

13	14
12 15	5 5
16 / 20	

## SMD-Abgabe

### 5. Übungsblatt

Lars Kolk

[lars.kolk@tu-dortmund.de](mailto:lars.kolk@tu-dortmund.de)

Julia Sobolewski

[julia.sobolewski@tu-dortmund.de](mailto:julia.sobolewski@tu-dortmund.de)

Jannine Salewski

[jannine.salewski@tu-dortmund.de](mailto:jannine.salewski@tu-dortmund.de)

Abgabe: 22.11.2018

TU Dortmund – Fakultät Physik

### Aufgabe 13

Abb.

a) und b) Für die Teilaufgabe b) ergibt sich das in 1 zu sehende Histogramm.

3P.

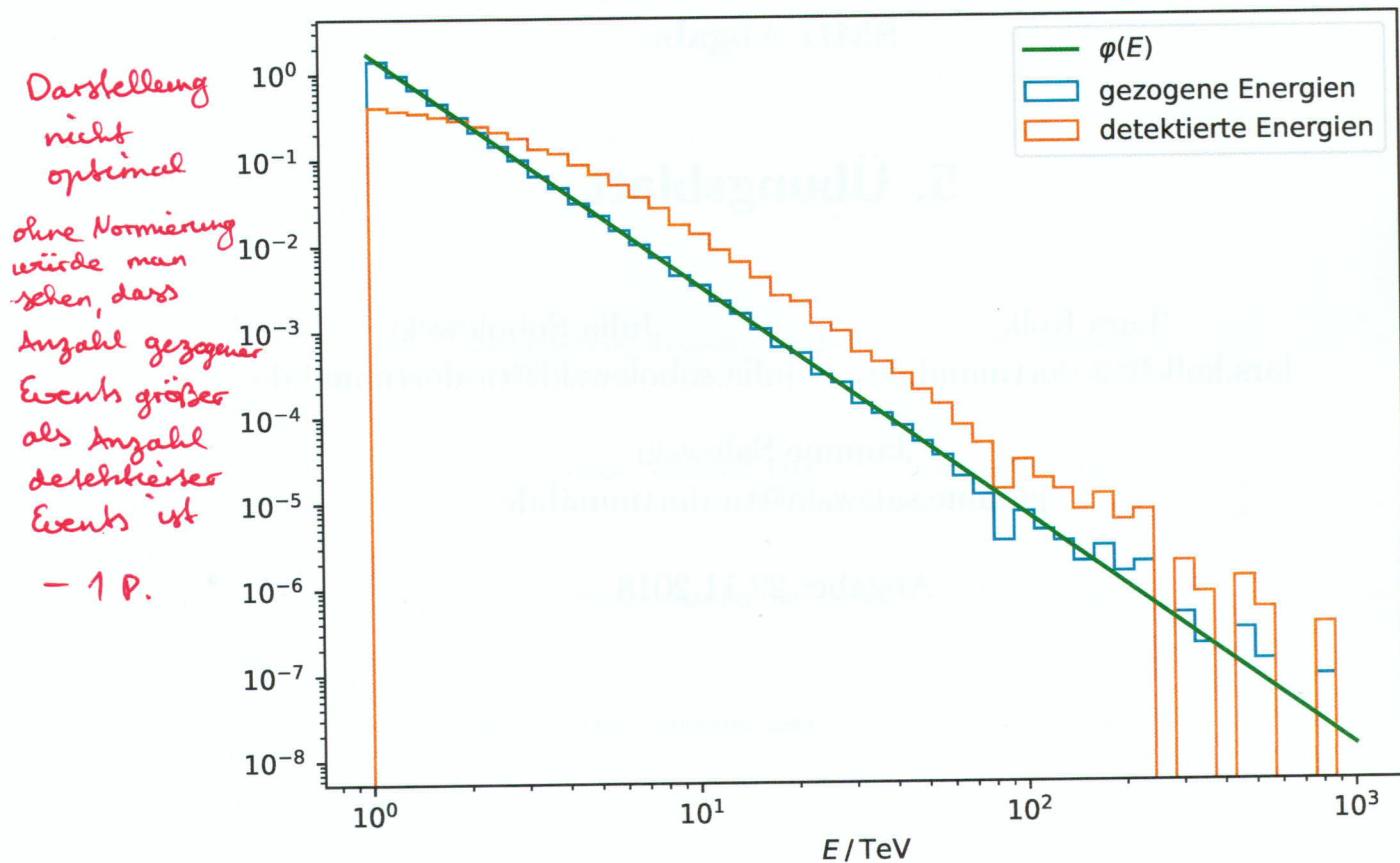


