Administrative Items: Homework 1

- Homework 1 Out Now (Due 9/7 @ 11:59pm)

- Counts as 6 points towards final grade
  - HWs are generally 6 or 7 points each for a total of 60%

- 4 Questions aimed at probability of random events

- Available on course website:
  pachecoj.com/courses/csc380_fall21/

- Submit PDF of answers + work on D2L

Administrative Items: Office Hours

Enfa

- Mondays, 10:30-11:30am (hybrid)
- In-person component: Gould-Simpson Rm 934, Desk #6

Saiful

- Tuesdays, 10-11am (hybrid)
- In-person component: Gould-Simpson Rm 942

Jason Wednesdays, 10-11am (Zoom only)

Zoom office hour meeting coordinates available on D2L

Recap

- A random process is modeled by:
  - **Sample space** Ω is the set of all possible outcomes
  - **Events** E each being a subset of Ω
  - **Probability function** P assigns a probability in $[0, 1]$ to each event

- Axioms of probability

1. For any event E, $0 \leq P(E) \leq 1$
2. $P(\Omega) = 1 \text{ and } P(\emptyset) = 0$
3. For any *finite* or *countably infinite* sequence of pairwise mutually disjoint events $E_1, E_2, E_3, \ldots$

$$P\left(\bigcup_{i\geq 1} E_i\right) = \sum_{i\geq 1} P(E_i)$$

Recap

- A **random variable** is a function of samples to real values: $X : \Omega \to \mathbb{R}$

- $X = x$ is an event with probability: $p(X = x) = \sum_{\omega \in \Omega : X(\omega) = x} P(\omega)$

- Some fundamental rules of probability:
  - Conditional: $p(X \mid Y) = \frac{p(X,Y)}{p(Y)} = \frac{p(X,Y)}{\sum_x p(X=x,Y)}$
  - Law of total probability: $p(Y) = \sum_x p(Y, X = x)$
  - Probability chain rule: $p(X, Y) = p(Y)p(X \mid Y)$

Outline

- Useful Discrete Distributions (+numpy.random)
- Continuous Probability
- Useful Continuous Distributions

Outline

- Useful Discrete Distributions (+numpy.random)
- Continuous Probability
- Useful Continuous Distributions

Numpy Library

Package containing many useful numerical functions…

CONDA

If you use `conda`, you can install NumPy from the `defaults` or `conda-forge` channels:

```
# Best practice, use an environment rather than install in the base env
conda create -n my-env
conda activate my-env
# If you want to install from conda-forge
conda config --env --add channels conda-forge
# The actual install command
conda install numpy
```

PIP

If you use `pip`, you can install NumPy with:

```
pip install numpy
```

…we are interested in numpy.random at the moment

numpy.random

NumPy

- Lightweight library for sampling random variables
- Supports most standard discrete PMFs and continuous PDFs
- Also handles random permutations of lists
- Imported along with Numpy as,

```
import numpy as np
```

- Functions accessible via np.random.functionname
- There are multiple random number generators… distinguishing them and seeding them can get a bit confusing…

Docs: https://numpy.org/doc/1.16/reference/routines.random.html

numpy.random

Allows sampling from many common distributions

Set (global) random seed as,

```
import numpy as np

seed = 12345
np.random.seed(seed)
```

- Fine for this class, but a bad habit to get into
- Sets global seed, which can be problematic in larger projects
- Better to create new instance of the Random Number Generator (RNG)

```
seed = 67890
rng = np.random.default_rng(seed)
```

