Histogram

1. V Pythonu vytvořte histogram z naměřených hodnot uložených v souboru `data.dat`. Nalezněte optimální šířku binu.

Práce s histogramy v Pythonu:

- vytvoření histogramu

```
hist,bin_edges=np.histogram(data,bins=nbins,density='True')
```

pole obsahující počty hodnot v jednotlivých binech

pole obsahující hranice binů

data, ze kterých vytvoříme histogram

počet binů

normalizace histogramu

Histogram

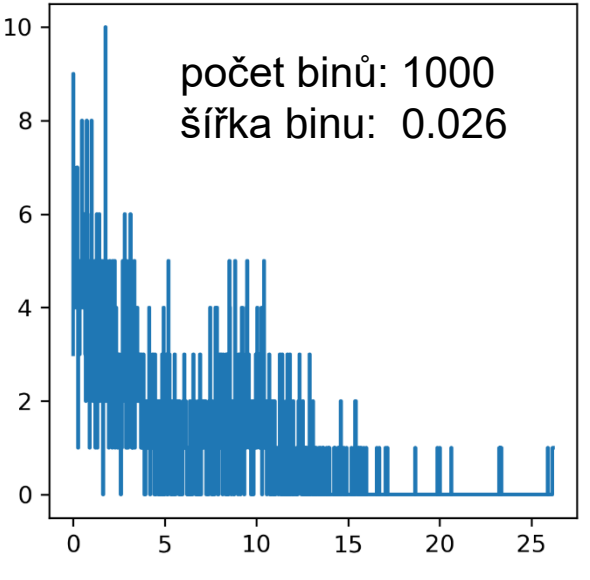
histogram.py

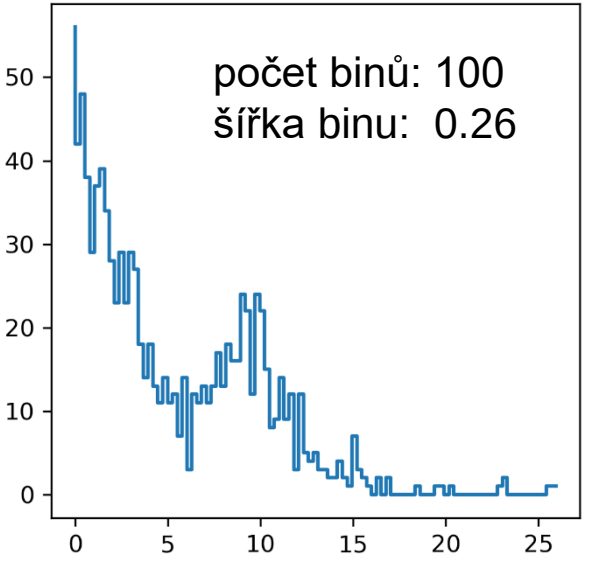
```python
import numpy as np
import matplotlib.pyplot as plt

nbins=10 #pocet binu
data=np.loadtxt('data.dat') #nacteni dat ze souboru data.dat
hist,bin_edges=np.histogram(data,bins=nbins) #vytvoreni histogramu

fig,ax=plt.subplots(figsize=(4,4))
ax.step(bin_edges[0:nbins],hist) #nakresleni histogramu

print('pocet dat:',np.size(data))
print('pocet binu: ',nbins)
print('min. hodnota:',np.min(data))
print('max. hodnota:',np.max(data))
print('sirka binu: ',(np.max(data)-np.min(data))/nbins)
```