Abbildung 1: Histogramm der gezogenen und detektierten Energien

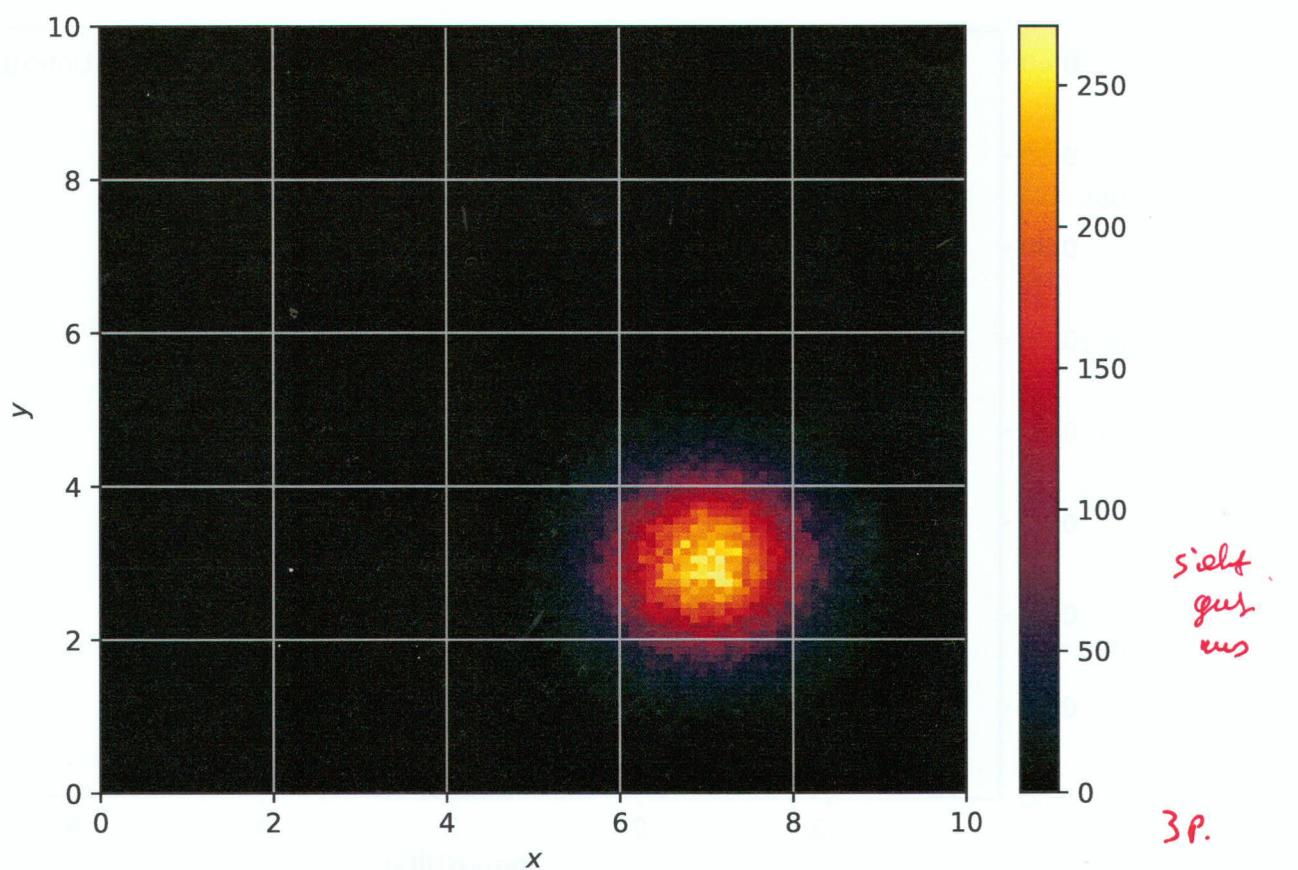
2P.

c) und d) Mit den in c) erzeugten, normalverteilten Hits folgt das in 2 zu sehende 2D-Histogramm für die gemessenen Ereignisse auf einem quadratischen Detektor.

1P.

'Ihr könnt ruhig etwas mehr aufschreiben in der PDF was ihr berechnet habt z.B. bei Teilaufgabe a'

-2P.



**Abbildung 2:** Histogramm der gemessenen Ereignisse auf einem quadratischen Detektor

- e) Für den Logarithmus der Anzahl der Hits folgt das in 3 zu sehende *Histogramm*.

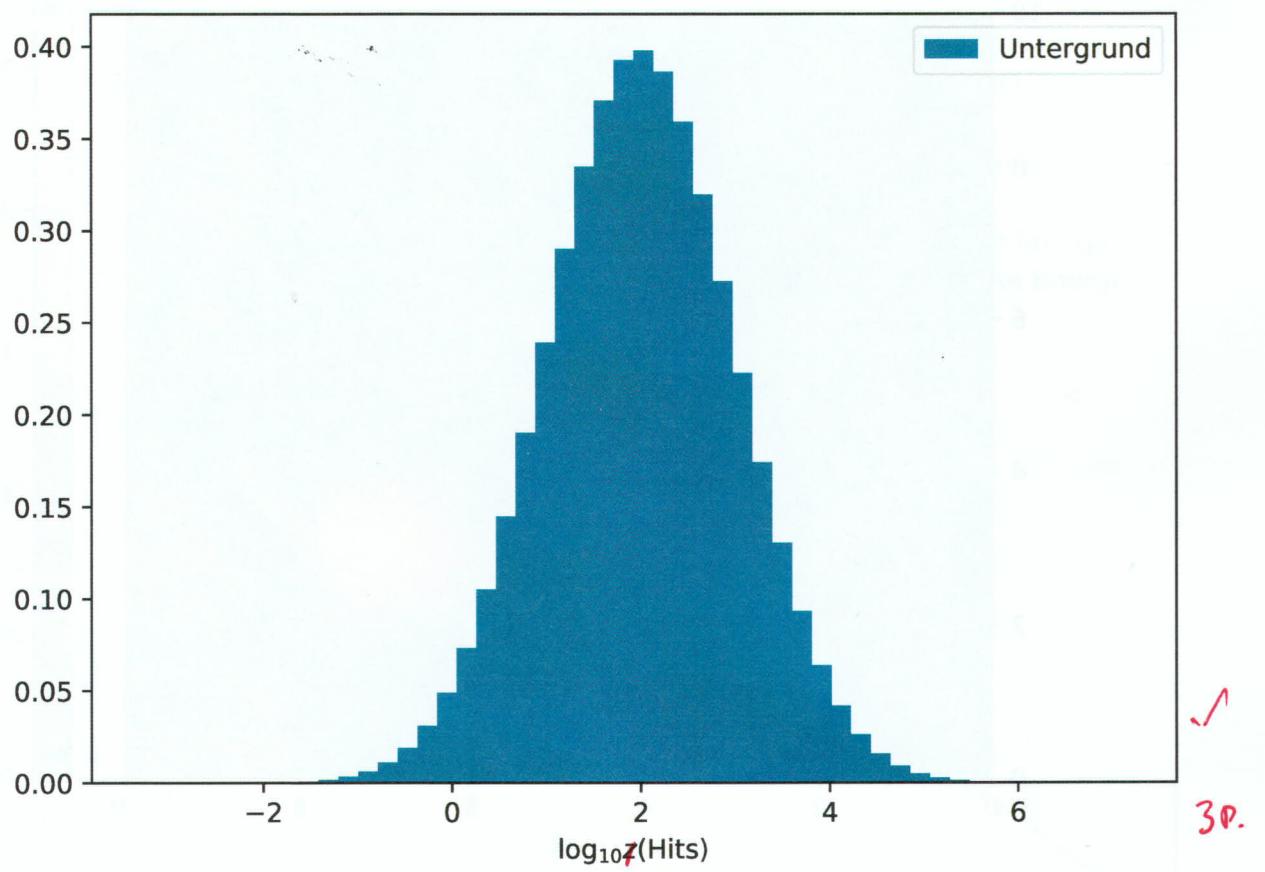


Abbildung 3: Histogramm für Logarithmus der Anzahl der Hits

Für die Orte der Untergrundereignisse folgt das in 4 zu sehende 2D-Histogramm.

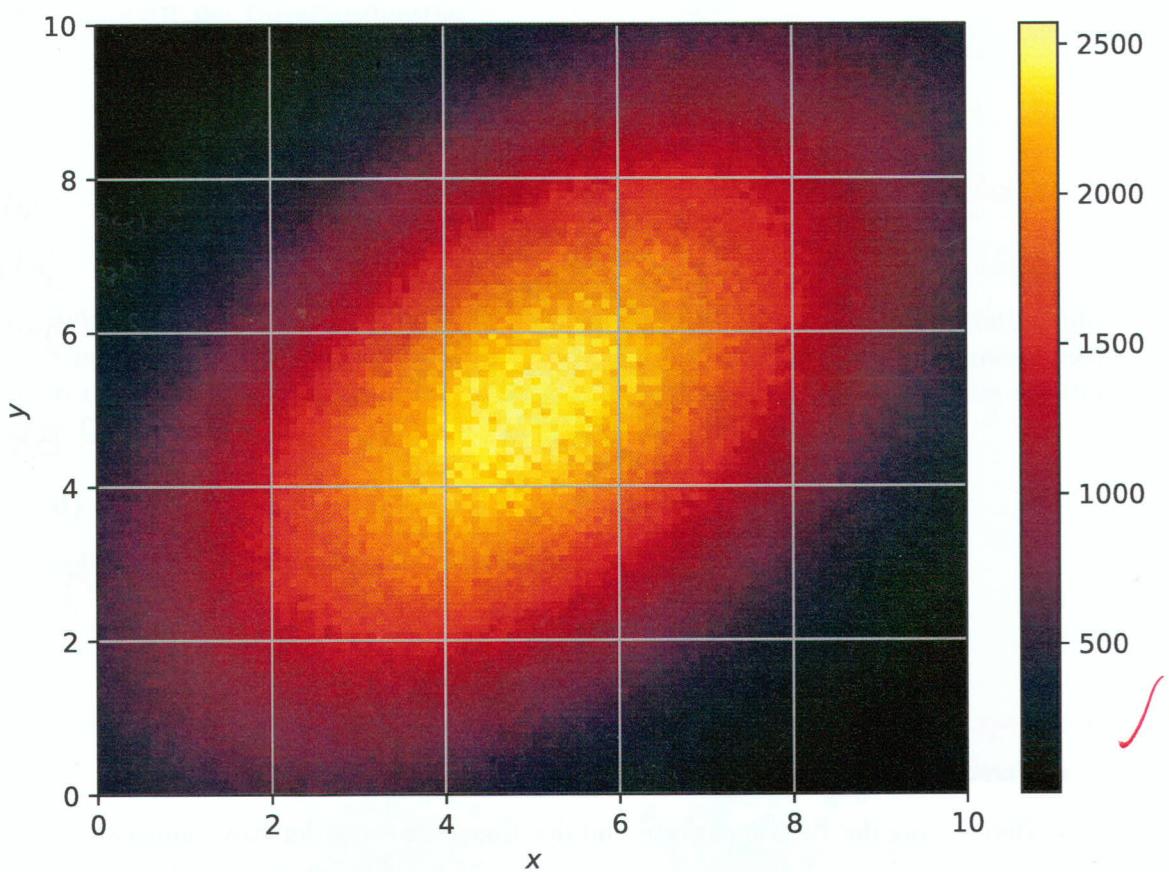


Abbildung 4: Histogramm für Logarithmus der Anzahl der Hits

## 1 Aufgabe 14: Hauptkomponentenanalyse

### 1.1 a)

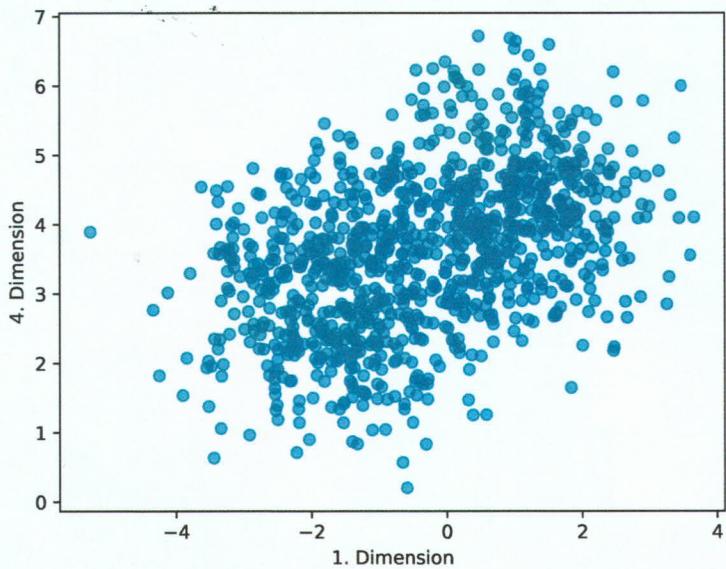
(Siehe Python-Datei) In Abbildung 5 ist die gegebene Verteilung für die 1. und die 4. Dimension gegeneinander aufgetragen.

*gegen die 2. oder? 1P.*

### 1.2 b)

Die Hauptkomponentenanalyse ist dazu da, um eine n-dimensionale Verteilung auf eine p-dimensionale Verteilung zu reduzieren ( $n > p$ ) oder auch eine mehrdimensionale Verteilung zu dekorrelieren, indem die Achsen transformiert werden. Hierbei sollen möglichst wenige Informationen verloren gehen, wobei man mehrere korrelierende Komponenten zu einer "Hauptkomponente" zusammenfasst. Reihenfolge der Hauptkomponentenanalyse:

- Zentrieren der Daten auf den Nullpunkt  
*Mittelwert*



wo seht ihr  
da jetzt die  
2 Populationen?

