Statistics

Jorge Sosa

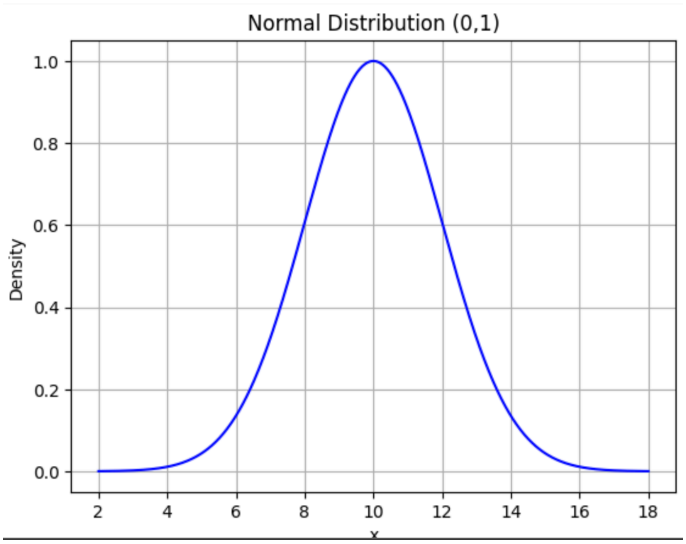
August 15, 2023

Distributions

1.-

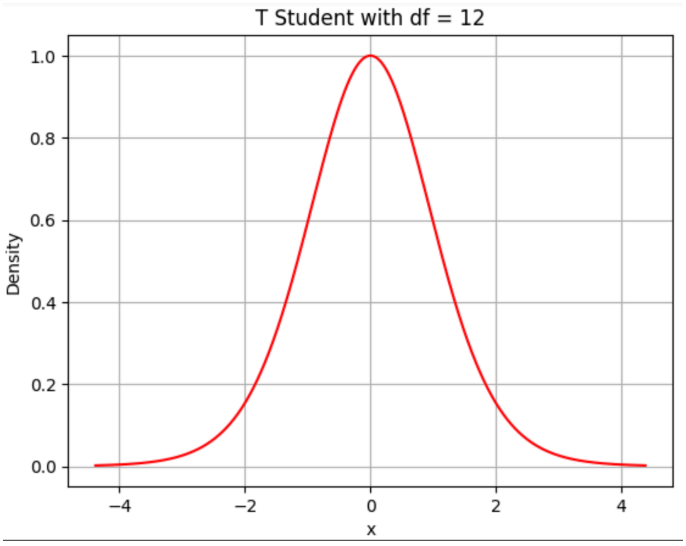
```
import numpy as np
import matplotlib.pyplot as plt

sigma = 2
miu = 10
def f(x):
  return (1 / (sigma * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - miu) / sigma)**2)
def main():
  xi = np.linspace(sigma,9*sigma,1000)
  y = [f(i) for i in xi]
  print(type(y))
  print(type(xi))
  plt.plot(xi, y/max(y), color='blue')
  plt.grid()
  plt.xlabel('x')
  plt.ylabel('Density')
  plt.title('Normal Distribution (0,1)')
  plt.show()
main()
```