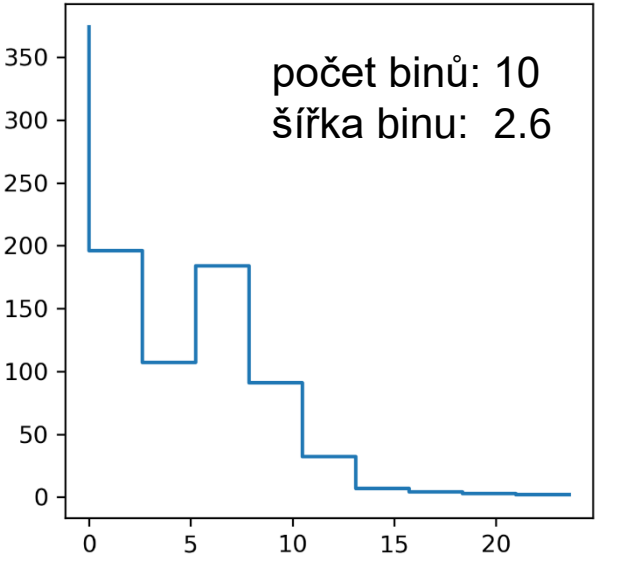
počet dat: 1000
minimum: 0.004
maximum: 26.198

Excel $m = \lceil \sqrt{N} \rceil \approx 32$

Sturges $m = \left\lceil \frac{\log N}{\log 2} + 1 \right\rceil \approx 11$







Histogram – šířka binu

`histogram-auto.py`

Algoritmus pro nalezení optimální šířky binu
(Shimazaki and Shinomoto, Neural.Comput. (2007), 19(6), p. 1503 – 1527)

1. Zvol počet binů m a vypočítej šířku binu $\Delta = (x_{\max} - x_{\min})/m$ a vyrob histogram.

2. Vypočítej:

   - odhad střední hodnoty výšky sloupečků histogramu: $\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} n_i$

   - odhad rozptylu výšek sloupečků histogramu: $\hat{\sigma}^2 = \frac{1}{m}\sum_{i=1}^{m} (n_i - \hat{\mu})^2$

3. Vypočítej **ztrátovou funkci**: $C(\Delta) = \frac{2\hat{\mu} - \hat{\sigma}^2}{\Delta^2}$

Opakuj pro různé počty binů m (tj. různé Δ).

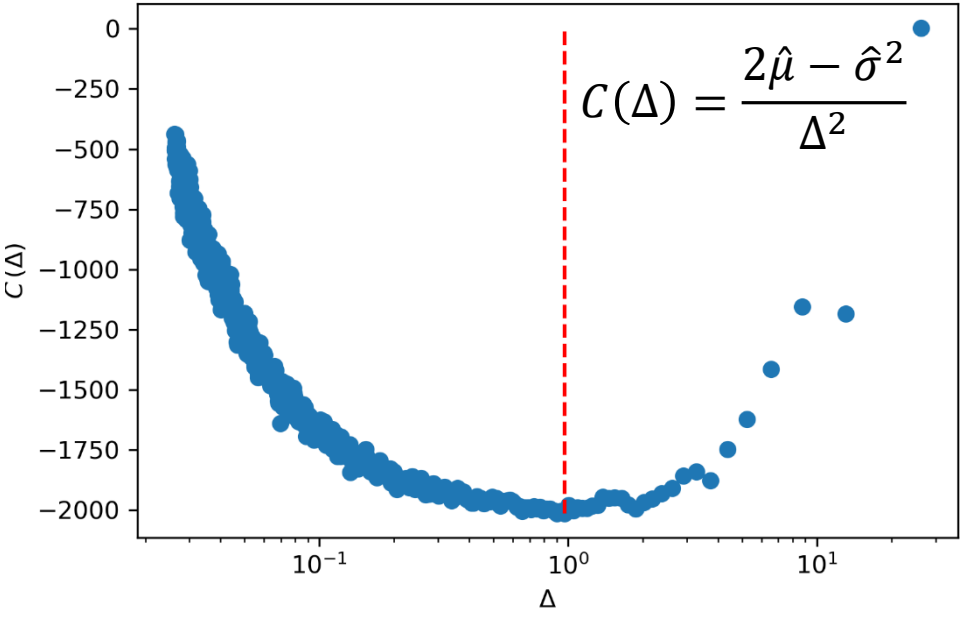
4. Vyber takové Δ, pro které je C minimální.

Histogram – šířka binu

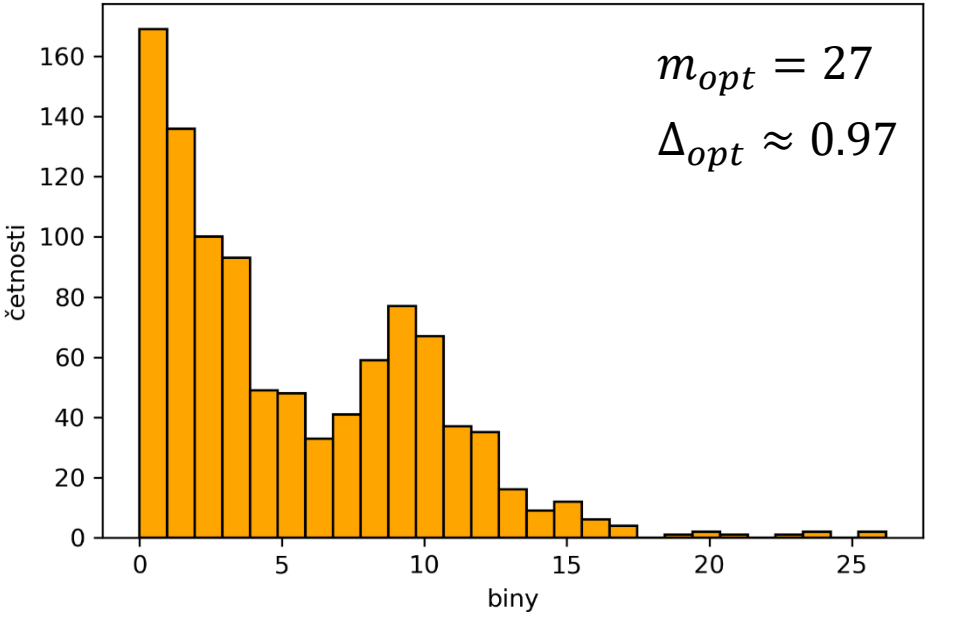
`histogram-auto.py`

Algoritmus pro nalezení optimální šířky binu
(Shimazaki and Shinomoto, Neural.Comput. (2007), 19(6), p. 1503 – 1527)

ztrátová funkce



optimální histogram



Histogram + Monte Carlo simulace

`hist-simulate.py`

2. Hodnoty v souboru `data.dat` odpovídají situaci, kdy v experimentu mohou být detekovány dva typy událostí:
   - v 3/4 případů exponenciální rozpad s časovou konstantou $\tau = 4$
   - v 1/4 případů výběr z normálního rozdělení s očekávanou hodnotou $\mu = 10$ a standardní odchylkou $\sigma = 2$

   V Pythonu proveďte simulaci tohoto experimentu a nasimulovaná data uložte do histogramu.

Generátor n náhodných čísel v Pythonu:

- rovnoměrné rozdělení $u \in U(0,1)$ `u=np.random.random_sample(n)`
- normální rozdělení $u \in N(\mu,\sigma)$ `u=np.random.normal(mu,sigma,n)`
- exponenciální rozdělení $u \in E(\tau)$ `u=np.random.exponential(tau,n)`
- binomické rozdělení $u \in B(N,p)$ `u=np.random.binomial(N,p,n)`
- Poissonovo rozdělení $u \in P(\nu)$ `u=np.random.poisson(nu,n)`

Histogram + Monte Carlo simulace

`hist-simulate.py`

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

P=0.75
mu=10
sigma=2
tau=4
Ntot=1000
nbins=100
data=np.empty(Ntot)

for i in range(1,Ntot):
    r=np.random.random_sample()
    if r>P:
        data[i]=np.random.normal(mu,sigma,1)
    else:
        data[i]=np.random.exponential(tau,1)
hist,bin_edges=np.histogram(data,bins=nbins)

fig,ax=plt.subplots()
ax.step(bin_edges[0:nbins],hist) #nakresleni histogramu
ax.set_xlabel('$n_k$',fontsize=14)
ax.set_ylabel('$k$',fontsize=14)
```