- 0,5 P.

(V)

Abbildung 5: Scatterplot.

- Berechnung der Kovarianzmatrix
- Berechnung der Eigenwerte und der Eigenvektoren der Kovarianzmatrix
- Wähle die  $k$  größten Eigenwerte und zugehörigen Eigenvektoren aus
- Bilde eine  $d \times k$  Matrix  $W$  mit den  $k$  Eigenvektoren als Spalten
- Wende  $W$  auf jede Zeile aus  $x$  aus  $X$  an  $x' = W^T \cdot x^T$

1.3 c)

1P.

(Siehe Python-Datei) Die Kovarianzmatrix:

$$\text{Cov} = \begin{pmatrix} 2.63 & 0.80 & 2.12 & -4.38 \\ 0.80 & 1.34 & 1.10 & -2.19 \\ 2.12 & 1.10 & 3.70 & -5.66 \\ -4.38 & -2.19 & -5.66 & 12.75 \end{pmatrix}$$

Die Eigenwerte der Kovarianzmatrix:

$$\lambda_1 = 17.52$$

$$\lambda_2 = 0.90$$

$$\lambda_3 = 1.00$$

$$\lambda_4 = 17.52$$

1P.

copy paste Fehler

Die Hauptkomponente mit dem größten Eigenwert (hier  $\lambda_1$ ) ist am signifikantesten, falls eine Reduktion der Dimensionen vorgenommen werden soll. Die anderen Dimensionen haben nur einen Eigenwert von etwas unter 1 und sind somit viel kleiner als der der ersten Dimension.

1P.

#### 1.4 d)

in Abbildung 6 ist der Scatterplot abgebildet.

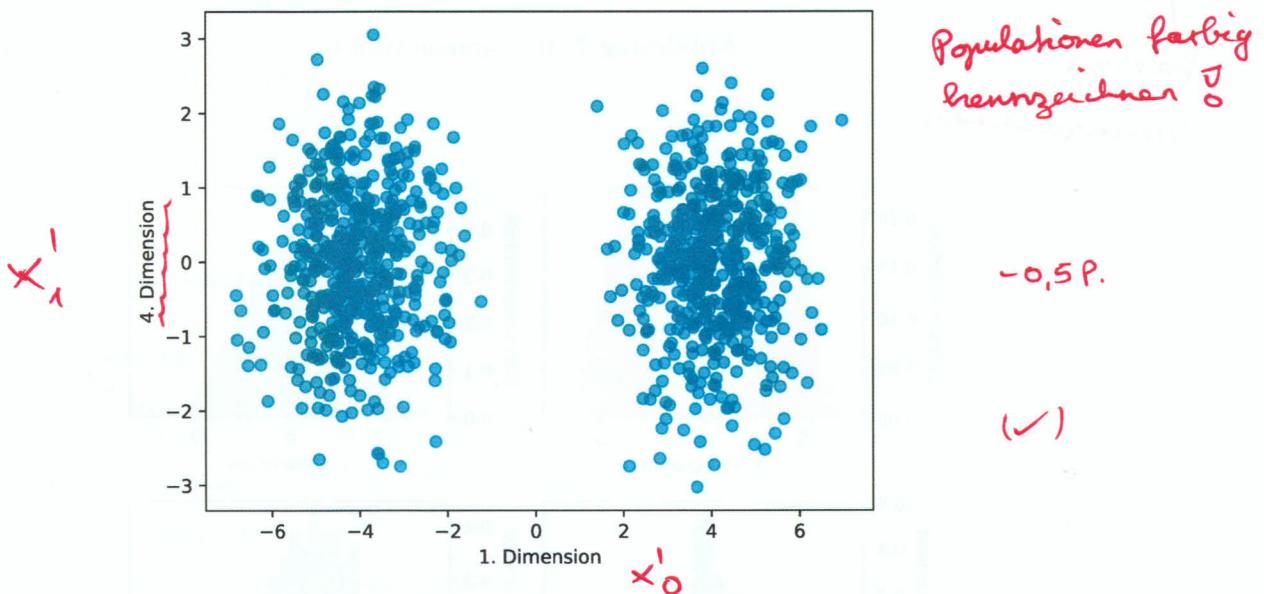
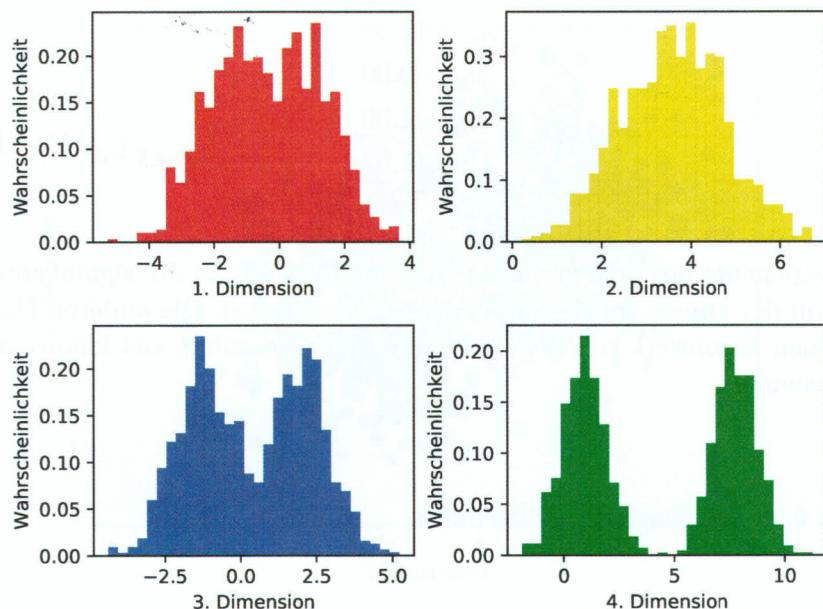


