Topics:

→ simple, visual

① QQ-plot → Is the data Gaussian?

↳ why, how, code, variations

② Transformations → convert data to Gaussian dist.

↳ Box-Cox : code ✓

③ Mode; Bi-modal distributions

Ops:

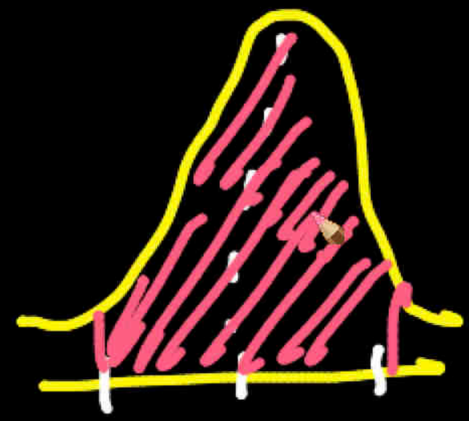
Ques → tab

↳ "END! ---"

chat → interactivity, Y/N

C.I using CLT

$$\left[\mu - 1.96\sigma \quad \mu + 1.96\sigma\right]$$



$$\bar{X}_n \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right) \text{ as } n \to \infty$$

QQ-plot

height

X

$x_1, x_2, \ldots x_n$

m, s

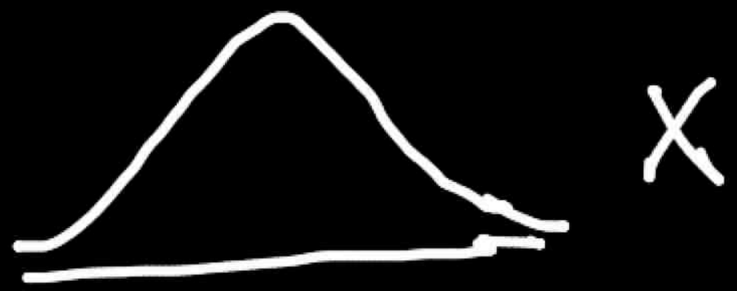
200 (let)

obs

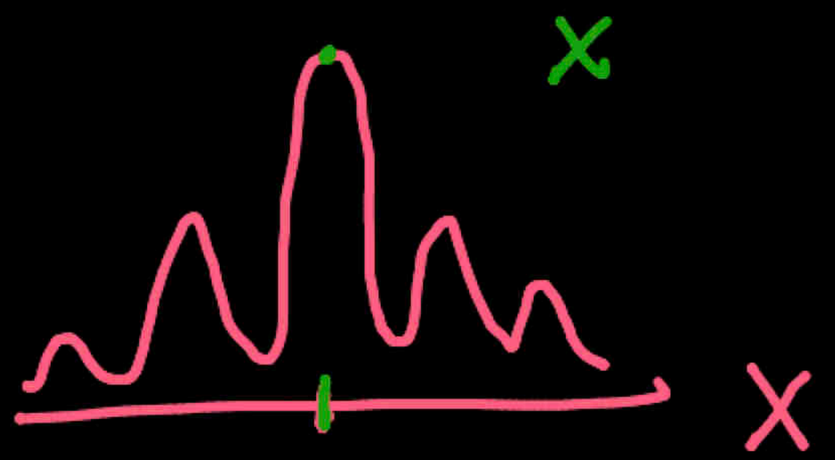
μ, σ

Q Is X Gaussian dist?

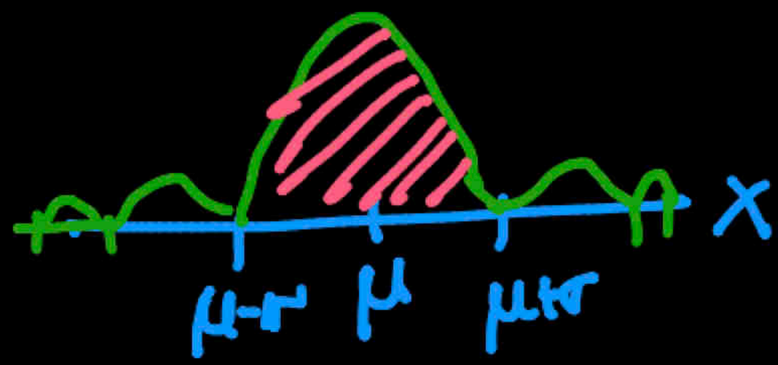
- → PDF

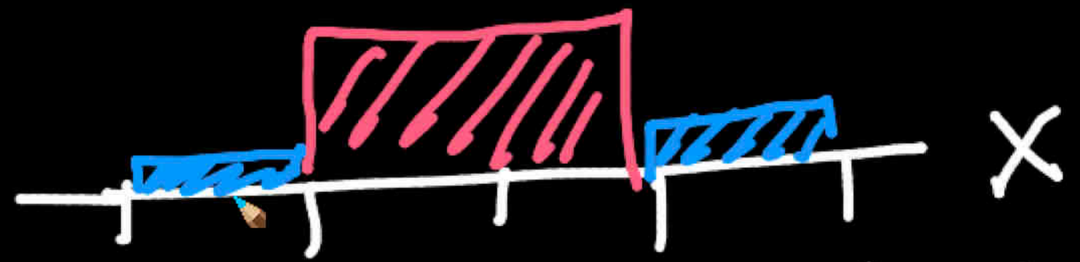


- → mean = mode = median



- → 68-97-99 rule





$\mu-2\sigma$ $\mu-\sigma$ μ $\mu+\sigma$ $\mu+2\sigma$

KS-test (later)

X : $x_1, x_2 \dots x_n$ ✓

x_1, x_2 ...

M, S

$N(M, S)$

equations

$\mu \approx M$

$\sigma \approx S$

Max gap

CDF

1

0.85

0

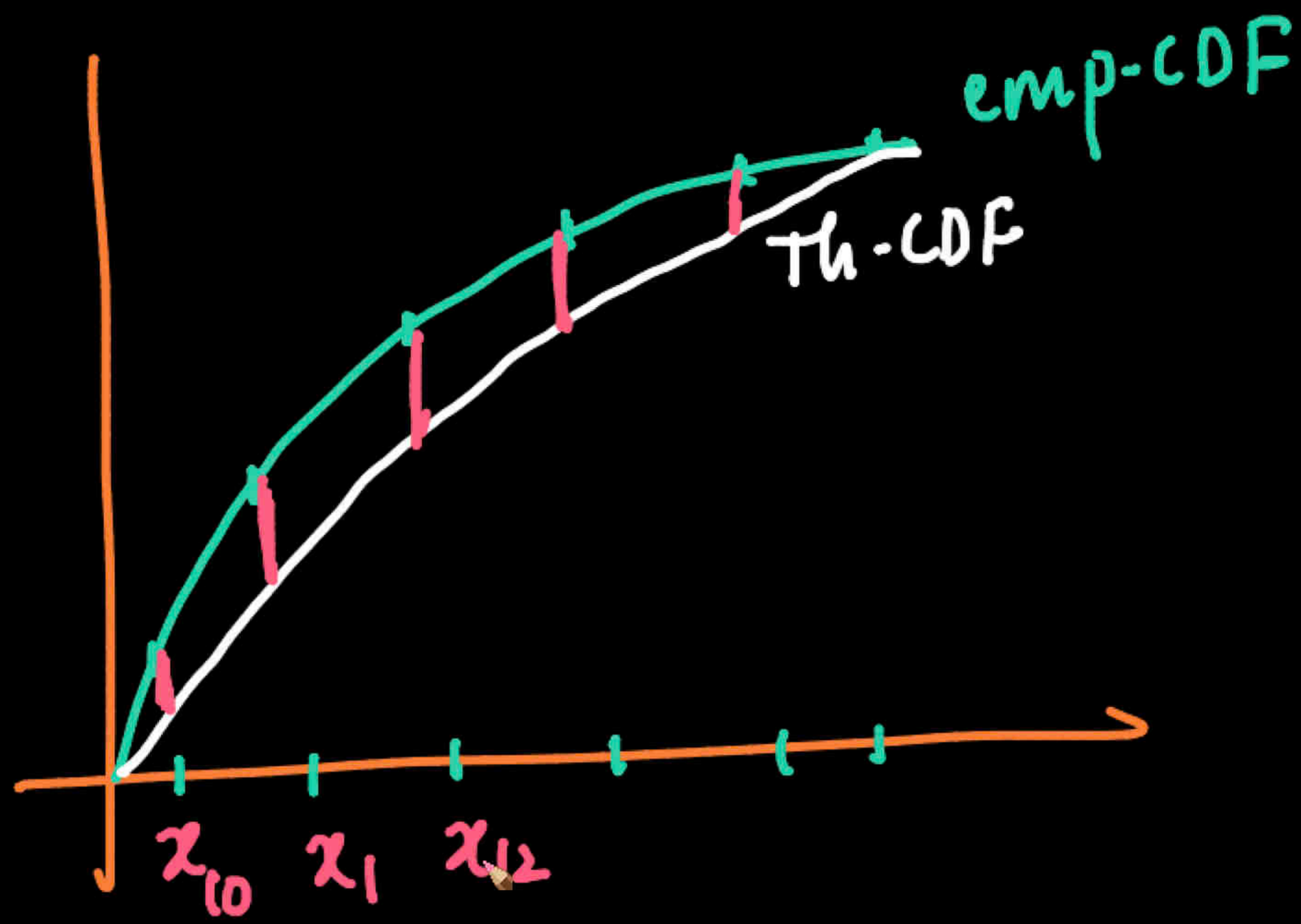
x_2

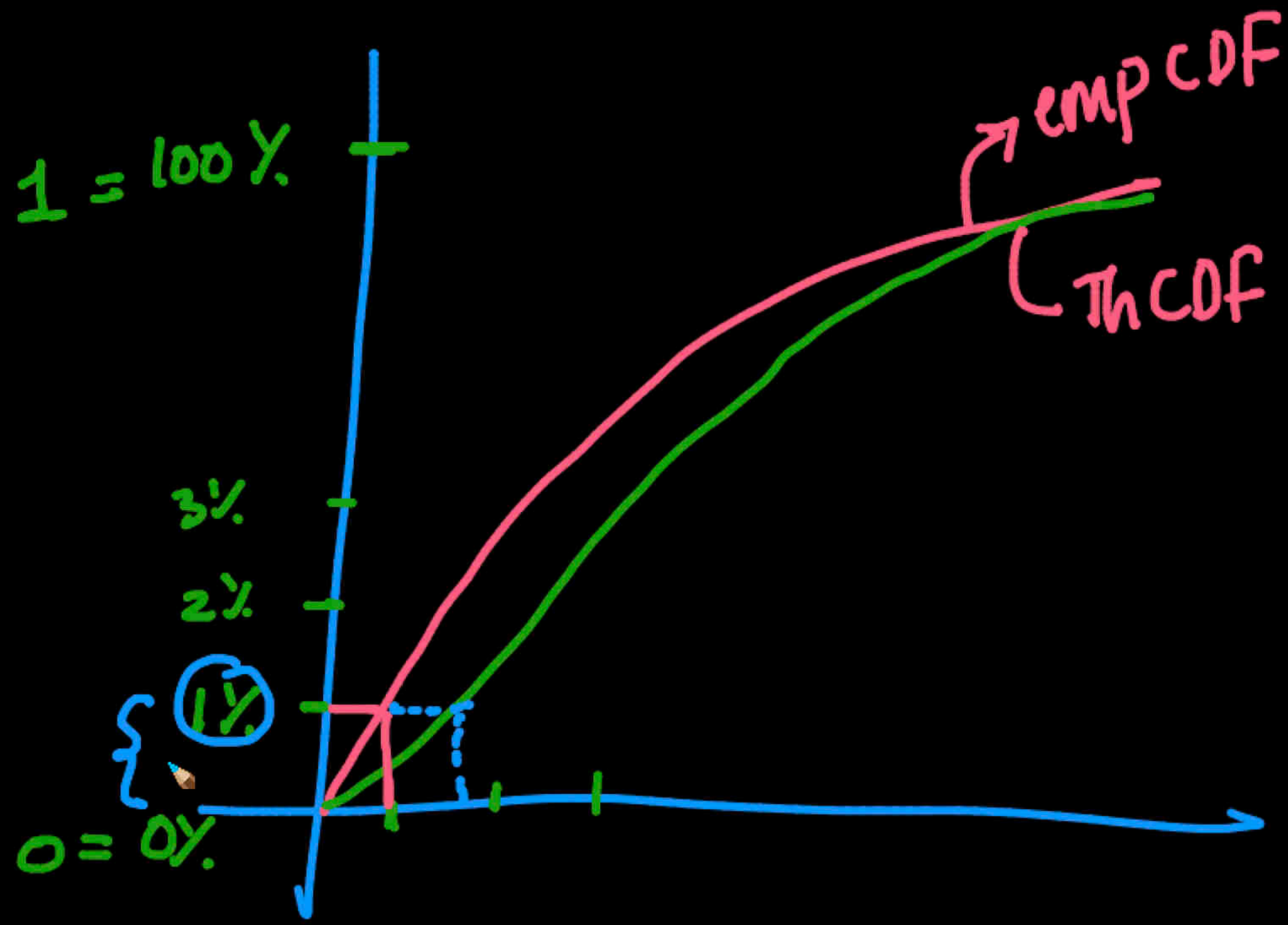
X

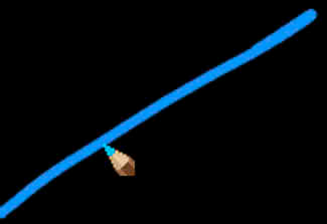
empirical CDF

Th. CDF

$P(X \le x_i)$


emp-CDF
Th-CDF
x_{10}
x_1
x_{12}





$X \Rightarrow x_1, x_2 \ldots\ldots x_n \rightarrow m, s$

Standardize (Not Mandatory)

$\left(\frac{x_i - m}{s}\right)$

Normal

$N(0,1) = Z(0,1)$

empirical

$X \rightarrow x_1, \ldots, x_n$

M, S ✓ → PPF → PDF, CDF

(VS)

Normal (M, S)

D_1 1.2 → tp_1^e

D_2 1.6 → tp_2^e

D_{10} 64.2 → tp_{10}^e

✓

1.25 → tp_1^T

1.68 → tp_2^T

2.18

65.1 → Tp_{10}^T

PPF

Normal(m,s)

$P(x \leq \text{val}) = \text{py}$

CDF(val)

(tp_i^e, tp_i^t) for all

$i: 1$ to n

Th: percentiles

QQ-plot

$y = x$: eqn of line

1.68

1.25

45°-degree

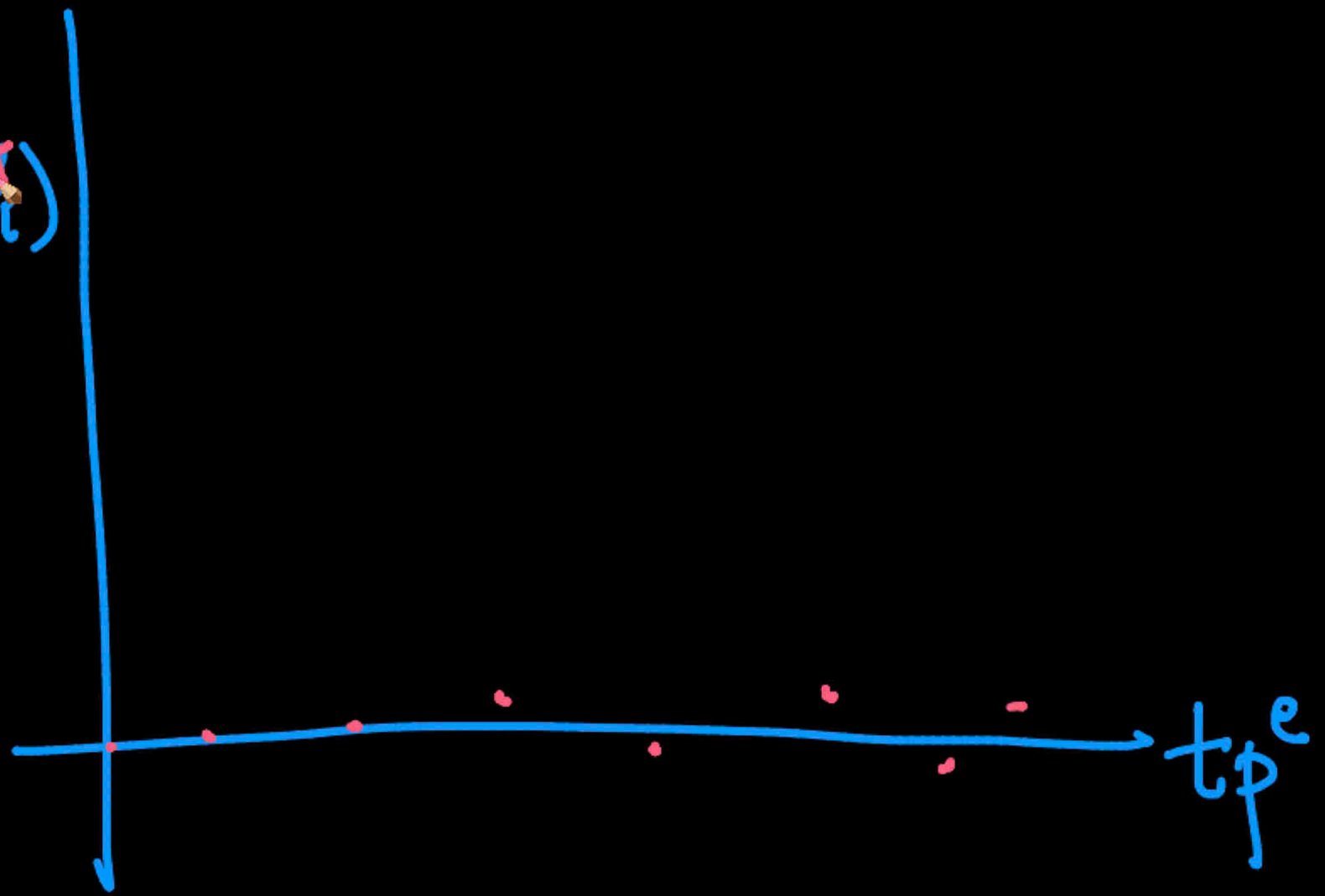
1.2 1.6

1 2 . . . . n

emp. percentiles

diff

(tp_i, tp_j)