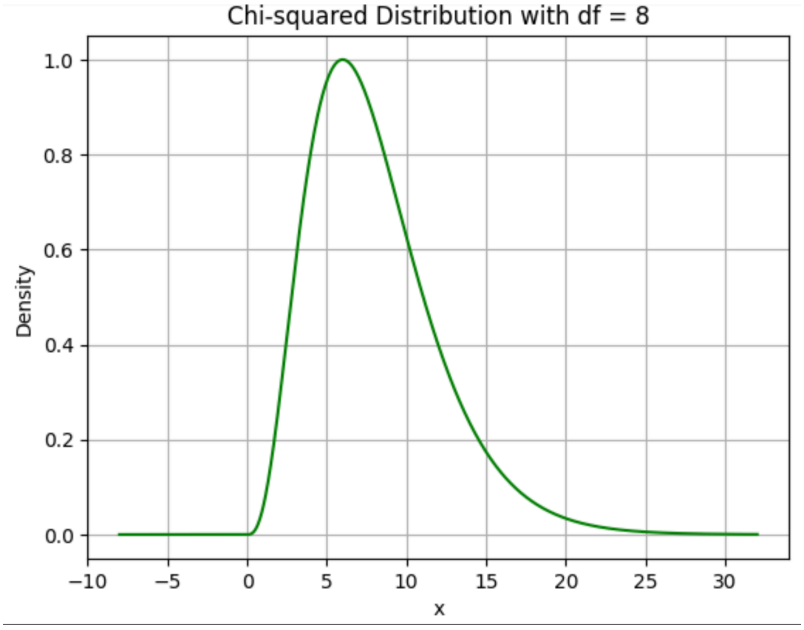
2.-

```
from scipy.stats import t
gl = 12  # Degrees of freedom
sigma = np.sqrt(gl / (gl - 2))
def f(x):
  return t.pdf(x,gl)
def main():
  xi = np.linspace(-4*sigma,4*sigma,1000)
  y = [f(i) for i in xi]
  plt.plot(xi,y/max(y),  color = "red")
  plt.grid()
  plt.xlabel('x')
  plt.ylabel('Density')
  plt.title(f'T Student with df = {gl}')
  plt.show()
main()
```



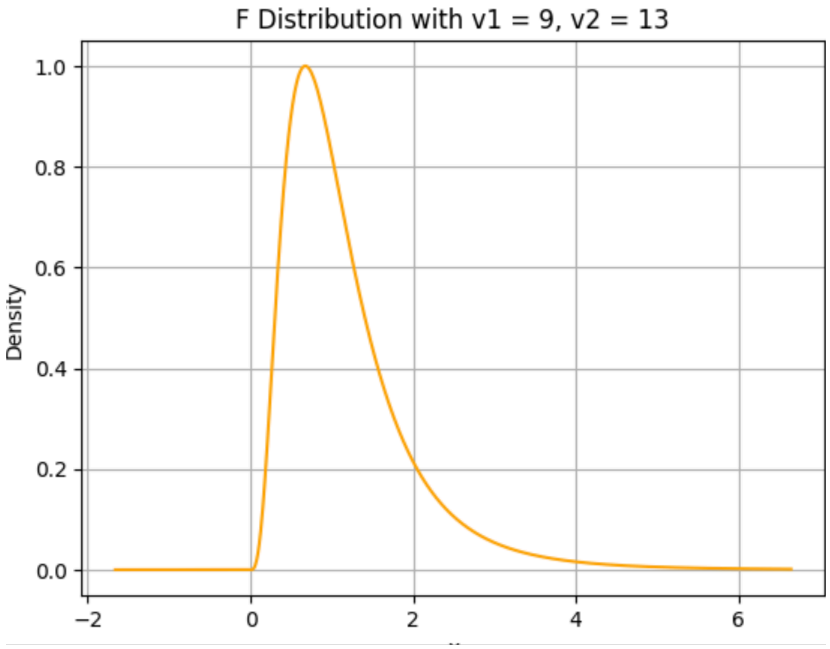
3.-

```
from scipy.stats import chi2
gl = 8  # Degrees of freedom
sigma = np.sqrt(2 * gl)
def f(x):
  return chi2.pdf(x, gl)
def main():
  xi = np.linspace(-2*sigma,8*sigma,1000)
  y = [f(i) for i in xi]
  plt.plot(xi,y/max(y),  color = "green")
  plt.grid()
  plt.xlabel('x')
  plt.ylabel('Density')
  plt.title(f'Chi-squared Distribution with df = {gl}')
  plt.show()
main()
```