Abbildung 6: Scatterplot

In Abbildung 7 und 8 sind die Histogramme der Dimensionen vor und nach der PCA dargestellt.



Populations  
farbig  
brennzeichnen

Abbildung 7: Histogramm vorher

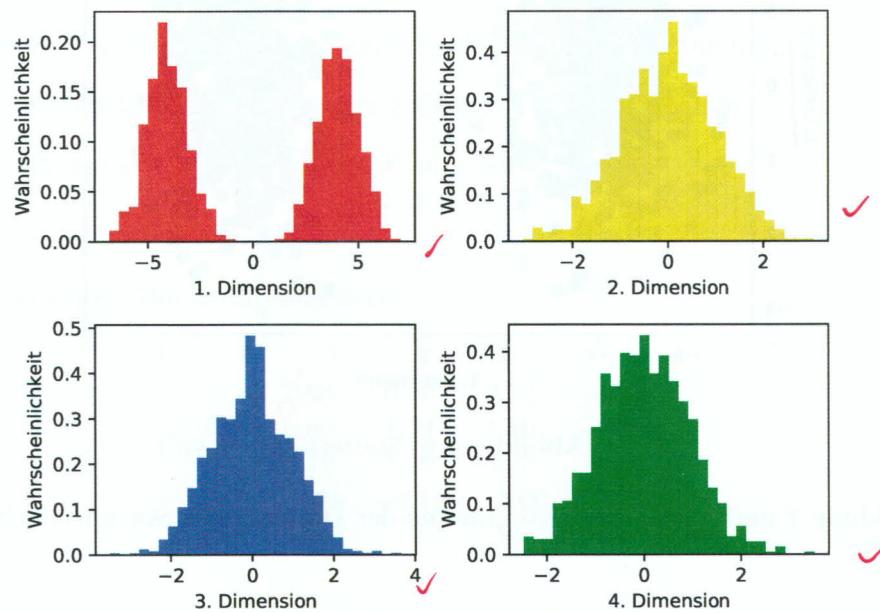


Abbildung 8: Histogramm nachher

1P.

# Code fuer Blatt05

Kolk, Sobolewski, Salewski

23. November 2018

```
.../B/7/Blatt05_Kolk_Sobolewski_Salewski/Aufgabe13.py
```

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # a)
6 gamma = 2.7 ✓
7 phi_0 = 1.7
8
9 def phi(x):
10     return phi_0 * x**(-gamma)
11
12 def Phi(x):
13     return phi_0/(gamma-1) * (1-x**(-(gamma-1)))
14
15 def Phi_inv(x):
16     return 1/(1-x)**(1/(gamma-1))
17
18 np.random.seed(10)
19 r = np.random.uniform(0,1, 100000)
20
21 energy = Phi_inv(r)
22 dfEnergy = pd.DataFrame({'Energy': energy})
23
24
25 # b)
26 def P(x):
27     return (1-np.exp(-x/2))**3
28
29 np.random.seed(42)
30 xx = np.random.uniform(0,1, 100000)
31
32 acceptance = xx < P(energy)
33 detected = energy[acceptance == True]
34
35 dfAcceptance = pd.DataFrame({'AcceptanceMask': acceptance})
36
37 x = np.linspace(1, 1000, 10000)
38
39 plt.hist(energy, density=True, bins=np.logspace(0,3), histtype='step', label='gezogene Energien')
40 plt.hist(detected, density=True, bins=np.logspace(0,3), histtype='step', label='detektierte
Energien')
41 plt.plot(x, phi(x), label=r'$\varphi(E)$')
42
43 plt.xlabel(r'$E \text{ :/ : } \mathrm{TeV}$')
44
45 plt.xscale('log')
46 plt.yscale('log')
47
48 plt.legend()
49 plt.tight_layout()
50 plt.savefig('b.pdf')
51
52 plt.clf()
53
54 # c)
55
56 np.random.seed(42)
```