- Pass around `rng` object to generate random numbers

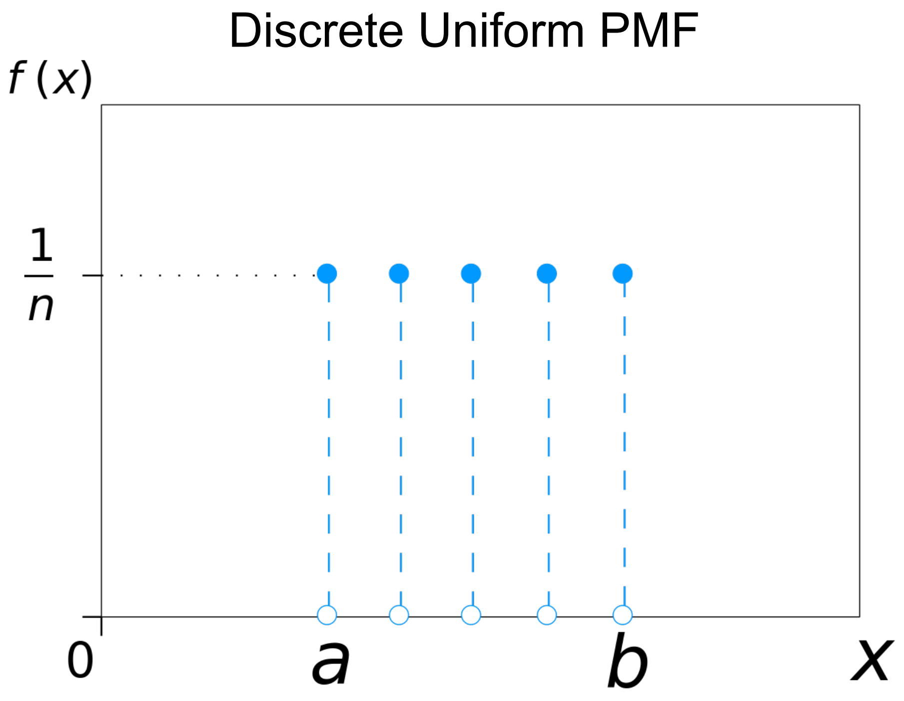
Useful Discrete Distributions

Until now, we have worked with the **uniform distribution** on N values with PMF,

$$p(X = k) = \frac{1}{N}$$

More generally, we define on an interval [a,b] and assign a *named* PMF:

$$\text{Uniform}(X = k; a, b, N) = \frac{1}{N}$$



The *average* or *mean* or *expected* value from the Uniform is:

$$E[X] = \frac{a + b}{2}$$

There are many other useful, but non-uniform, probability distributions

numpy.random

numpy.random.randint

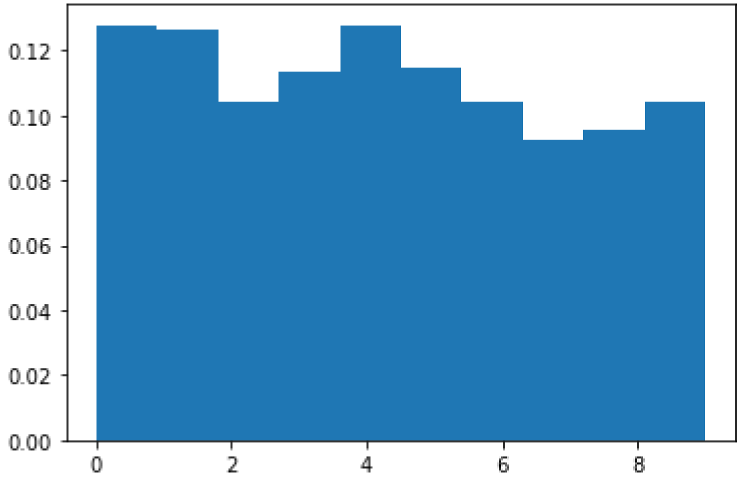
numpy.random.randint(*low*, *high=None*, *size=None*, *dtype='l'*)

Return random integers from *low* (inclusive) to *high* (exclusive).

Return random integers from the "discrete uniform" distribution of the specified dtype in the "half-open" interval [*low*, *high*). If *high* is None (the default), then results are from [0, *low*).

Sample a discrete uniform random variable,

```
import matplotlib.pyplot as plt
X = np.random.randint(0,10,1000)
count, bins, ignored = plt.hist(X, 10, density=True)
plt.show()
```



- **Caution** Interval is [low,high) and upper bound is **exclusive**
- Most calls (**but not** all) in numpy involving intervals follow this pattern
- `Size` argument accepts tuples for sampling ndarrays

Useful Discrete Distributions

Bernoulli *A.k.a. the **coinflip** distribution on binary RVs* $X \in \{0, 1\}$

$$p(X) = \pi^X (1-\pi)^{(1-X)}$$

Where π *is the probability of **success** (e.g. heads), and also the mean*

$$\mathbf{E}[X] = \pi \cdot 1 + (1-\pi) \cdot 0 = \pi$$

Suppose we flip N independent coins $X_1, X_2, \ldots, X_N$, what is the distribution over their sum $Y = \sum_{i=1}^N X_i$

