Sheet 6 - Noise and Kurtosis

Team name: DataFun

Members:

Fabian Frank Jan Botsch David Munkacsi

In [1]:

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

6.1 Natural Gradient

In [2]:

```
#defines
samplerate = 8192
N = 3
p = 18000
```

Load sounds and create noise source

In [3]:

```
#load the data
dataSet1 = np.loadtxt('./sounds/sound1.dat')
dataSet2 = np.loadtxt('./sounds/sound2.dat')
s = np.stack([dataSet1, dataSet2], axis=0)

#create noise data sampled from a Gaussian
mean_data = np.mean(s,axis=1, keepdims=True)
std_data = np.std(s,axis=1, keepdims=True)
dataSet3a = np.random.normal(np.mean(mean_data), np.mean(std_data), p);

#create noise data sampled from a Laplacian
dataSet3b = np.random.laplace(0, 1, p);

#combine the data and the noise
s1 = np.vstack([s, dataSet3a])
s2 = np.vstack([s, dataSet3b])
```

Mix the sounds and Preprocess for ICA

In [4]:

```
# create random and invertible NxN (3x3) matrix
while True:
    A = np.random.rand(N,N)
    if np.linalg.det(A) != 0.0:
        break
print("A=" + str(A))

#mix the sources
x1 = np.matmul(A, s1)
x2 = np.matmul(A, s2)

# remove temporal structure by permutation
permutation = np.random.permutation(range(0,p))
x_per1 = x1[:, permutation]
x_per2 = x2[:, permutation]

#center the permuted data
mean1 = np.mean(x_per1,axis=1, keepdims=True)
x_per_cent1 = x_per1 - mean1;
mean2 = np.mean(x_per2,axis=1, keepdims=True)
x_per_cent2 = x_per2 - mean1;

#center the non-permuted data
x_cent1 = x1 - np.mean(x1,axis=1, keepdims=True)
x_cent2 = x2 - np.mean(x2,axis=1, keepdims=True)


#initialize W at random
while True:
    W_init = np.random.rand(N,N)
    if np.linalg.det(W_init) != 0.0:
        break
print("W_init=" + str(W_init))
```

```
A=[[ 0.89350613  0.88644697  0.46145254]
 [ 0.49579903  0.53295556  0.66126576]
 [ 0.12238335  0.09726446  0.8465295 ]]
W_init=[[ 0.14629045  0.62685782  0.52186495]
 [ 0.1435121   0.96301866  0.3998416 ]
 [ 0.19052239  0.57131675  0.2979037 ]]
```

Perform ICA with decaying conversion rate

In [5]:

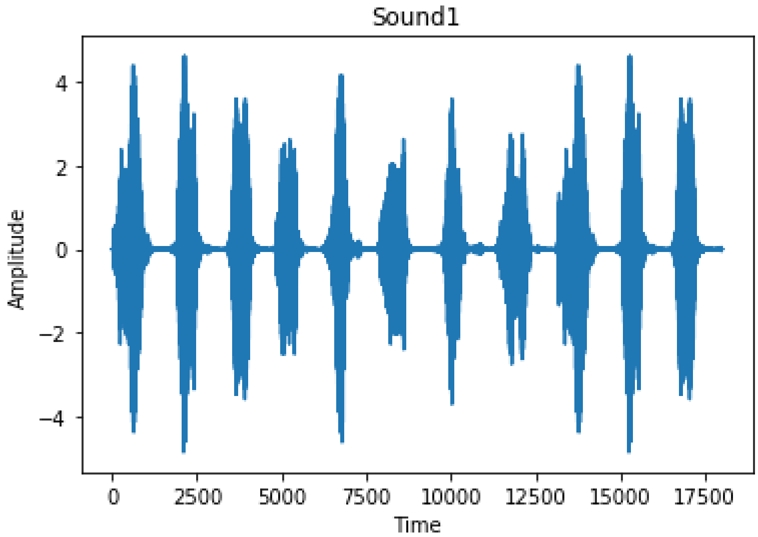
```
#function that calculates f''/f'
def stepSigmoid(y):
    return 1 - 2 * (1 / (1 + np.exp(-y)))

#vectorize function
vStepSigmoid = np.vectorize(stepSigmoid)

def perform_ica(x, W, eps=0.1, f_decay = 0.9995):
    #perform ica
    eps_curr = eps
    convSpeed = np.empty([0,2])
    #for i in range(0,p):
    i=0
    cnt = 0;
    while eps_curr > 0.0000001:
        #natural gradient
        unmixed = np.dot(W,x[:,i])
        dW = np.dot(np.eye(N) + np.outer(vStepSigmoid(unmixed),unmixed), W)
        W = W + eps_curr * dW
        eps_curr = eps_curr * f_decay
        i = (i+1) % p
        cnt = cnt + 1
    return (W, cnt)

W1, iter1 = perform_ica(x_per_cent1, W_init)
print("Iterations=" + str(iter1))
print("Gaussian: W final=" + str(W1))

W2, iter2 = perform_ica(x_per_cent2, W_init)
print("Iterations=" + str(iter2))
print("Laplacian: W final=" + str(W2))

unmixedGaussian = np.matmul(W1, x_cent1)
unmixedLaplacian = np.matmul(W2, x_cent2)
```

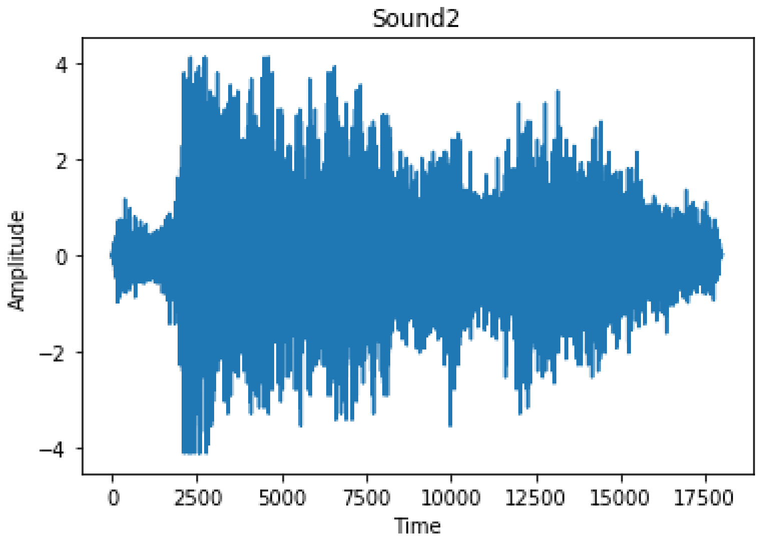
```
Iterations=27625
Gaussian: W final=[[ -0.73835998   0.84062938   1.81202471]
 [ 16.65247195 -34.34475022  17.69264899]
 [ 22.38123853 -40.83286919  19.72004894]]
Iterations=27625
Laplacian: W final=[[ 22.39376237 -40.8533382   19.71164721]
 [-16.61584616  34.28758402 -17.72417125]
 [  0.0849769    0.27659829  -1.85881284]]
```

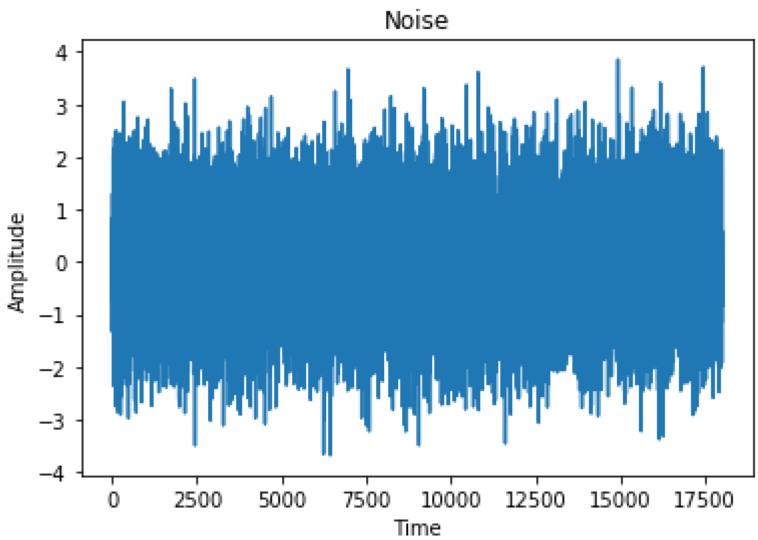
Plot original, mixed and unmixed for Gaussian

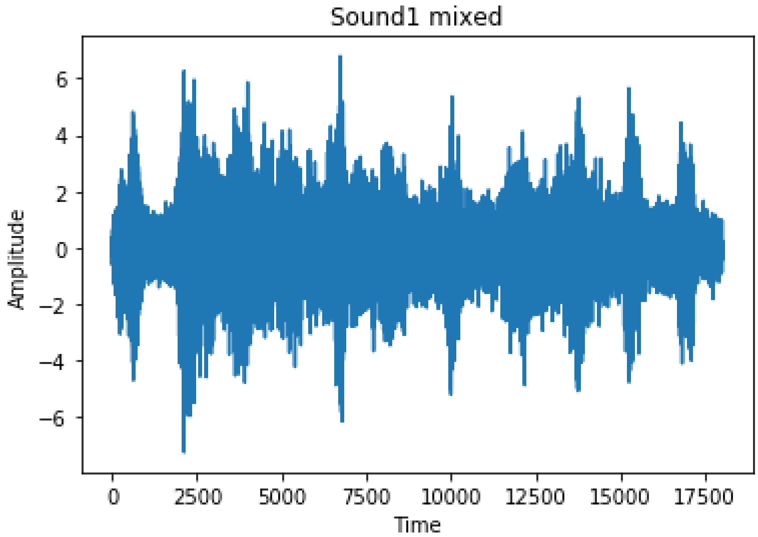
In [6]:

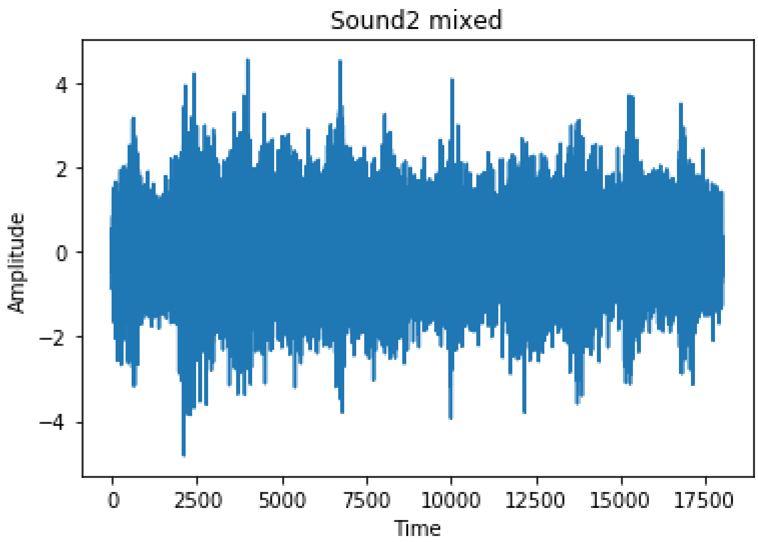
```
plt.figure()
plt.plot(dataSet1)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound1')
plt.figure()
plt.plot(dataSet2)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound2')
plt.figure()
plt.plot(dataSet3a)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Noise')

plt.figure()
plt.plot(x1[0,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound1 mixed')
plt.figure()
plt.plot(x1[1,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound2 mixed')
plt.figure()
plt.plot(x1[2,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Noise mixed')

plt.figure()
plt.plot(unmixedGaussian[0,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal 1 unmixed natural gradient')
plt.figure()
plt.plot(unmixedGaussian[1,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal 2 unmixed natural gradient')
plt.figure()
plt.plot(unmixedGaussian[2,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal 3 unmixed natural gradient')

plt.show()
```

