Topics :

- PMF & CDF (Contd..)
- PDF & CDF
- Airline overbooking (solve)
- Bernoulli & Binomial r.v
- Gaussian/ Normal r.v

revised

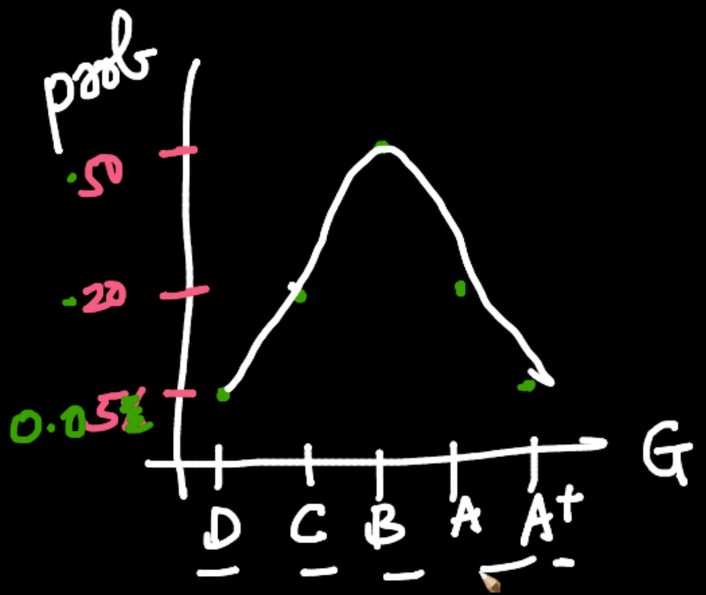
Revise → move @ a decent pace

Ops:
- Ques tab (↱ track...)
- Chat: conversation

Assignments → concepts completed

Q

PMF



Visual

Grade: G

A+ → 5%.

A → 20%.

B → 50%.

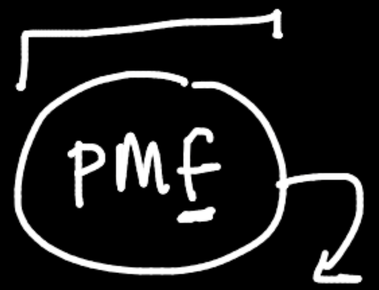
C → 20%.

D → 5%.

– avg –

median

1. discrete
2. ordinal ✓

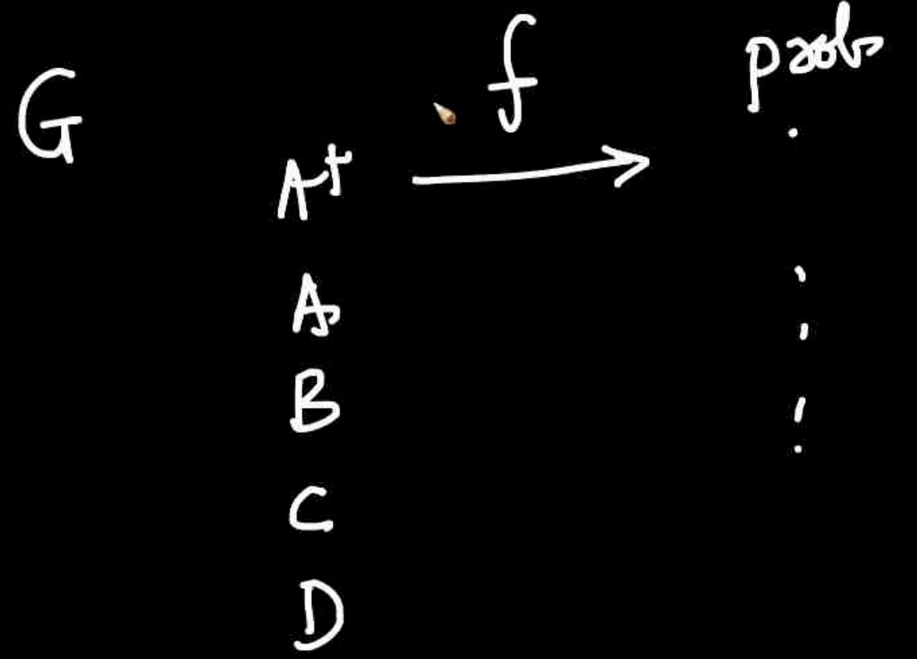


$P(G = A^+) = \checkmark$

$P(G = A) = \checkmark$

⋮

PMF:

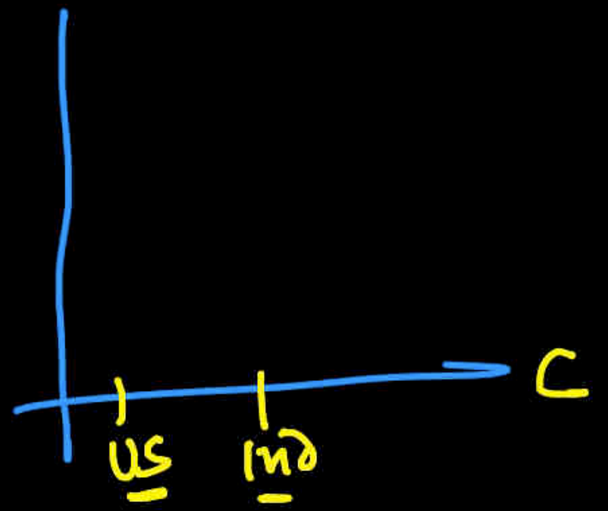


C: countries

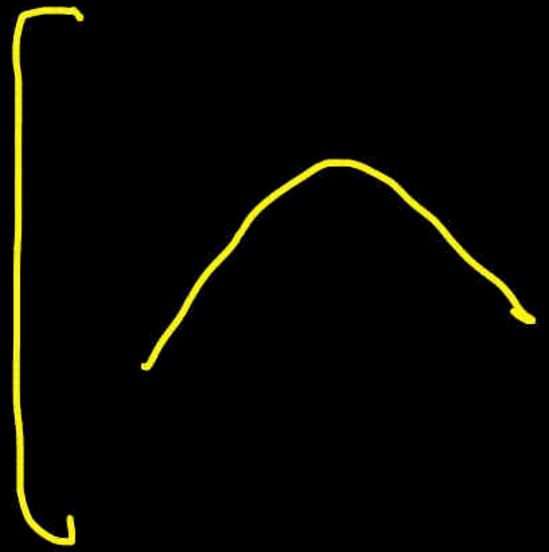
Amazon's Customers (200 M+)

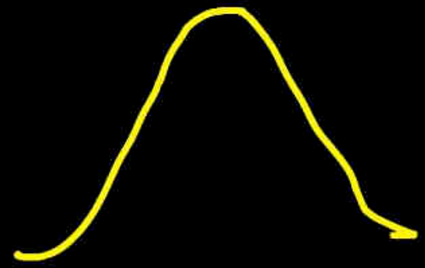
PMF

$$\begin{cases} P(C = US) = 0.5 \\ P(C = Ind) = 0.2 \\ \vdots \end{cases}$$



Misconception





X not always normal/Gaussian

PDF

PMF → $P(X=i)$ (X)

H: heights of learners

↳ continous ; real

need not be integers

150.0 to 190.0 CM ← range

won't work here

INTUITION:

$P(H = 180.0\text{CM}) = 0$

180.000000000000 ....

PMF → discrete

PDF → continous

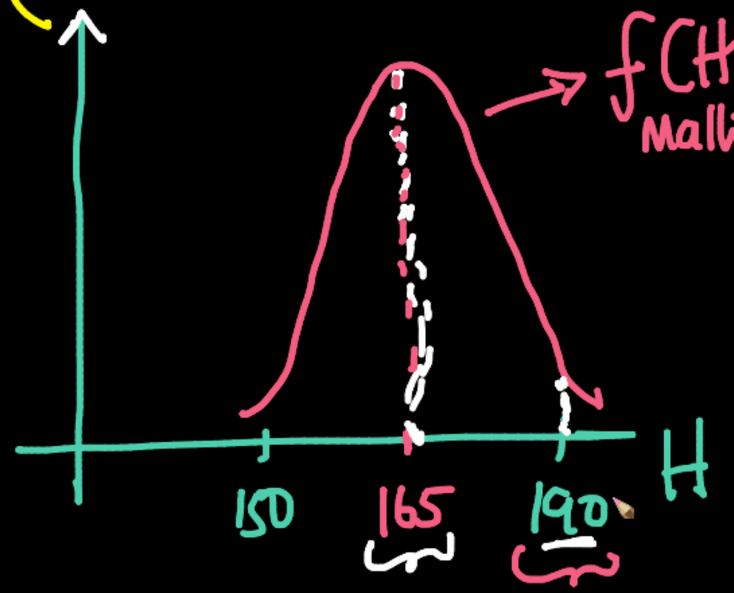
PDF → visually

{ how to construct (later)

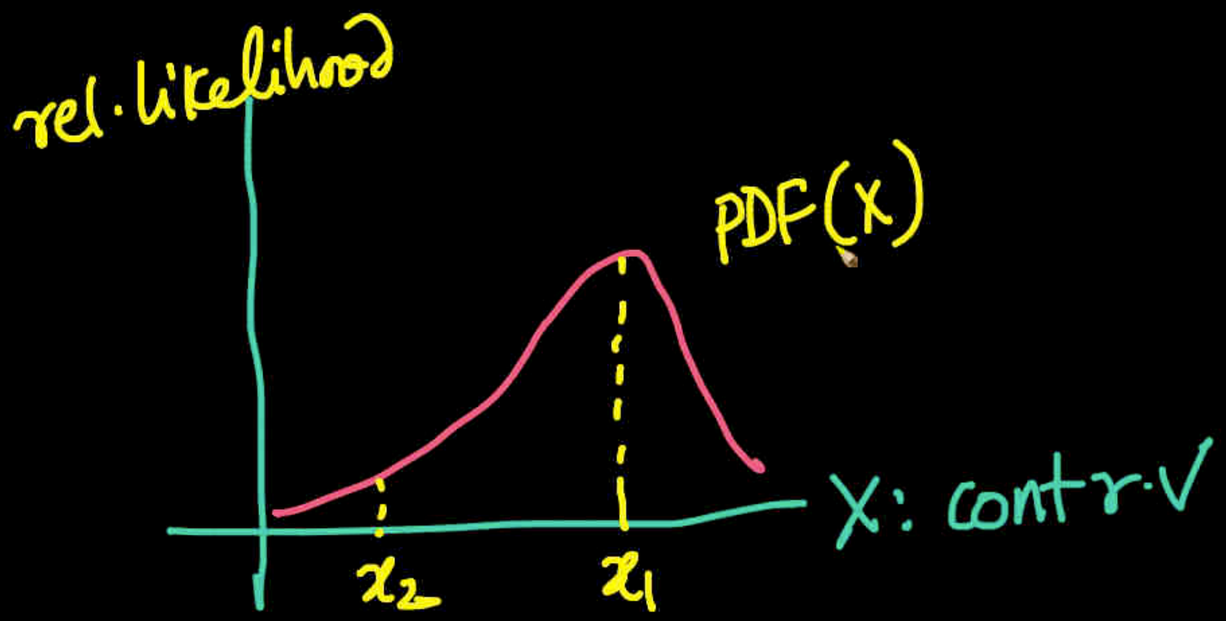
relative likehood

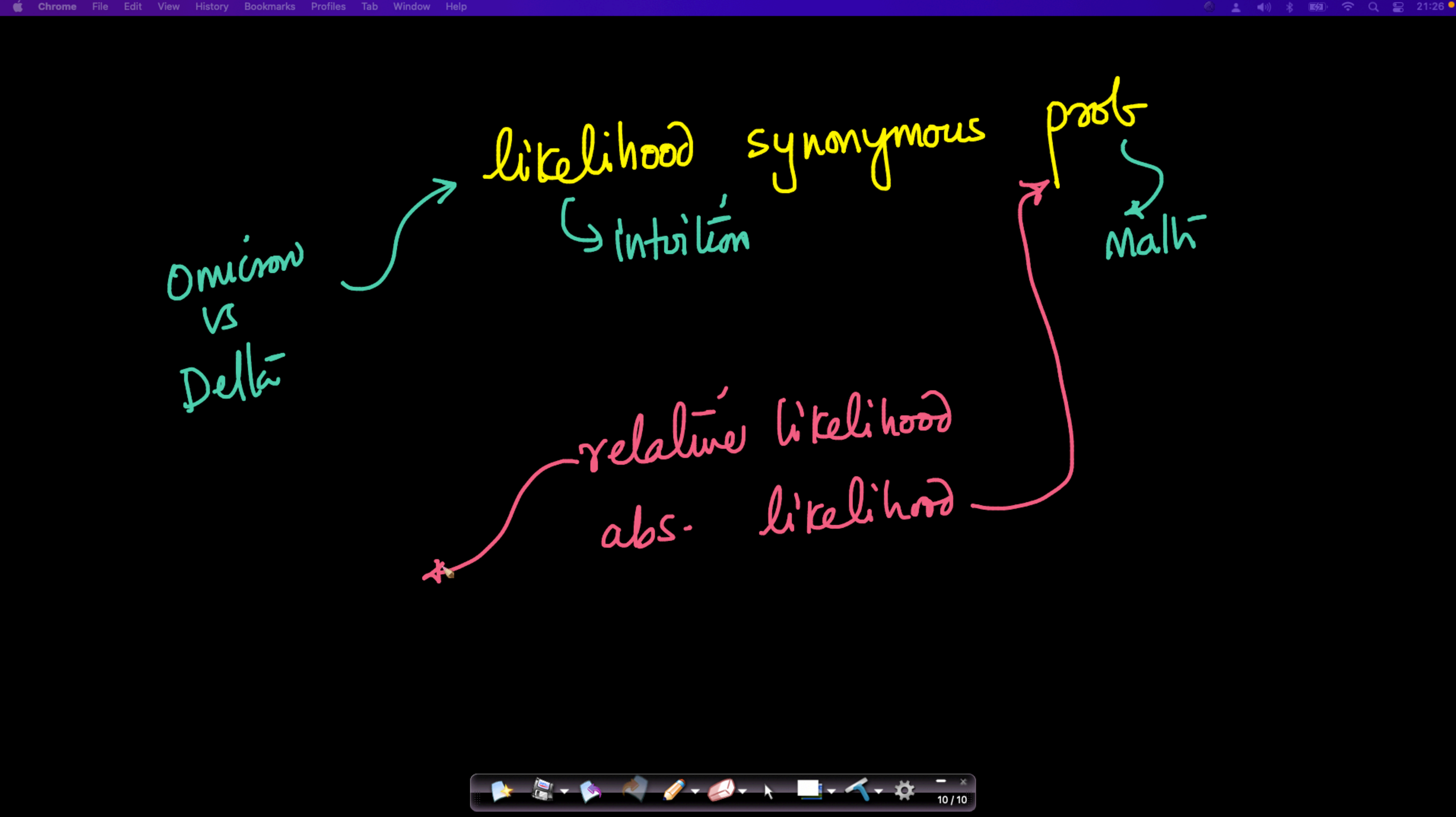
PDF

f(H) Math



absolute likelihood = probability


rel. likelihood
PDF(x)
X: contr. V
x_2
x_1


likelihood synonymous
intuition
proof
Malh-
Omicron vs Delta
relative likelihood
abs. likelihood
10 / 10

H:

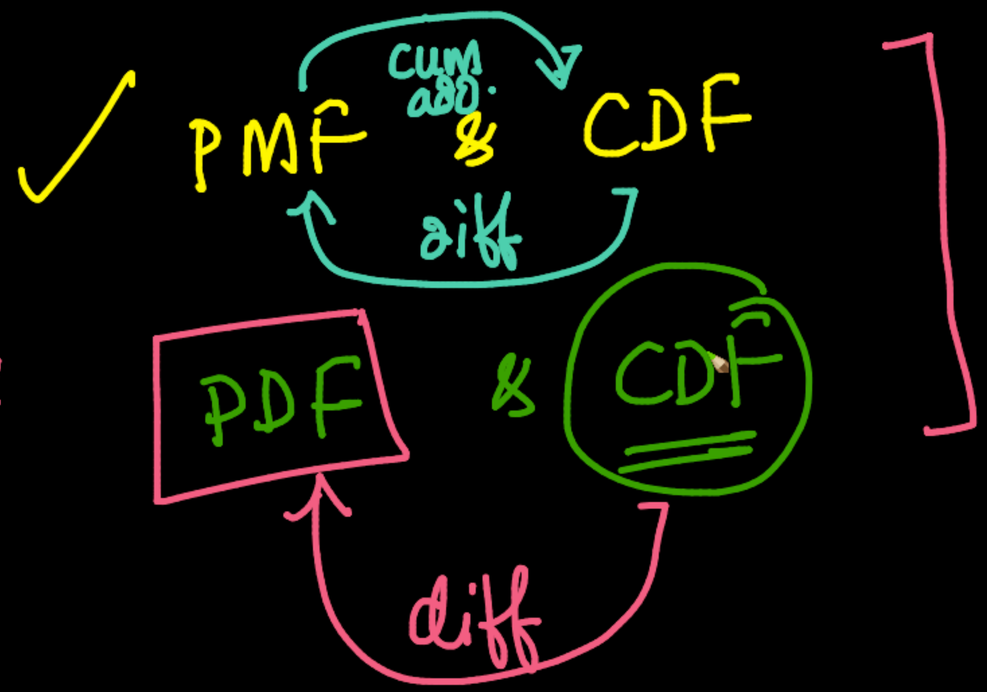
(2)

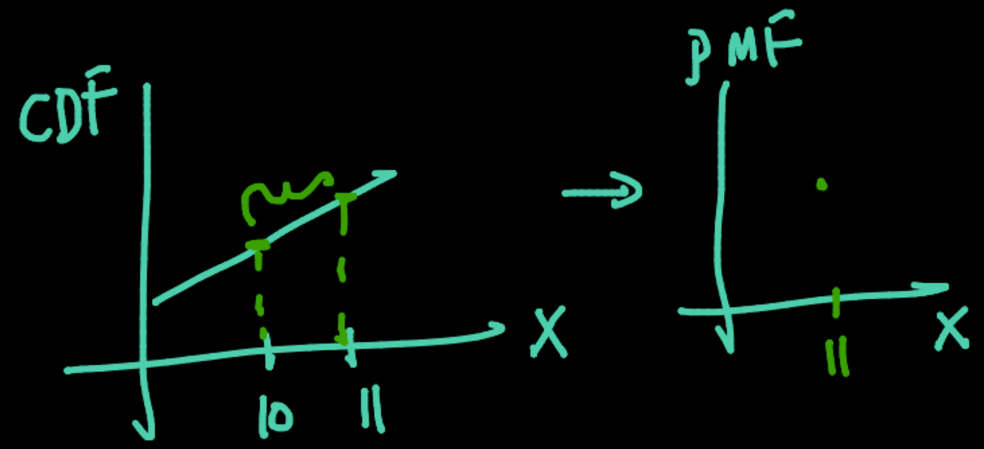
$h_{10} = 175.5$

$P(H = 175.5) = \frac{1}{100}$ X

$H = 175.5000000\ldots$

(1) Sample → 100





H : continous

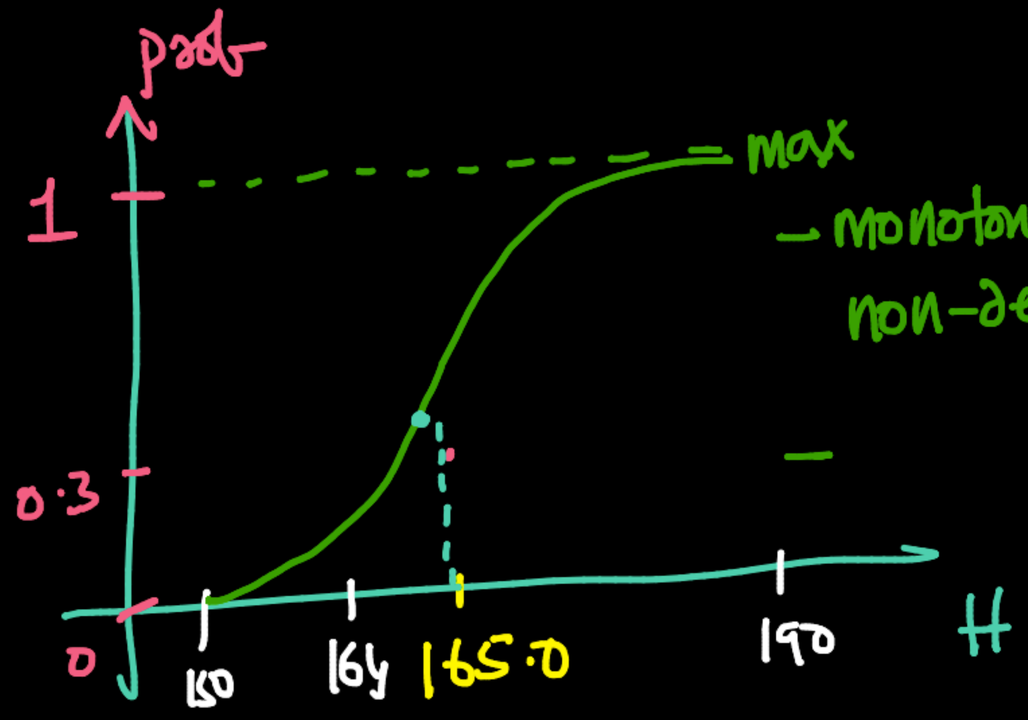
$H_1 \rightarrow 155.5$

$H_2 \rightarrow 160.2$

$H_{100} \rightarrow 181.2$

$\frac{30}{100}$

CDF



$P(H \in 165.0) = \frac{30}{100}$

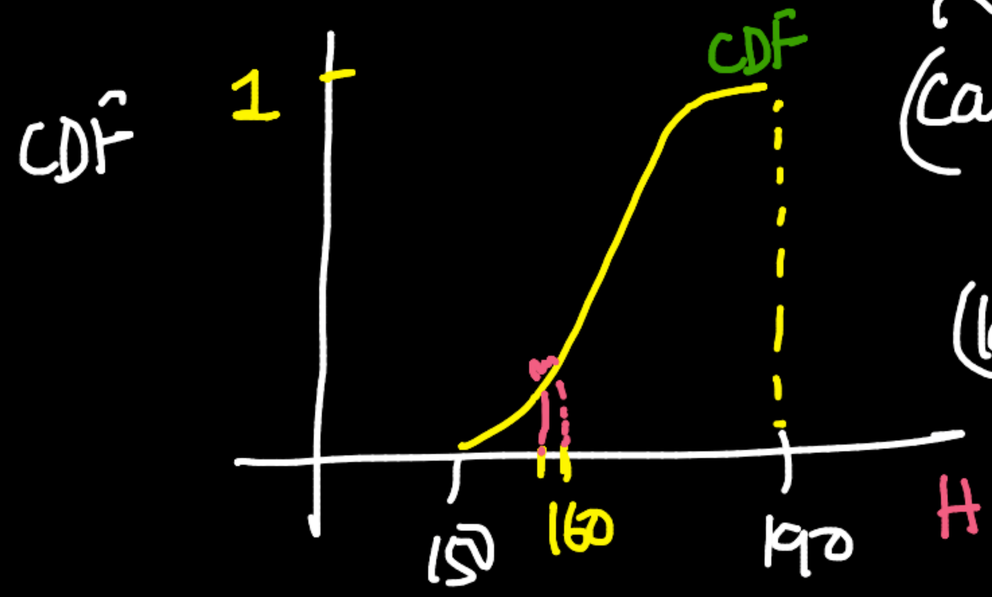
$165.0 \leq 165.0$ ✓

CDF of a Continous r.V

PDF

(calculus)

(later)



NOT probs

rel. likelihood

(lol.y) DONT WORRY

$\frac{d\ CDF(H)}{dH}$

H=160

infinitely small differing

∞-small difference → derivative

CDF → PDF

discrete r.v

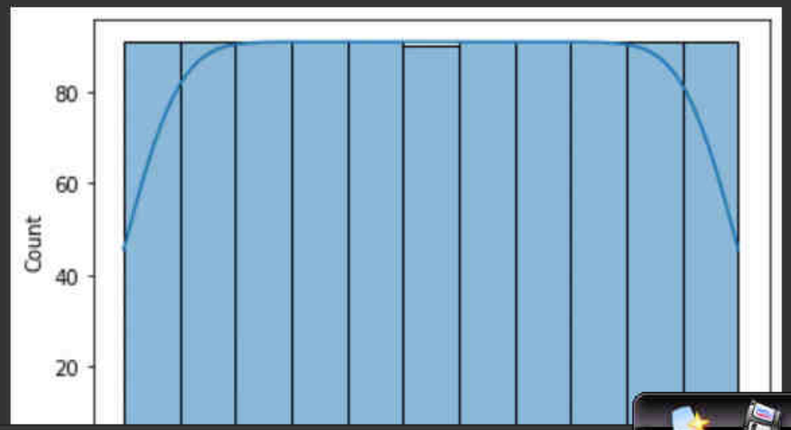
CDF → PMF

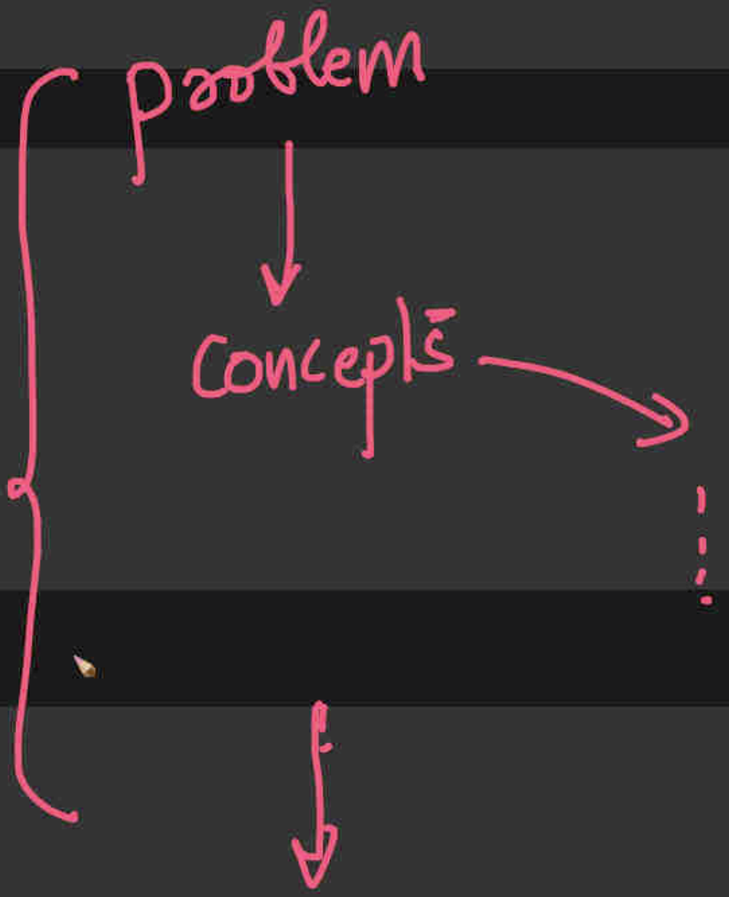
```
dtypes: int64(2), object(1)
memory usage: 23.6+ KB
```

```
[ ] flights.head()
```

	Passenger_ID	Flight_ID	Arrived
0	1811	A320	1
1	1812	A320	1
2	1813	B777	1
3	1814	B737	1
4	1815	B737	1

```
sns.histplot(flights["Passenger_ID"], kde=True)
plt.show()
```





```
dtypes: int64(2), object(1)
memory usage: 23.6+ KB
```

```
[ ] flights.head()
```

	Passenger_ID	Flight_ID	Arrived
0	1811	A320	1
1	1812	A320	1
2	1813	B777	1
3	1814	B737	1
4	1815	B737	1

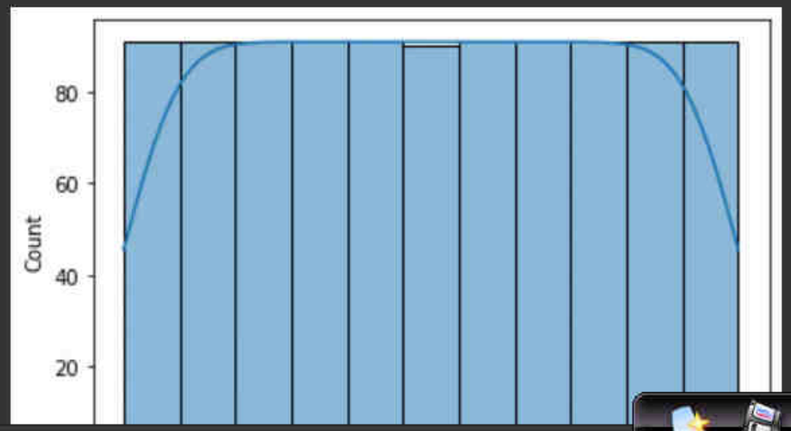
pID: discrete or continous

1000 – 9999

numeric

many values

```
[ ] sns.histplot(flights["Passenger_ID"], kde=True)
    plt.show()
```



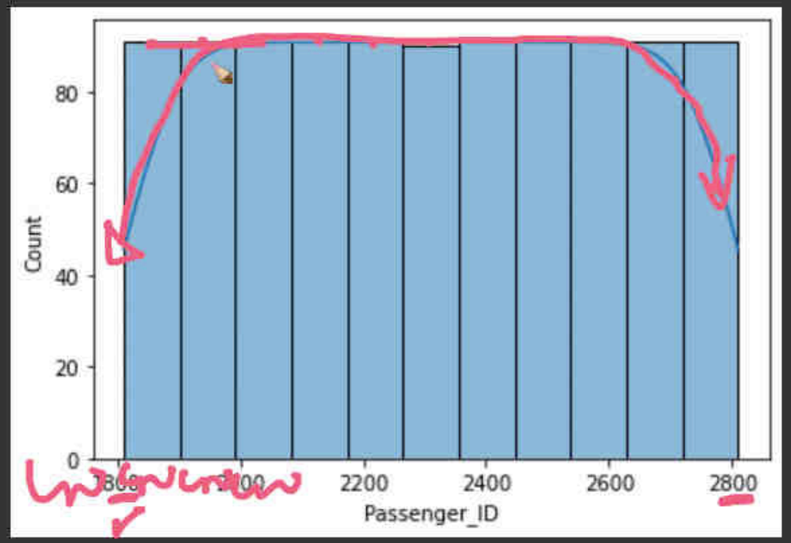
Continous: real-valued (not always)

↳ numeric if there are many values

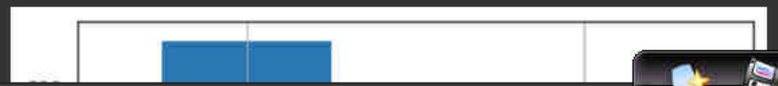
2	1813	B777	1
3	1814	B737	1
4	1815	B737	1

smoothed histogram

```
sns.histplot(flights["Passenger_ID"], kde=True)
plt.show()
```

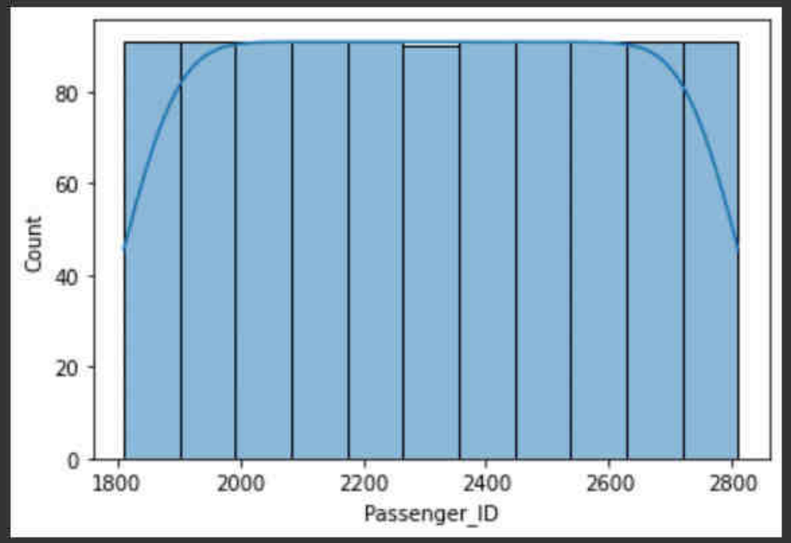


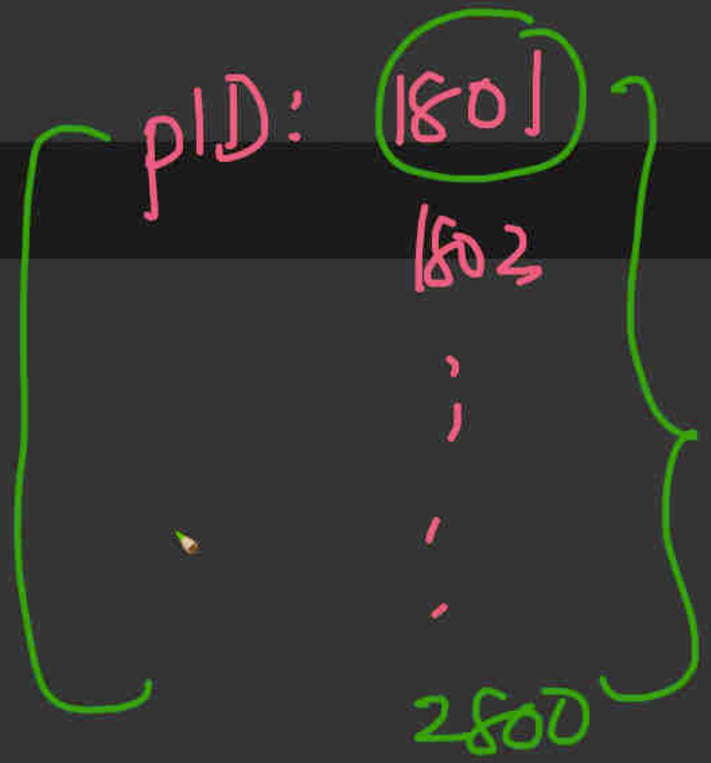

```
flights["Arrived"].value_counts().plot.bar()
plt.grid()
plt.show()
```



2	1813	B777	1
3	1814	B737	1
4	1815	B737	1

```
sns.histplot(flights["Passenger_ID"], kde=True)
plt.show()
```



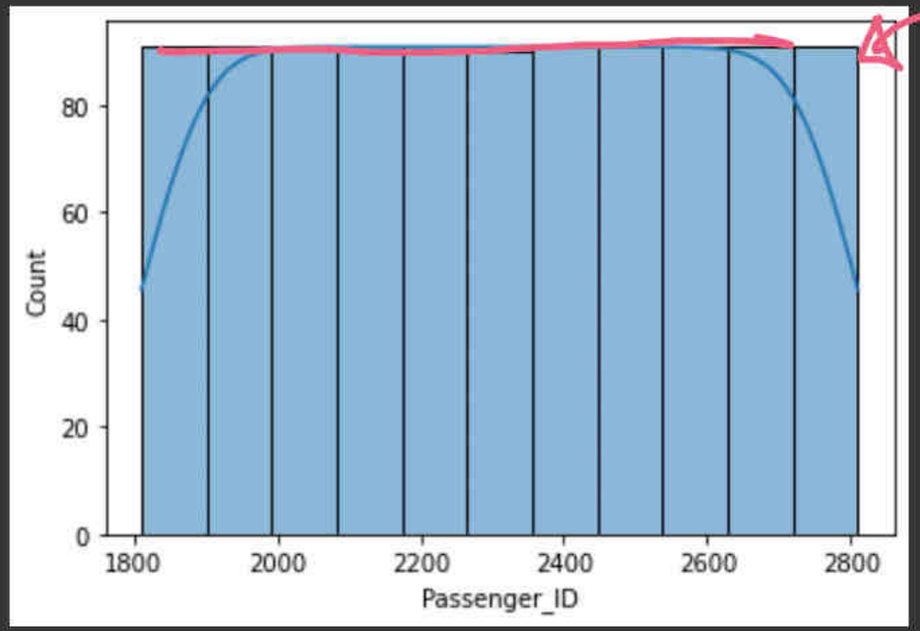



```
flights["Arrived"].value_counts().plot.bar()
plt.grid()
plt.show()
```

2	1813	B777	1
3	1814	B737	1
4	1815	B737	1

Uniform disb discrete r.v
Dice

```
sns.histplot(flights["Passenger_ID"], kde=True)
plt.show()
```



PID: Uniform disb continous r.v

```
flights["Arrived"].value_counts().plot.bar()
```

$$P(H \leq 165.0) = P(H < 165.0)$$

30/100 — 29/100

100

120.0
150.0
161.2
165.0
168.1
⋮

Theoritically – Yes

Practically → may not be equal

How much overlooking?

+ Code + Text

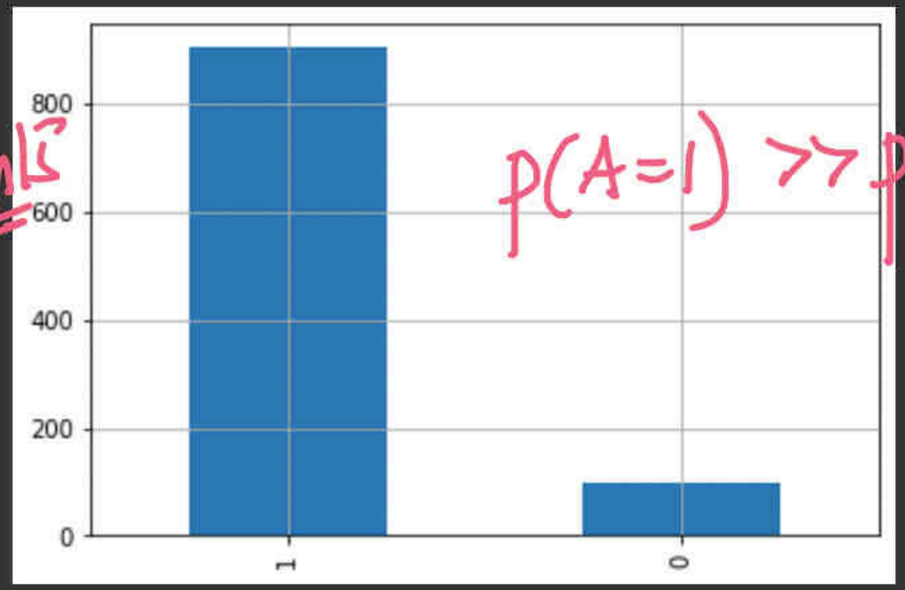
Connect

binary r.v

A → 0 → n_1

→ 1 → n_2

```
[ ] flights["Arrived"].value_counts().plot.bar()
    plt.grid()
    plt.show()
```



Counts

$p(A=1) >> p(A=0)$

$$p(A=1) = \frac{n_2}{n_1+n_2}$$

```
[ ] showsup_probability = flights['Arrived'].value_counts(normalize=True)[1]
    print(showsup_probability)
```

```
0.902
```

```
[ ] showsup_probability = flights['Arrived'].value_counts(normalize=True)[1]
    print(showsup_probability)
```

```
0.902
```

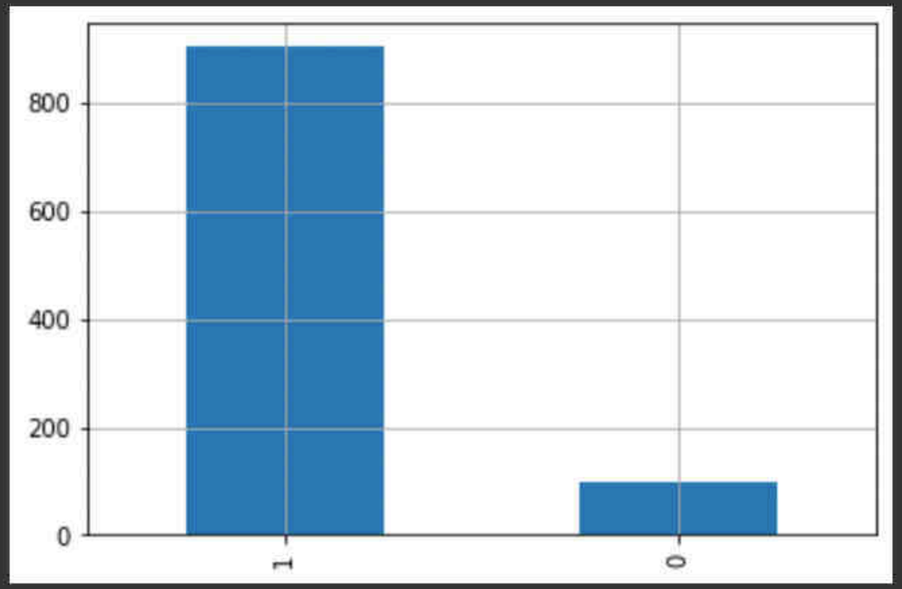
```
[ ] flights['Arrived'].value_counts()
```

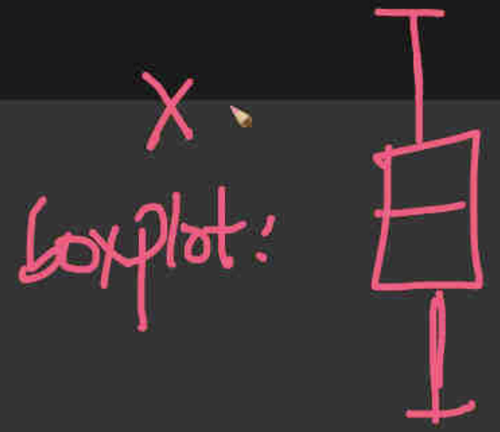
```
1    902
0     98
Name: Arrived, dtype: int64
```

```
[ ] import math

    PENALTY = 10000
    def comb(n, r):

      num1 = math.factorial(n)
      num2 = math.factorial(r)
      num3 = math.factorial(n-r)

      return num1/(num2*num3)
```

+ Code + Text Connect

```
[ ] flights["Arrived"].value_counts().plot.bar()
    plt.grid()
    plt.show()
```






```
[ ] showsup_probability = flights['Arrived'].value_counts(normalize=True)[1]
    print(showsup_probability)
```

```
0.902
```

```
[ ] flights['Arrived'].value_counts()
```

```
[ ] showsup_probability = flights['Arrived'].value_counts(normalize=True)[1]
    print(showsup_probability)
```

```
0.902
```

```
[ ] flights['Arrived'].value_counts()
```

```
1    902
0     98
Name: Arrived, dtype: int64
```

$P(A=1) = 0.9$

```
[ ] import math

    PENALTY = 10000
    def comb(n, r):

      num1 = math.factorial(n)
      num2 = math.factorial(r)
      num3 = math.factorial(n-r)

      return num1/(num2*num3)
```

```
[ ] def calculate_expected_penalty(ticket_sold):
```

$A = 1$

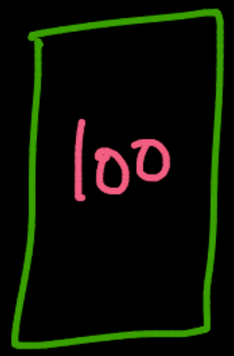
$P(\text{shows up}) = 0.9$

across all planes on an avg..

$P(\text{not shows up}) = 0.1$

$A = 0$

cap



overbook: 'x' seats (let)

$90\% \text{ of } (100 + x) \leq 100$

cap

x

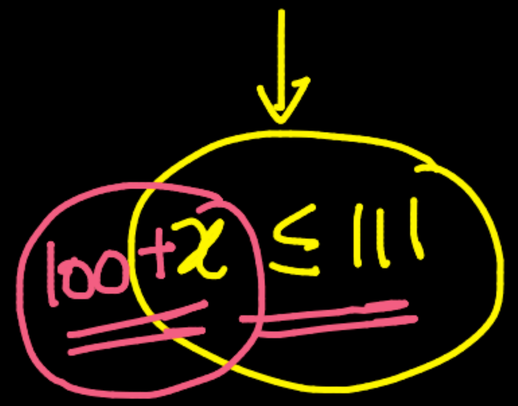
on an average

prob of

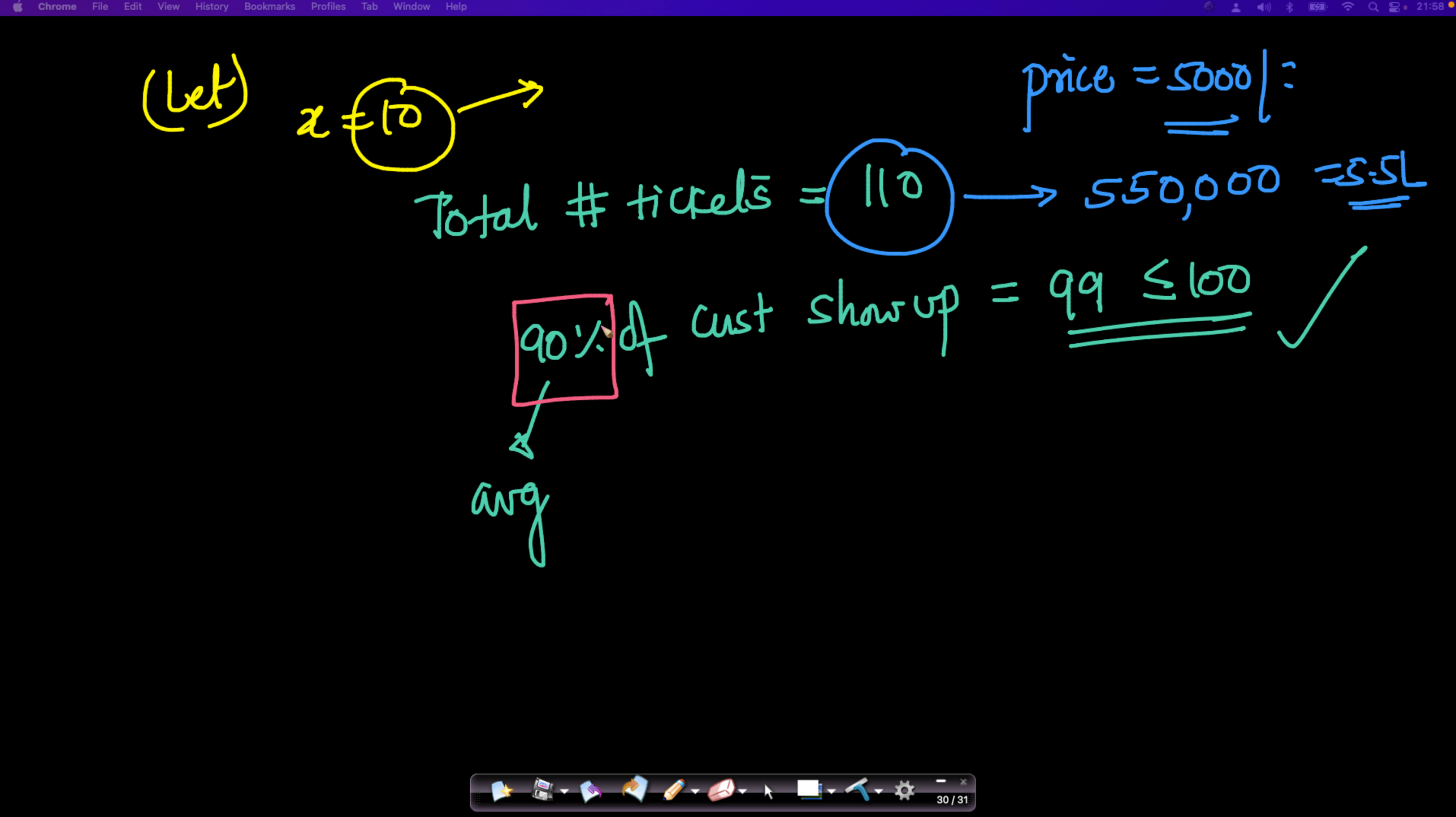
showing up

$$0.9(100+x) \leq 100$$

$$100+x \leq \frac{100}{0.9}$$



$$x \leq 11$$ ✓


(Let) x = 10
price = 5000 /:
Total # tickets = 110
550,000 = 5.5L
90% of cust show up = 99 ≤ 100
avg

Penalty = 10,000 Rs

110

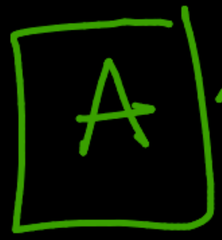
Net_Rev = (110 x 5000) - 10,000

101 people showed-up

= 540,000

1 extra

binary



0 → failure

$P(A=0) = 0.1 = q \quad q = 1-p$

1 → success

$P(A=1) = 0.9 = p$

$p+q=1$

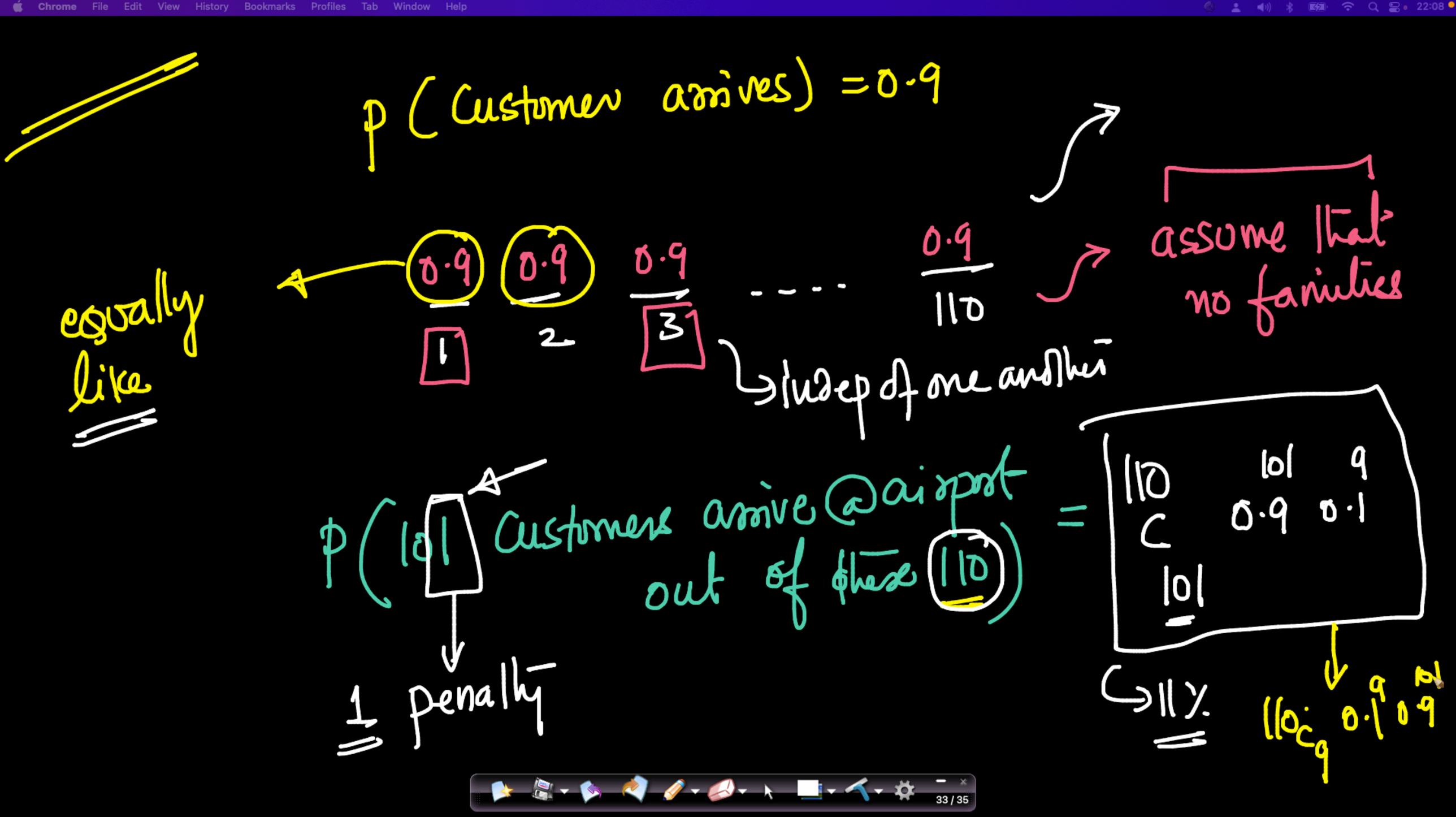
takes 2-values

→ mutually excl & exhaustive

Bernoulli r.v

$X \sim \text{Bernoulli}(p=0.9)$

Discrete: PMF & CDF


p (Customer arrives) = 0.9
equally like
0.9
0.9
0.9
1
2
3
0.9
110
assume that no families
independent of one another
p (101 Customers arrive @ airport out of these 110) =
1 penalty
110 C 101
101
9
0.9
0.1
≈ 11%
loc 9 0.1 0.9

$$P(A \text{ and } B) = P(A)\,P(B) \quad \text{iff}$$

A & B are indep

A: bernouli r.V → 0, 1 ; p = 0.9

Coin toss → H : S, T : F

p = 0.5 (Bernouli)