Dis-adv:

↳ [only visual ✓

↳ how confident

$\widetilde{x_1 \dots x_n} \sim N(M, S)$

↳ Hyp-testing

QQ-plot

- $x_1, \ldots x_n$ vs Gaussian
- $x_1 \ldots x_n$ vs Poisson

X: $x_1 \ldots x_n$ → emp. per-x

Y: $y_1, y_2 \ldots y_m$ → emp-per-y

Th. per

$N(\mu, \sigma)$

$y = x$

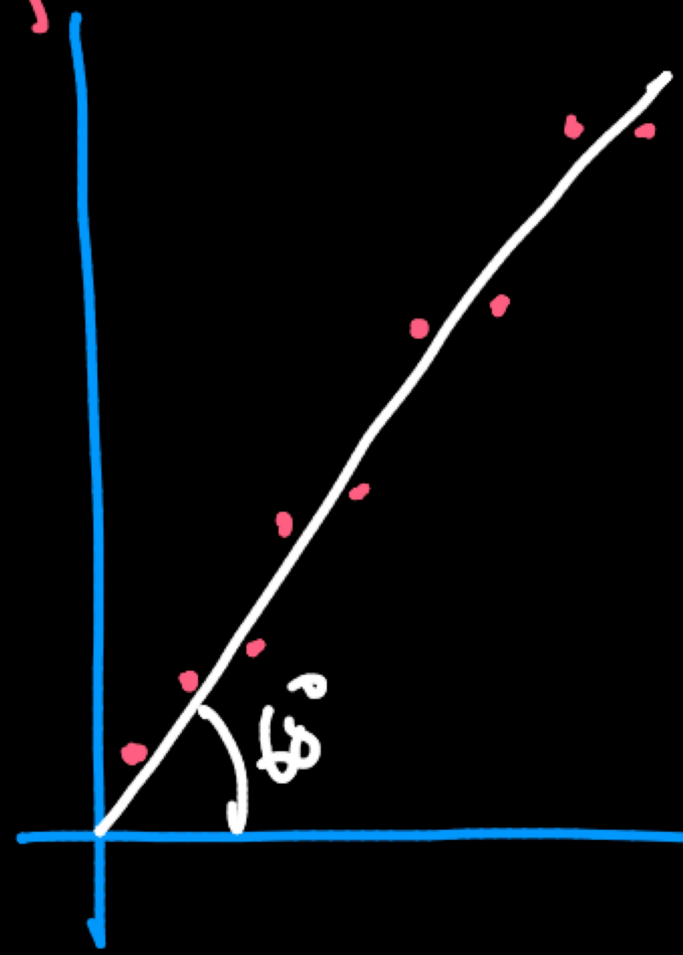
45°

emp. per

$\frac{}{N}(\mu, \sigma)$

Normal

Th. per



emp. per

⇒ Same dist ✓

⇒ params are diff

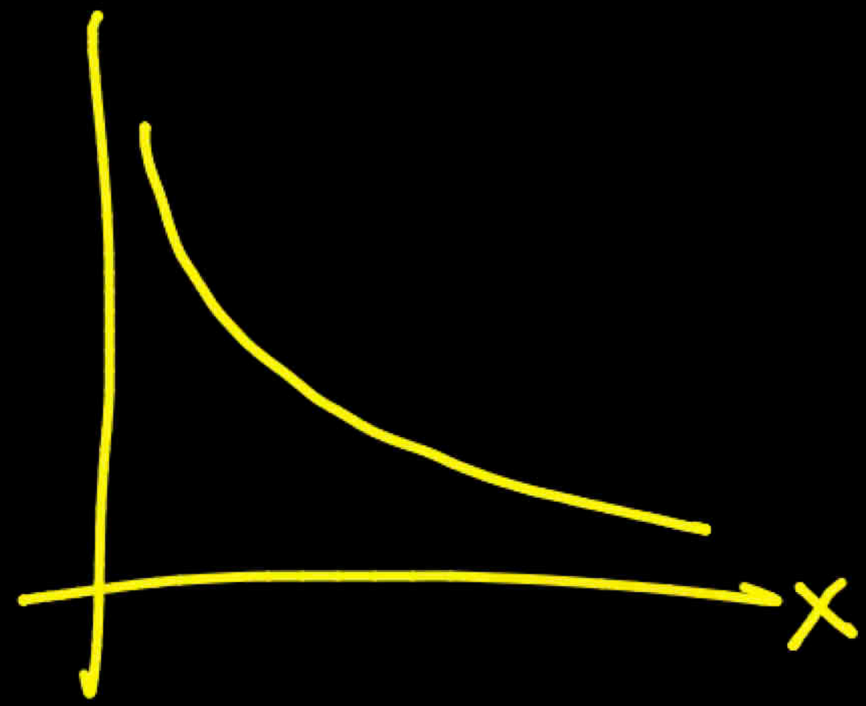
$x_1 \cdots x_n$

$\downarrow$

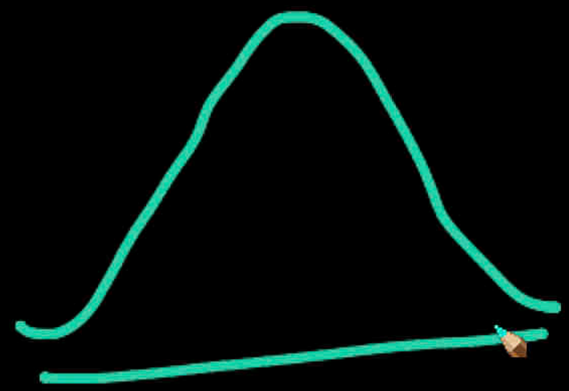
$M.S = 0.4$

$\Rightarrow$

PDF



$N(\mu = 10, S = 0.4)$



```
[ ]  -0.61101705,  0.64726664,  0.71829533,  0.37560761,  0.32028192,
      0.46012344,  0.92880202,  0.67717555,  0.04629136,  0.47546512,
      1.4513086 ,  1.45343272,  1.54991229,  0.62037232,  0.01407354,
      0.46979478,  0.05595689, -1.73249288,  0.23003225,  1.29352827,
      0.80189453,  1.61966331,  0.69681106,  1.03215339, -1.21549361,
      0.93475221,  1.30143537,  0.7254352 ,  0.22841529,  1.50249735,
     -0.02415314, -0.18205881,  0.95388083,  0.66182587,  0.08282857,
      1.53432986,  1.07818559,  1.04804152,  0.62920033,  0.2221568 ,
      1.11689153,  0.70328342,  1.48907562, -0.85967934,  0.37330663,
      0.10042743,  0.43601618, -0.84872277, -0.18902961,  1.16872747,
      0.49445364,  0.97912906,  0.16970087,  1.43121388,  0.67825154,
      0.8233865 ,  1.20263091,  0.49206124,  0.34548617,  1.58287164]
```

```
[ ]  x = np.array(data)
```

```
x.shape
```

```
(500,)
```

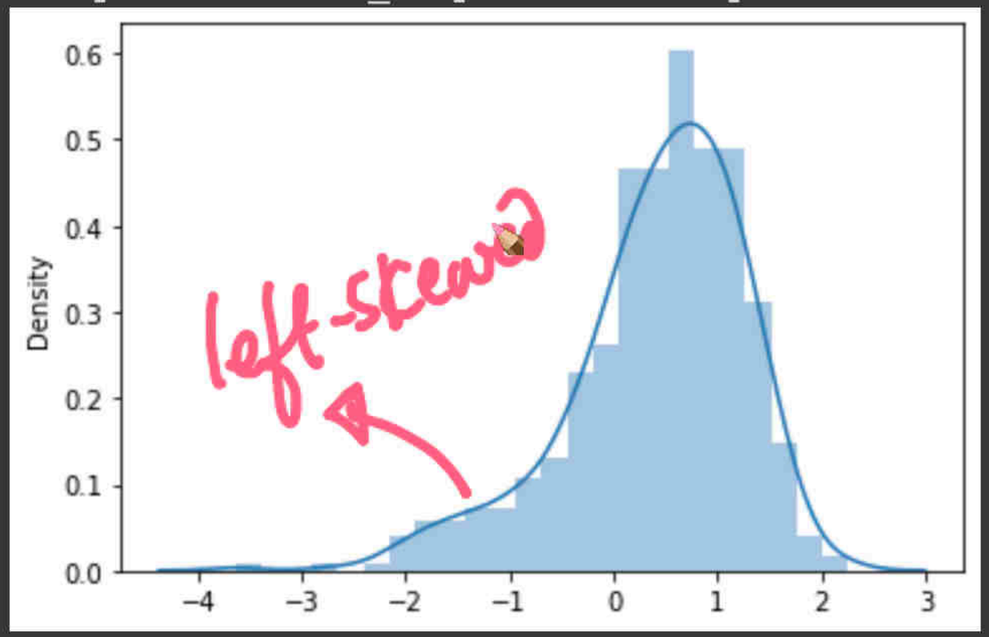
```
sns.distplot(x)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprec
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f146d788c50>
```

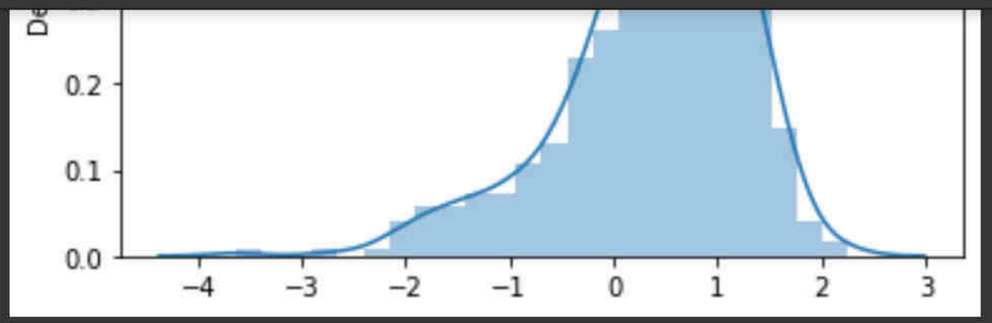
+ Code + Text Reconnect

```
sns.distplot(x)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprec
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f146d788c50>
```

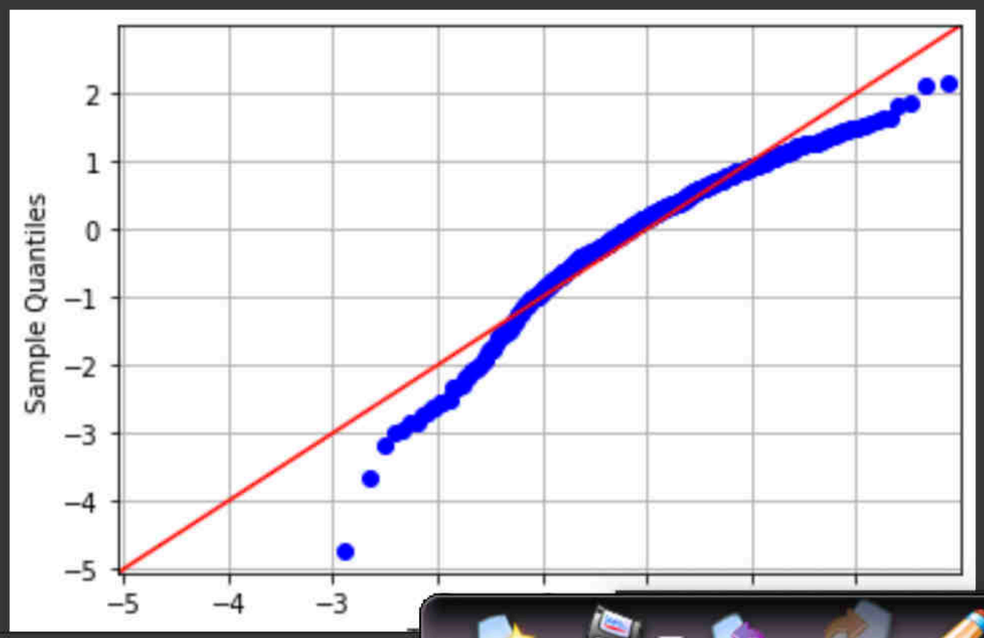


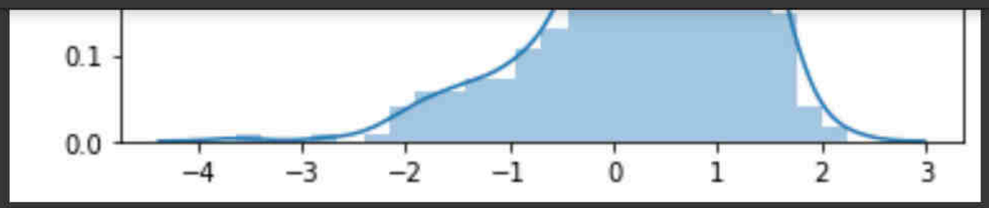

```
import statsmodels.api as sm
fig = sm.qqplot(x, line='45', fit=True)
plt.grid()
```



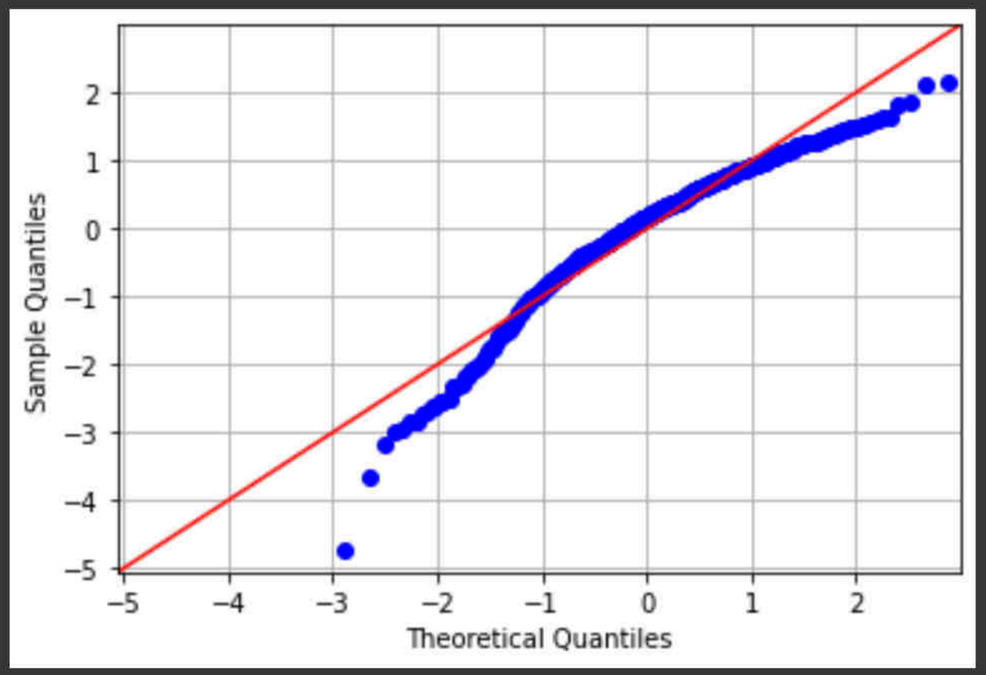
scipy.stats

```
import statsmodels.api as sm
fig = sm.qqplot(x, line='45', fit=True)
plt.grid()
```

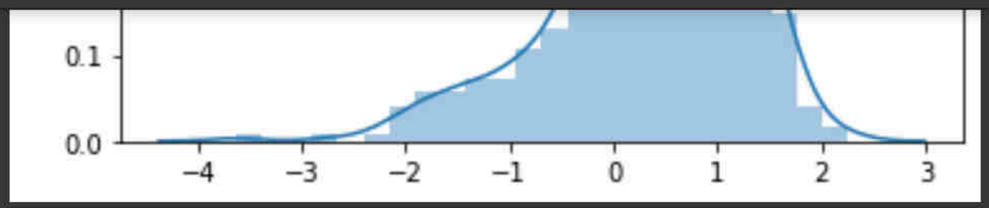





```
[ ] import statsmodels.api as sm
    fig = sm.qqplot(x, line='45', fit=True)
    plt.grid()
```

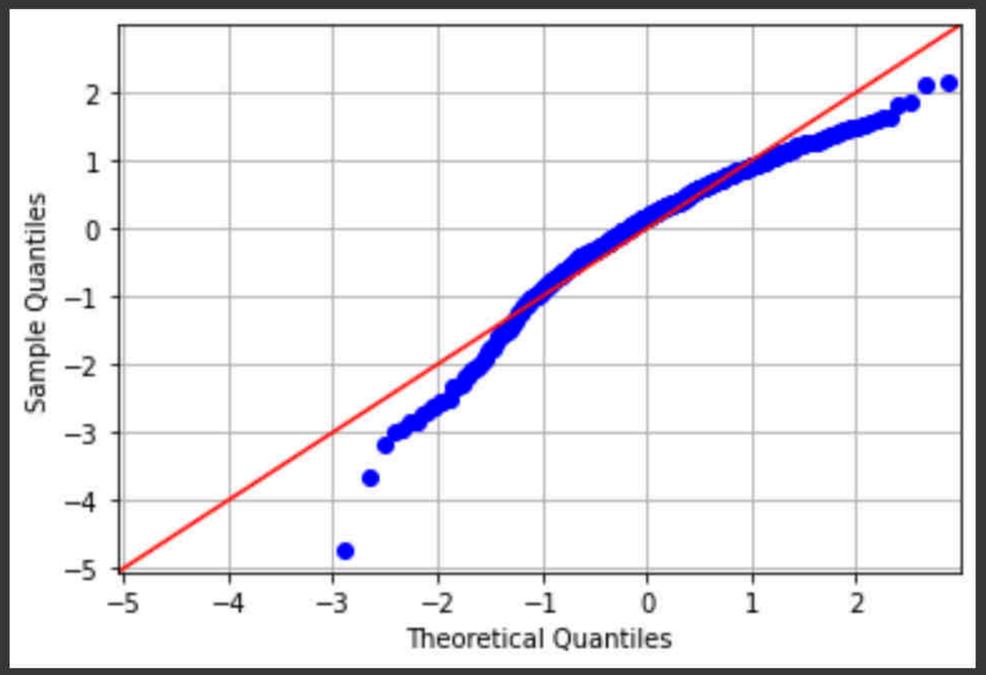


[ ]

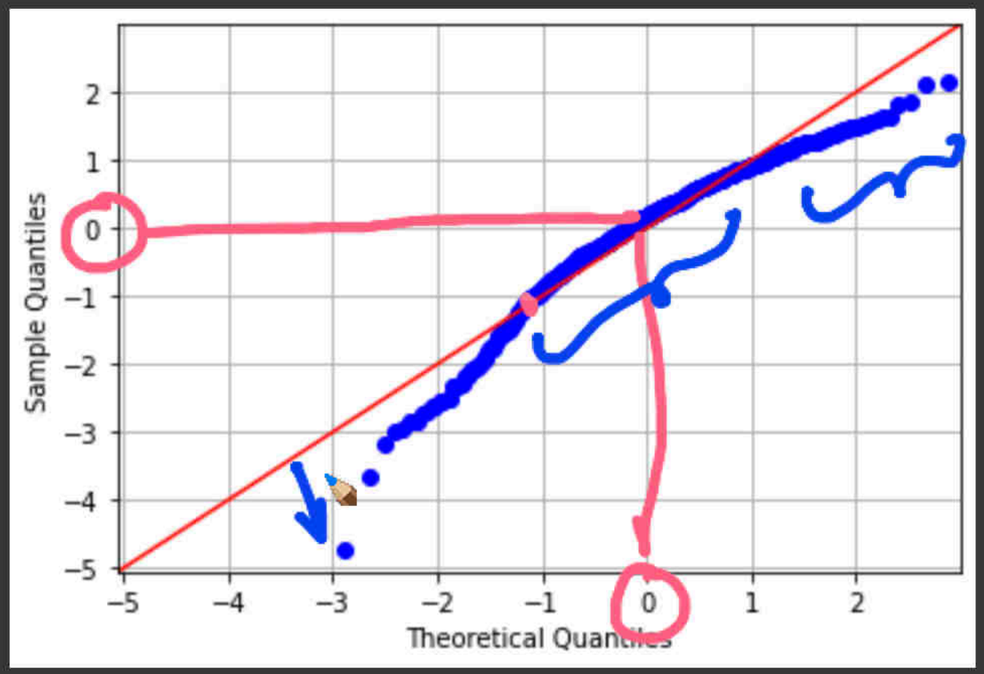


N(m,s)

```
import statsmodels.api as sm
fig = sm.qqplot(x, line='45', fit=True)
plt.grid()
```



```
import statsmodels.api as sm
fig = sm.qqplot(x, line='45', fit=True)
plt.grid()
```



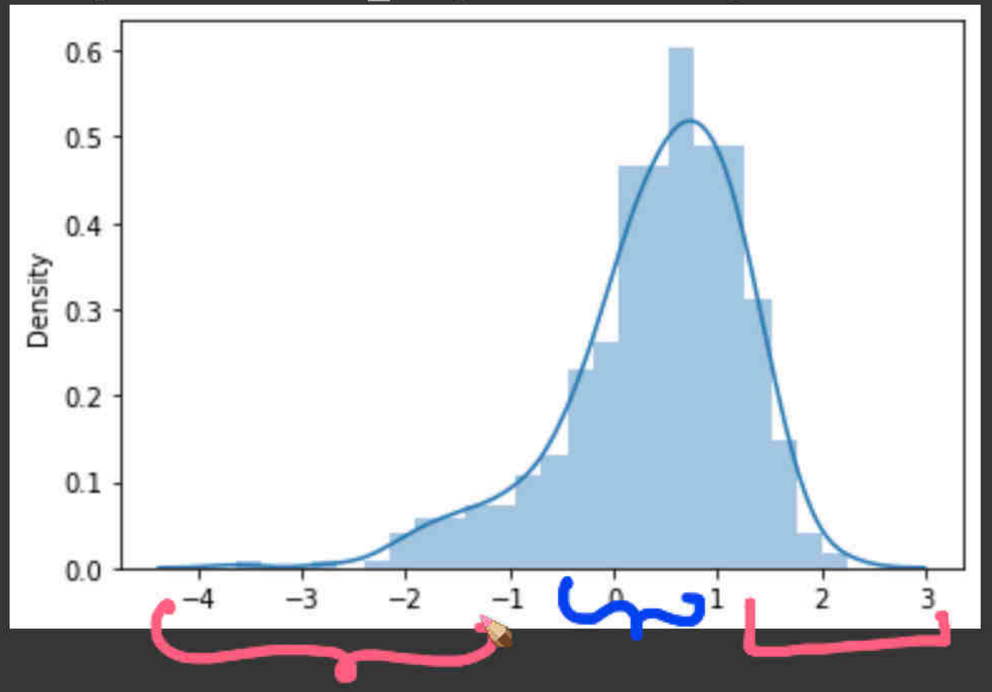

```
[ ] # Let us compare CDFs also
    mu = np.mean(x)
    s = np.std(x)
    print(mu,s)
```

```
[ ] x.shape
```

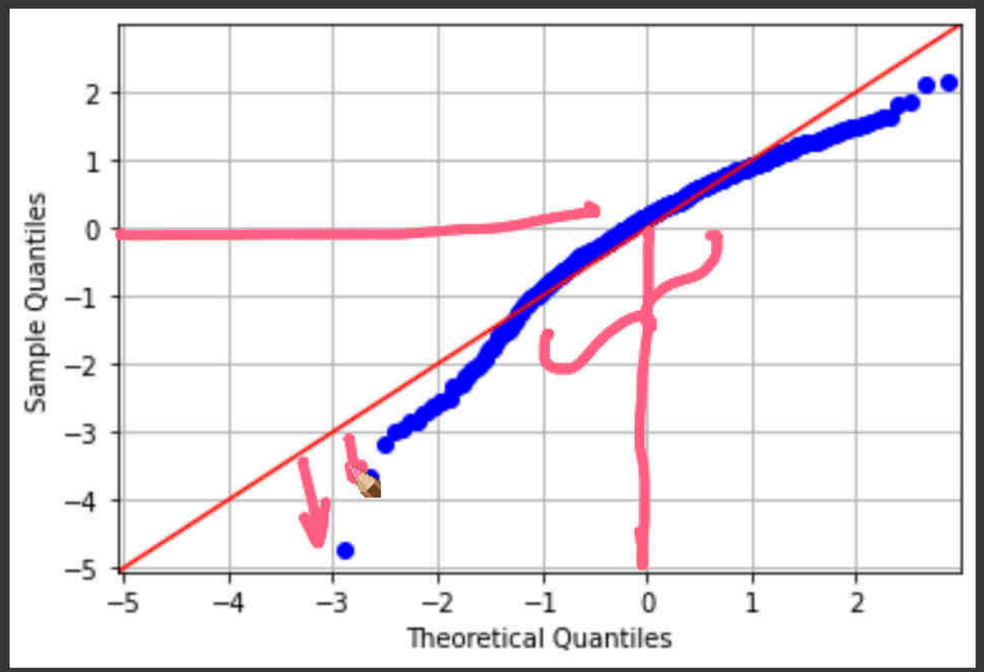
```
(500,)
```

```
[ ] sns.distplot(x)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprec
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f146d788c50>
```



```
import statsmodels.api as sm
fig = sm.qqplot(x, line='45', fit=True)
plt.grid()
```




```
[ ] # Let us compare CDFs also
    mu = np.mean(x)
    s = np.std(x)
    print(mu,s)
```

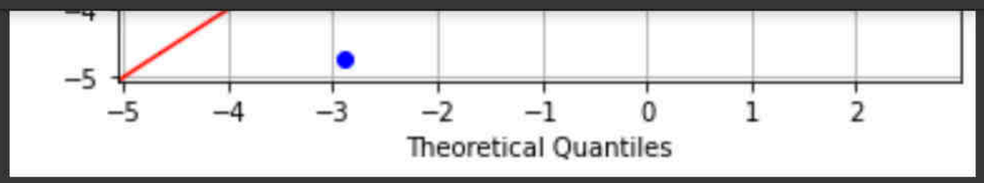
optional Task:

{ Code
scipy....
qqplot

$x_1 \dots x_n$

scipy.stats
lineplot

WAF to plot QQ-plot




```
# Let us compare CDFs also
mu = np.mean(x)
s = np.std(x)
print(mu,s)
```

→ M, S

```
0.40812679178 0.8571215209422999
```

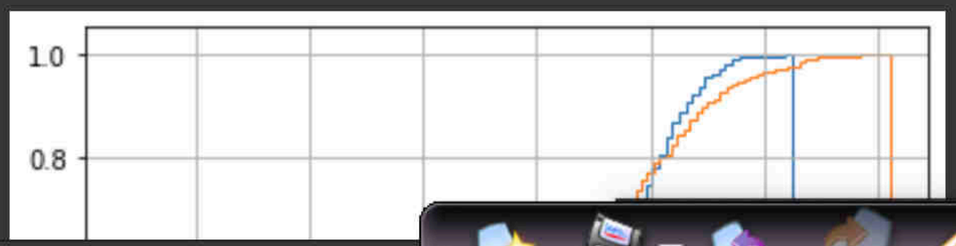
$y_1, y_2 \cdots y_{500}$

$N(M, S)$

```
# normally distributed data with mean=mu, std-dev=s
y = stats.norm.rvs(loc=mu, scale=s, size=500)
```

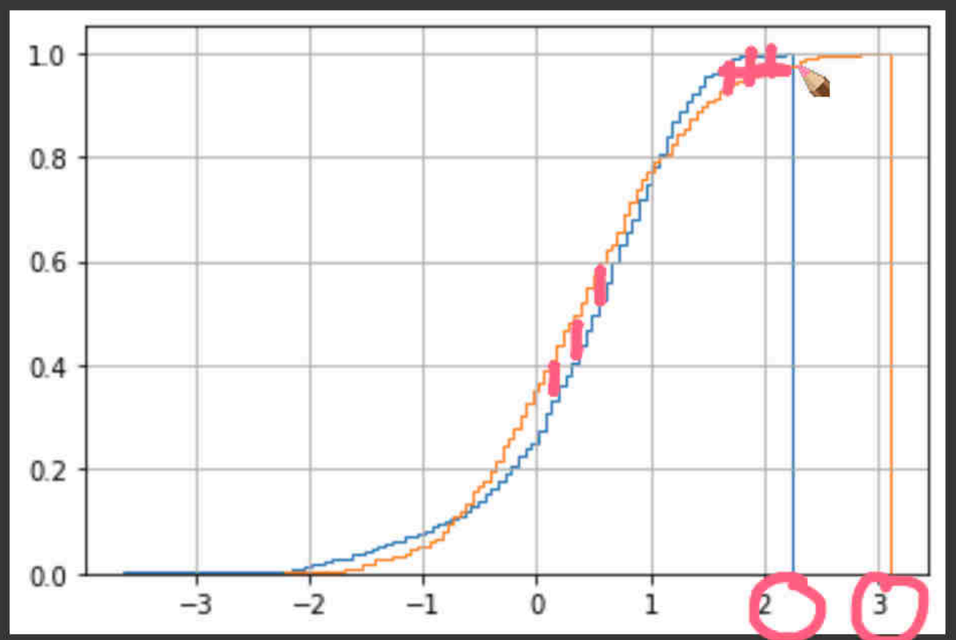
(w

```
plt.grid()
a = plt.hist(x, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
b = plt.hist(y, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
plt.show()
```



```
# normally distributed data with mean=mu, std-dev=s
y = stats.norm.rvs(loc=mu, scale=s, size=500)
```

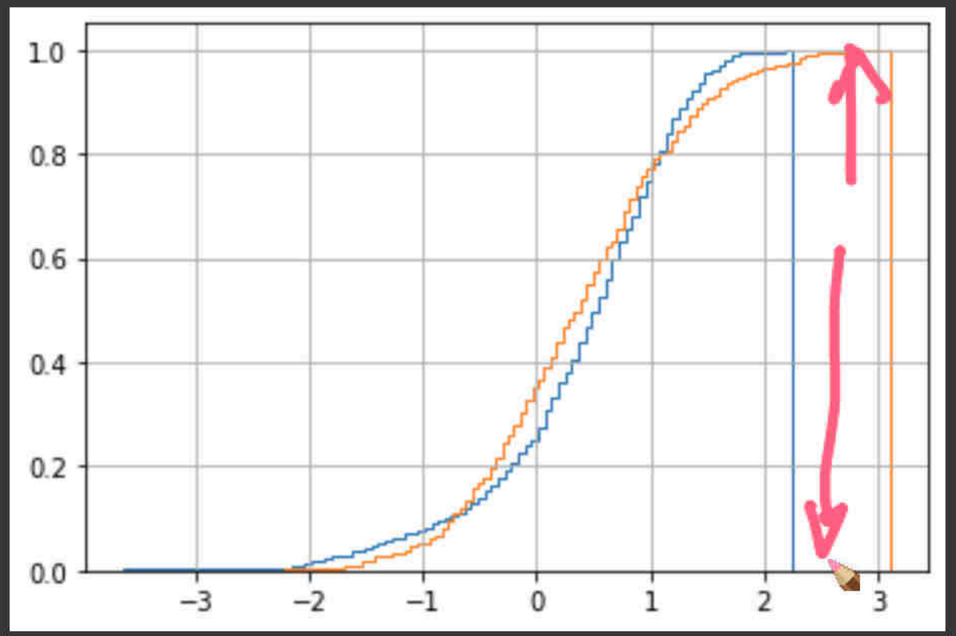
```
plt.grid()
a = plt.hist(x, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
b = plt.hist(y, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
plt.show()
```



QQ Plot: Gaussian vs Gaussian

```
# normally distributed data with mean=mu, std-dev=s
y = stats.norm.rvs(loc=mu, scale=s, size=500)
```

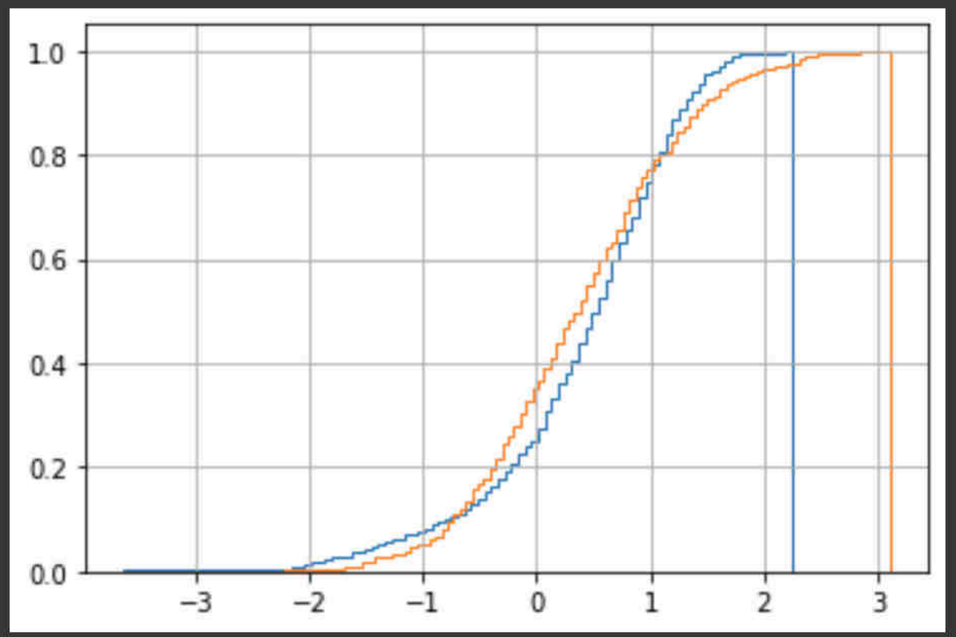
```
plt.grid()
a = plt.hist(x, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
b = plt.hist(y, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
plt.show()
```



QQ Plot: Gaussian vs Gaussian

```
# normally distributed data with mean=mu, std-dev=s
y = stats.norm.rvs(loc=mu, scale=s, size=500)
```

```
plt.grid()
a = plt.hist(x, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
b = plt.hist(y, bins=100, cumulative=True, label='CDF', density=True, histtype='step')
plt.show()
```

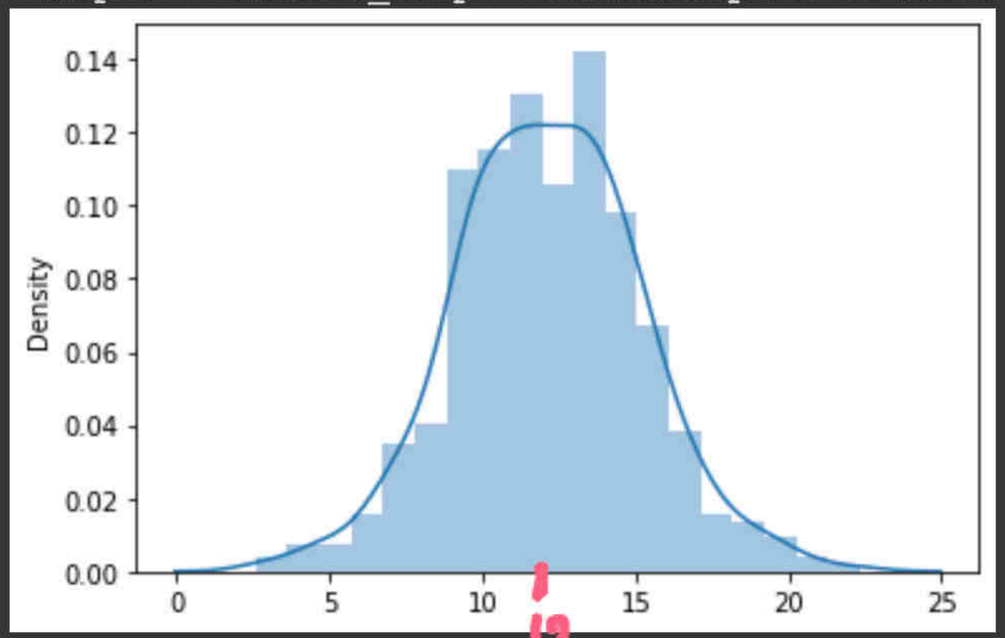


QQ Plot: Gaussian vs Gaussian

```
[ ]        7.13412165, 10.26586752, 14.90528691, 13.65151123, 11.46129682,
          14.28189362,  9.25854953, 10.40331241, 14.82192117,  9.85303781])
```

```
[ ] sns.distplot(x)
```

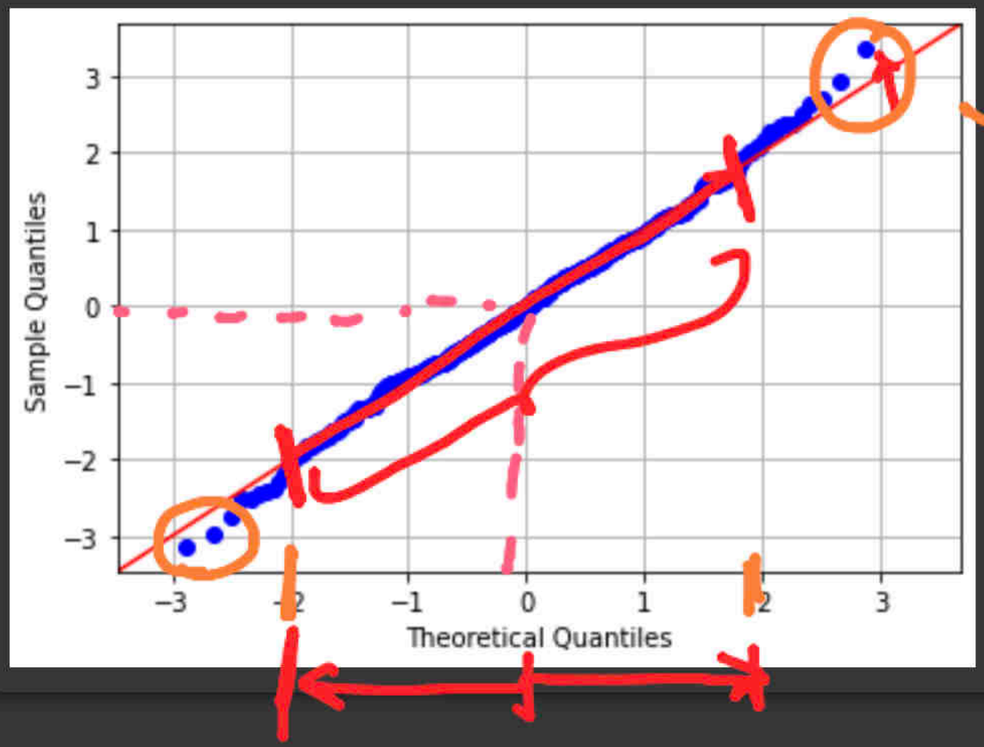
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprec
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f146d5e9690>
```



sized
500 sample

```
[ ] fig = sm.qqplot(x, line='45', fit=True)
    plt.grid()
```

```
fig = sm.qqplot(x, line='45', fit=True)
plt.grid()
```



run 500 obs → scarcely Adala

5M obs

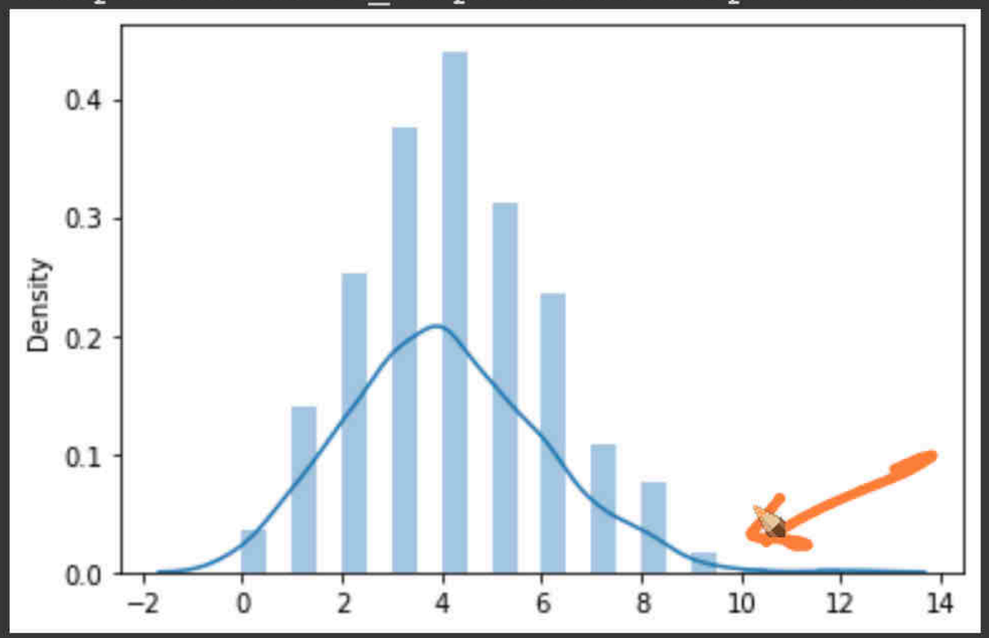
QQ Plot between Non Gaussian and Non-gaussian

[ ] ↳ 4 cells hidden

```
        4,  6,  8,  3,  3,  2,  2])
```

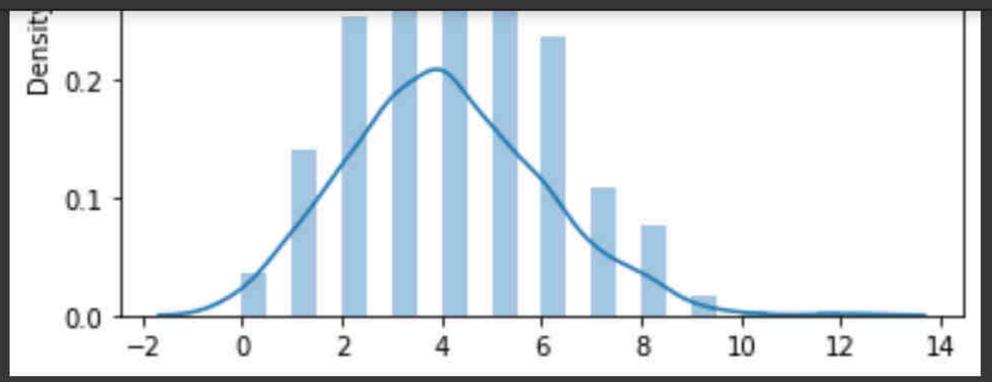
```
[ ]  sns.distplot(x)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprec
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f146d913f90>
```



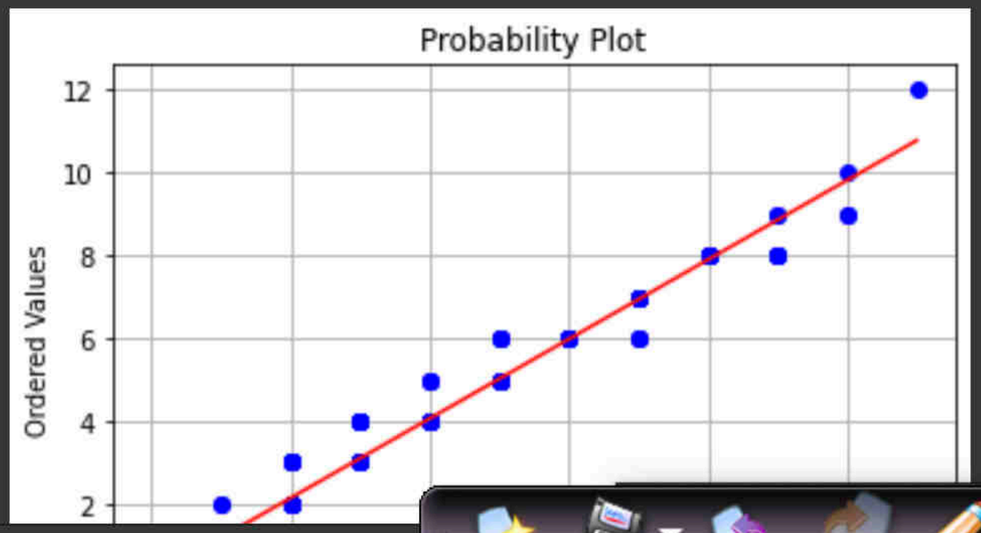

```
[ ]  import scipy.stats as stats
     stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
     plt.grid()
```

[ ]

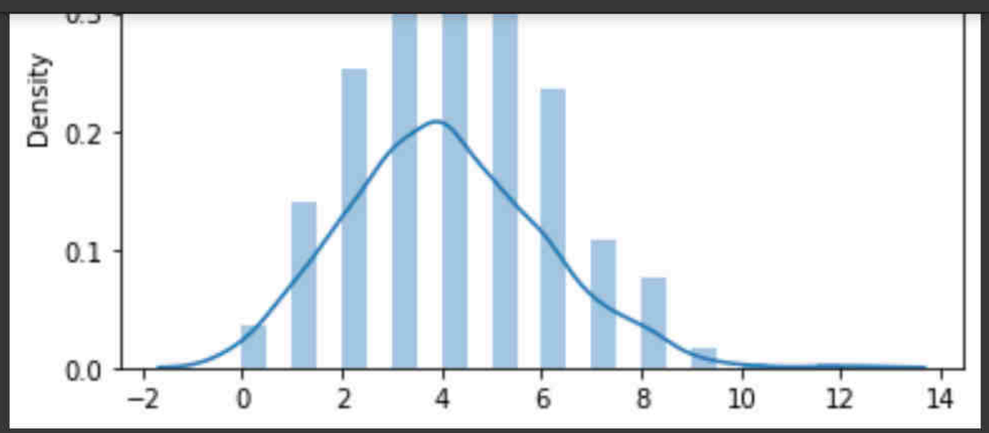


Poisson (λ)

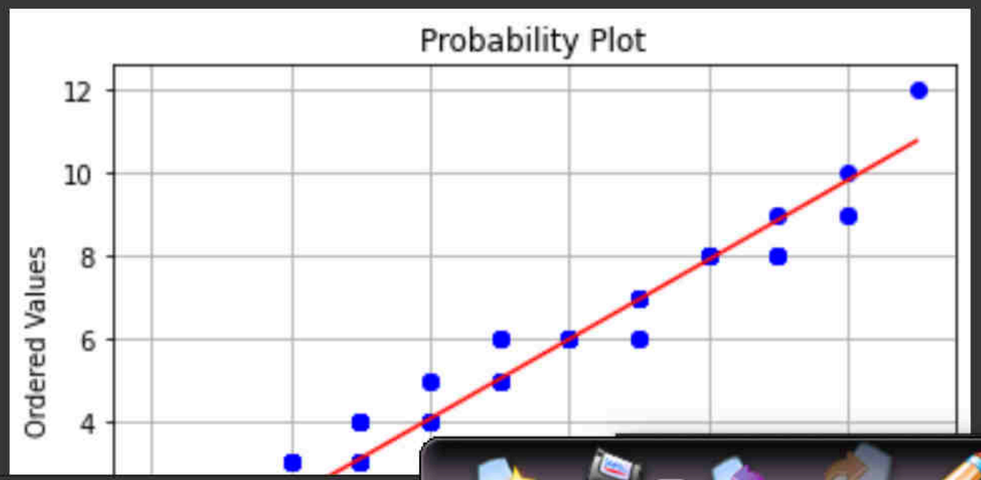
```
import scipy.stats as stats
stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
plt.grid()
plt.show()
```



[ ]

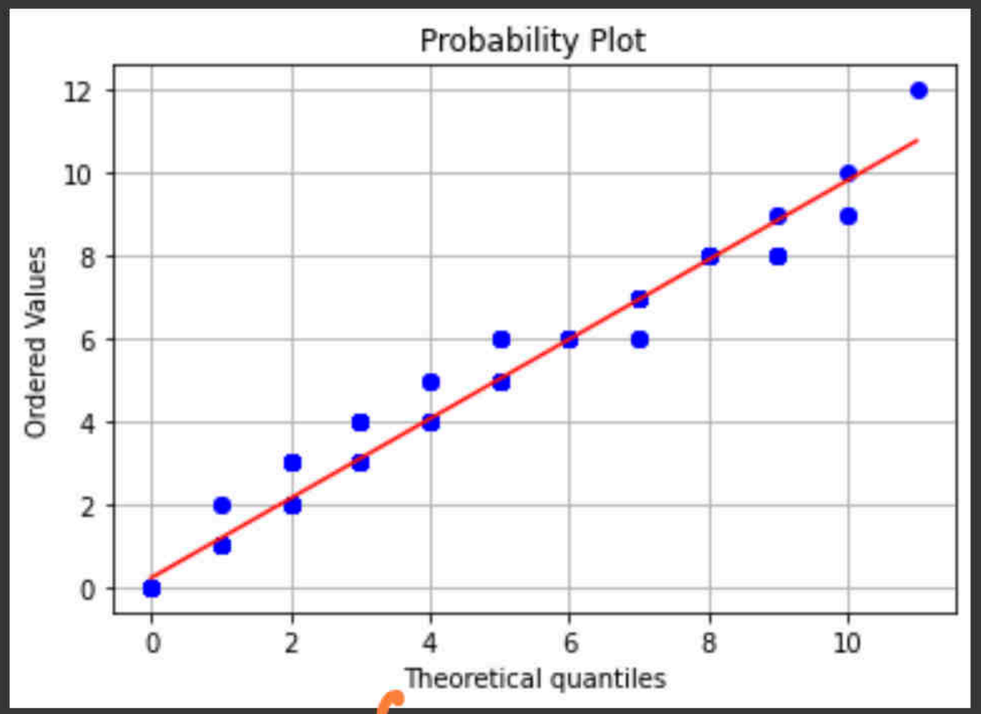



```
import scipy.stats as stats
stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
plt.grid()
plt.show()
```



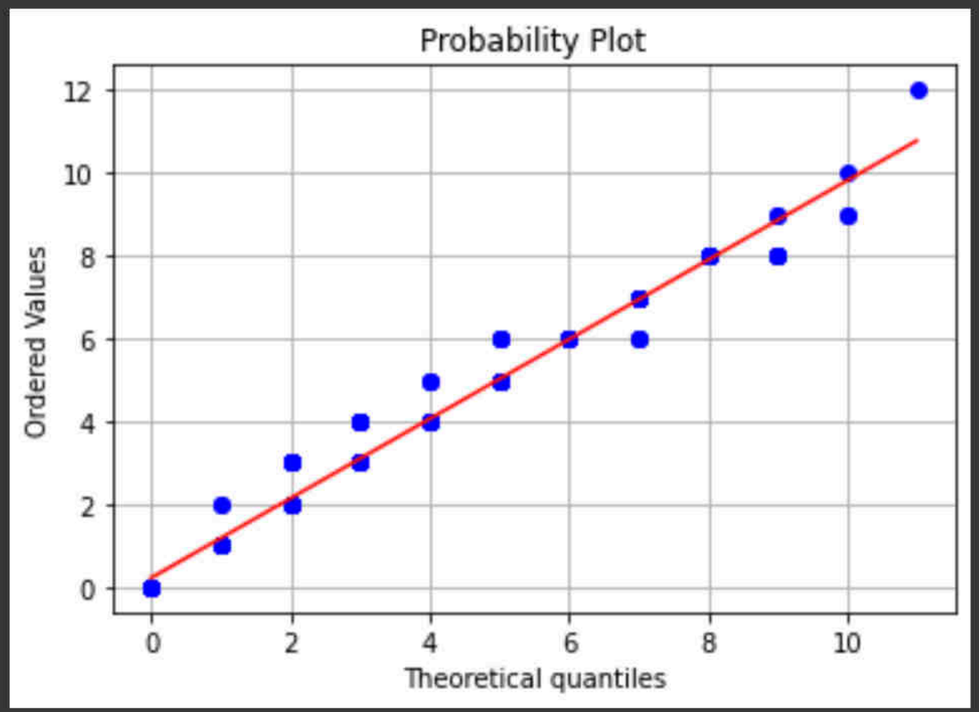
+ Code + Text Reconnect

```
import scipy.stats as stats
stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
plt.grid()
plt.show()
```



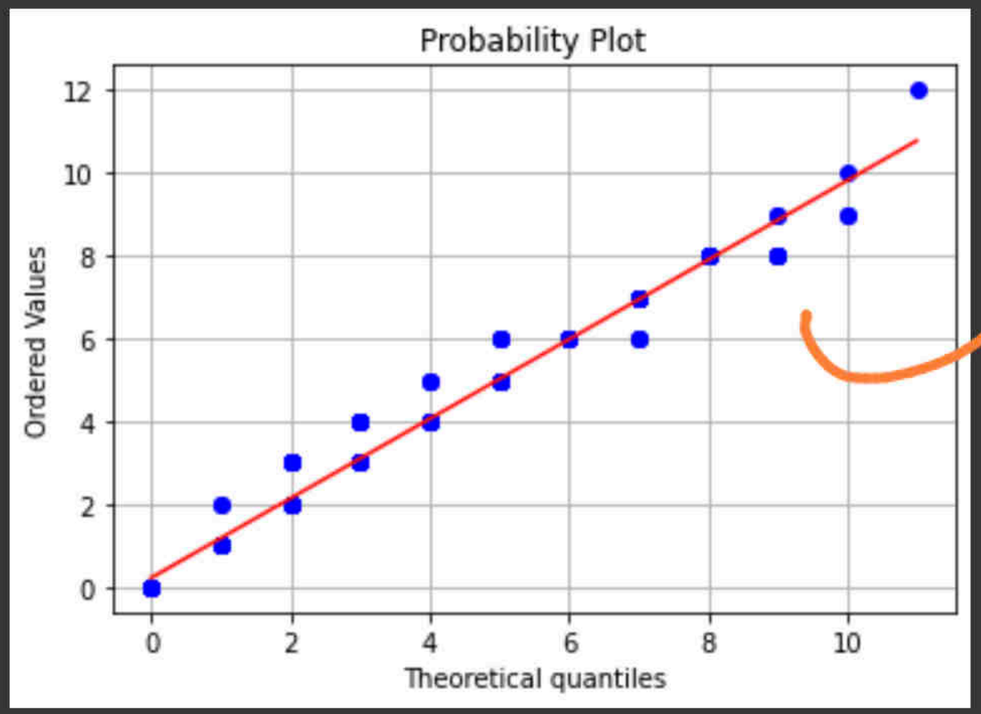

```
[ ]
```

```
import scipy.stats as stats
stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
plt.grid()
plt.show()
```




```
[ ]
```

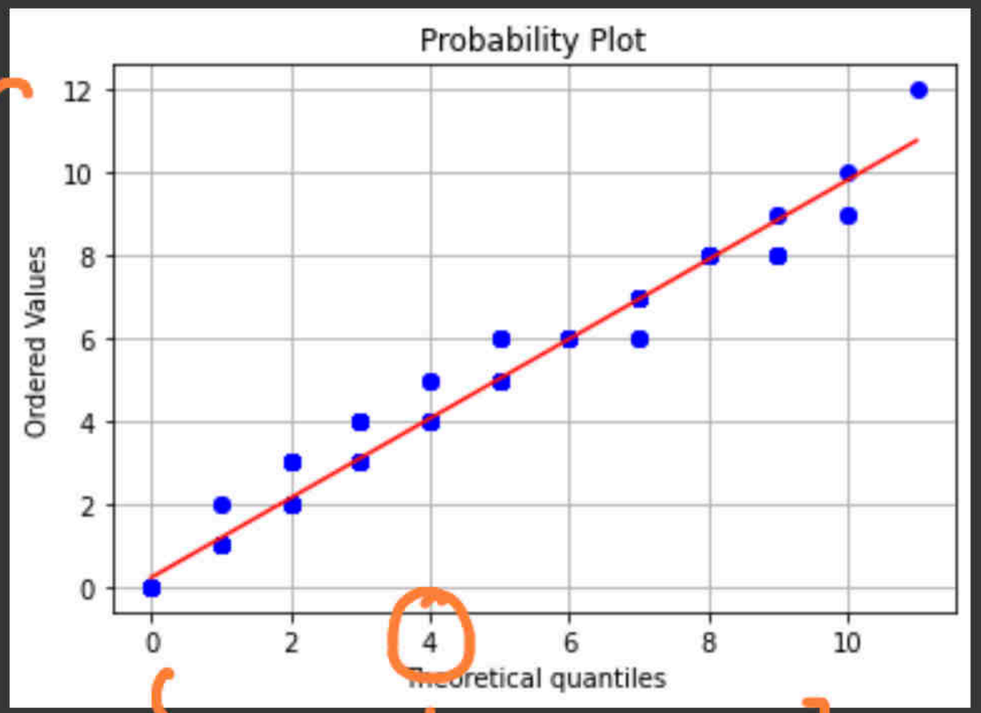
```
import scipy.stats as stats
stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
plt.grid()
plt.show()
```



$X \sim Poisson(\lambda = 4)$

```
[ ]
```

```
    plt.show()
[ ]
```

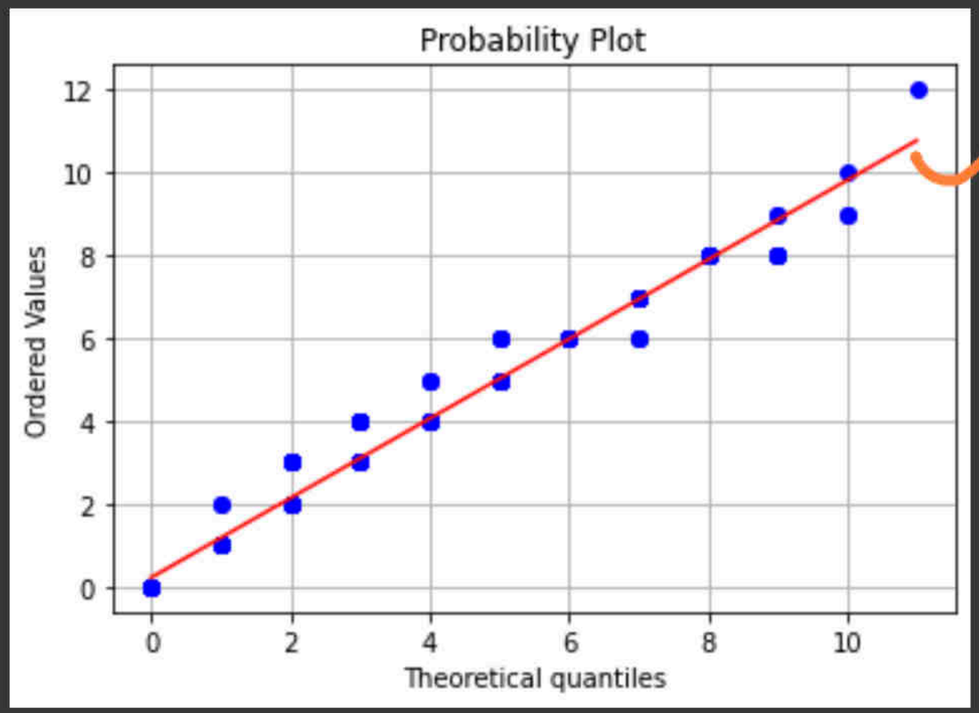



```
[ ]
```

Box cox transformation

```
[ ]  ↳ 9 cells hidden
```

```
    plt.show()
[ ]
```



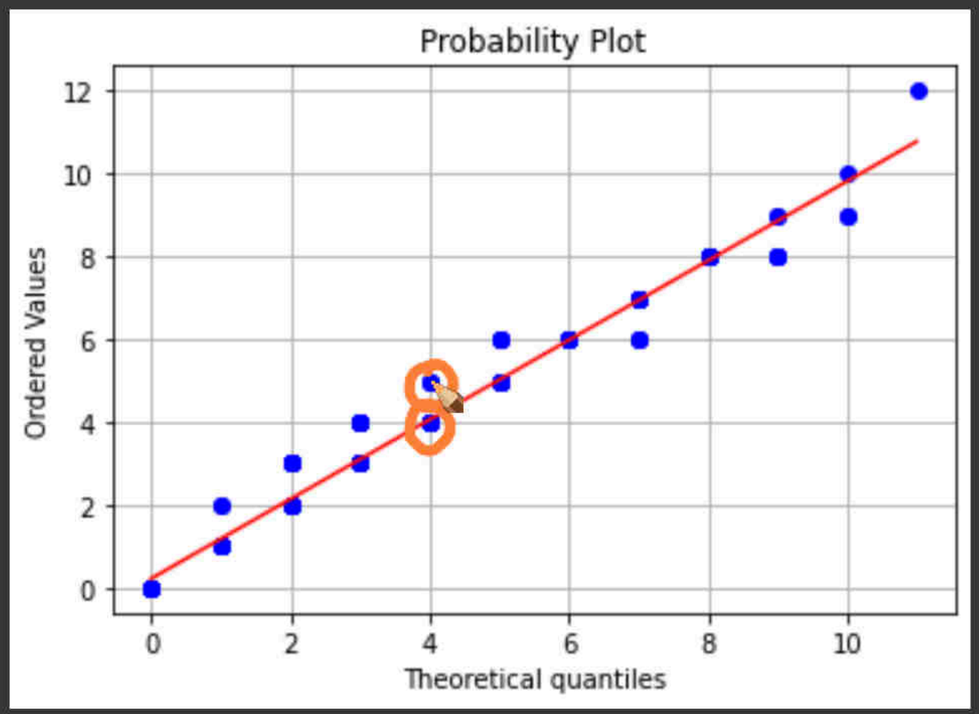
Plus dog

[ ]

Box cox transformation

[ ] ↳ *9 cells hidden*

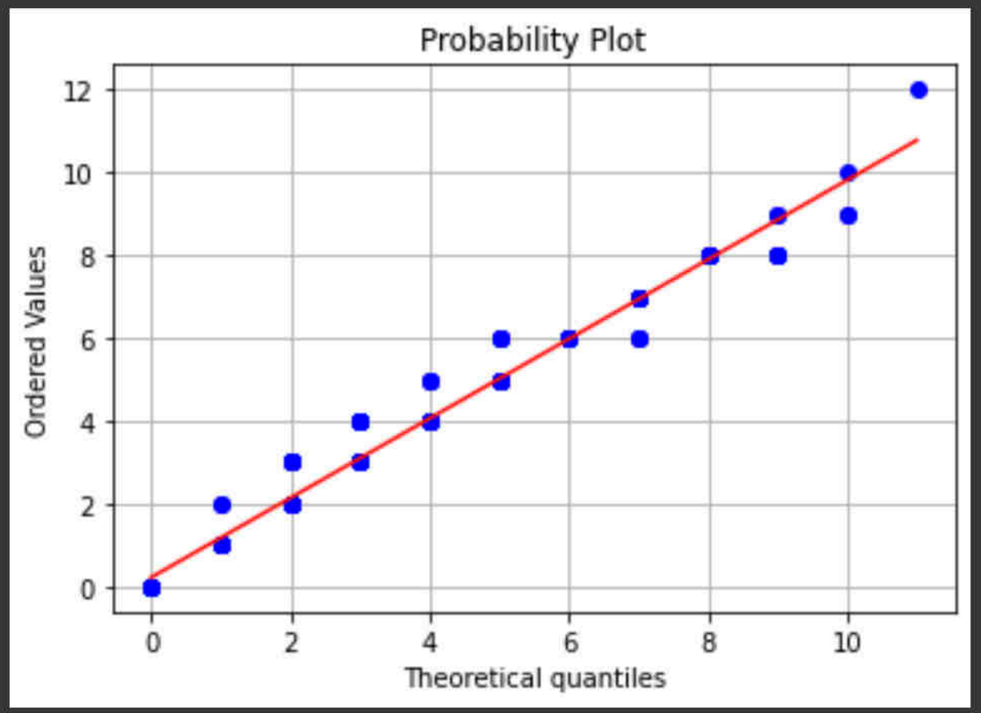
```
[ ] import scipy.stats as stats
    stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
    plt.grid()
    plt.show()
```




```
[ ]
```

Box cox transformation

```
import scipy.stats as stats
stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
plt.grid()
plt.show()
```

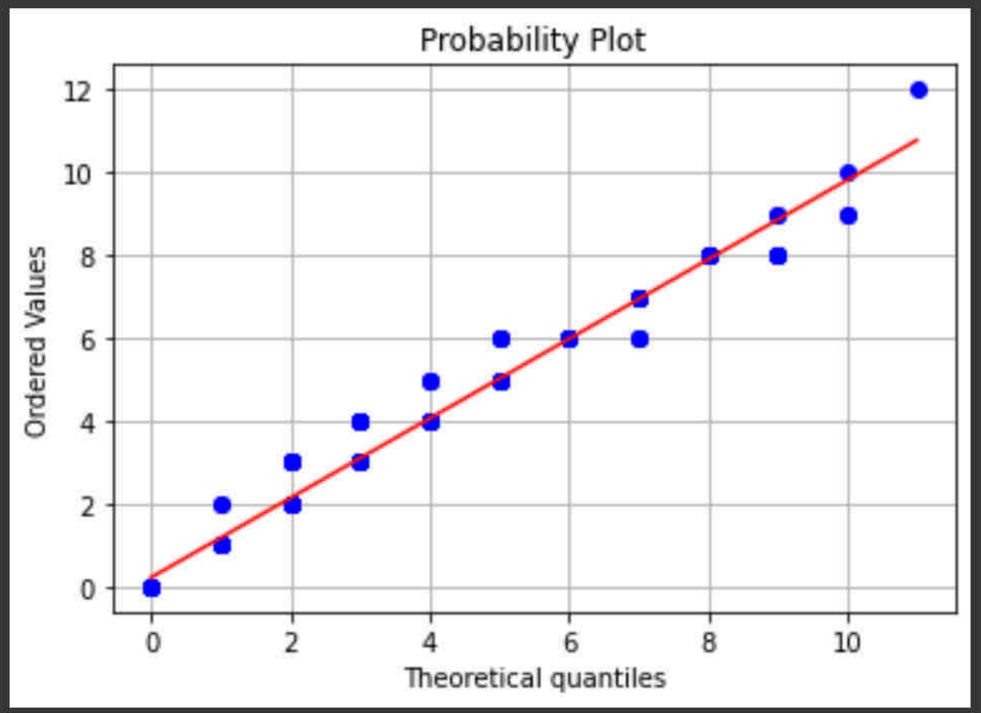


Pop: Poisson(λ=4)

Sample: $x_1 \cdots x_{50}$

[ ]

```
import scipy.stats as stats
stats.probplot(x, dist='poisson', sparams=(4,), plot=plt)
plt.grid()
plt.show()
```



Poisson (λ ← mean)

$x_1 \cdots x_{50}$

Sample Mean

[ ]

Task:

$x_1 \cdots x_{500}$

Normal ( )

QQ-plot

scipy.stats
np
:
line plot

babyweights

$x_1, \ldots x_n$

QQplot

Normal(M,S)

disb (params)

(model)

$[P(x \leq 1.8\text{kgs})$

Transformations

Task:

X

$x_1\ x_2 \cdots x_i \cdots x_n$

$\xrightarrow{f()}$

X'

$x_1', x_2', \cdots x_i', \cdots x_n'$

Not-Gaussian

Gaussian

easy to explain

→ tons of Math

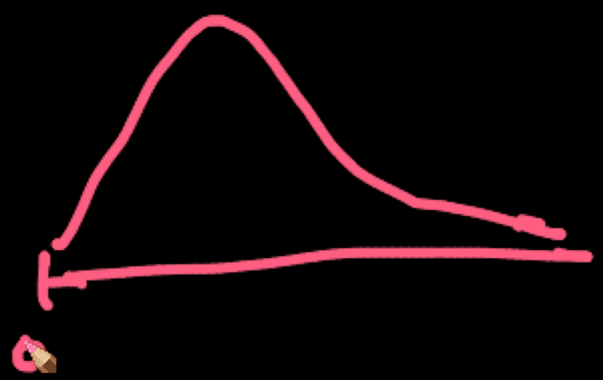
[ML: Th. proof ←

(Q) guess a transformation:

$x_1, x_2 \dots x_i \dots x_n \xrightarrow{\ln(\,)} x_1' x_2' \dots x_i' \dots$

Normal

log-normal



$\log_e = \ln$

$f(\ ) \longrightarrow \log()$ ✓

↳ sqrt() → not popular

⋮

$f(x_i) = \overbrace{x_i'}$ for all $i : 1 \to n$

↳ QQ-plot

Power transform → Box-cox

$$x_i^{(\lambda)} = \begin{cases} \dfrac{x_i - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(x_i) & \text{if } \lambda = 0 \end{cases}$$

$x_1, x_2 \ldots\ldots x_n \longrightarrow$ BoxCox $\longrightarrow \lambda$

λ ↑ optmzn

Box-cox:

$\lambda = 0 \Rightarrow x_1 \dots x_n \sim$ log-normal

$\lambda \neq 0 \Rightarrow$

Box–Cox transformation [edit]

The one-parameter Box–Cox transformations are defined as

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln y_i & \text{if } \lambda = 0, \end{cases}$$

and the two-parameter Box–Cox transformations as

$$y_i^{(\lambda)} = \begin{cases} \dfrac{(y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{if } \lambda_1 \neq 0, \\ \ln(y_i + \lambda_2) & \text{if } \lambda_1 = 0, \end{cases}$$

as described in the original article.[8][9] Moreover, the first transformations hold for $y_i > 0$, and the second for $y_i > -\lambda_2$.[8]

The parameter λ is estimated using the profile likelihood function and using goodness-of-fit tests.[10]

Confidence interval [edit]

Confidence interval for the Box–Cox transformation can be asymptotically constructed using Wilks's theorem on the profile likelihood function to find all the possible values of λ that fulfill the following restriction:[11]

$\ln(L(\lambda)) = \max_\lambda(\ln(L(\lambda))) - \tfrac{1}{2}\chi^2_{1,1-\alpha}$

$x_1 - x_i \; x_n$ → Boxcox → best λ → $x_1' \; x_2' \; \ldots \; x_i' \; - x_n'$

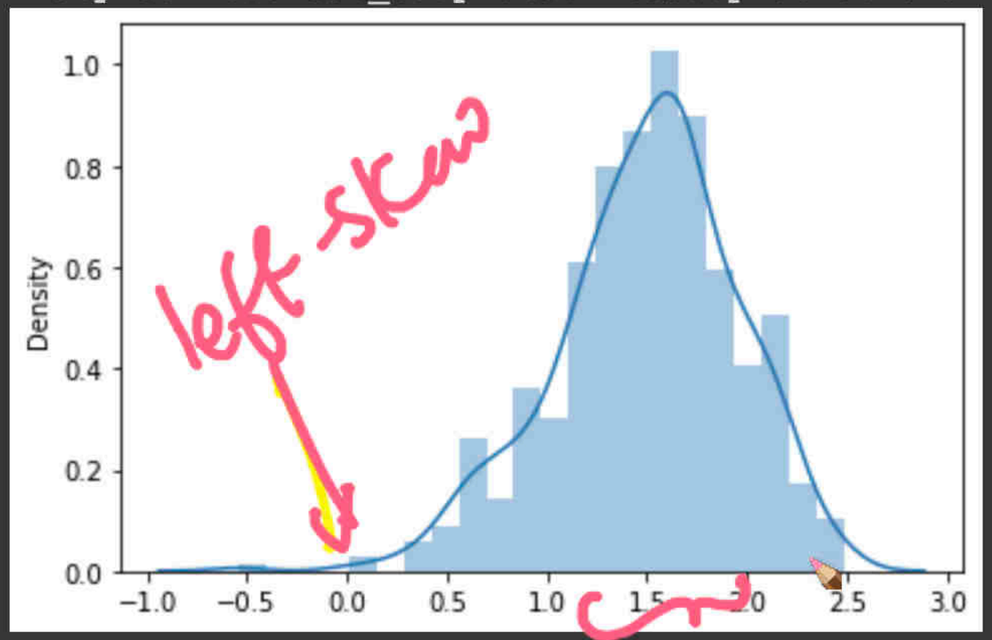
non-normal

↓

QQ-plot (verify)

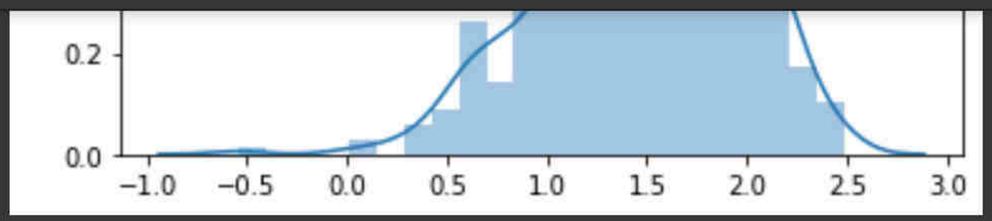
[not always guaranteed to work

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprec
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f146d53c650>
```

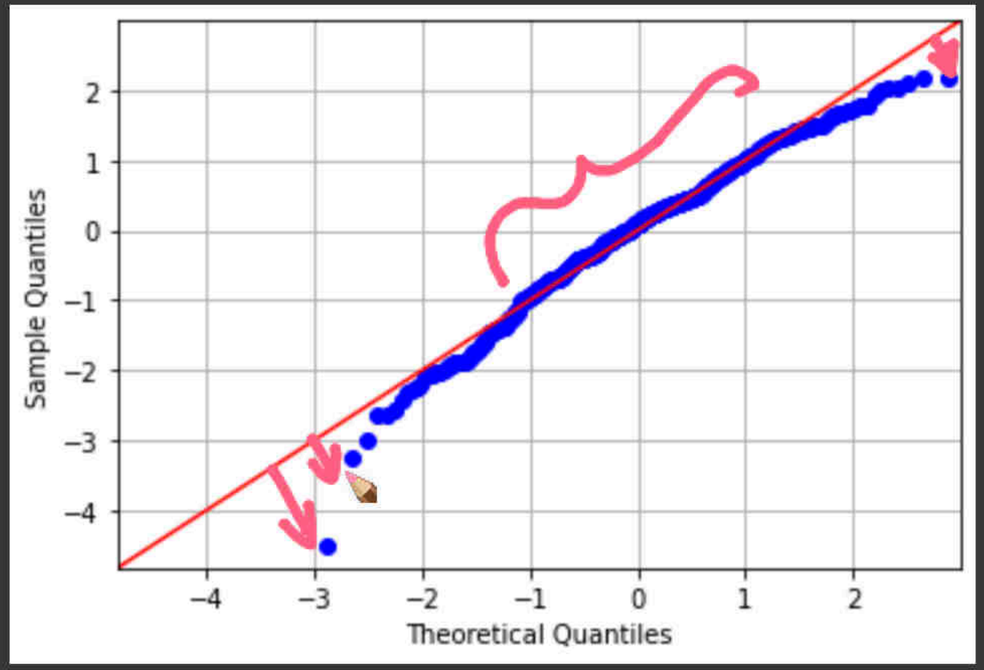



```
fig = sm.qqplot(x, line='45', fit=True)
plt.grid()
```

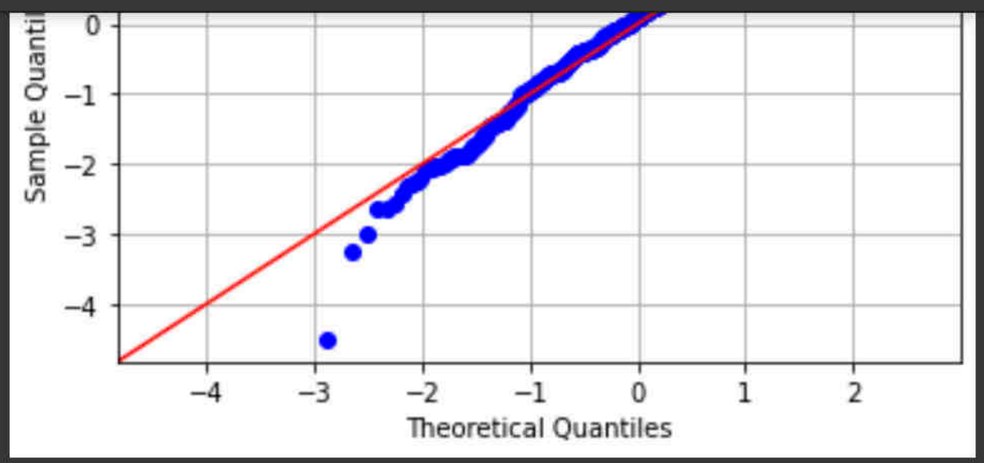
[ ]




```
[ ]  fig = sm.qqplot(x, line='45', fit=True)
     plt.grid()
```



[ ] xt, l = stats.box

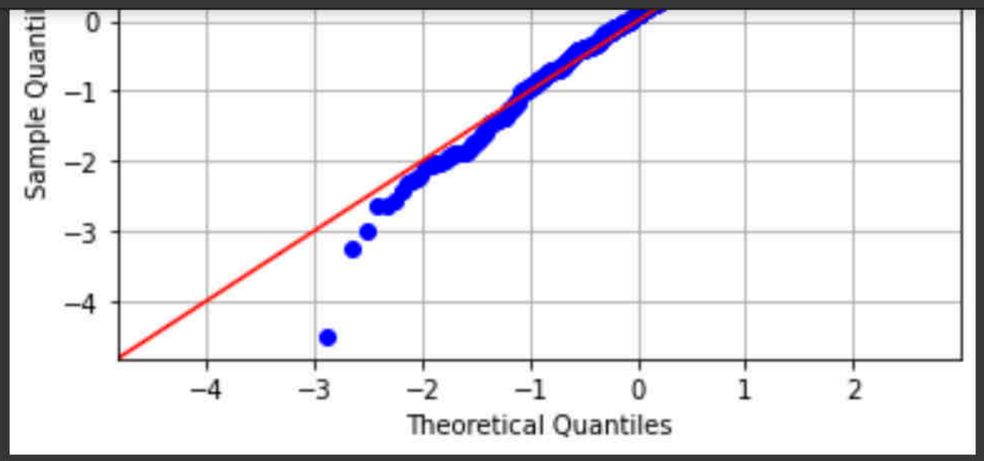



```
xt, l = stats.boxcox(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-80d99d196c18> in <module>()
----> 1 xt, l = stats.boxcox(x)

/usr/local/lib/python3.7/dist-packages/scipy/stats/morestats.py in boxcox(x, lmbda, alpha)
   1043
   1044     if any(x <= 0):
-> 1045         raise ValueError("Data must be positive.")
   1046
   1047     if lmbda is not None:  # single transformation

ValueError: Data must be positive.
```

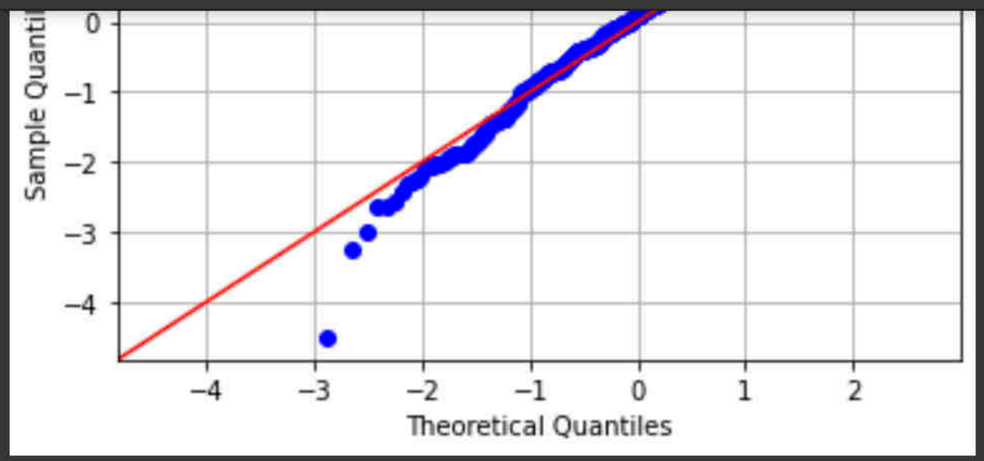



```
[ ] xt, l = stats.boxcox(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-80d99d196c18> in <module>()
----> 1 xt, l = stats.boxcox(x)

/usr/local/lib/python3.7/dist-packages/scipy/stats/morestats.py in boxcox(x, lmbda, alpha)
   1043
   1044     if any(x <= 0):
-> 1045         raise ValueError("Data must be positive.")
   1046
   1047     if lmbda is not None:  # single transformation

ValueError: Data must be positive.
```




```
[ ] xt, l = stats.boxcox(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-80d99d196c18> in <module>()
----> 1 xt, l = stats.boxcox(x)

/usr/local/lib/python3.7/dist-packages/scipy/stats/morestats.py in boxcox(x, lmbda, alpha)
   1043
   1044     if any(x <= 0):
-> 1045         raise ValueError("Data must be positive.")
   1046
   1047     if lmbda is not None:  # single transformation

ValueError: Data must be positive.
```

$X \rightarrow x_1, x_2 \ldots x_n$

x_2 → 0.54

$X+1 \rightarrow x_1+1, x_2+1 \ldots x_n+1 \quad > 0$

+ Code + Text Reconnect

SEARCH STACK OVERFLOW

```
[ ] print(np.min(x))
```

```
-0.54601206
```

$\lambda = \lambda = 1.816$

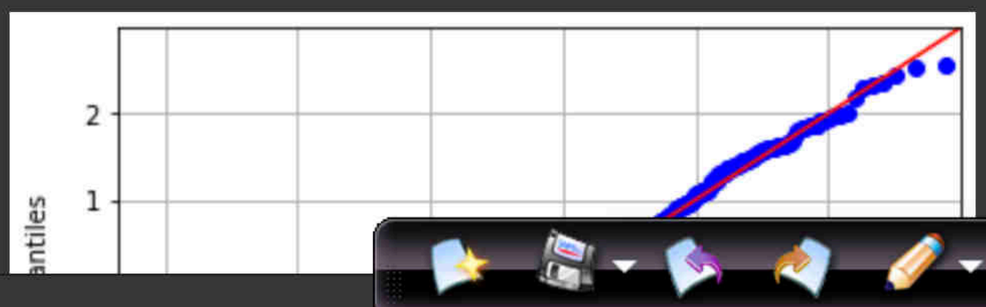
```
[ ] x1 = x + 1
```

```
[ ] xt, l = stats.boxcox(x1)
```

```
print(l)
```

```
1.8162672858204911
```

```
fig = sm.qqplot(xt, line='45', fit=True)
plt.grid()
```



$$x \xrightarrow{+1} (x+1) \xrightarrow{\text{boxcox}}$$

$x_i \qquad (x_i+1) \qquad \lambda = 1.86$

$$x_i \xrightarrow{f(\cdots)} x_i'$$

$$\left\{ \frac{(x_i+1)^{\lambda} - 1}{\lambda} \right. = x_i'$$

$\tilde{x}_i$ → $\tilde{x}_i'$

$\tilde{x}_i$: days

$\tilde{x}_i'$: Gaussian

68-95-99

$(\underline{10.5}, \underline{12.1})$

$m-s$ $\underline{3.1}$ 68% $m+s$ $\underline{4.6}$

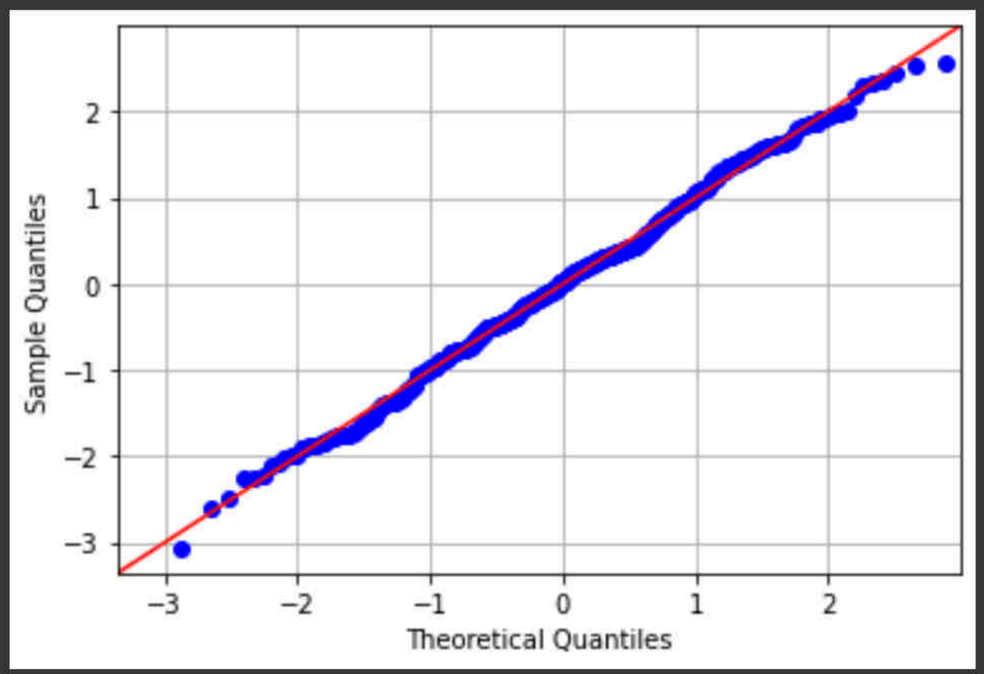
$x_i \xrightarrow{f} x_i'$, f^{-1}

$x_i \xrightarrow{\log} x_i$, $e^{\cdot}$

```
[ ] print(l)
```

```
1.8162672858204911
```

```
fig = sm.qqplot(xt, line='45', fit=True)
plt.grid()
```



Box–Cox transformation [edit]

The one-parameter Box–Cox transformations are defined as

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln y_i & \text{if } \lambda = 0, \end{cases}$$

and the two-parameter Box–Cox transformations as

$$y_i^{(\boldsymbol{\lambda})} = \begin{cases} \dfrac{(y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{if } \lambda_1 \neq 0, \\ \ln(y_i + \lambda_2) & \text{if } \lambda_1 = 0, \end{cases}$$

as described in the original article.[8][9] Moreover, the first transformations hold for $y_i > 0$, and the second for $y_i > -\lambda_2$.[8]

The parameter λ is estimated using the profile likelihood function and using goodness-of-fit tests.[10]

Confidence interval [edit]

Confidence interval for the Box–Cox transformation can be asymptotically constructed using Wilks's theorem on the profile likelihood function to find all the possible values of λ that fulfill the following restri[11]

70

$x_1 = -0.5$ → square → 0.25 → Loxcox

$x_{10} = 0.5$ → 0.25

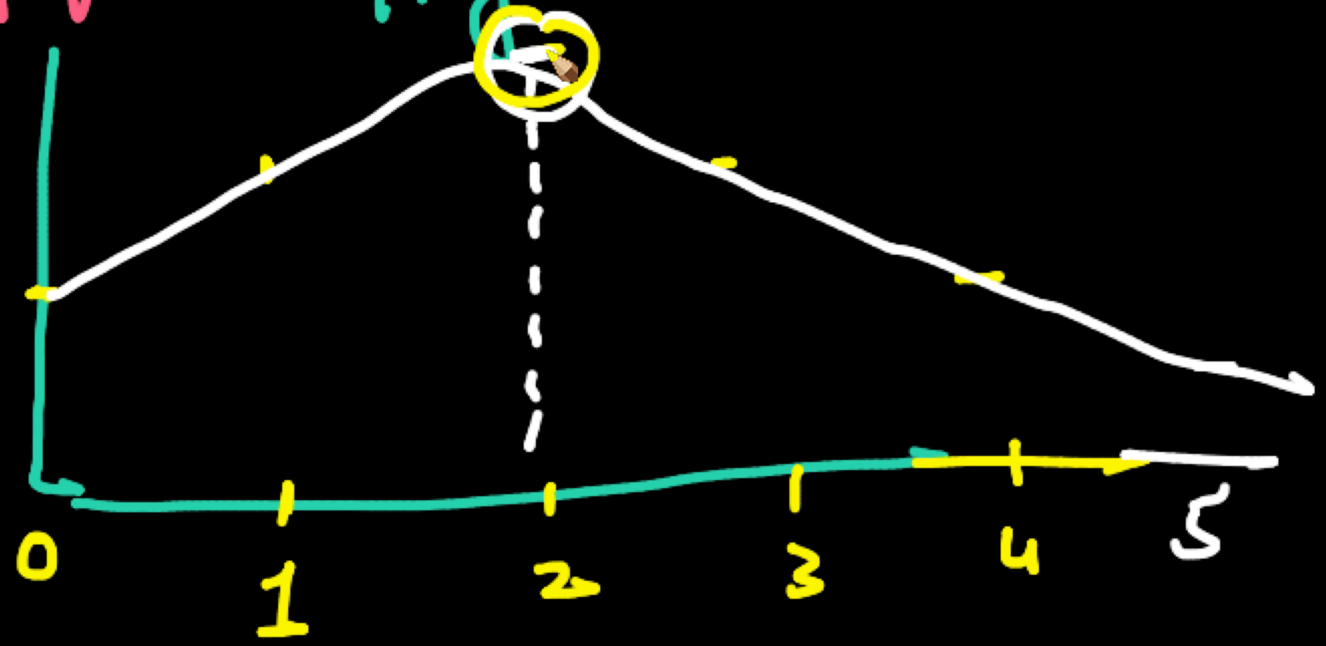
Mode

→ most freq. occurring value

discrete r.v

PMF

#goals in a WC- Soccer



heights

Cont r.V

PDF

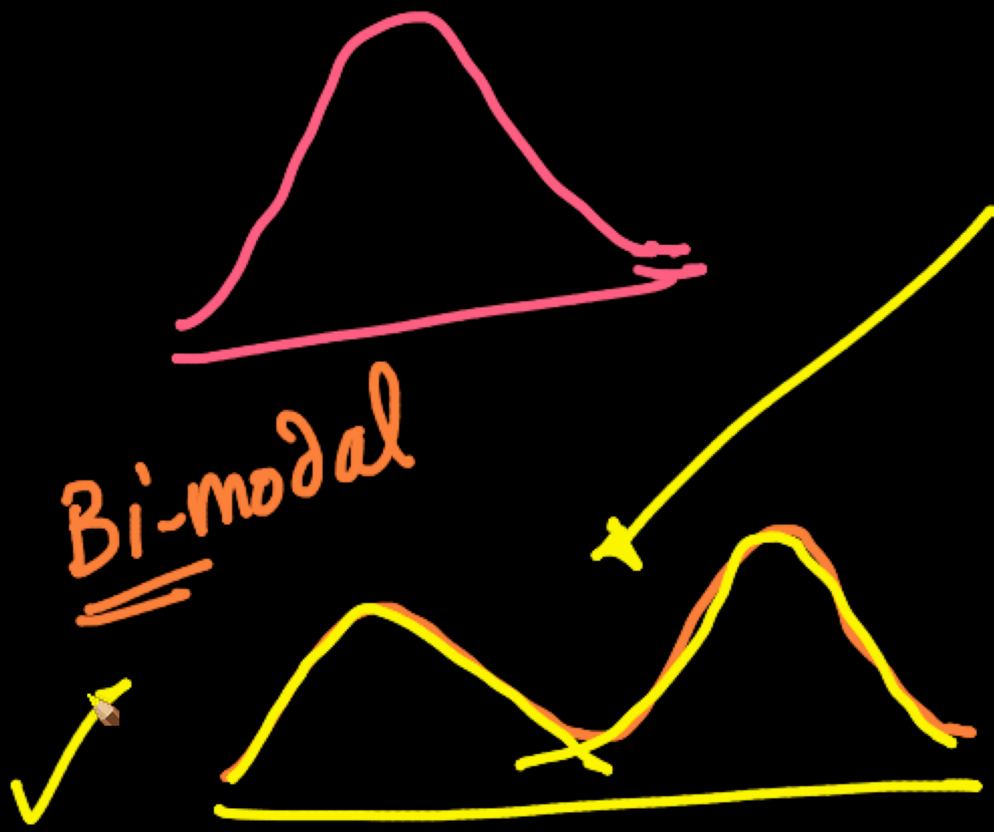
maxima (later)

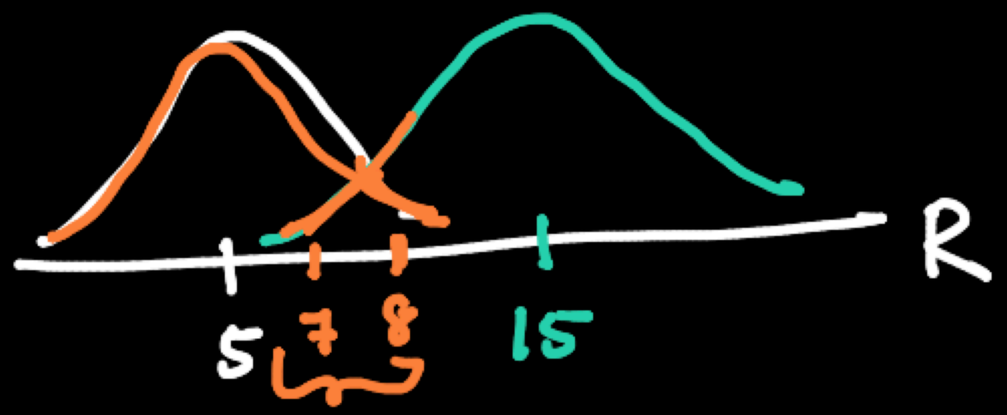
Mode

Covid recovery times → Kids (<5yrs) 100

→ elderly 60+ 100

200 points



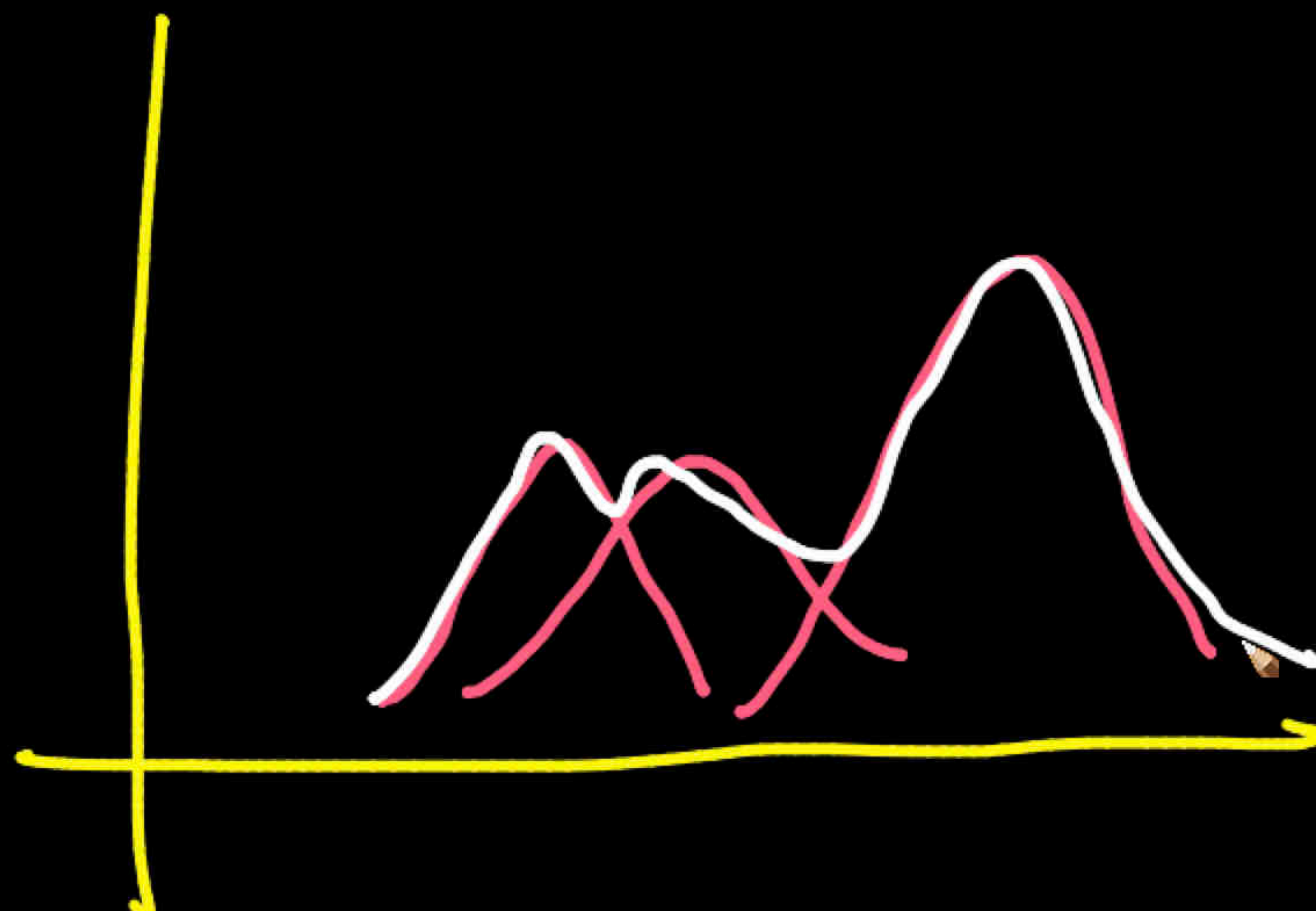


Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:19

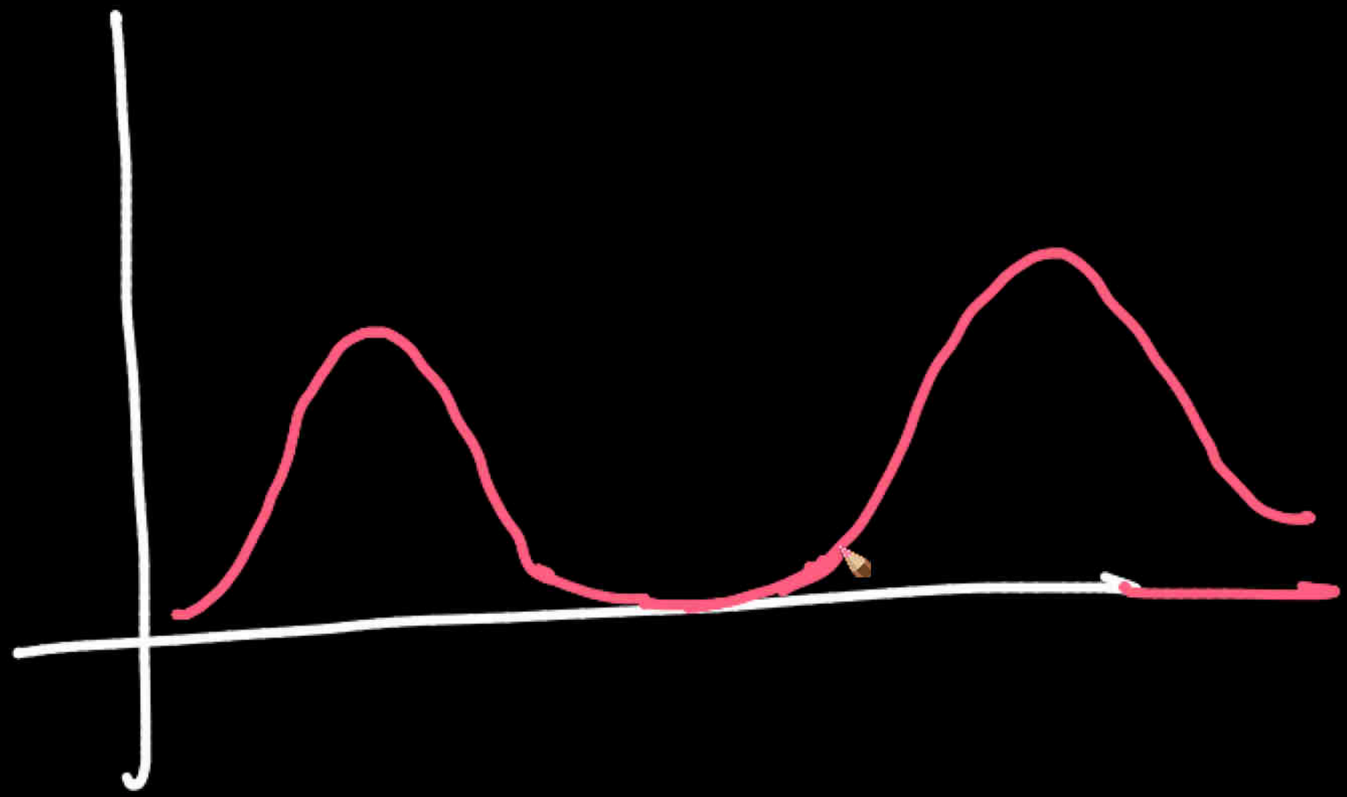
Indians

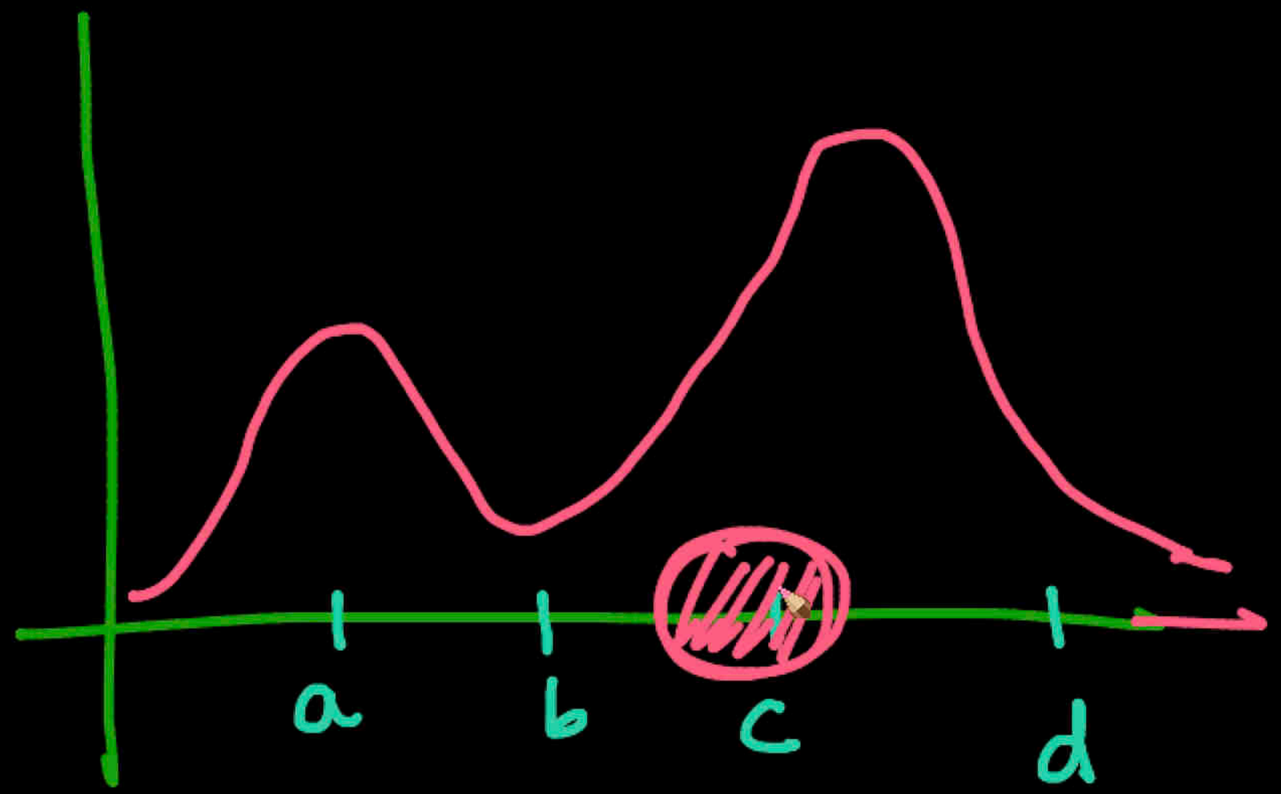
Chinese

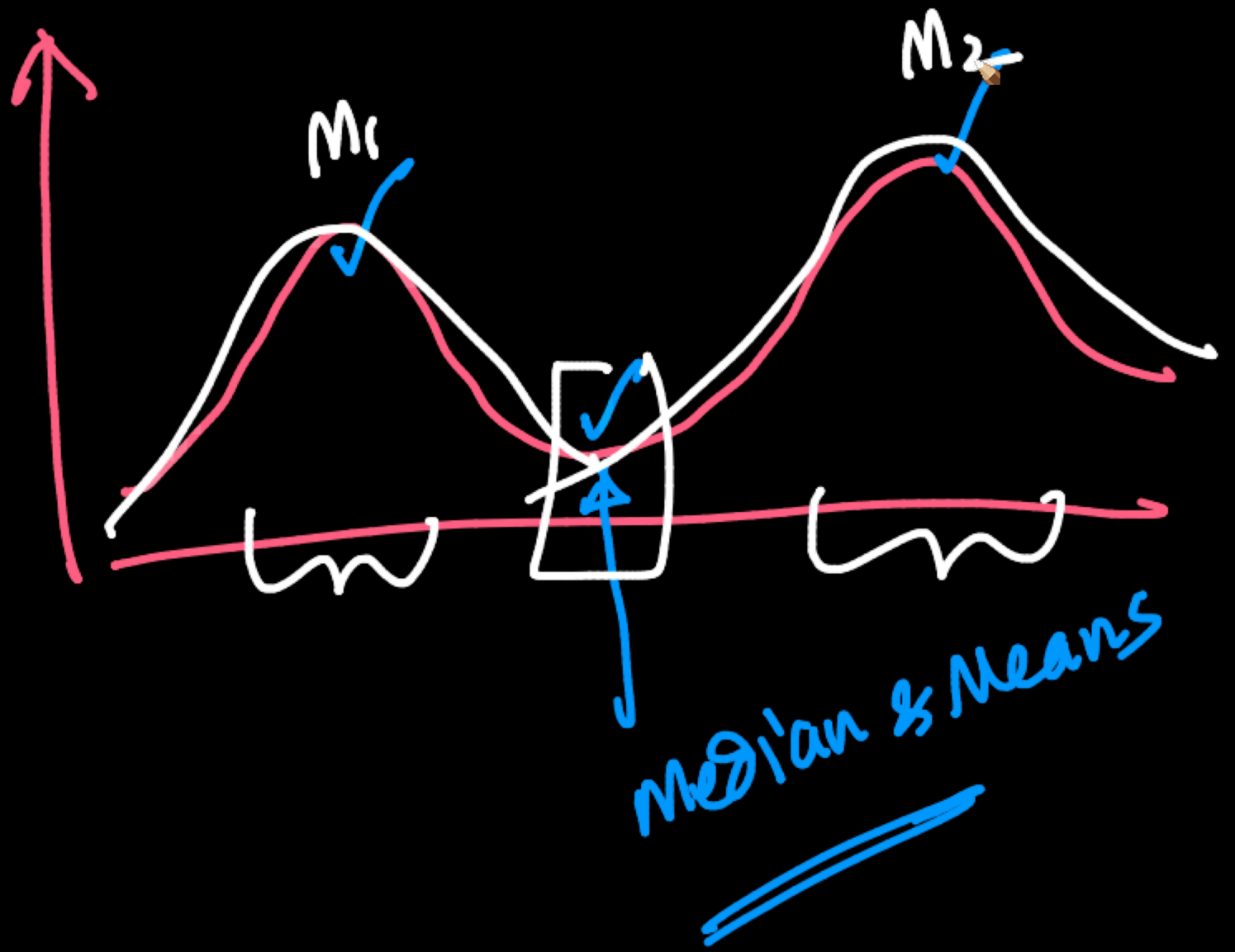
Norweg~

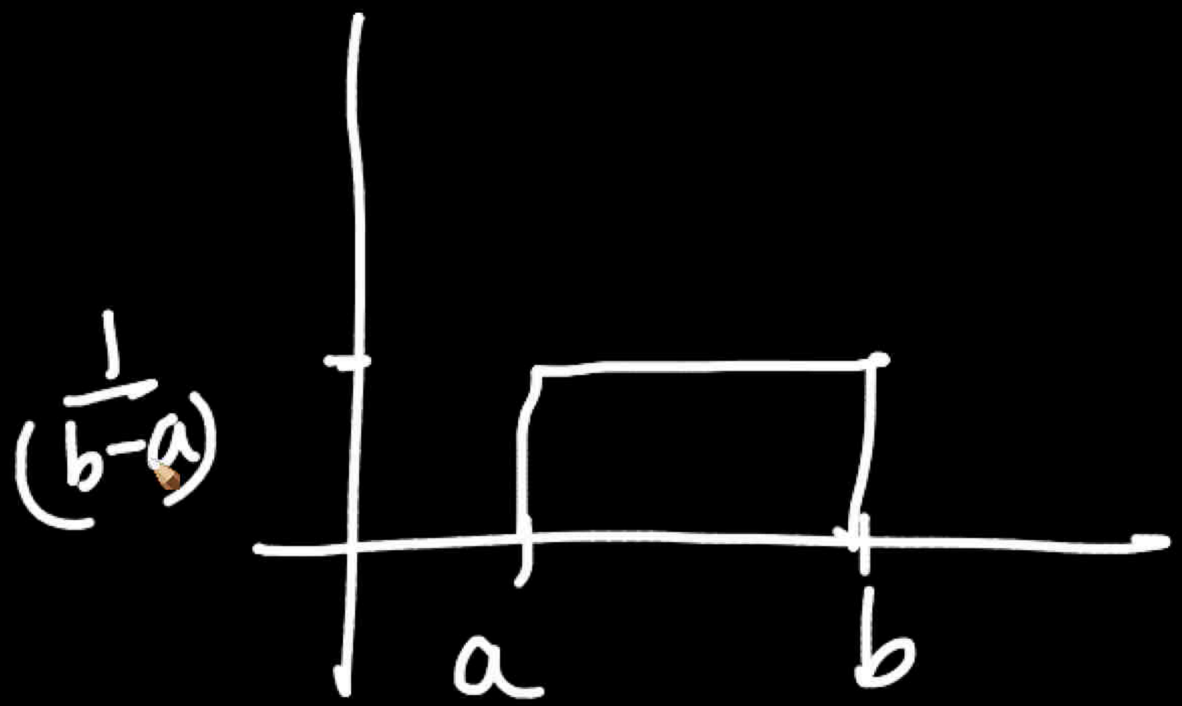


72 / 72



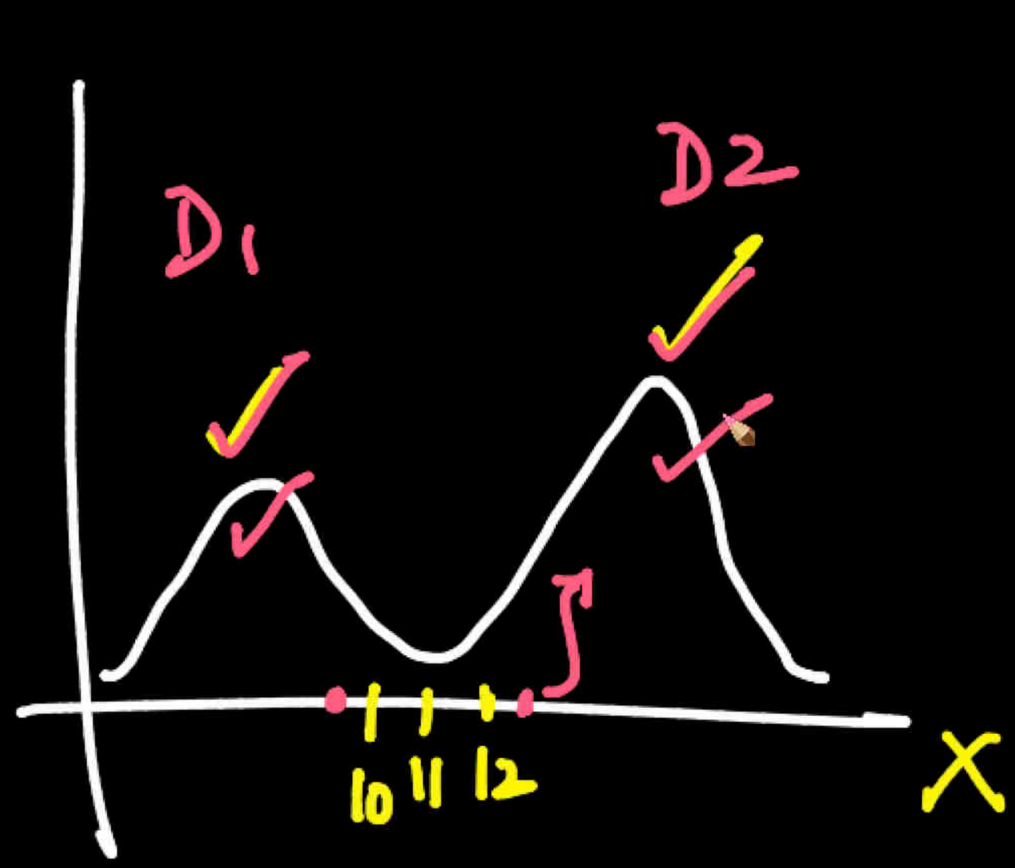

a
b
c
d





$x_1 = 10.5$

PDF



Mixture-models (ML)

X → $x_1, x_2 \dots x_n$

log-normal

del-time

log

right skew

$\log(x_1)$ $\log(x_2) \dots \log(x_n$ incomes

x_1' x_2' $\dots$ x_n' → Gaussian


79 / 79

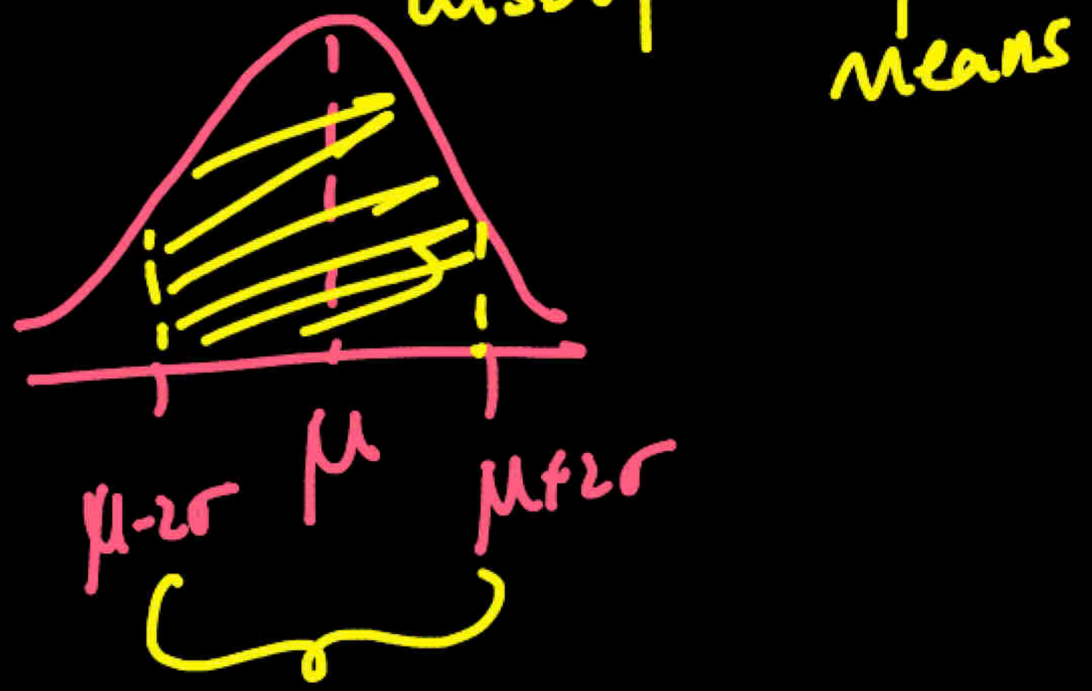
$x_1, x_2 \dots x_m$

$S_1^n \longrightarrow M_1$

$S_2^n \longrightarrow M_2$

$\vdots \qquad \vdots$

$S_\gamma^n \longrightarrow M_\gamma$



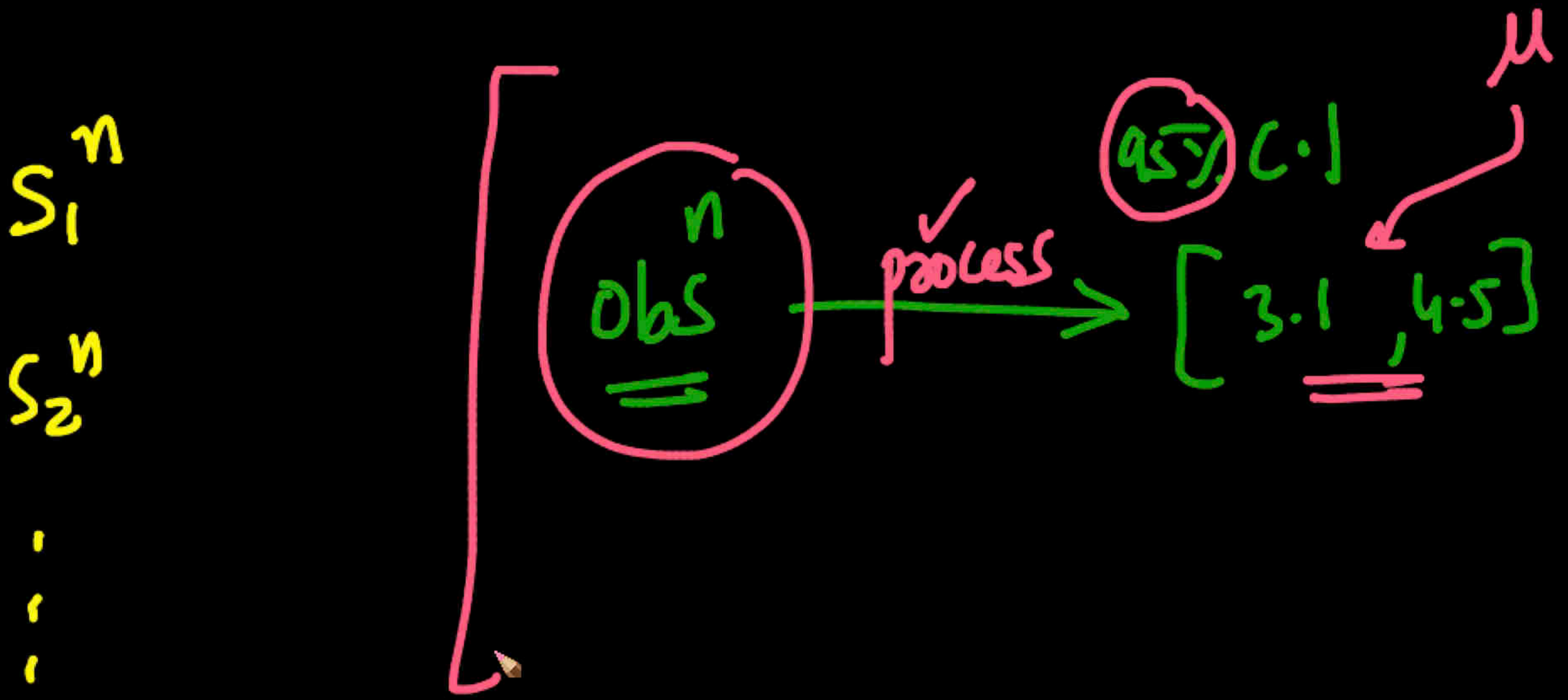
S_1^n

S_2^n

.

.

.



Remedial-2.ipynb

File Edit View Insert Runtime Tools Help Last edited on 8 July

Comment Share

+ Code + Text Connect Editing

Number of surgeries in a hospital per day.

```
[ ]  import numpy as np
     from scipy.stats import poisson, expon
     import matplotlib.pyplot as plt
     import pandas as pd
     import seaborn as sns
```

```
# Number of surgeries in a hospital per day : Poisson
# Sample of 200 observations
V = [ 2,  5,  6,  3,  2,  6,  2, 10,  4,  4,  8,  3,  3,  3,  3,  8,  7,
        6,  9,  4,  5,  5,  5,  5,  4, 10,  9,  4,  0,  8,  4,  6,  5,  5,
        6,  1,  6,  7,  2,  3,  5,  7,  2,  3,  4,  5,  7,  4,  5,  4,  3,
        3,  8,  1,  6,  8,  6,  6,  3,  9,  8,  2,  3,  6,  5,  8,  8,  5,
        7,  7,  6,  7,  6,  6,  9,  3,  6, 12,  4,  4,  3,  2,  2,  2,  4,
        8,  5,  3,  6,  2,  5,  3,  2,  6,  9,  2,  7,  4,  4,  3,  4,  5,
        8,  9,  4,  6,  6,  2,  2,  7,  2, 10,  8,  2,  6,  3,  7,  2,  3,
        8, 11,  4
```