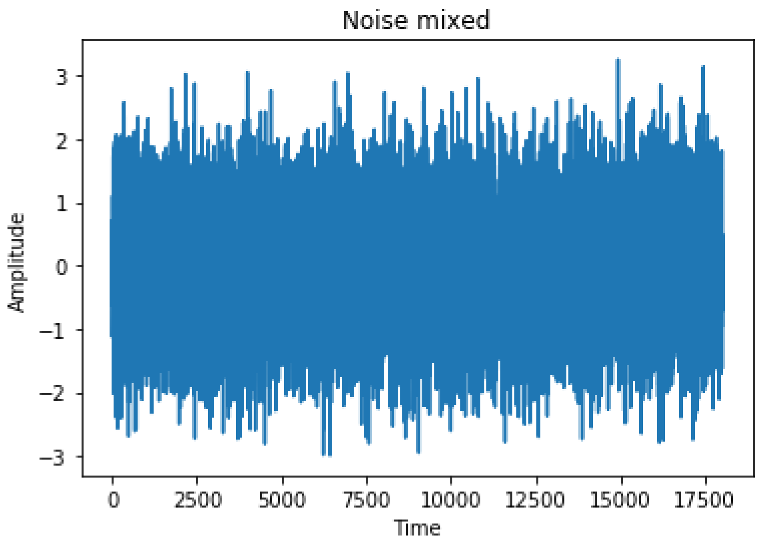

Sound1
Amplitude
Time
4
2
0
−2
−4
0
2500
5000
7500
10000
12500
15000
17500

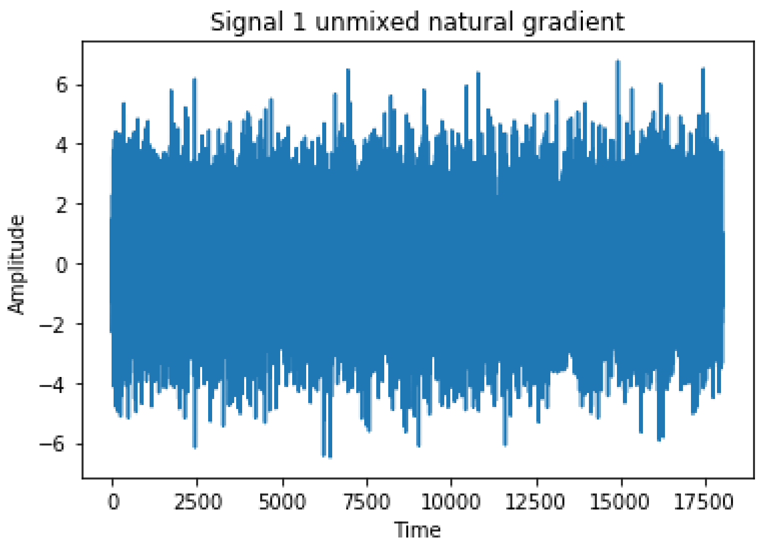

Sound2
Amplitude
Time
4
2
0
−2
−4
0
2500
5000
7500
10000
12500
15000
17500

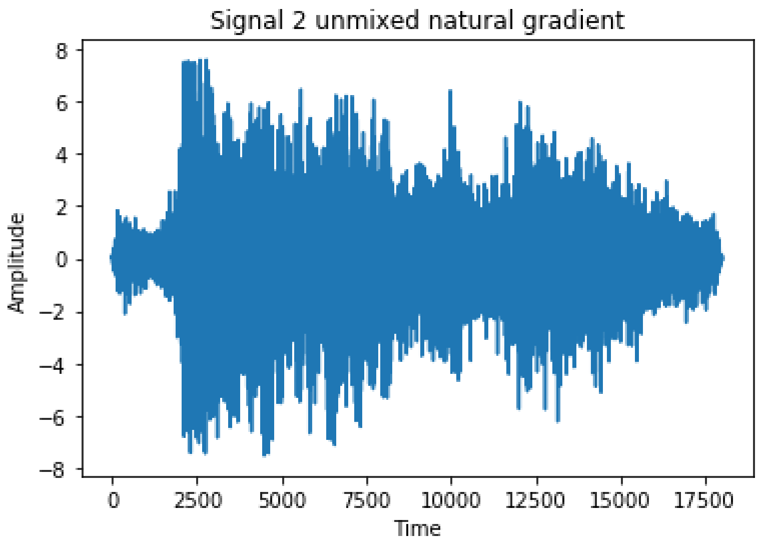

Noise
Amplitude
Time
4
3
2
1
0
−1
−2
−3
−4
0
2500
5000
7500
10000
12500
15000
17500

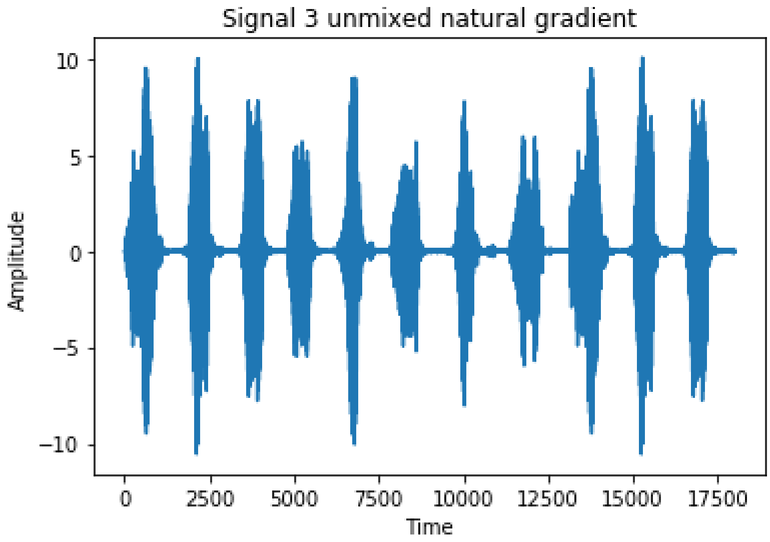

Sound1 mixed
Amplitude
Time
6
4
2
0
−2
−4
−6
0
2500
5000
7500
10000
12500
15000
17500


Sound2 mixed
Amplitude
Time
4
2
0
−2
−4
0
2500
5000
7500
10000
12500
15000
17500


Noise mixed
Amplitude
Time
3
2
1
0
−1
−2
−3
0
2500
5000
7500
10000
12500
15000
17500



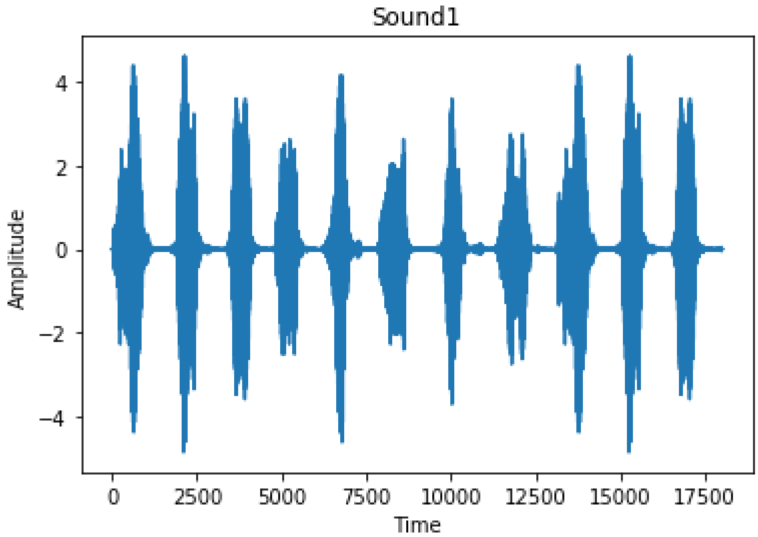


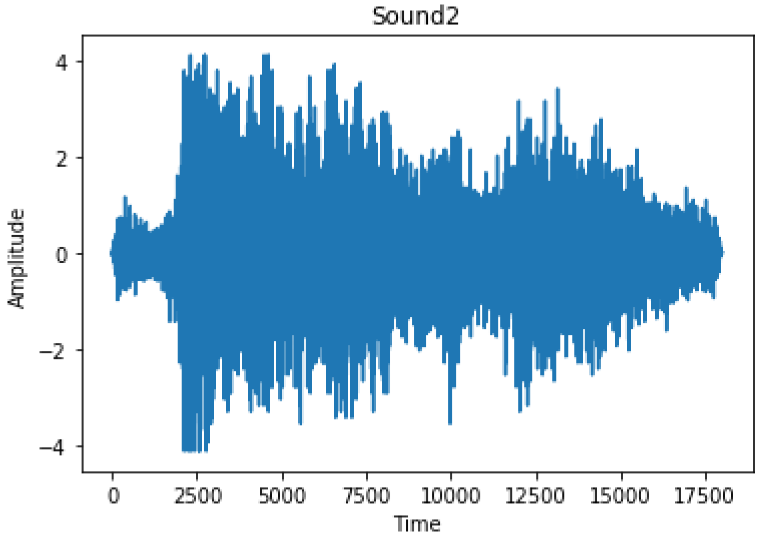


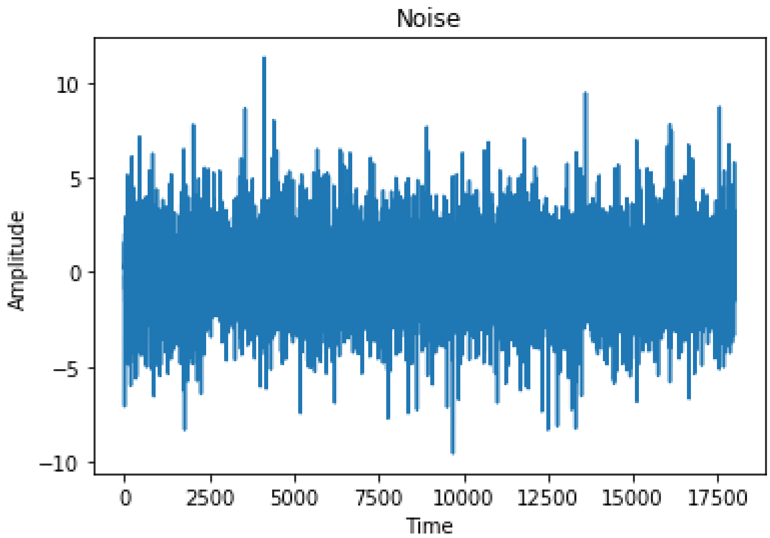
Plot original, mixed and unmixed for Laplacian

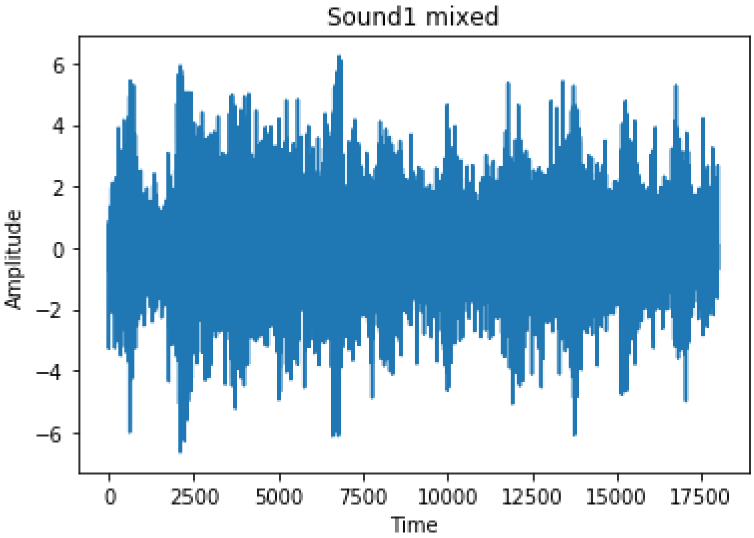
In [7]:

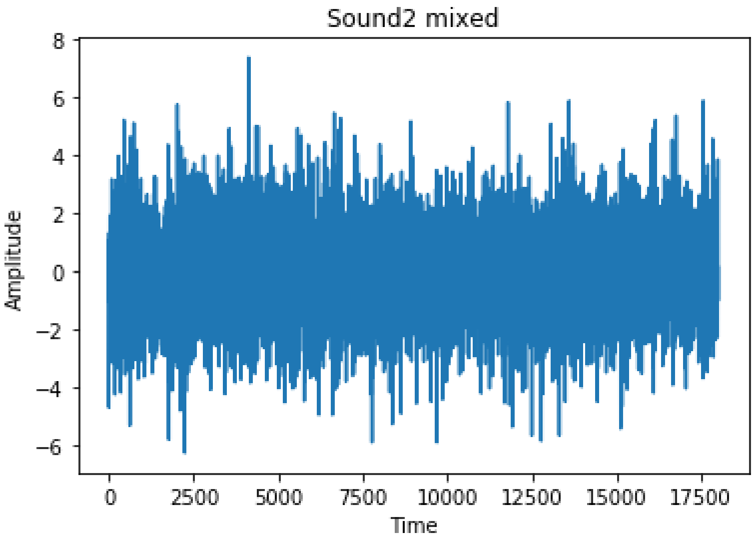
```
plt.figure()
plt.plot(dataSet1)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound1')
plt.figure()
plt.plot(dataSet2)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound2')
plt.figure()
plt.plot(dataSet3b)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Noise')

plt.figure()
plt.plot(x2[0,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound1 mixed')
plt.figure()
plt.plot(x2[1,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sound2 mixed')
plt.figure()
plt.plot(x2[2,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Noise mixed')

plt.figure()
plt.plot(unmixedLaplacian[0,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal 1 unmixed natural gradient')
plt.figure()
plt.plot(unmixedLaplacian[1,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal 2 unmixed natural gradient')
plt.figure()
plt.plot(unmixedLaplacian[2,:])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal 3 unmixed natural gradient')

plt.show()
```

