exercise12

April 20, 2020

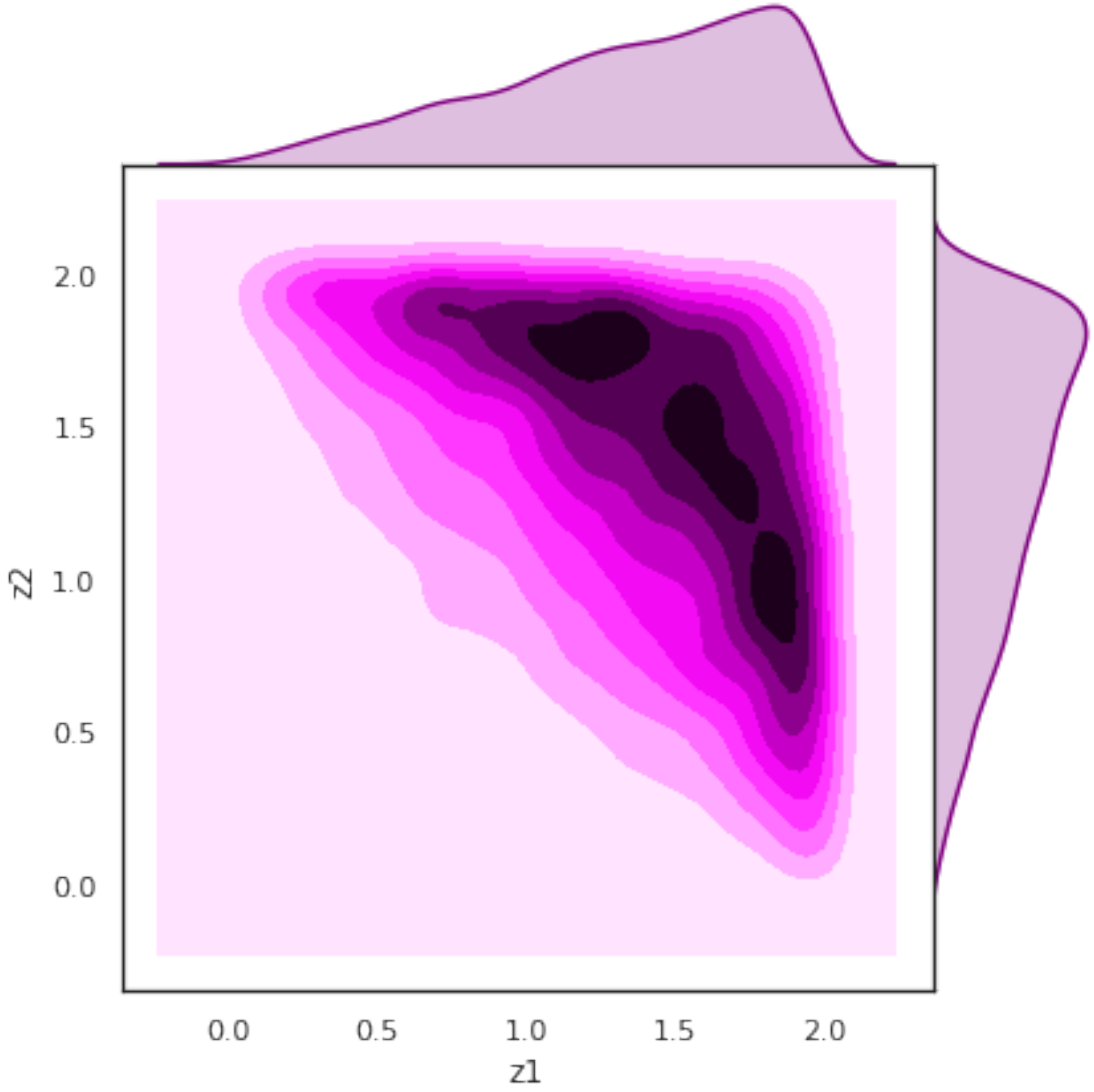
[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(style="white", color_codes=True)

from copy import copy

PI = np.pi

%matplotlib inline
```

[2]:
```
def empirical_cdf(x0, X):

    for i,x in enumerate(X):
        if x>x0:
            break

    return (i-1)/len(X)
```

[3]:
```
raw_data = pd.read_csv('dat.csv', sep=' ', names = ('z1', 'z2'))
Z1 = np.array(raw_data['z1'])
Z2 = np.array(raw_data['z2'])
```

[4]:
```
sns.jointplot("z1", "z2", data=raw_data, kind="kde", space=0, color="purple")
```

[4]:
```
<seaborn.axisgrid.JointGrid at 0x7fbab753ea90>
```



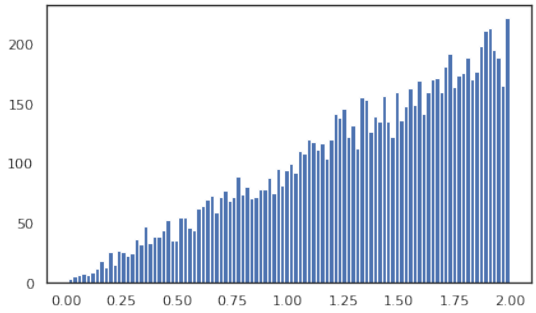
0.1 Brute force - sort and plot (quite good for small number of samples)

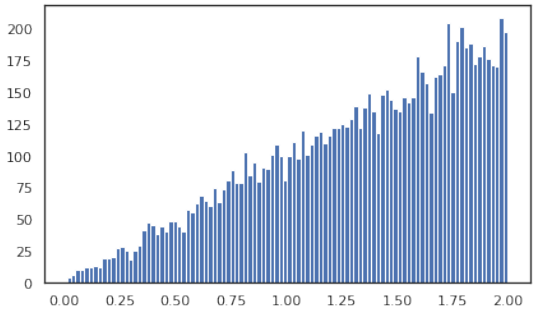
[5]:
```
Z1_sorted = copy(Z1)
Z2_sorted = copy(Z2)

Z1_sorted.sort()
Z2_sorted.sort()

CDF_Z1 = []
CDF_Z2 = []
```

```
for z1,z2 in zip(Z1,Z2):
    CDF_Z1.append(empirical_cdf(z1, Z1_sorted))
    CDF_Z2.append(empirical_cdf(z2, Z1_sorted))
```

[6]:
```
fig, axs = plt.subplots(ncols=2, figsize = (15,4))
axs[0].hist(Z1_sorted, bins = 100)
axs[1].hist(Z2_sorted, bins = 100)
```

[6]:
```
(array([  4.,   6.,  10.,  10.,  12.,  12.,  13.,  12.,  19.,  19.,  20.,
         28.,  29.,  26.,  18.,  26.,  30.,  42.,  48.,  46.,  39.,  45.,
         41.,  49.,  49.,  45.,  41.,  58.,  56.,  63.,  69.,  65.,  61.,
         75.,  64.,  74.,  81.,  89.,  79.,  79., 103.,  85.,  95.,  80.,
         91.,  90., 101., 109., 100.,  81., 100., 111.,  98., 120., 101.,
        109., 116., 119., 110., 116., 122., 122., 125., 123., 129., 140.,
        122., 139., 150., 136., 118., 149., 153., 145., 138., 136., 147.,
        143., 147., 179., 167., 158., 135., 163., 165., 172., 205., 151.,
        191., 202., 186., 189., 173., 179., 187., 177., 172., 171., 209.,
        198.]),
 array([0.01100008, 0.03088876, 0.05077743, 0.07066611, 0.09055478,
        0.11044346, 0.13033214, 0.15022081, 0.17010949, 0.18999816,
        0.20988684, 0.22977551, 0.24966419, 0.26955287, 0.28944154,
        0.30933022, 0.32921889, 0.34910757, 0.36899625, 0.38888492,
        0.4087736 , 0.42866227, 0.44855095, 0.46843963, 0.4883283 ,
        0.50821698, 0.52810565, 0.54799433, 0.567883  , 0.58777168,
        0.60766036, 0.62754903, 0.64743771, 0.66732638, 0.68721506,
        0.70710374, 0.72699241, 0.74688109, 0.76676976, 0.78665844,
        0.80654712, 0.82643579, 0.84632447, 0.86621314, 0.88610182,
        0.9059905 , 0.92587917, 0.94576785, 0.96565652, 0.9855452 ,
        1.00543387, 1.02532255, 1.04521123, 1.0650999 , 1.08498858,
        1.10487725, 1.12476593, 1.14465461, 1.16454328, 1.18443196,
        1.20432063, 1.22420931, 1.24409799, 1.26398666, 1.28387534,
        1.30376401, 1.32365269, 1.34354136, 1.36343004, 1.38331872,
        1.40320739, 1.42309607, 1.44298474, 1.46287342, 1.4827621 ,
        1.50265077, 1.52253945, 1.54242812, 1.5623168 , 1.58220548,
        1.60209415, 1.62198283, 1.6418715 , 1.66176018, 1.68164885,
        1.70153753, 1.72142621, 1.74131488, 1.76120356, 1.78109223,
        1.80098091, 1.82086959, 1.84075826, 1.86064694, 1.88053561,
        1.90042429, 1.92031297, 1.94020164, 1.96009032, 1.97997899,
        1.99986767]),
 <a list of 100 Patch objects>)
```

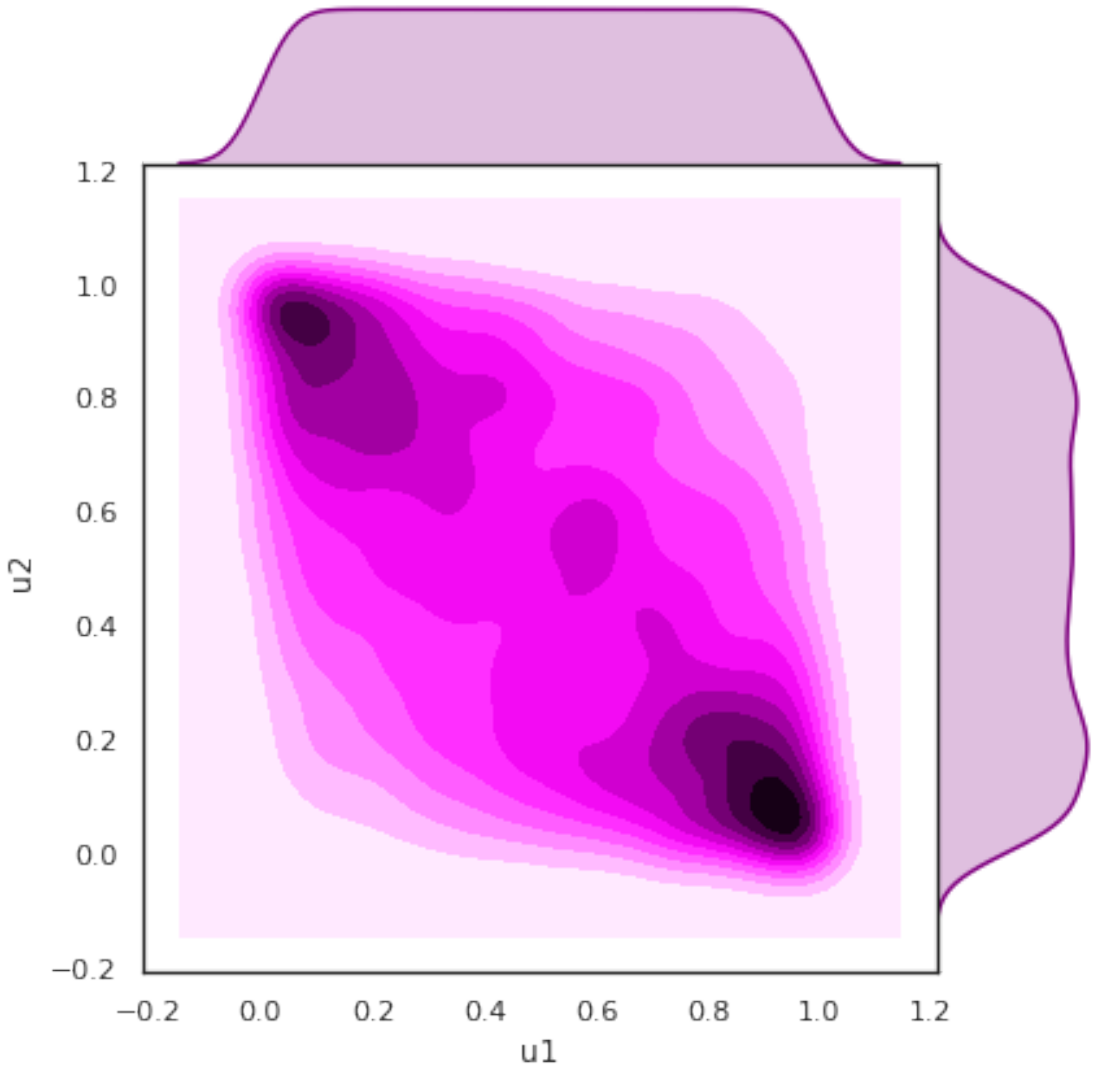




[7]:
```
uniform = np.array([CDF_Z1, CDF_Z2]).T
df_uniform = pd.DataFrame(uniform, columns = ['u1','u2'])
sns.jointplot("u1", "u2", data=df_uniform, kind="kde", space=0, color="purple")

df_uniform.corr()
```

[7]:

	u1	u2
u1	1.000000	-0.488999
u2	-0.488999	1.000000



0.2 Inverse Gauss CDF

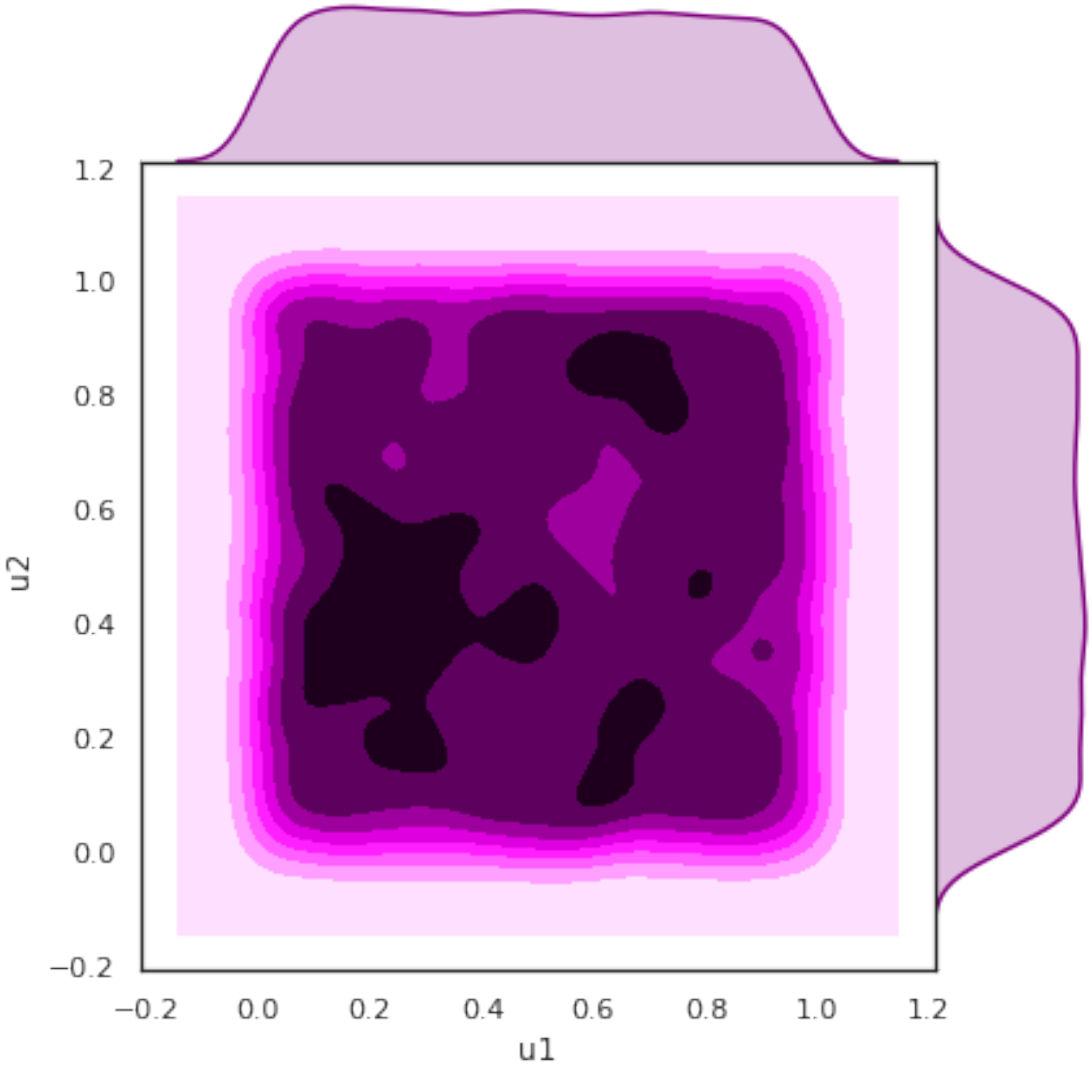
[8]:
```
from scipy.stats import norm
from scipy import special


## (two alternative functions)
def inverse_gauss(x):
    return  norm.ppf(x)
    #return -np.sqrt(2)* special.erfcinv(2*x)
```

0.3 Testing on non-correlated, 2D uniform distribution:

[9]:
```
XX = np.random.uniform(0,1,10000)
YY = np.random.uniform(0,1,10000)

uniform = np.array([XX, YY]).T
df_uniform = pd.DataFrame(uniform, columns = ['u1','u2'])
sns.jointplot("u1", "u2", data=df_uniform, kind="kde", space=0, color="purple")

df_uniform.corr()
```

[9]:

	u1	u2
u1	1.000000	0.008884
u2	0.008884	1.000000

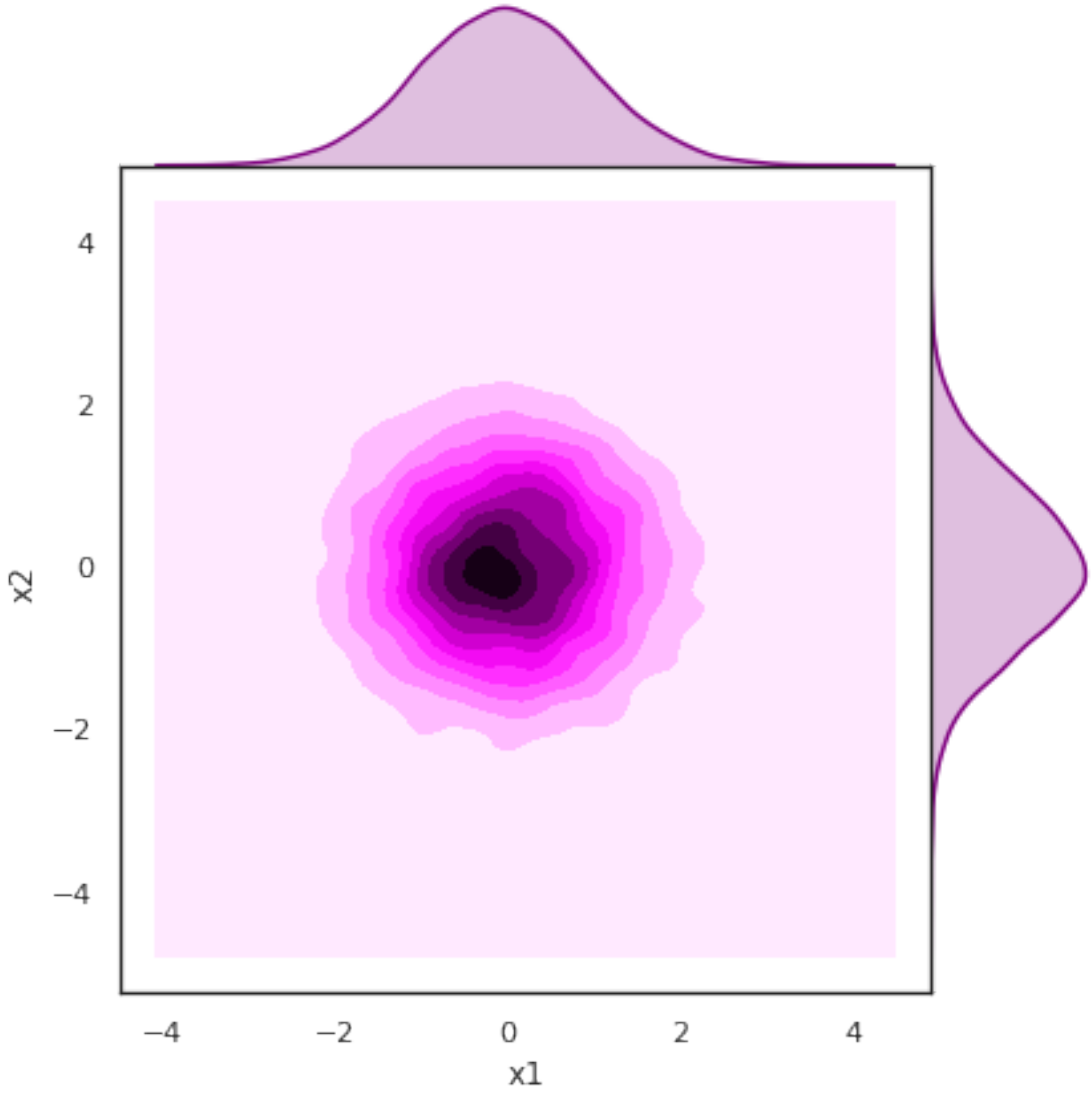


[10]:
```
XX1 = []
XX2 = []

for u1,u2 in zip(XX, YY):
    XX1.append(inverse_gauss(u1))
    XX2.append(inverse_gauss(u2))
```

[11]:
```
gauss = np.array([XX1, XX2]).T
gauss_cleared = gauss[~np.isinf(gauss).any(axis = 1)]

df_gauss = pd.DataFrame(gauss_cleared, columns = ['x1','x2'])
sns.jointplot("x1", "x2", data=df_gauss, kind="kde", space=0, color="purple")

df_gauss.corr()
```

[11]:

	x1	x2
x1	1.00000	0.00486
x2	0.00486	1.00000



0.4 Inverse Gauss for 'our' 2D correlated distribution

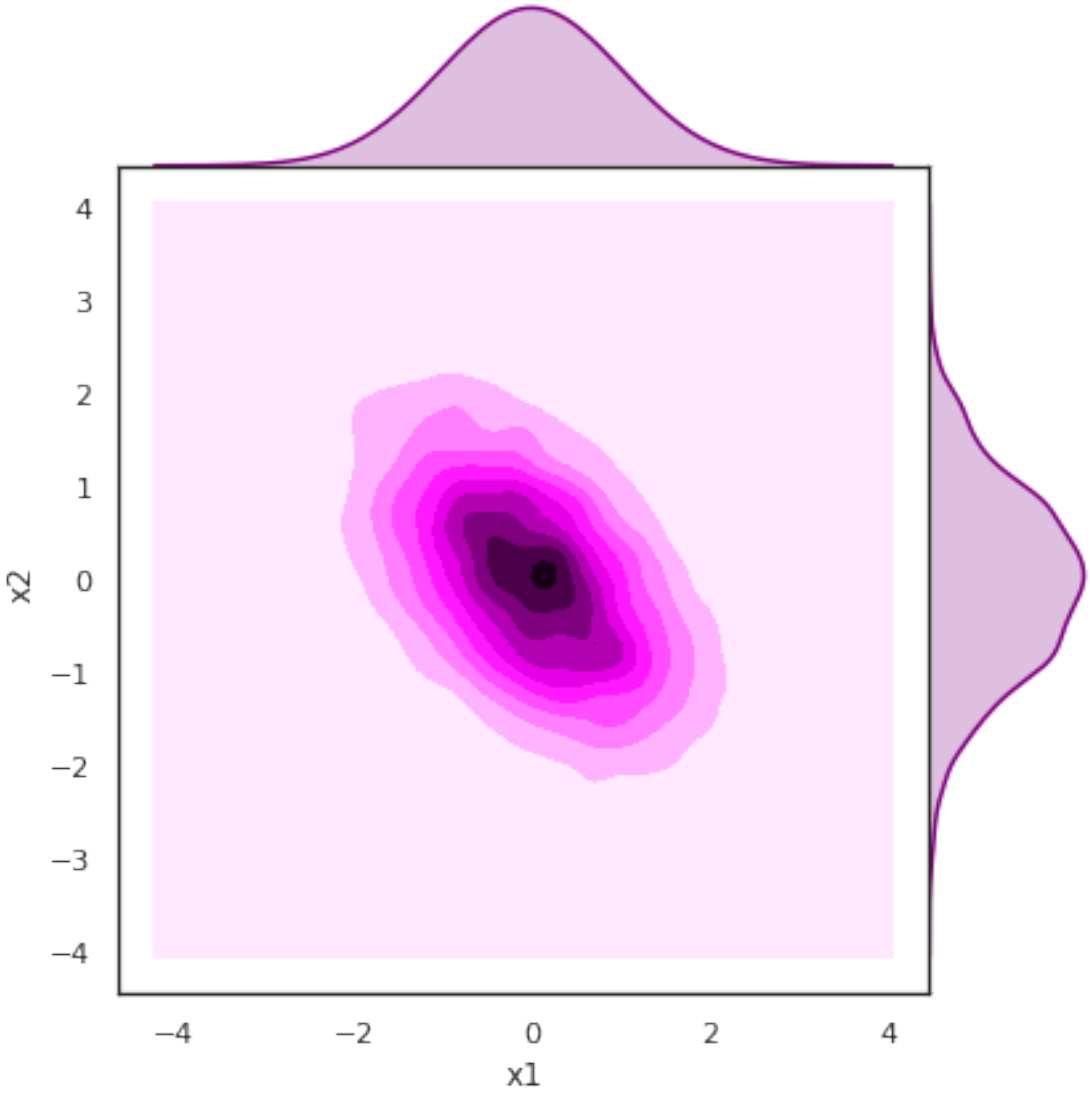
[12]:
```
X1 = []
X2 = []

for u1,u2 in zip(CDF_Z1, CDF_Z2):
    X1.append(inverse_gauss(u1))
    X2.append(inverse_gauss(u2))
```

[13]:
```
gauss = np.array([X1, X2]).T
gauss_cleared = gauss[~np.isinf(gauss).any(axis = 1)]
```

```
df_gauss = pd.DataFrame(gauss_cleared, columns = ['x1','x2'])
sns.jointplot("x1", "x2", data=df_gauss, kind="kde", space=0, color="purple")
df_gauss.corr()
```

[13]:

	x1	x2
x1	1.000000	-0.503214
x2	-0.503214	1.000000

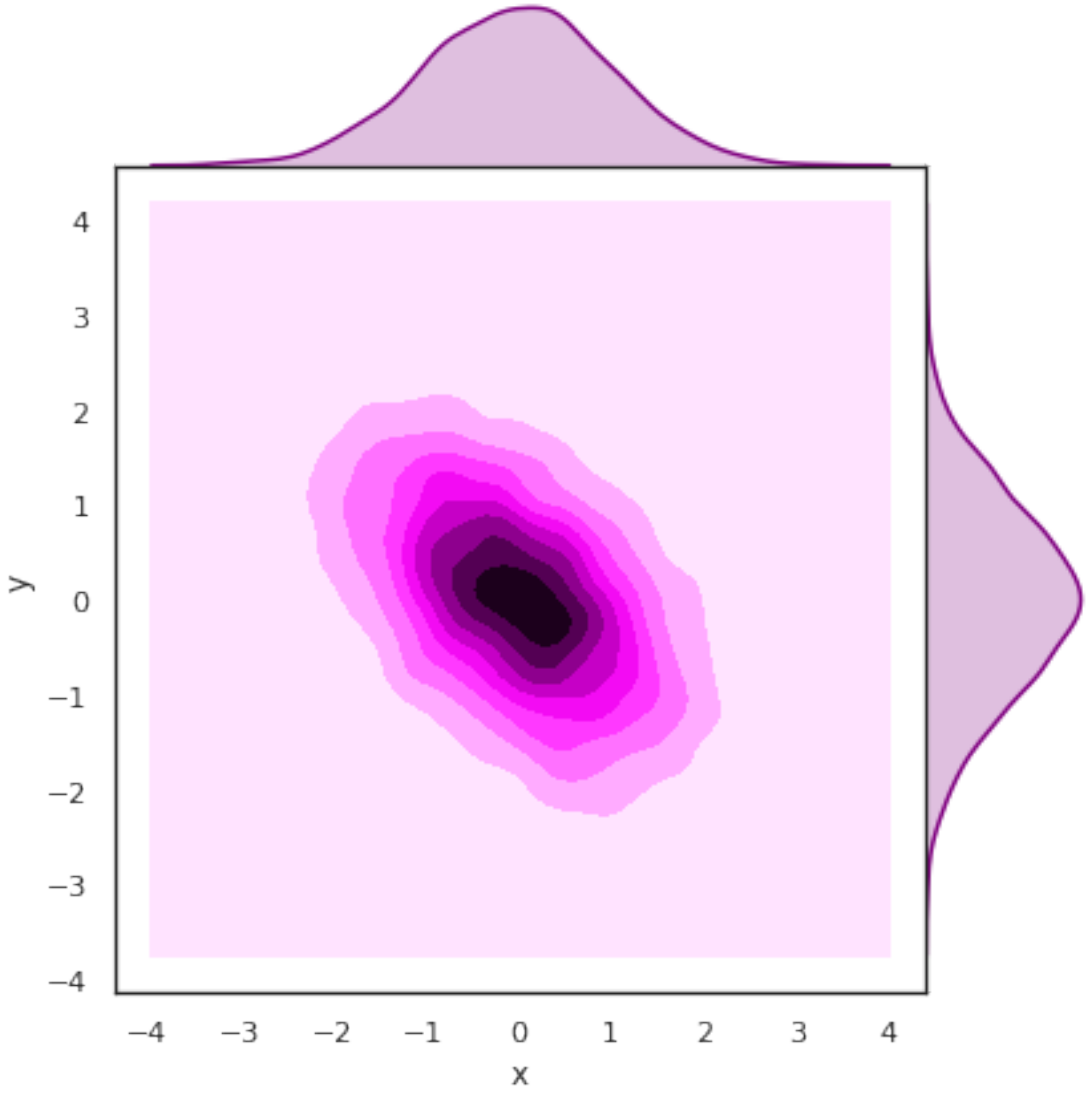


0.5 CHECKING THE SOLUTIONS

[14]:
```
sigma_x = 1;
sigma_y = 1;
rho = -0.488999

mean = [0, 0]
cov = [[sigma_x**2, sigma_x*sigma_y*rho], [sigma_x*sigma_y*rho, sigma_y**2]]  #␣
 ↪diagonal covariance
data = np.random.multivariate_normal(mean, cov, 5000)

df = pd.DataFrame(data, columns=['x','y'])
sns.jointplot("x", "y", data=df, kind="kde", space=0, color="purple")

df.corr()
```

[14]:

	x	y
x	1.000000	-0.485343
y	-0.485343	1.000000



[15]:
```
sigma_x = 1;
sigma_y = 1;
rho = -0.503214

mean = [0, 0]
cov = [[sigma_x**2, sigma_x*sigma_y*rho], [sigma_x*sigma_y*rho, sigma_y**2]]  #␣
 ↪diagonal covariance
data = np.random.multivariate_normal(mean, cov, 5000)

df = pd.DataFrame(data, columns=['x','y'])
sns.jointplot("x", "y", data=df, kind="kde", space=0, color="purple")

df.corr()
```

[15]:
```
          x        y
x  1.00000 -0.49489
y -0.49489  1.00000
```