Num. "successes" out of N trials

Num. ways to obtain k successes out of N

Binomial Dist. $$p(Y = k) = \binom{N}{k} \pi^k (1-\pi)^{N-k}$$

Binomial Mean: $\mathbf{E}[Y] = N \cdot \pi$ Sum of means for N indep. Bernoulli RVs

numpy.random

numpy.random.binomial

numpy.random.binomial(*n*, *p*, *size=None*)

Draw samples from a binomial distribution.

Samples are drawn from a binomial distribution with specified parameters, n trials and p probability of success where n an integer >= 0 and p is in the interval [0,1]. (n may be input as a float, but it is truncated to an integer in use)

Binomial PMF

$$p(Y=k) = \binom{N}{k} \pi^k (1-\pi)^{N-k}$$

Example A company drills 9 wild-cat oil exploration wells, each with an estimated probability of success of 0.1. All nine wells fail. What is the probability of that happening?

Answer this by simulating 20,000 trials…

```
N = 20000
p = 0.1
wells = 9
X = np.random.binomial(wells, p, N)
odds = sum( X == 0 )/N
odds
```

```
0.38685
```

Useful Discrete Distributions

Question: How many flips until we observe a success?

Geometric *Distribution on number of independent draws of* $X \sim \text{Bernoulli}(\pi)$ *until success:*

$$p(Y = n) = (1 - \pi)^{n-1}\pi \qquad \mathbf{E}[Y] = \frac{1}{\pi}$$

E.g. for fair coin $\pi = 1/2$ takes two flips on avg.

e.g. there must be n-1 failures (tails) before a success (heads).

Question: How many more flips of we have already seen k failures?

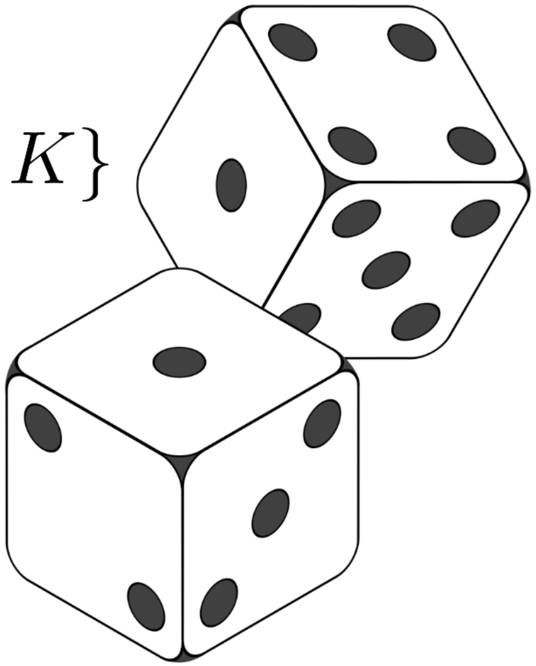
$$p(Y = n + k \mid Y > k) = \frac{p(Y{=}n{+}k, Y{>}k)}{p(Y{>}k)} = \frac{p(Y{=}n{+}k)}{p(Y{>}k)}$$

$$= \frac{(1{-}\pi)^{n+k-1}\pi}{\sum_{i=k}^{\infty}(1{-}\pi)^i\pi} = \frac{(1{-}\pi)^{n+k-1}\pi}{(1{-}\pi)^k} = (1 - \pi)^{n-1}\pi = p(Y = n)$$

For $0 < x < 1, \sum_{i=k}^{\infty} x^i = x^k/(1 - x)$

Corollary: $p(Y > k) = (1 - \pi)^{k-1}$

Useful Discrete Distributions



Categorical *Distribution on integer-valued RV* $X \in \{1, \ldots, K\}$

$$p(X) = \prod_{k=1}^{K} \pi_k^{\mathbf{I}(X=k)} \quad \text{or} \quad p(X) = \sum_{k=1}^{K} \mathbf{I}(X=k) \cdot \pi_k$$

with parameter $p(X=k) = \pi_k$ *and Kronecker delta:*

$$\mathbf{I}(X=k) = \begin{cases} 1, & \text{If } X = k \\ 0, & \text{Otherwise} \end{cases}$$

Can also represent X as *one-hot* binary vector,

$$X \in \{0,1\}^K \quad \text{where} \quad \sum_{k=1}^{K} X_k = 1 \quad \text{then} \quad p(X) = \prod_{k=1}^{K} \pi_k^{X_k}$$

This representation is special case of the **multinomial distribution**

Useful Discrete Distributions

What if we count outcomes of *N* independent categorical RVs?

