stochacalculus

David Peña Peralta

2025-01-11

Table of contents

1 Ecuaciones Diferenciales Estocásticas

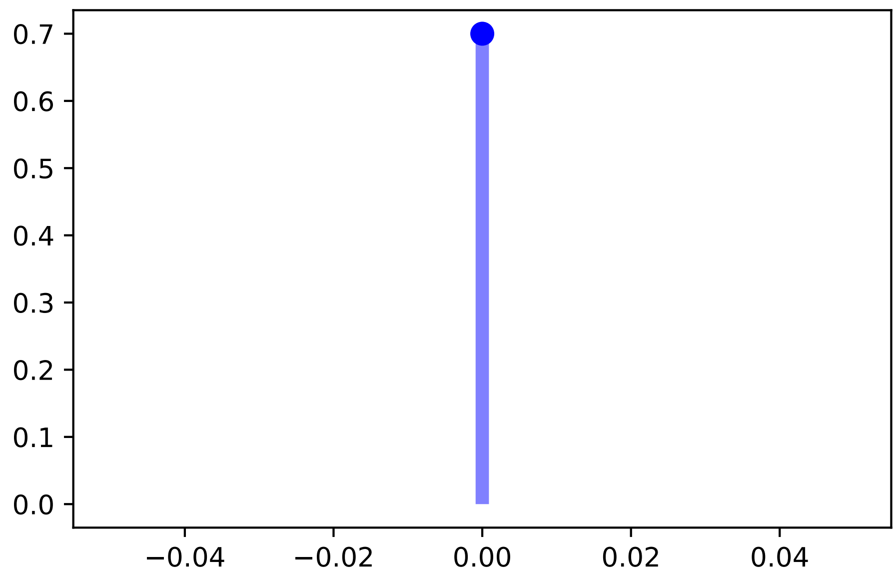
Ahora, vamos

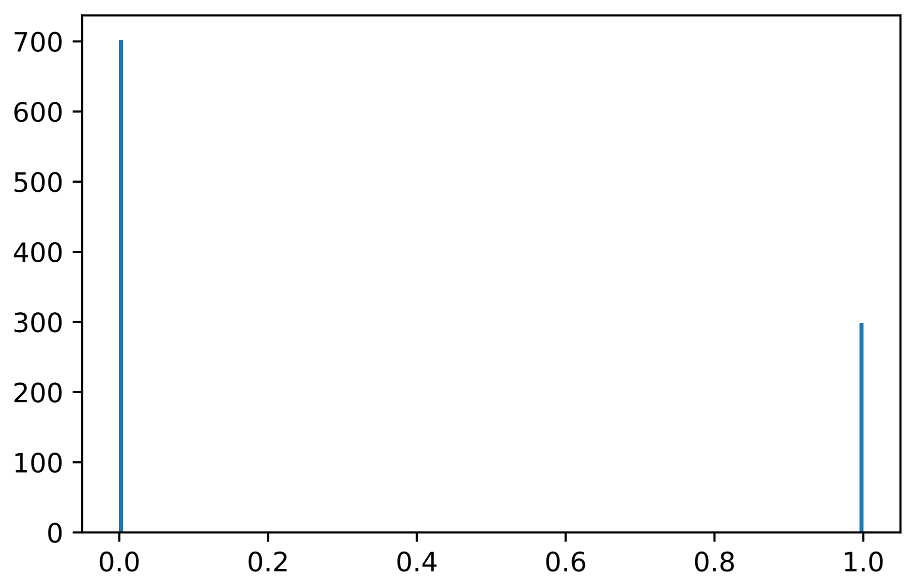
2 plt.show()

Ejecute y explica en pocas palabras la salida del código ex_001.py

```
from scipy.stats import multivariate_normal
from mpl_toolkits.mplot3d import axes3d
from scipy.stats import norm
import numpy as np
from scipy.stats import bernoulli
import matplotlib.pyplot as plt
fig_01, ax_01 = plt.subplots(1, 1)
fig_02, ax_02 = plt.subplots(1, 1)
p = 0.3
mean, var, skew, kurt = bernoulli.stats(p, moments='mvsk')
print(mean, var, skew,kurt)

x = np.arange(bernoulli.ppf(0.01, p), bernoulli.ppf(0.99, p))
ax_01.plot(x, bernoulli.pmf(x, p), 'bo', ms=8, label='bernoulli pmf')
ax_01.vlines(x, 0, bernoulli.pmf(x, p), colors='b', lw=5, alpha=0.5)
r = bernoulli.rvs(p, size=1000)
ax_02.hist(r, bins=200)
plt.show()
```

```
0.3 0.21 0.8728715609439694 -1.2380952380952381
```





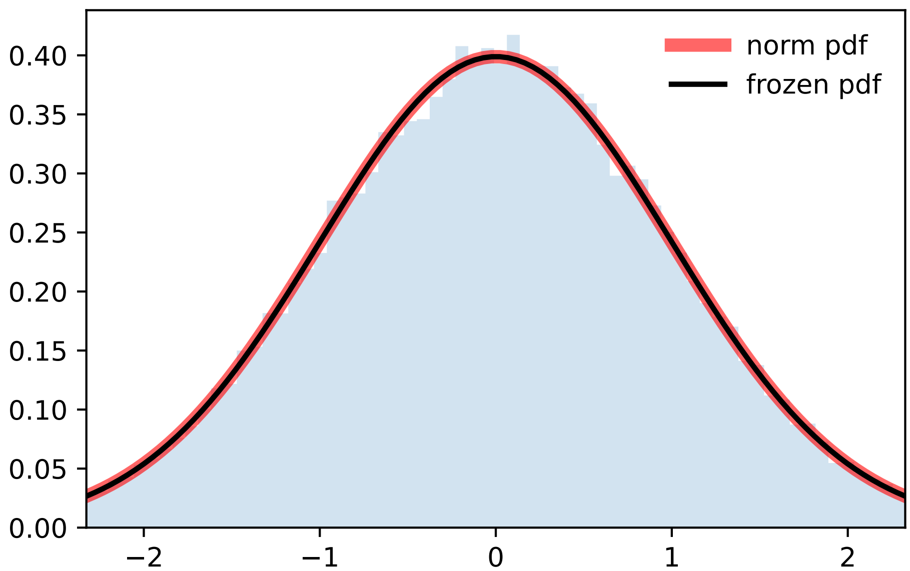
El código posee 3 salidas: * Un vector $[0.3, 0.21, 0.87, -1.23]$ * Dos gráficas.

El vector hace referencia a los momentos de la distribución bernoulli con parámetro $p = 0.3$. * mean hace referencia a la media. * var hace referencia a la varianza. * skew hace referencia al sesgo. * kurt hace referencia a la kurtosis.

Finalmente, las dos gráficas: * La primera hace referencia a la función de probabilidad. Notemos que $\mathcal{P}[X = 0] = 0.7$, lo que muestra la gráfica. Notemos que la gráfica va de -0.04 a 0.04, por lo tanto no se iba a mostrar el caso $X = 1$.
* La segunda hace referencia a una simulación: Se generaron una muestra de tamaño N variables aleatorias con distribución bernoulli. Como la distribución bernoulli tiene media Np, pretende mostrar que en efecto, habrá de forma aproximada Np valores igual a 1 y $N(1-p)$ valores igual a 0.

Ejecute y explica en pocas palabras la salida del código ex_002.py

```
fig, ax = plt.subplots(1, 1)
mean, var, skew, kurt = norm.stats(moments='mvsk')

x = np.linspace(norm.ppf(0.01), norm.ppf(0.99), 100)
ax.plot(
    x,
    norm.pdf(x),
    'r-',
    lw=5,
    alpha=0.6,
    label='norm pdf'
)
rv = norm()
ax.plot(x, rv.pdf(x), 'k-', lw=2, label='frozen pdf')
vals = norm.ppf([0.001, 0.5, 0.999])

np.allclose([0.001, 0.5, 0.999], norm.cdf(vals))

r = norm.rvs(size=50000)

ax.hist(r, density=True, bins='auto', histtype='stepfilled', alpha=0.2)
ax.set_xlim([x[0], x[-1]])
ax.legend(loc='best', frameon=False)
plt.show()
```