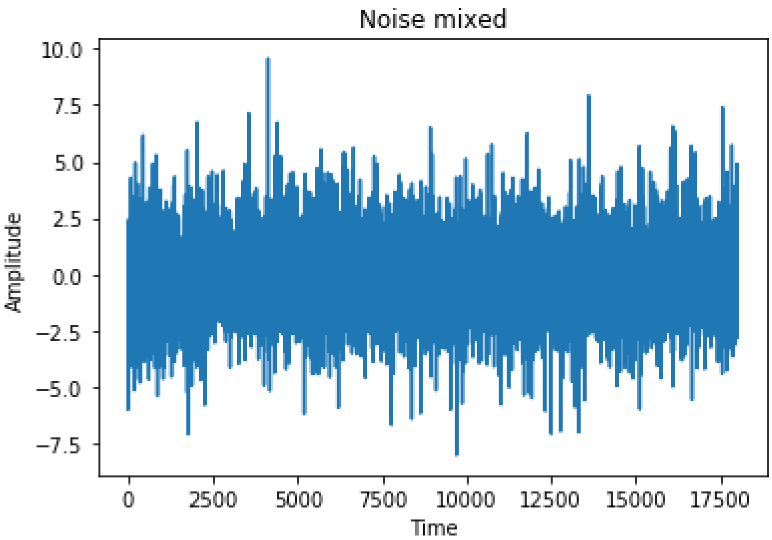

Sound1
Amplitude
Time
4
2
0
−2
−4
0
2500
5000
7500
10000
12500
15000
17500

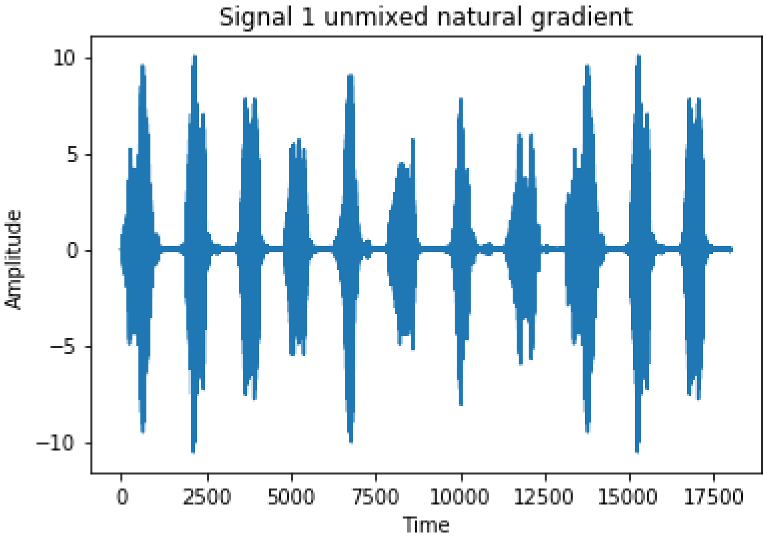

Sound2
Amplitude
Time
4
2
0
−2
−4
0
2500
5000
7500
10000
12500
15000
17500

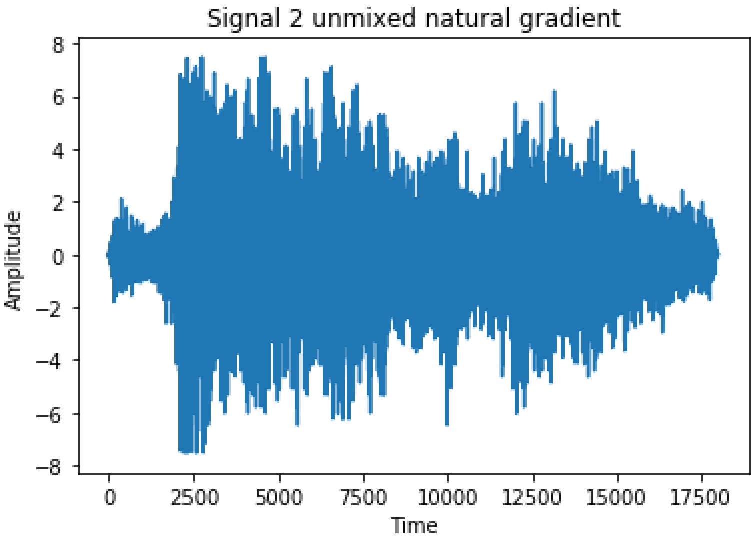

Noise
Amplitude
Time
10
5
0
−5
−10
0
2500
5000
7500
10000
12500
15000
17500

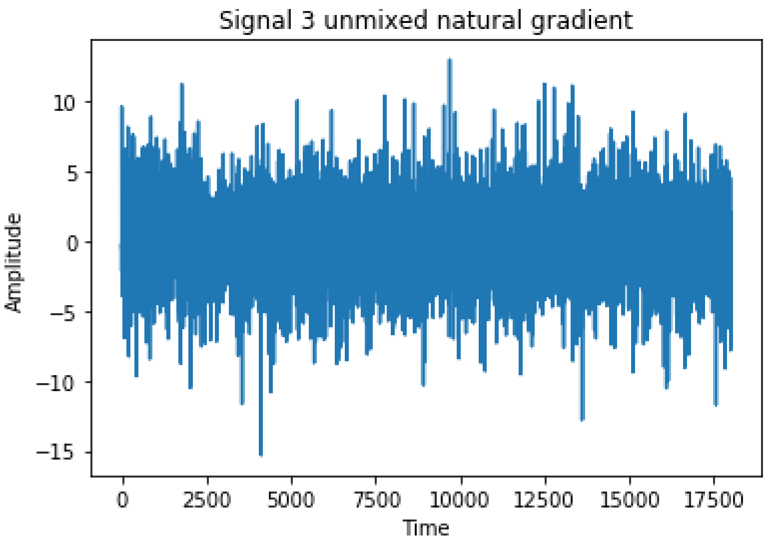

Sound1 mixed
Amplitude
Time
6
4
2
0
−2
−4
−6
0
2500
5000
7500
10000
12500
15000
17500


Sound2 mixed
Amplitude
Time
8
6
4
2
0
−2
−4
−6
0
2500
5000
7500
10000
12500
15000
17500


Noise mixed
Amplitude
Time
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0
2500
5000
7500
10000
12500
15000
17500







6.2 Moments of univariate distributions

a) Laplace Distributon

Moment generating function:

$$M_x(t) = \frac{e^{\mu t}}{1-\sigma^2 t^2}$$

First moment (derivating one time and setting t to null)

$$\cdots = \frac{e^{\mu t}(\mu - \mu\sigma^2 t^2 + 2\sigma^2 + t)}{(1-\sigma^2 t^2)^2}\Big|_{t=0} = \mu$$

Second moment

$$\cdots = \frac{4\sigma^2 t e^{\mu t}(\mu - \mu\sigma^2 t^2 + 2\sigma^2 t))}{(1-\sigma^2 t^2)^3} + \frac{e^{\mu t}(2\sigma^2 - 2\mu\sigma^2 t)}{(1-\sigma^2 t^2)^2} + \frac{\mu e^{\mu t}(\mu - \mu\sigma^2 t^2 + 2\sigma^2 t)}{(1-\sigma^2 t^2)^2}\Big|_{t=0} =$$

Third moment

$$\cdots = \mu + 6\mu\sigma^2$$

Standardized

$$\cdots = \frac{\mu + 6\mu\sigma^2}{(2\sigma^2)^{\frac{3}{2}}}$$

Fourth moment

$$\cdots = \mu^4 + 12\mu^2\sigma^2 + 24\sigma^4$$

Standardized:

$$\cdots = \frac{\mu^4}{4\sigma^4} + \frac{3\mu^2}{\sigma^2} + 6$$

b) Gauss Distribution Moment generating function:

$$M_x(t) = exp(\mu t + \frac{\sigma^2 t^2}{2})$$

We get the 1st moment by derivating the generating function and setting t to 0

$$\dots = \mu$$

Second moment (derivating the generating function two times and setting t to zero)

$$\dots = \mu^2 + \sigma^2$$

Third moment

$$\dots = \mu^3 + 3\mu\sigma^2$$

Standardized:

$$\dots = \frac{\mu^3 + 3\mu\sigma^2}{(\mu^2 + \sigma^2)^{\frac{3}{2}}}$$

Fourth moment

$$\dots = \mu^4 + 6\mu^2\sigma^2 3\sigma^4$$

Standardized:

$$\dots = \frac{\mu^4 + 6\mu^2\sigma^2 3\sigma^4}{(\mu^2 + \sigma^2)^2}$$

c) Uniform Distribution

Moment generating function:

$$M_x = E(e^{tx}) = \int_a^b e^{tx}\frac{1}{b-a} = \frac{1}{b-a}\left[\frac{e^{tx}}{t}\right]_a^b = \frac{e^{tb}-e^{ta}}{t(b-a)} = \frac{1}{t(b-a)}\left[bt + \frac{(bt)^2}{2!} + \frac{(bt)}{3!}\right.$$

$$= \frac{1}{(b-a)}\left[b + \frac{b^2t}{2!} + \frac{b^3t^2}{3!} - \left(a + \frac{a^2t}{2!} + \frac{a^3t^2}{3!}\right)\right]$$

We get the 1st moment by derivating M_x one time and setting t to 0:

$$\cdots = \frac{1}{(b-a)}\left[\frac{b^2}{2!} - \frac{a^2}{2!}\right] = \frac{b+a}{2}$$

We get the 2nd moment by derivating M_x two times and setting t to 0:

$$\cdots = \frac{1}{(b-a)}\left[\frac{b^3}{3} - \frac{a^3}{3}\right] = \frac{b^3-a^3}{3(b-a)} = \frac{b^2+ba+a^2}{3}$$

We get the 3rd moment by derivating M_x three times and setting t to 0:

$$\cdots = \frac{1}{(b-a)}\left[\frac{b^4}{4} - \frac{a^4}{4}\right] = \frac{b^4-a^4}{4(b-a)}$$

Hence the 3rd standardized moment is

$$\frac{\frac{b^4-a^4}{4(b-a)}}{\left(\frac{b^3-a^3}{3(b-a)}\right)^{\frac{3}{2}}} = \frac{b^4-a^4}{4(b-a)}\left(\frac{3(b-a)}{b^3-a^3}\right)^{\frac{2}{3}}$$

We get the 4th moment by derivating M_x four times and setting t to 0:

$$\cdots = \frac{1}{(b-a)}\left[\frac{b^5}{5} - \frac{a^5}{5}\right] = \frac{b^5-a^5}{5(b-a)}$$

Hence the 3rd standardized moment is

$$\frac{\frac{b^5-a^5}{5(b-a)}}{\left(\frac{b^3-a^3}{3(b-a)}\right)^{\frac{4}{2}}} = \frac{b^5-a^5}{5(b-a)}\left(\frac{3(b-a)}{b^3-a^3}\right)^{2} = \frac{9(b^5-a^5)(b-a)}{5(b^3-a^3)^2}$$

	Laplace (μ, b)	**Gauss** (μ, σ)	**Uniform** (a, b)
mean: first moment $\langle X \rangle$	μ	μ	$\frac{b+a}{2}$
variance: second centered moment $\langle X \rangle_c^2$	$2\sigma^2$	$\mu + \sigma$	$\frac{b^2+ba+a^2}{3}$
skewness: third standardized moment $\langle X \rangle_s^3$	$\frac{\mu+6\mu\sigma^2}{(2\sigma^2)^{\frac{3}{2}}}$	$\frac{\mu^3 + 3\mu\sigma^2}{(\mu^2+\sigma^2)^{\frac{3}{2}}}$	$\frac{b^4-a^4}{4(b-a)}\left(\frac{3(b-a)}{b^3-a^3}\right)^{\frac{2}{3}}$
kurtosis: fourth standardized moment $\langle X \rangle_s^4$	$\frac{\mu^4}{4\sigma^4} + \frac{3\mu^2}{\sigma^2} + 6$	$\frac{\mu^4 + 6\mu^2\sigma^2 3\sigma^4}{(\mu^2+\sigma^2)^2}$	$\frac{9(b^5-a^5)(b-a)}{5(b^3-a^3)^2}$

$$\langle X^i \rangle_c = \langle (X - \langle X \rangle)^i \rangle$$

$$\langle X^i \rangle_s = \frac{\langle X^i \rangle_c}{\langle X^2 \rangle_c^{i/2}}$$

6.3 Kurtosis

In [8]:

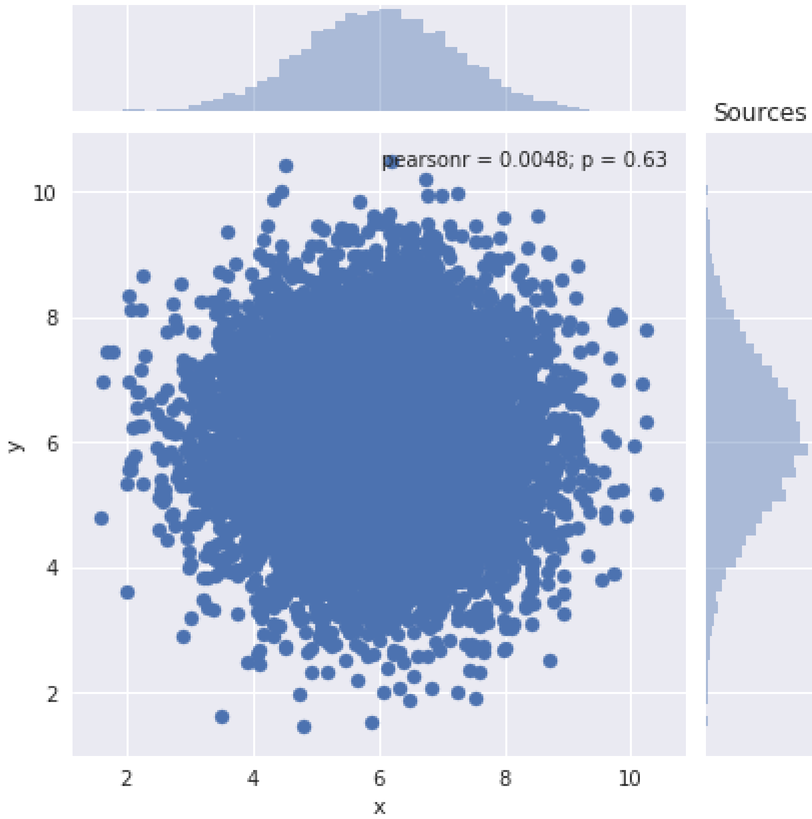
```
import numpy as np
import scipy as scipy
import scipy.io as io
from sklearn.decomposition import PCA
from sklearn import preprocessing as pre
import math
import matplotlib.pyplot as plt
import seaborn as sea
import pandas as pd

#load data from mat file
distrib = io.loadmat("distrib.mat")

normal = distrib.get("normal")
uniform = distrib.get("uniform")
laplacian = distrib.get("laplacian")

# a) apply mixing
def mix(x):
    A = np.asarray([[4,3],[2,1]])
    return np.dot(A,x)

# b) center to zero mean
def center(x):
    return x - x.mean(axis=1, keepdims=True)

# c) apply PCA and project data onto PC
def applyPCA(x):
    pca = PCA()
    pca.fit(x)
    return pca.transform(x)

# d) scale data to unit variance
def scale(x):
```

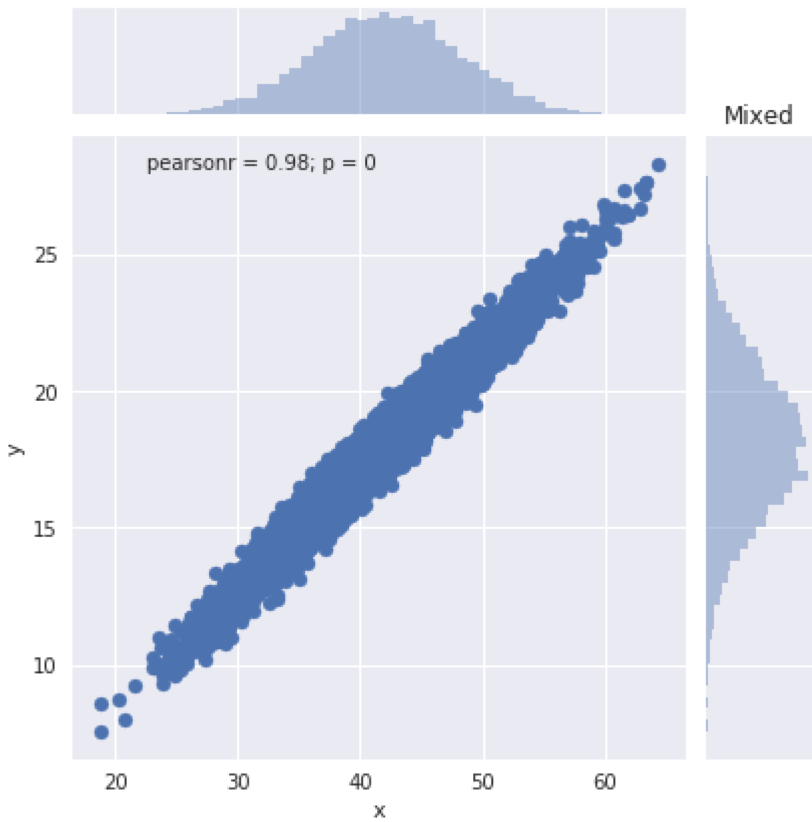
```
    return pre.maxabs_scale(x)

# e rotate data
def rotate(x, angle): # x of shape (2,n)
    R = np.asarray([[math.cos(angle), -math.sin(angle)],[math.sin(angle), math.c
os(angle)]])
    return np.dot(R, x)

def rotateAndGetKurtosis(x, angle):
    scaledRot = rotate(x.T, angle).T
    kurt = scipy.stats.kurtosis(scaledRot)
    #print("kurt: ", kurt)
    return kurt

def findMinAndMaxKurt(x, angles):
    kurts = np.empty((angles.shape[0],2))
    idx = 0
    for angle in angles:
        kurt = rotateAndGetKurtosis(x, angle)
        kurts[idx] = kurt
        idx += 1

    maxIdx = np.argmax(kurts[:,0])
    minIdx = np.argmin(kurts[:,1])
    kurtPlotData = np.asarray((angles, kurts[:,0], kurts[:,1]))
    return maxIdx, minIdx, kurtPlotData

def plotdata(source, mixed, centered, projected, scaled, rotMin, rotMax, kurtDat
a):
    # Plotting the results.

    df = pd.DataFrame(source, columns=["x", "y"])
    sea.jointplot(x="x", y="y", data=df)
    plt.title('Sources')

    df = pd.DataFrame(mixed, columns=["x", "y"])
    sea.jointplot(x="x", y="y", data=df)
    plt.title('Mixed')

    df = pd.DataFrame(centered, columns=["x", "y"])
    sea.jointplot(x="x", y="y", data=df)
    plt.title('Centered')

    df = pd.DataFrame(projected, columns=["x", "y"])
    sea.jointplot(x="x", y="y", data=df)
    plt.title('Projected')

    df = pd.DataFrame(scaled, columns=["x", "y"])
    sea.jointplot(x="x", y="y", data=df)
    plt.title('Scaled')

    df = pd.DataFrame(rotMin, columns=["x", "y"])
    sea.jointplot(x="x", y="y", data=df)
    plt.title('Rotation min kurtosis')

    df = pd.DataFrame(rotMax, columns=["x", "y"])
    sea.jointplot(x="x", y="y", data=df)
    plt.title('Rotation max kurtosis')

    plt.figure()
    plt.plot(kurtData[0,:], kurtData[1,:], label="Dimension 1")
```

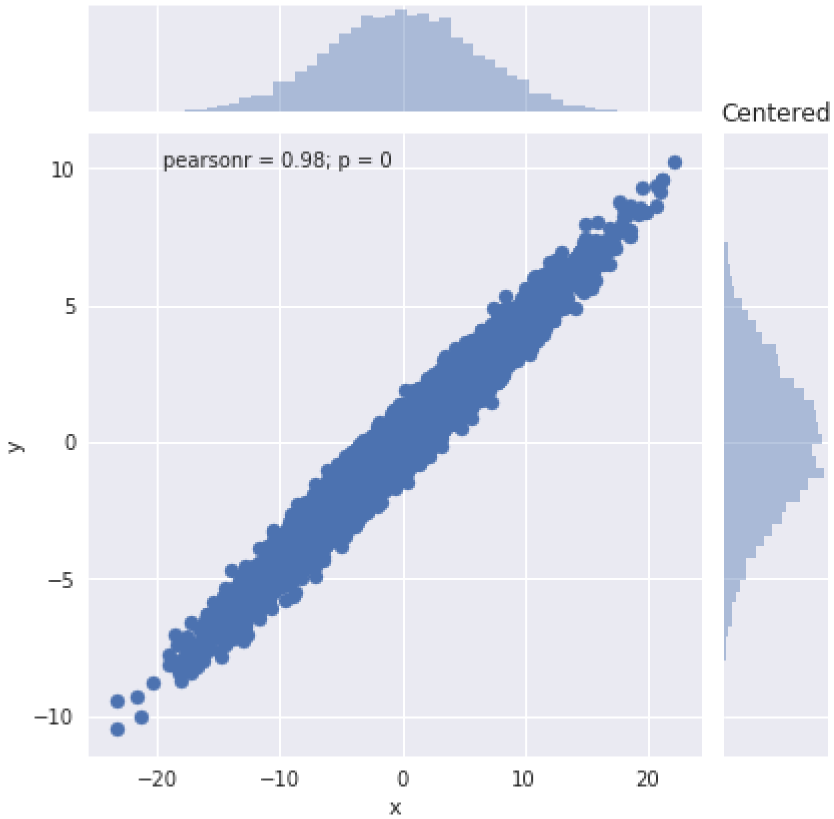
```
    plt.plot(kurtData[0,:], kurtData[2,:], label="Dimension 2")
    plt.legend()
    plt.xlabel('Angle')
    plt.ylabel('Kurtosis')
    plt.title("Kurtosis over angle")

def runExercise(x):
    xNormal = mix(x)
    xNormalCentered = center(xNormal)
    # transpose data for further processing
    xNormalCentered = xNormalCentered.T

    projectedNormalCentered = applyPCA(xNormalCentered)
    scaledNormal = scale(projectedNormalCentered)

    angles = np.arange(0, 2, 1/50)
    angles = angles * math.pi
    maxIdx, minIdx, kurtData = findMinAndMaxKurt(scaledNormal, angles)

    rotNormMax = rotate(scaledNormal.T, angles[maxIdx])
    rotNormMin = rotate(scaledNormal.T, angles[minIdx])

    plotdata(normal.T, xNormal.T, xNormalCentered, projectedNormalCentered, scal
edNormal, rotNormMin.T, rotNormMax.T, kurtData)
    plt.show()

print("Normal")
runExercise(normal)
print("Uniform")
runExercise(uniform)
print("Laplacian")
runExercise(laplacian)
```

Normal

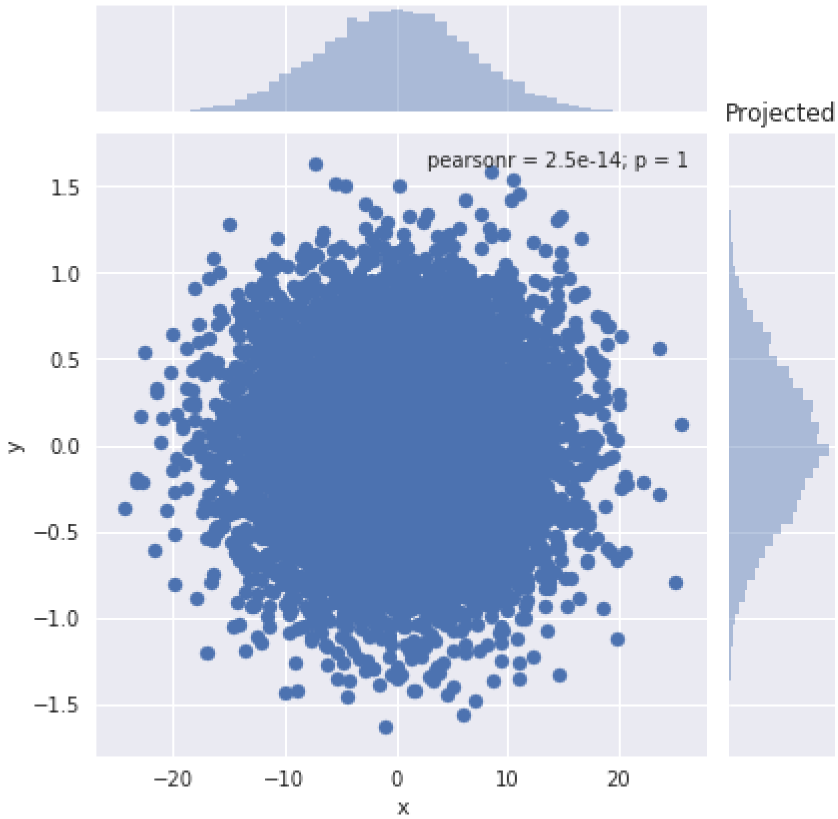
```
/usr/lib/python3.6/site-packages/matplotlib/font_manager.py:1297: U
serWarning: findfont: Font family ['sans-serif'] not found. Falling
back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```



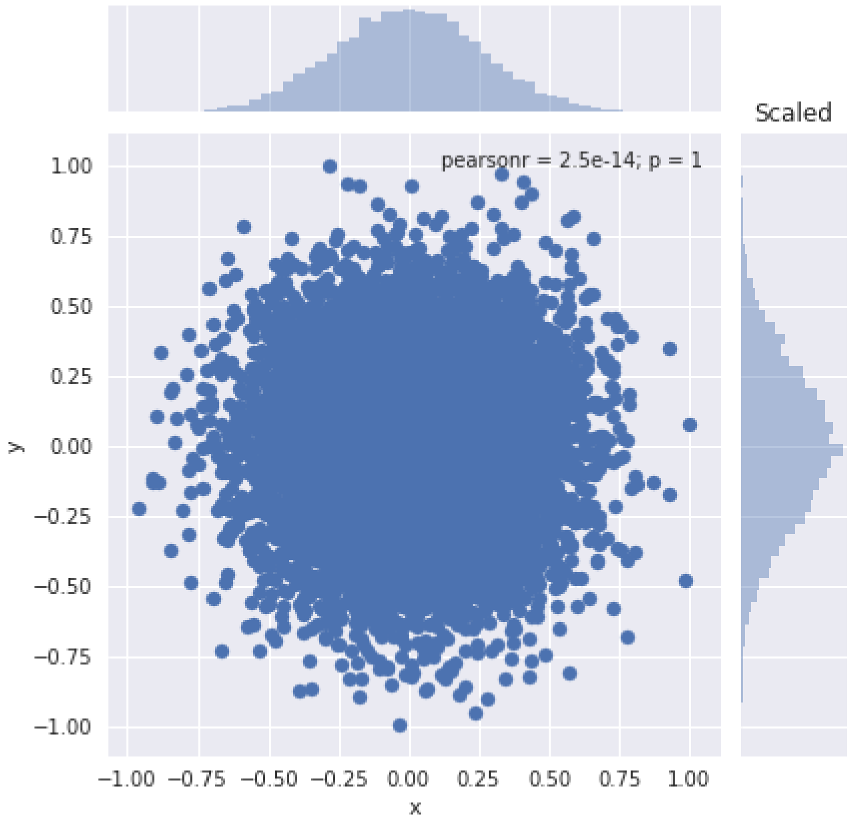


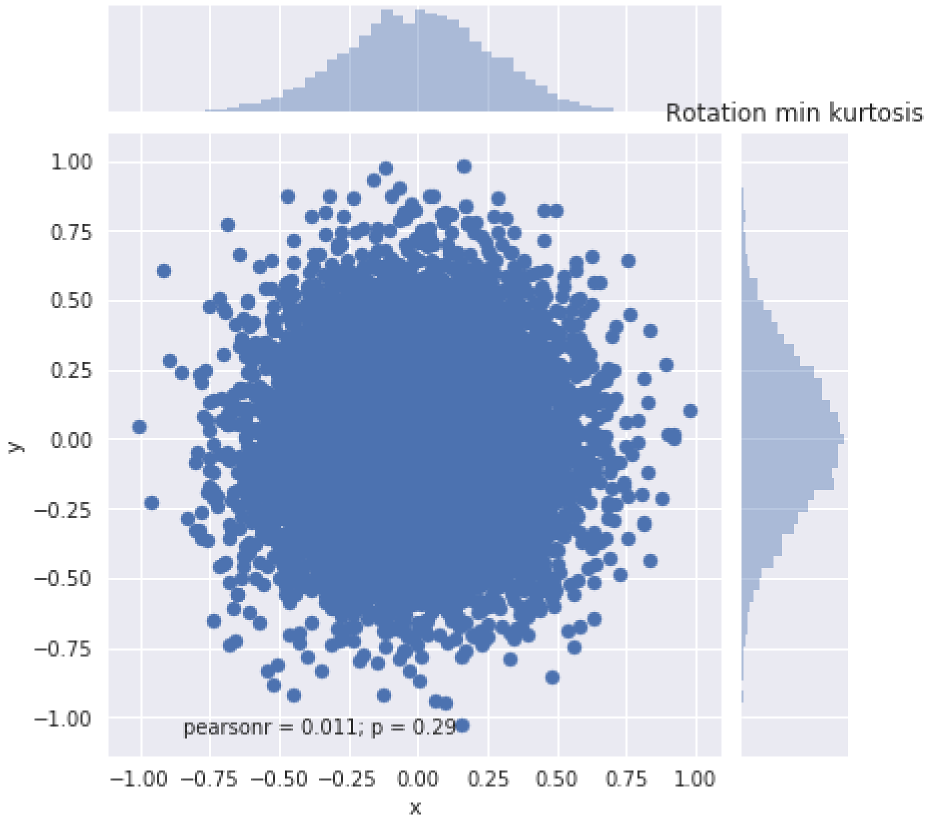


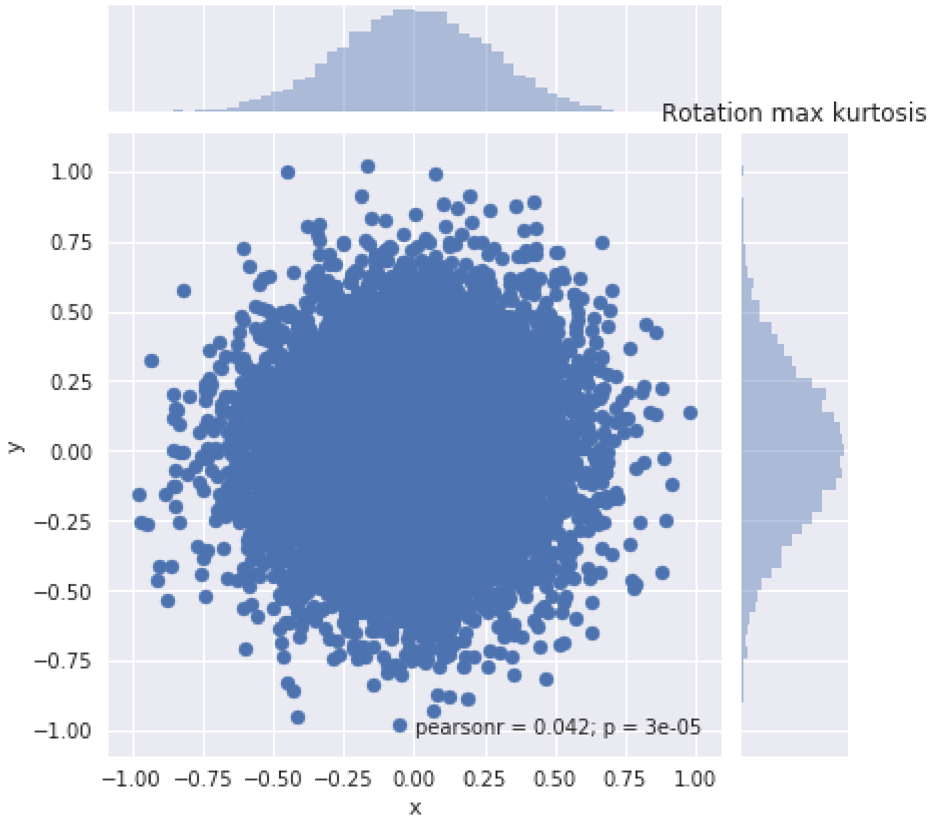
Centered
pearsonr = 0.98; p = 0
10
5
0
−5
−10
y
−20
−10
0
10
20
x

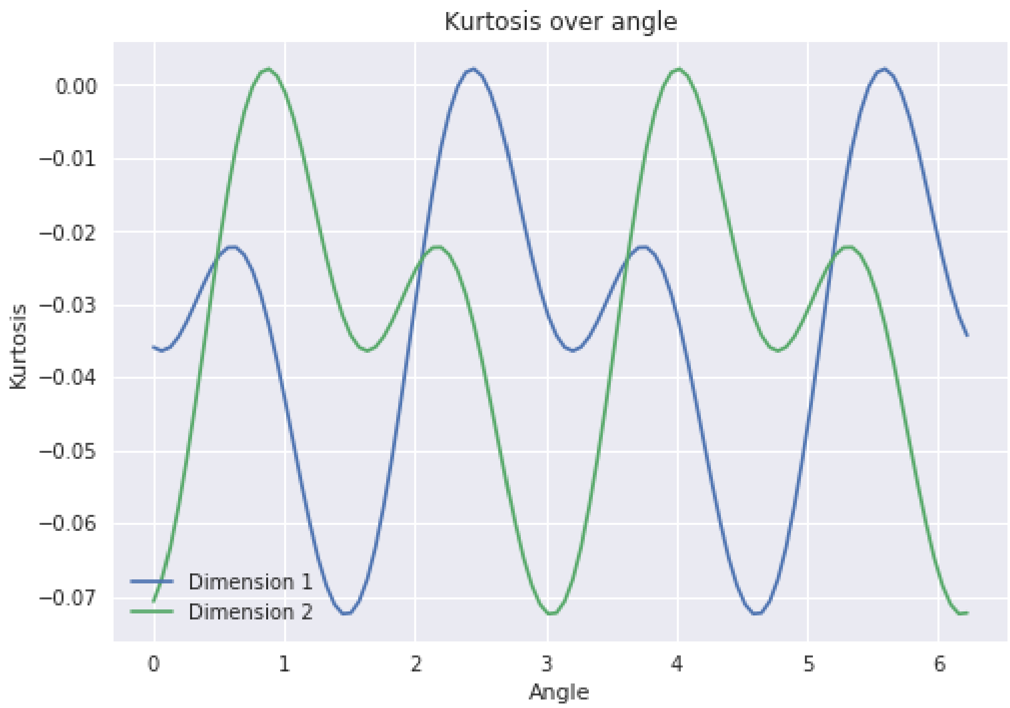


Projected
pearsonr = 2.5e-14; p = 1
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−20
−10
0
10
20
x

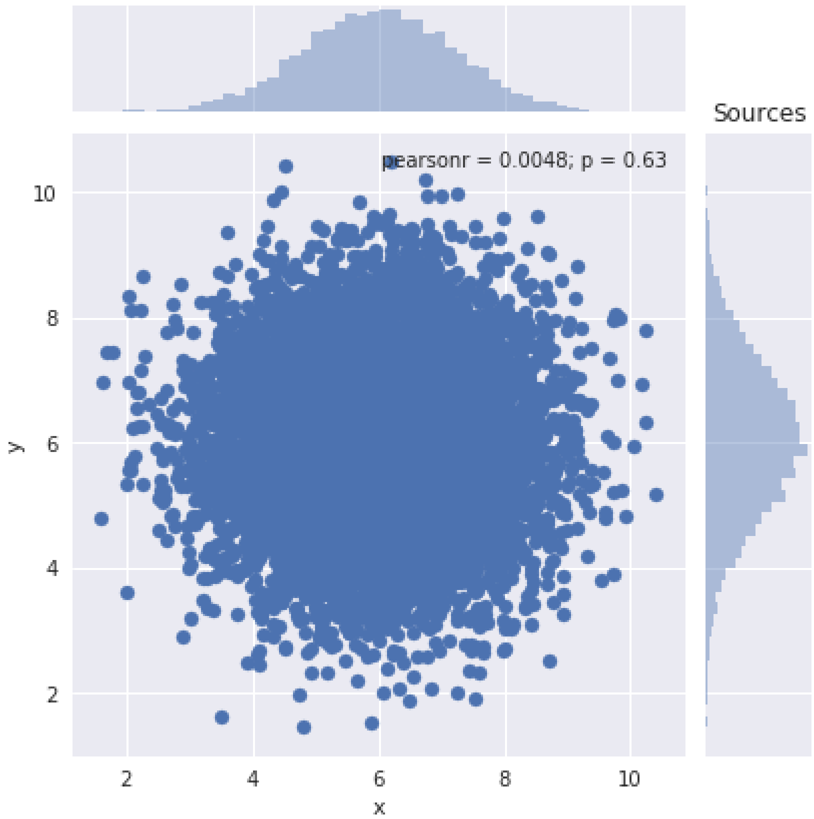

Scaled
pearsonr = 2.5e-14; p = 1
y
x

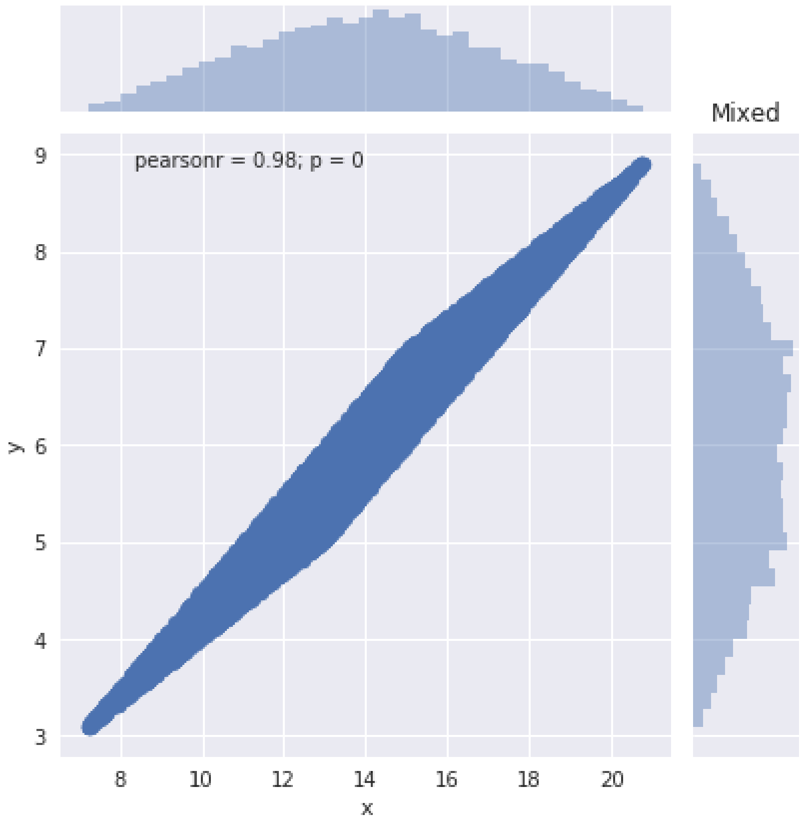

Rotation min kurtosis
