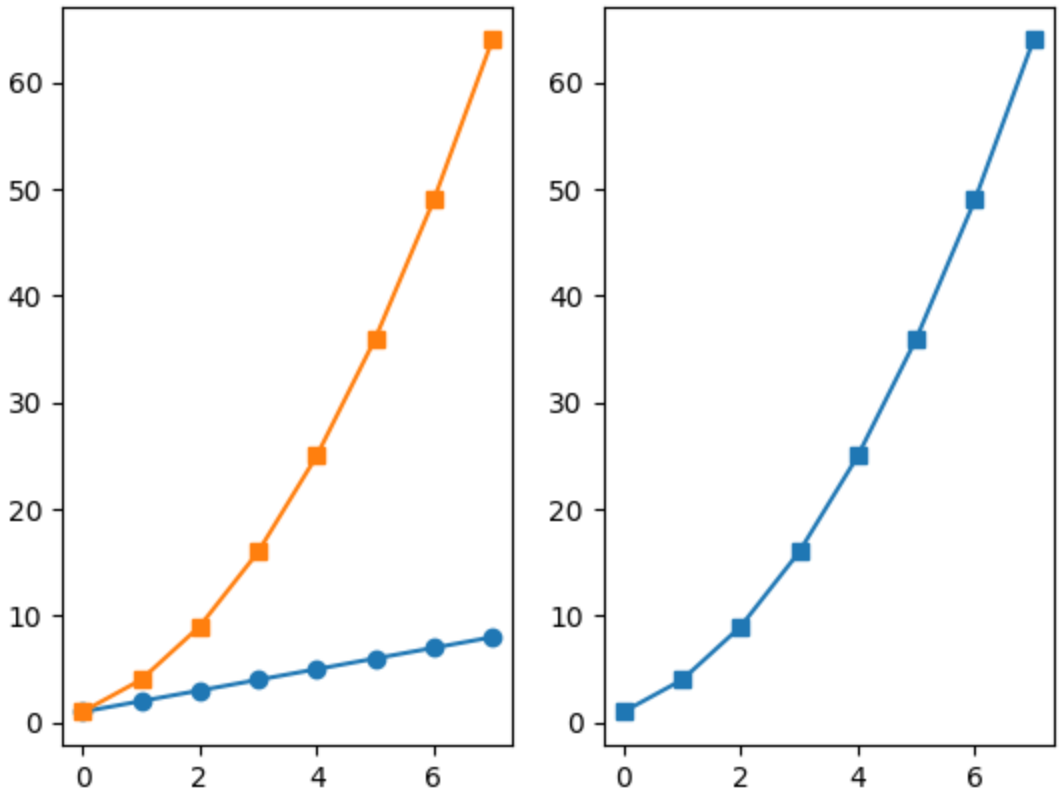
Subplots

1 - Função subplot

In [1]:
```
%matplotlib notebook

import matplotlib.pyplot as plt
import numpy as np

plt.subplot?
```

2 - Primeiro subplot

In [2]:
```
plt.figure()
# subplot com 1 linha, 2 colunas e o "eixos" atual são os eixos da 1ª sub-parcela
plt.subplot(1,2,1)

linear = np.array([1,2,3,4,5,6,7,8])

plt.plot(linear,'-o')
```



Out[2]:
```
[<matplotlib.lines.Line2D at 0x7fa1831b0ee0>]
```

3 - Segundo subplot

In [3]:
```
exponential = linear**2
plt.subplot(1,2,2)
```

```
plt.plot(exponential, '-s')
# subplot com 1 linha, 2 colunas e o "eixos" atual são os eixos da 2ª sub-parcela
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x7fa1831f44f0>]
```

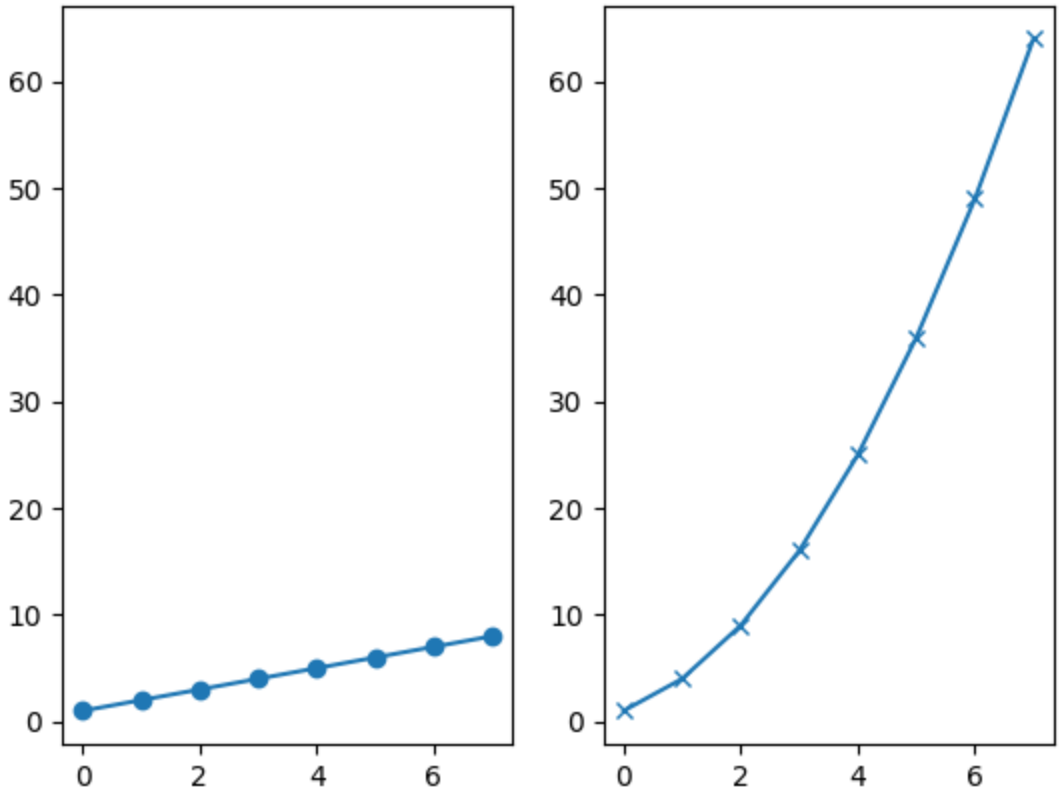
4 - Trocando de "eixos"

In [4]:
```
# plot exponential na 1ª sub-parcela
plt.subplot(1,2,1)
plt.plot(exponential, "-s")
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x7fa1831f4be0>]
```

5 - Compartilhamento de eixo

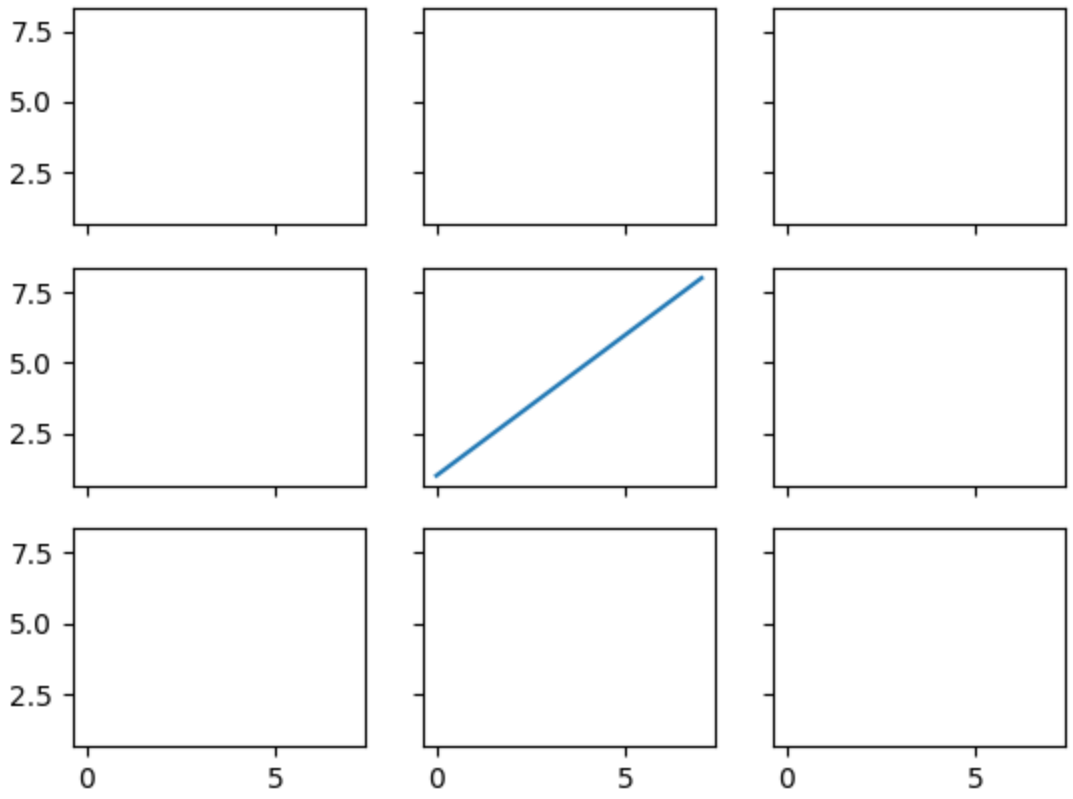
In [5]:
```
plt.figure()
ax1 = plt.subplot(1, 2, 1)
plt.plot(linear, '-o')
# sharey=ax1 garante que os subplots compartilham o mesmo (tamanho) eixo y
ax2 = plt.subplot(1, 2, 2, sharey=ax1)
plt.plot(exponential, '-x')
```



