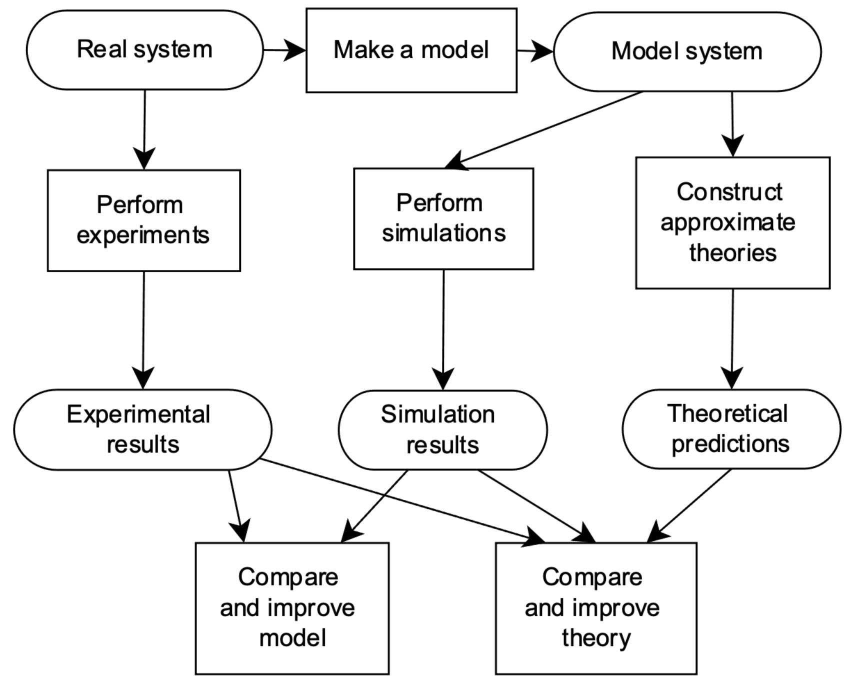
# Geospatial Data Science
# Content Block II: *Techniques*
# Lab 6
# Generate & Analyze Spatial RVs

Austin J. Brockmeier, Ph.D.

Wednesday, March 15th, 2023

# Probability and Spatial Statistics for Geospatial Systems and Data

1. Statistical analysis
2. Generate data from random variables and processes through **computer simulation** (Lab 6&7)
3. Analyze spatial patterns in data



# Outline

- Generating random numbers
- Counting: binning/histograms
- Basic statistics
- Bayes rule: joint and conditional
- Spatial correlation

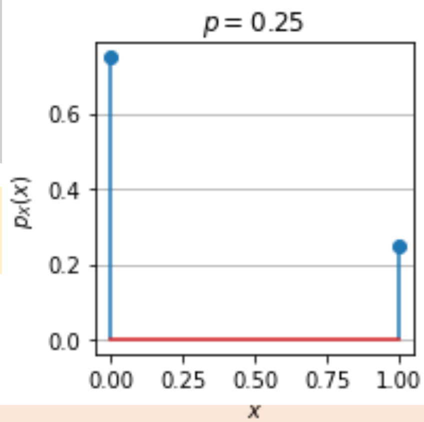
# Coin data

Binary data: Flip a fair coin, create a **Bernoulli** random variable $X$

$$X(\omega) = \begin{cases} +1, \omega = \text{Heads} \\ 0, \omega = \text{Tails} \end{cases}$$

$$P_X(1) = \Pr(X = 1) = p$$
$$P_X(0) = \Pr(X = 0) = 1 - p$$

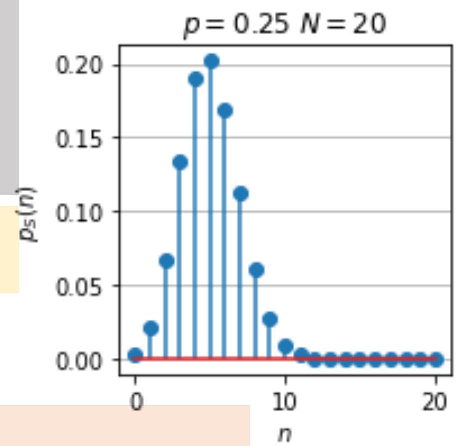
$X \in \{0,1\}$


Count data:
Flip a fair coin $N$ times, count of heads is **binomial** random variable $S$ with probability mass function

$$P_S(n) = \binom{n}{N} p^N (1-p)^{n-N}$$

$S \in \{0,1,\ldots,N\}$



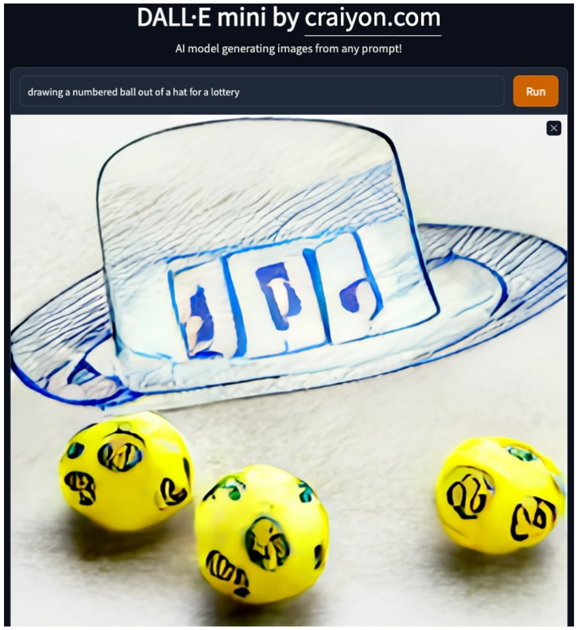

<u>Analysis Questions</u>

What is the proportion of heads if we observe 10, 100, or 1000 coin flips?
What is mean value?
What are the confidence intervals for the proportion of heads?

# Random number generation

Pseudo random number generators (deterministic if seed is known):
https://en.wikipedia.org/wiki/Pseudorandom_number_generator
Generating normal RVs from uniform:
https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform

```
import numpy as np
import scipy.stats as st


rng = np.random.default_rng(seed=0)
```



# Coin data

[scipy.stats.bernoulli]

```
X = st.bernoulli.rvs(p, size=n_trials)
print(X)
print("{} flips with {} heads ({}%)".
      format(n_trials,np.sum(X==1),np.mean(X==1)))
```

```
1000 flips with 519 heads (0.519%)
```

```
[1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1
 1 0 0 0 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 1
 0 1 0 1 1 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 1 0 0 0 1 0
 0 1 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0
 0 1 0 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 0 0 0 0 1 0
 0 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1
 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1
 0 1 1 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 1 1 1 0 1 0 1
 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0
 1 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 0 0 0 0 0 1
 1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0
 1 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0
 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1
 1 1 1 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0 1
 1 1 0 1 1 1 0 1 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1
 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0
 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 0 0 0 1
 1 0 1 0 1 1 0 1 0 0 1 1 0 1 0 1 1 0 1 0 1 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1
 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0
 1 1 1 0 0 0 1 0 0 0 1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0
 0 0 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0
 1 1 0 0 1 0 1 0 1 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 0 1
 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0
 0 0 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 0 1 1
 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 1 1 0 0 0 1 1 1 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1
 0]
```