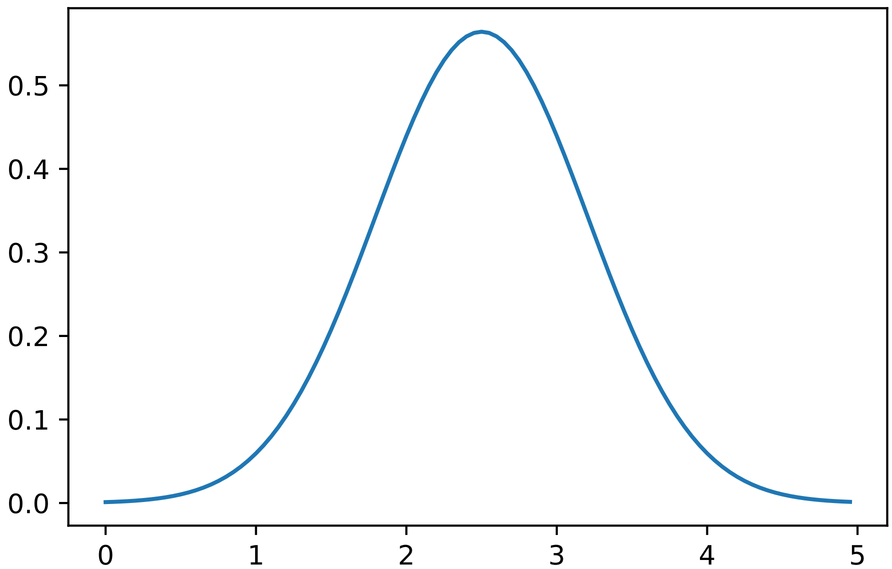
El código posee una gráfica. Que hace referencia a una simulación de variables aleatorias normales. Notemos que * El elemento en azul, hace referencia a un histograma que refleja las frecuencias de los valores generados. * Mientras que la linea roja, muestra la función de densidad de una variable aleatoria estándar.

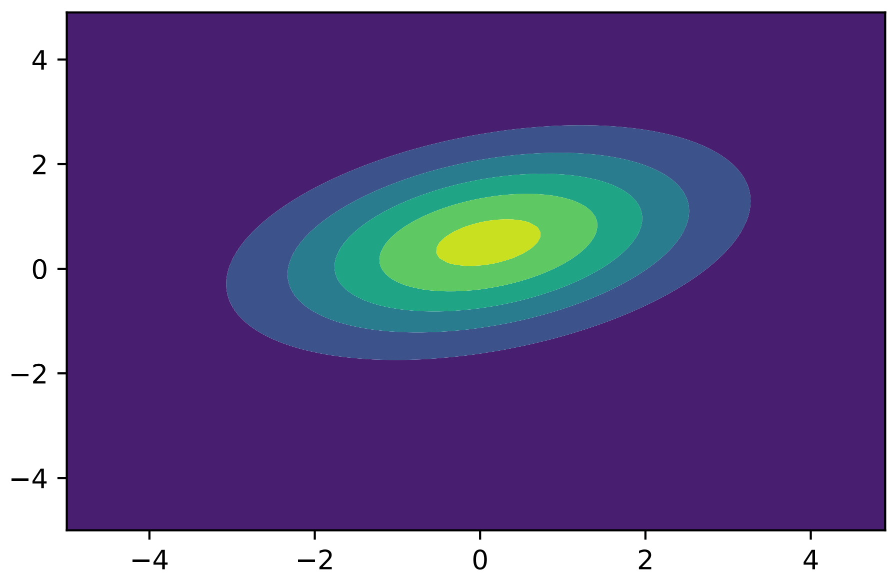
Ejecute y explica en pocas palabras la salida del código ex_003.py