Multinomial *Distribution on K-vector* $X \in \{0, N\}^K$ *of counts of N repeated trials* $\sum_{k=1}^K X_k = N$ *with PMF:*

$$p(x_1, \ldots, x_K) = \binom{n}{x_1 x_2 \ldots x_K} \prod_{k=1}^{K} \pi_k^{x_k}$$

Number of ways to partition N objects into K groups:

$$\binom{n}{x_1 x_2 \ldots x_K} = \frac{n!}{x_1! x_2! \ldots x_K!}$$

Leading term ensures PMF is properly normalized:

$$\sum_{x_1} \sum_{x_2} \ldots \sum_{x_K} p(x_1, x_2, \ldots, x_K) = 1$$

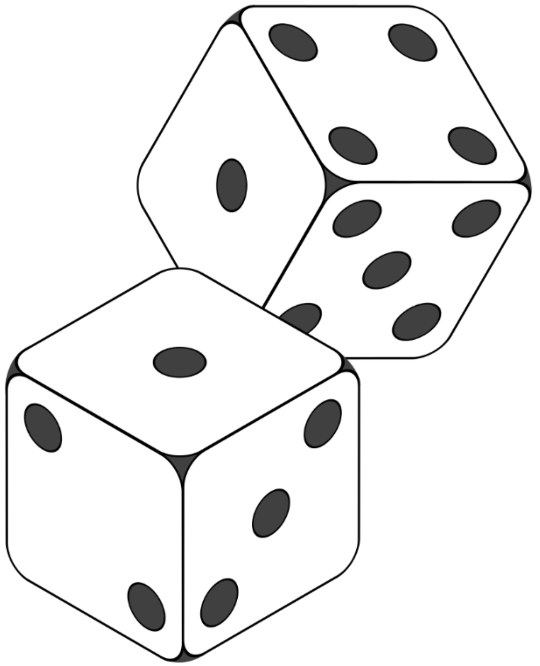
numpy.random

numpy.random.multinomial

numpy.random.multinomial(*n*, *pvals*, *size=None*)

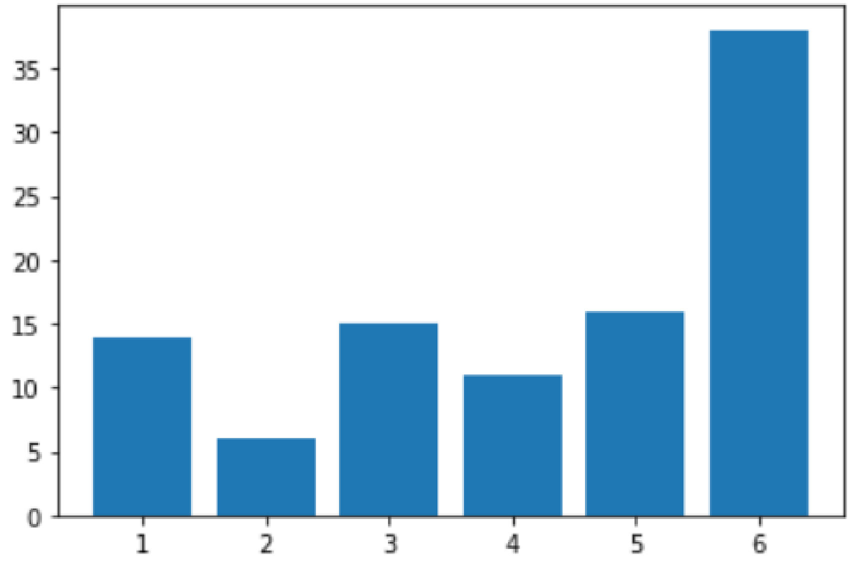
Draw samples from a multinomial distribution.

The multinomial distribution is a multivariate generalisation of the binomial distribution. Take an experiment with one of `p` possible outcomes. An example of such an experiment is throwing a dice, where the outcome can be 1 through 6. Each sample drawn from the distribution represents *n* such experiments. Its values, `X_i = [X_0, X_1, ..., X_p]`, represent the number of times the outcome was `i`.



Example Simulate 100 throws of a *"loaded"* die that has 3X the chance of rolling 6, and equal chance for remaining numbers.

```
N = 100
p_unnorm = np.array([1,1,1,1,1,3])
p = p_unnorm / sum(p_unnorm) # normalize
X = np.random.multinomial(N, p)
plt.bar(np.arange(6) + 1, X)
plt.show()
```



Note: Probability vector <u>must</u> be valid PMF (nonnegative, normalized a.k.a sum to 1)

numpy.random

How to simulate Bernoulli? Categorical?

Bernoulli is equivalent to a single draw from a binomial,

```
X = np.random.binomial(n=1, p=0.5) # fair coin flip
print(X)
```

```
0
```

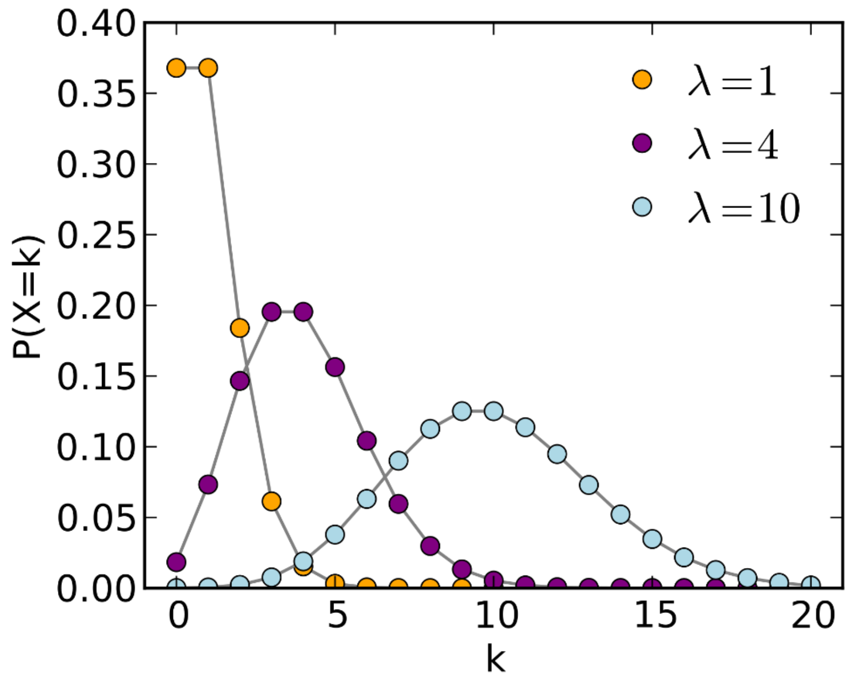
Categorical is equivalent to a single draw from a multinomial,

```
X = np.random.multinomial(1, [0.5, 0.5]) # also a fair coin flip
print(X)
```

```
[0 1]
```

Useful Discrete Distributions

A ***Poisson*** *RV X with rate parameter* λ *has the following distribution:*



$$p(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

Mean and variance both scale with parameter

$$\mathbf{E}[X] = \mathbf{Var}[X] = \lambda$$

Represents number of times an *event* occurs in an interval of time or space.

Ex. *Probability of overflow floods in 100 years,*

$$p(k \text{ overflow floods in 100 yrs}) = \frac{e^{-1}1^k}{k!}$$

Avg. 1 overflow flood every 100 years, makes setting rate parameter easy.

Additivity *The sum of a finite number of Poisson RVs is a Poisson RV.*

$$X \sim \text{Poisson}(\lambda_1), \quad Y \sim \text{Poisson}(\lambda_2), \quad X + Y \sim \text{Poisson}(\lambda_1 + \lambda_2)$$

numpy.random

numpy.random.poisson

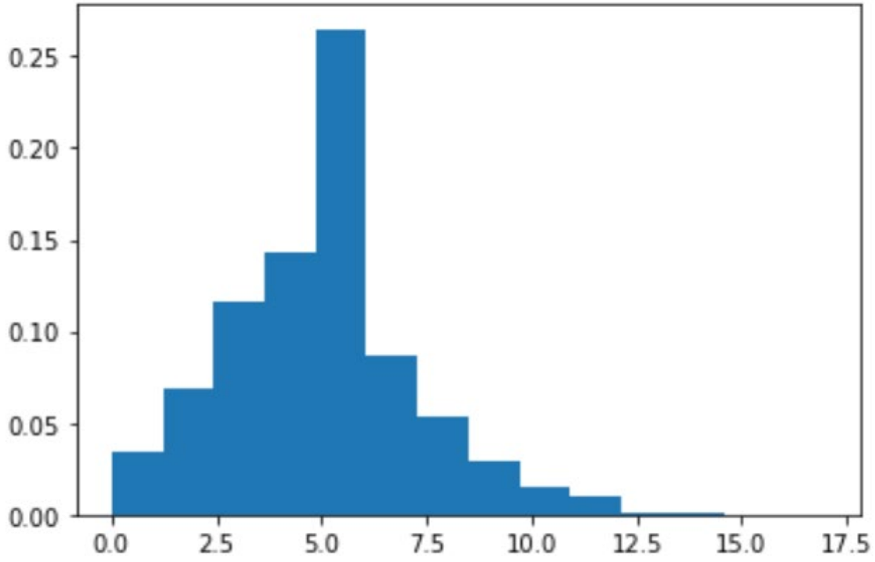
numpy.random.poisson(*lam=1.0, size=None*)

Draw samples from a Poisson distribution.

The Poisson distribution is the limit of the binomial distribution for large N.

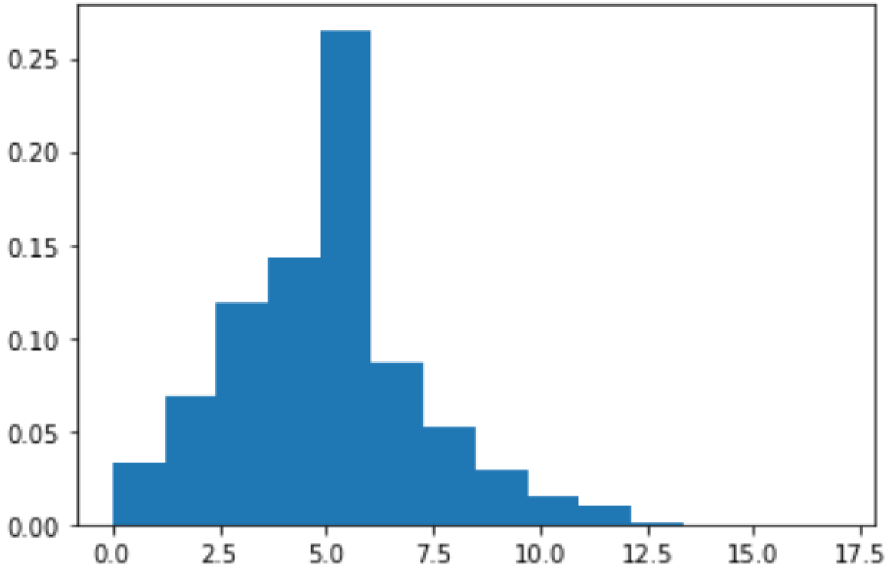
Example Simulate 100,000 draws from a Poisson with rate 5.0,

```
X = np.random.poisson(5, 100000)
ount, bins, ignored = plt.hist(X, 14, density=True)
plt.show()
```



Additivity *The sum of a finite number of Poisson RVs is a Poisson RV.*

```
X = np.random.poisson(2.5, 100000)
Y = np.random.poisson(2.5, 100000)
count, bins, ignored = plt.hist(X+Y, 14, density=True)
plt.show()
```



Administrative Items

- Special office hours
  - With Jason
  - Tomorrow @ 11am (Zoom)
  - See D2L Events for Zoom coordinates
  - TA office hours Monday (?) / Tuesday next week

- Revision to HW1 (See yesterday's note on Piazza)

- Problem 1(e) guidance…