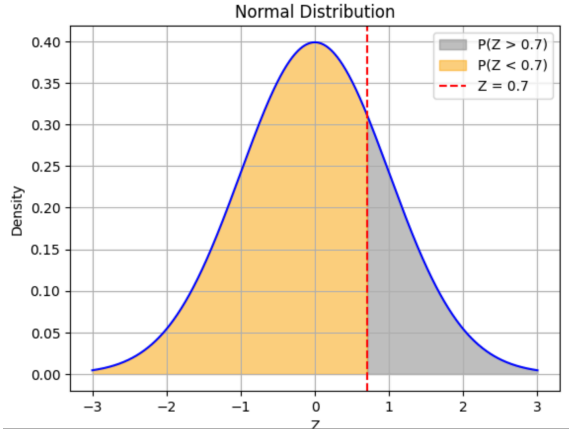
4.-

```
from scipy.stats import f
v1 = 9
v2 = 13
sigma = np.sqrt(2) * v2 * np.sqrt(v2 + v1 - 2) / (np.sqrt(v2 -
4) * (v2 - 2) * np.sqrt(v1))
def f_dist(x):
  return f.pdf(x, v1, v2)
def main():
  xi = np.linspace(-2*sigma,8*sigma,1000)
  y = [f_dist(i) for i in xi]
  plt.plot(xi, y/max(y), color='orange')
  plt.grid()
  plt.xlabel('x')
  plt.ylabel('Density')
  plt.title(f'F Distribution with v1 = {v1}, v2 = {v2}')
  plt.show()
main()
```



5.-

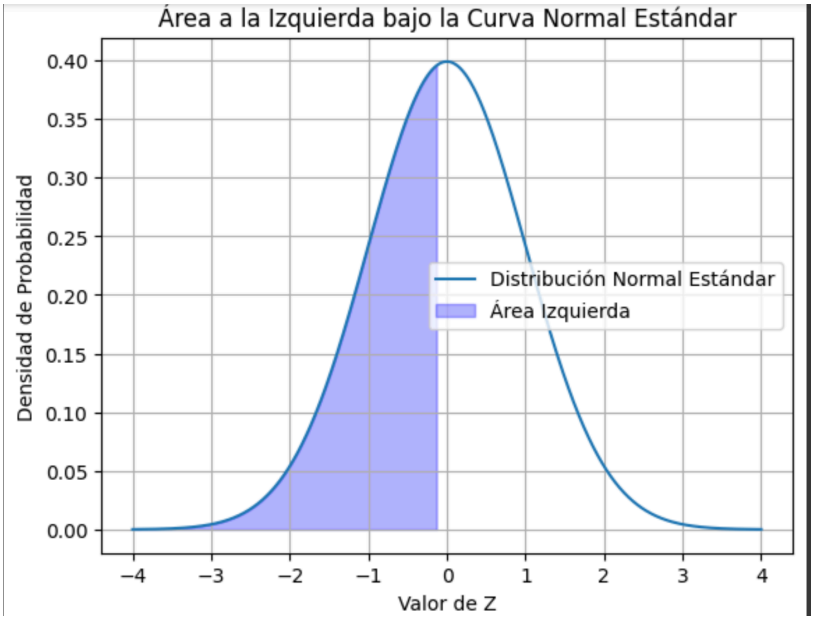
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
# Media y desviación estándar de la distribución normal estándar
media = 0
desviacion_estandar = 1
# Valores para el eje x (valores de Z)
x = np.linspace(-3, 3, 1000)
# Calcula la función de densidad de probabilidad (PDF) para la distribución normal
pdf = norm.pdf(x, loc=media, scale=desviacion_estandar)
# Crear la gráfica de la función de densidad de probabilidad
plt.plot(x, pdf, color='blue')
plt.fill_between(x, pdf, where=(x > 0.7), color='gray', alpha=0.5, label='P(Z > 0.7)')
plt.fill_between(x, pdf, where=(x < 0.7), color='orange', alpha=0.5, label='P(Z < 0.7)')
plt.axvline(x=0.7, color='red', linestyle='--', label='Z = 0.7')
plt.xlabel('Z')
plt.ylabel('Density')
plt.title('Normal Distribution')
plt.legend()
plt.grid()
plt.show()
# a) P(Z > 0.7)
prob_a = 1 - norm.cdf(0.7, loc=media, scale=desviacion_estandar)
print(f'a) P(Z > 0.7) = {prob_a:.7f}')
# b) P(Z < 0.7)
prob_b = norm.cdf(0.7, loc=media, scale=desviacion_estandar)
print(f'b) P(Z < 0.7) = {prob_b:.7f}')
# c) P(Z = 0.7) (la probabilidad de un valor puntual en una distribución continua es 0)
prob_c = 0
print(f'c) P(Z = 0.7) = {prob_c:.7f}')
```