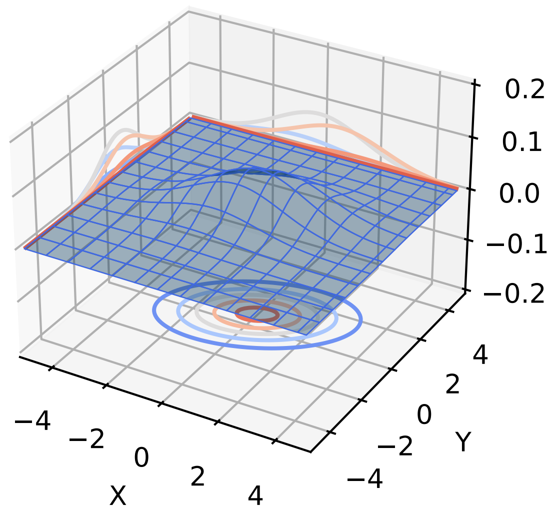
Para cambiar el vector de medias μ y la matriz Σ hay que prestar atención en la linea donde aparece la función `multivariate_normal()` que de forma simple posee dos parámetros: * El vector de medias $\mu = [0.5, -0.2]$ * La matriz de covarianza $\Sigma = [[2.0, 0.3], [0.3, 0.5]]$

```
x = np.linspace(0, 5, 100, endpoint=False)
y = multivariate_normal.pdf(x, mean=2.5, cov=0.5)

fig1 = plt.figure()
ax = fig1.add_subplot(111)
ax.plot(x, y)

x, y = np.mgrid[-5:5:.1, -5:5:.1]
pos = np.dstack((x, y))
rv = multivariate_normal([0.1, 0.5], [[3.0, 0.3], [0.75, 1.5]])
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(x, y, rv.pdf(pos))
# plt.show()
```

```python
ax = plt.figure().add_subplot(projection='3d')
ax.plot_surface(
    x,
    y,
    rv.pdf(pos),
    edgecolor='royalblue',
    lw=0.5,
    rstride=8,
    cstride=8,
    alpha=0.4
)
ax.contour(x, y, rv.pdf(pos), zdir='z', offset=-.2, cmap='coolwarm')
ax.contour(x, y, rv.pdf(pos), zdir='x', offset=-5, cmap='coolwarm')
ax.contour(x, y, rv.pdf(pos), zdir='y', offset=5, cmap='coolwarm')

ax.set(
    xlim=(-5, 5),
    ylim=(-5, 5),
    zlim=(-0.2, 0.2),
    xlabel='X',
    ylabel='Y',
    zlabel='Z'
)
plt.show()
```


0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5


4
2
0
−2
−4
−4
−2
0
2
4

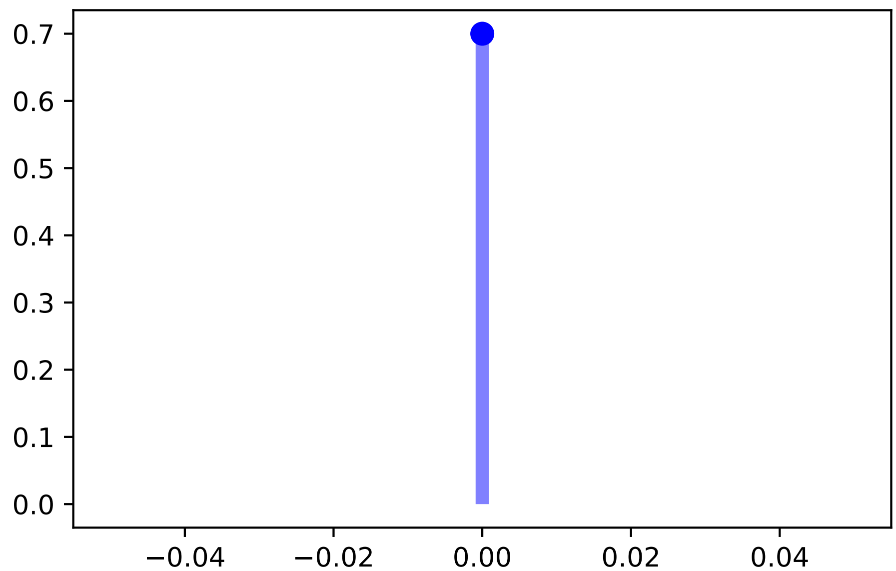


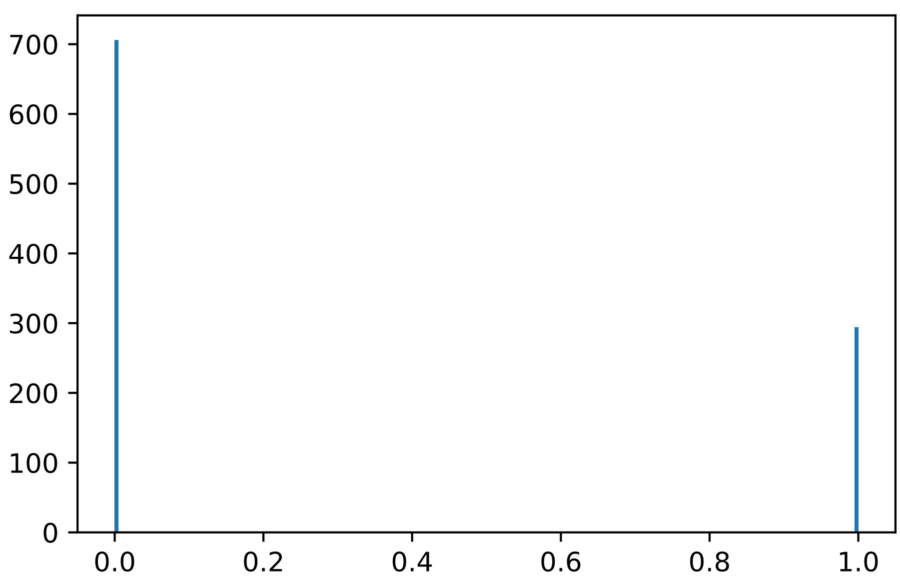
Generando Normales

```
import numpy as np
from scipy.stats import bernoulli
import matplotlib.pyplot as plt

fig_01, ax_01 = plt.subplots(1, 1)
fig_02, ax_02 = plt.subplots(1, 1)

p = 0.3

x = np.arange(bernoulli.ppf(0.01, p), bernoulli.ppf(0.99, p))
ax_01.plot(x, bernoulli.pmf(x, p), 'bo', ms = 8, label = 'bernoulli pmf')
ax_01.vlines(x, 0, bernoulli.pmf(x, p), colors = 'b', lw = 5, alpha = 0.5)

r = bernoulli.rvs(p, size = 1000)
ax_02.hist(r, bins = 200)
```

```
(array([706.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
```

```
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        0., 294.]),
array([0.   , 0.005, 0.01 , 0.015, 0.02 , 0.025, 0.03 , 0.035, 0.04 ,
       0.045, 0.05 , 0.055, 0.06 , 0.065, 0.07 , 0.075, 0.08 , 0.085,
       0.09 , 0.095, 0.1  , 0.105, 0.11 , 0.115, 0.12 , 0.125, 0.13 ,
       0.135, 0.14 , 0.145, 0.15 , 0.155, 0.16 , 0.165, 0.17 , 0.175,
       0.18 , 0.185, 0.19 , 0.195, 0.2  , 0.205, 0.21 , 0.215, 0.22 ,
       0.225, 0.23 , 0.235, 0.24 , 0.245, 0.25 , 0.255, 0.26 , 0.265,
       0.27 , 0.275, 0.28 , 0.285, 0.29 , 0.295, 0.3  , 0.305, 0.31 ,
       0.315, 0.32 , 0.325, 0.33 , 0.335, 0.34 , 0.345, 0.35 , 0.355,
       0.36 , 0.365, 0.37 , 0.375, 0.38 , 0.385, 0.39 , 0.395, 0.4  ,
       0.405, 0.41 , 0.415, 0.42 , 0.425, 0.43 , 0.435, 0.44 , 0.445,
       0.45 , 0.455, 0.46 , 0.465, 0.47 , 0.475, 0.48 , 0.485, 0.49 ,
       0.495, 0.5  , 0.505, 0.51 , 0.515, 0.52 , 0.525, 0.53 , 0.535,
       0.54 , 0.545, 0.55 , 0.555, 0.56 , 0.565, 0.57 , 0.575, 0.58 ,
       0.585, 0.59 , 0.595, 0.6  , 0.605, 0.61 , 0.615, 0.62 , 0.625,
       0.63 , 0.635, 0.64 , 0.645, 0.65 , 0.655, 0.66 , 0.665, 0.67 ,
       0.675, 0.68 , 0.685, 0.69 , 0.695, 0.7  , 0.705, 0.71 , 0.715,
       0.72 , 0.725, 0.73 , 0.735, 0.74 , 0.745, 0.75 , 0.755, 0.76 ,
       0.765, 0.77 , 0.775, 0.78 , 0.785, 0.79 , 0.795, 0.8  , 0.805,
       0.81 , 0.815, 0.82 , 0.825, 0.83 , 0.835, 0.84 , 0.845, 0.85 ,
       0.855, 0.86 , 0.865, 0.87 , 0.875, 0.88 , 0.885, 0.89 , 0.895,
       0.9  , 0.905, 0.91 , 0.915, 0.92 , 0.925, 0.93 , 0.935, 0.94 ,
       0.945, 0.95 , 0.955, 0.96 , 0.965, 0.97 , 0.975, 0.98 , 0.985,
       0.99 , 0.995, 1.   ]),
<BarContainer object of 200 artists>)
```