pearsonr = 0.011; p = 0.29
y
x

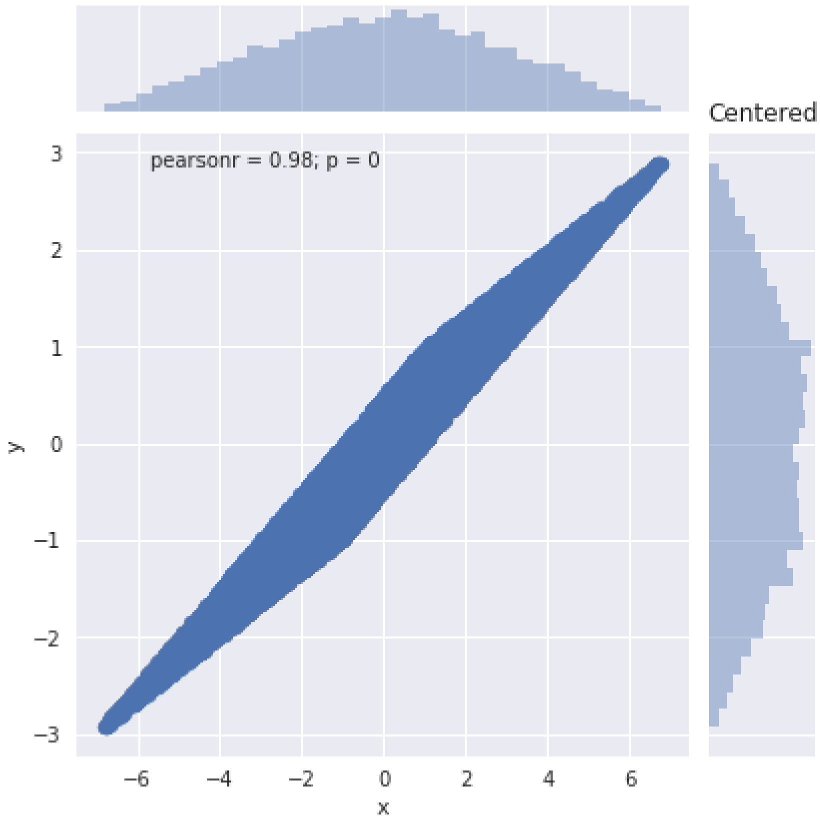

Rotation max kurtosis
pearsonr = 0.042; p = 3e-05
y
x
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

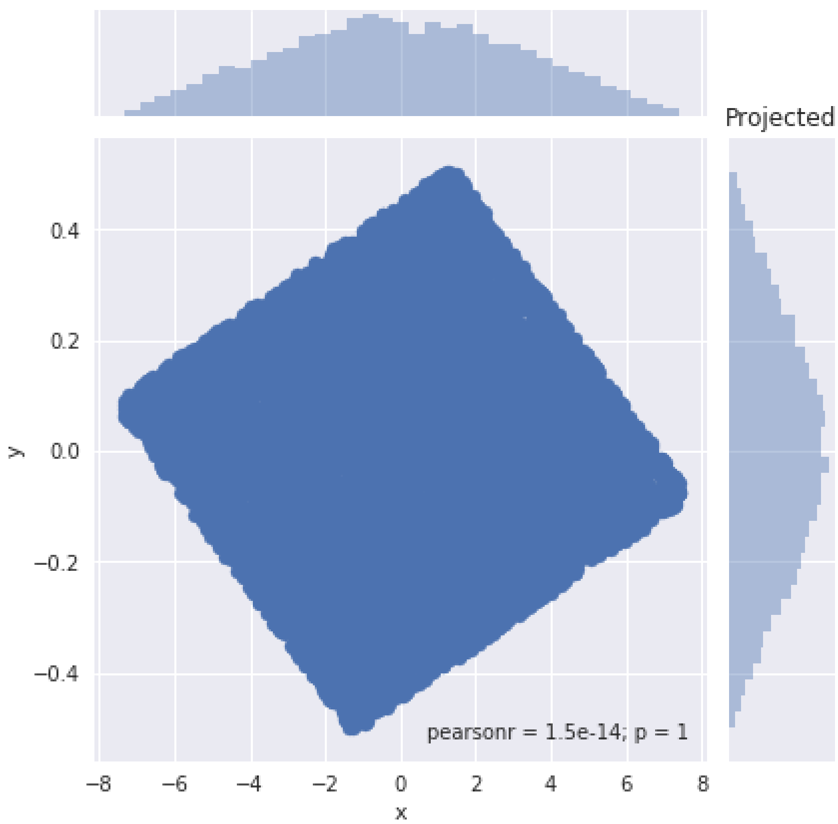

Kurtosis over angle
Kurtosis
Angle
Dimension 1
Dimension 2
0.00
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07
0 1 2 3 4 5 6

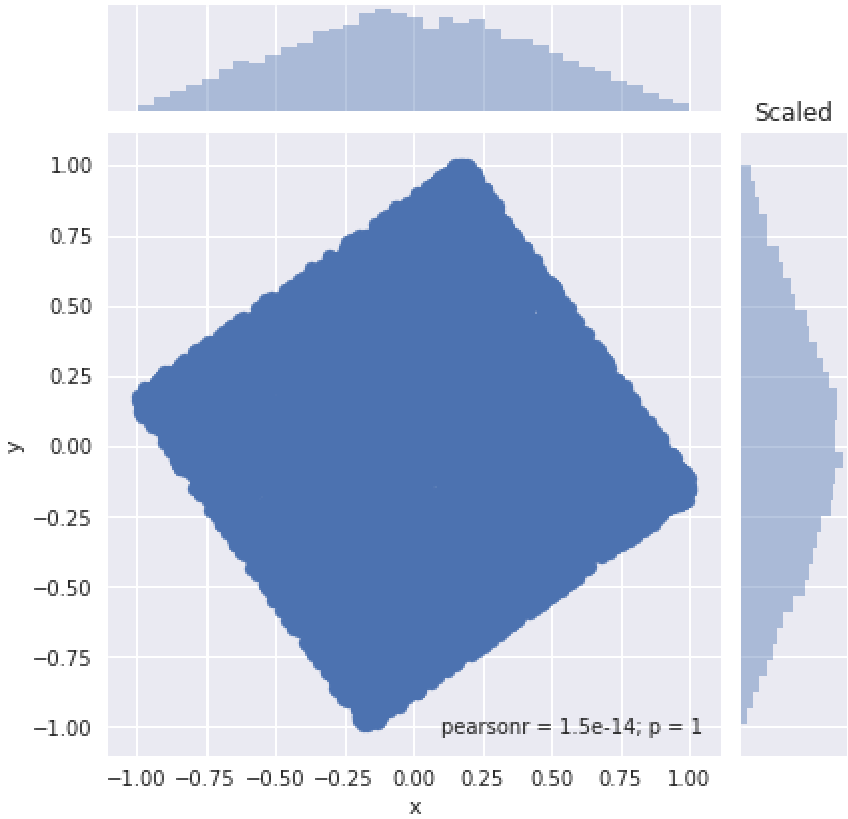
```
Uniform
```

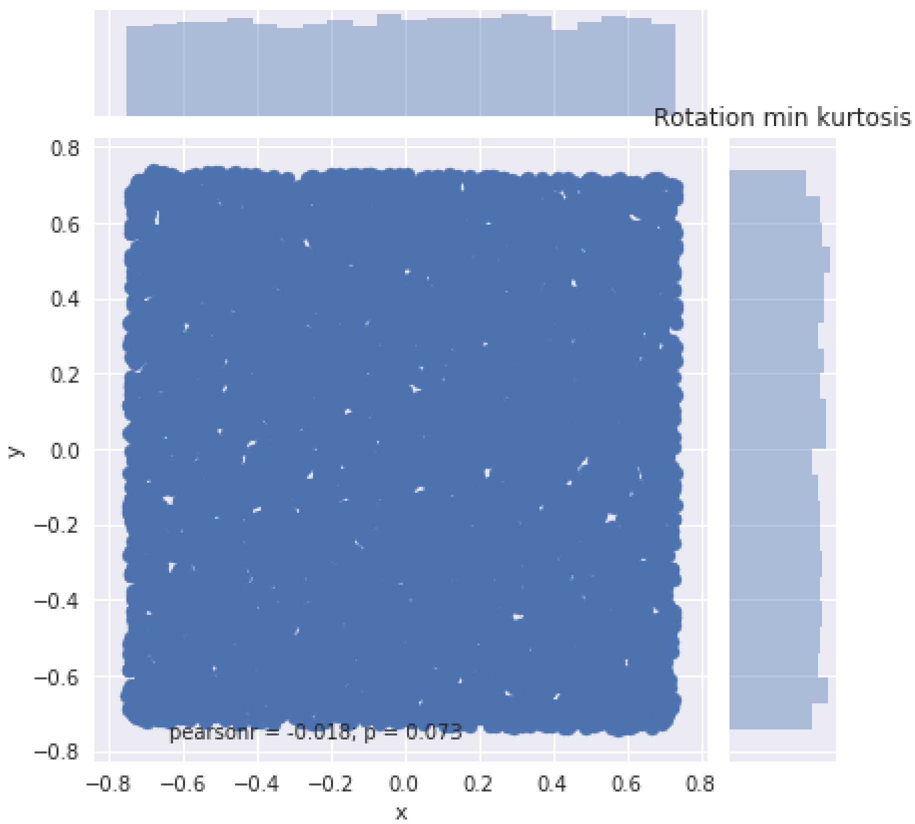

Sources
pearsonr = 0.0048; p = 0.63
y
x

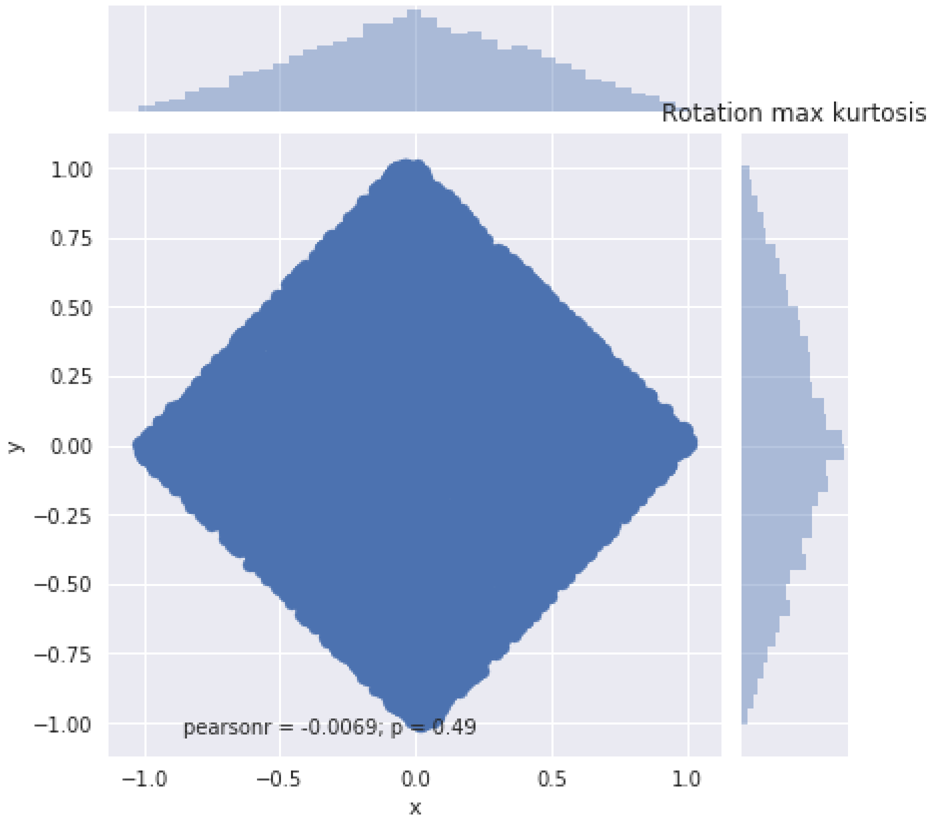

Mixed
pearsonr = 0.98; p = 0
y
x

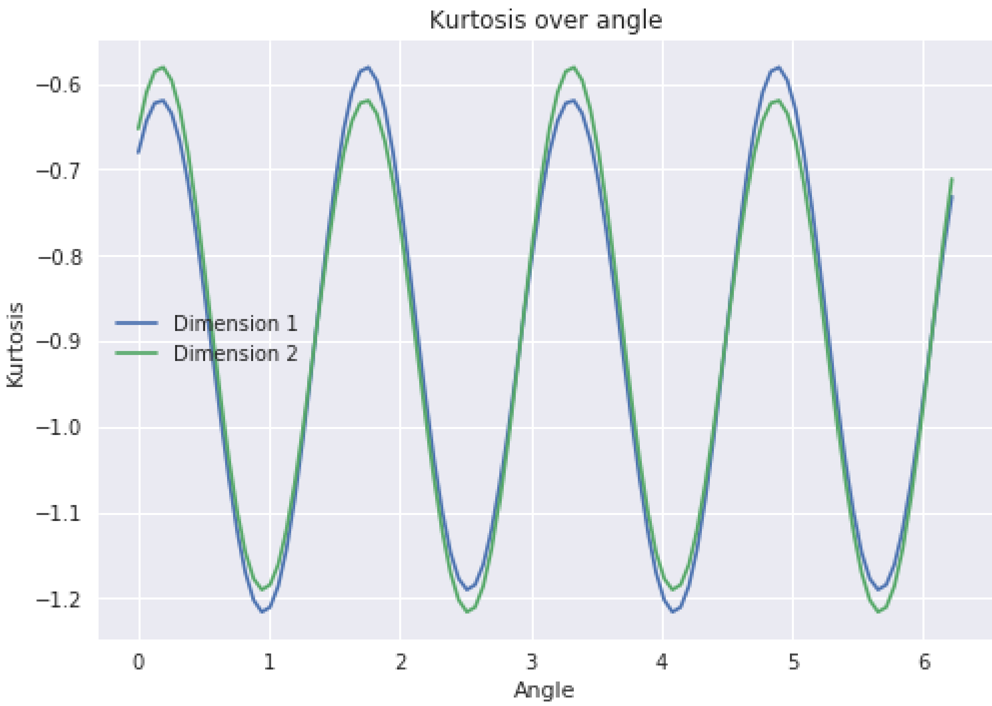

Centered
pearsonr = 0.98; p = 0
3
2
1
0
−1
−2
−3
y
−6
−4
−2
0
2
4
6
x


Projected
0.4
0.2
0.0
−0.2
−0.4
y
pearsonr = 1.5e-14; p = 1
−8
−6
−4
−2
0
2
4
6
8
x

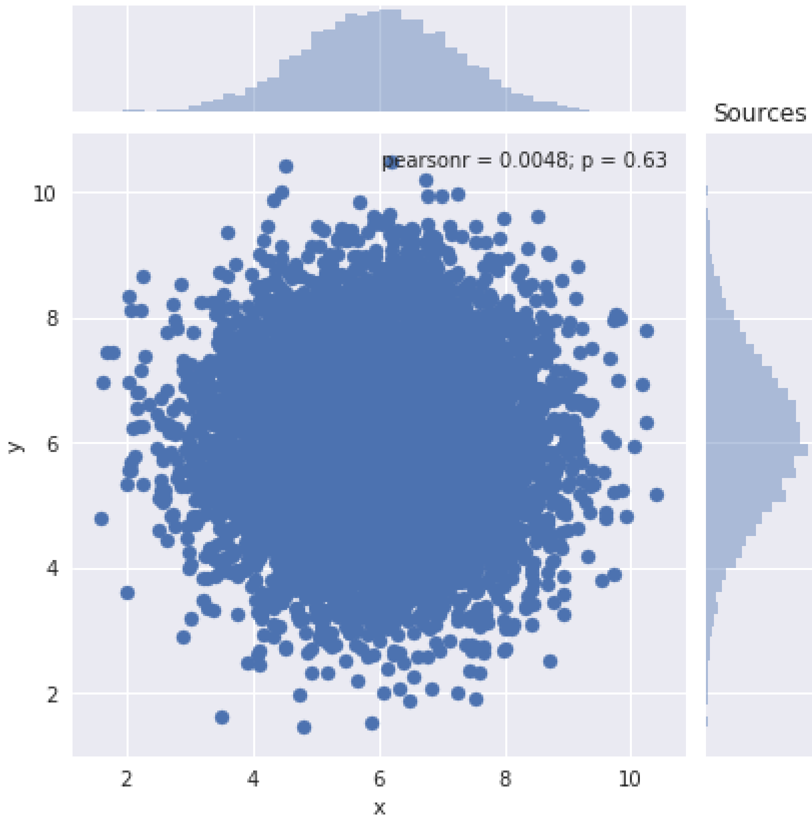

Scaled
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
pearsonr = 1.5e-14; p = 1
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

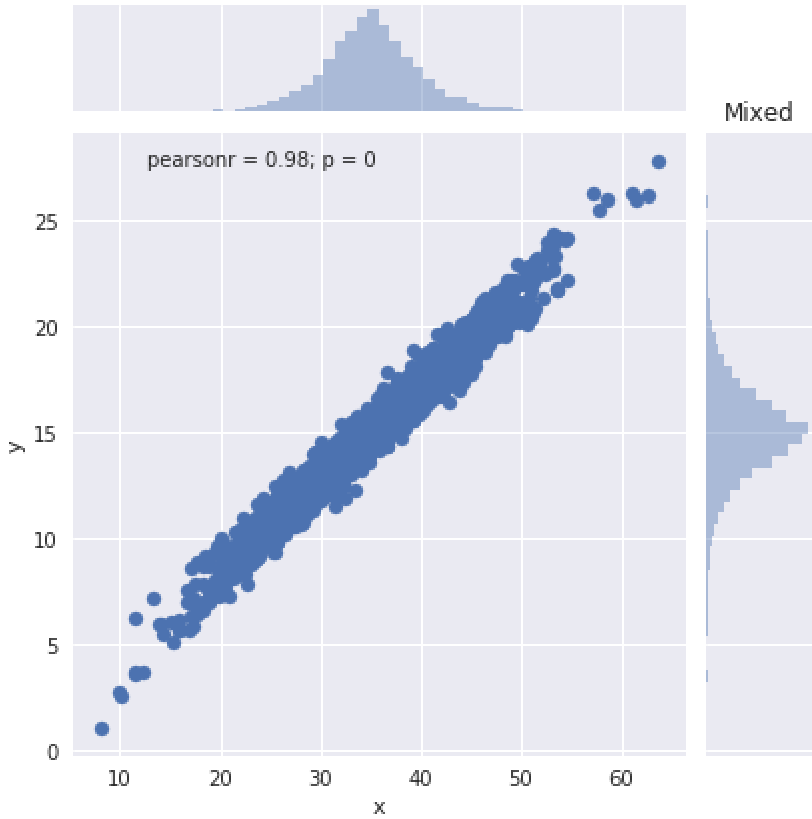

Rotation min kurtosis
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
y
pearsonr = -0.018; p = 0.073
−0.8 −0.6 −0.4 −0.2 0.0 0.2 0.4 0.6 0.8
x

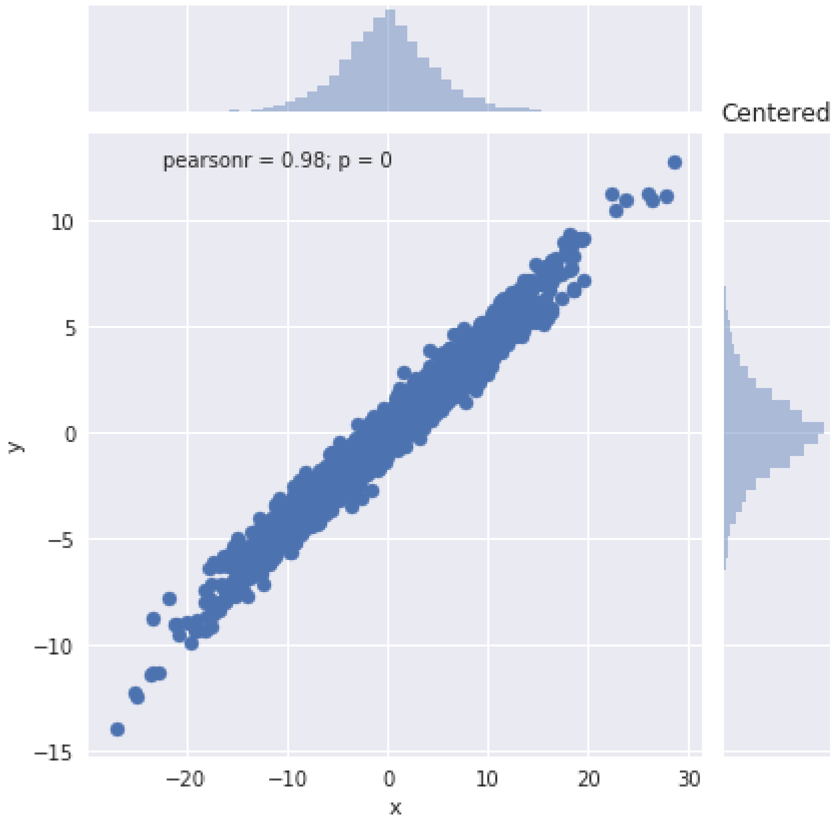

Rotation max kurtosis
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
pearsonr = -0.0069; p = 0.49
−1.0
−0.5
0.0
0.5
1.0
x

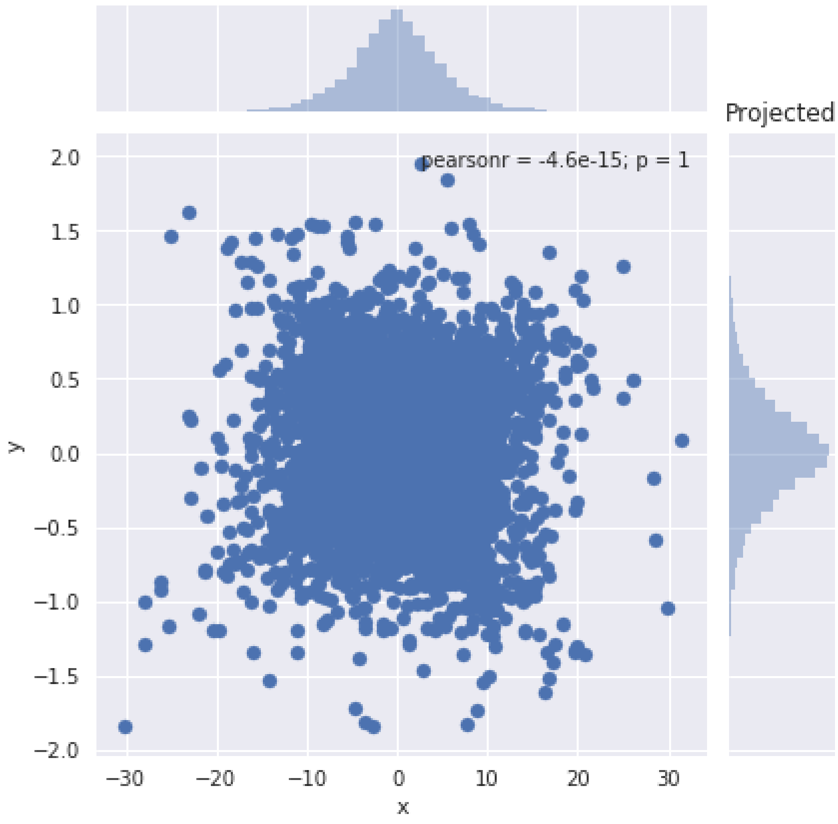

Kurtosis over angle
−0.6
−0.7
−0.8
−0.9
−1.0
−1.1
−1.2
Kurtosis
Dimension 1
Dimension 2
0
1
2
3
4
5
6
Angle