Out[5]:
```
[<matplotlib.lines.Line2D at 0x7fa183284c40>]
```

6 - Grid de subplots (função subplotsssss)

In [6]:
```
# 3x3 grid de subplots
fig, ((ax1,ax2,ax3), (ax4,ax5,ax6),(ax7,ax8,ax9)) = plt.subplots(3,3, sharex=True, sharey=
# plotar os dados lineares no 5° subplot
ax5.plot(linear, '-')
```

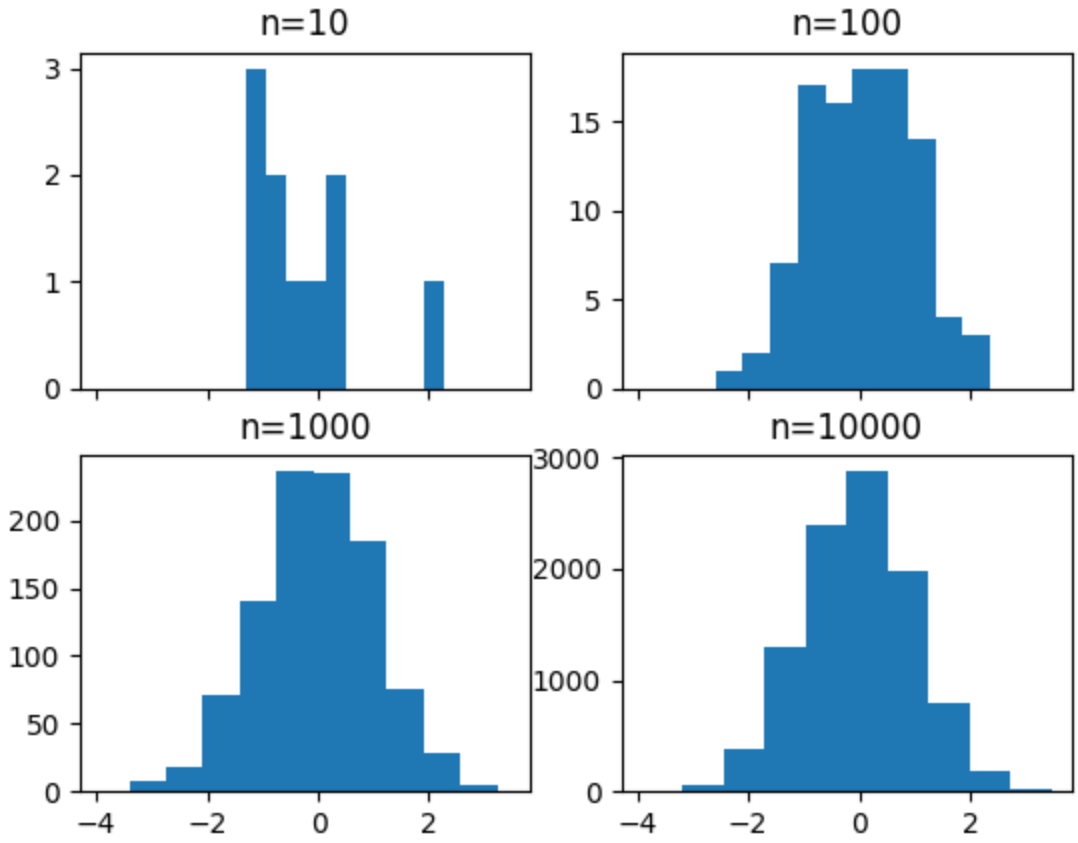


Out[6]: [<matplotlib.lines.Line2D at 0x7fa183489ee0>]

Histogramas

7 - Histograma através de amostragem

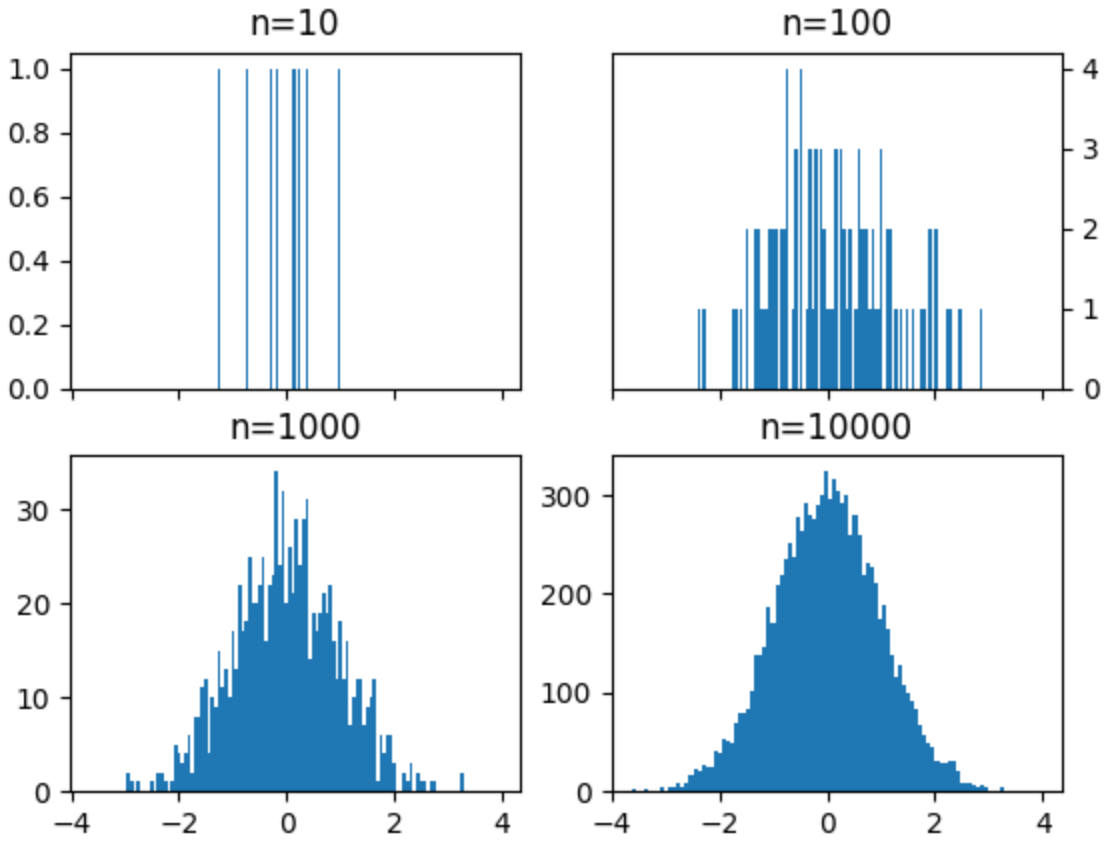
In [7]:
```
# criar 2 x 2 de objetos de eixos
fig, ((ax1,ax2), (ax3,ax4)) = plt.subplots(2,2, sharex=True)
axs = [ax1,ax2,ax3,ax4]
# n = 10, 100, 1000, and 10000 amostrar da distribuição normal
for n in range(0, len(axs)):
    sample_size = 10**(n+1)
    sample = np.random.normal(loc=0.0, scale=1.0, size = sample_size)
    axs[n].hist(sample)
    axs[n].set_title('n={}'.format(sample_size))
```