```

57 def polar(size, Energy):
58     E=[]
59     while len(E)<size:
60         u1=np.random.uniform(0,1)
61         u2=np.random.uniform(0,1)
62         v1=2*u1-1
63         v2=2*u2-1
64         s=v1**2+v2**2
65         if s <= 1:
66             x1 = v1*np.sqrt(-2/s*np.log(s))
67             x2 = v2*np.sqrt(-2/s*np.log(s))
68             p1 = int(np.sqrt(2*Energy[len(E)-1])*x1+10*Energy[len(E)-1])
69             p2 = int(np.sqrt(2*Energy[len(E)-1])*x2+10*Energy[len(E)-1])
70
71         if p1>0:
72             E.append(p1)
73
74     return E
75
76 Hits=polar(10**5, energy)
77
78 dfHits = pd.DataFrame({'NumberOfHits' :Hits})
79
80 # d)
81
82 np.random.seed(25)
83 def detector(size, Hits):
84     x=[]
85     y=[]
86     while len(x)<size:
87         u1=np.random.uniform(0,1)
88         u2=np.random.uniform(0,1)
89         v1=2*u1-1
90         v2=2*u2-1
91         s=v1**2+v2**2
92         if s <= 1:
93             xx=v1*np.sqrt(-2/s*np.log(s))/(np.log10(Hits[len(x)-1]+1))+7
94             yy=v2*np.sqrt(-2/s*np.log(s))/(np.log10(Hits[len(x)-1]+1))+3
95             if 0 <= xx <= 10 and 0 <= yy <= 10:
96                 x.append(xx)
97                 y.append(yy)
98
99     return x, y
100
101 xx, yy=detector(10**5, Hits)
102 plt.grid()
103 plt.hist2d(xx,yy, bins=[100,100], range=[[0,10],[0,10]], cmap='inferno')
104 plt.colorbar()
105 plt.xlabel(r'$\log_{10}(\text{Hits})$')
106 plt.ylabel(r'$\log_{10}(E)$')
107 plt.savefig('d.pdf')
108
109 plt.clf()
110
111 # e)
112
113 np.random.seed(44)
114 size = int(1e7)
115
116 mu = 2
117 sig = 1
118
119 log10hits = np.random.normal(mu, sig, size)
120 hits = [ int(10**x) for x in log10hits ]
121
122 dfBackHits = pd.DataFrame({'NumberOfHits': hits})
123
124 plt.hist(log10hits, bins=50, density=True, label=r'Untergrund')
125 plt.xlabel(r'$\log_{10}(\text{Hits})$')
126 plt.legend()
127 plt.tight_layout()

```

```

128 plt.savefig('e_hits.pdf')
129
130 plt.clf()
131
132 np.random.seed(4)
133
134 def ort(mu, sig, rho, size):
135     x=[]
136     y=[]
137     while len(x)<size:
138         xx = np.random.normal(0, 1)
139         yy = np.random.normal(0, 1)
140
141         xx = np.sqrt(1-rho**2)*sig*xx+rho*sig*yy+mu
142         yy = sig*yy+mu
143
144         if 0 <= xx <= 10 and 0 <= yy <= 10:
145             x.append(xx)
146             y.append(yy)
147
148     return x, y
149
150 mu = 5
151 sig = 3
152 rho = 0.5
153
154 x, y = ort(mu, sig, rho, size)
155
156 plt.grid()
157 plt.hist2d(x,y, bins=[100,100], range=[[0,10],[0,10]], cmap='inferno')
158 plt.colorbar()
159 plt.xlabel(r'$x$')
160 plt.ylabel(r'$y$')
161
162 plt.savefig('e_detektor.pdf')
163 plt.clf()
164
165 dfBackX = pd.DataFrame({'x': x})
166 dfBackY = pd.DataFrame({'y': y})
167
168 dfBackground = pd.concat([dfBackHits, dfBackX, dfBackY], axis=1)
169 dfBackground = dfBackground.to_hdf('NeutrinoMC.hdf5', key='Background')

```

.../B/7/Blatt05\_Kolk\_Sobolewski\_Salewski/Aufgabe14.py

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import pandas as pd
4 from scipy.optimize import curve_fit
5 import uncertainties.unumpy as unp
6 from uncertainties import ufloat
7 from scipy.stats import stats
8 from sklearn.datasets.samples_generator import make_blobs
9 from sklearn.decomposition import PCA
10 from scipy import linalg
11
12 X, y = make_blobs(n_samples=1000, centers=2, n_features=4,
13                     random_state=0)
14
15
16 plt.scatter(X[:,0],X[:,1], alpha=0.7)
17 plt.xlabel('1. Dimension')
18 plt.ylabel('4. Dimension')
19 plt.savefig('Scatterplot.pdf')
20 plt.clf()
21
22 #zentrieren der Werte um den Nullpunkt
23 mue = np.array([np.mean(X[:,0]), np.mean(X[:,1]),
24                 np.mean(X[:,2]), np.mean(X[:,3])])
25 X_zentriert = X-mue
26

```

*mit c=y hätte ich die 2 Populationen noch verschieden farbig darstellen können*

```

27 pca = PCA(n_components=4)
28 pca.fit(X_zentriert)
29 X_pca = pca.transform(X_zentriert)
30
31
32 #Eigenwerte und Eigenvektoren der Kovarianzmatrix
33 Cov = pca.get_covariance()
34 print(Cov)
35 Eigenwerte, Eigenvektoren = linalg.eig(Cov)
36 print('unsortiert')
37 print(Eigenwerte)
38 #print(Eigenvektoren)
39
40
41 plt.scatter(X_pca[:,0],X_pca[:,1], alpha=0.7)
42 plt.xlabel('1. Dimension')
43 plt.ylabel('4. Dimension')
44 plt.savefig('Scatterplot_pca.pdf')
45 plt.clf()
46
47
48 plt.subplot(2, 2, 1)
49 plt.hist(X[:,0],color='r', density=True, bins=30, label='1. Dimension', alpha=0.7)
50 plt.ylabel('Wahrscheinlichkeit')
51 plt.xlabel('1. Dimension')
52 plt.tight_layout()
53 plt.subplot(2, 2, 2)
54 plt.hist(X[:,1],color='y', density=True, bins=30, label='2. Dimension', alpha=0.7)
55 plt.ylabel('Wahrscheinlichkeit')
56 plt.xlabel('2. Dimension')
57 plt.tight_layout()
58 plt.subplot(2, 2, 3)
59 plt.hist(X[:,2],color='b', density=True, bins=30, label='3. Dimension', alpha=0.7)
60 plt.ylabel('Wahrscheinlichkeit')
61 plt.xlabel('3. Dimension')
62 plt.tight_layout()
63 plt.subplot(2, 2, 4)
64 plt.hist(X[:,3],color='g', density=True, bins=30, label='4. Dimension', alpha=0.7)
65 plt.ylabel('Wahrscheinlichkeit')
66 plt.xlabel('4. Dimension')
67 plt.savefig('Histogramm_vorher.pdf')
68 plt.tight_layout()
69 plt.clf()
70
71
72 plt.subplot(2, 2, 1)
73 plt.hist(X_pca[:,0],color='r', density=True, bins=30, label='1. Dimension', alpha=0.7)
74 plt.ylabel('Wahrscheinlichkeit')
75 plt.xlabel('1. Dimension')
76 plt.tight_layout()
77 plt.subplot(2, 2, 2)
78 plt.hist(X_pca[:,1],color='y', density=True, bins=30, label='2. Dimension', alpha=0.7)
79 plt.ylabel('Wahrscheinlichkeit')
80 plt.xlabel('2. Dimension')
81 plt.tight_layout()
82 plt.subplot(2, 2, 3)
83 plt.hist(X_pca[:,2],color='b', density=True, bins=30, label='3. Dimension', alpha=0.7)
84 plt.ylabel('Wahrscheinlichkeit')
85 plt.xlabel('3. Dimension')
86 plt.tight_layout()
87 plt.subplot(2, 2, 4)
88 plt.hist(X_pca[:,3],color='g', density=True, bins=30, label='4. Dimension', alpha=0.7)
89 plt.ylabel('Wahrscheinlichkeit')
90 plt.xlabel('4. Dimension')
91 plt.savefig('Histogramm.pdf')
92 plt.tight_layout()
93 plt.clf()

```

mit  
histtype='step'  
wird es noch  
übersichtlicher