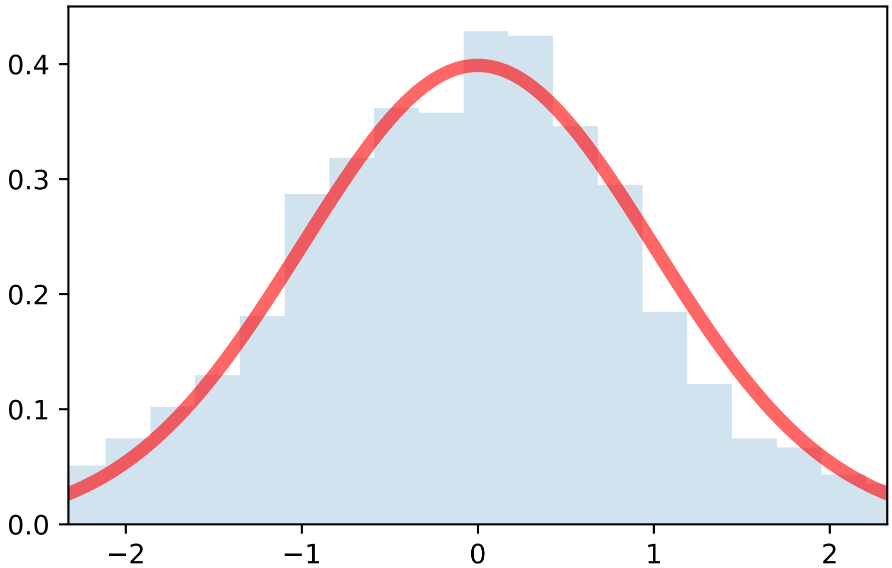



```
from scipy.stats import norm

fig, ax = plt.subplots(1,1)

x = np.linspace(norm.ppf(0.01),norm.ppf(0.99), 100)
```

```python
ax.plot(x, norm.pdf(x),'r-', lw = 5, alpha = 0.6)

r = norm.rvs(size = 1000)

ax.hist(r, density = True, bins = 'auto', histtype = 'stepfilled', alpha = 0.2)
ax.set_xlim(x[0], x[-1])
ax.legend(loc='best', frameon = False)
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an u
```

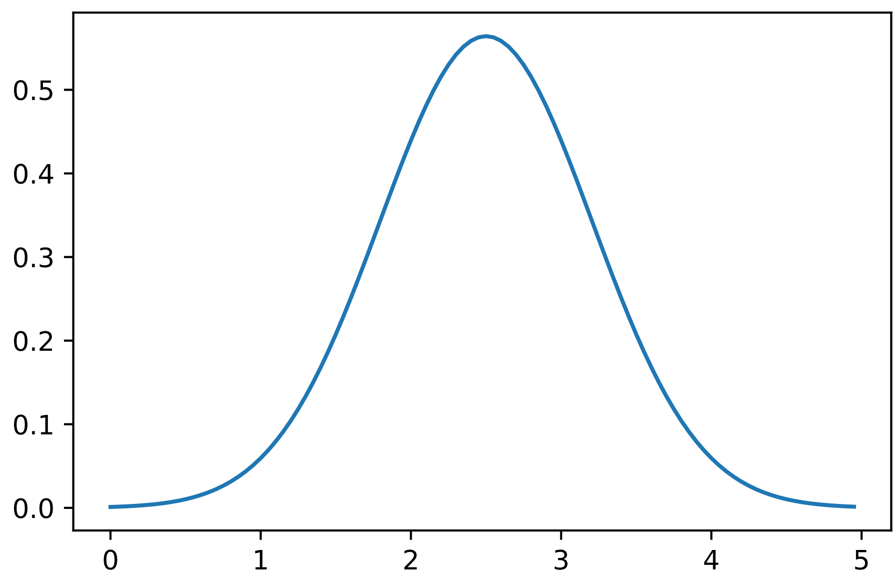
```
<matplotlib.legend.Legend at 0x171c6c412d0>
```

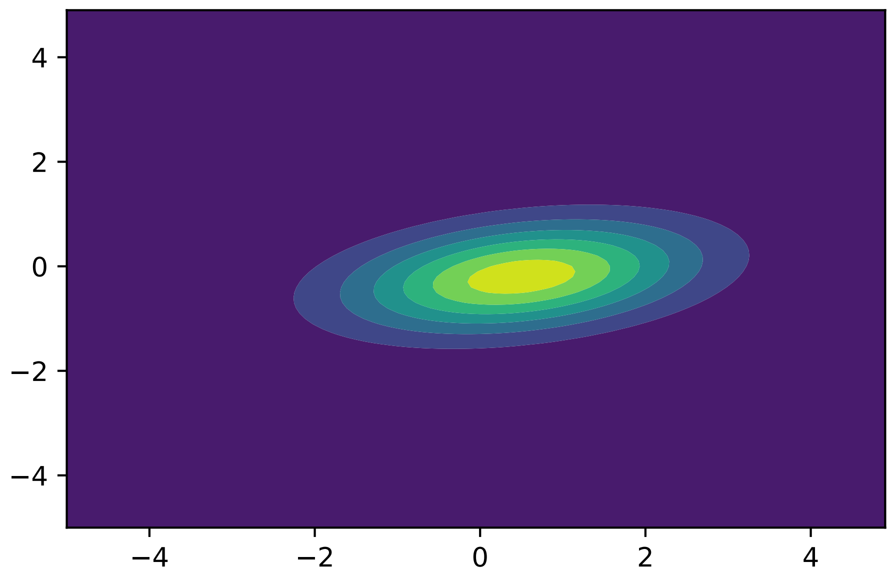


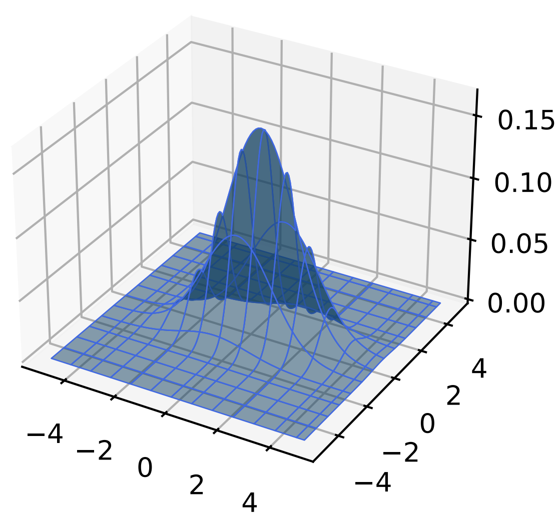

```python
from mpl_toolkits.mplot3d import axes3d
from scipy.stats import multivariate_normal

x = np.linspace(0,5,100,endpoint = False)
y = multivariate_normal.pdf(x , mean = 2.5, cov = 0.5)

fig1 = plt.figure()
ax = fig1.add_subplot(111)
ax.plot(x,y)
```

```
x, y = np.mgrid[-5:5:.1, -5:5:.1]

pos = np.dstack((x,y))
rv = multivariate_normal([0.5, -0.2], [[2.0,0.3], [0.3,0.5]])
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(x,y, rv.pdf(pos))

ax = plt.figure().add_subplot(projection = '3d')
ax.plot_surface(x,
                y,
                rv.pdf(pos),
                edgecolor = 'royalblue',
                lw = 0.5,
                rstride = 8,
                cstride = 8,
                alpha = 0.5)
```

```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x171c62a7a90>
```




4
2
0
−2
−4
−4
−2
0
2
4


0.15
0.10
0.05
0.00
−4
−2
0
2
4
4
2
0
−2
−4

References