8 - O número de Bins

In [8]:

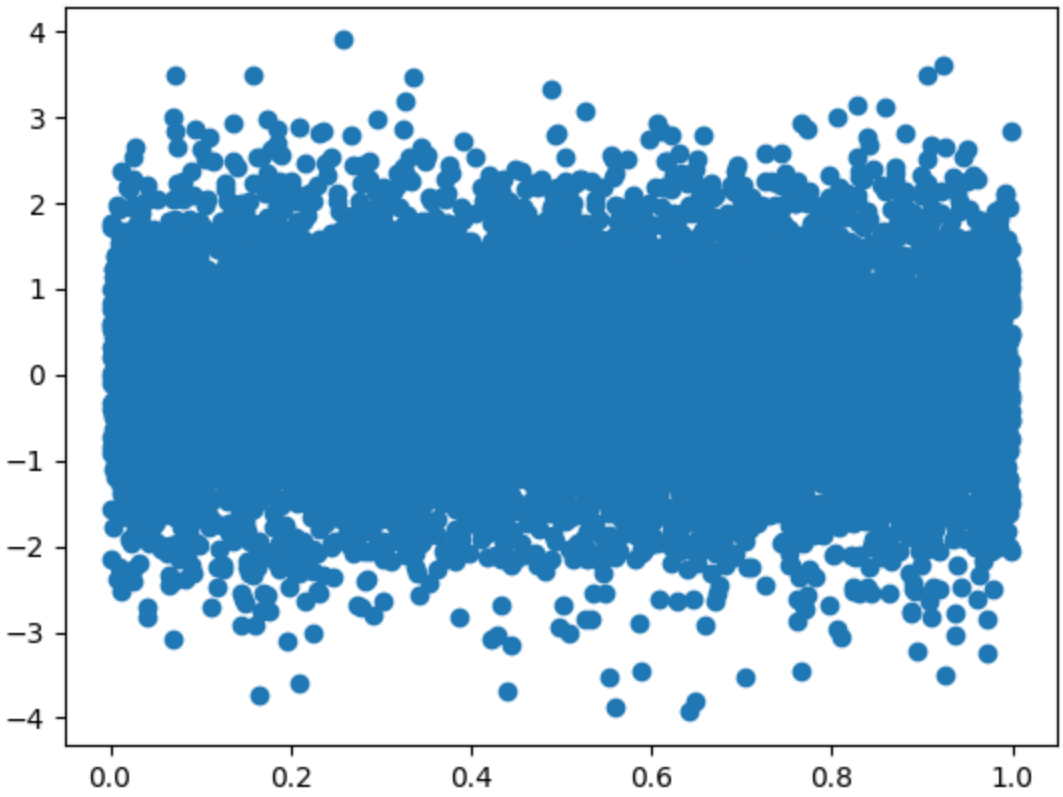
```
# criar 2 x 2 de objetos de eixos
fig, ((ax1,ax2), (ax3,ax4)) = plt.subplots(2,2, sharex=True)
axs = [ax1,ax2,ax3,ax4]
# n = 10, 100, 1000, and 10000 amostrar da distribuição normal
for n in range(0, len(axs)):
    sample_size = 10**(n+1)
    sample = np.random.normal(loc=0.0, scale=1.0, size = sample_size)
    axs[n].hist(sample, bins=100)
    axs[n].set_title('n={}'.format(sample_size))
```



9 - Scatter plot para usar com GridSpec

In [9]:

```
plt.figure()
Y = np.random.normal(loc=0.0, scale=1.0, size=10000)
X = np.random.random(size=10000)
plt.scatter(X,Y)
```

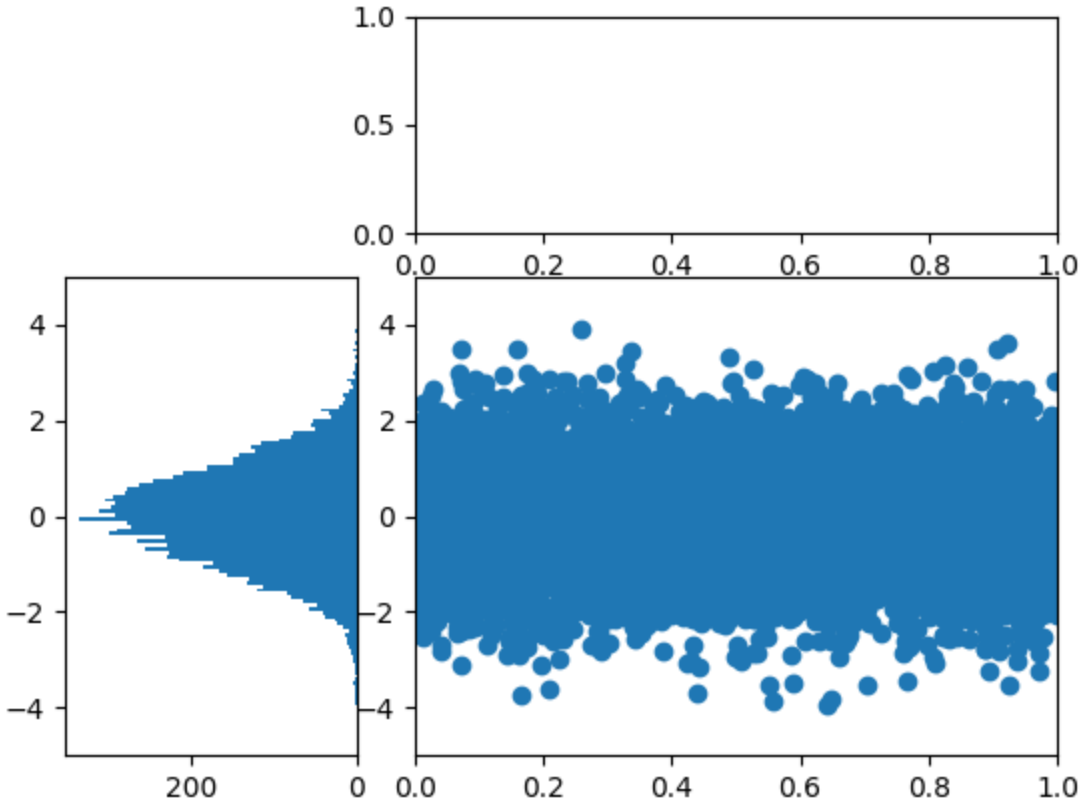


Out[9]: <matplotlib.collections.PathCollection at 0x7fa183a068b0>

10 - Exemplo com o GridSpec

In [10]:

```
# gridspec permite particionar a figura em subplots,
# de formas mais espertas

import matplotlib.gridspec as gridspec

plt.figure()
gspec = gridspec.GridSpec(3, 3)

top_histogram = plt.subplot(gspec[0, 1:])
side_histogram = plt.subplot(gspec[1:, 0])
lower_right = plt.subplot(gspec[1:, 1:])
```



11 - Adicionando dados aos subplots

In [11]:

```
lower_right.scatter(X,Y)
top_histogram.hist(X, bins=100)
side_histogram.hist(Y, bins=100, orientation='horizontal');
```

12 - Funções clear(), hist() e invert_xaxis()

In [12]:

```
# limpar os histogramas e criar histogramas normalizados
top_histogram.clear()
# virar o eixo x do histograma lateral
side_histogram.invert_xaxis()
```

13 - Funções set_xlim() e set_ylim()

In [13]:

```
# mudar limites dos eixos
for ax in [top_histogram, lower_right]:
    ax.set_xlim(0, 1)
for ax in [side_histogram, lower_right]:
    ax.set_ylim(-5, 5)
```

Plotagem de Caixas - Box Plot

14 - Criar 3 amostragens distintas

In [14]:

```
import pandas as pd
normal_sample = np.random.normal(loc=0.0, scale=1.0, size=10000)
random_sample = np.random.random(size=10000)
```

```
gamma_sample = np.random.gamma(2, size=10000)

df = pd.DataFrame({'normal': normal_sample,
                   'random': random_sample,
                   'gamma': gamma_sample})
```