```
a) P(Z > 0.7) = 0.2419637
b) P(Z < 0.7) = 0.7580363
c) P(Z = 0.7) = 0.0000000
```

6.-

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
# Área a la izquierda bajo la curva (probabilidad acumulada)
area_izquierda = 0.45
# Encontrar el valor de Z correspondiente al área dada
valor_z = norm.ppf(area_izquierda)
print("El valor de Z correspondiente al 45% de los valores inferiores es:",
valor_z)
# Crear valores para la gráfica de la distribución normal estándar
x = np.linspace(-4, 4, 1000)
y = norm.pdf(x)
# Crear la gráfica
plt.plot(x, y, label='Distribución Normal Estándar')
plt.fill_between(x, y, where=(x <= valor_z), color='blue', alpha=0.3,
label='Área Izquierda')
plt.xlabel('Valor de Z')
plt.ylabel('Densidad de Probabilidad')
plt.title('Área a la Izquierda bajo la Curva Normal Estándar')
plt.legend()
plt.grid()
plt.show()
```



7.-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
# Datos proporcionados
miu = 100
sigma = 7
# Resultados dados
res1 = 0.031645
res2 = 0.968354
res3 = 0.89179
# Verificación de P(X < 87)
verif_1 = norm.cdf(87, loc=miu, scale=sigma)
# Verificación de P(X > 87)
verif_2 = 1 - norm.cdf(87, loc=miu, scale=sigma)
# Verificación de P(87 < X < 110)
verif_3 = norm.cdf(110, loc=miu, scale=sigma) - norm.cdf(87, loc=miu, scale=sigma)
# Crear una lista con los resultados y verificaciones
datos = [
    [res1, verif_1],
    [res2, verif_2],
    [res3, verif_3]
]
# Crear un DataFrame con la lista de datos
df = pd.DataFrame(datos, columns=['Resultado', 'Verificación'])
# Establecer el número de decimales
num_decimales = 6
# Cambiar el formato de impresión del DataFrame
pd.set_option('display.float_format', lambda x: f'{x:.{num_decimales}f}')
# Imprimir el DataFrame con formato
print("Matriz de Resultados y Verificaciones:\n")
print(df)
```

```
Matriz de Resultados y Verificaciones:

   Resultado  Verificación
0   0.031645      0.031645
1   0.968354      0.968355
2   0.891790      0.891791
```

8.-

```
import scipy.stats as stats
# Resultados dados
resultado_1 = 0.6860532
resultado_2 = 0.082253
valor_t_5_percentil = -1.812461
# Verificación de P(X < 0.5)
verif_1_ = stats.t.cdf(0.5, df=10)
print(f'Verificación P(X < 0.5): {verif_1}')
# Verificación de P(X > 1.5)
verif_2_ = 1 - stats.t.cdf(1.5, df=10)
print(f'Verificación P(X > 1.5): {verif_2}')
# Verificación de la t que sólo el 5% son inferiores a ella
verif_3_ = stats.t.ppf(0.05, df=10)
print(f'Verificación t que sólo el 5% son inferiores a ella: {verif_3}')
# Crear una lista con los resultados y verificaciones
dat = [
    [resultado_1, verif_1_],
    [resultado_2, verif_2_],
    [valor_t_5_percentil, verif_3_],
]
# Crear un DataFrame con la lista de datos
df = pd.DataFrame(dat, columns=['Resultado', 'Verificación'])
# Establecer el número de decimales
num_decimales = 8
# Cambiar el formato de impresión del DataFrame
pd.set_option('display.float_format', lambda x: f'{x:.{num_decimales}f}')
# Imprimir el DataFrame con formato
print("Matriz de Resultados y Verificaciones:\n")
print(df)
```