# Coin data


## numpy.random.Generator.binomial

```
rng = np.random.default_rng(seed=0)
N = 1
n_trials, p = 1000, 0.5
X = rng.binomial(N, p, n_trials)
print("{} flips with {} heads ({}%)".
       format(n_trials,np.sum(X==1),np.mean(X==1)))
```

```
1000 flips with 527 heads (0.527%)
```

# Counting in NumPy

numpy.sum numpy.newaxis

```
p = 0.25
n_trials = 4
X = rng.binomial(1, p, n_trials)
L=X[:,np.newaxis]==[0,1]
counts = np.sum(L,axis=0)
```

X

```
[0 0 0 1]
```

→

X[:,np.newaxis]

```
[[0]
 [0]
 [0]
 [1]]
```

X[:,np.newaxis]==[0,1]

```
[[0]
 [0]
 [0]
 [1]]
```

==

```
[0 1]
```

→

L

```
[[True False]
 [True False]
 [True False]
 [False True]]
```

axis=0

np.sum(L, axis=0)

→

counts

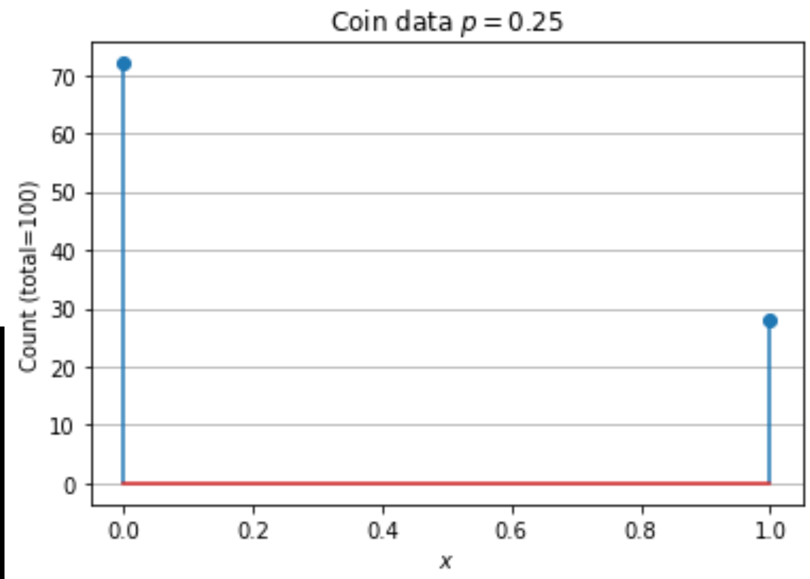
```
[3 1]
```

# Coin data

## matplotlib.stem

```
p = 0.25
n_trials = 100
X = rng.binomial(1, p, n_trials)
print(X)
plt.figure()
plt.stem([0,1],np.sum(X[:,np.newaxis]==[0,1],axis=0))
plt.ylabel("Count (total={})".format(n_trials))
plt.xlabel(r'$x$')
plt.grid(axis='y')
plt.title('Histogram {}{}'.format(r"$p=$",p))
```

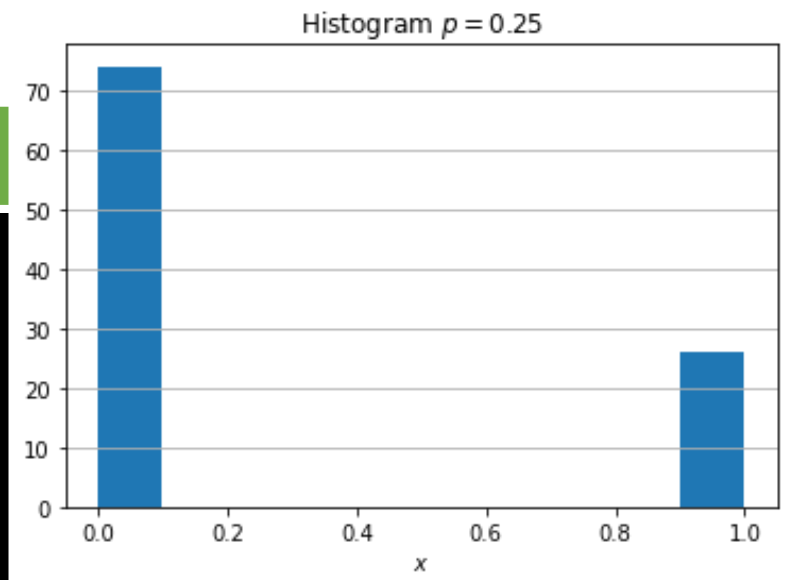
```
[0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 1]
```



# Coin data

numpy.histogram or matplotlib.hist

```
p = 0.25
n_trials = 100
X = rng.binomial(1, p, n_trials)
print(X)
plt.figure()

plt.hist(X)
plt.xlabel(r'$x$')
plt.grid(axis='y')
plt.title('Histogram {}{}'.format(r"$p=$",p))
```

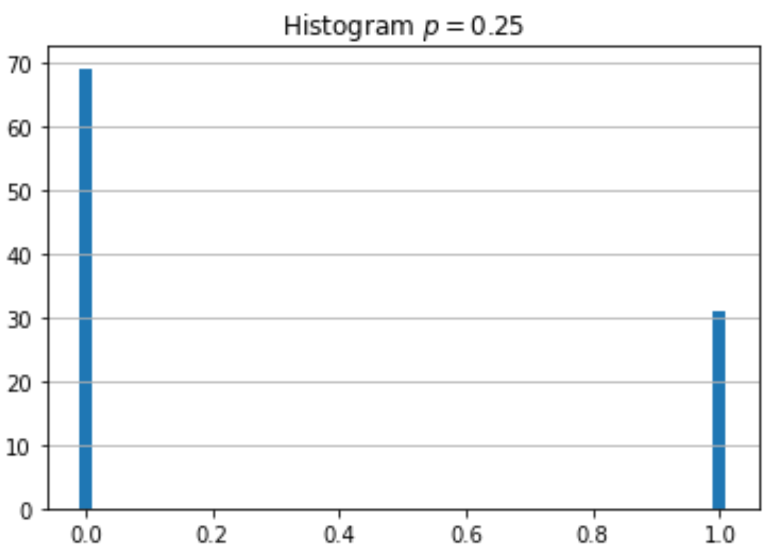



**Not clear what values X takes.... 0.1? 0.9**

# Coin data

## matplotlib.hist

```
plt.figure()
plt.hist(X,bins=[-0.01,0.01,0.99,1.01])
plt.xlabel(r'$x$')
plt.grid(axis='y')
plt.title('Histogram {}{}'.format(r"$p=$",p))
```

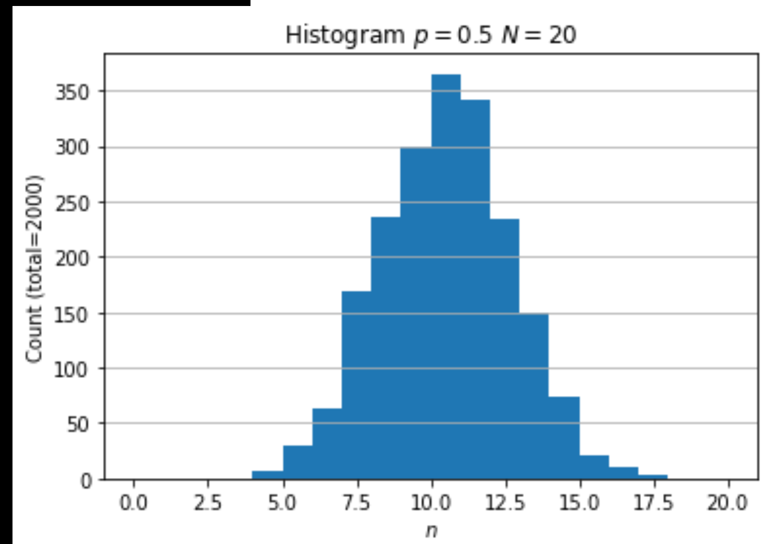



**Notes:** bins
All but the last (righthand-most) bin is half-open.
In other words, if *bins* is:
**[1, 2, 3, 4]**
then the first bin is **[1, 2)** (including 1, but excluding 2) and the second **[2, 3)**. The last bin, however, is **[3, 4]**, which *includes* 4.

# Coin data

numpy.histogram or matplotlib.hist

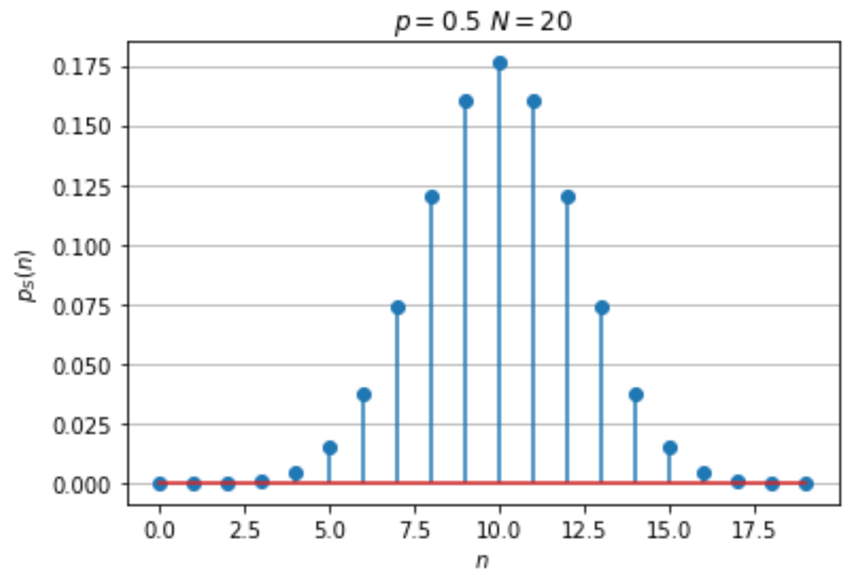
```
p, N = 0.5,20
S = rng.binomial(N, p, n_trials)
n_vals = np.linspace(0,N,N+1)

plt.figure()
plt.hist(S, bins=n_vals)
plt.ylabel("Count
(total={})".format(n_trials))
plt.xlabel(r'$n$')
plt.grid(axis='y')
plt.title('Histogram {}{} {}{}’.
          format(r"$p=$",p,r"$N=$",N))
```



# Coin data

scipy.stats.binom

```
plt.figure()
n_vals = np.arange(0,N)
p_S = st.binom.pmf(n_vals, N, p)
ax = plt.stem(n_vals,p_S)
plt.xlabel(r'$n$')
plt.ylabel(r"$p_S(n)$")
plt.grid(axis='y')
plt.title('{}{} {}{}'.
          format(r"$p=$",p,r"$N=$",N))
plt.show()
```

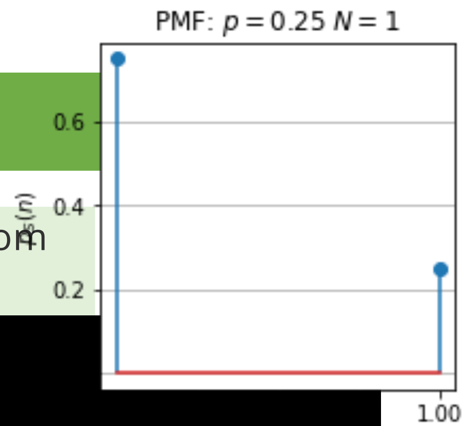


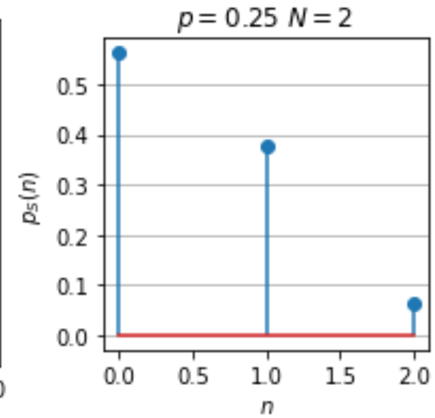
# Coin data

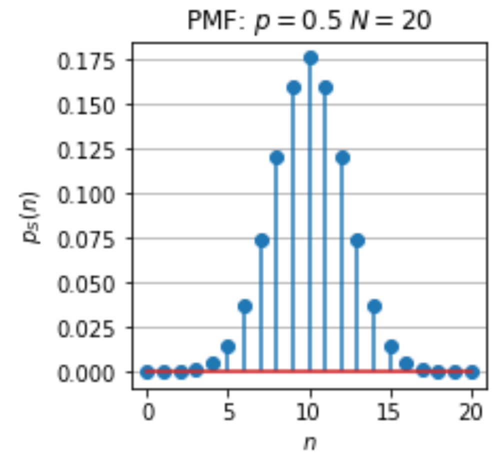
## numpy.convolve

The probability mass function (pmf) of the sum of two random variables is equal to the convolution of their pmfs

```python
def binomial_pmf(p,N):
  pmf = np.array((1-p,p))
  pmf_binomial = pmf
  for i in range(N-1):
    pmf = np.convolve(pmf,pmf_binomial)
  return np.linspace(0,N,N+1), pmf

N, p = 2, 0.25
plt.figure(figsize=(3,3))
n_vals,p_N = binomial_pmf(p,N)
ax = plt.stem(n_vals,p_N)
plt.xlabel(r'$n$')
plt.ylabel(r"$p_S(n)$")
plt.grid(axis='y')
plt.title('{}{} {}{}'.format(r"$p=$",p,r"$N=$",N))
```

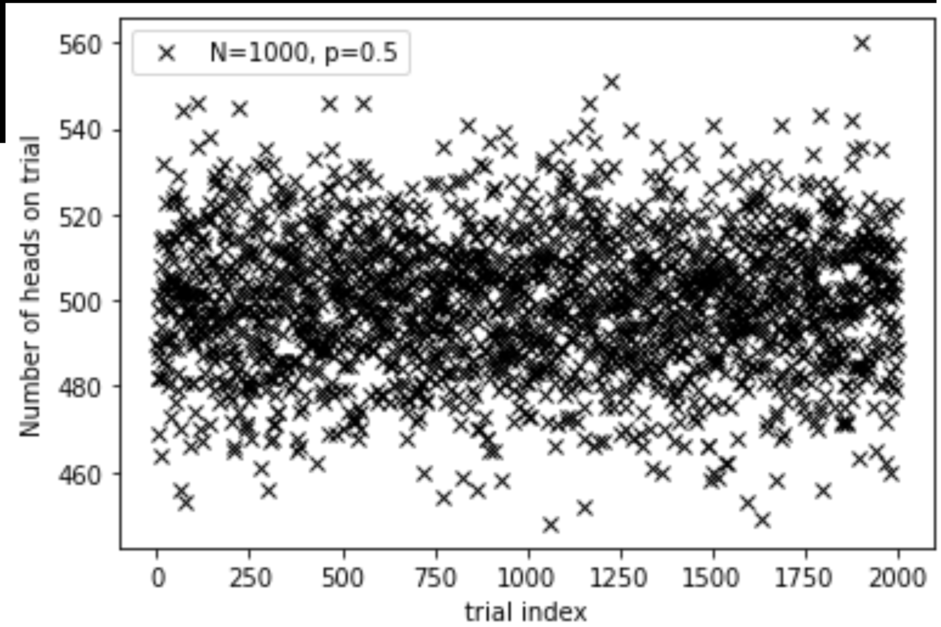


# Coin data

```
N, n_trials = 1000, 2000
S = rng.binomial(N, p, n_trials)
plt.plot(S,linestyle='None',marker='x',color='k',
          label="N={}, p={}".format(N,p))
plt.ylabel('Number of heads on trial')
plt.xlabel('trial index')
plt.legend()
```

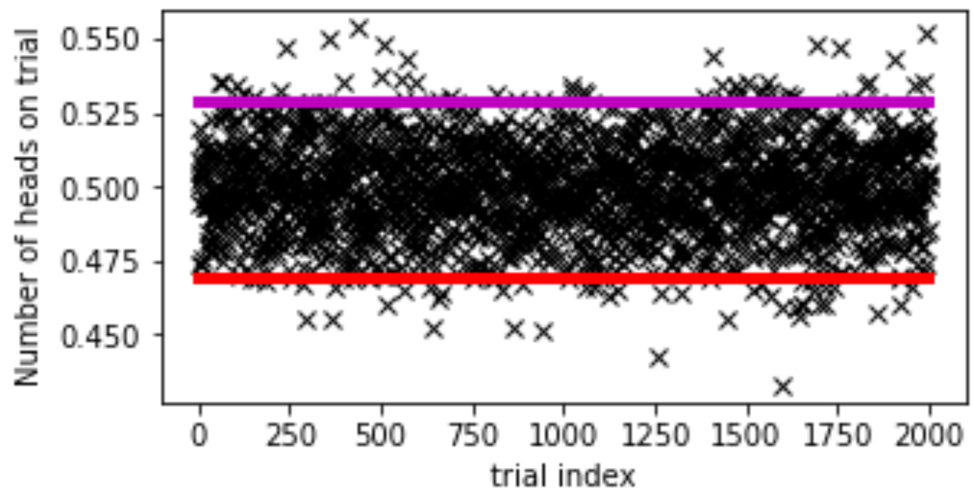


# Coin data

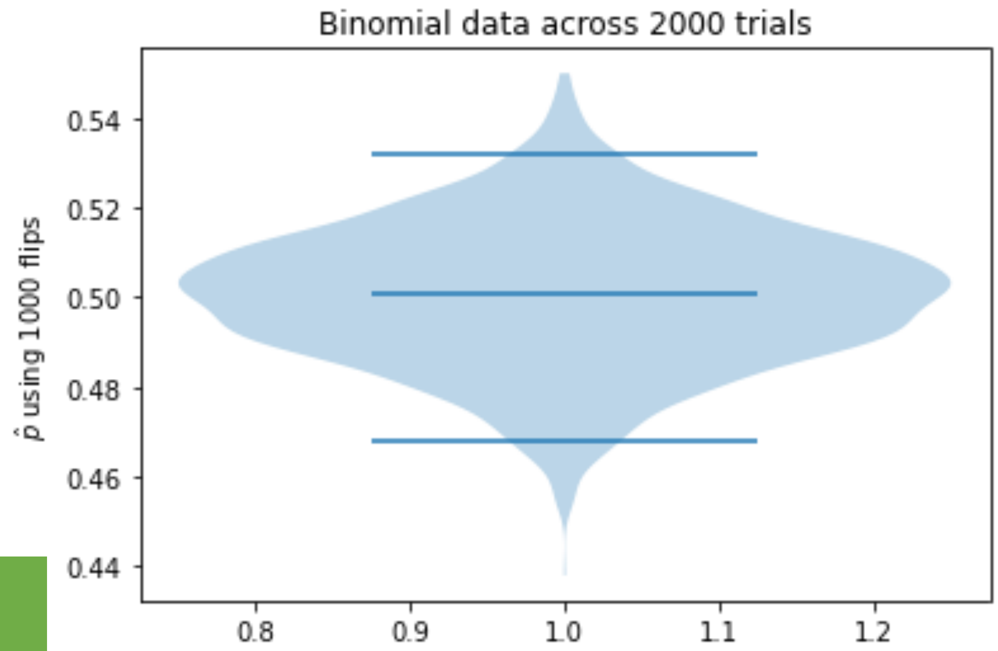
## numpy.quantile

```
quantiles95 =np.array((0.025,0.975))
conf_interval = np.quantile(S/N,quantiles95)
print("95% confidence interval on prob of heads is
{}".format(conf_interval))
plt.plot(S/N,linestyle='None',marker='x',color='k')
plt.plot([-1,n_trials],conf_interval[0]*np.ones(2),color='r',linewidth=4)
plt.plot([-1,n_trials],conf_interval[1]*np.ones(2),color='m',linewidth=4)
plt.xlabel('trial index')
plt.ylabel('Number of heads on trial')
```

```
95% confidence interval on prob
of heads is [0.469 0.529025]
```



# Coin data



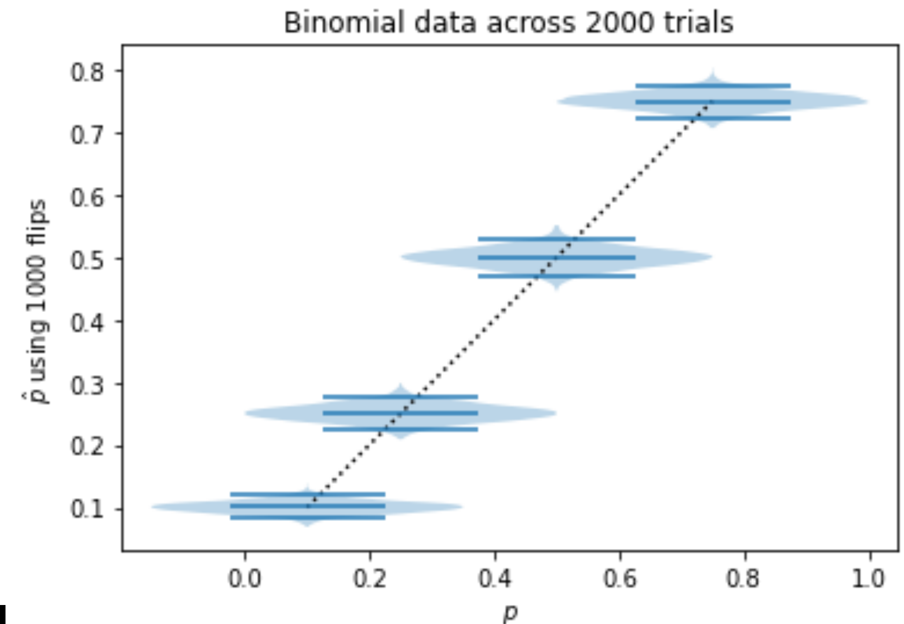

The shape of the violin is a density estimate!

[matplotlib.pyplot.violinplot]()

```
p = 0.5
p_hat = np.random.binomial(n_flips, p, N)/N
plt.violinplot(p_hat,showextrema=False, showmeans=True,
                 quantiles=quantiles95)
plt.xlabel('$p$')
plt.ylabel('{} using {} flips'.format(r"$\hat{p}$",n_flips) )
plt.title('Binomial data across {} trials'.format(n_trials))
```

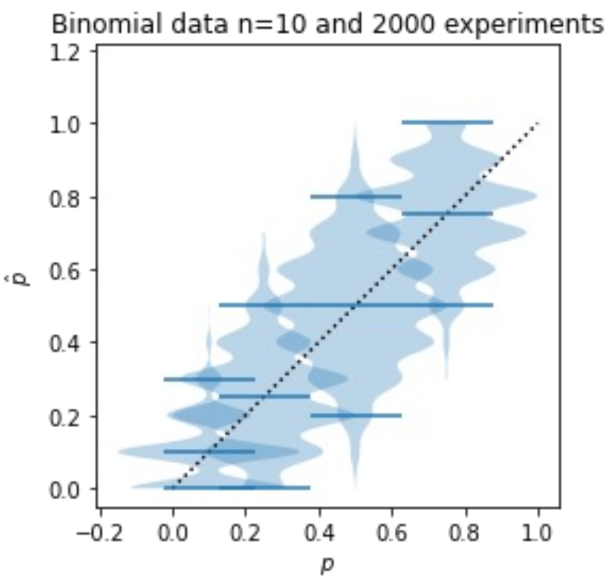
# Coin data

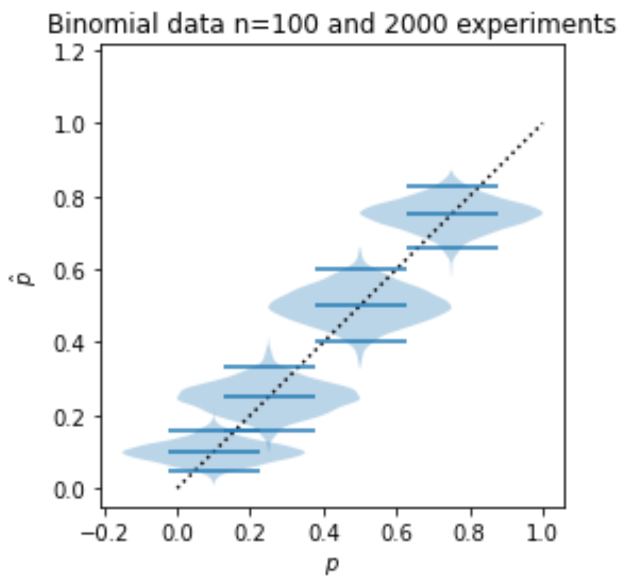
[matplotlib.pyplot.violinplot](matplotlib.pyplot.violinplot)

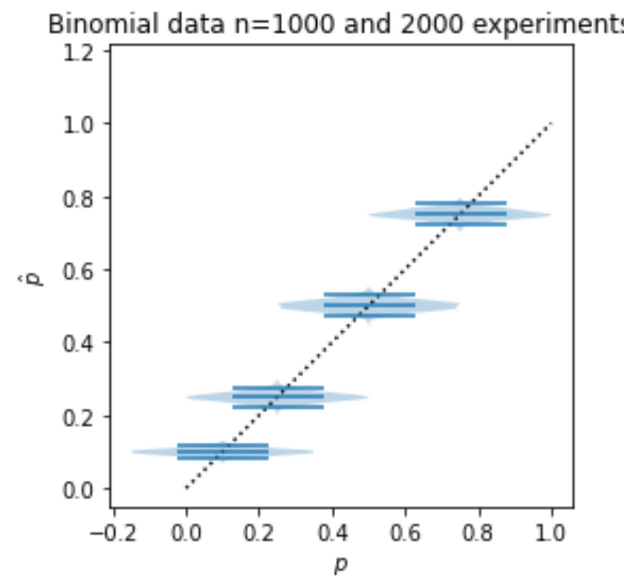



```
p_list = [0.1, 0.25, 0.5, 0.75]
p_hat_across_p = [rng.binomial(N, p, n_trials)/N for p in p_list]
plt.plot([0,1],[0,1],'k',linestyle=":")
plt.violinplot(p_hat_across_p, positions=p_list, showextrema=False,
showmeans=True, quantiles=np.repeat(quantiles95[:,np.newaxis],
                     len(p_list),axis=1))
plt.axis('square')
plt.xlabel('$p$')
plt.ylabel('{} using {} flips'.format(r"$\hat{p}$",N) )
plt.title('Binomial data across {} trials'.format(n_trials))
```

# Coin data


Binomial data n=10 and 2000 experiments
$\hat{p}$
$p$


Binomial data n=100 and 2000 experiments
$\hat{p}$
$p$


Binomial data n=1000 and 2000 experiments
$\hat{p}$
$p$

# Estimators: bias, variance, mean squared error

- **Expected bias of a parameter estimate:**

$$\mathbb{E}[\widehat{\theta(X)}] - \theta^*$$

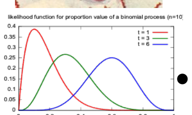
**Variance of a parameter estimate:**

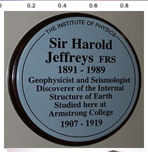
$$\mathbb{E}\left[\left(\widehat{\theta(X)} - \mathbb{E}[\widehat{\theta(X)}]\right)^2\right]$$

**RMSE of a parameter estimate:**

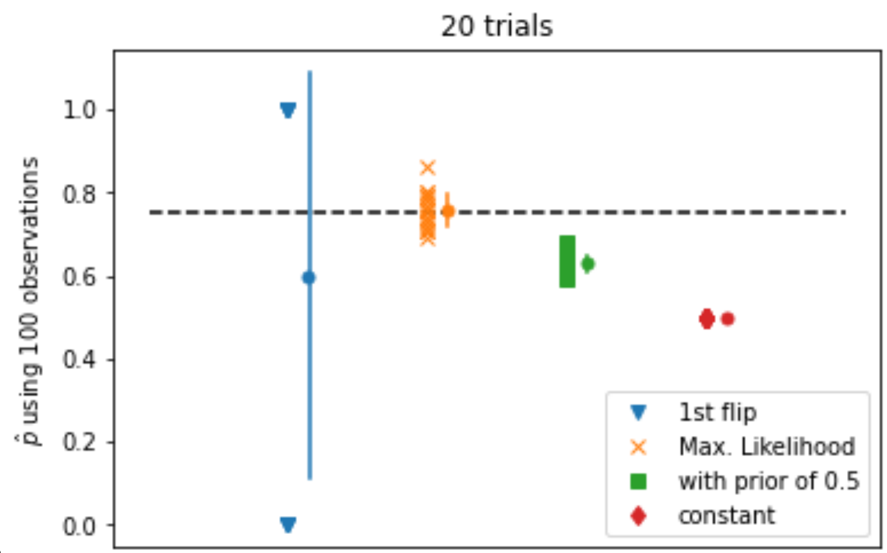
$$\sqrt{\mathbb{E}\left[\left(\widehat{\theta(X)} - \theta^*\right)^2\right]}$$

***Consider an unfair/loaded coin. The goal is to estimate the proportion of tails.***






- **Estimator Silly:** Use first flip to estimate probability of tail.
- **Estimator ML:** Observed proportion, this maximizes the likelihood (ML) of the data.
- **Estimator Prior:** Weighted average of the observed proportion (weight = $N$) and 0.5 (weight = 100).
- **Estimator Ignorant:** Say the probability of a tail is 0.5 no matter what.



# Estimators: bias, variance, mean squared error

**Expected bias of a parameter estimate:**

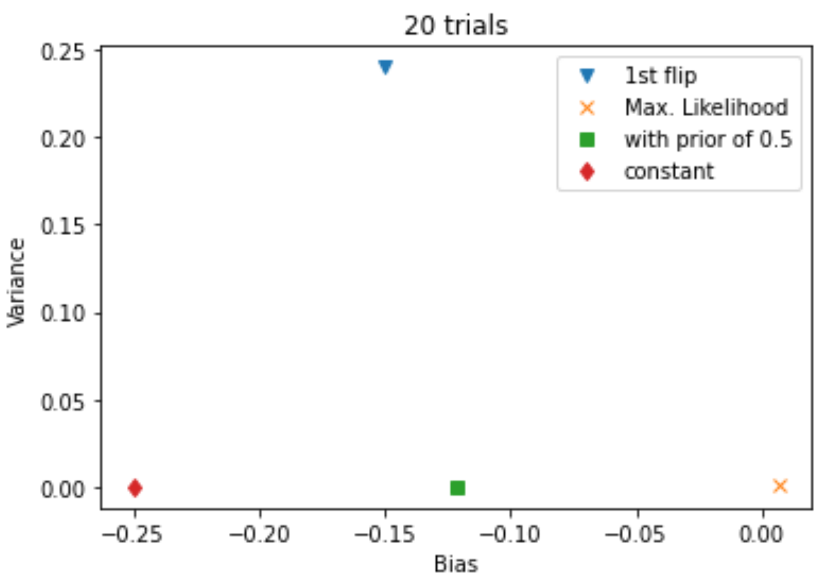
$$\mathbb{E}\left[\widehat{\theta(X)}\right] - \theta^*$$

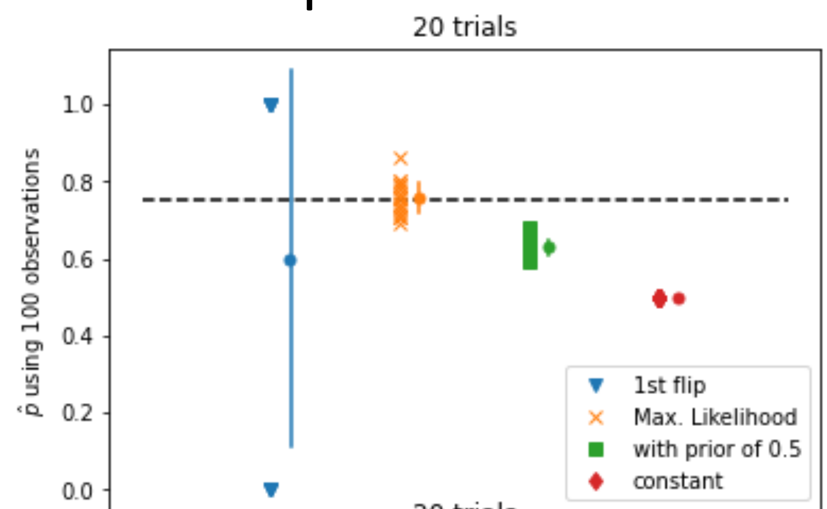
**Variance of a parameter estimate:**

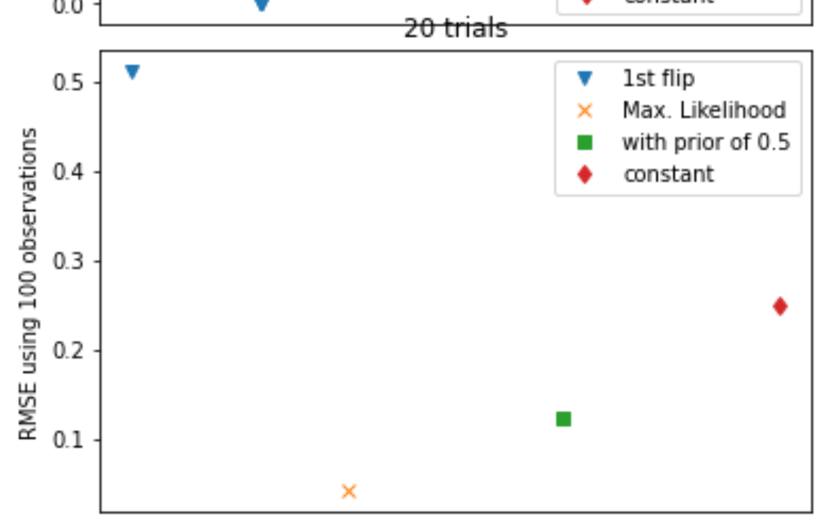
$$\mathbb{E}\left[\left(\widehat{\theta(X)} - \mathbb{E}[\widehat{\theta(X)}\,]\right)^2\right]$$

**RMSE of a parameter estimate:**

$$\sqrt{\mathbb{E}\left[\left(\widehat{\theta(X)} - \theta^*\right)^2\right]}$$

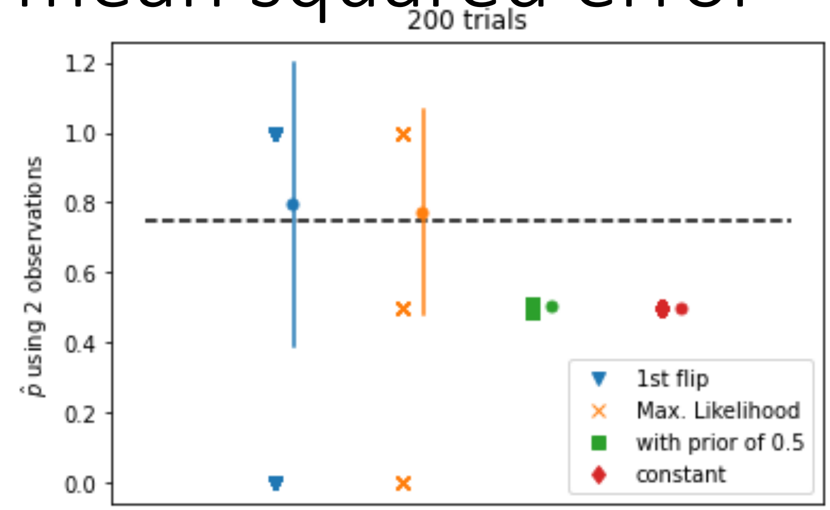


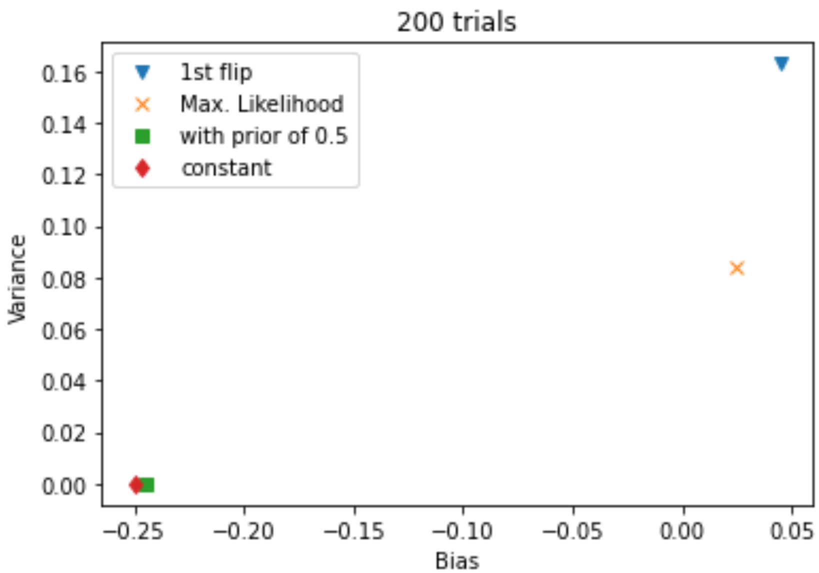



# Estimators: bias, variance, mean squared error

- With only 2 observations ML no longer wins in terms of RMSE.




# Coin data

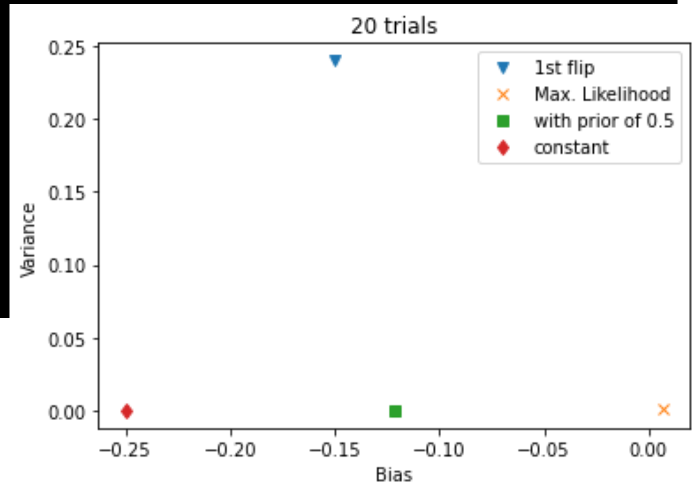
numpy.sum, numpy.ones, numpy.vstack

```
X = bernoulli.rvs(p, size=(n_flips,n_trials))

good_p = np.sum(X==0,axis=0)/n_flips
silly_p = X[0,:]==0
ignorant_p = 0.5*np.ones(n_trials)
prior_weight = 100
prior_value =0.5
prior_p = (n_flips*np.sum(X==0,axis=0)/n_flips
            + prior_weight*prior_value)/(n_flips+prior_weight)

names = ['1st flip',"Max. Likelihood",
           "with prior of 0.5",'constant']
estimates = np.vstack([silly_p,good_p,prior_p,ignorant_p])
```

# Coin data

**numpy.mean, numpy.var, numpy.std**

```
markers = ['v','x','s','d','+']
plt.figure()
for i, estimate_name in enumerate(names):
  bias = np.mean( estimates[i])-(1-p)
  h = plt.plot(bias, np.var(estimates[i]), marker=markers[i],
                linestyle='None', label=estimate_name)

plt.xlabel('Bias')
plt.ylabel('Variance')
plt.title('{} trials'.format(n_trials))
plt.legend(loc='best')
```

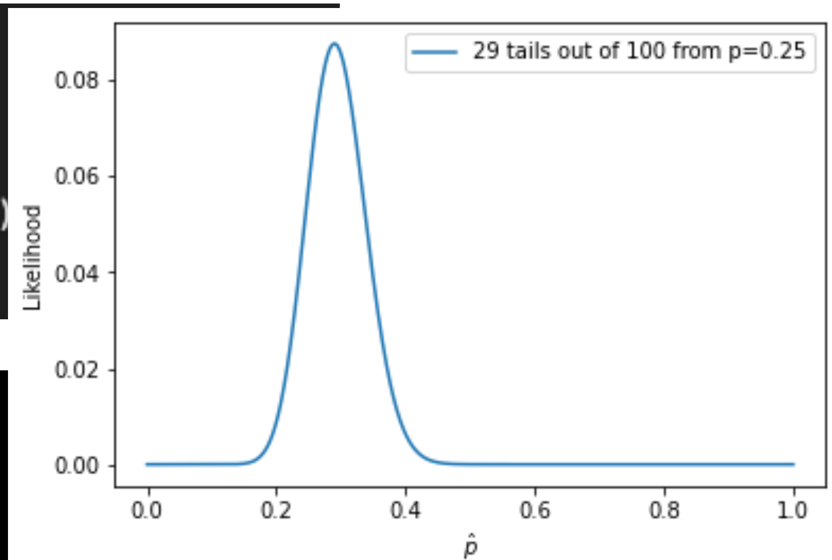


# Coin data

**Maximum likelihood estimation**

scipy.stats.binom.pmf

```
def bernoulli_likelihood(ps, X):
    like = []
    for p in ps:
        like.append(st.binom.pmf(np.sum(X),len(X)
    return like
```

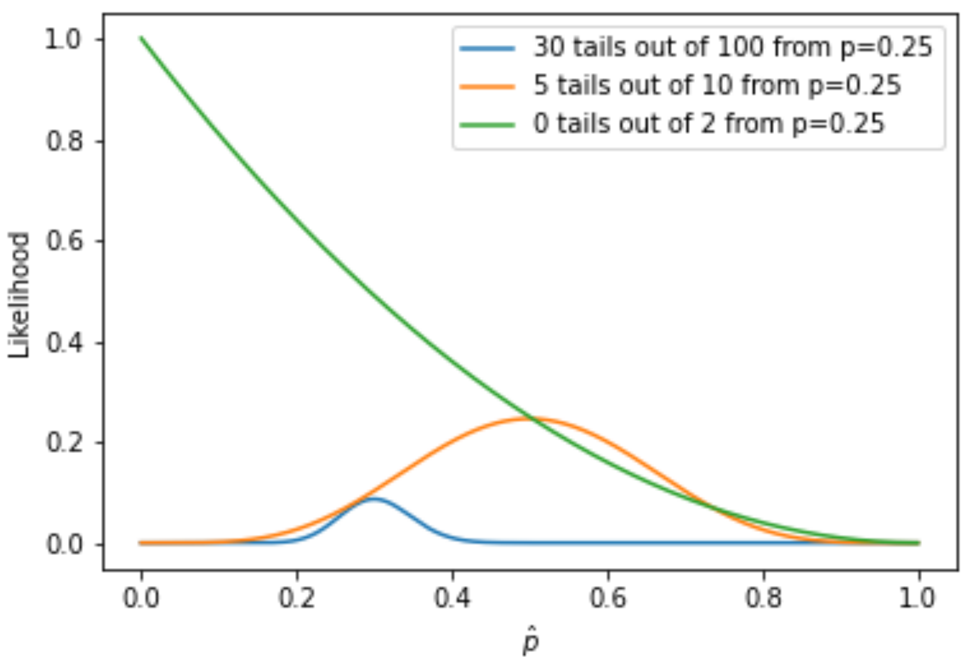


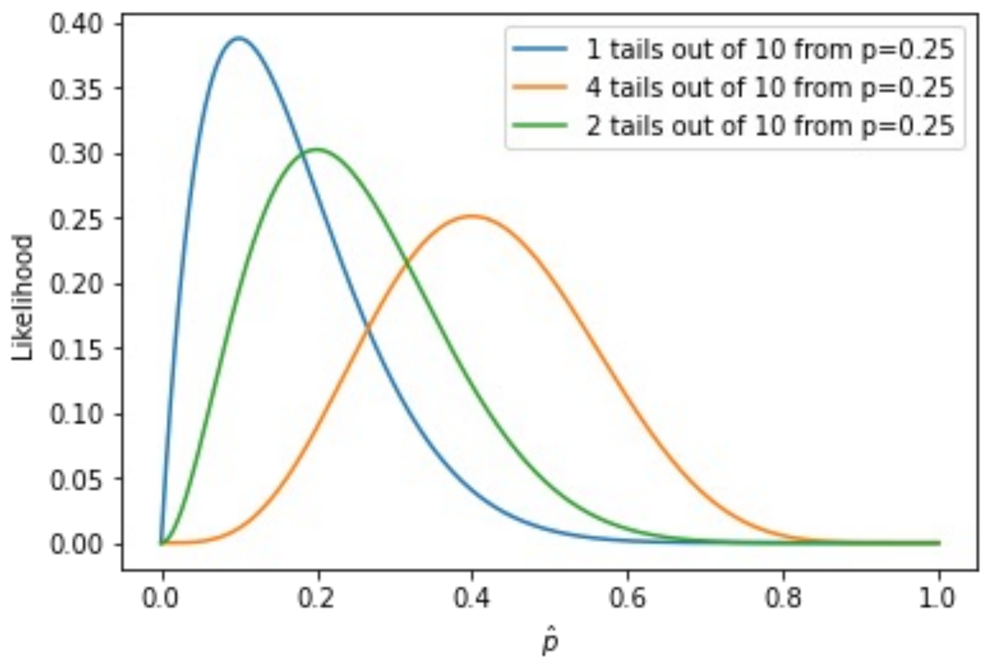

```
ps = np.linspace(0,1,1000)

N, p = 100, 0.25
X = st.bernoulli.rvs(p,size=N)
plt.plot(ps,bernoulli_likelihood(ps,X), l
      label ='{} tails out of {} from p={}'.format(np.sum(X),N,p))
plt.xlabel(r'$\hat{p}$')
plt.ylabel('Likelihood')
plt.legend()
```

# Coin data

**Maximum likelihood estimation**

## scipy.stats.binom.pmf



# Fair dice data

Roll two dices and observe the numbers on top.

<u>Attributes</u>

(die 1, die 2) $\in \{1, \dots, 6\}^2$

<u>Analysis Questions</u>

What is the distribution of the first die after 100 rolls?

What is the distribution of the sum of the two die?

How likely is the event A={(5,1)}?

How likely is the event B={(3,3),(4,3),(4,4),(4,5),(5,4)}?

# Modified dice data

Roll one die and observe the number on top, reroll the second die until it is equal or greater than the first.

<u>Attributes</u>

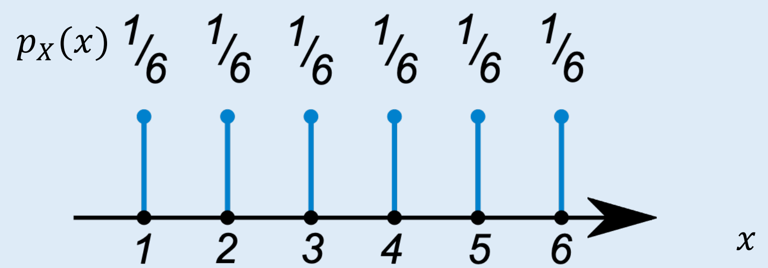
(die 1, die 2) $\in \{1, \ldots, 6\}^2$

<u>Analysis Questions</u>

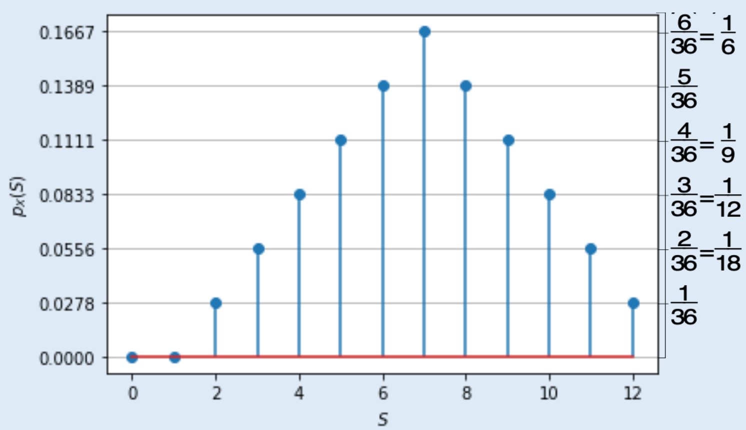
What is the distribution of the first die after 100 rolls?
What is the distribution of the sum of the two die?
How likely is the event A={(5,1)}?
How likely is the event B={(3,3),(4,3),(4,4),(4,5),(5,4)}?

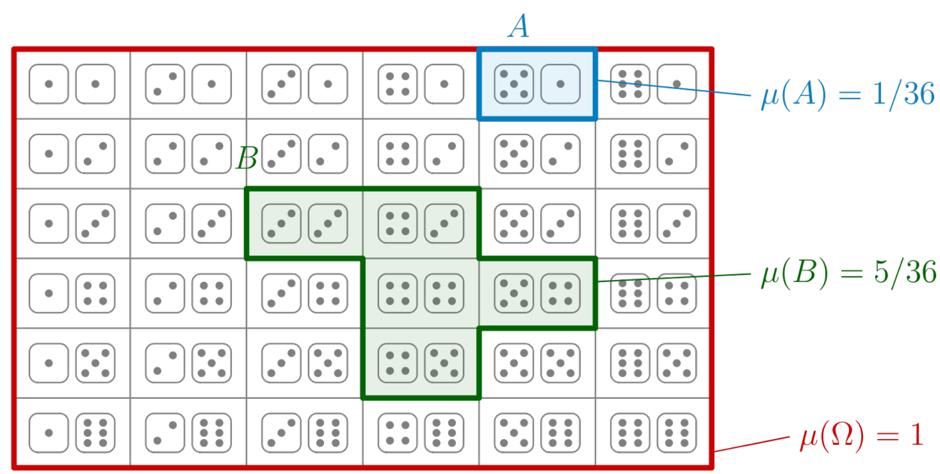
# Fair dice data

$X$ is the number rolled on a fair die



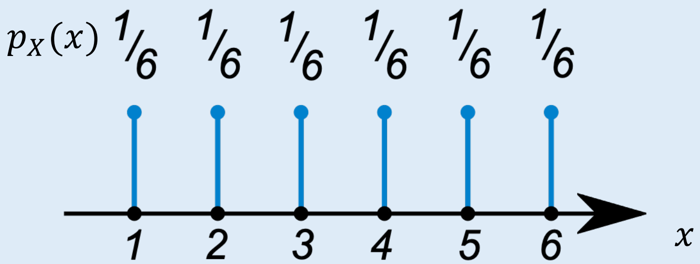

$X$ is the sum of numbers rolled on 2 dice



# Modified dice data

$X$ is the number rolled on a fair die




$X$ is the sum of numbers rolled on 2 dice