Binomial r.V

100 n times

p = 0.5 (same prob. of success)

10 S successes

represents → count # successes in n indep bernouli trials

Airline:

Bernoulli r.V

Success

Count # passengs @ airport out of $n=110$ → independent

↓

A : bernoulli $p=0.9$

Coin toss: H, T Bernoulli ($p=0.5$)

r.v

X : → Count of # success (heads) in $n=100$ tosses

↓

$$P(X = s=10 \mid X \sim \text{Binomial}(n=100; p=0.5))$$

$$= {}^{n}C_{s}\, p^{s}(1-p)^{n-s}$$

X: # passengers who arrive @ airport

$X \sim \text{Binomial}(n = 110 ; p = 0.9)$

$P(X = 101) = {}^{110}C_{101} \, 0.9^{101} \, 0.1^{9} =$ 

Chance of penalty of 10,000$

1000 flights → 11% of flights will have 101 passengers

{ 110 tickets
  100 : cap

110 one extra passenger

Bernoulli trail → tossing a coin

Binomial r.v ← count of # successes in n bernoulli trials

$$P(X=101 \mid X \sim Bin(n=110;\ p=0.9)) = 11\%$$

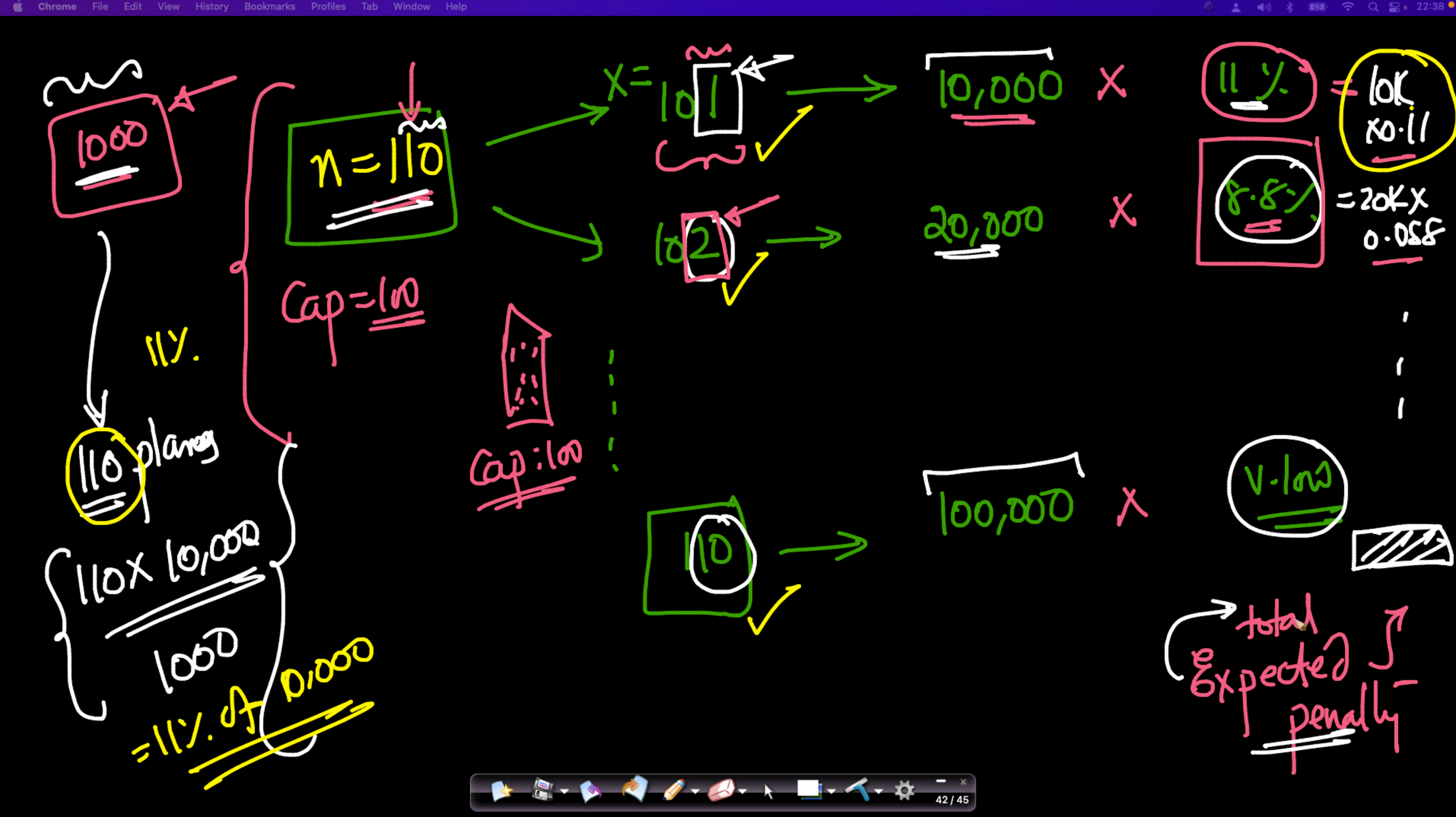
$$P(X=102 \mid X \sim Bin(n=110;\ p=0.9)) =$$

$$\binom{110}{102} 0.9^{102}\, 0.1^{8} = 8.8\%$$

not $X \leq 102$

$\hookrightarrow$ 2 penalties

$\hookrightarrow$ $2 \times 10{,}000$

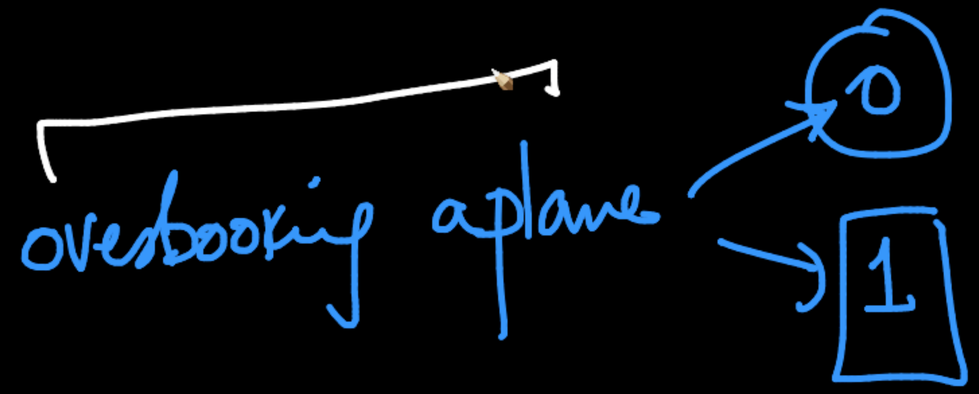

1000
n = 110
Cap = 100
11%
110 planes
110 x 10,000
1000
= 11% of 10,000
X =
101
10,000
x
11%
= 10K x 0.11
102
20,000
x
8.8%
= 20K x 0.088
Cap: 100
110
100,000
x
V. low
total Expected penalty

each passenger @ airport → 0
→ 1 $p = 0.9$

X each plane → 95
→ 96
→ 100
→ 102
⋮

different formulation

- n planes → binomial r.v
- p = prob of overbooking a plane → bernoulli r.v



#bookings $n = 100 \text{ — } 1000$

Code

Max. net-rev

= Total-rev − penalty expected total

IIO as an example

110 → net rev$_{110}$ − total exp. penalty$_{110}$

111

112

113

:

:

/

```
0     98
Name: Arrived, dtype: int64
```

```
[ ] import math

    PENALTY = 10000
    def comb(n, r):

      num1 = math.factorial(n)
      num2 = math.factorial(r)
      num3 = math.factorial(n-r)

      return num1/(num2*num3)
```

```
[ ] def calculate_expected_penalty(ticket_sold):

        total_penalty = 0.0

        for i in range(1, ticket_sold - 100+1):
            ##pmf for k successes, n trials, p=success probab
            prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
            penalty = prob*PENALTY*i
            total_penalty += penalty
```

```
0     98
Name: Arrived, dtype: int64
```

```python
import math

PENALTY = 10000
def comb(n, r):

  num1 = math.factorial(n)
  num2 = math.factorial(r)
  num3 = math.factorial(n-r)

  return num1/(num2*num3)
```

$$\frac{n!}{r!\,(n-r)!} = {}^{n}C_{r}$$

```python
def calculate_expected_penalty(ticket_sold):

    total_penalty = 0.0

    for i in range(1, ticket_sold - 100+1):
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty
```

```
    num2 = math.factorial(r)
    num3 = math.factorial(n-r)

    return num1/(num2*num3)
```

```
def calculate_expected_penalty(ticket_sold):

    total_penalty = 0.0

    for i in range(1, ticket_sold - 100+1):
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty

    return total_penalty
```

```
for i in range(100, 150):
    sales = 5000*i
    penalty  = calculate_expected_penalty(i)
    netsales = (sales - penalty)
    print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
```

```
    num2 = math.factorial(r)
    num3 = math.factorial(n-r)

    return num1/(num2*num3)
```

```
def calculate_expected_penalty(ticket_sold):

    total_penalty = 0.0

    for i in range(1, ticket_sold - 100+1):
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty

    return total_penalty
```

```
for i in range(100, 150):
    sales = 5000*i
    penalty  = calculate_expected_penalty(i)
    netsales = (sales - penalty)
    print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
```

```
    num2 = math.factorial(r)
    num3 = math.factorial(n-r)

    return num1/(num2*num3)
```

$$n_{C_r} p^r (1-p)^{n-r}$$

```
def calculate_expected_penalty(ticket_sold):

    total_penalty = 0.0

    for i in range(1, ticket_sold - 100+1):
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty

    return total_penalty
```

```
for i in range(100, 150):
    sales = 5000*i
    penalty  = calculate_expected_penalty(i)
    netsales = (sales - penalty)
    print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
```

```
    num2 = math.factorial(r)
    num3 = math.factorial(n-r)

    return num1/(num2*num3)
```


```
def calculate_expected_penalty(ticket_sold):

    total_penalty = 0.0

    for i in range(1, ticket_sold - 100+1):
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty

    return total_penalty
```

$X \sim Bin(n; p)$

↳ discrete

```
for i in range(100, 150):
    sales = 5000*i
    penalty  = calculate_expected_penalty(i)
    netsales = (sales - penalty)
    print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
```

```
    num2 = math.factorial(r)
    num3 = math.factorial(n-r)

    return num1/(num2*num3)
```

n ; p ;

```
def calculate_expected_penalty(ticket_sold):

    total_penalty = 0.0

    for i in range(1, ticket_sold - 100+1):
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty

    return total_penalty
```

1,2,3, .. 10

0.9

101, 102, .. 110

P(X=k | X~Bin(n,p))

PMF

```
for i in range(100, 150):
    sales = 5000*i
    penalty  = calculate_expected_penalty(i)
    netsales = (sales - penalty)
    print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
```

53 / 58

Search the docs ...

(scipy.sparse.csgraph)

Spatial algorithms and data structures
(scipy.spatial)

Distance computations
(scipy.spatial.distance)

Special functions (scipy.special)

Statistical functions (scipy.stats)

Result classes

Contingency table functions
(scipy.stats.contingency)

Statistical functions for masked arrays
(scipy.stats.mstats)

Quasi-Monte Carlo submodule
(scipy.stats.qmc)

Random Number Generators
(scipy.stats.sampling)

Low-level callback functions

hypergeom, **nbinom**, **nhypergeom**

Notes

The probability mass function for **binom** is:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for $k \in \{0, 1, \ldots, n\}, 0 \leq p \leq 1$

binom takes n and p as shape parameters, where p is the probability of a single success and $1 - p$ is the probability of a single failure.

The probability mass function above is defined in the "standardized" form. To shift distribution use the `loc` parameter. Specifically, `binom.pmf(k, n, p, loc)` is identically equivalent to `binom.pmf(k - loc, n, p)`.

Examples

```
>>> from scipy.stats import binom
>>> import matplotlib.pyplot as plt
>
```

54 / 59

Search the docs ...

(scipy.sparse.csgraph)

Spatial algorithms and data structures (scipy.spatial)

Distance computations (scipy.spatial.distance)

Special functions (scipy.special)

Statistical functions (scipy.stats)

Result classes

Contingency table functions (scipy.stats.contingency)

Statistical functions for masked arrays (scipy.stats.mstats)

Quasi-Monte Carlo submodule (scipy.stats.qmc)

Random Number Generators (scipy.stats.sampling)

Low-level callback functions

hypergeom, **nbinom**, **nhypergeom**

Notes

The probability mass function for **binom** is:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for $k \in \{0, 1, \ldots, n\}, 0 \leq p \leq 1$

binom takes n and p as shape parameters, where p is the probability of a single success and $1 - p$ is the probability of a single failure.

The probability mass function above is defined in the "standardized" form. To shift distribution use the `loc` parameter. Specifically, `binom.pmf(k, n, p, loc)` is identically equivalent to `binom.pmf(k - loc, n, p)`.

Examples

```
>>> from scipy.stats import binom
>>> import matplotlib.pyplot as plt
>
```

55 / 60

+ Code + Text Connect

120

```
def calculate_expected_penalty(ticket_sold):

    total_penalty = 0.0

    for i in range(1, ticket_sold - 100+1):
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty

    return total_penalty
```

```
[ ] for i in range(100, 150):
        sales = 5000*i
        penalty  = calculate_expected_penalty(i)
        netsales = (sales - penalty)
        print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
Total seats: 101 , Net Sales : 505000
Total seats: 102 , Net Sales : 509996
Total seats: 103 , Net Sales : 514979
Total seats: 104 , Net Sales : 519913
Total seats: 105 , Net Sales : 524725
```

```
    return total_penalty
```

```
[ ] for i in range(100, 150):
        sales = 5000*i
        penalty  = calculate_expected_penalty(i)
        netsales = (sales - penalty)
        print("Total seats:",i, ", Net Sales :",round(netsales))
```

100 →

```
Total seats: 100 , Net Sales : 500000
Total seats: 101 , Net Sales : 505000
Total seats: 102 , Net Sales : 509996
Total seats: 103 , Net Sales : 514979
Total seats: 104 , Net Sales : 519913
Total seats: 105 , Net Sales : 524725
Total seats: 106 , Net Sales : 529288
Total seats: 107 , Net Sales : 533425
Total seats: 108 , Net Sales : 536929
Total seats: 109 , Net Sales : 539603
Total seats: 110 , Net Sales : 541302
Total seats: 111 , Net Sales : 541959
Total seats: 112 , Net Sales : 541595
Total seats: 113 , Net Sales : 540305
Total seats: 114 , Net Sales : 538233
Total seats: 115 , Net Sales : 535544
Total seats: 116 , Net Sales : 532393
Total seats: 117 , Net Sales : 528919
Total seats: 118 , Net Sales : 525227
```

```
    return total_penalty
```

```
[ ] for i in range(100, 150):
        sales = 5000*i
        penalty  = calculate_expected_penalty(i)
        netsales = (sales - penalty)
        print("Total seats:",i, ", Net Sales :",round(netsales))
```

sales -

```
Total seats: 100 , Net Sales : 500000
Total seats: 101 , Net Sales : 505000
Total seats: 102 , Net Sales : 509996
Total seats: 103 , Net Sales : 514979
Total seats: 104 , Net Sales : 519913
Total seats: 105 , Net Sales : 524725
Total seats: 106 , Net Sales : 529288
Total seats: 107 , Net Sales : 533425
Total seats: 108 , Net Sales : 536929
Total seats: 109 , Net Sales : 539603
Total seats: 110 , Net Sales : 541302
Total seats: 111 , Net Sales : 541959
Total seats: 112 , Net Sales : 541595
Total seats: 113 , Net Sales : 540305
Total seats: 114 , Net Sales : 538233
Total seats: 115 , Net Sales : 535544
Total seats: 116 , Net Sales : 532393
Total seats: 117 , Net Sales : 528919
Total seats: 118 , Net Sales : 525227
```

```
        ##pmf for k successes, n trials, p=success probab
        prob = stats.binom.pmf(k=100+i,n=ticket_sold,p=showsup_probability)
        penalty = prob*PENALTY*i
        total_penalty += penalty

    return total_penalty
```

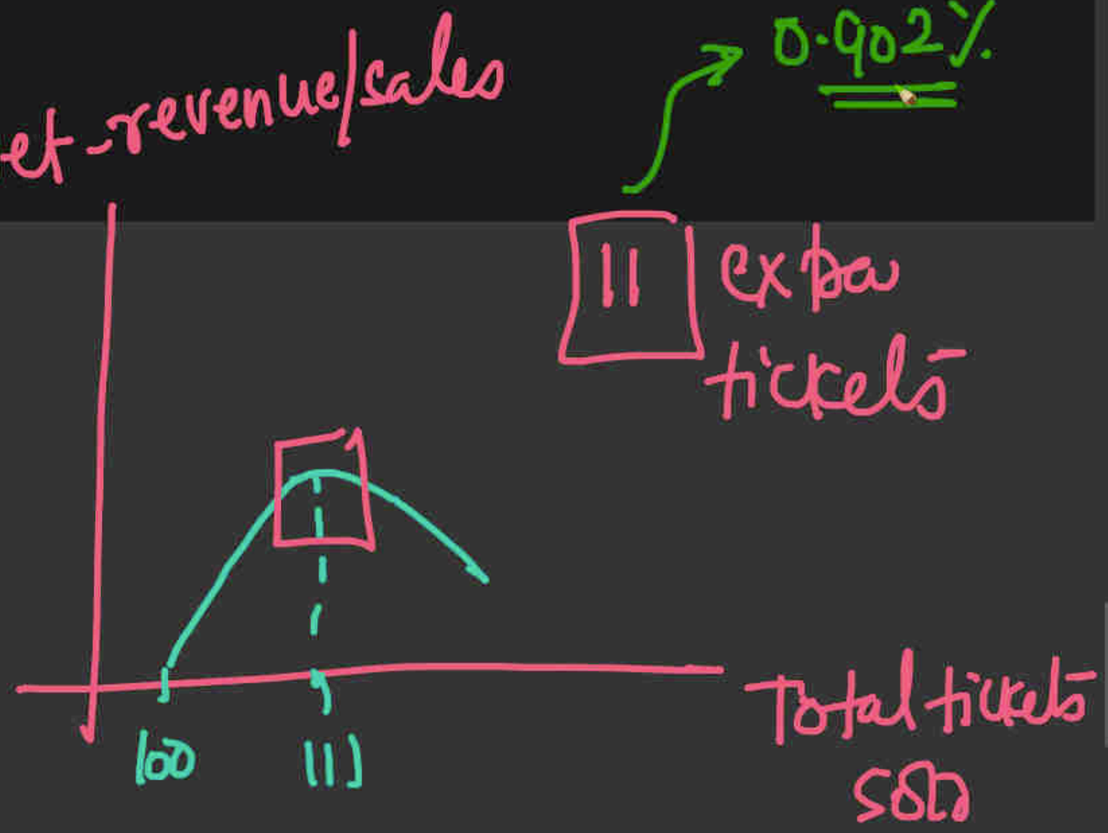
```
[ ] for i in range(100, 150):
        sales = 5000*i
        penalty  = calculate_expected_penalty(i)
        netsales = (sales - penalty)
        print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
Total seats: 101 , Net Sales : 505000
Total seats: 102 , Net Sales : 509996
Total seats: 103 , Net Sales : 514979
Total seats: 104 , Net Sales : 519913
Total seats: 105 , Net Sales : 524725
Total seats: 106 , Net Sales : 529288
Total seats: 107 , Net Sales : 533425
Total seats: 108 , Net Sales : 536929
Total seats: 109 , Net Sales : 539603
Total seats: 110 , Net Sales : 541302
Total seats: 111 , Net Sales : 541959
Total seats: 112 , Net Sales : 541595
```

```
    return total_penalty
```

```
[ ] for i in range(100, 150):
        sales = 5000*i
        penalty  = calculate_expected_penalty(i)
        netsales = (sales - penalty)
        print("Total seats:",i, ", Net Sales :",round(netsales))
```

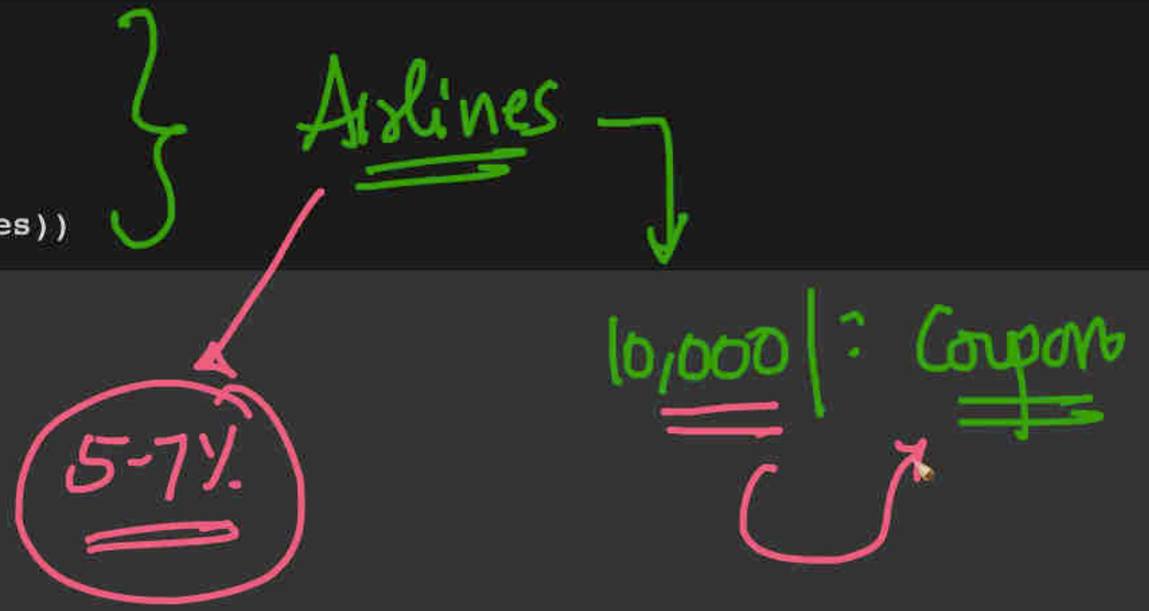
```
Total seats: 100 , Net Sales : 500000
Total seats: 101 , Net Sales : 505000
Total seats: 102 , Net Sales : 509996
Total seats: 103 , Net Sales : 514979
Total seats: 104 , Net Sales : 519913
Total seats: 105 , Net Sales : 524725
Total seats: 106 , Net Sales : 529288
Total seats: 107 , Net Sales : 533425
Total seats: 108 , Net Sales : 536929
Total seats: 109 , Net Sales : 539603
Total seats: 110 , Net Sales : 541302
Total seats: 111 , Net Sales : 541959
Total seats: 112 , Net Sales : 541595
Total seats: 113 , Net Sales : 540305
Total seats: 114 , Net Sales : 538233
Total seats: 115 , Net Sales : 535544
Total seats: 116 , Net Sales : 532393
Total seats: 117 , Net Sales : 528919
Total seats: 118 , Net Sales : 525227
```



```
    return total_penalty
```

```
[ ] for i in range(100, 150):
        sales = 5000*i
        penalty  = calculate_expected_penalty(i)
        netsales = (sales - penalty)
        print("Total seats:",i, ", Net Sales :",round(netsales))
```

```
Total seats: 100 , Net Sales : 500000
Total seats: 101 , Net Sales : 505000
Total seats: 102 , Net Sales : 509996
Total seats: 103 , Net Sales : 514979
Total seats: 104 , Net Sales : 519913
Total seats: 105 , Net Sales : 524725
Total seats: 106 , Net Sales : 529288
Total seats: 107 , Net Sales : 533425
Total seats: 108 , Net Sales : 536929
Total seats: 109 , Net Sales : 539603
Total seats: 110 , Net Sales : 541302
Total seats: 111 , Net Sales : 541959
Total seats: 112 , Net Sales : 541595
Total seats: 113 , Net Sales : 540305
Total seats: 114 , Net Sales : 538233
Total seats: 115 , Net Sales : 535544
Total seats: 116 , Net Sales : 532393
Total seats: 117 , Net Sales : 528919
Total seats: 118 , Net Sales : 525227
```



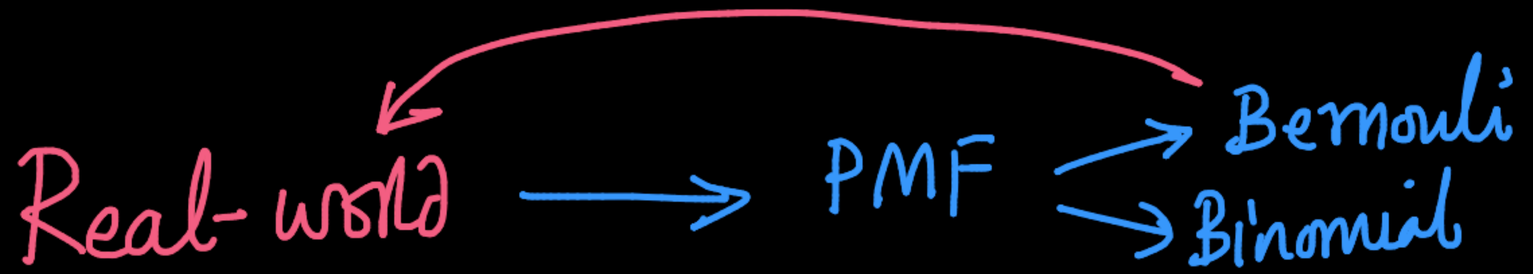
Real-world:

Simulation ⟶ Seasonality in mind

↳ { Sat Morning $p = 0.95$

Diwali ⟶ $p = 0.999$

Weather (Rainy) ⟶ ∴



CDF

PDF

code + Simulation

$X \sim Bin(n, p)$

Q

A: out of 100 people ; count # people w/ covid

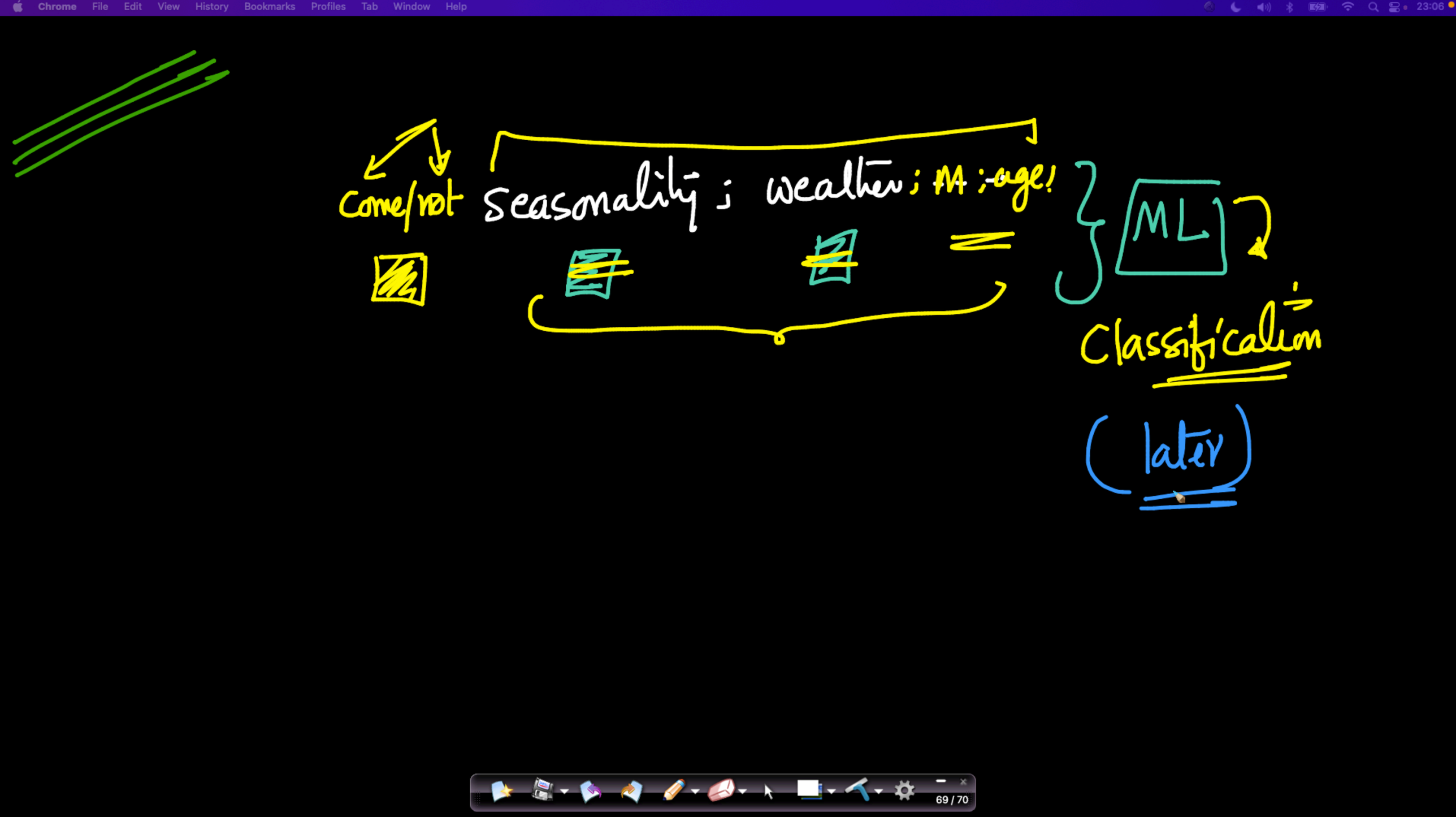
$p = 0.05$

→ discrete
not binary

$A \sim \text{Binomial}(n=100, p=0.5)$

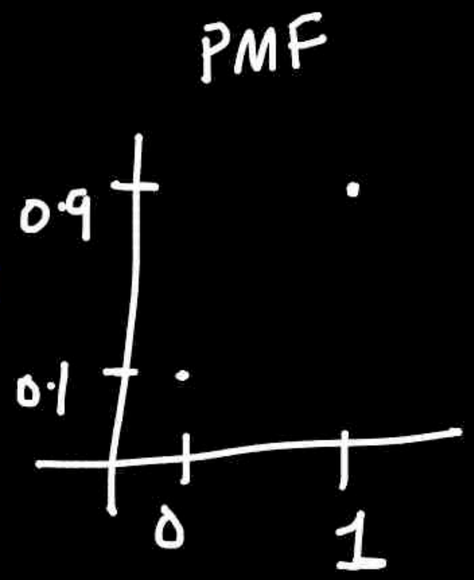
$P(A = 20)$ (k) → # success

pmf


come/not
seasonality ; weather ; MA ; age!
ML
Classification
later

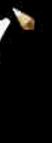
Bernoulli r.v

$X \sim \text{Bernoulli}(p = 0.9)$



$X \sim$ Binomial r.v

PMF & CDF (code)



```
#PMF and CDF of Binomial r.v
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

```
[ ] showsup_probability = 0.9
```

```
[ ] # create 1000 points from a X ~ Binomial(n=110,p=0.9)
    showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

    showsup_data = showsup_distribution.rvs(1000)
```

```
[ ] type(showsup_distribution)
```

```
    scipy.stats._distn_infrastructure.rv_frozen
```

```
[ ] showsup_data[:100]
```

```
#PMF and CDF of Binomial r.v
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

$X \sim Bin(n=110, p=0.9)$

```
[ ] showsup_probability = 0.9
```

```
[ ] # create 1000 points from a X ~ Binomial(n=110,p=0.9)
    showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

    showsup_data = showsup_distribution.rvs(1000)
```

```
[ ] type(showsup_distribution)
```

```
    scipy.stats._distn_infrastructure.rv_frozen
```

```
[ ] showsup_data[:100]
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

```
showsup_probability = 0.9
```

```
# create 1000 points from a X ~ Binomial(n=110,p=0.9)
showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

showsup_data = showsup_distribution.rvs(1000)
```

```
type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

```
showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
```

```
[ ] showsup_probability = 0.9
```

```
[ ] # create 1000 points from a X ~ Binomial(n=110,p=0.9)
    showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

    showsup_data = showsup_distribution.rvs(1000)
```

```
[ ] type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

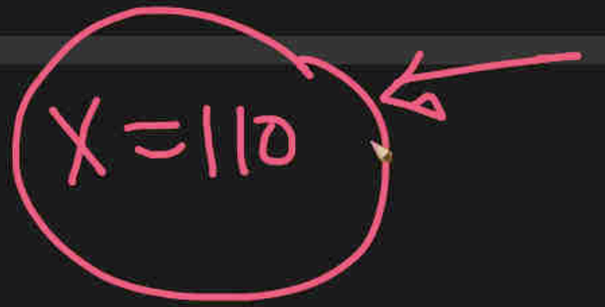
```
[ ] showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
       101, 100, 102, 101, 106,  97, 103, 102, 103,  99, 101, 102, 100,
       101,  93,  98, 102,  96, 100, 100, 102,  90, 100, 103,  98,  98,
       101, 100, 102,  98, 102,  99,  93,  98, 103])
```

+ Code + Text Connect

```
[ ] showsup_probability = 0.9
```

```
[ ] # create 1000 points from a X ~ Binomial(n=110,p=0.9)
    showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

    showsup_data = showsup_distribution.rvs(1000)
```




```
[ ] type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

```
[ ] showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
       101, 100, 102, 101, 106,  97, 103, 102, 103,  99, 101, 102, 100,
       101,  93,  98, 102,  96, 100, 100, 102,  90, 100, 103,  98,  98,
       101, 100, 102,  98, 102,  99,  93,  98, 103])
```

```
[ ]  showsup_probability = 0.9
```

```
[ ]  # create 1000 points from a X ~ Binomial(n=110,p=0.9)
     showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

     showsup_data = showsup_distribution.rvs(1000)
```

Viz PMF & CDF

```
[ ]  type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

```
[ ]  showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
       101, 100, 102, 101, 106,  97, 103, 102, 103,  99, 101, 102, 100,
       101,  93,  98, 102,  96, 100, 100, 102,  90, 100, 103,  98,  98,
       101, 100, 102,  98, 102,  99,  93,  98, 103])
```

```
[ ] showsup_probability = 0.9
```

```
[ ] # create 1000 points from a X ~ Binomial(n=110,p=0.9)
    showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

    showsup_data = showsup_distribution.rvs(1000)
```

```
[ ] type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

```
[ ] showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
       101, 100, 102, 101, 106,  97, 103, 102, 103,  99, 101, 102, 100,
       101,  93,  98, 102,  96, 100, 100, 102,  90, 100, 103,  98,  98,
       101, 100, 102,  98, 102,  99,  93,  98, 103])
```

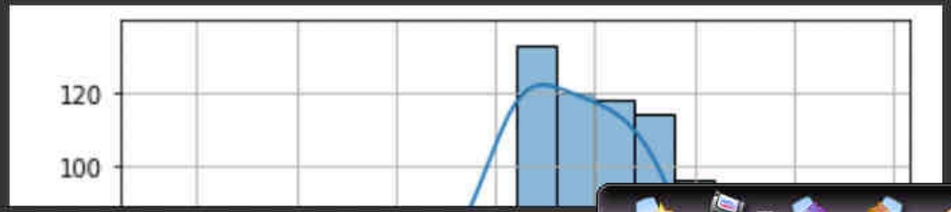
```
type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

```
showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
       101, 100, 102, 101, 106,  97, 103, 102, 103,  99, 101, 102, 100,
       101,  93,  98, 102,  96, 100, 100, 102,  90, 100, 103,  98,  98,
       101, 100, 102,  98, 102,  99,  93,  96, 103])
```

```
#PMF of data
sns.histplot(showsup_data, bins=18, kde=True)
plt.xlabel("Number of passangers showed up")
plt.grid()
plt.show()
```



$P(X=101) = 11\%$

~11%

101

8-8%

102

Chrome File Edit View History Bookmarks Profiles Tab Window Help

InterviewBit Software Services | ProbabilityDisb_1.ipynb - Colab | MoreDistributions.ipynb - Colab

colab.research.google.com/drive/1TetdffpLSis3xG6bRy1M8Zoow8DUrML4

+ Code + Text Connect

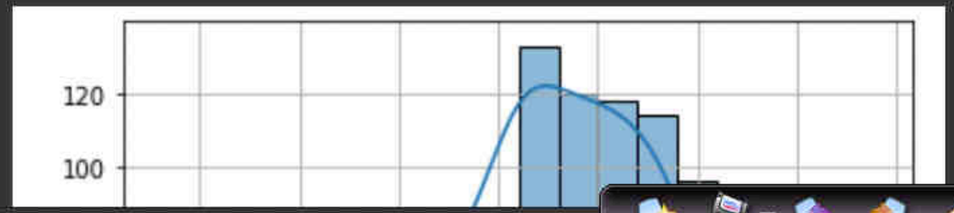
```
[ ] type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

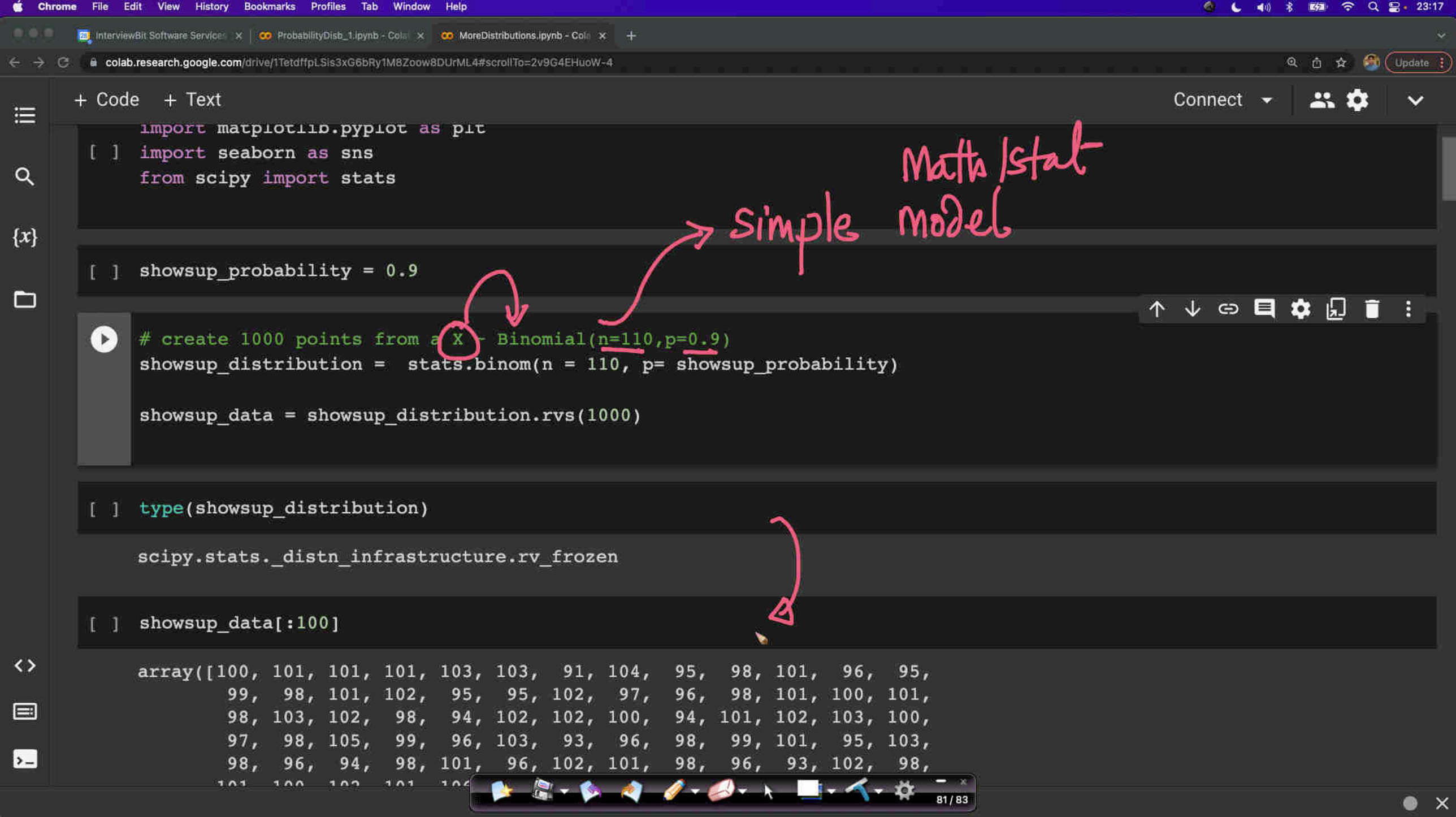
```
[ ] showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
       101, 100, 102, 101, 106,  97, 103, 102, 103,  99, 101, 102, 100,
       101,  93,  98, 102,  96, 100, 100, 102,  90, 100, 103,  98,  98,
       101, 100, 102,  98, 102,  99,  93,  98, 103])
```

```
#PMF of data
sns.histplot(showsup_data, bins=18, kde=True)
plt.xlabel("Number of passangers showed up")
plt.grid()
plt.show()
```



80 / 82


Chrome File Edit View History Bookmarks Profiles Tab Window Help
InterviewBit Software Services
ProbabilityDisb_1.ipynb - Colab
MoreDistributions.ipynb - Colab
colab.research.google.com/drive/1TetdffpLSis3xG6bRy1M8Zoow8DUrML4#scrollTo=2v9G4EHuoW-4
Update
+ Code + Text
Connect
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
Maths Istat
simple model
showsup_probability = 0.9
# create 1000 points from a X ~ Binomial(n=110,p=0.9)
showsup_distribution = stats.binom(n = 110, p= showsup_probability)
showsup_data = showsup_distribution.rvs(1000)
type(showsup_distribution)
scipy.stats._distn_infrastructure.rv_frozen
showsup_data[:100]
array([100, 101, 101, 101, 103, 103, 91, 104, 95, 98, 101, 96, 95,
99, 98, 101, 102, 95, 95, 102, 97, 96, 98, 101, 100, 101,
98, 103, 102, 98, 94, 102, 102, 100, 94, 101, 102, 103, 100,
97, 98, 105, 99, 96, 103, 93, 96, 98, 99, 101, 95, 103,
98, 96, 94, 98, 101, 96, 102, 101, 98, 96, 93, 102, 98,
81 / 83

```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

```
showsup_probability = 0.9
```

```
# create 1000 points from a X ~ Binomial(n=110,p=0.9)
showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

showsup_data = showsup_distribution.rvs(1000)
```

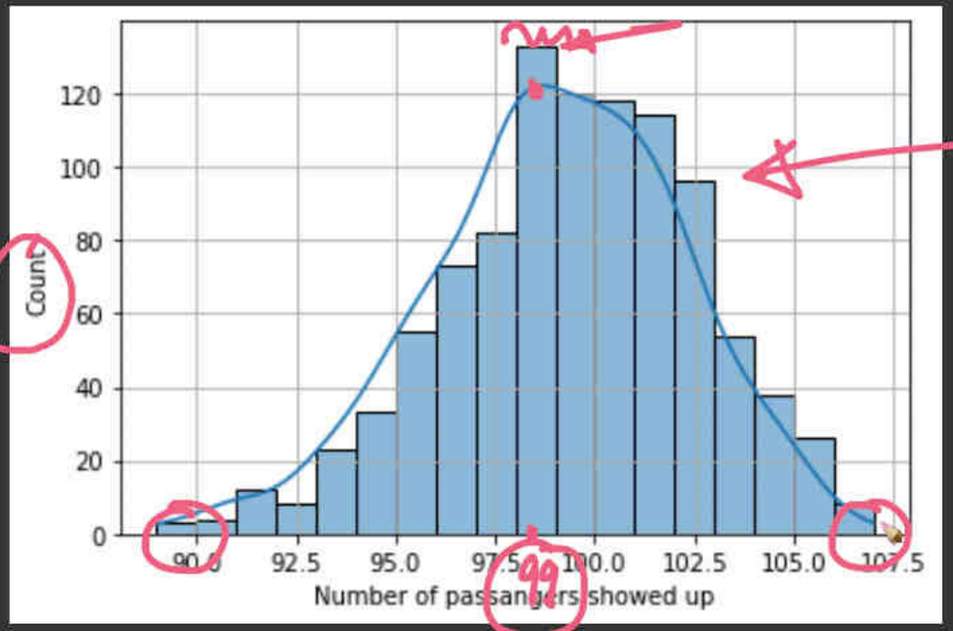
```
type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

```
showsup_data[:100]
```

```
array([100, 101, 101, 101, 103, 103,  91, 104,  95,  98, 101,  96,  95,
        99,  98, 101, 102,  95,  95, 102,  97,  96,  98, 101, 100, 101,
        98, 103, 102,  98,  94, 102, 102, 100,  94, 101, 102, 103, 100,
        97,  98, 105,  99,  96, 103,  93,  96,  98,  99, 101,  95, 103,
        98,  96,  94,  98, 101,  96, 102, 101,  98,  96,  93, 102,  98,
```

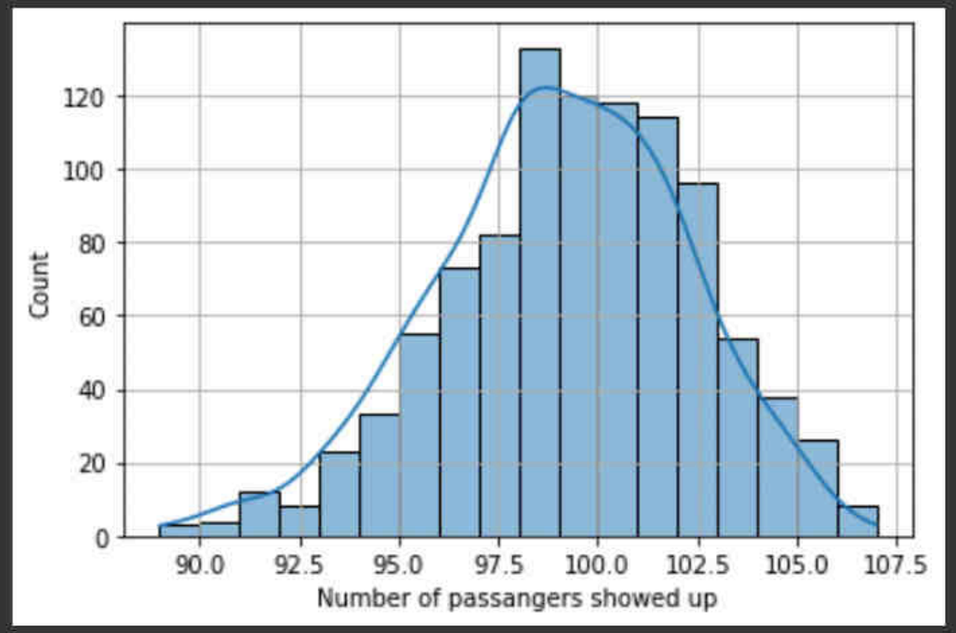
```
[ ] #PMF of data
    sns.histplot(showsup_data, bins=18, kde=True)
    plt.xlabel("Number of passangers showed up")
    plt.grid()
    plt.show()
```

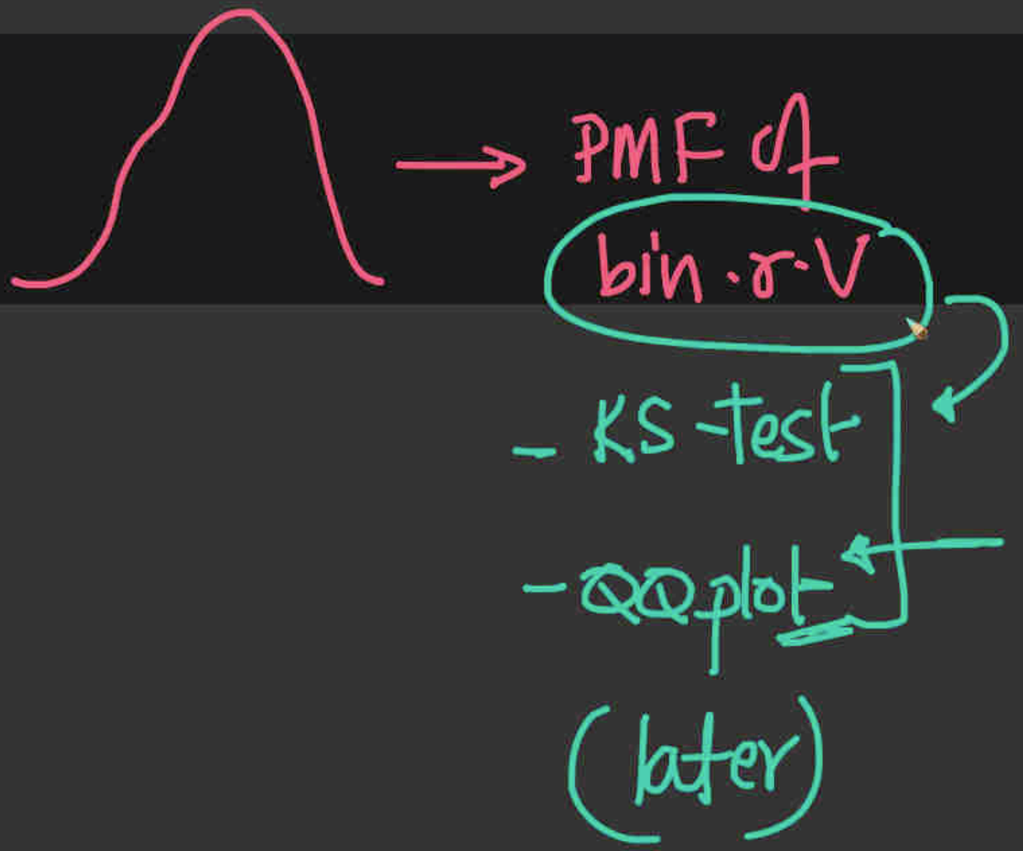


PMF of $X \sim Bin(n=110; p=0.9)$

```
[ ] #CDF:
    x = np.linspace(80, 115)
```

```
[ ] #PMF of data
    sns.histplot(showsup_data, bins=18, kde=True)
    plt.xlabel("Number of passangers showed up")
    plt.grid()
    plt.show()
```



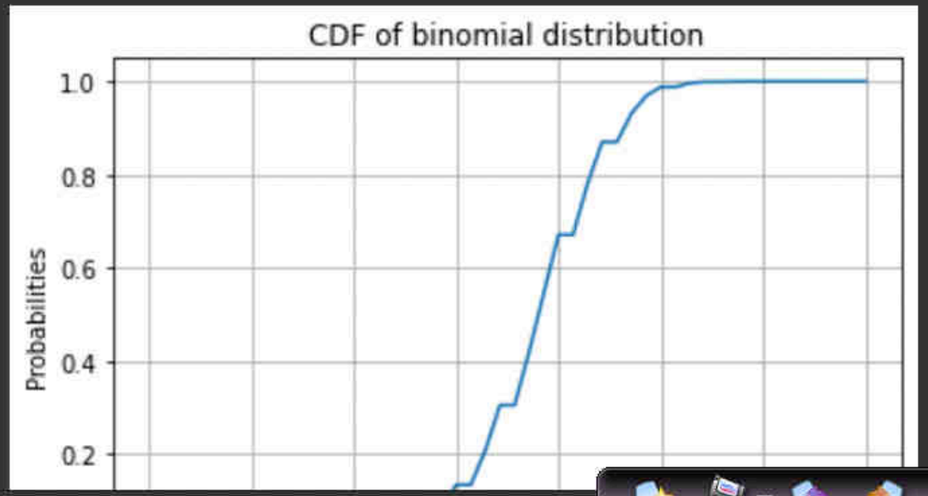



```
[ ] #CDF:
    x = np.linspace(80, 115)
```

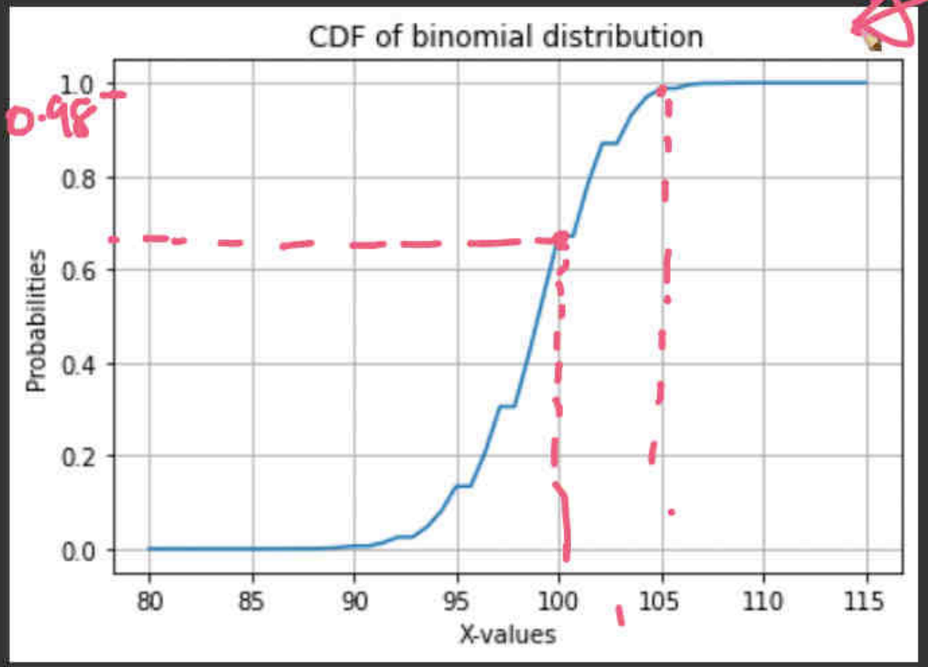
90.0 92.5 95.0 97.5 100.0 102.5 105.0 107.5
Number of passangers showed up

```
#CDF:
x = np.linspace(80, 115)

cdf = showsup_distribution.cdf(x)

plt.plot(x, cdf)
plt.xlabel("X-values")
plt.ylabel("Probabilities")
plt.title("CDF of binomial distribution")
plt.grid()
plt.show()
```

$X \sim Bin(n=110; p=0.9)$



```
cdf = showsup_distribution.cdf(x)

plt.plot(x, cdf)
plt.xlabel("X-values")
plt.ylabel("Probabilities")
plt.title("CDF of binomial distribution")
plt.grid()
plt.show()
```

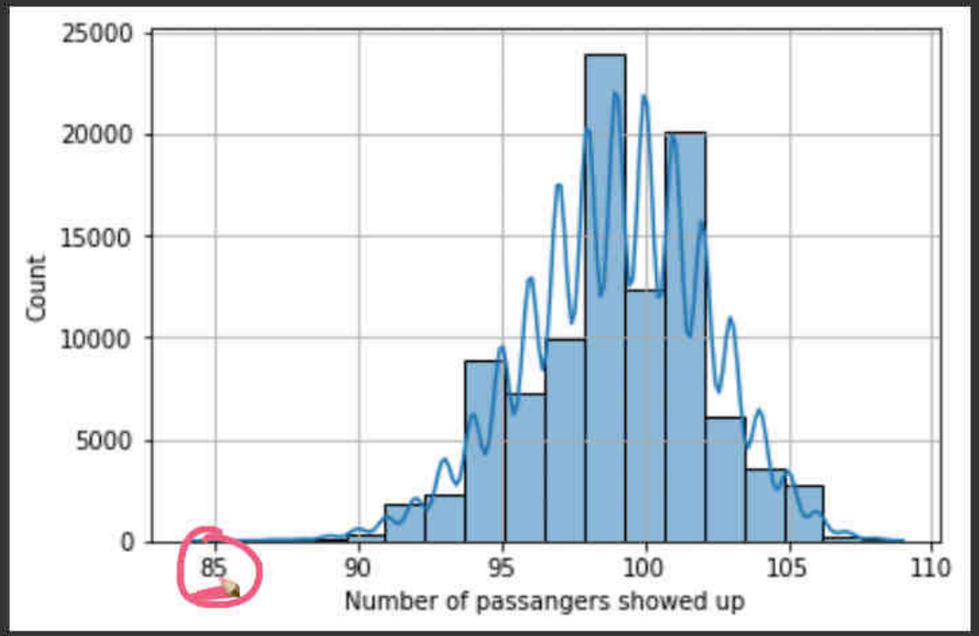


~ 65% prob of 100 or fewer passengers

~ 95% prob of ≤ 105 passengers

```
[ ]  ## T-Shirt sizes problem
```

```
#PMF of data
sns.histplot(showsup_data, bins=18, kde=True)
plt.xlabel("Number of passangers showed up")
plt.grid()
plt.show()
```




```
[ ] #CDF:
    x = np.linspace(80, 115)

    cdf = showsup distribution.cdf
```

[1]
```
#PMF and CDF of Binomial r.v
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

[2]
```
showsup_probability = 0.9
```

[6]
```
# create 1000 points from a X ~ Binomial(n=110,p=0.9)
showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

showsup_data = showsup_distribution.rvs(1000)
```

#passengers

Simplest stat models of real phenomena

[PMF ≈ real-data]

[ ]
```
type(showsup_distribution)
```

```
scipy.stats._distn_infrastructure.rv_frozen
```

[4]
```
showsup_data[:100]
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 23:25

InterviewBit Software Services | ProbabilityDisb_1.ipynb - Colab | MoreDistributions.ipynb - Colab

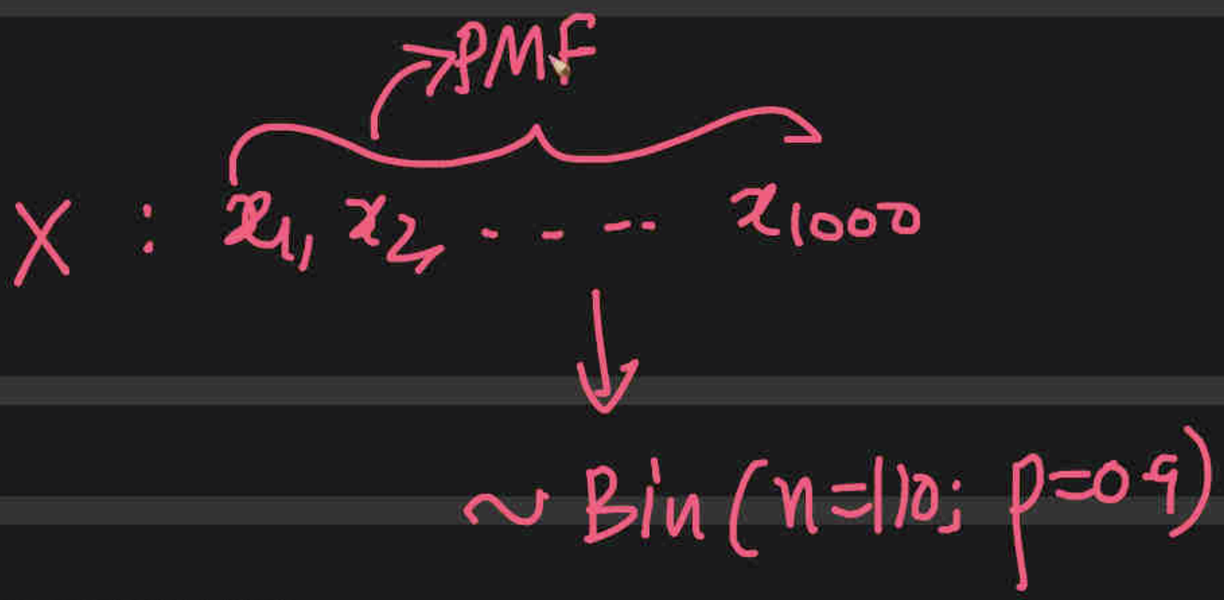
colab.research.google.com/drive/1TetdffpLSis3xG6bRy1M8Zoow8DUrML4#scrollTo=gRLHViDJqxFr

Update

+ Code + Text

RAM Disk

```
[1]  #PMF and CDF of Binomial r.v
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from scipy import stats
```




```
[2]  showsup_probability = 0.9
```

```
[6]  # create 1000 points from a X ~ Binomial(n=110,p=0.9)
     showsup_distribution =  stats.binom(n = 110, p= showsup_probability)

     showsup_data = showsup_distribution.rvs(1000)
```

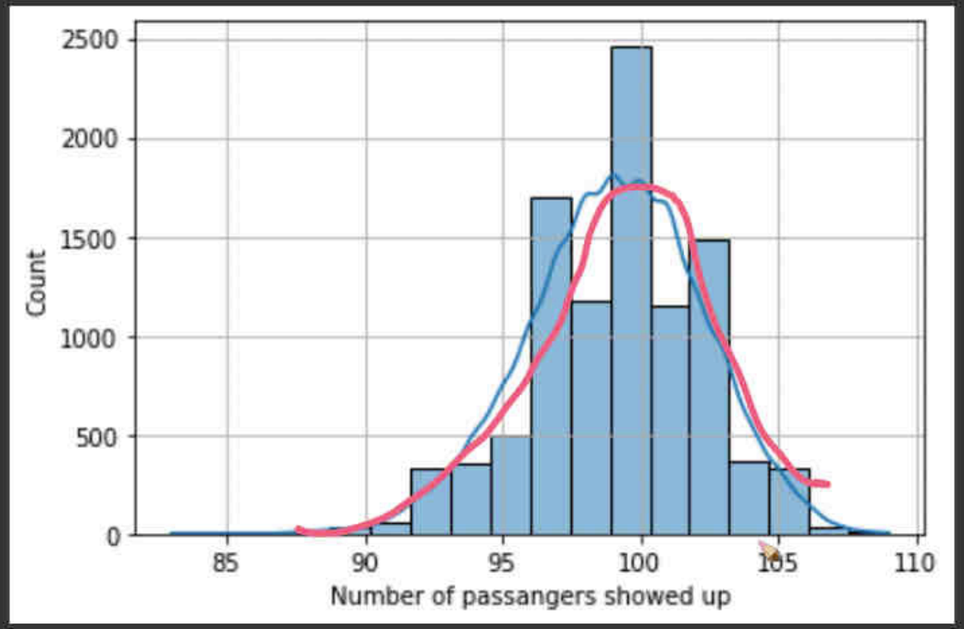
```
[ ]  type(showsup_distribution)
```

```
     scipy.stats._distn_infrastructure.rv_frozen
```

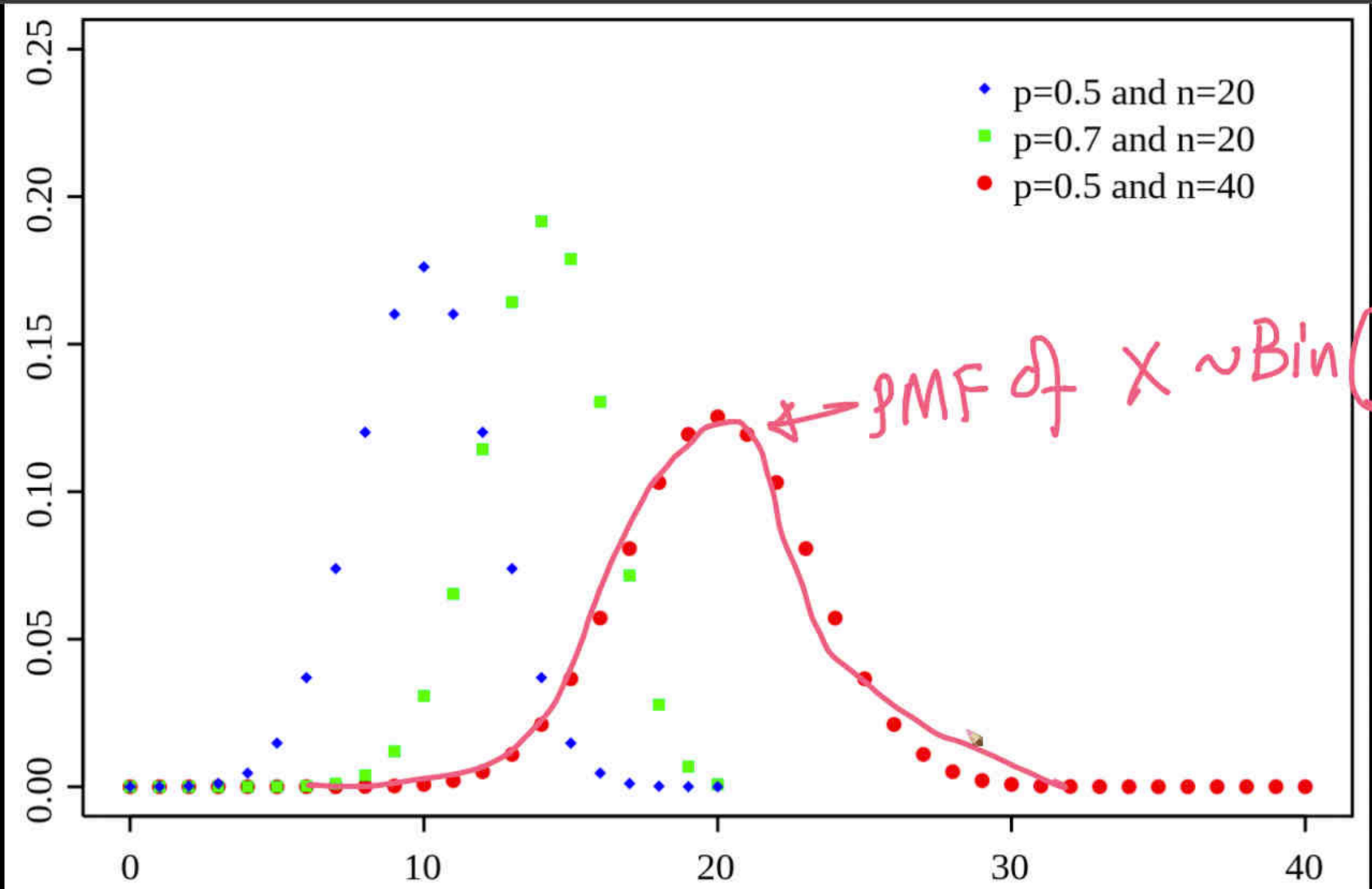
```
[4]  showsup_data[:100]
```

89 / 91

```
#PMF of data
sns.histplot(showsup_data, bins=18, kde=True)
plt.xlabel("Number of passangers showed up")
plt.grid()
plt.show()
```




```
[ ] #CDF:
    x = np.linspace(80, 115)

    cdf = showsup distribution.cdf
```



Probability mass function for the binomial distribution

DS - HR @ Google:

XXS, XS, S, M, L, XL, XXL

Normal / Gaussian dist

PDF or CDF

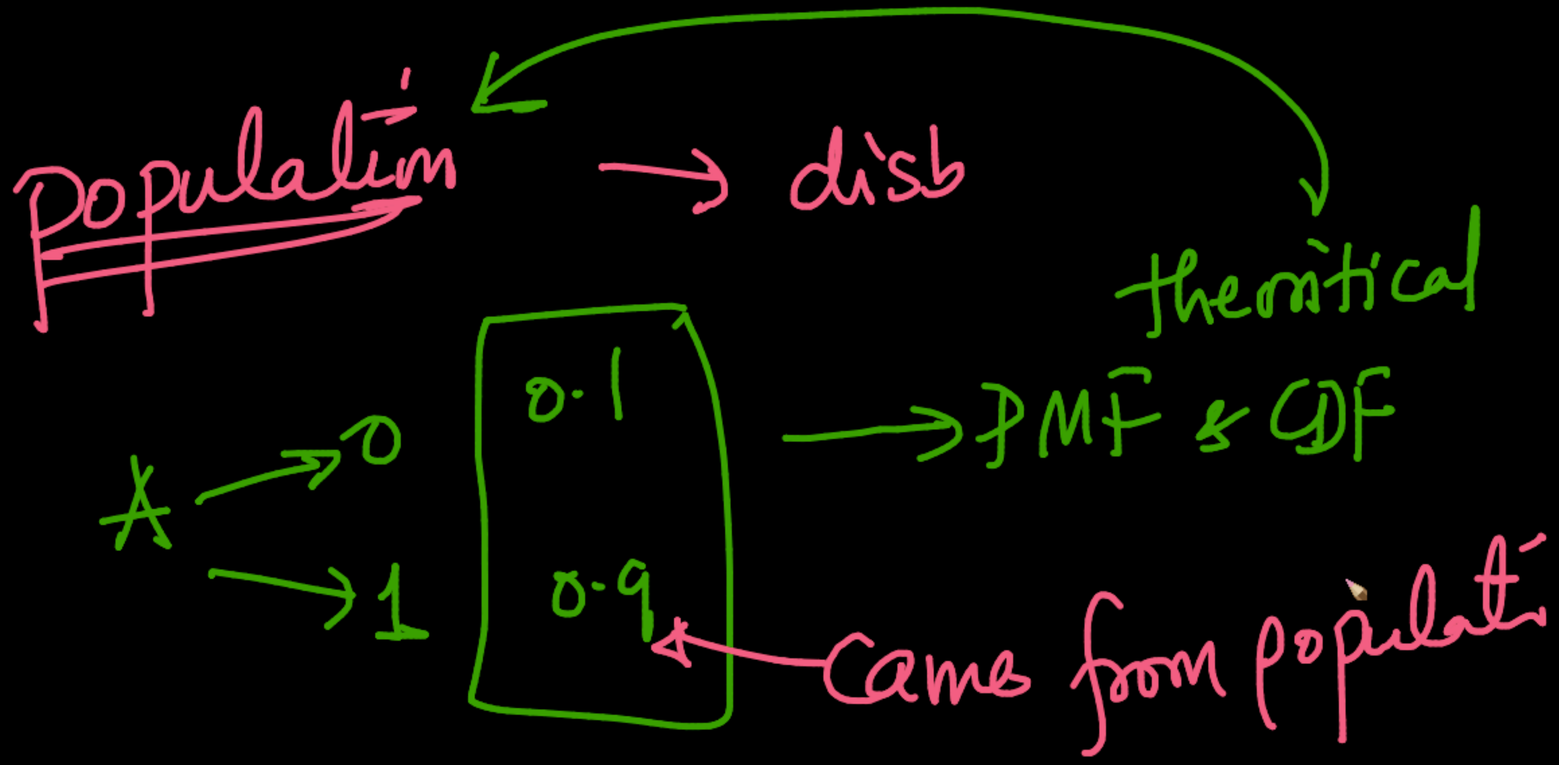
sample or population

Data: sample → seaborn / Matplotlib

PDF or CDF

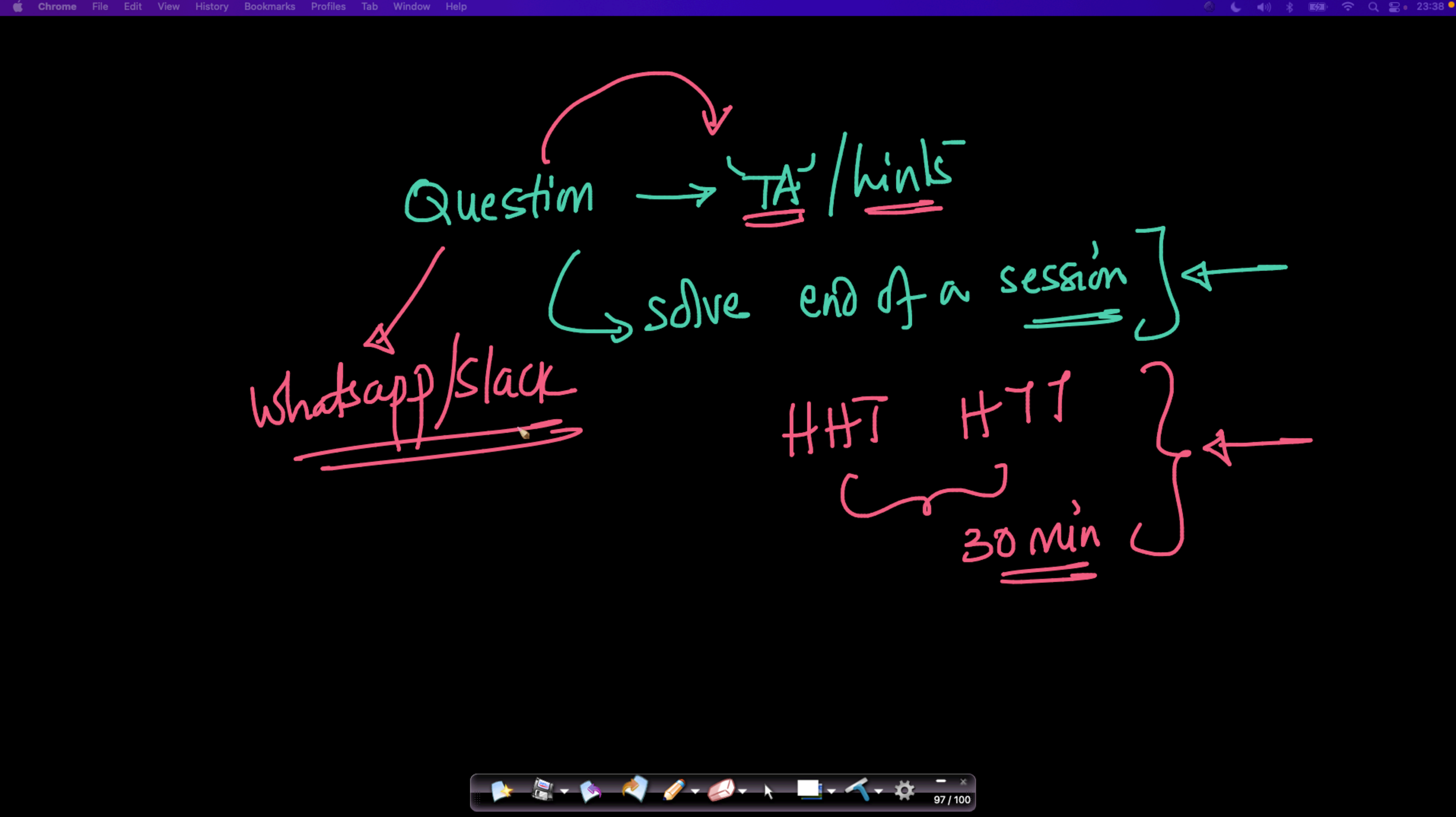
↓

Sample ← empirical PDF or CDF


Populatim
→ distb
theoritical
0·1
A → 0
→ 1
0·9
→ PMF & CDF
came from populati

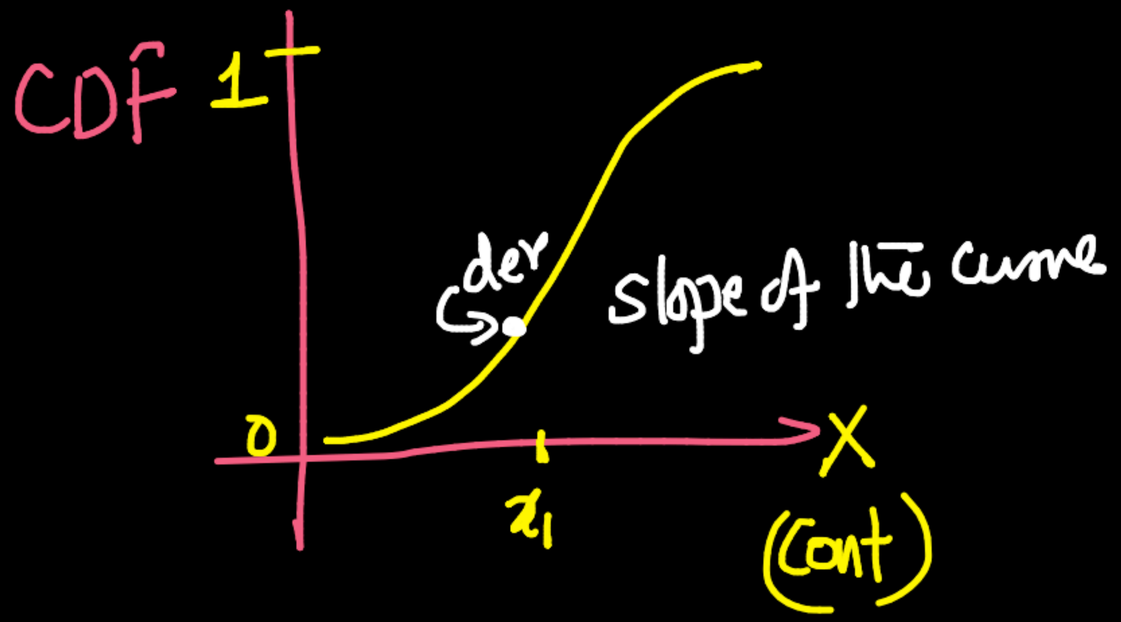
→ Assume assignment

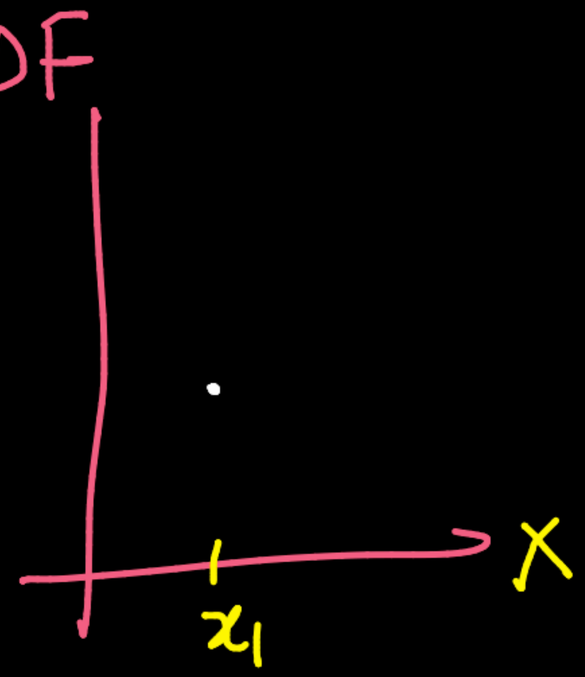
→ Tackle → after next Class...


Question
TA / hints
solve
end of a session
whatsapp/slack
HHT
HTT
30 min

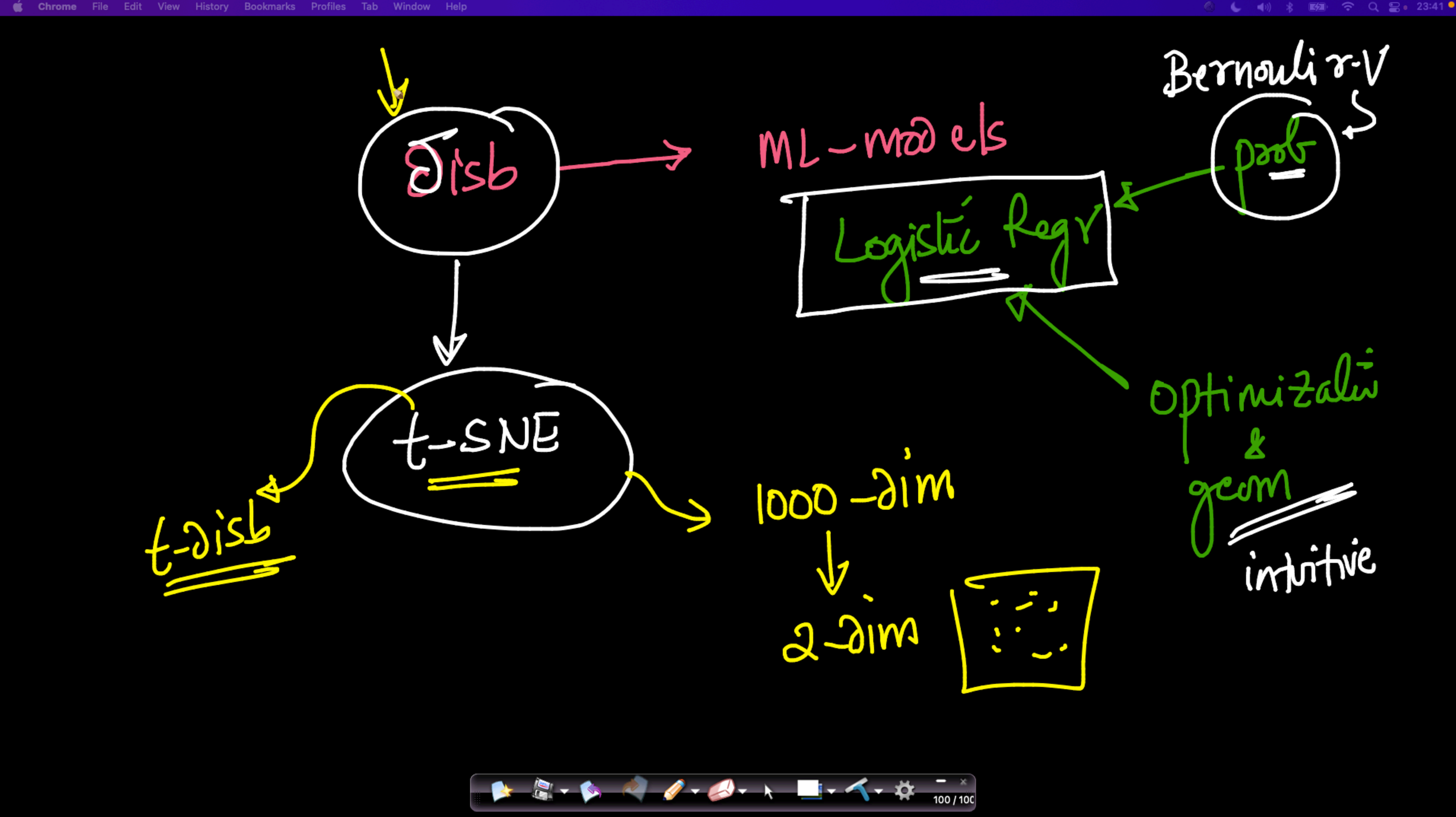
Binomial → Multinomial dist] → older ML techniques

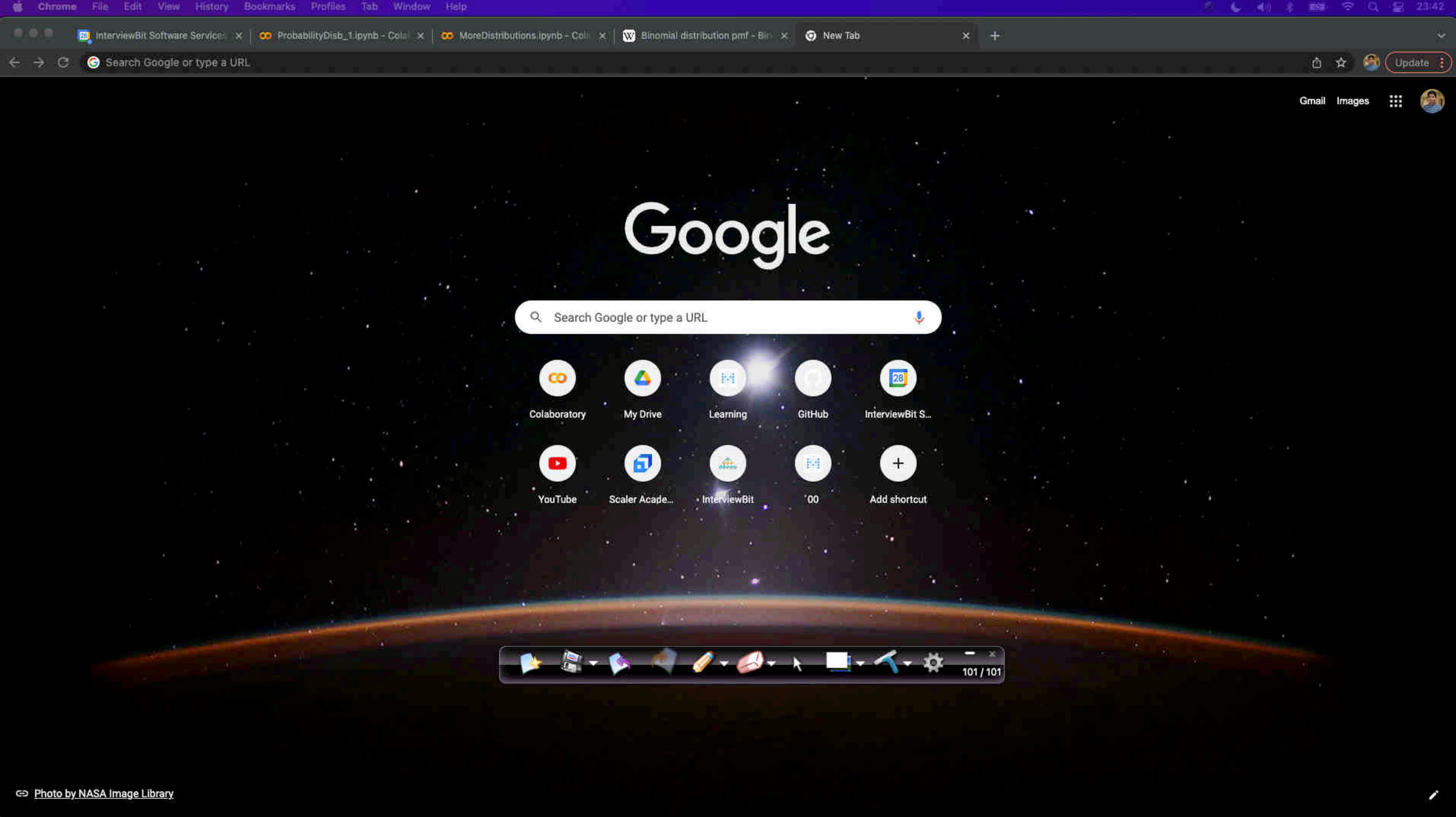
Stats:





{Math for ML [(later) → derivative (differencing)


Chrome File Edit View History Bookmarks Profiles Tab Window Help
23:41
Bernoulli r-V
probs
ML-models
Logistic Regr
optimizaliu
&
geom
intuitive
disb
t-SNE
t-disb
1000-dim
2-dim
100 / 100


Chrome File Edit View History Bookmarks Profiles Tab Window Help
InterviewBit Software Services
ProbabilityDisb_1.ipynb - Colab
MoreDistributions.ipynb - Colab
Binomial distribution pmf - Bin
New Tab
Search Google or type a URL
Update
Gmail
Images
Google
Search Google or type a URL
Colaboratory
My Drive
Learning
GitHub
InterviewBit S...
YouTube
Scaler Acade...
InterviewBit
00
Add shortcut
101 / 101
Photo by NASA Image Library