```
Verificación P(X < 0.5): 0.031645416116672626
Verificación P(X > 1.5): 0.9683545838833274
Verificación t que sólo el 5% son inferiores a ella: 0.8917908583734926
Matriz de Resultados y Verificaciones:

    Resultado  Verificación
0  0.68605320    0.68605320
1  0.08225300    0.08225366
2 -1.81246100   -1.81246112
```

9.-

```
import scipy.stats as stats
# Grados de libertad
gl = 6
# Resultados dados
res1 = 0.1911532
res2 = 0.9196986
res3 = 12.59159
# Verificación de P(X2 < 3)
verif_1 = stats.chi2.cdf(3, df=gl)
# Verificación de P(X2 > 2)
verif_2 = 1 - stats.chi2.cdf(2, df=gl)
# Verificación del valor x de chi
verif_3 = stats.chi2.ppf(1-0.05, df=gl)
# Crear una lista con los resultados y verificaciones
datos = [
    [res1, verif_1],
    [res2, verif_2],
    [res3, verif_3],
]
# Crear un DataFrame con la lista de datos
df = pd.DataFrame(datos, columns=['Resultado', 'Verificación'])
# Establecer el número de decimales
num_decimales = 8
# Cambiar el formato de impresión del DataFrame
pd.set_option('display.float_format', lambda x: f'{x:.
{num_decimales}f}')
# Imprimir el DataFrame con formato
print("Matriz de Resultados y Verificaciones:\n")
print(df)
```

```
Matriz de Resultados y Verificaciones:

     Resultado  Verificación
0  0.19115320    0.19115317
1  0.91969860    0.91969860
2 12.59159000   12.59158724
```

10.-

```
import scipy.stats as stats
# Grados de libertad
v1 = 8
v2 = 10
# Resultados dados
res11 = 0.8492264
res22 = 0.05351256
res33 = 0.6131229
# Verificación de P(X < 2)
verif_11 = stats.f.cdf(2, dfn=v1, dfd=v2)
# Verificación de P(X > 3)
verif_22 = 1 - stats.f.cdf(3, dfn=v1, dfd=v2)
# Verificación del valor x
verif_33 = stats.f.ppf(0.25, dfn=v1, dfd=v2)
# Crear una matriz con los resultados y verificaciones
data1 = [
    [res11, verif_11],
    [res22, verif_22],
    [res33, verif_33],
]
# Crear un DataFrame con la lista de datos
df1 = pd.DataFrame(data1, columns=['Resultado', 'Verificación'])
# Establecer el número de decimales
num_decimales = 8
# Cambiar el formato de impresión del DataFrame
pd.set_option('display.float_format', lambda x: f'{x:.
{num_decimales}f}')
# Imprimir el DataFrame con formato
print("Matriz de Resultados y Verificaciones:\n")
print(df1)
```

```
Matriz de Resultados y Verificaciones:

    Resultado  Verificación
0 0.84922640    0.84922644
1 0.05351256    0.05351256
2 0.61312290    0.61312285
```

11.-

```
import scipy.stats as stats
# Parámetros de la distribución normal
miu = 65
sigma = 20
# Calcular la proporción de servicios en menos de 60 minutos
p = norm.cdf(60, loc=miu, scale=sigma) * 100
# Imprimir el resultado con dos decimales
print("Proporción de servicios en menos de 60 minutos:",
round(p, 2), "%")
```

```
Proporción de servicios en menos de 60 minutos: 40.13 %
```