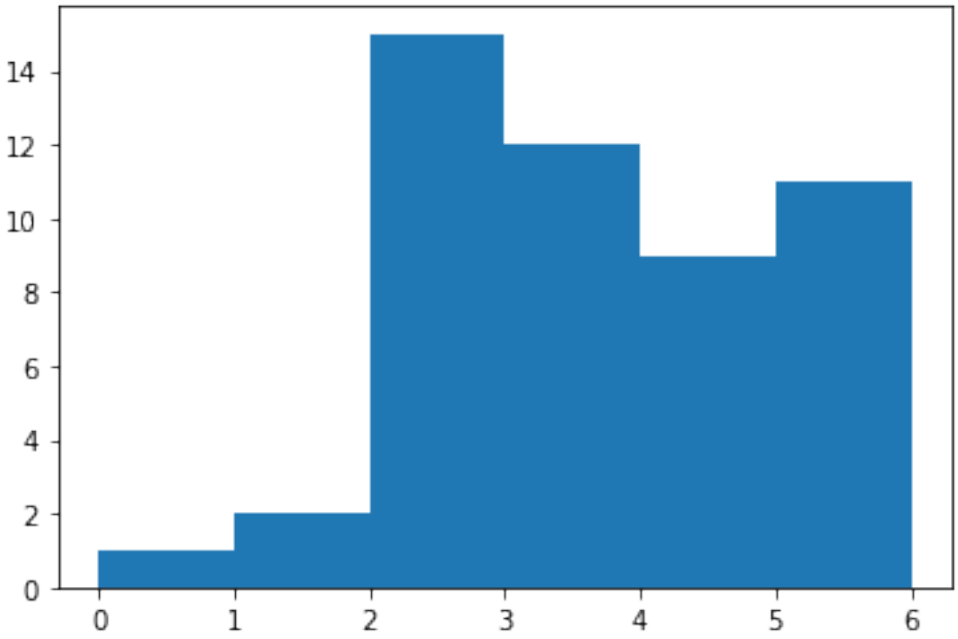
```
Media muestral: 2.94 - Varianza muestral: 1.85
```



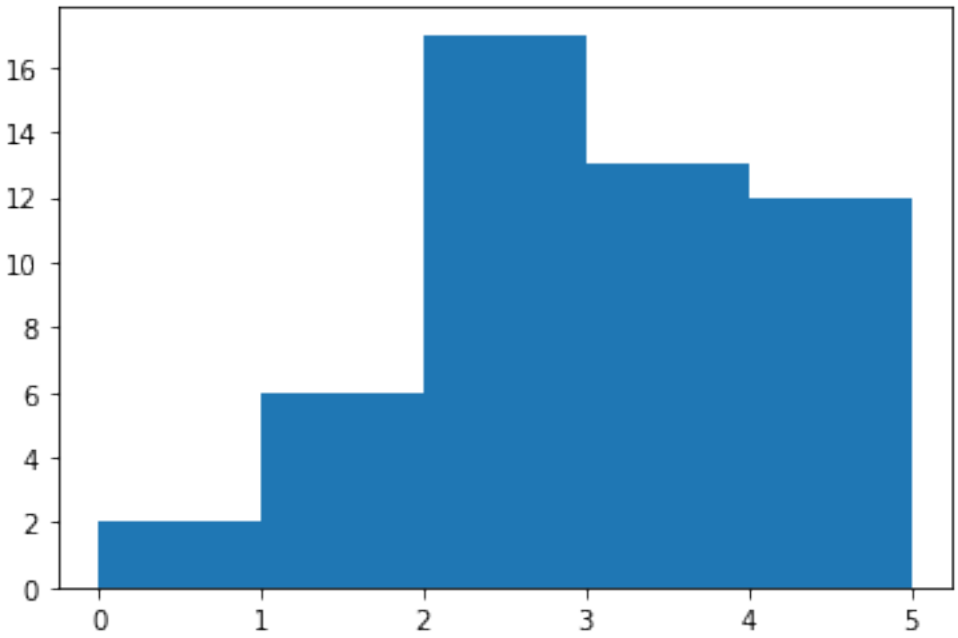
Ahora simulamos una segunda muestra de 50 dotas que tendrán por base la distribución empírica que resulta de la muestra anterior. Calculamos también su media y varianza muestral además de graficar el histograma.

[9]:
```
bino = binomial(50, 10, 0.3)
emp = empirica(50, bino)
print("Muestra binomial - ", end="")
estadist(bino)
histograma(bino, [1])
print("Muestra empírica - ", end="")
estadist(emp)
histograma(emp, [1])
```

```
Media muestral: 3.22 - Varianza muestral: 1.89
```



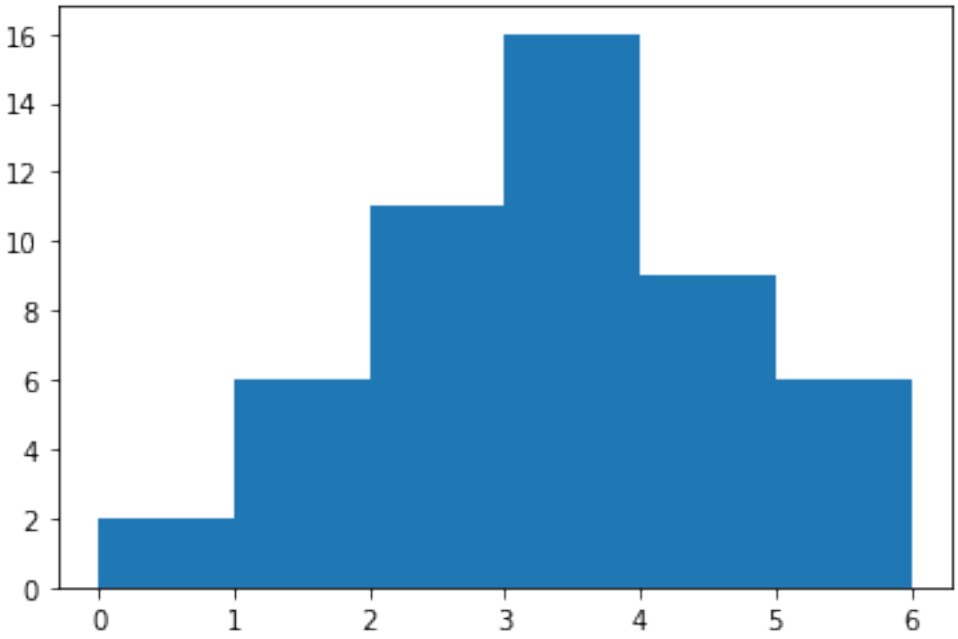

```
Muestra empírica - Media muestral: 2.60 - Varianza muestral: 1.47
```



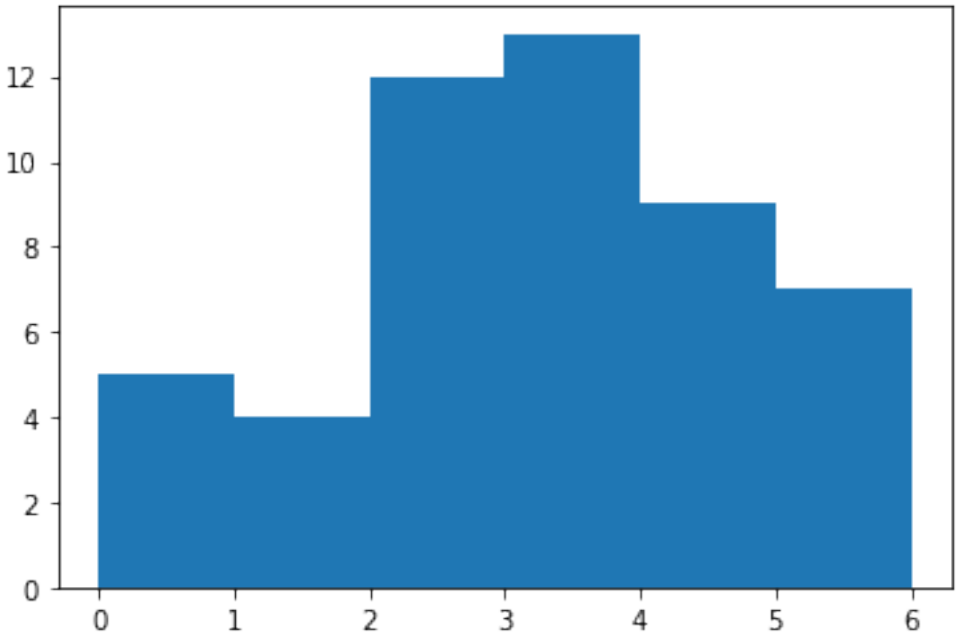
[11]:
```
bino = binomial(50, 10, 0.3)
emp = empirica(50, bino)
```

```
print("Muestra binomial - ", end="")
estadist(bino)
histograma(bino, [1])
print("Muestra empírica - ", end="")
estadist(emp)
histograma(emp, [1])
```

```
Muestra binomial - Media muestral: 2.86 - Varianza muestral: 1.84
```




```
Muestra empírica - Media muestral: 2.80 - Varianza muestral: 2.41
```



Como conclusión podemos decir que si bien las muestras binomiales siguen una implementación que tiende a aproximar sus estadísticos a los que se esperarían en base a los parámetros de la distribución binomial de base, lo hace de una manera que a simple observación parece más azarosa que la que resulta de obtener nuevas muestras utilizándolas de distribución empírica.

Parte 3: Convergencia

Al igual que en la parte 2, es necesario importar las funciones auxiliares que hemos implementado para simular las muestras de variables aleatorias:

[1]:
```
import sys; sys.path.insert(0, '..')
from simulador import *
```

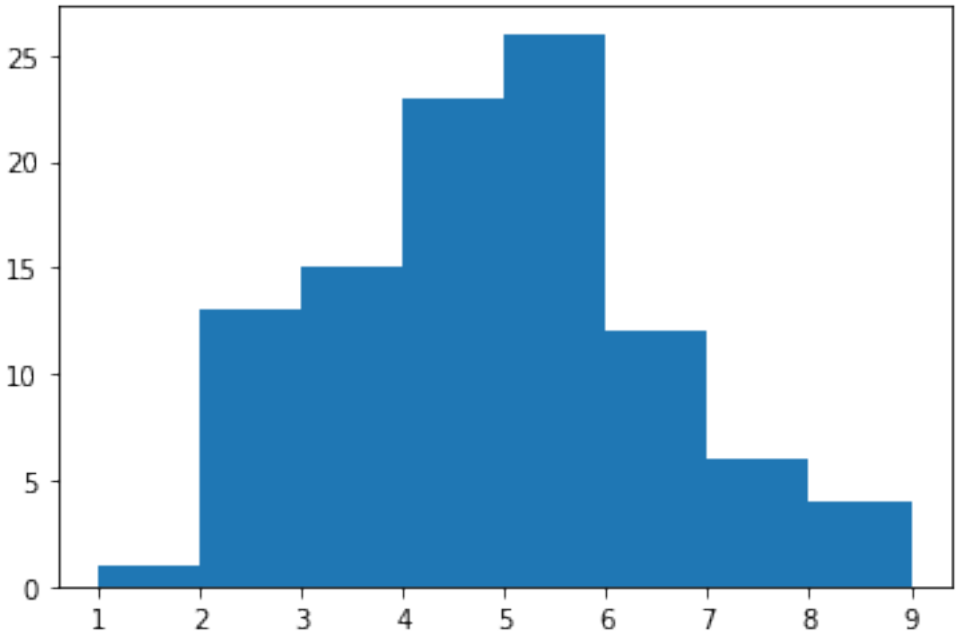
Luego obtenemos cuatro muestras con distribucion binomial de tamaño 100 con probabilidad p igual a 0.4, pero cada una con 10, 20, 50 y 100 experimentos Bernoulli, que se denotaran como `bin10`, `bin20`, `bin50` y `bin100` respectivamente:

[2]:
```
bin10 = binomial(100, 10, 0.4)
bin20 = binomial(100, 20, 0.4)
bin50 = binomial(100, 50, 0.4)
bin100 = binomial(100, 100, 0.4)
```

Ahora, procedemos a graficar los histogramas correspondientes a cada muestra de datos obtenida.

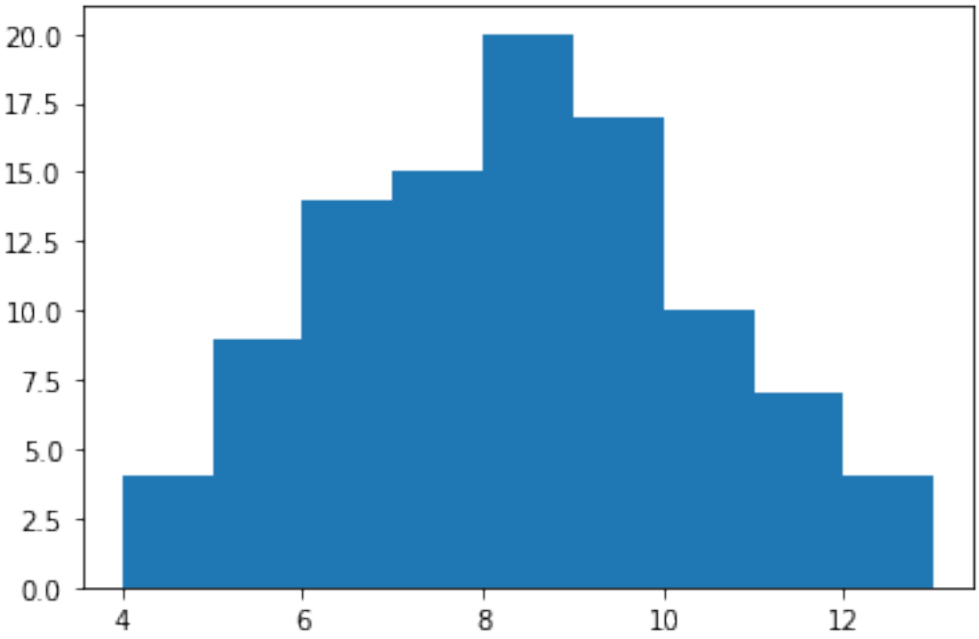
Para una muestra binomial con 10 experimentos Bernoulli, probabilidad 0.4 y con 100 repeticiones, el histograma es el siguiente:

[3]:
```
%matplotlib inline
from graficas import histograma
histograma(bin10, [1])
```



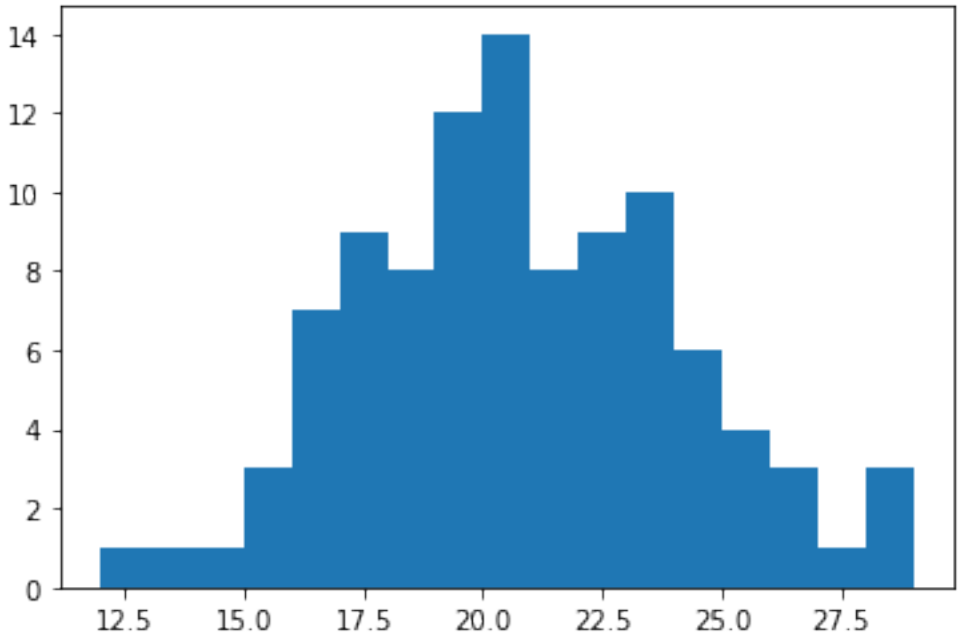
Para una muestra binomial con 20 experimentos Bernoulli, probabilidad 0.4 y con 100 repeticiones, el histograma es el siguiente:

[4]:
```
histograma(bin20, [1])
```



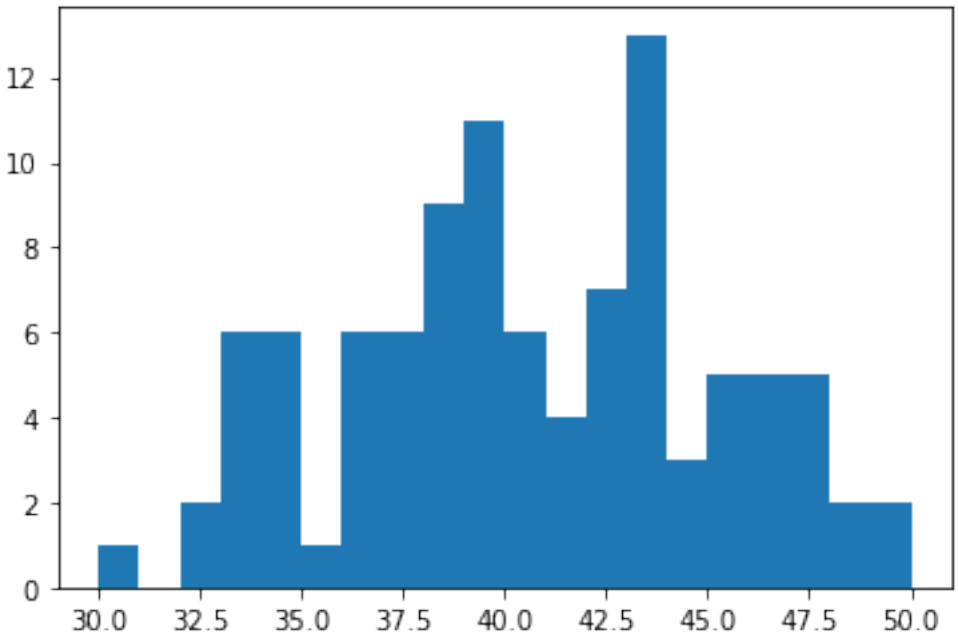
Para una muestra binomial con 50 experimentos Bernoulli, probabilidad 0.4 y con 100 repeticiones, el histograma es el siguiente:

[5]:
```
histograma(bin50, [1])
```



Para una muestra binomial con 100 experimentos Bernoulli, probabilidad 0.4 y con 100 repeticiones, el histograma es el siguiente:

[6]:
```
histograma(bin100, [1])
```

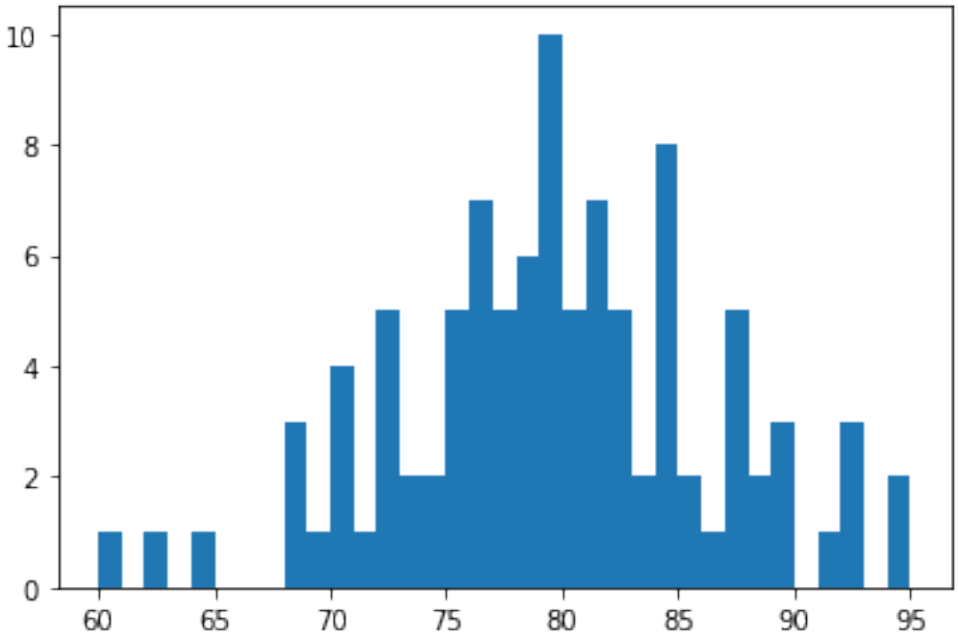


Lo que se puede observar es que a medida de que la cantidad de experimentos de la muestra crece, es decir n crece, la muestra empieza a tener la forma de una campana de Gauss y que se aproxima a una muestra de distribucion normal. Si se observa desde el primer histograma cuyo n es 10, hasta el ultimo cuyo n es 100, la forma de campana se hace mas notoria.

Ahora procedemos a graficar una muestra de tamaño 100 con probabilidad de éxito 0.4 para 200 experimentos Bernoulli. Calcularemos también su media y desviacion estandar muestrales.

[7]:
```
from analisis import media_muestral as mm
from analisis import desviacion_estandar_muestral as dem
bin200 = binomial(100, 200, 0.4)
bin200_mm = mm(bin200)
bin200_dem = dem(bin200)
print("Media muestral: %.2f | Desviacion estandar muestral: %.2f" % (bin200_mm,␣
 ↪bin200_dem))
histograma(bin200, [1])
```

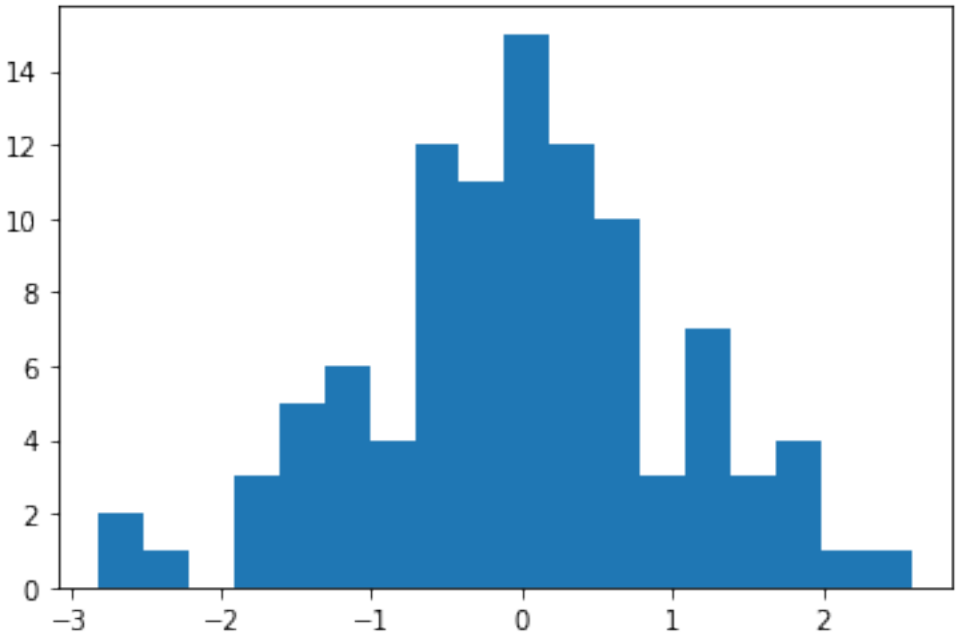
```
Media muestral: 79.19 | Desviacion estandar muestral: 6.82
```



La forma de la campana caracteristica de una muestra con distribucion normal es mas marcada. Las 100 repeticiones del experimento hacen que haya resultados que se puedan promediar con el fin de obtener mayor precision en el mismo. Es asi que lo que se puede observar es que el promedio de todas estas repeticiones con probabilidad p (en este caso se tomo un p de valor 0.4) tienden o convergen al valor de la media (cuyo valor teorico es de 80 para esta muestra en particular).

Luego, con la media y desvío estandard muestrales estandarizaremos cada uno de los datos de la muestra y graficamos el histograma de la muestra estandarizada obtenida:

[8]:
```
bin200_normalizada = [ (x - bin200_mm)/bin200_dem for x in bin200]
histograma(bin200_normalizada, [0.3])
```



Por ultimo, computamos las medias muestrales para cada una de las muestras anteriores y las escribimos en la siguiente tabla junto con su error relativo con respecto a la media de la distribución binomial de base:

[9]:
```
from IPython.display import display, Markdown
display(Markdown('''
| Tamaño |  Media Muestral | Media Binomial | Err. Rel |
|--------|-----------------|----------------|----------|
| %d     |  %.2f           | %d             | %.3f     |
| %d     |  %.2f           | %d             | %.3f     |
| %d     |  %.2f           | %d             | %.3f     |
| %d     |  %.2f           | %d             | %.3f     |
| %d     |  %.2f           | %d             | %.3f     |  ''' % ( 10, mm(bin10),␣
↪  4, abs(mm(bin10)/4 - 1),
       20, mm(bin20),   8, abs(mm(bin20)/8 - 1),
       50, mm(bin50),  20, abs(mm(bin50)/20 - 1),
      100, mm(bin100), 40, abs(mm(bin100)/40 - 1),
      200, mm(bin200), 80, abs(mm(bin200)/80 - 1))))
```

Tamaño	Media Muestral	Media Binomial	Err. Rel
10	4.43	4	0.107
20	7.90	8	0.012
50	20.31	20	0.015
100	40.14	40	0.004
200	79.19	80	0.010

Repitiendo la experiencia muchas veces podemos observar que, si bien no de manera exacta dada la naturaleza probabilística de nuestro experimento, al aumentar la cantidad de experimentos Bernoulli de cada muestra y mantener el parámetros de probabilidad de éxito, el error relativo tiende a disminuir.

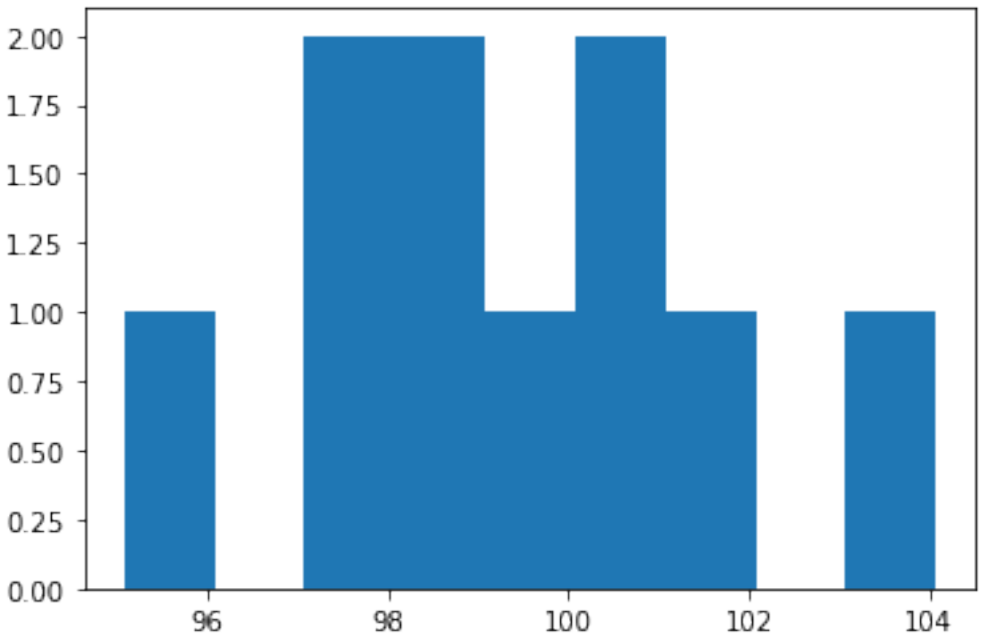
Parte 4: Estadística inferencial

Tiene como objetivo hacer inferencias sobre muestras generadas, para asi obtener informacion sobre sus correspondientes distribuciones. Se importar las funciones auxiliares que hemos implementado para simular las muestras de variables aleatorias:

[136]:
```
import sys; sys.path.insert(0, '..')
from simulador import *
from math import *
```

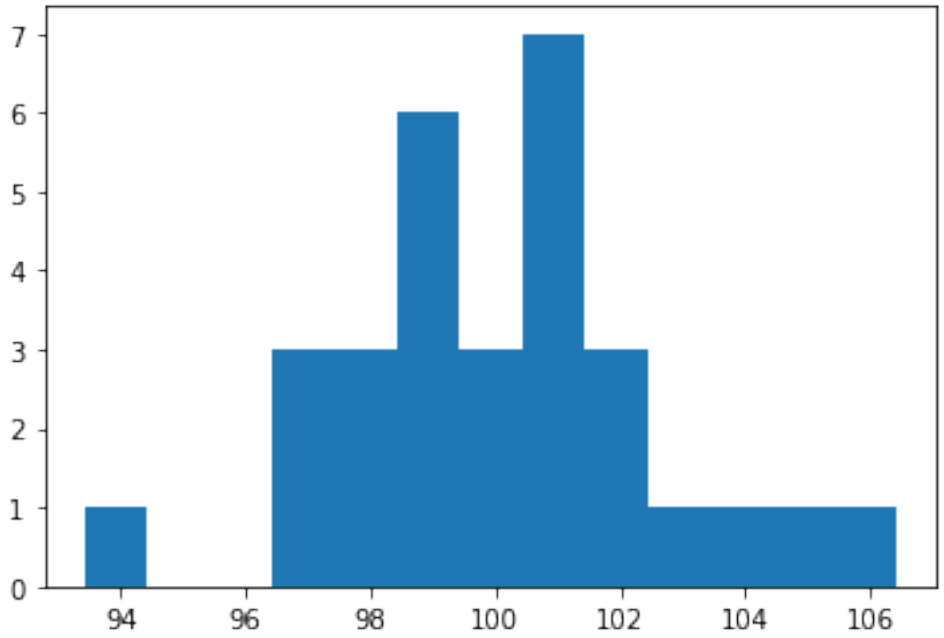
A continuacion, se generaran dos muestras con distribucion normal con media igual a 100 y varianza igual a 5, pero una con 10 repeticiones y la otra con 30 repeticiones. La primera muestra con n = 10 esta representada en el siguiente histograma:

[137]:
```
%matplotlib inline
from graficas import histograma
nor10 = normal(10, 100, sqrt(5))
histograma(nor10, [1])
```



La segunda muestra con n = 30 esta representada en el siguiente histograma:

[138]:
```
nor30 = normal(30, 100, sqrt(5))
histograma(nor30, [1])
```



Para las muestra con n = 10 y n = 30 respectivamente, se obtienen las estimaciones puntuales de sus medias y de sus varianzas.

Como se vio a lo largo del curso la media muestral (o $\bar{x}$) es un estimador insesgado de la media por lo que es el estimador que utilizaremos.

La varianza muestral(o s^2) es un estimador insesgado de la varianza por lo que es el estimador que utilizaremos. Ya que el estimador debe recibir como input los datos de las muestras, tales estimaciones estan representadas en la siguiente tabla :

[139]:
```
from IPython.display import display, Markdown
from analisis import media_muestral as mm
from analisis import varianza_muestral as vm
display(Markdown('''
| Tamaño    | Media estimada | Varianza estimada|
|-----------|--------------- |------------------|
| %d        | %.2f           | %.2f             |
| %d        | %.2f           | %.2f             |
''' % (10, mm(nor10), vm(nor10),
       30, mm(nor30), vm(nor30))))
```

Tamaño	Media estimada	Varianza estimada
10	99.36	5.41

Tamaño	Media estimada	Varianza estimada
30	100.01	6.32

Ahora, se pide que dado que se conoce el dato de que la distribucion tiene varianza 5, se obtengan intervalos de confianza del 95% y del 98% para la media estimada de ambas muestras. Utilizaremos la librería `scipy` para obtener los valores de la inversa de la función distribución para una una distribución normal estandar, es decir Z ~ N(0,1), correspondientes a un intervalo de confianza $1 - \alpha$, o lo que es lo mismo, el valor de $Z_{\alpha/2}$.

Para la primera muestra, el intervalo de 95% de confianza es:

[140]:
```
from scipy.stats import norm
marg = lambda a, n : norm.ppf((1 - a)/2) * sqrt(5) / sqrt(n)
interv = lambda s, a : (mm(s) + marg(a, len(s)), mm(s) - marg(a, len(s)))
interv(nor10, .95)
```

[140]:
```
(97.9711346260824, 100.74294227478177)
```

Para la misma muestra, obtenemos el intervalo de 98% de confianza para la media:

[141]:
```
interv(nor10, .98)
```

[141]:
```
(97.7120620932989, 101.00201480756527)
```

De la misma forma obtenemos los intervalos de confianza para la muestra de 30 datos con 95% de confianza:

[142]:
```
interv(nor30, .95)
```

[142]:
```
(99.20858170831391, 100.80888560043235)
```

Y finalmente un intervalo del 98% de confianza para la muestra de 30 datos:

[143]:
```
interv(nor30, .98)
```

[143]:
```
(99.05900611177172, 100.95846119697454)
```

Como es posible observar ante la repetición de la experiencia, la gran mayoría de las veces, pero no todas, las medias de las distribuciones de base se encuentran en el intervalo de confianza inferido a partir de las muestras y la varianza conocida.

Ahora repetiremos la experienza pero utilizando la varianza muestral como estimador. Por ello redefiniremos nuestro método utilizando la distribución T de Student y calcularemos el intervalo de 95% de confianza para la muestra de diez datos de población normal con varianza desconocida:

[144]:
```
from scipy.stats import t
marg2 = lambda a, s : t.ppf((1 - a)/2, len(s) - 1) * sqrt(vm(s)/len(s))
interv2 = lambda s, a : (mm(s) + marg2(a, s), mm(s) - marg2(a, s))
interv2(nor10, .95)
```

[144]: (97.69323182468034, 101.02084507618383)

El intervalo de 98% de confianza para la misma muestra será:

[145]:
```
interv2(nor10, .98)
```

[145]: (97.28188328957371, 101.43219361129046)

Finalmente calcularemos los intervalos de confianza de la media, asumiendo una distribución normal con varianza desconocida, que se infieren a partir de la muestra de 30 datos. Primero con un 95% de confianza:

[146]:
```
interv2(nor30, .95)
```

[146]: (99.06963427474926, 100.947833033997)

Y con 98% de confianza:

[147]:
```
interv2(nor30, .98)
```

[147]: (98.87825779762437, 101.13920951112189)

Así hemos verificado empíricamente que repitiendo la experiencia, los intervalos bajo la asunción de de una varianza conocida son menores (para el mismo porcentaje de confianza) que los obtenidos desconociéndola y reemplazándola por la varianza muestral.

A continuación probaremos si nuestras muestras nos permiten inferir que provienen de distribuciones normales cuya varianza sea mayor a algún valor dado. Para ello definimos una función auxiliar que calcula el límite inferior de un intervalo de confianza unilateral con la ayuda de la distribución Chi cuadrado.

[148]:
```
from scipy.stats import chi2
lowbound = lambda s, p: ( len(s) - 1 ) * vm(s) / chi2.ppf(p, len(s) - 1)
```

Para probar a continuación con un 99% de confianza que la muestra de 30 datos proviene de una distribución normal cuya varianza es mayor a 5, su límite inferior unilateral debe ser entonces también mayor.

[149]:
```
print(lowbound(nor30, .99))
```

```
3.6989854435135374
```

Finalmente terminaremos nuestro trabajo realizando una prueba de bondad de ajuste, para la cual también nuestro estimador se basará en la distribución Chi cuadrado.

Generaremos primero los límites de las clases para un anchos de banda de 0.5

[150]:
```
from numpy import arange
gaps = arange(min(nor30), max(nor30), 0.5)
print(gaps)
```

```
[ 93.42722231  93.92722231  94.42722231  94.92722231  95.42722231
  95.92722231  96.42722231  96.92722231  97.42722231  97.92722231
  98.42722231  98.92722231  99.42722231  99.92722231 100.42722231
 100.92722231 101.42722231 101.92722231 102.42722231 102.92722231
 103.42722231 103.92722231 104.42722231 104.92722231 105.42722231]
```

Obtendremos ahora las frecuencias observadas para cada clase:

[151]:
```
obs = [len([x for x in nor30 if x >= l and x < l + 0.5]) for l in gaps]
print(obs)
```

```
[1, 0, 0, 0, 0, 0, 1, 2, 1, 2, 1, 5, 3, 0, 4, 3, 2, 1, 1, 0, 1, 0, 1, 0, 1]
```

Calcularemos entonces las frecuencias esperadas teniendo en cuenta que la muestra proviene de una distribución normal:

[152]:
```
esp = [round((norm.cdf(x + 0.5, loc=100, scale=sqrt(5)) \
              - norm.cdf(x, loc=100, scale=sqrt(5))) \
                  * 30) for x in gaps]
print(esp)
```

```
[0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 2, 2, 2, 1, 1, 1, 0, 0, 0, 0]
```

Para que la prueba de bondad de ajuste funcione, es condición que las clases tengan frecuencias mayores o iguales a 5, por lo que realizaremos una unificación de clases hasta que ambas series, observadas y esperadas, cumplan con lo requerido.

[153]:
```
obs2 = []
esp2 = []
obsacm = 0
espacm = 0
for pos in range(len(obs)):
    obsacm += obs[pos]
    espacm += esp[pos]
    if obsacm >= 5 and espacm >= 5:
        obs2.append(obsacm)
        esp2.append(espacm)
        obsacm = 0
        espacm = 0
if obsacm or espacm:
    esp2[-1] += espacm
    obs2[-1] += obsacm
print(obs2)
print(esp2)
```

```
[5, 8, 7, 10]
[5, 6, 9, 9]
```

Ahora si podemos calcular nuestro estimador como la sumatoria de los errores cuadrados relativos:

[154]:
```
squarerr = sum([(obs2[pos] - val)**2/val for pos, val in enumerate(esp2)])
print(squarerr)
```

```
1.2222222222222223
```

Finalmente, para rechazar con 99 por ciento de confianza la hipótesis de que nuestra muestra aleatoria proviene de una población con una distribución normal cuyos parámetros son los especificados, nuestro estimador debiera ser mayor al Chi cuadrado que corresponde a un alfa de 0.01 y los grados de libertad que se deducen de la unificación de las clases.

[155]:
```
squarerr > chi2.ppf(.01, len(esp2) - 1)
```

[155]:
```
True
```

Conclusiones

La realizació de la presente práctica ha significado para nosotros una interesante aproximación a problemas relativos a la simulación que son parte integral de un serio proyecto ingenieril. A la vez, los conceptos de inferencia estadística han sido lo sufiente introductorios para abordar la comprensión de ámbitos que los incluyen y que son crecientemente requisitos de la industria, como es el *machine learning* y *pattern recognition*.

No queremos desaprovechar la circunstancia para hacer notar el agradecimiento al acompañamiento pedagógico en vista de la situación epidemiológica, que junto con nuestro esfuerzo ha permitido la continuidad de proyecto universitario.

Referencias

[1] Dekking, F. & otros. (2005). *"A Modern Introduction to Probability and Statistics. "* London: Springer.

[2] *"Unit testing framework — Python 3.8.2 documentation"*. Recuperado de: https://docs.python.org/3/library/unittest.html

[3] *"IPython Kernel for Jupyter"*. (Version 5.3.4; IPython Development Team). Recuperado de https://ipython.org

[4] *"Python plotting package"*. (Version 3.3.3; John D. Hunter, Michael Droettboom). Recuperado de https://matplotlib.org

[5] *"SciPy: Scientific Library for Python"*. (Version 1.5.4). Recuperado de https://www.scipy.org

[6] *"Python documentation generator"*. (Version 3.3.1; Georg Brandl). Recuperado de http://sphinx-doc.org/

[7] *"A tool to merge / concatenate Jupyter (IPython) notebooks"*. (Version 0.0.4; John Bjorn Nelson). Recuperado de https://github.com/jbn/nbmerge

[8] *"GitHub: Where the World builds software"*. Recuperado de https://github.com/

[9] Goodger, D., van Rossum, G. (2001). *"PEP 257 – Docstring Conventions"* Recuperado de https://www.python.org/dev/peps/pep-0257/