15 - Função describe() do pandas

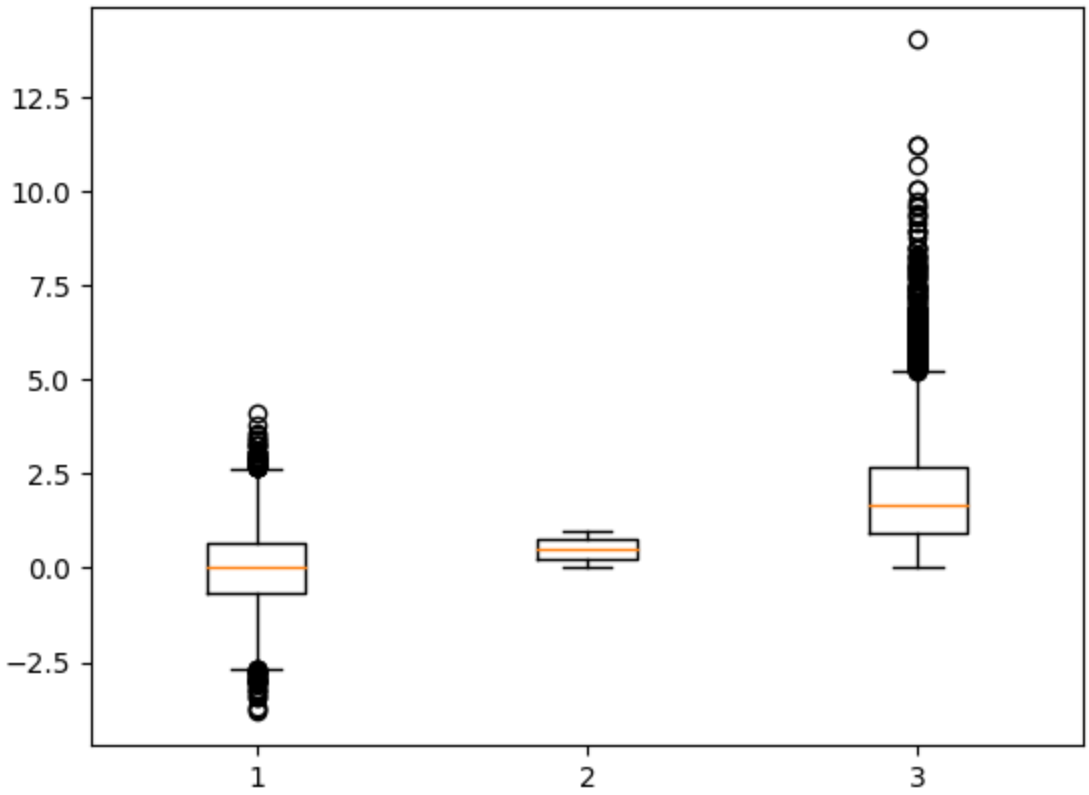
In [15]:
```
df.describe()
```

Out[15]:

	normal	random	gamma
count	10000.000000	10000.000000	10000.000000
mean	0.000269	0.504667	1.980618
std	0.997726	0.288264	1.403966
min	-3.823549	0.000068	0.006168
25%	-0.672595	0.256254	0.945860
50%	0.001535	0.500068	1.658335
75%	0.654593	0.759523	2.668764
max	4.112338	0.999972	14.038100

16 - Primeiro Boxplot

In [16]:
```
plt.figure()
# cria um boxplot dos dados normais, atribui a saída a uma variável para suprimir a saída
plt.boxplot(df['normal'], whis='range');
```




```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/01/_r7b02r11p15j0s54gb9x0040000gn/T/ipykernel_3739/2632892574.py in <module>
      1 plt.figure()
      2 # cria um boxplot dos dados normais, atribui a saída a uma variável para suprimir
 a saída
----> 3 plt.boxplot(df['normal'], whis='range');

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/pyplot.py in boxplot
(x, notch, sym, vert, whis, positions, widths, patch_artist, bootstrap, usermedians, conf_
intervals, meanline, showmeans, showcaps, showbox, showfliers, boxprops, labels, flierprop
s, medianprops, meanprops, capprops, whiskerprops, manage_ticks, autorange, zorder, data)
   2689         whiskerprops=None, manage_ticks=True, autorange=False,
   2690         zorder=None, *, data=None):
-> 2691     return gca().boxplot(
   2692         x, notch=notch, sym=sym, vert=vert, whis=whis,
   2693         positions=positions, widths=widths, patch_artist=patch_artist,

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/__init__.py in inner(a
x, data, *args, **kwargs)
   1359     def inner(ax, *args, data=None, **kwargs):
   1360         if data is None:
-> 1361             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1362
   1363         bound = new_sig.bind(ax, *args, **kwargs)

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/axes/_axes.py in boxpl
ot(self, x, notch, sym, vert, whis, positions, widths, patch_artist, bootstrap, usermedian
s, conf_intervals, meanline, showmeans, showcaps, showbox, showfliers, boxprops, labels, f
lierprops, medianprops, meanprops, capprops, whiskerprops, manage_ticks, autorange, zorde
r)
   3743             bootstrap = rcParams['boxplot.bootstrap']
   3744
-> 3745         bxpstats = cbook.boxplot_stats(x, whis=whis, bootstrap=bootstrap,
   3746                                        labels=labels, autorange=autorange)
```

```
   3747         if notch is None:

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/cbook/__init__.py in b
oxplot_stats(X, whis, bootstrap, labels, autorange)
   1250             hival = q3 + whis * stats['iqr']
   1251         else:
-> 1252             raise ValueError('whis must be a float or list of percentiles')
   1253
   1254         # get high extreme

ValueError: whis must be a float or list of percentiles
```

17 - Segundo Boxplot

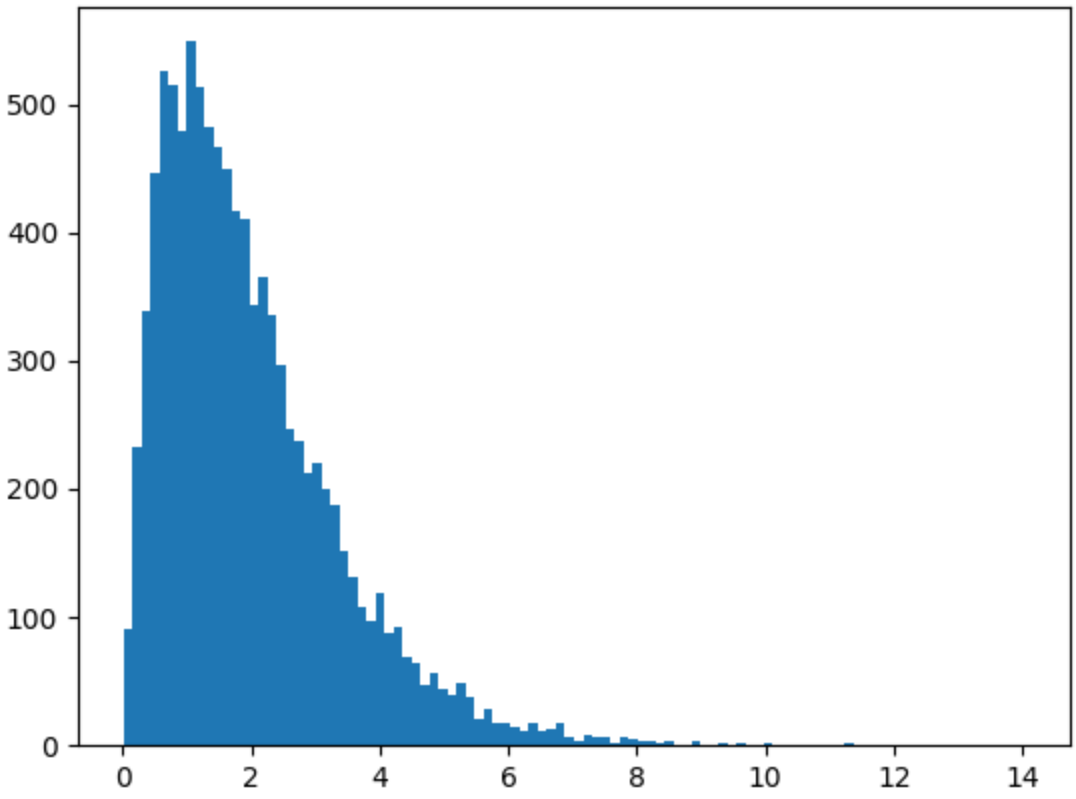
In [18]:

```
# limpa a figura corrente
plt.clf()
# plota caixas para três tipos de distribuição
plt.boxplot([df['normal'], df['random'], df['gamma']]);
```

18 - Histograma da distribuição gamma

In [19]:

```
plt.figure()
plt.hist(df['gamma'], bins=100)
```



Out[19]:

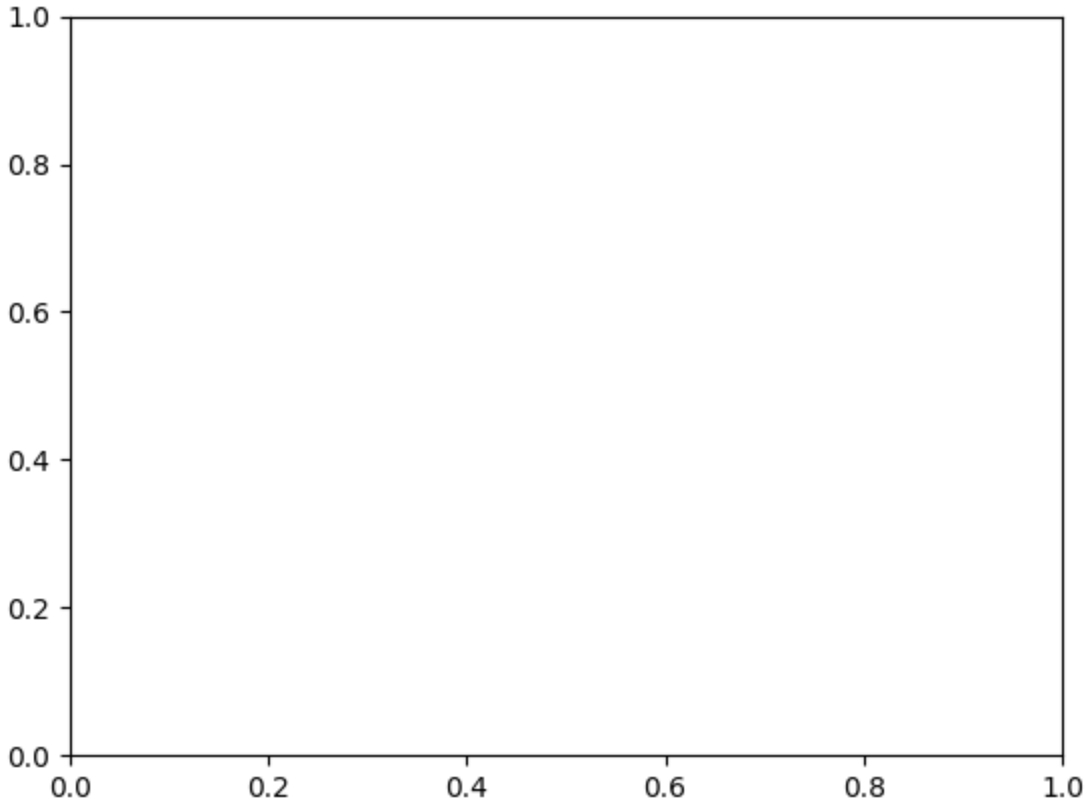
```
(array([ 90., 232., 339., 447., 526., 515., 479., 549., 514., 482., 467.,
        449., 417., 410., 343., 366., 336., 297., 247., 238., 213., 220.,
        200., 187., 151., 132., 108.,  97., 119.,  87.,  93.,  69.,  64.,
         47.,  56.,  44.,  40.,  48.,  38.,  20.,  29.,  17.,  18.,  15.,
         12.,  17.,  12.,  13.,  18.,   6.,   3.,   8.,   6.,   6.,   2.,
          6.,   5.,   4.,   4.,   2.,   4.,   0.,   1.,   3.,   0.,   1.,
          2.,   1.,   2.,   1.,   0.,   2.,   0.,   0.,   0.,   0.,   1.,
          0.,   0.,   0.,   2.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
```

```
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          1.]),
 array([6.16769320e-03, 1.46487012e-01, 2.86806331e-01, 4.27125650e-01,
        5.67444968e-01, 7.07764287e-01, 8.48083606e-01, 9.88402925e-01,
        1.12872224e+00, 1.26904156e+00, 1.40936088e+00, 1.54968020e+00,
        1.68999952e+00, 1.83031884e+00, 1.97063816e+00, 2.11095748e+00,
        2.25127679e+00, 2.39159611e+00, 2.53191543e+00, 2.67223475e+00,
        2.81255407e+00, 2.95287339e+00, 3.09319271e+00, 3.23351203e+00,
        3.37383134e+00, 3.51415066e+00, 3.65446998e+00, 3.79478930e+00,
        3.93510862e+00, 4.07542794e+00, 4.21574726e+00, 4.35606658e+00,
        4.49638590e+00, 4.63670521e+00, 4.77702453e+00, 4.91734385e+00,
        5.05766317e+00, 5.19798249e+00, 5.33830181e+00, 5.47862113e+00,
        5.61894045e+00, 5.75925976e+00, 5.89957908e+00, 6.03989840e+00,
        6.18021772e+00, 6.32053704e+00, 6.46085636e+00, 6.60117568e+00,
        6.74149500e+00, 6.88181432e+00, 7.02213363e+00, 7.16245295e+00,
        7.30277227e+00, 7.44309159e+00, 7.58341091e+00, 7.72373023e+00,
        7.86404955e+00, 8.00436887e+00, 8.14468818e+00, 8.28500750e+00,
        8.42532682e+00, 8.56564614e+00, 8.70596546e+00, 8.84628478e+00,
        8.98660410e+00, 9.12692342e+00, 9.26724274e+00, 9.40756205e+00,
        9.54788137e+00, 9.68820069e+00, 9.82852001e+00, 9.96883933e+00,
        1.01091586e+01, 1.02494780e+01, 1.03897973e+01, 1.05301166e+01,
        1.06704359e+01, 1.08107552e+01, 1.09510746e+01, 1.10913939e+01,
        1.12317132e+01, 1.13720325e+01, 1.15123518e+01, 1.16526712e+01,
        1.17929905e+01, 1.19333098e+01, 1.20736291e+01, 1.22139484e+01,
        1.23542677e+01, 1.24945871e+01, 1.26349064e+01, 1.27752257e+01,
        1.29155450e+01, 1.30558643e+01, 1.31961837e+01, 1.33365030e+01,
        1.34768223e+01, 1.36171416e+01, 1.37574609e+01, 1.38977803e+01,
        1.40380996e+01]),
 <BarContainer object of 100 artists>)
```

19 - Inset Plot usando toolkits

In [20]:

```
import mpl_toolkits.axes_grid1.inset_locator as mpl_il

plt.figure()
plt.boxplot([ df['normal'], df['random'], df['gamma'] ], whis='range')
# eixo de sobreposição
ax2 = mpl_il.inset_axes(plt.gca(), width='60%', height='40%', loc=2)
ax2.hist(df['gamma'], bins=100)
ax2.margins(x=0.5)
```




```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/01/_r7b02r11p15j0s54gb9x0040000gn/T/ipykernel_3739/3207680226.py in <module>
      2
      3 plt.figure()
----> 4 plt.boxplot([ df['normal'], df['random'], df['gamma'] ], whis='range')
      5 # eixo de sobreposição
      6 ax2 = mpl_il.inset_axes(plt.gca(), width='60%', height='40%', loc=2)

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/pyplot.py in boxplot
(x, notch, sym, vert, whis, positions, widths, patch_artist, bootstrap, usermedians, conf_
intervals, meanline, showmeans, showcaps, showbox, showfliers, boxprops, labels, flierprop
s, medianprops, meanprops, capprops, whiskerprops, manage_ticks, autorange, zorder, data)
   2689         whiskerprops=None, manage_ticks=True, autorange=False,
   2690         zorder=None, *, data=None):
-> 2691     return gca().boxplot(
   2692         x, notch=notch, sym=sym, vert=vert, whis=whis,
   2693         positions=positions, widths=widths, patch_artist=patch_artist,

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/__init__.py in inner(a
x, data, *args, **kwargs)
   1359     def inner(ax, *args, data=None, **kwargs):
   1360         if data is None:
-> 1361             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1362
   1363         bound = new_sig.bind(ax, *args, **kwargs)

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/axes/_axes.py in boxpl
ot(self, x, notch, sym, vert, whis, positions, widths, patch_artist, bootstrap, usermedian
s, conf_intervals, meanline, showmeans, showcaps, showbox, showfliers, boxprops, labels, f
lierprops, medianprops, meanprops, capprops, whiskerprops, manage_ticks, autorange, zorde
r)
   3743             bootstrap = rcParams['boxplot.bootstrap']
   3744
-> 3745         bxpstats = cbook.boxplot_stats(x, whis=whis, bootstrap=bootstrap,
```

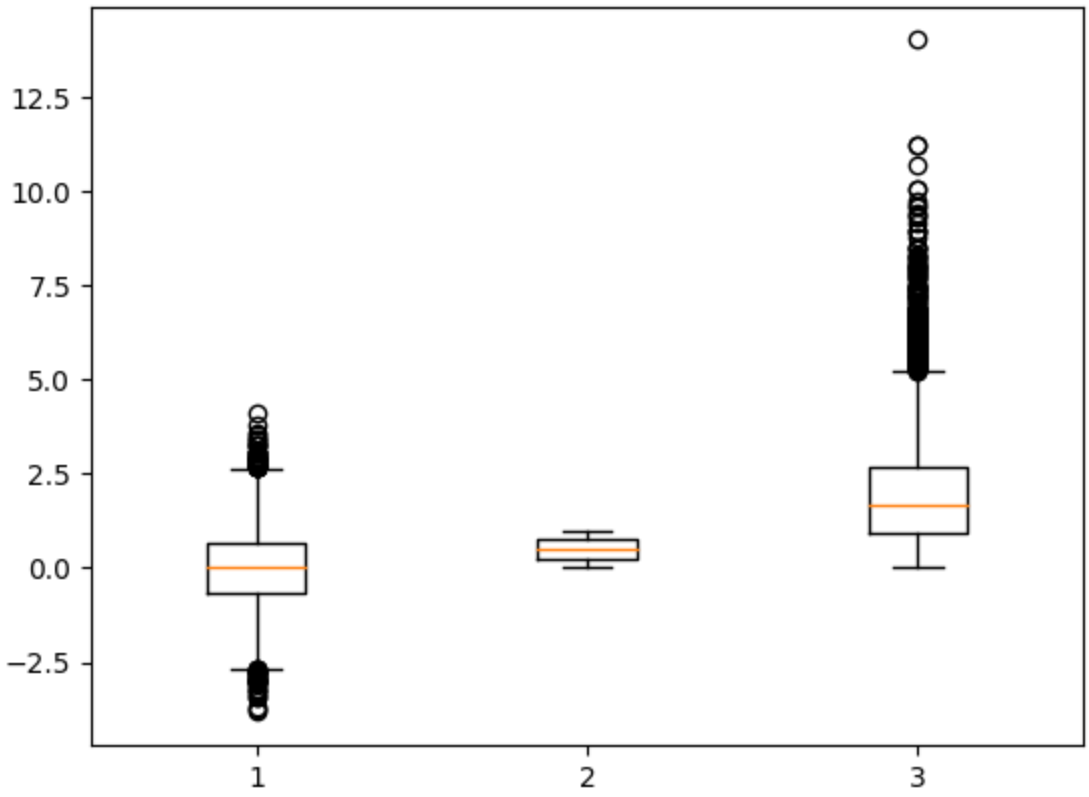
```
   3746                                            labels=labels, autorange=autorange)
   3747         if notch is None:

~/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/matplotlib/cbook/__init__.py in b
oxplot_stats(X, whis, bootstrap, labels, autorange)
   1250             hival = q3 + whis * stats['iqr']
   1251         else:
-> 1252             raise ValueError('whis must be a float or list of percentiles')
   1253
   1254         # get high extreme

ValueError: whis must be a float or list of percentiles
```

20 - Troca o lado dos ticks para a figura inset

In [21]:
```
# marcações do eixo y para ax2 - lado direito
ax2.yaxis.tick_right()
```

21 - Detectando outliers

In [22]:
```
# se o argumento `whis` não for passado, o boxplot é padronizado para mostrar bigodes inte
plt.figure()
_ = plt.boxplot([ df['normal'], df['random'], df['gamma'] ] )
```

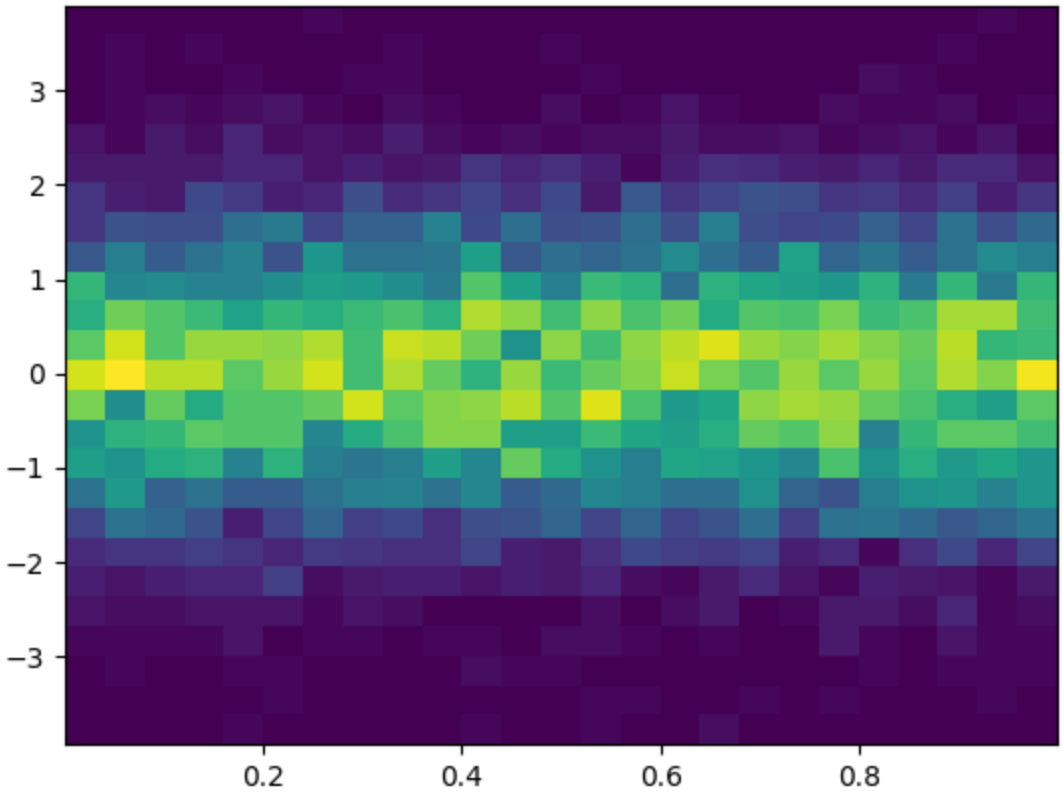


Mapas de calor

22 - Primeiro mapa de calor

In [23]:
```
plt.figure()
```

```
plt.hist2d(X,Y, bins=25)
```



Out[23]:

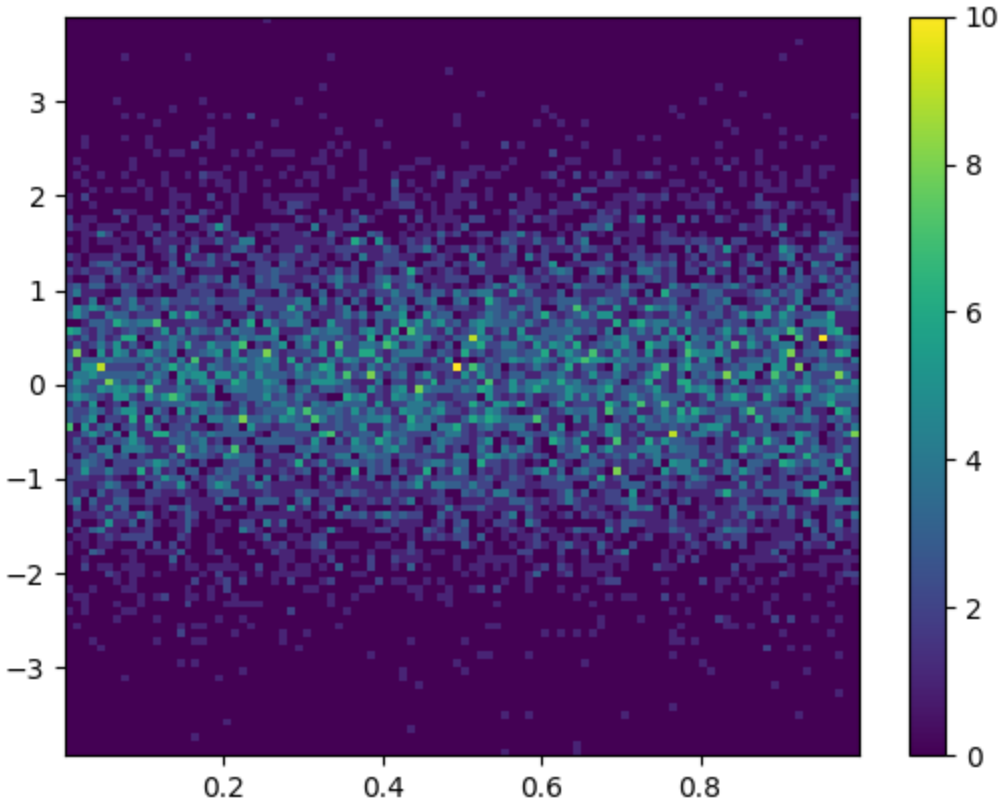
```
(array([[ 0.,  0.,  0.,  1.,  3.,  5.,  7., 12., 22., 33., 30., 47., 55.,
         44., 37., 39., 16.,  9.,  9.,  4.,  3.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  1.,  1.,  2.,  3.,  9., 22., 32., 30., 38., 29., 59.,
         55., 46., 27., 25., 15.,  5.,  4.,  1.,  1.,  1.,  1.,  0.],
        [ 0.,  0.,  0.,  1.,  2.,  5.,  9., 20., 18., 36., 39., 45., 53.,
         43., 43., 28., 17., 14.,  4.,  4.,  4.,  2.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  1.,  3.,  6., 11., 15., 22., 38., 44., 36., 53.,
         50., 40., 26., 22., 14., 13.,  4.,  2.,  1.,  0.,  1.,  0.],
        [ 1.,  0.,  1.,  3.,  3.,  6.,  9.,  5., 17., 26., 43., 43., 44.,
         50., 34., 26., 26., 21., 10.,  6.,  6.,  2.,  1.,  0.,  0.],
        [ 0.,  1.,  1.,  0.,  3., 11.,  6., 12., 17., 38., 43., 43., 50.,
         49., 39., 29., 15., 24.,  5.,  6.,  2.,  3.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  1.,  1.,  2., 10., 19., 22., 25., 27., 45., 55.,
         52., 37., 33., 31., 12.,  6.,  3.,  3.,  1.,  0.,  0.,  1.],
        [ 0.,  0.,  0.,  1.,  3.,  4.,  9., 11., 25., 23., 36., 55., 41.,
         41., 40., 32., 22., 18., 14.,  5.,  2.,  0.,  1.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  2.,  5.,  8., 13., 26., 25., 42., 44., 52.,
         54., 42., 29., 22., 18.,  7.,  3.,  5.,  2.,  1.,  1.,  0.],
        [ 0.,  0.,  0.,  1.,  0.,  5.,  8.,  8., 22., 33., 48., 48., 45.,
         53., 38., 24., 23., 26.,  9.,  4.,  2.,  1.,  0.,  0.,  0.],
        [ 1.,  0.,  2.,  1.,  0.,  3., 12., 14., 27., 27., 48., 49., 38.,
         46., 52., 43., 33., 13., 12.,  9.,  1.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  1.,  0.,  0.,  5.,  5., 15., 17., 45., 33., 53., 50.,
         30., 49., 33., 16., 21.,  8.,  6.,  2.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  1.,  2.,  0.,  4.,  4., 19., 20., 36., 33., 43., 40.,
         49., 41., 25., 21., 14., 13.,  8.,  1.,  2.,  0.,  1.,  0.],
        [ 1.,  1.,  0.,  2.,  2.,  6.,  8., 12., 27., 30., 40., 56., 45.,
         41., 49., 40., 19., 15.,  4.,  5.,  2.,  0.,  1.,  0.,  0.],
        [ 0.,  1.,  0.,  1.,  0.,  2., 13., 19., 25., 25., 35., 42., 48.,
         49., 42., 38., 22., 21., 16.,  1.,  2.,  1.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  2.,  1., 11., 12., 21., 35., 33., 32., 54.,
```

```
        53., 46., 21., 28., 14.,  9.,  5.,  4.,  3.,  0.,  0.,  0.],
       [ 2.,  0.,  0.,  1.,  4.,  4., 10., 15., 21., 34., 37., 35., 47.,
        56., 36., 38., 21., 25., 12.,  8.,  2.,  1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  7., 12., 21., 30., 31., 45., 49., 43.,
        50., 43., 35., 17., 14., 15.,  7.,  2.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  3.,  5., 11., 19., 27., 43., 51., 50.,
        48., 42., 33., 34., 12., 14.,  5.,  3.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  4.,  4.,  1.,  7., 22., 15., 42., 49., 50., 44.,
        51., 47., 31., 19., 13.,  9.,  4.,  1.,  2.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  1.,  4.,  5.,  1., 23., 25., 30., 26., 45., 50.,
        48., 40., 38., 23., 18., 10.,  6.,  2.,  1.,  2.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  2.,  4.,  8., 20., 30., 37., 39., 42., 44.,
        45., 42., 24., 17., 12.,  7.,  4.,  3.,  1.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  3.,  6.,  3., 13., 16., 31., 32., 44., 37., 52.,
        53., 51., 39., 22., 22., 11.,  7.,  1.,  2.,  0.,  1.,  0.],
       [ 0.,  1.,  1.,  1.,  1.,  1.,  6., 19., 26., 35., 44., 33., 48.,
        39., 51., 24., 28., 14.,  5.,  7.,  3.,  0.,  0.,  0.,  1.],
       [ 0.,  0.,  1.,  1.,  2.,  4., 12., 23., 31., 31., 41., 44., 58.,
        40., 41., 39., 25., 20.,  9.,  3.,  0.,  1.,  0.,  0.,  0.]]),
array([2.94503878e-05, 4.00177951e-02, 8.00061399e-02, 1.19994485e-01,
       1.59982829e-01, 1.99971174e-01, 2.39959519e-01, 2.79947864e-01,
       3.19936208e-01, 3.59924553e-01, 3.99912898e-01, 4.39901242e-01,
       4.79889587e-01, 5.19877932e-01, 5.59866277e-01, 5.99854621e-01,
       6.39842966e-01, 6.79831311e-01, 7.19819656e-01, 7.59808000e-01,
       7.99796345e-01, 8.39784690e-01, 8.79773035e-01, 9.19761379e-01,
       9.59749724e-01, 9.99738069e-01]),
array([-3.93019456, -3.61676786, -3.30334117, -2.98991448, -2.67648779,
       -2.36306109, -2.0496344 , -1.73620771, -1.42278101, -1.10935432,
       -0.79592763, -0.48250093, -0.16907424,  0.14435245,  0.45777915,
        0.77120584,  1.08463253,  1.39805923,  1.71148592,  2.02491261,
        2.3383393 ,  2.651766  ,  2.96519269,  3.27861938,  3.59204608,
        3.90547277]),
<matplotlib.collections.QuadMesh at 0x7fa185f68df0>)
```

23 - Aumentando o número de bins no mapa de calor

In [24]:

```
plt.figure()

plt.hist2d(X,Y, bins=100)
```



Out[24]:
```
(array([[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.]]),
 array([2.94503878e-05, 1.00265366e-02, 2.00236228e-02, 3.00207089e-02,
        4.00177951e-02, 5.00148813e-02, 6.00119675e-02, 7.00090537e-02,
        8.00061399e-02, 9.00032260e-02, 1.00000312e-01, 1.09997398e-01,
        1.19994485e-01, 1.29991571e-01, 1.39988657e-01, 1.49985743e-01,
        1.59982829e-01, 1.69979916e-01, 1.79977002e-01, 1.89974088e-01,
        1.99971174e-01, 2.09968260e-01, 2.19965346e-01, 2.29962433e-01,
        2.39959519e-01, 2.49956605e-01, 2.59953691e-01, 2.69950777e-01,
        2.79947864e-01, 2.89944950e-01, 2.99942036e-01, 3.09939122e-01,
        3.19936208e-01, 3.29933294e-01, 3.39930381e-01, 3.49927467e-01,
        3.59924553e-01, 3.69921639e-01, 3.79918725e-01, 3.89915812e-01,
        3.99912898e-01, 4.09909984e-01, 4.19907070e-01, 4.29904156e-01,
        4.39901242e-01, 4.49898329e-01, 4.59895415e-01, 4.69892501e-01,
        4.79889587e-01, 4.89886673e-01, 4.99883760e-01, 5.09880846e-01,
        5.19877932e-01, 5.29875018e-01, 5.39872104e-01, 5.49869191e-01,
        5.59866277e-01, 5.69863363e-01, 5.79860449e-01, 5.89857535e-01,
        5.99854621e-01, 6.09851708e-01, 6.19848794e-01, 6.29845880e-01,
        6.39842966e-01, 6.49840052e-01, 6.59837139e-01, 6.69834225e-01,
        6.79831311e-01, 6.89828397e-01, 6.99825483e-01, 7.09822569e-01,
        7.19819656e-01, 7.29816742e-01, 7.39813828e-01, 7.49810914e-01,
        7.59808000e-01, 7.69805087e-01, 7.79802173e-01, 7.89799259e-01,
        7.99796345e-01, 8.09793431e-01, 8.19790517e-01, 8.29787604e-01,
        8.39784690e-01, 8.49781776e-01, 8.59778862e-01, 8.69775948e-01,
        8.79773035e-01, 8.89770121e-01, 8.99767207e-01, 9.09764293e-01,
        9.19761379e-01, 9.29758466e-01, 9.39755552e-01, 9.49752638e-01,
        9.59749724e-01, 9.69746810e-01, 9.79743896e-01, 9.89740983e-01,
        9.99738069e-01]),
 array([-3.93019456, -3.85183788, -3.77348121, -3.69512454, -3.61676786,
        -3.53841119, -3.46005452, -3.38169784, -3.30334117, -3.2249845 ,
```

```
        -3.14662782, -3.06827115, -2.98991448, -2.9115578 , -2.83320113,
        -2.75484446, -2.67648779, -2.59813111, -2.51977444, -2.44141777,
        -2.36306109, -2.28470442, -2.20634775, -2.12799107, -2.0496344 ,
        -1.97127773, -1.89292105, -1.81456438, -1.73620771, -1.65785103,
        -1.57949436, -1.50113769, -1.42278101, -1.34442434, -1.26606767,
        -1.18771099, -1.10935432, -1.03099765, -0.95264097, -0.8742843 ,
        -0.79592763, -0.71757095, -0.63921428, -0.56085761, -0.48250093,
        -0.40414426, -0.32578759, -0.24743091, -0.16907424, -0.09071757,
        -0.01236089,  0.06599578,  0.14435245,  0.22270913,  0.3010658 ,
         0.37942247,  0.45777915,  0.53613582,  0.61449249,  0.69284917,
         0.77120584,  0.84956251,  0.92791919,  1.00627586,  1.08463253,
         1.16298921,  1.24134588,  1.31970255,  1.39805923,  1.4764159 ,
         1.55477257,  1.63312925,  1.71148592,  1.78984259,  1.86819926,
         1.94655594,  2.02491261,  2.10326928,  2.18162596,  2.25998263,
         2.3383393 ,  2.41669598,  2.49505265,  2.57340932,  2.651766  ,
         2.73012267,  2.80847934,  2.88683602,  2.96519269,  3.04354936,
         3.12190604,  3.20026271,  3.27861938,  3.35697606,  3.43533273,
         3.5136894 ,  3.59204608,  3.67040275,  3.74875942,  3.8271161 ,
         3.90547277]),
 <matplotlib.collections.QuadMesh at 0x7fa185e8bf70>)
```

24 - Adicionando legenda colorbar()

In [25]:
```
# adiciona barra de cores
plt.colorbar()
```

Out[25]:
```
<matplotlib.colorbar.Colorbar at 0x7fa188202fd0>
```

Animações

25 - Preparação da animação

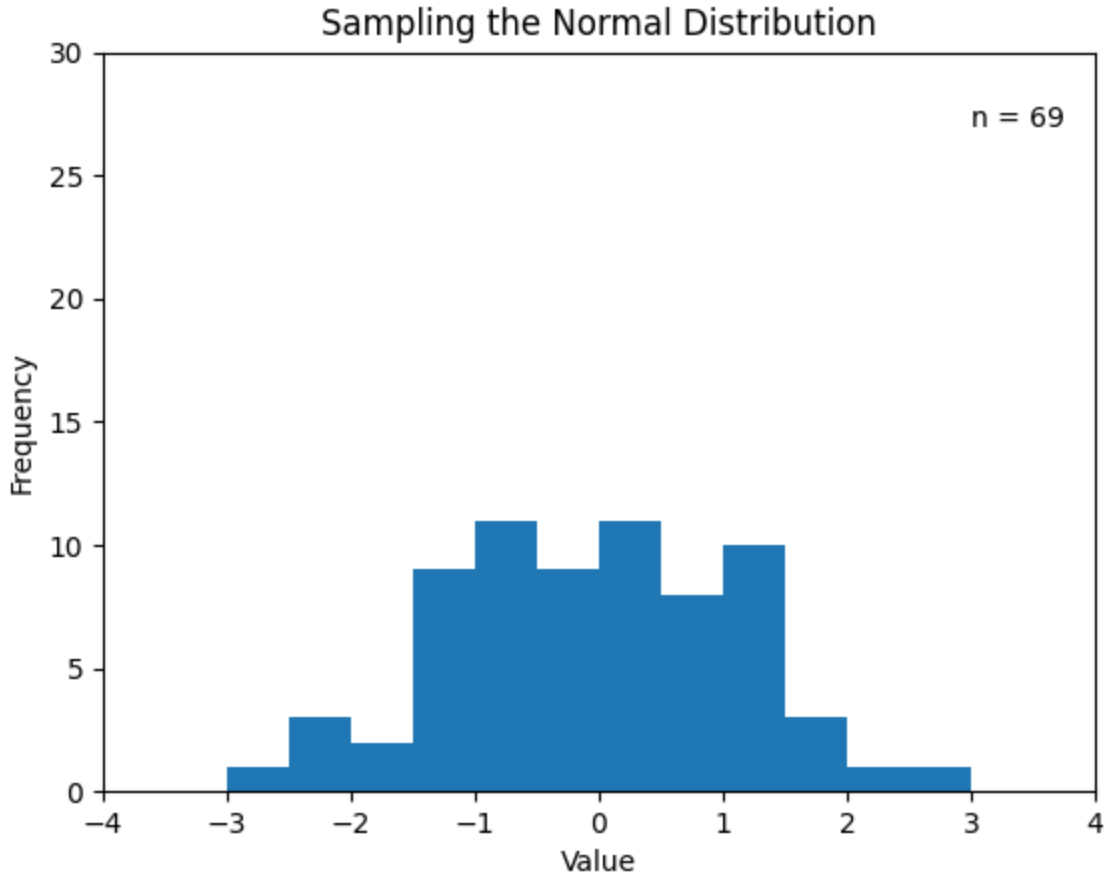
In [26]:
```
import matplotlib.animation as animation

n = 100
x = np.random.randn(n)
```

26 - Função para animar

In [27]:
```
# cria a função que fará a plotagem, onde "curr" é o quadro atual
def update(curr):
    # se a animação está no último quadro, pare a animação
    if curr == n:
        a.event_source.stop()
    plt.cla()
    bins = np.arange(-4, 4, 0.5)
    plt.hist(x[:curr], bins=bins)
    plt.axis([-4,4,0,30])
    plt.gca().set_title('Sampling the Normal Distribution')
    plt.gca().set_ylabel('Frequency')
    plt.gca().set_xlabel('Value')
    plt.annotate('n = {}'.format(curr), [3,27])
```

27 - Inicia a animação

In [28]:
```
fig = plt.figure()
a = animation.FuncAnimation(fig, update, interval=100)
```


Sampling the Normal Distribution
n = 69
Frequency
Value
30
25
20
15
10
5
0
−4
−3
−2
−1
0
1
2
3
4

In [ ]: