METIS

Day 2: Overview of Linear Algebra & Basic Statistics

John Navarro
john.navarro@thisismetis.com
https://www.linkedin.com/in/johnnavarro/

METIS

Why Linear Algebra?

Linear Systems

Linear Systems: A system of linear equations

So what's that??

Who remembers Algebra class?

METIS

Solving Linear Systems

- Let's say we have to pick between 2 different photo hosting subscription services that have extremely bizarre payment plans.
- The first service costs $800 upfront and $20 a month after that. We can represent this plan as

  y = 800 + 20x

  where y represents dollars paid out over x amount of months.

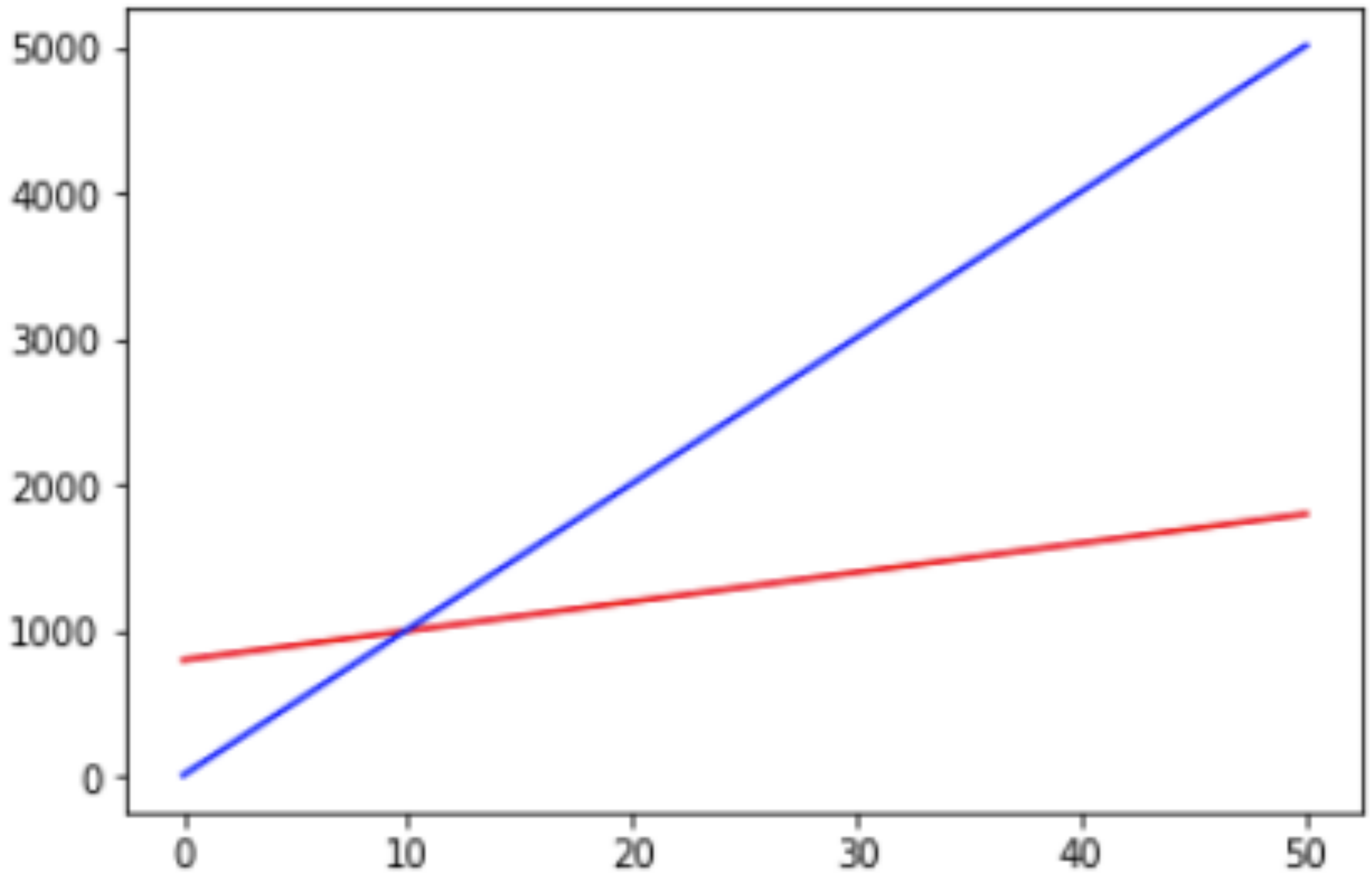
- The second service has a setup fee of $10, and costs $100 a month after that to keep it running. We can represent this offer as

  y = 10 +100x

  where y represents dollars paid out over x amount of months.

Does this kind of problem seem familiar?

Solving Linear Systems



$y = 800 + 20x$ $y = 10 + 100x$

$800 + 20x = 10 + 100x$

$790 = 80x$

$x = 9.875$

Solution:

$x = 9.875$, $y = 997.5$

aka: (9.875, 997.5)

METIS

The Augmented Matrix

Point Slope Form: $y = mx + b$ $y = 800 + 20x$ $y = 10 + 100x$

General Form: $Ax + By = c$ $20x - 1y = 800$ $100x - 1y = 10$

Augmented Matrix:

$$\begin{array}{c} \left[\begin{array}{cc|c} A & B & c \\ A & B & c \end{array}\right] \end{array} \begin{array}{l} \leftarrow \text{from eq. 1} \\ \leftarrow \text{from eq. 2} \end{array}$$

$$\left[\begin{array}{cc|c} 20 & -1 & -800 \\ 100 & -1 & -10 \end{array}\right]$$

Gaussian Elimination

So how does one solve the linear system in this format?

$$
\begin{array}{cc|c} Ax & +\,By & = \quad c \end{array}
\qquad\qquad
\begin{array}{cc|c} Ax & +\,By & = \quad c \end{array}
$$

$$
\left[\begin{array}{cc|c} 20 & -1 & -800 \\ 100 & -1 & -10 \end{array}\right]
\longrightarrow
\left[\begin{array}{cc|c} 1 & 0 & 9.875 \\ 0 & 1 & 997.5 \end{array}\right]
\quad
\begin{array}{l} 1x + 0y = 9.875 \\ 0x + 1y = 997.5 \end{array}
$$

You don't need to know how to do this particular transformation for this class. However, I've created slides detailing this procedure. Right now, I can either:

1) explain it now, or
2) Skip it! and let you look at it later if you'd like.

Gaussian Elimination

Matrix Row Operations

1. Any Two Rows can be Swapped
2. Any row can be multiplied by a nonzero constant
3. Any row can be added to another row

This slide can be skipped... Here for reference.

METIS

Gaussian Elimination

1. Any Two Rows can be Swapped
2. Any row can be multiplied by a nonzero constant
3. Any row can be added to another row

$$\left[\begin{array}{cc|c} 20 & -1 & -800 \\ 100 & -1 & -10 \end{array}\right] \longrightarrow R1 = R1 * (1/20) \longrightarrow \left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 100 & -1 & -10 \end{array}\right]$$

This slide can be skipped... Here for reference.

Gaussian Elimination

1. Any Two Rows can be Swapped
2. Any row can be multiplied by a nonzero constant
3. Any row can be added to another row

$$\left[\begin{array}{cc|c} 20 & -1 & -800 \\ 100 & -1 & -10 \end{array}\right] \longrightarrow R1 = R1 * (1/20) \longrightarrow \left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 100 & -1 & -10 \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 100 & -1 & -10 \end{array}\right] \longrightarrow R2 = R2 - (R1 * 100) \longrightarrow \left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 0 & 4 & 3990 \end{array}\right]$$

This slide can be skipped... Here for reference.

METIS

Gaussian Elimination

1. Any Two Rows can be Swapped
2. Any row can be multiplied by a nonzero constant
3. Any row can be added to another row

$$\left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 0 & 4 & 3990 \end{array}\right] \longrightarrow R2 = R2 * (1/4) \longrightarrow \left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 0 & 1 & 997.5 \end{array}\right]$$

This slide can be skipped... Here for reference.

METIS

Gaussian Elimination

1. Any Two Rows can be Swapped
2. Any row can be multiplied by a nonzero constant
3. Any row can be added to another row

$$\left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 0 & 4 & 3990 \end{array}\right] \longrightarrow R2 = R2 * (1/4) \longrightarrow \left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 0 & 1 & 997.5 \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & -1/20 & -40 \\ 0 & 1 & 997.5 \end{array}\right] \longrightarrow R1 = R1 + R2 * (1/20) \longrightarrow \left[\begin{array}{cc|c} 1 & 0 & 9.875 \\ 0 & 1 & 997.5 \end{array}\right]$$

This slide can be skipped... Here for reference.

Gaussian Elimination

Voila!

$$\left[\begin{array}{cc|c} 1 & 0 & 9.875 \\ 0 & 1 & 997.5 \end{array}\right]$$

$1x + 0y = 9.875$

$0x + 1y = 997.5$

Solution:

x = 9.875 and y = 997.5

aka: (9.875, 997.5)

This slide can be skipped... Here for reference.

METIS

Complex Linear Systems

- In most real-world situations, the dependent (y) variable is connected to more than one independent variable (x).
- Your equation starts looking a little more like this...

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n = c$$

- Example: If you're trying to figure out what a house would sell for, you might take into consideration
    - Square foot
    - Age of Home
    - Number of bedrooms
    - Number of bathrooms

One variable isn't enough to come up with a good estimate...

METIS

Matrices

- A matrix is a way to represent a table of numbers.
- Let's look at the augmented matrix we looked at earlier...

$$\left[\begin{array}{cc|c} 20 & -1 & -800 \\ 100 & -1 & -10 \end{array}\right]$$

- The convention in linear algebra when describing matrices is to specify the number of rows first, and then the number of columns. Therefore, this is a 2x3 (pronounced "two by three") matrix.

Let's build one

METIS

Matrices

with Numpy

We're not going to use Numpy much in this class, and will instead focus on pandas operation in future lessons. However, it's good to know it exists in case you ever need to do such operations outside of pandas.

Numpy: A Python package for scientific computing

```
import numpy as np

matrix = np.array([
  [20, -1, 80],
  [100, -1, 10]
] , dtype=np.float32)
```

Setting the dtype to float so that precision is preserved when doing mathematical operations.

Matrix Components

Let's zoom out and look at the individual components of a matrix

- **Scalars:** A mathy way of saying an individual number
- **Vector:** Otherwise known as an array or list or collection of scalars
- **Matrix:** A 2 dimensional collection of vectors

METIS

Vectors

$$\left[\begin{array}{cc|c} 20 & -1 & -800 \\ 100 & -1 & -10 \end{array}\right]$$

- **Vector:** Otherwise known as an array or list or collection of scalar
- **Row Vector:** a row from a matrix

  [20 -1 -800] and [100 -1 -10]

- **Column Vector**: a column from a matrix

$$\begin{bmatrix} 20 \\ 100 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \text{ and } \begin{bmatrix} -800 \\ -10 \end{bmatrix}$$

The word **vector** generally refers to the column vector

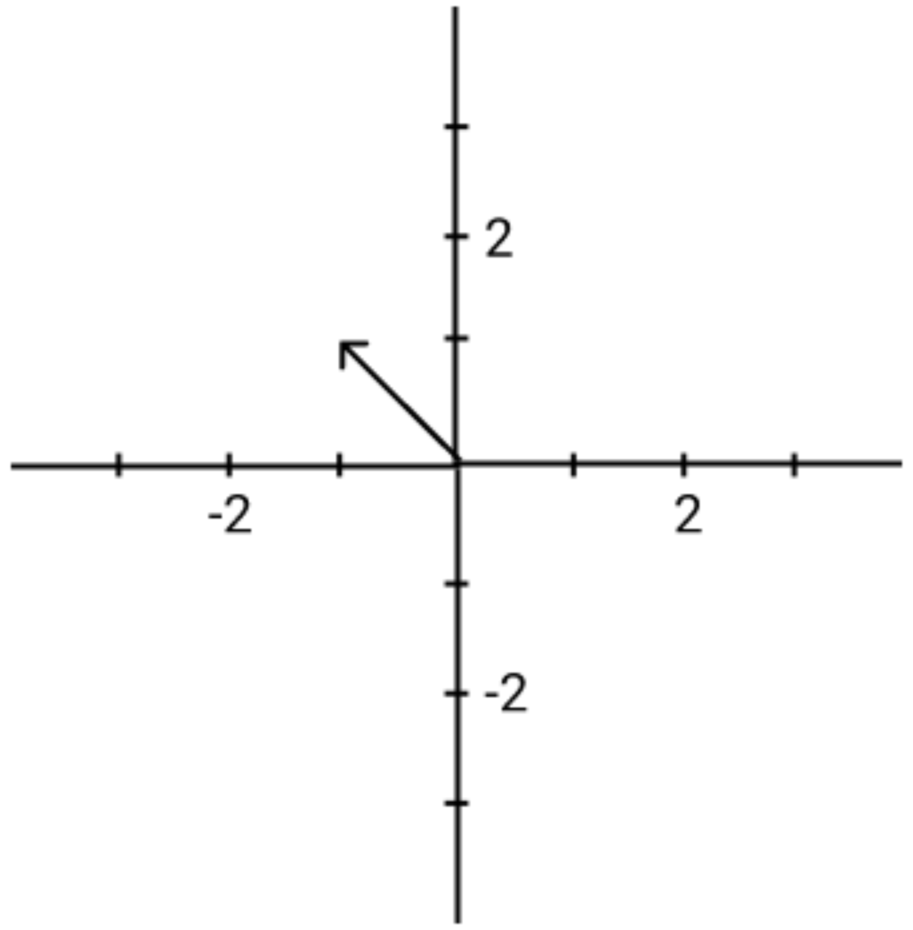
Vector Representation

- When a vector contains 2 or 3 elements, it can be visualized on a coordinate grid easily

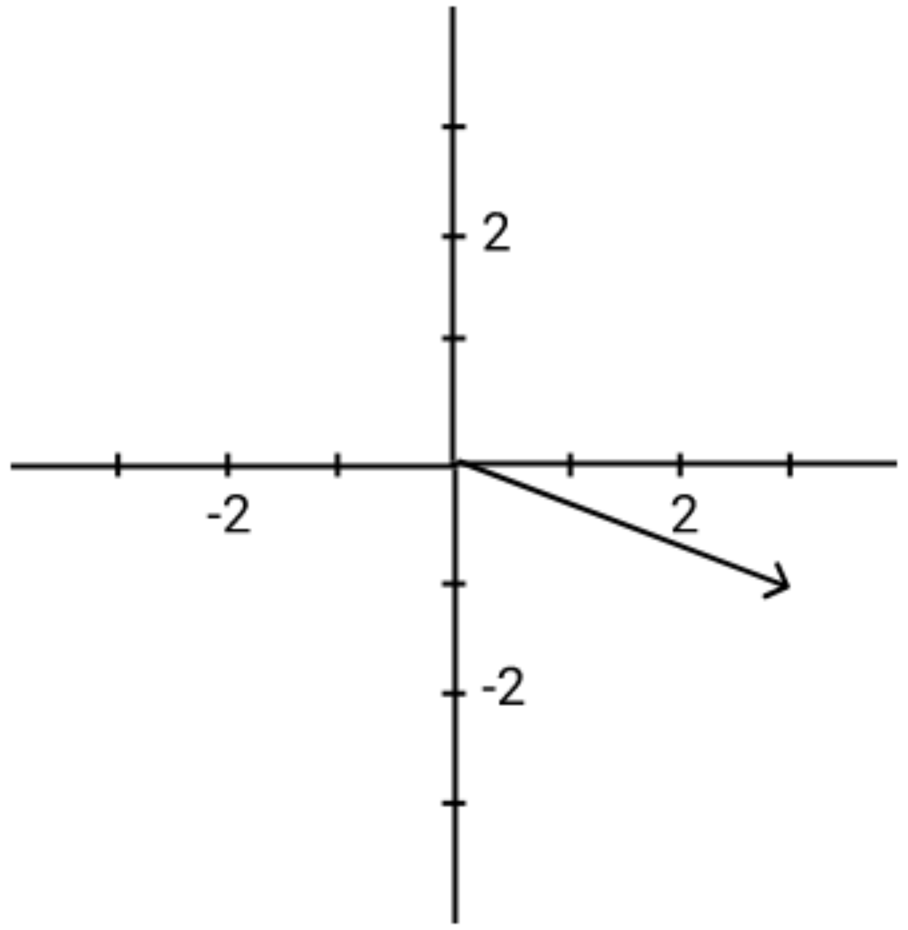
- A vector is visualized on a coordinate grid using arrows, not using coordinates, from the origin (0, 0)

- Arrows are used to visualize vectors because it emphasizes the two key properties each vector has - **direction** (the way it's pointing) and **magnitude** (its length)

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$



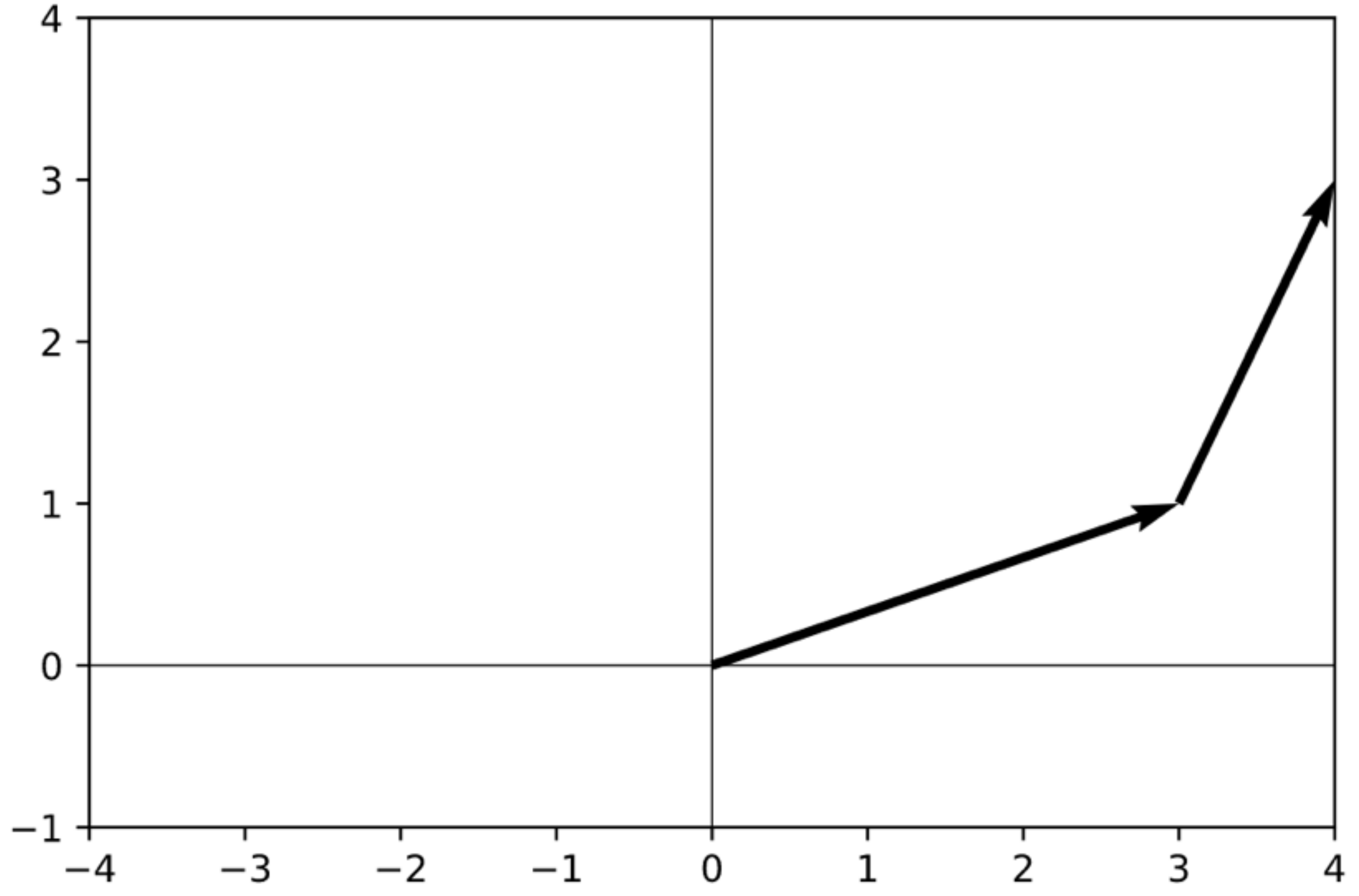
$$\begin{bmatrix} 3 \\ -1 \end{bmatrix}$$



Vector Addition

- Vectors can be added and subtracted together.
- When two vectors are added together, a new vector is created.

$$\begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

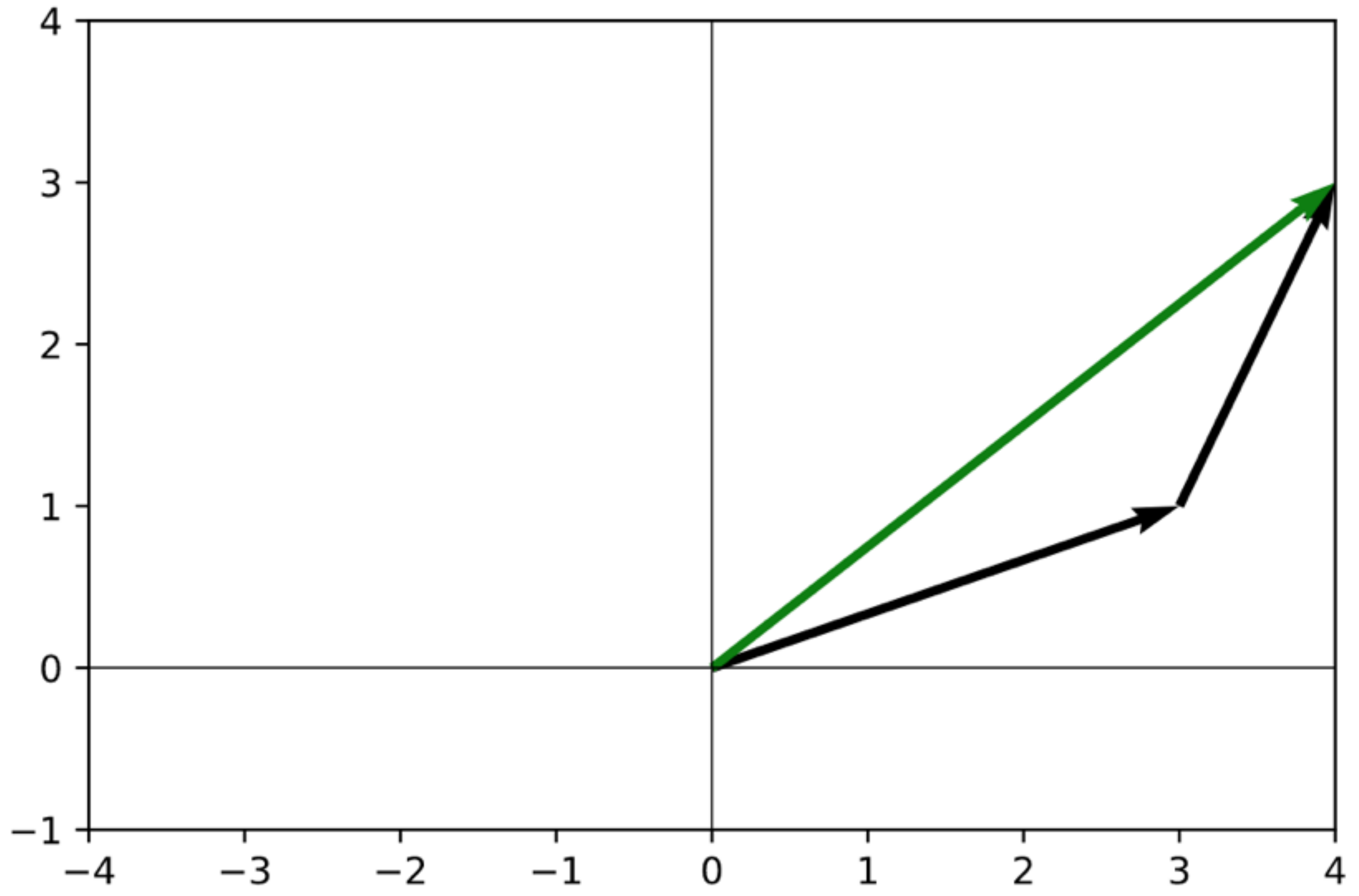


Vector Addition

- Vectors can be added and subtracted together.

- When two vectors are added together, a new vector is created.

$$\begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$



METIS

Vector Addition

with Numpy

- Vectors can be added and subtracted together.
- When two vectors are added together, a new vector is created.
- Only works if the vectors are the same shape (i.e. same value count)

$$\begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

```
vector_one = np.asarray([3, 1], dtype=np.float32)

vector_two = np.asarray([1, 2], dtype=np.float32)

vector_one + vector_two
```

Output:

```
array([ 4.,  3.], dtype=float32)
```

NumPy Arrays / Vectors

```
my_first_vector = np.array([10,12,13])
my_second_vector = np.array([1,4,3.5])

vectors_added = my_first_vector + my_second_vector
vectors_divided = my_first_vector / my_second_vector
vectors_multiplied = my_first_vector * my_second_vector
vectors_subtracted = my_first_vector - my_second_vector
```

```
Original vectors: [10 12 13] [ 1.   4.   3.5]
Added:      [ 11.   16.   16.5]
Divided:    [ 10.          3.          3.71428571]
Multiplied:   [ 10.   48.   45.5]
Subtracted:   [ 9.   8.   9.5]
```

METIS

Vector Scalar Multiplication

- We can scale vectors by multiplying or dividing them by a scalar (a real number)

$$3 * \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 3 \end{bmatrix}$$

```
vector_one = np.asarray([3, 1], dtype=np.float32)

3 * vector_one
```

Output:

```
array([ 9.,  3.], dtype=float32)
```

METIS

Vector Scalar Addition

- Similarly, you can add or subtract scalars from a vector

$$\begin{bmatrix} 3 \\ 1 \end{bmatrix} + 1 = ?$$

What vector will this return?

NumPy Arrays / Vectors

```
my_array = np.array([12,46,3.4])
my_mult_array = my_array * 3
my_div_array = my_array / 3
my_sub_array = my_array - 3
my_add_array = my_array + 3
```

```
Original array:   [ 12.   46.    3.4]
Multiplied by 3:  [  36.   138.    10.2]
Divided by 3:   [  4.          15.33333333   1.13333333]
Subtracted 3 from:[  9.   43.    0.4]
Added 3 to:   [ 15.   49.    6.4]
```

What does this do?

```
my_array = np.array([12,46,3.4])

my_array > 10
```

You might remember this from pandas...

```
array([ True,  True, False], dtype=bool)
```

Filtering a NumPy Array

```
my_array = np.array([12,46,3.4])

my_array[my_array > 10]
```

Exercise

- Create a `NumPy array` called `heights` = [6.5,7.3,5.1,4.9]
- Multiply all of the heights by 4 and store that into a new variable called `huge_people`
- Create a new array `basketballers` with only those entries in heights that are > 6
- Create another array `little_people` with only those entries in heights that are < 5

Exercise

```
heights = np.array([6.5,7.3,5.1,4.9])
huge_people = heights * 4
basketballers = heights[heights > 6]
little_people = heights[heights < 5]
```

METIS

Questions?

Matrix Scalar Operations

You can use the same scalar operations on matrices that we practiced on vectors

```
scalar_addition_matrix = my_first_matrix + 5
```

```
Original matrix:
[[20 12]
 [30  1]]
Adding the value 5 to the matrix:
[[25 17]
 [35  6]]
```

METIS

```
scalar_addition_matrix = my_first_matrix * 2
```

```
Original matrix:
[[20 12]
 [30  1]]
Each matrix value doubled:
[[40 24]
 [60  2]]
```

METIS

Matrix Operations

```
my_first_matrix = np.array([[20,12],[30,1]], dtype=float32)
my_add_matrix = my_first_matrix + 5
my_sub_matrix = my_first_matrix -20
my_div_matrix = my_first_matrix / 5
my_mult_matrix = my_first_matrix * 20
```

matrices must be the same size for these to work

```
Original matrix:
[[20 12]
 [30  1]]

Adding the value 5 to the matrix:
[[25 17]
 [35  6]]

Subtracting 20 from the matrix:
[[  0  -8]
 [ 10 -19]]
```

```
Dividing the matrix by 5:
[[4 2]
 [6 0]]

Multiplying the matrix by 20:
[[400 240]
 [600  20]]
```

METIS

The Dot Product

METIS Vector Multiplication (Dot Product)

- Dot product is a vector multiplication operation that involves multiplying the vector elements in a particular way.
- Results in a scalar value rather than creating a new vector
- Only works if vectors are the same shape

$$\begin{bmatrix} 1 & 3 & -5 \end{bmatrix} \begin{bmatrix} 4 \\ -2 \\ -1 \end{bmatrix} = (1 * 4) + (3 * -2) + (-5 * -1)$$

$$= 4 - 6 + 5$$

$$= 3$$

```
vector_one = np.asarray([1, 3, -5], dtype=np.float32)

vector_two = np.asarray([4, -2, -1], dtype=np.float32)

vecto_one.dot(vector_two); np.dot(vector_one, vector_two)
```

Output:

```
3.0
```

METIS Matrix Multiplication (Dot Product)

In order for multiplication of two matrices/vectors AxB to be defined, the number of columns in the first operand must equal the number of rows in the second operand.

The resulting new matrix will have a number of rows equal to the number of rows in the first operand and the number of columns equal to the second operand.

```
matrix_one:   matrix_two:

[[1  2  3]     [[7   8]
 [4  5  6]]  ·  [9  10]
                [11 12]]
```

What will it's shape be?

Let's look at what matrix will result...

METIS

Matrix Multiplication (Dot Product)

```
matrix_one:

[[1  2  3]
 [4  5  6]]
```

```
matrix_two:

[[7    8]
 [9   10]
 [11  12]]
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ & \end{bmatrix}$$

Can you fill in the rest?

$$\begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

METIS

Matrix Multiplication (Dot Product)

```
matrix_one = np.array([[1, 2, 3],[4, 5, 6]])
matrix_two = np.array([[7, 8],[9, 10],[11, 12]])
dot_product = matrix_one.dot( matrix_two )
```

How would this look in pandas?

```
DataFrame1.dot(DataFrame2)
```

METIS

Matrix Multiplication

`Does matrix_a.dot( matrix_b ) = matrix_b.dot( matrix_a )?`

METIS

Linear Combination

- We've seen so far that we can multiply vectors by a scalar value and combine vectors using vector addition and vector subtraction.
- Using these operations, we can determine if a certain vector can be obtained by combining other vectors.
- Perhaps we want to know if the vectors [3, 1] and [1, 2] can combine to obtain [4, -2]. This would look like:

$$x * [3, 1] + y * [1, 2] = [4, -2]$$

Linear Combination

Let's return to our augmented matrix from our our solving linear systems problem

$$\left[\begin{array}{cc|c} 20 & -1 & -800 \\ 100 & -1 & -10 \end{array}\right]$$

How might we re-write this using linear combination, as discussed on the last slide?

$$x \begin{bmatrix} 20 \\ 100 \end{bmatrix} + y \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -800 \\ -10 \end{bmatrix}$$

The Matrix Equation

- **Matrix Equation:** representation of a linear system using only matrices and vectors

Using what we know about the dot product, we can rearrange our equation as follows:

$$x\begin{bmatrix} 20 \\ 100 \end{bmatrix} + y\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -800 \\ -10 \end{bmatrix} \longrightarrow \begin{array}{cccc} A & x & = & b \\ \begin{bmatrix} 20 & -1 \\ 100 & -1 \end{bmatrix} & \begin{bmatrix} x \\ y \end{bmatrix} & = & \begin{bmatrix} -800 \\ -10 \end{bmatrix} \end{array}$$

How might we solve for x and y?

METIS

Solving for x & the Inverse

Let's look at a simple algebra equation.

$5x = 10$

How do we solve for x?

Divide both sides by 5. This is the same as multiplying each side by the inverse of 5 (which is 1/5 or 5^{-1}).

$5x * 1/5 = 10 * 1/5$

$1x = 2$

$5^{-1} * 5x = 5^{-1} * 10$

$1x = 2$

METIS

the inverse

Any number multiplied by its reciprocal gives you 1.

Note: 1/Y is the inverse of Y

Y * 1/Y = 1

METIS

Matrix Inversion

We can apply a similar principle here:

$$\begin{bmatrix} 20 & -1 \\ 100 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -800 \\ -10 \end{bmatrix}$$

Matrix Inversion

We can apply a similar principle here

$$\begin{bmatrix} 20 & -1 \\ 100 & -1 \end{bmatrix}^{-1} \begin{bmatrix} 20 & -1 \\ 100 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 20 & -1 \\ 100 & -1 \end{bmatrix}^{-1} \begin{bmatrix} -800 \\ -10 \end{bmatrix}$$

METIS

Matrix Inversion

We can apply a similar principle here

$$\cancel{\begin{bmatrix}20 & -1\\100 & -1\end{bmatrix}^{-1}\begin{bmatrix}20 & -1\\100 & -1\end{bmatrix}}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}20 & -1\\100 & -1\end{bmatrix}^{-1}\begin{bmatrix}-800\\-10\end{bmatrix}$$

METIS

Matrix Inversion

We can apply a similar principle here

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 20 & -1 \\ 100 & -1 \end{bmatrix}^{-1} \begin{bmatrix} -800 \\ -10 \end{bmatrix}$$

Matrix Inversion

For any 2 matrices, A and B, for A/B to give an actual result, B must be square and have an inverse.

To get A/B, just compute A x inv(B)

Matrix Inversion

```
my_first_matrix = np.array([[20, 12],[30, 1]],
dtype=np.float32)
```

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix}$$

```
my_first_matrix_inverse = np.linalg.inv(my_first_matrix)
my_first_matrix_inverse
[[-0.00294118  0.03529412]
 [ 0.08823529 -0.05882353]]
```

What will happen when we run this?

```
my_first_matrix.dot(my_first_matrix_inverse))
```

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

METIS

Matrix Inversion

The Identity Matrix

```
my_first_matrix = np.array([[20, 12],[30, 1]])
```

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix}$$

```
identity = np.eye(2)
```

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

```
my_first_matrix.dot(identity) == ?
```

METIS

Matrix Inversion

The Identity Matrix

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} & \\ & \end{bmatrix}$$

METIS

Matrix Inversion

The Identity Matrix

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & \\ & \end{bmatrix}$$

Matrix Inversion

The Identity Matrix

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 12 \\ & \end{bmatrix}$$

Matrix Inversion

The Identity Matrix

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 12 \\ 30 & \end{bmatrix}$$

METIS

Matrix Inversion

The Identity Matrix

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix}$$

Matrix Inversion

The Identity Matrix

```
my_first_matrix.dot(identity)
```

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix}$$

dot product

```
my_first_matrix * identity
```

$$\begin{bmatrix} 20 & 12 \\ 30 & 1 \end{bmatrix} \text{x} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 0 \\ 0 & 1 \end{bmatrix}$$

scalar multiplication

METIS

Matrix Transpose

- The **transpose** of a matrix swaps the rows and columns of a matrix
- Because of the requirements for matrix multiplication, we sometimes want to take the transpose of a matrix to allow us to multiply matrices together

METIS

Matrix Transposition

A =

$$\begin{bmatrix} 10 & 2 \\ -1 & 5 \\ 3 & 3 \end{bmatrix}$$

transpose(A) =

$$\begin{bmatrix} 10 & -1 & 3 \\ 2 & 5 & 3 \end{bmatrix}$$

B =

$$\begin{bmatrix} 2 & 3 & 31 \end{bmatrix}$$

transpose(B) =

$$\begin{bmatrix} 2 \\ 3 \\ 31 \end{bmatrix}$$

Matrix Transposition

```
matrix_a.T
```

Exercise

- Create a 3x3 matrix C = [[12,3,4],[1,-1,10],[2,5,-2]] and a 3 x 1 vector d = [3,-2,10]
- Does the inverse of C exist? What is it?
- Mutiply C x d
- Get d x C to work using transpose

METIS

Questions?

METIS

Questions?

Bring it all together

- Example: If you're trying to figure out what a house would sell for, you might take into consideration
    - Square foot
    - Age of Home
    - Number of bedrooms
    - Number of bathrooms

(c1 * square foot) + (c2 * home age) + (c3 * #br) + (c4 * #bath) = selling price

Would end up with a similar equation for each listing, leaving you with quite a big linear system to solve!

This will become more clear when we dig into machine learning.

METIS

Everything you need to know about statistics*

** Well… not quite everything, but enough for this course at least*

METIS

Intro Stats

The Iris Data Set

- Introduced by British Statistician Ronald Fisher in 1936
- Consists of 50 samples of the three species of Iris and various features
- This data set is very commonly used to teach statistical classification

Intro Stats

The **mean** is the arithmetic average of a group of values, found by dividing the total of all values by the number of values.

The **median** is the middle value in a group of values, found by ordering the values from smallest to largest and locating the one that occurs in the middle. If the size of the group is even, it is found by averaging the middle two values.

The **mode** is the value that occurs most often in a group of values, and is found by counting the frequency of every distinct value in the group and outputting the one that occurs most frequently.

Intro Stats

Mean of each column:
`iris_data_final.mean(axis=0)) #must pass in an axis`

Median of each column: `np.median(iris_data_final,axis=0))`

Mode of each column: `stats.mode(iris_data_final))`
`#implied axis=0 for this function`

Intro Stats: Variance

The **variance** of a set of values, σ^2, is the average of the square of the difference of the values in the dataset from the dataset's mean.

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

```
iris_data_final.var(axis=0)
```

Variance Example

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

array = [600, 470, 170, 430, 300]

mean = (600, 470, 170, 430, 300) / 5 = 394

$$\frac{(600 - \text{mean})^{**}2 + (470 - \text{mean})^{**}2 + \ldots + (300 - \text{mean})^{**}2}{5} = 21704$$

METIS

Intro Stats: Std. Deviation

The standard deviation, σ, is the square root of the variance:

We use the **standard deviation** much more regularly than the **variance** because it is on the same scale as the original data:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

So, in our last example it would be

sqrt(21704)

```
iris_data_final.std(axis=0)
```

METIS

Anscombe's Quartet

AKA The Dangers of Relying on Stats for Description...

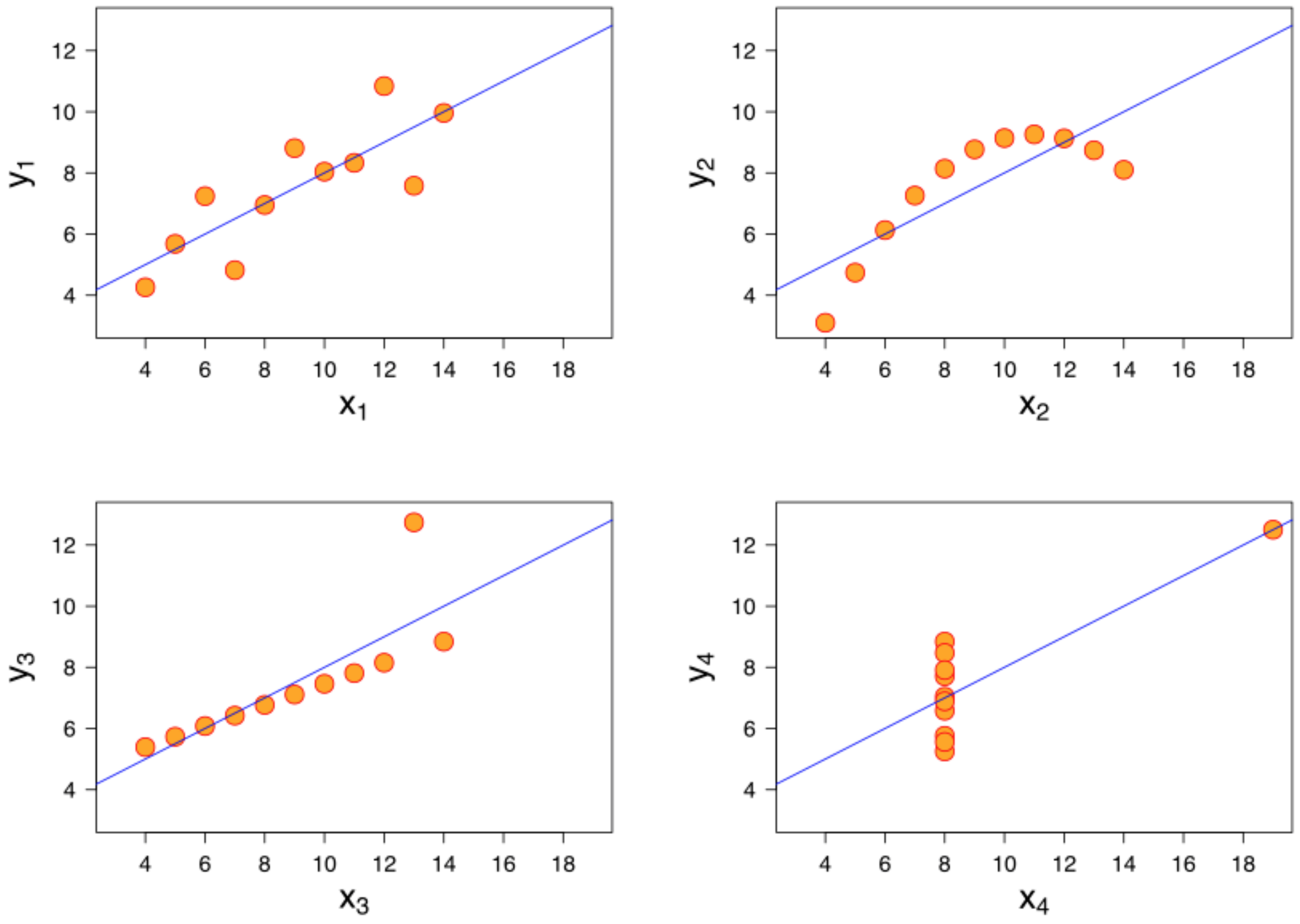
Anscombe's quartet

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

METIS

Anscombe's Quartet

AKA The Dangers of Relying on Stats for Description...



Intro Stats: Covariance

Covariance, like the variance, is a measure of spread, however it also measures how closely two datasets track each other.

- **Covariance** is a squared quantity, so it is not on the same scale as the mean

- **Covariance** of different pairs of variables can have completely different scales

$$Cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{n}$$

Intro Stats: Covariance

```
iris_data_final_column_cov = np.cov(iris_data_final.T)

[[ 0.68569351 -0.03926846  1.27368233  0.5169038 ]
 [-0.03926846  0.18800403 -0.32171275 -0.11798121]
 [ 1.27368233 -0.32171275  3.11317942  1.29638747]
 [ 0.5169038  -0.11798121  1.29638747  0.58241432]]
```

Exercise

- Compute the row-based covariance matrix of `iris_data_final`. What does this matrix measure?

Intro Stats: Correlation

Correlation is sort of like **standard deviation** generalized to pairs of datasets.

Really, the **correlation** is a **covariance** scaled by each dataset's **standard deviation**, so that it can only take on values from -1 to +1. This allows us to compare two pairs of variables and quickly tell if one pair of datasets is more related than an other .

- A **positive correlation** indicates the sets of values change together.

- A **negative correlation** indicates that the variables change in opposite directions.

METIS

Intro Stats: Correlation

$$r(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

METIS

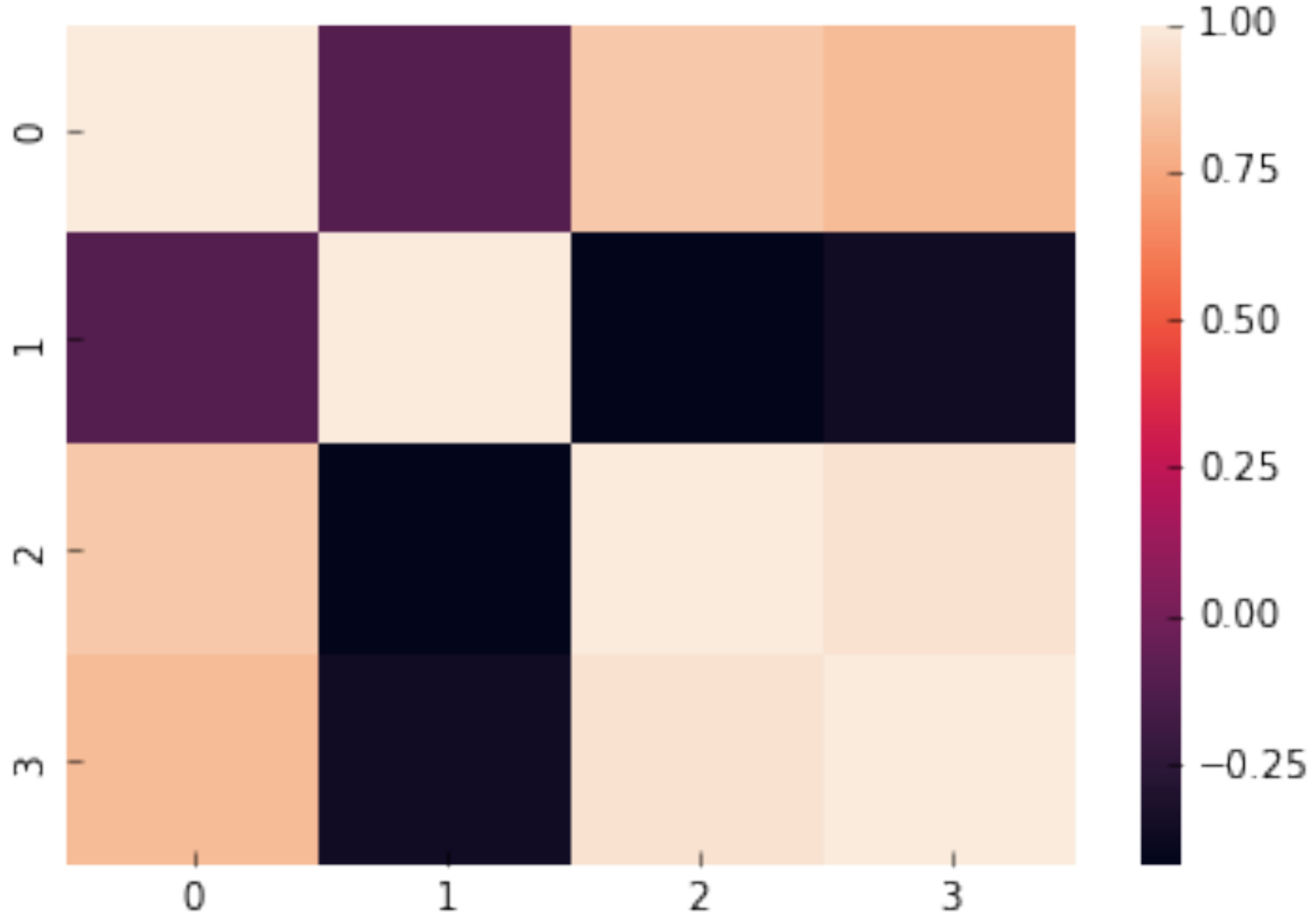
Intro Stats:
Correlation

```
iris_data_final_column_corr = np.corrcoef(iris_data_final.T)

[[ 1.         -0.10936925  0.87175416  0.81795363]
 [-0.10936925  1.         -0.4205161  -0.35654409]
 [ 0.87175416 -0.4205161   1.          0.9627571 ]
 [ 0.81795363 -0.35654409  0.9627571   1.        ]]
```

Intro Stats: Correlation

```
sns.heatmap(iris_data_final_column_corr)
```



METIS

Exercise

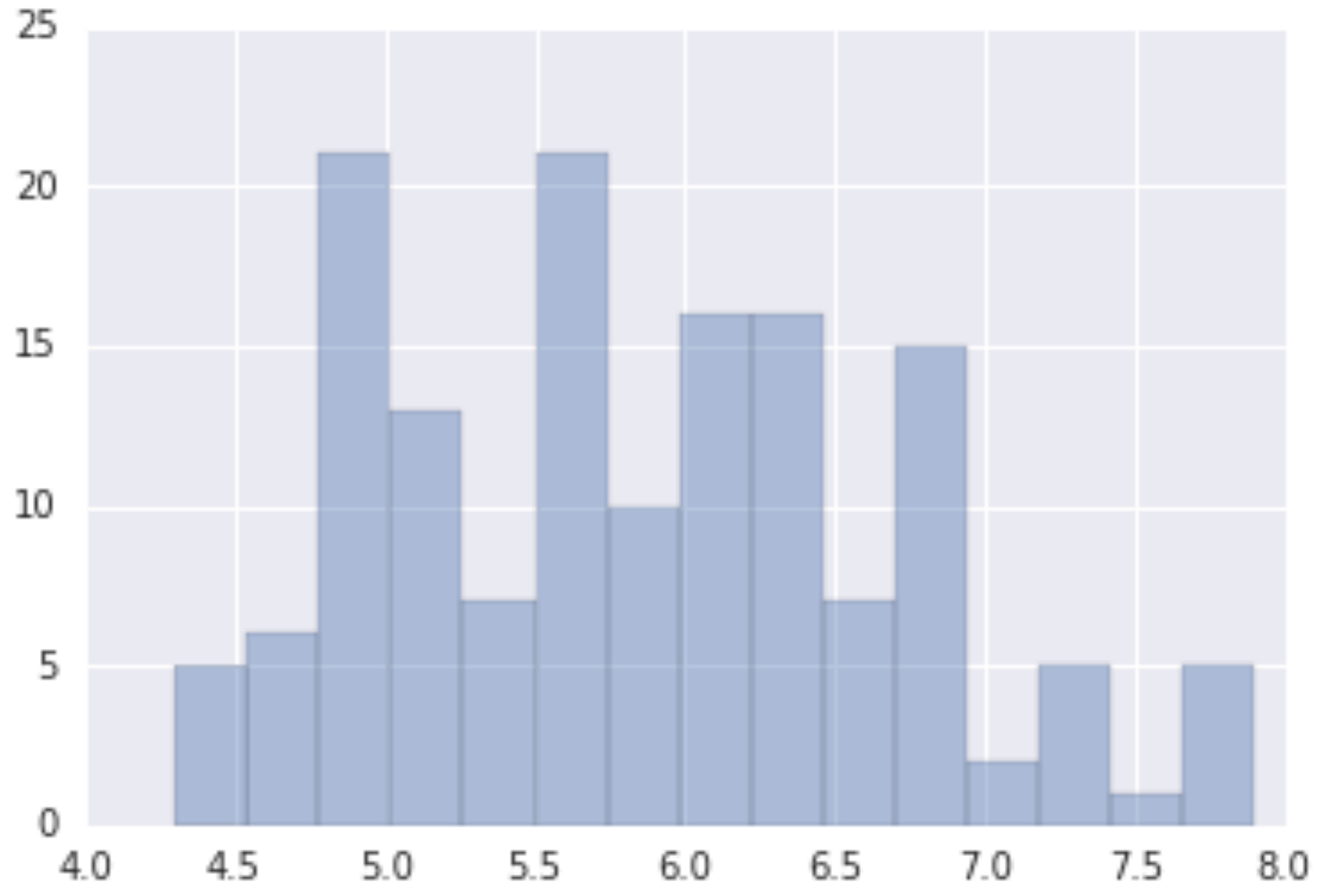
- Compute the row-based correlation matrix for `iris_data_final`.
- Visualize the correlation matrix as a heatmap. Notice anything?

Using the vertebral_data in `vertebral_values_final`:

- Compute the mean of each column.
- Compute the median of each column.
- Compute the mode of each column.
- What would it mean if the mean and the median for a given column are very far apart?
- What is useful about knowing the mode?
- Compute the variance and standard deviation of each column.
- Generate the columnar covariance and correlation matrices for this matrix.
- Do any columns appear to change together (based on their covariances/correlations)?
- What conclusions can we draw from these column-based statistics?
- If we had computed all of the row-based statistics here, what would their interpretation be?

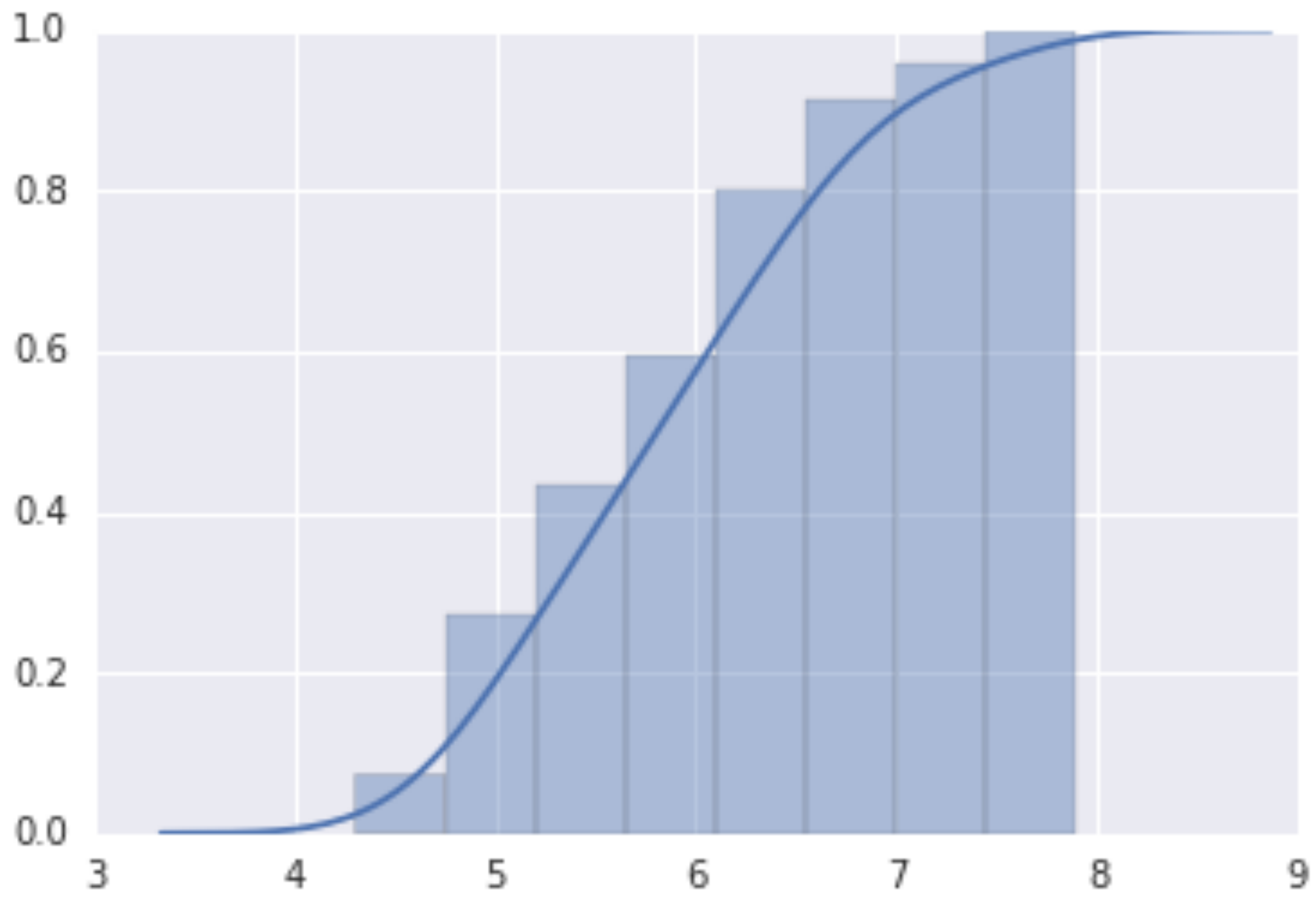
Intro Stats: Histograms

```
sns.distplot(iris_data_final[:,0],kde=False,bins=15)
```



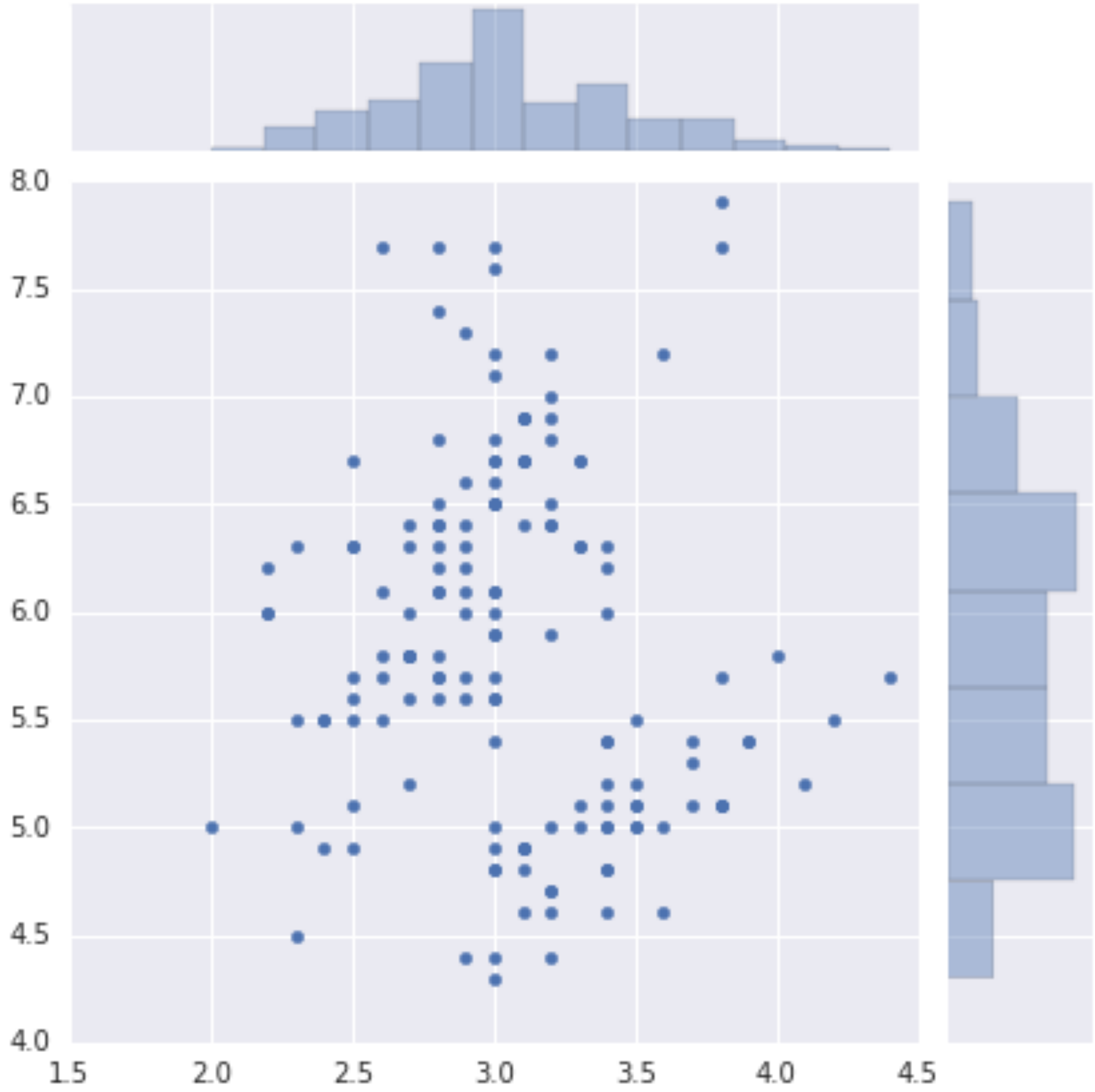
Intro Stats: Histograms

```
sns.distplot(iris_data_final[:,0],
hist_kws={"cumulative":True},kde_kws={"cumulative":True})
```



Intro Stats: Scatterplot

```
sns.jointplot(iris_data_final[:,1], iris_data_final[:,0],stat_func=None)
```



METIS

Exercise

Using the vertebral_values_final dataset:

- Compute the column-wise correlation matrix
- Visualize the scatter plot for two columns that are positively correlated
- Visualize the scatter plot for two columns that are negatively correlated (anti-correlated)

Intro Stats:
Random Variables and Distributions

```
n_trials = 10
prob_heads = 0.5
num_heads = np.random.binomial(n_trials, prob_heads)

print("Num heads:",num_heads)
print("Fraction heads:”,float(num_heads)/n_trials)

Num heads: 6
Fraction heads: 0.6
```

Intro Stats:
Continuous Random Variables

```
n_trials = 10
prob_heads = 0.5
num_heads = np.random.binomial(n_trials, prob_heads)

print("Num heads:",num_heads)
print("Fraction heads:”,float(num_heads)/n_trials)

Num heads: 6
Fraction heads: 0.6
```

Exercise

- Generate 100 coin flips, what fraction of them come up heads?
- Generate 1,000 coin flips, what fraction of them come up heads?
- Generate 100,000 coin flips, what fraction of them come up heads? Notice a pattern?
- Generate 100 bed making experiments, whats the average bed making time?
- Generate 1,000 bed making experiments, whats the average bed making time?
- Generate 100,000 bed making experiments, whats the average bed making time? Notice a pattern?

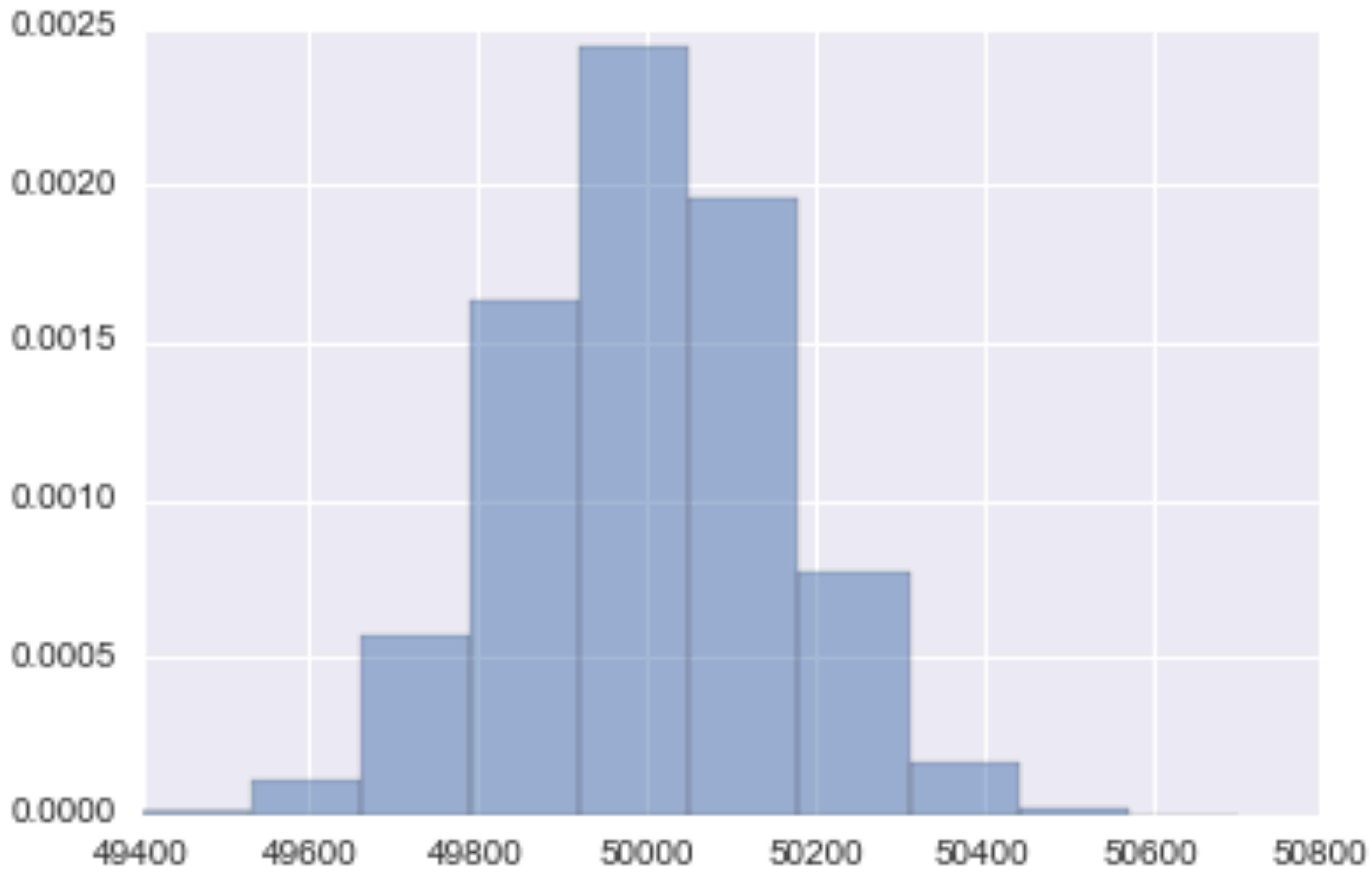
Intro Stats:
Distributions: Binomial Distribution

The **binomial distribution** is a distribution over discrete values. It has two parameters, n and p, and is the discrete probability distribution of the number of successes in a sequence of n independent yes/no experiments, each of which yields success (yes) with probability p.

```
n_trials_binomial=10000
binomial_data = np.random.binomial(n_trials,
prob_heads,n_trials_binomial)
plt.hist(binomial_data,normed=True,alpha=0.5)
```

Intro Stats:
Distributions: Binomial Distribution

```
n_trials_binomial=10000
binomial_data = np.random.binomial(n_trials,
prob_heads,n_trials_binomial)
plt.hist(binomial_data,normed=True,alpha=0.5)
```



Intro Stats:
Distributions: Normal Distribution

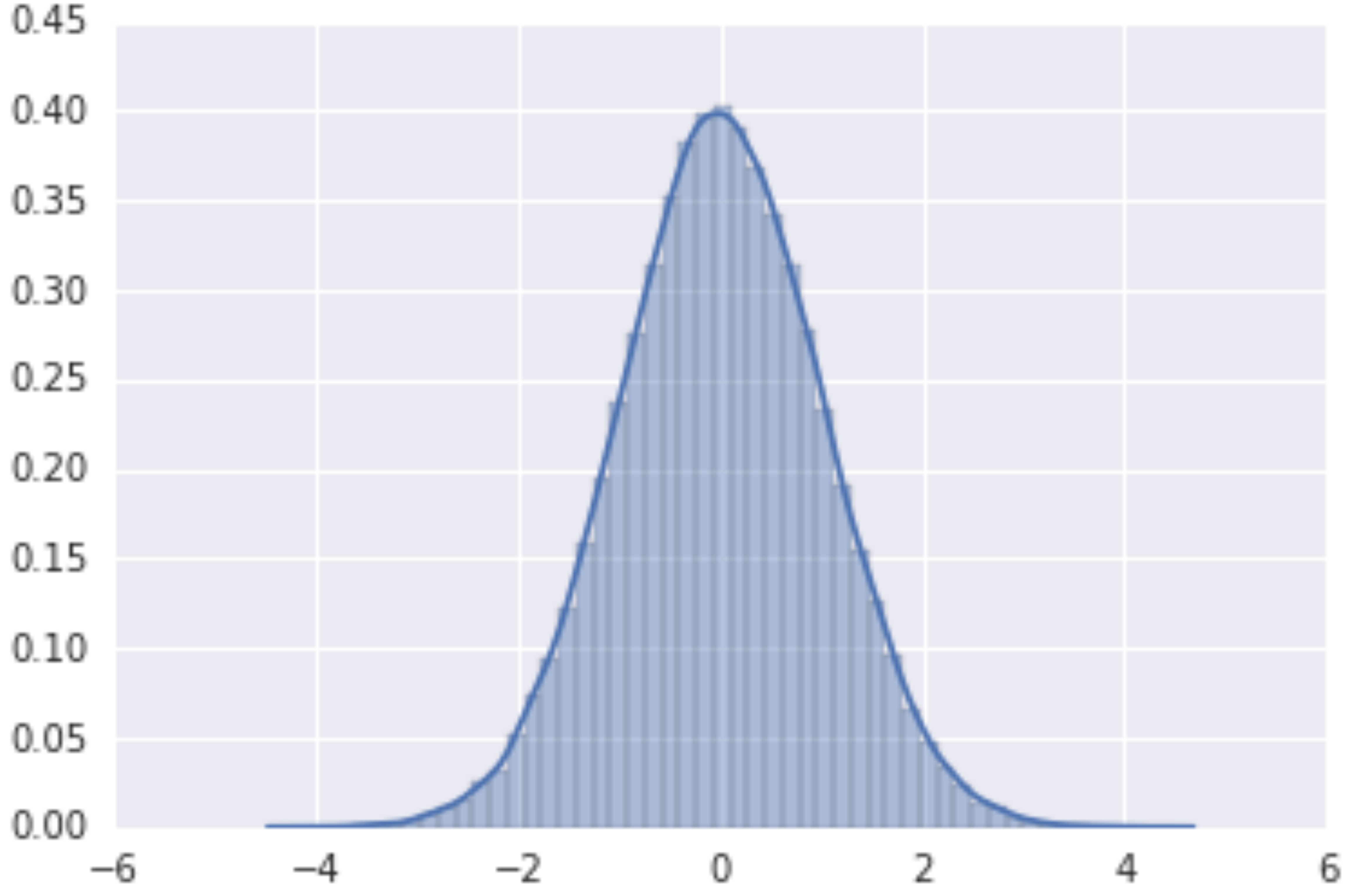
The normal distribution is the most important distribution in all of statistics. It is a continuous probability distribution, unlike the binomial, which is discrete.

It's super important because as datasets become larger and larger and larger, they tend to look more and more like the normal distribution.

The normal distribution is fully specified by 2 parameters, the mean (μ) and the standard deviation (σ).

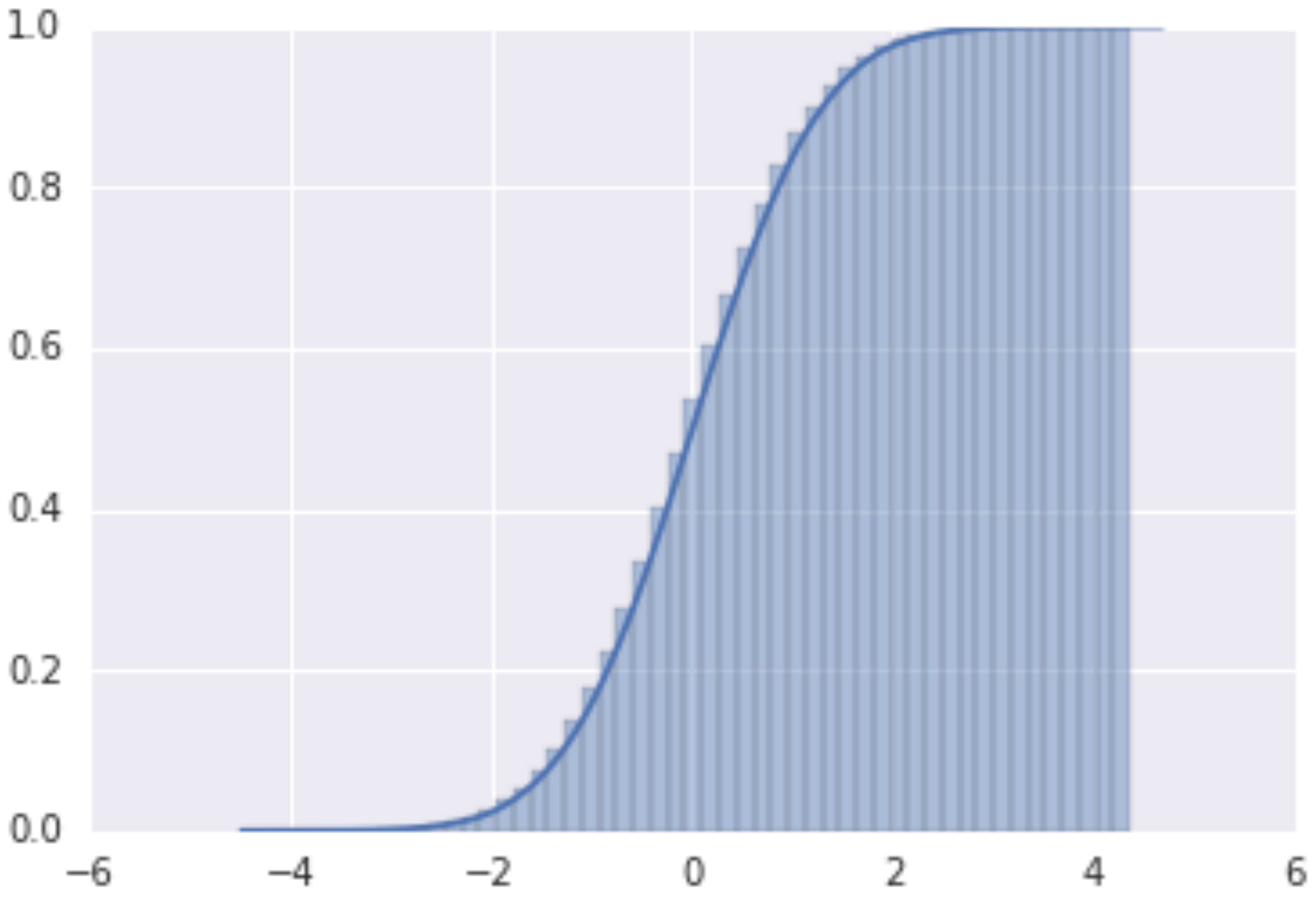
Intro Stats: Distributions: Normal Distribution

```
mu = 0
sigma = 1
n_samples = 100000
normal_data = np.random.normal(mu, sigma, n_samples)
sns.distplot(normal_data)
```



Intro Stats: Distributions: Normal Distribution

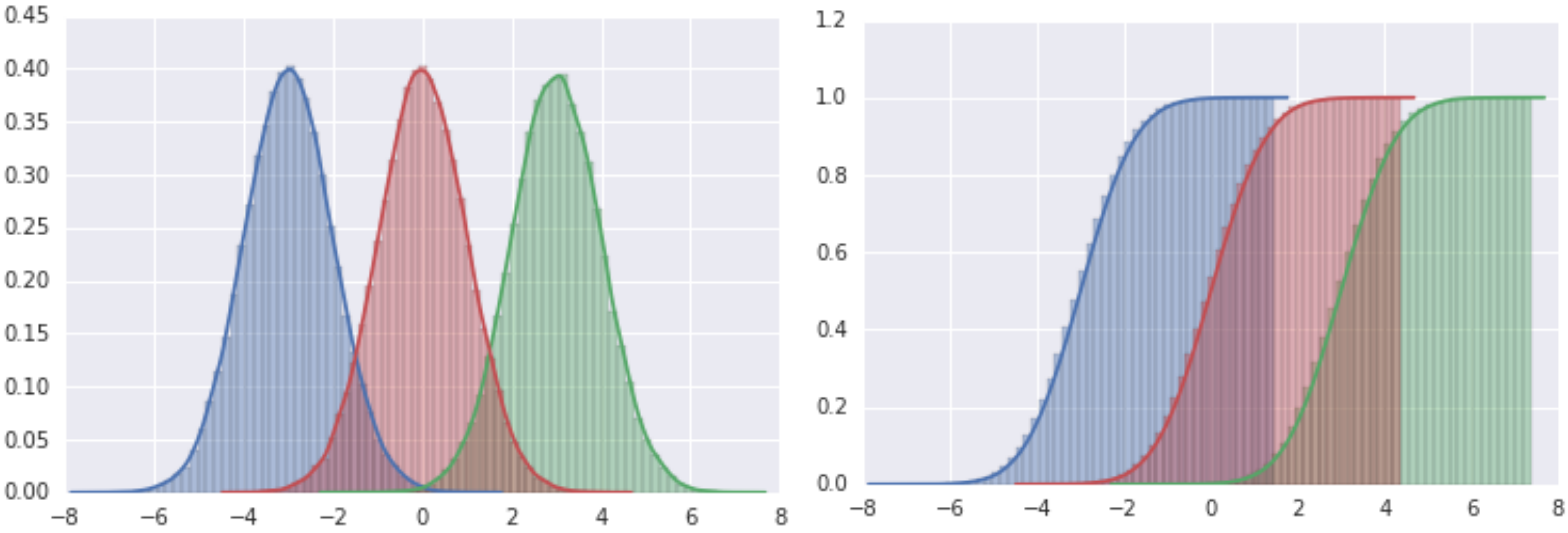
```
sns.distplot(normal_data,
hist_kws={"cumulative":True},kde_kws={"cumulative":True})
```



METIS

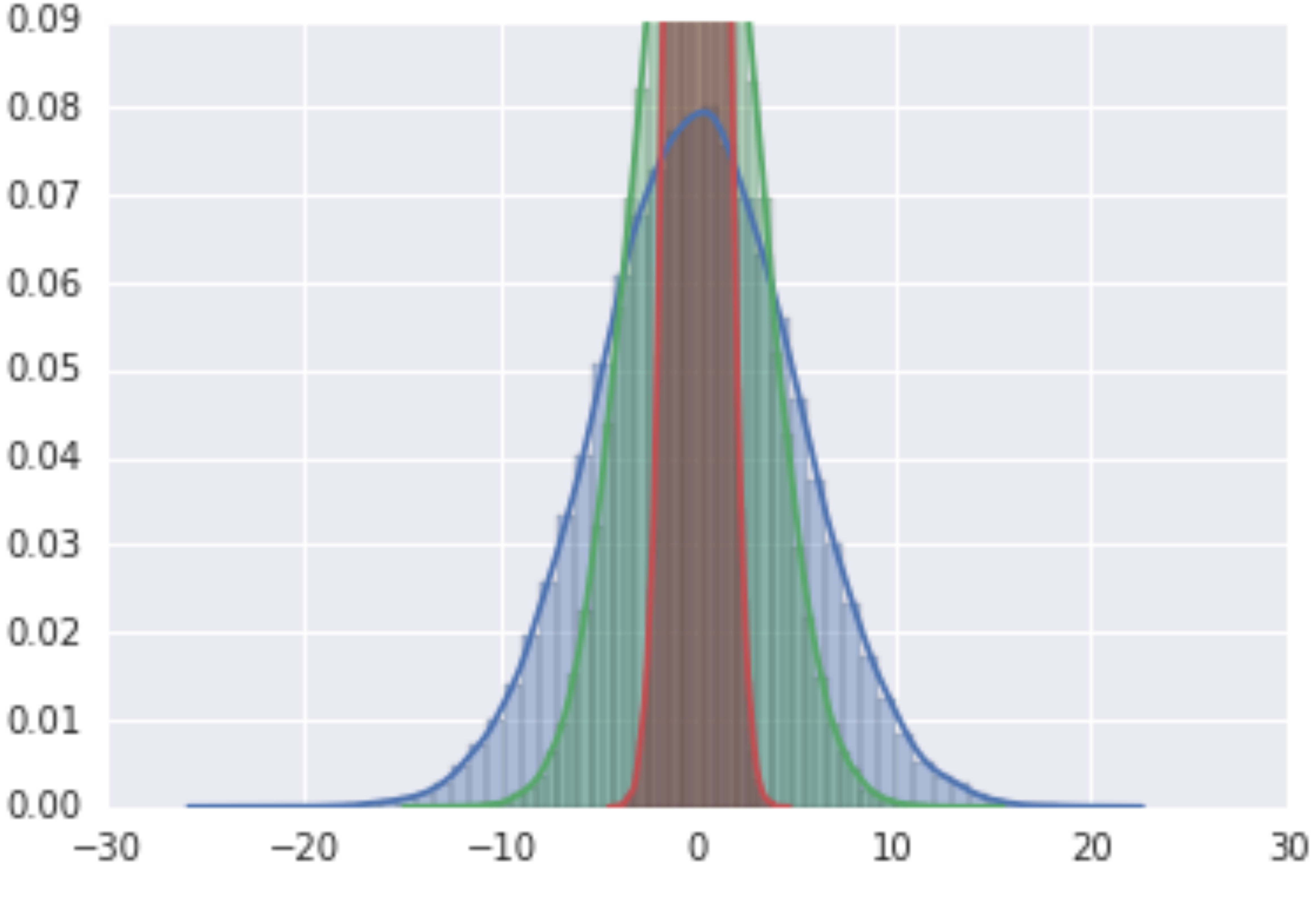
Intro Stats:
Distributions: Normal Distribution

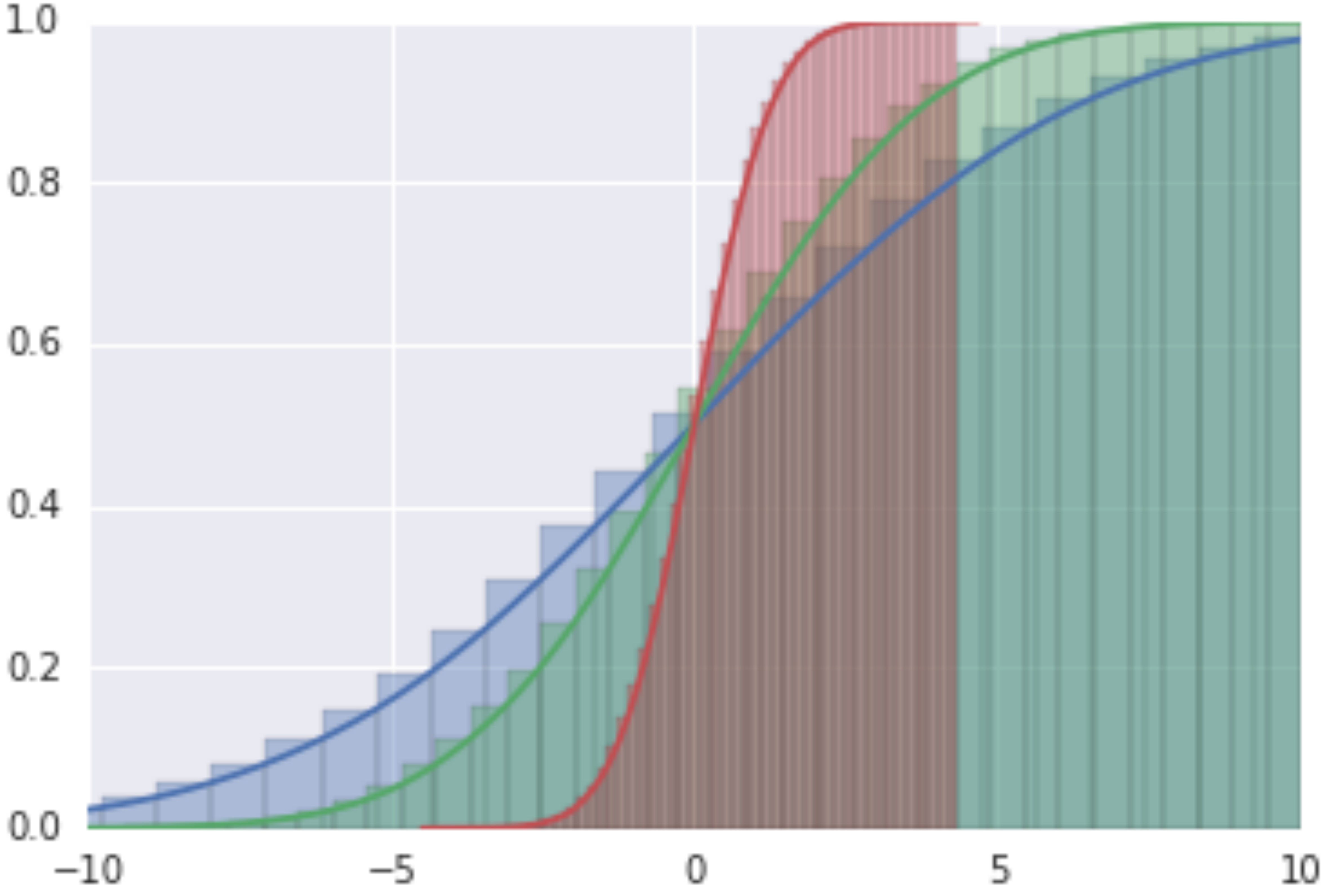
Altering the mean changes the location



Intro Stats:
Distributions: Normal Distribution

Altering the std. deviation changes the width of the curve.





Intro Stats:
Distributions: Exponential Distribution

The **exponential distribution** is another very common distribution. In stats-speak, it is the probability distribution that describes the time between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate.

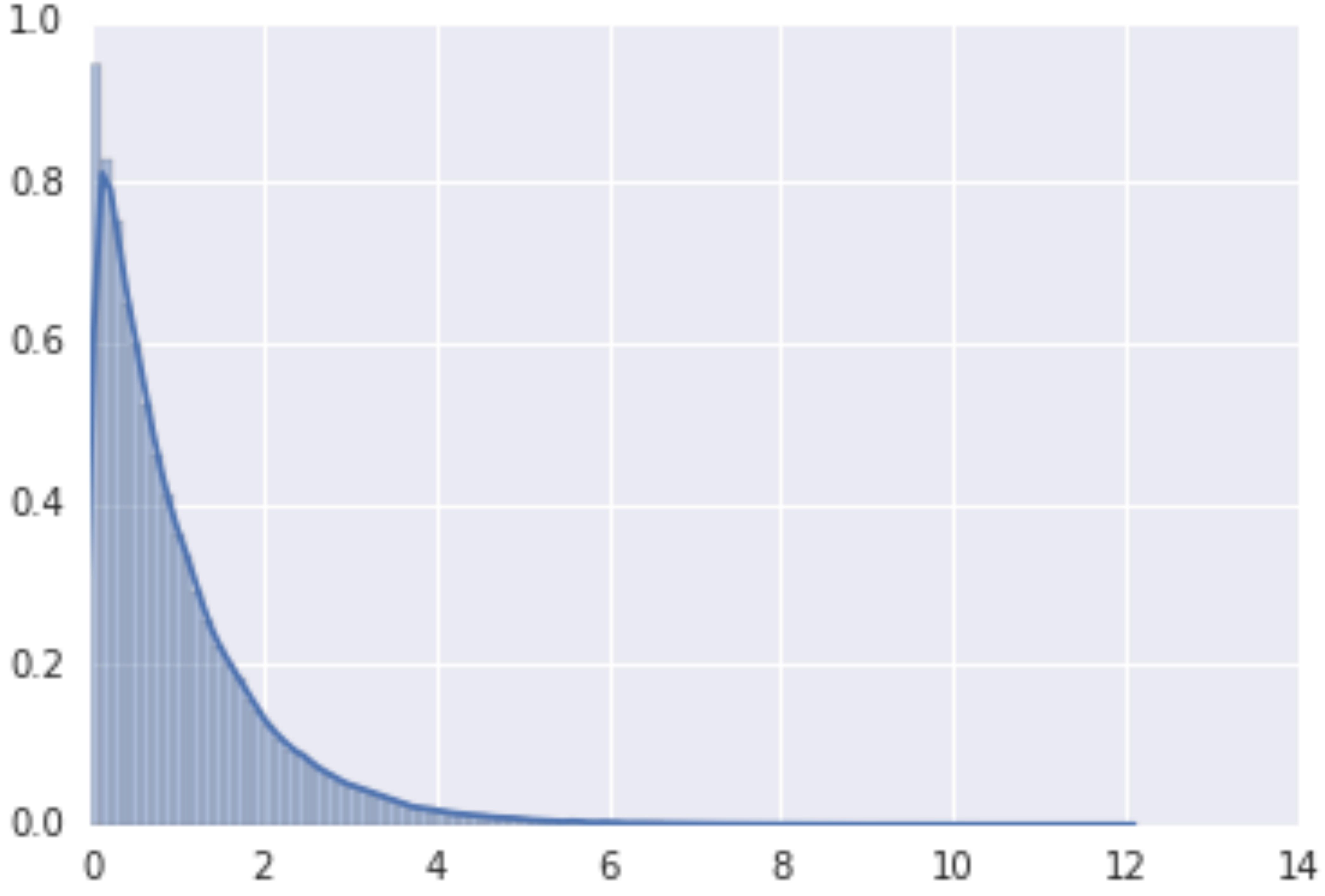
An example exponential distribution would describe the time between airplanes landing and taking off at LaGuardia airport during hours that the airport is operating.

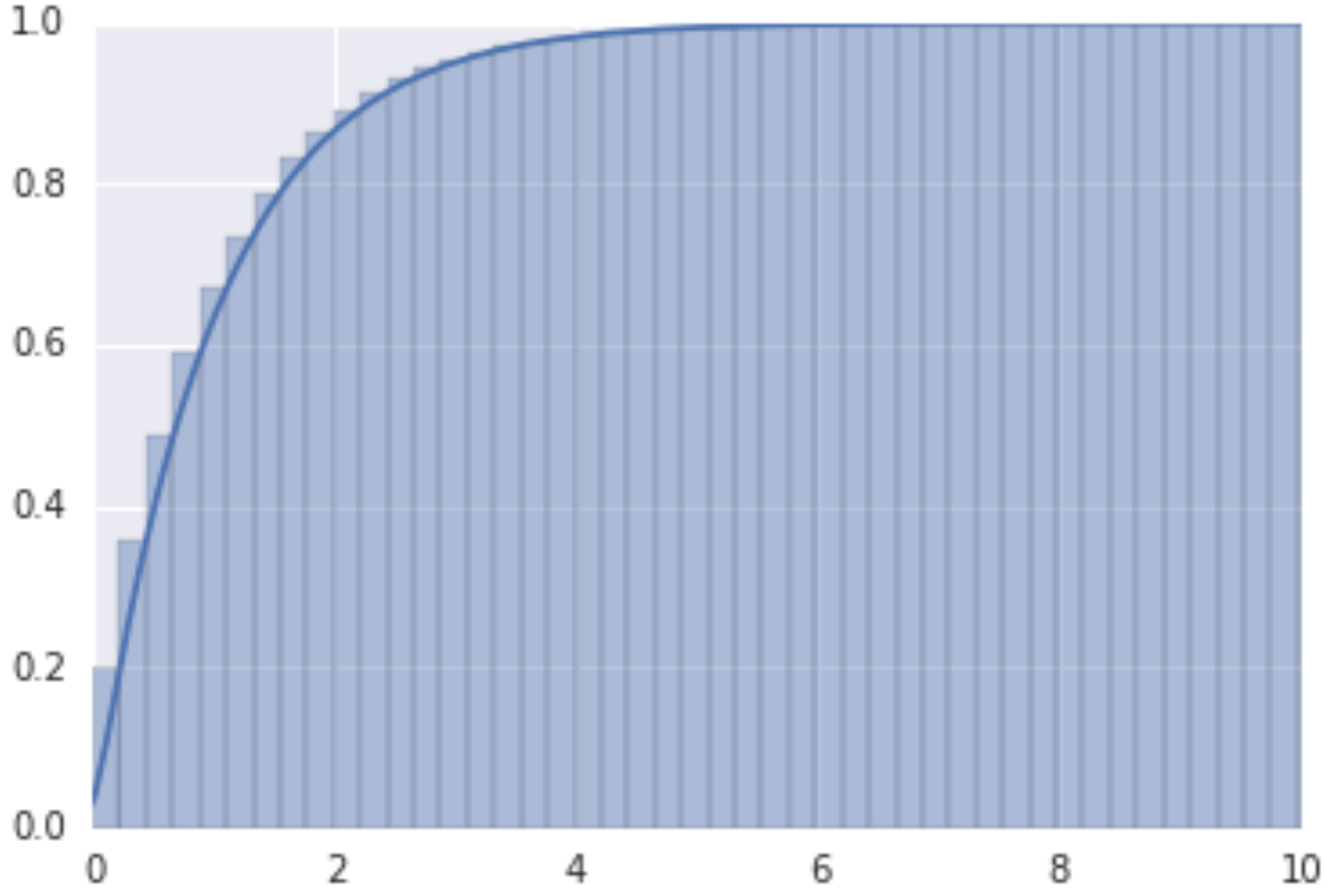
The exponential distribution is interesing because it has the property known as being **memoryless**. This means that knowing the history of the distribution has no effect on being able to predict what will happen next.

METIS

Intro Stats: Distributions: Exponential Distribution

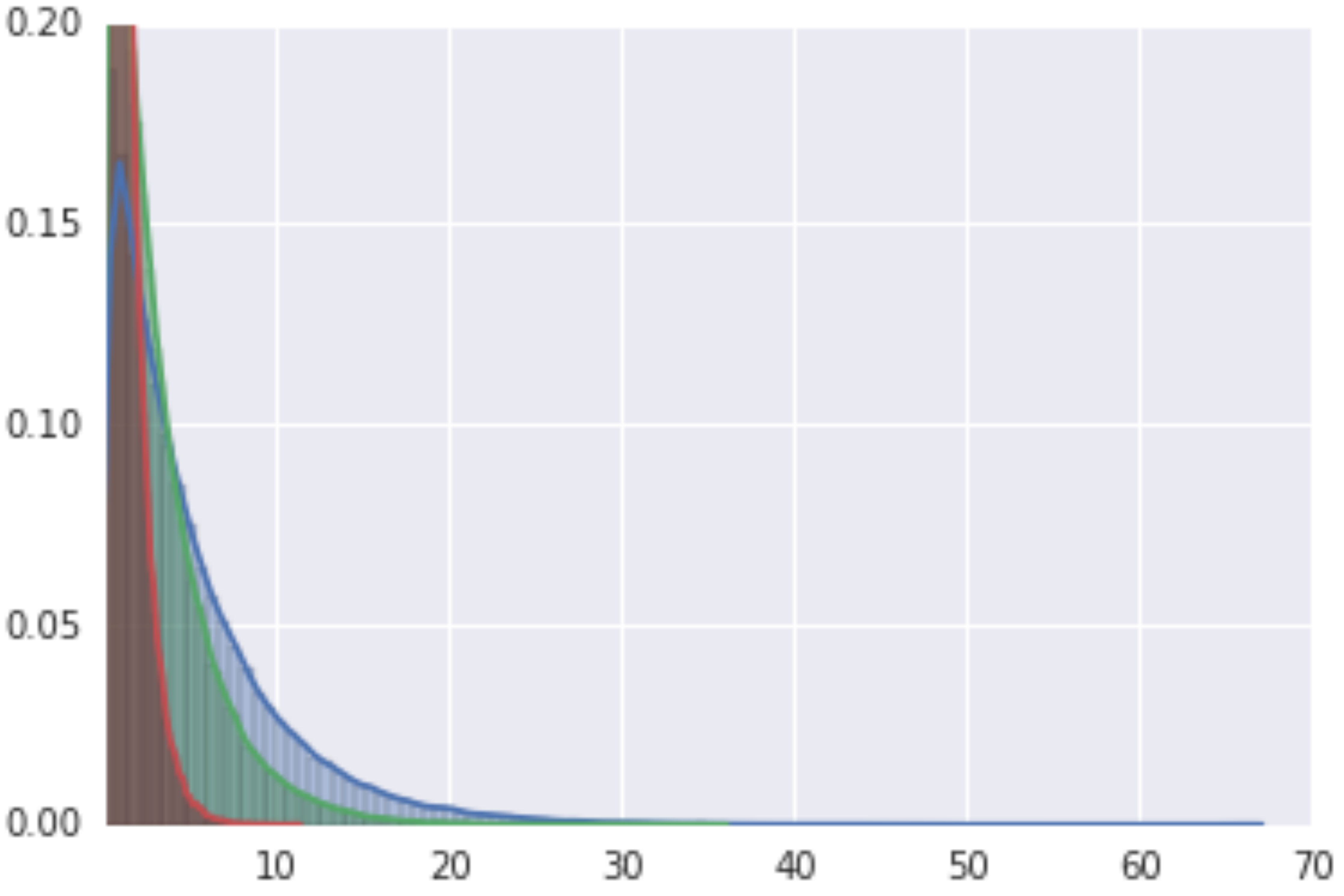
```
e = np.random.exponential(size=n_samples)
sns.distplot(e,bins=100)
plt.xlim(0,)
```

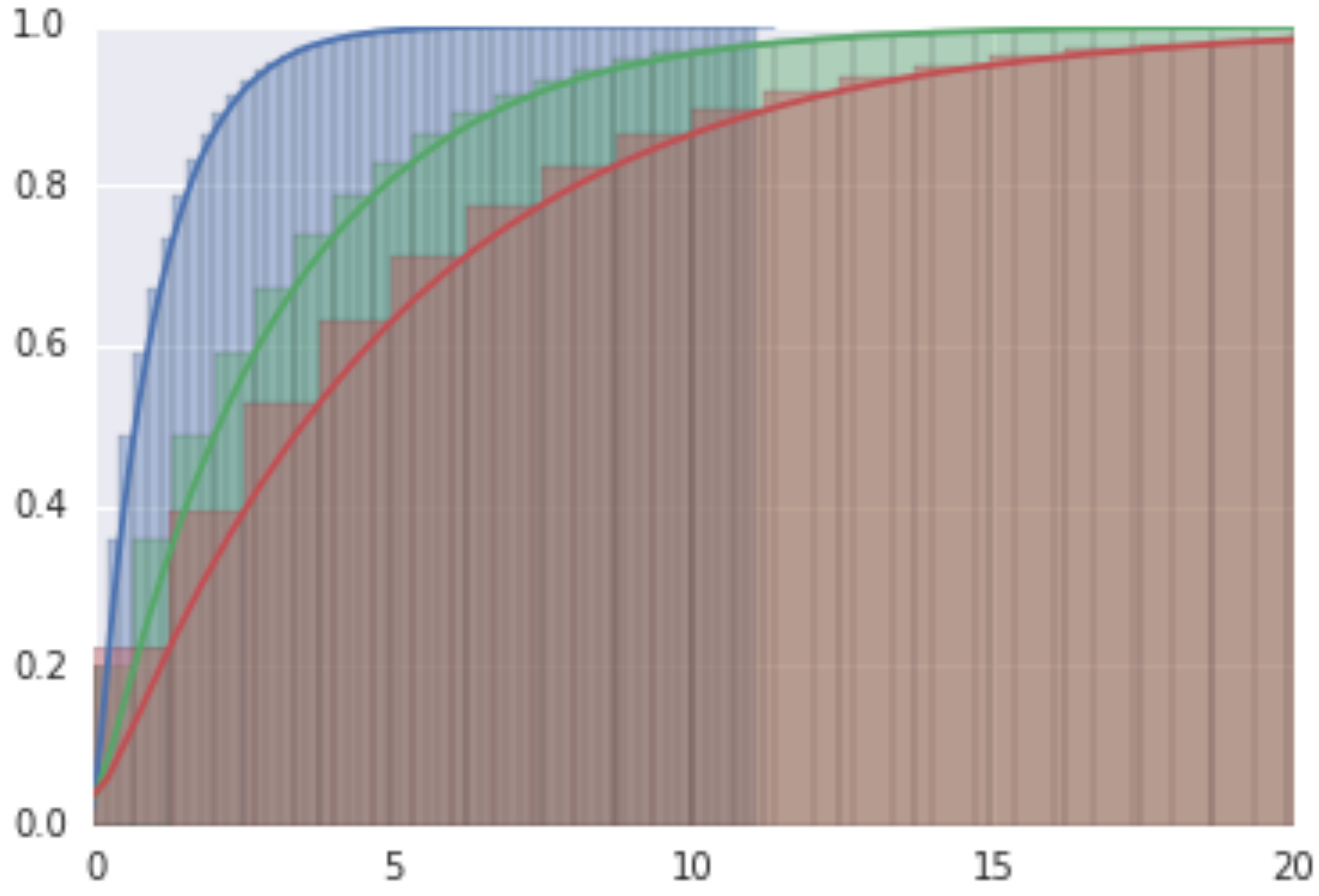




Intro Stats: Distributions: Exponential Distribution

Altering the beta (exponent) changes the location





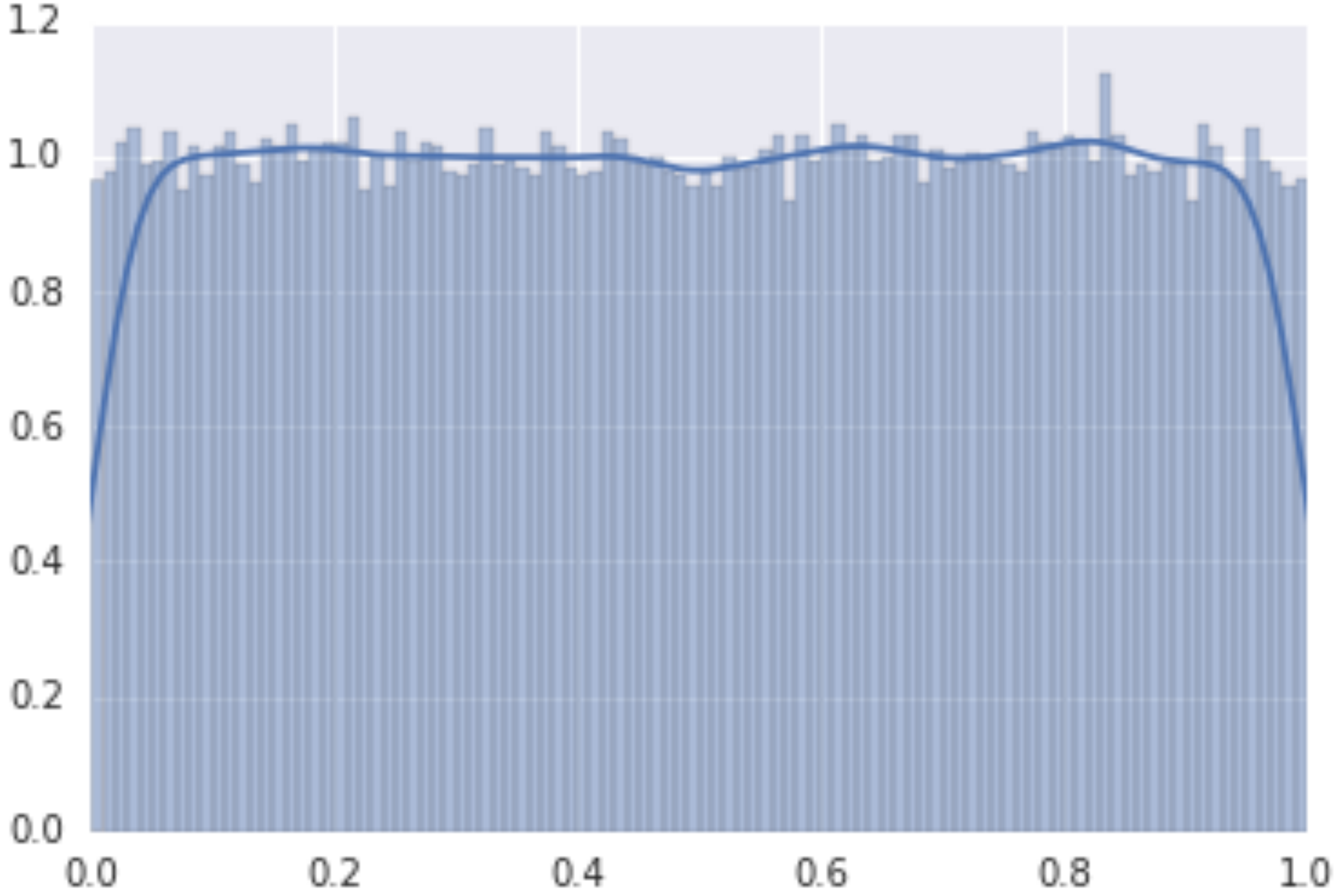
Intro Stats: Distributions: Uniform Distribution

The final distribution we will talk about is the **uniform distribution**. It simply describes cases where all values have the exact same frequency. It is a useful distribution because it is used for unbiased sampling.

```
uni = np.random.uniform(size=n_samples)
sns.distplot(uni, hist=True,bins=100)
plt.xlim(0,1)
```

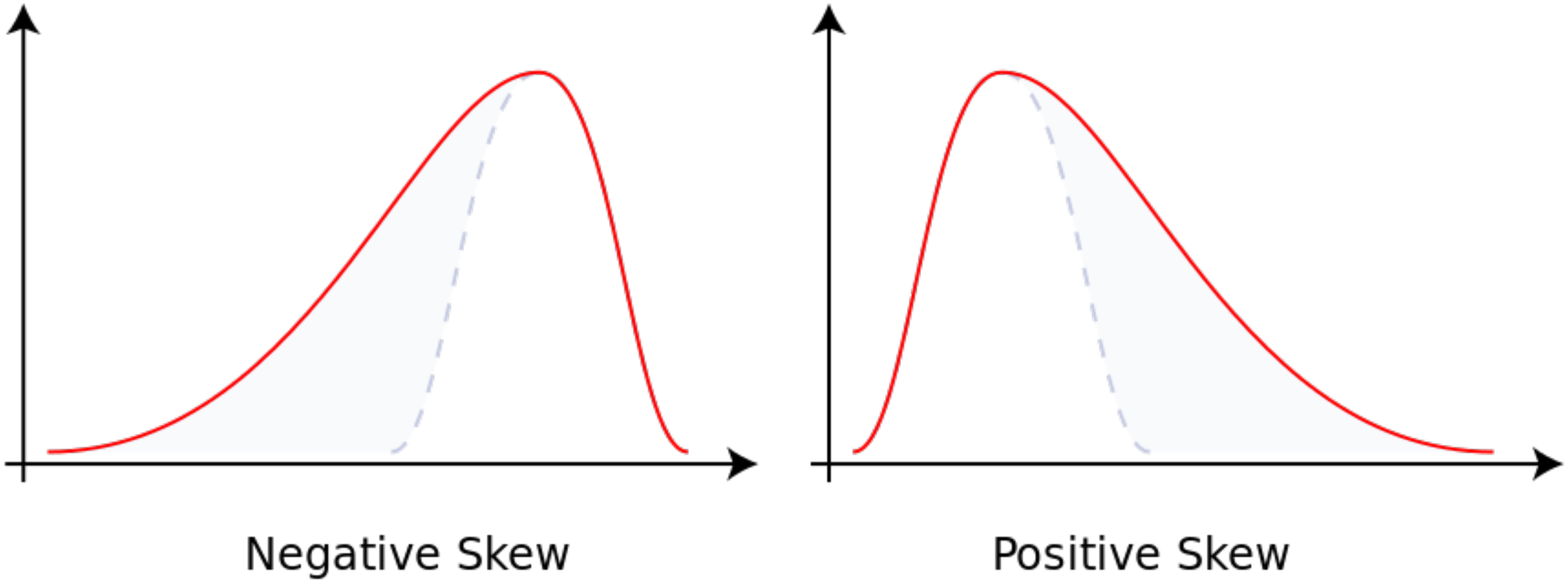
Intro Stats: Distributions: Uniform Distribution

```
uni = np.random.uniform(size=n_samples)
sns.distplot(uni, hist=True,bins=100)
plt.xlim(0,1)
```



Intro Stats: Skewness

```
uni = np.random.uniform(size=n_samples)
sns.distplot(uni, hist=True,bins=100)
plt.xlim(0,1)
```



Intro Stats: Kurtosis

Krutosis is a measure of the "peakedness" of a given probability distribution and can only be measured for continuous distributions.