Einfuehrung_Statistik

April 28, 2023

1 Einführung Statistik

Dieses Jupyter-Notebook enthält die Python-Programme aus dem Skript *Einführung Statistik*.

1.1 Deskiptive Statistik

1.1.1 Mittelwert

[1]:
```
import numpy as np

data = np.array([1,2,4,5,7,9,11,13,15,16,19,19,21])
x_quer = data.mean()
print(x_quer)
```

```
10.923076923076923
```

1.1.2 Median

[2]:
```
import numpy as np

data = np.array([1,2,4,5,7,9,11,13,15,16,19,19,21])
x_tilde = np.median(data)
print(x_tilde)
```

```
11.0
```

1.1.3 Quantile

[3]:
```
import numpy as np
# Falls Fehlermeldungen: Änderungen beim Argument "method" in dieser Funktion␣
 ↪erst ab Numpy Version 1.22

data = np.array([1,2,4,5,7,9,11,13,15,16,19,19,21])
p = 0.7

if np.version.version >= "1.22":
    x_p_groesser = np.quantile(data, p, method="higher")
    x_p_gleich = np.quantile(data, p, method="nearest")
    x_p_interpol = np.quantile(data, p, method="interpolated_inverted_cdf")
```

```
    print(x_p_groesser)
    print(x_p_gleich)
    print(x_p_interpol)
else:
    print("Numpy-Version muss mindestens 1.22 sein")
    print(np.version.version)
```

```
Numpy-Version muss mindestens 1.22 sein
1.21.5
```

1.1.4 Varianz

[4]:
```
import numpy as np

sample = np.array([1,2,4,5,7,9,11,13,15,16,19,19,21])
print(((sample**2).sum()/13-sample.mean()**2)*13/12)
print(sample.var(ddof=1)) #ddof=delta degrees of freedom
```

```
46.576923076923094
46.57692307692307
```

1.1.5 Standardabweichung

[5]:
```
import numpy as np

sample = np.array([1,2,4,5,7,9,11,13,15,16,19,19,21])

print(np.sqrt(((sample**2).sum()/13-sample.mean()**2)*13/12))

print(sample.std(ddof=1))
```

```
6.824728791455606
6.8247287914556045
```

1.1.6 Kovarianz

[6]:
```
import numpy as np

sample_x = np.array([1,2,4,5,7,9,11,13,15,16,19,19,21])
sample_y = np.array([7,13,17,23,35,47,59,77,91,113,129,190,210])

x_bar = sample_x.mean()
y_bar = sample_y.mean()

print((1/13*(sample_x*sample_y).sum() - x_bar*y_bar)*13/12)
print(np.cov(sample_x, sample_y, ddof=1))
```

```
427.89743589743586
[[  46.57692308  427.8974359 ]
 [ 427.8974359  4428.85897436]]
```

1.2 Diskrete Verteilungen

1.2.1 Binomialverteilung

[7]:
```
import scipy.special
print(scipy.special.binom(10,7))
```

```
120.0
```

[8]:
```
from scipy.stats import binom
print(binom.pmf(5,10,0.4))
```

```
0.20065812479999992
```

[9]:
```
from scipy.stats import binom
print(binom.cdf(5,10,0.4))
```

```
0.8337613824
```

[10]:
```
from scipy.stats import binom
print(1-binom.cdf(7,10,0.4))
```

```
0.01229455359999998
```

1.2.2 Hypergeometrische Verteilung

[11]:
```
from scipy.stats import hypergeom

k,N,M,n = 3,6,49,6
print(hypergeom.pmf(k, M, n, N))
```

```
1.765040386687012e-02
```

[12]:
```
from scipy.stats import hypergeom

k,N,M,n = 4,10,40,15

print(hypergeom.pmf(k, M, n, N))
```

```
2.851866897357771e-01
```

1.2.3 Poissonverteilung

[13]:
```
from scipy.stats import poisson

print(poisson.pmf(3,5))
```

```
1.403738958142805e-01
```

[14]:
```
from scipy.stats import poisson

print(1-poisson.cdf(170,150))
```

```
0.049365532495901254
```

1.3 Stetige Verteilungen

1.3.1 Normalverteilung

[15]:
```
from scipy.stats import norm

x = 185
mu = 178
sigma = 8
print(1-norm.cdf(x, mu, sigma))
```

```
0.19078695285251068
```

[16]:
```
from scipy.stats import norm

p = norm.cdf([4.99,5.02], 5, .01)
print(p[1]-p[0])
```

```
0.8185946141203563
```

[17]:
```
import numpy as np
from scipy.stats import binom
from scipy.stats import norm

n, p, k = 1000, 0.4, 390
s = np.sqrt(n*p*(1-p)) # Standardabweichung
mu = n*p #Erwartungswert

print("Approx. erlaubt") if s>3 else print("Approx. nicht erlaubt")

print(binom.cdf(k,n,p)) # Binomialverteilung
print(norm.cdf(k+0.5,mu,s)) # Approx. Normalverteilung
```

```
Approx. erlaubt
0.27031962529018677
```

0.2698646599190479

1.3.2 t-Verteilung - Konfidenzintervalle

[18]:
```
import numpy as np
from scipy.stats import sem

stichprobe = np.array([12, 17, 18, 18, 22, 17, 15, 15, 19, 19, 22])
print(np.std(stichprobe, ddof=1)/np.sqrt(11))
print(sem(stichprobe))
```

0.8971948939254784
0.8971948939254784

[19]:
```
import numpy as np
from scipy.stats import sem, t

stichprobe = np.array([12, 17, 18, 18, 22, 17, 15, 15, 19, 19, 22])
s = np.std(stichprobe, ddof=1)
x_quer = stichprobe.mean()
serror = sem(stichprobe)

T = t.isf(0.025,10)
ugrenze = x_quer - T * serror
ogrenze = x_quer + T * serror
print(ugrenze, ogrenze)

# Mit scipy.stats.t.interval
print(t.interval(0.95, df=10, loc=x_quer, scale=serror))
```

15.637288835423716 19.635438437303556
(15.637288835423716, 19.635438437303556)

1.4 Hypothesentests

1.4.1 t-Tests

[20]:
```
import numpy as np
from scipy.stats import ttest_1samp

stichprobe = np.array([5.1,6.0,4.8,5.0,5.6,6.1,3.9,5.8,5.8,5.7,6.0])
print(ttest_1samp(stichprobe, popmean=5.7))
```

Ttest_1sampResult(statistic=-1.3026638133042359, pvalue=0.22188898434318105)

[21]:
```
import numpy as np
from scipy.stats import ttest_1samp
from scipy.stats import t
```

```
stichprobe = np.array([5.1,6.0,4.8,5.0,5.6,6.1,3.9,5.8,5.8,5.7,6.0])

t_stat = ttest_1samp(stichprobe, popmean=5.7)[0]
print(t.cdf(t_stat, df=10)*2)
```

```
0.22188898434318105
```

[22]:
```
import numpy as np
from scipy.stats import ttest_1samp

stichprobe = np.array([98,100,103,92,101,101,96,95,110,102,99,98,102,105])

print(ttest_1samp(stichprobe, popmean=100, alternative="greater"))
```

```
Ttest_1sampResult(statistic=0.12005172573287078, pvalue=0.45313885267457843)
```

[23]:
```
import numpy as np
from scipy.stats import ttest_1samp
from scipy.stats import t

stichprobe = np.array([98,100,103,92,101,101,96,95,110,102,99,98,102,105])

t_stat = ttest_1samp(stichprobe, popmean=100, alternative="greater")[0]
print(1-t.cdf(t_stat, df=13))
```

```
0.45313885267457843
```

[24]:
```
import numpy as np
from scipy.stats import ttest_ind

stichprobe_IT  = np.array([110,130,122,130,130,110,135,120,150,120])
stichprobe_BWL = np.array([120,110,110,120,105,130,105,125,90])

print(ttest_ind(stichprobe_IT, stichprobe_BWL))
```

```
Ttest_indResult(statistic=2.3147231780569513, pvalue=0.0333900746665659)
```

[25]:
```
import numpy as np
from scipy.stats import ttest_1samp

stichprobe_vorher  = np.array([150,180,162,160,175,210,187,160,155,160])
stichprobe_nachher = np.array([145,180,155,155,180,190,170,155,160,150])


print(ttest_1samp(stichprobe_nachher-stichprobe_vorher, popmean=0,␣
 ↪alternative="less"))
```

```
Ttest_1sampResult(statistic=-2.2572050843131675, pvalue=0.025203155262256305)
```

1.4.2 F-Test

[26]:
```python
import numpy as np
from scipy.stats import f

stichprobe1 = np.array([110,130,122,130,130,110,135,120,150,120])
stichprobe2 = np.array([120,110,110,120,105,130,105,125,90])

df_zaehler = stichprobe2.size-1
df_nenner = stichprobe1.size-1
F = stichprobe2.var(ddof=1)/stichprobe1.var(ddof=1)
p_value = f.sf(F, df_zaehler, df_nenner)
print(f"F-statistics = {F}, p-value={p_value}")
```

```
F-statistics = 1.0399892646269457, p-value=0.4724541917012181
```

1.4.3 χ^2-Test

[27]:
```python
import numpy as np
from scipy.stats import chi2_contingency

observed = np.array([[355,420,172], [325,380,348]])
print(chi2_contingency(observed))
```

```
(57.43609818078531, 3.372172240731047e-13, 2, array([[321.98, 378.8 , 246.22],
       [358.02, 421.2 , 273.78]]))
```

1.5 Lineare Regression

[28]:
```python
import pandas as pd
from scipy.stats import linregress

# Daten importieren
data = pd.read_csv("https://raw.githubusercontent.com/troescherw/datasets/
 ↪master/Lernzeit_Punkte.csv")

model =linregress(x=data.Stunden, y=data.Punkte)
print(f"beta_0 = {model[1]}")
print(f"beta_1 = {model[0]}")
print(f"r = {model[2]}")
print(f"R^2 = {model[2]**2}")
```

```
beta_0 = 2.4836734053731817
beta_1 = 9.775803390787473
r = 0.9761906560220887
R^2 = 0.9529481969048358
```

1.6 Der p-Wert

[29]:
```
from scipy.stats import hypergeom

p = hypergeom.pmf(4,10,5,5) + hypergeom.pmf(5,10,5,5)
print(f"p-Wert = {p}")
```

```
p-Wert = 0.10317460317460328
```