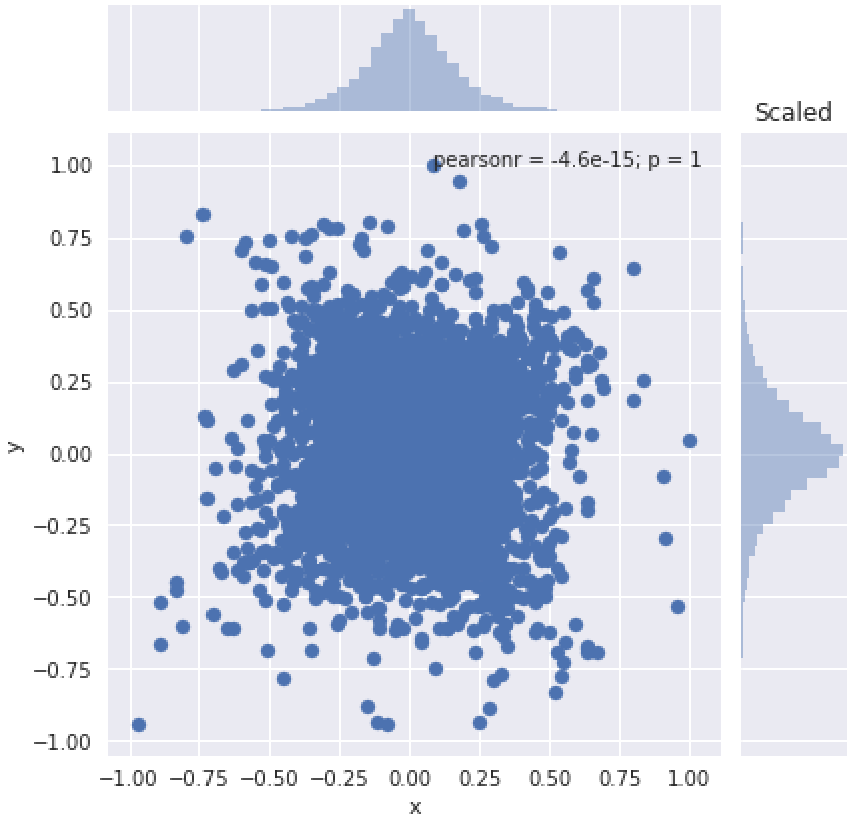
```
Laplacian
```

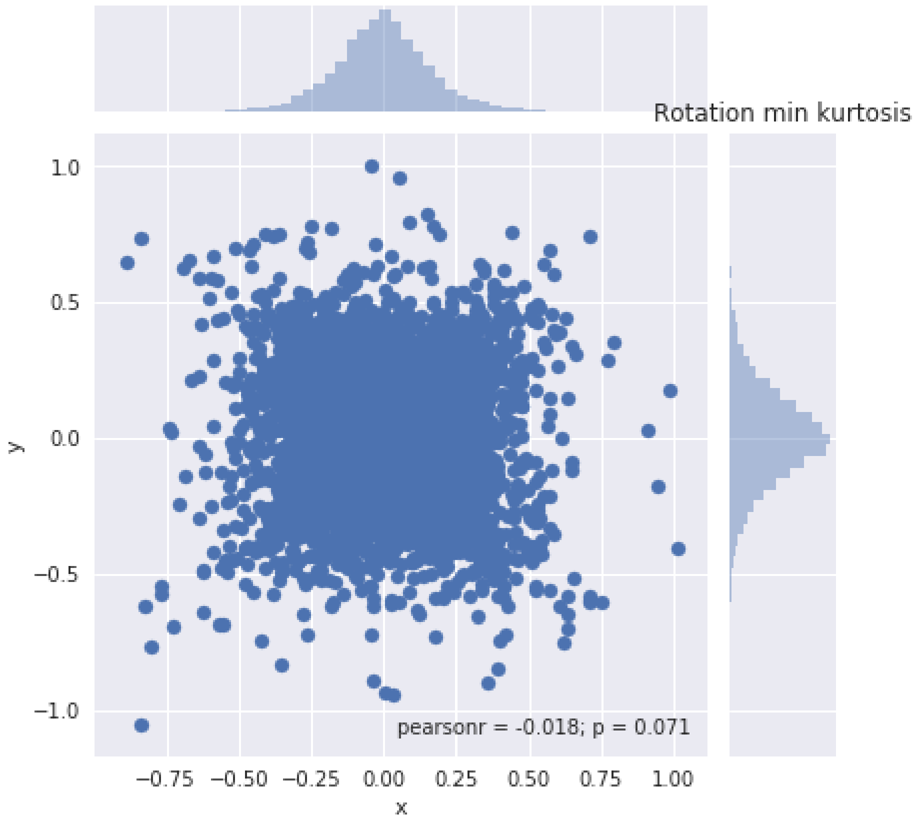

Sources
pearsonr = 0.0048; p = 0.63
y
x
2
4
6
8
10

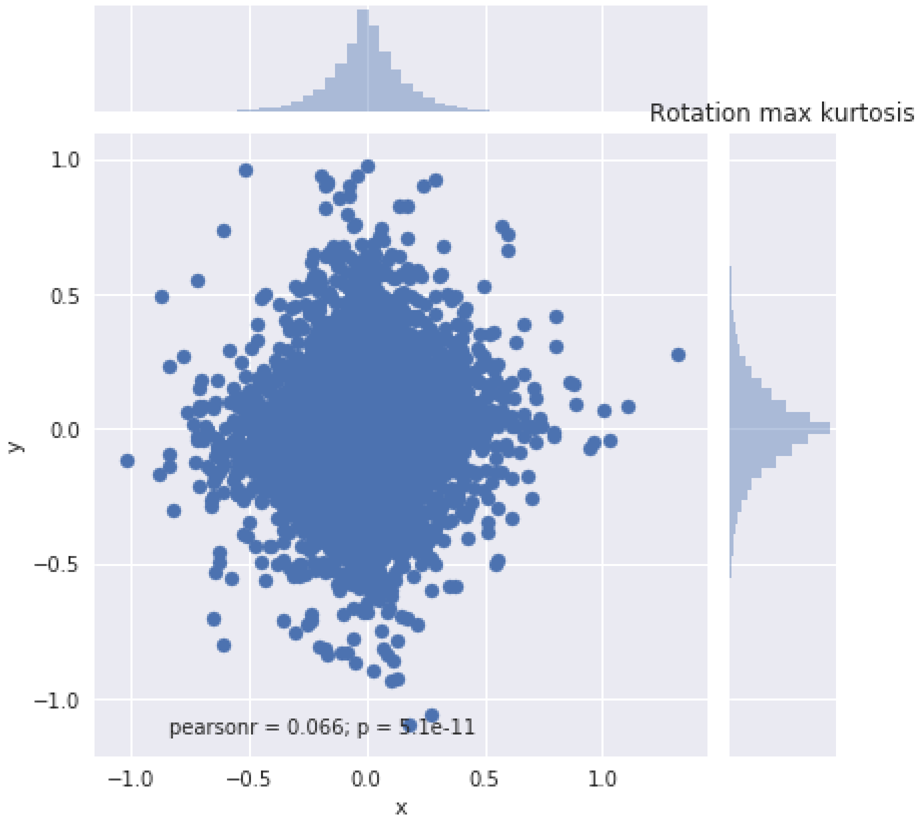

Mixed
pearsonr = 0.98; p = 0
y
x
0
5
10
15
20
25
30
40
50
60

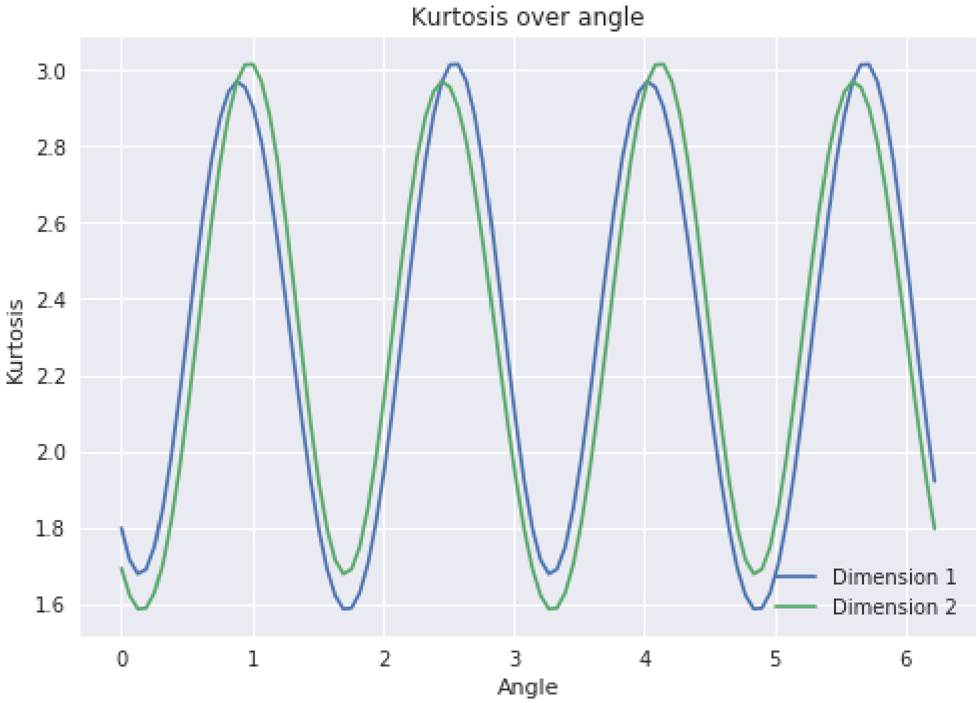

Centered
pearsonr = 0.98; p = 0
y
x


Projected
pearsonr = -4.6e-15; p = 1
y
x


Scaled
pearsonr = -4.6e-15; p = 1
y
x


Rotation min kurtosis
pearsonr = -0.018; p = 0.071
y
x





Discussion Histograms

The histograms for normal distribution has a bell-shape, for uniform distibution a bulky shape and for laplacian distibution a peaky shape with some outliers. The shapes were expected and correspond to the examples shown in the lecture.

In [ ]: