

COMS BC1016

Introduction to Computational Thinking and Data Science

# Lecture 6: Charts and Histograms

Sept 30, 2025

Copyright © 2025 Barnard College

Sept 22, 2025



# Reminders

- Questions about the assignments? Please post on the EdStem!
- Clarification questions are helpful for everyone :)
- If it's specific to your submission, you can post to instructors-only
- HW 1 is due on Wednesday
- HW 2 is out today, due next week Wednesday

# Lecture Outline

- Table Review
  - Manipulating Tables
  - Operating on Tables
  - Functions
- Line Plots and Scatter Plots
- Bar Charts and Histograms

# Table Review

# Prof Lee's Cat Census

Professor Lee is in a cat picture group chat. She has collected data on the cats shared in this chat:

Name	Age	Weight	Coloring	Sex	Owner
Ruby	14	8	tuxedo	F	Alice
Gertrude	15	12	tuxedo	F	Alice
Hamby	8	16	tabby	M	Bob
Fig	3	7	tabby	F	Bob
Corina	6	10	tortie	F	Carol
Frito	2	8.5	tabby	M	Carol

# Manipulating Tables

Keep or remove by columns:

- `tbl.select(c1, c2, ...)`
- `tbl.drop(c1, c2, ...)`

Keep or remove by row:

- `tbl.where(c, predicate)`
- `tbl.take(row_indices)`

Reorder table entries:

- `tbl.sort(c[, descending=False])`

# Operating on Tables

Organize table entries by values in column `c`:

- `tbl.group(c)`
- `tbl.group(c, func)`

Apply a function `func` to all entries in a column `c`:

- `tbl.apply(func, c)`

# Anatomy of a Function

Name, Parameters, Body, Return Statement

Example:

```
def convert_to_figs(weight):  
    new_weight = (weight/fig_weight).round(1)  
    return new_weight
```



# Anatomy of a Function

Name, Parameters, Body, Return Statement

Example:

```
def convert_to_figs(weight):  
    new_weight = (weight / fig_weight).round(1)  
    return new_weight
```


The diagram illustrates the mapping of function components to the example code. Arrows connect the labels 'Name', 'Parameters', 'Body', and 'Return Statement' to their respective parts in the code: 'Name' points to 'convert\_to\_figs', 'Parameters' points to '(weight)', 'Body' points to the assignment line, and 'Return Statement' points to the 'return' line.

# apply

Use **apply** to call a function on each element in a column

```
def convert_to_figs(weight):  
    new_weight = (weight/fig_weight).round(1)  
    return new_weight
```

```
cat_tbl1.apply(convert_to_figs, 'Weight')
```



Returns an array with `convert_to_figs` called on each element in the `'Weight'` column

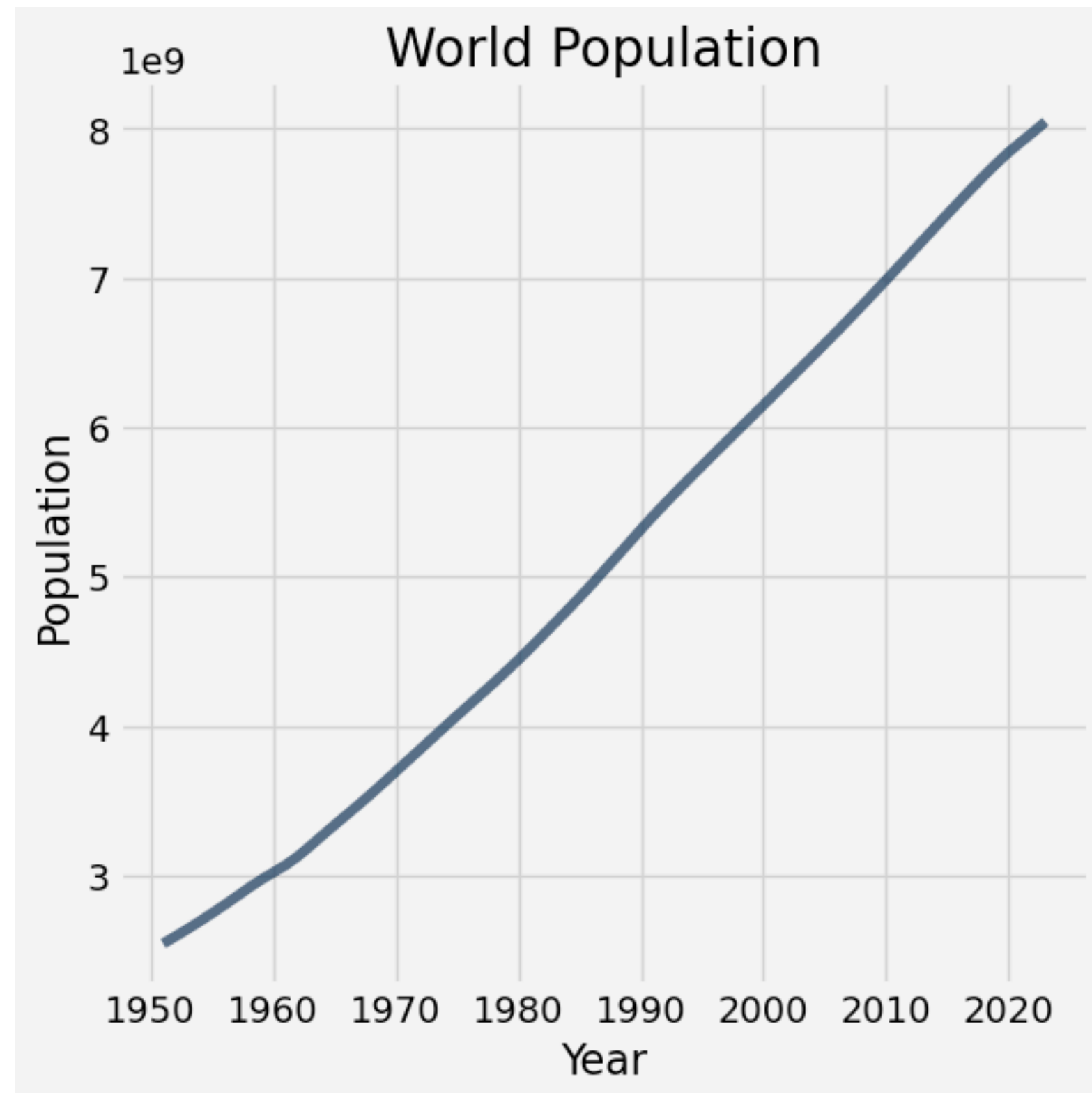
# Plotting Relations Between Numerical Values



# Line and Scatter Plots

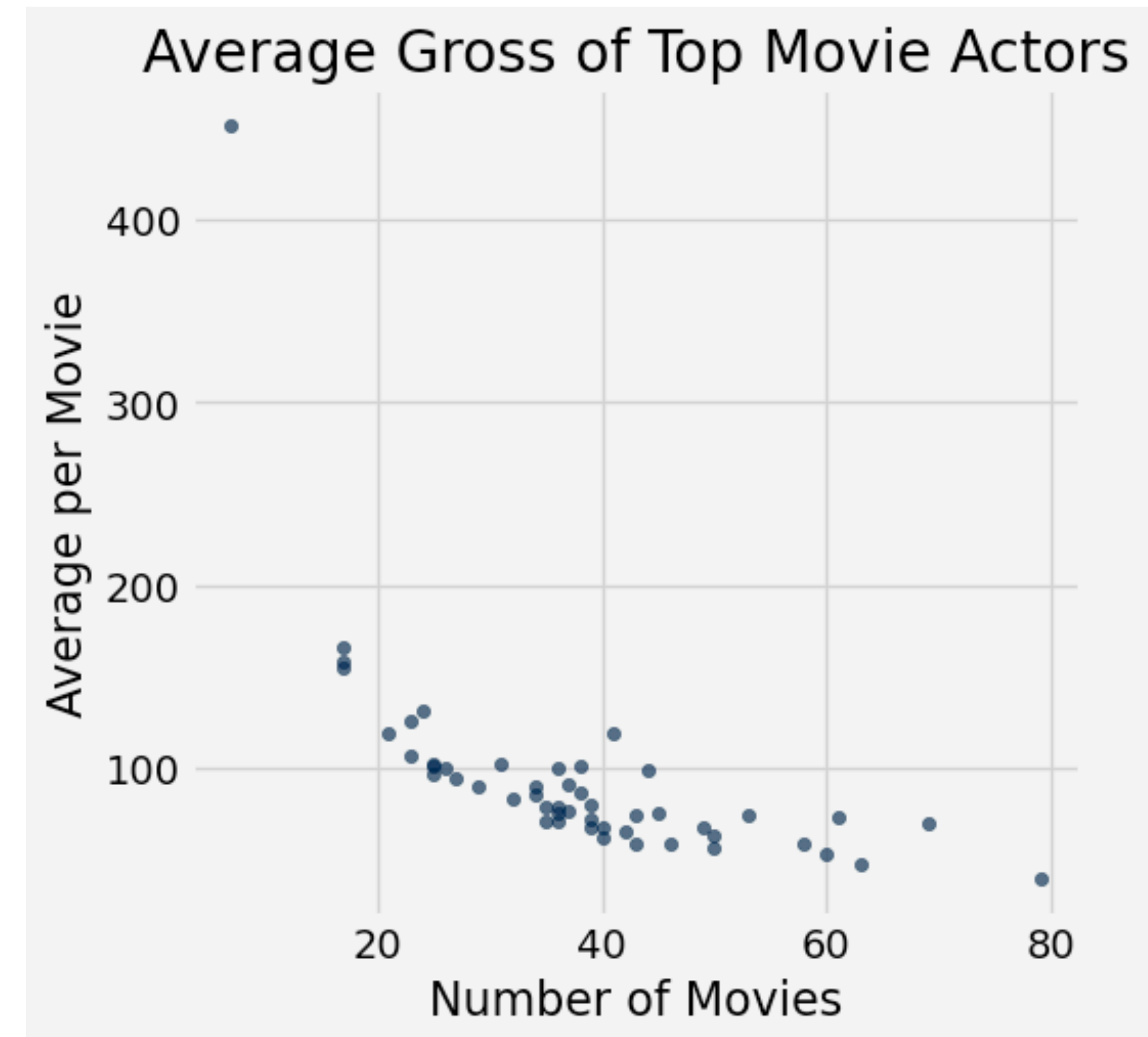
Line Plot

`plot`



Scatter Plot

`scatter`



# Line Plots

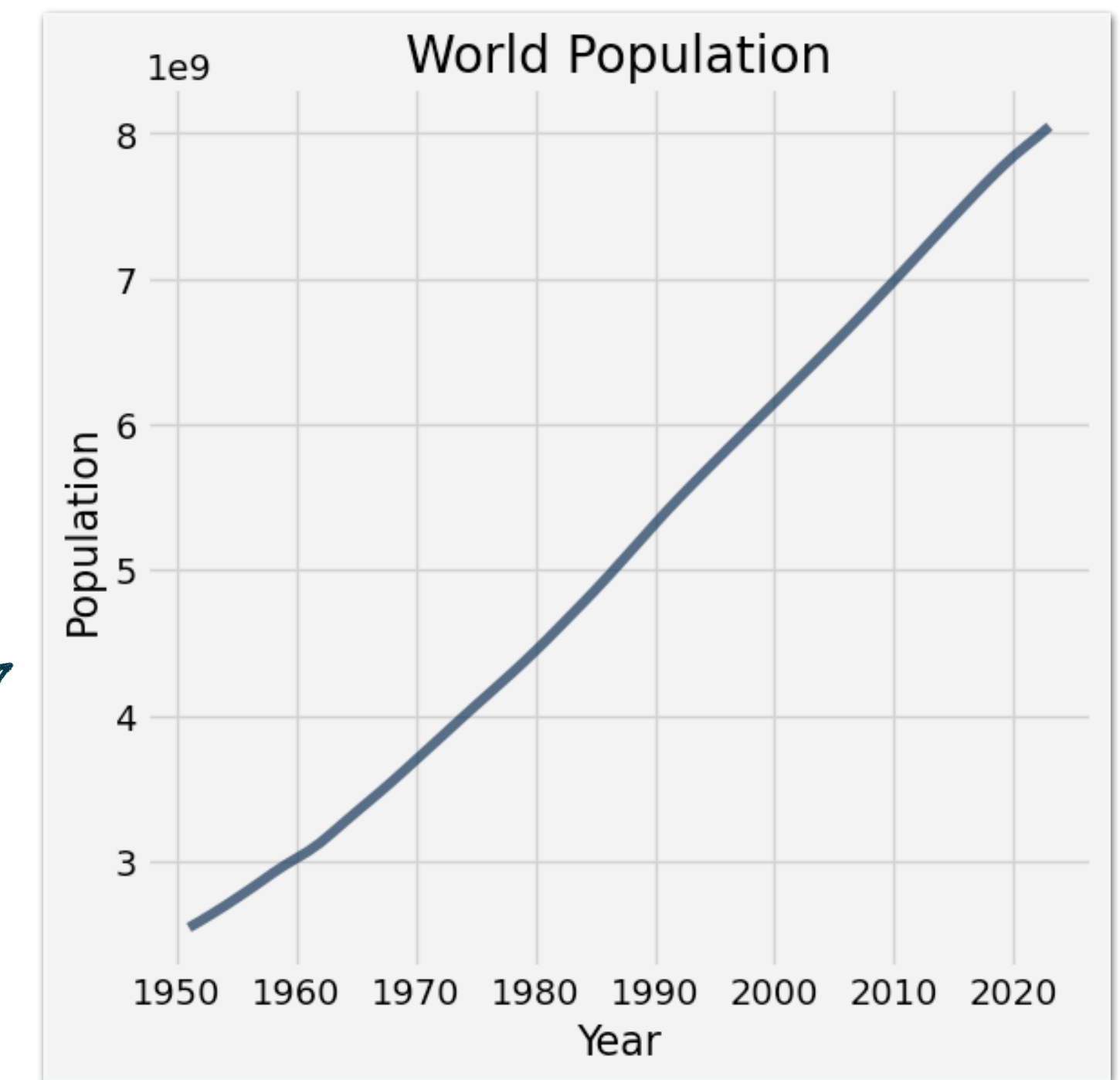
Line plots: good for sequential data if

- x-axis has an order (e.g., time, years, distance)
- sequential differences in y value are meaningful
- there's only one y-value for each x-value

Year	Population
1951	2.54313e+09
1952	2.59027e+09
1953	2.64028e+09
1954	2.69198e+09
1955	2.74607e+09
1956	2.801e+09
1957	2.85787e+09
1958	2.91611e+09
1959	2.97029e+09
1960	3.01923e+09
... (63 rows omitted)	

```
tbl.plot(x_axis, y_axis)
```

```
- pop_data.plot('Year', 'Population')
```



y-axis

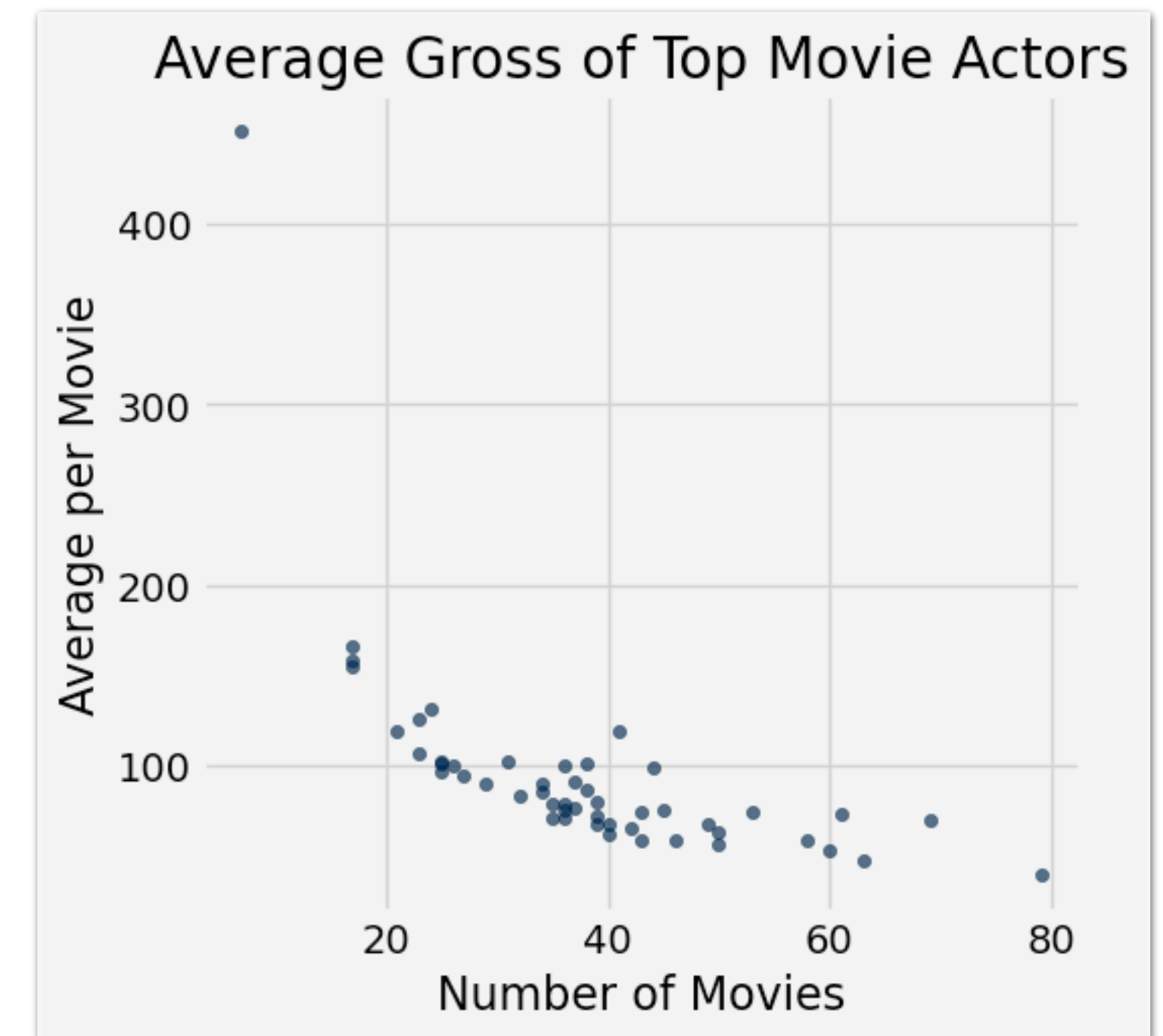
x-axis

# Scatter Plots

Scatter plots: good for non-sequential quantitative data

- Great for looking for associations

Actor	Total Gross	Number of Movies	Average per Movie	#1 Movie	Gross
Harrison Ford	4871.7	41	118.8	Star Wars: The Force Awakens	936.7
Samuel L. Jackson	4772.8	69	69.2	The Avengers	623.4
Morgan Freeman	4468.3	61	73.3	The Dark Knight	534.9
Tom Hanks	4340.8	44	98.7	Toy Story 3	415
Robert Downey, Jr.	3947.3	53	74.5	The Avengers	623.4
Eddie Murphy	3810.4	38	100.3	Shrek 2	441.2
Tom Cruise	3587.2	36	99.6	War of the Worlds	234.3
Johnny Depp	3368.6	45	74.9	Dead Man's Chest	423.3
Michael Caine	3351.5	58	57.8	The Dark Knight	534.9
Scarlett Johansson	3341.2	37	90.3	The Avengers	623.4
... (40 rows omitted)					



```
tbl.scatter(x_axis, y_axis)
```

```
- actor.scatter('Number of Movies', 'Average per Movie')
```



# Plot Notebook Demo

# Visualizing Categorical Data

# Recall: Types of Attributes

- Attributes are the names of columns in tables
- All values in a column should be the same type and comparable to each other
  - **Numerical** - Values are on a numerical scale (e.g., years)
    - Values are *ordered*
    - Differences are meaningful
  - **Categorical** - Each value is from a fixed inventory (e.g., material)
    - May not have an ordering
    - Categories are either the same or different



# Recall: Types of Attributes

- Attributes are the names of columns in tables
- All values in a column should be the same type and comparable to each other

- **Numerical** - Values are on a numerical scale (e.g., years)

- Values are *ordered*
  - Differences are meaningful

can visualize with  
a line or scatter plot

- **Categorical** - Each value is from a fixed inventory (e.g., material)

- May not have an ordering
  - Categories are either the same or different

# Is this a numerical or categorical attribute?

Track and field: The **distance a shotput** is thrown (in feet)

[77, 54, 63, 70]

# Is this a numerical or categorical attribute?

Gymnastics: **Events that gymnasts can perform in**

['floor exercise', 'pommel horse', 'rings', 'parallel bars']



# Is this a numerical or categorical attribute?

Swimming: **Race distances (in meters)**

[50, 100, 200, 400, 800]

# Terminology

- **Individuals** are rows in a column (instances of data)
- **Categorical Variable** are attributes (columns)
  - Each individual has exactly one value
  - Has a distribution
    - For each different value of a variable, there is a frequency of individuals that have that value

# Distribution

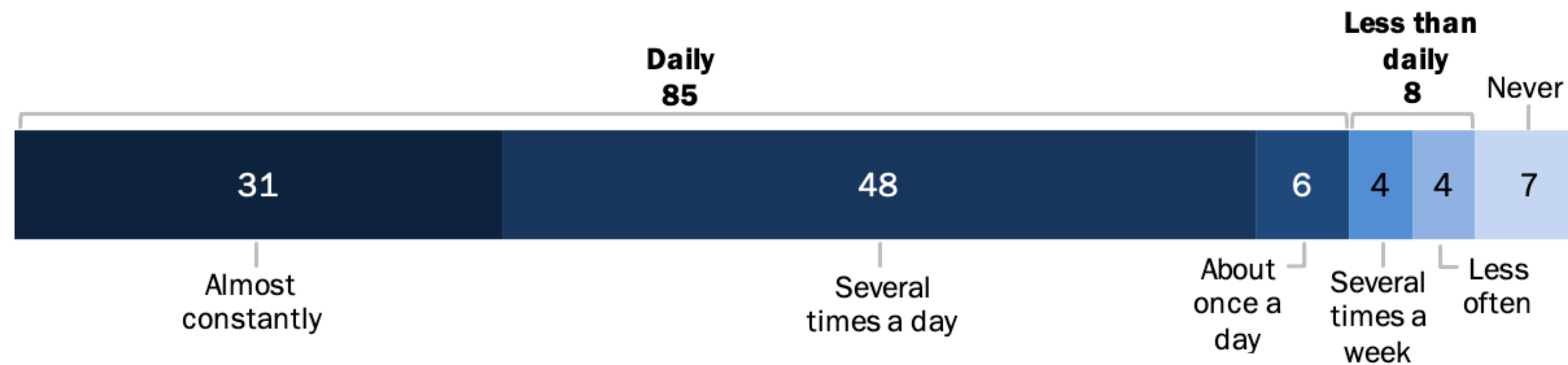
Each individual has exactly one category

- Percents add to 100

---

## More than eight-in-ten U.S. adults go online at least daily

*% of U.S. adults who say they go online ...*



Note: Respondents who did not give an answer are not shown.

Source: Survey of U.S. adults conducted Jan. 25-Feb. 8, 2021.

PEW RESEARCH CENTER

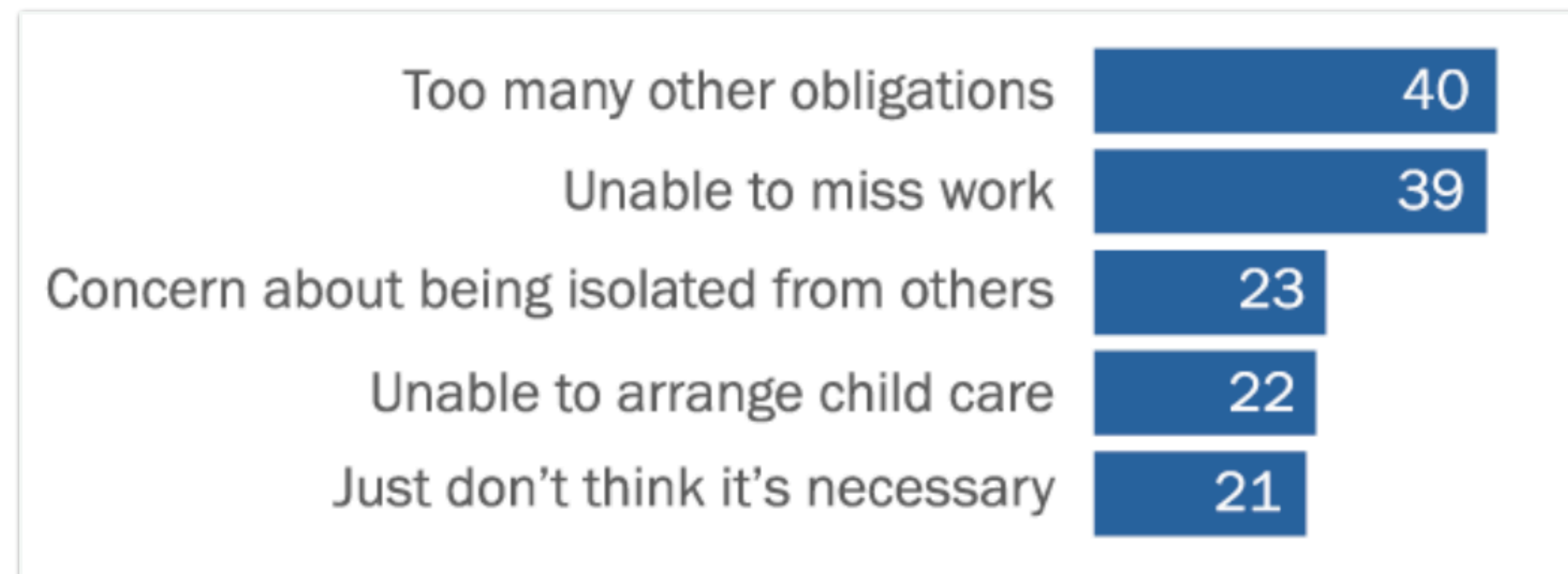
---

# Not a distribution

When individuals can pick more than one answer

- Percents don't necessarily add up to 100

Survey question: “A major reason you would find it difficult to quarantine for 14 days”





# Bar Chart: Categorical Distributions

Bar charts: display **categorical variables** and **frequencies**

- One bar for each category
- Ordering of the bar can be specified (e.g. `.sort`)
- Length of bar is either count or % of individuals in that particular category

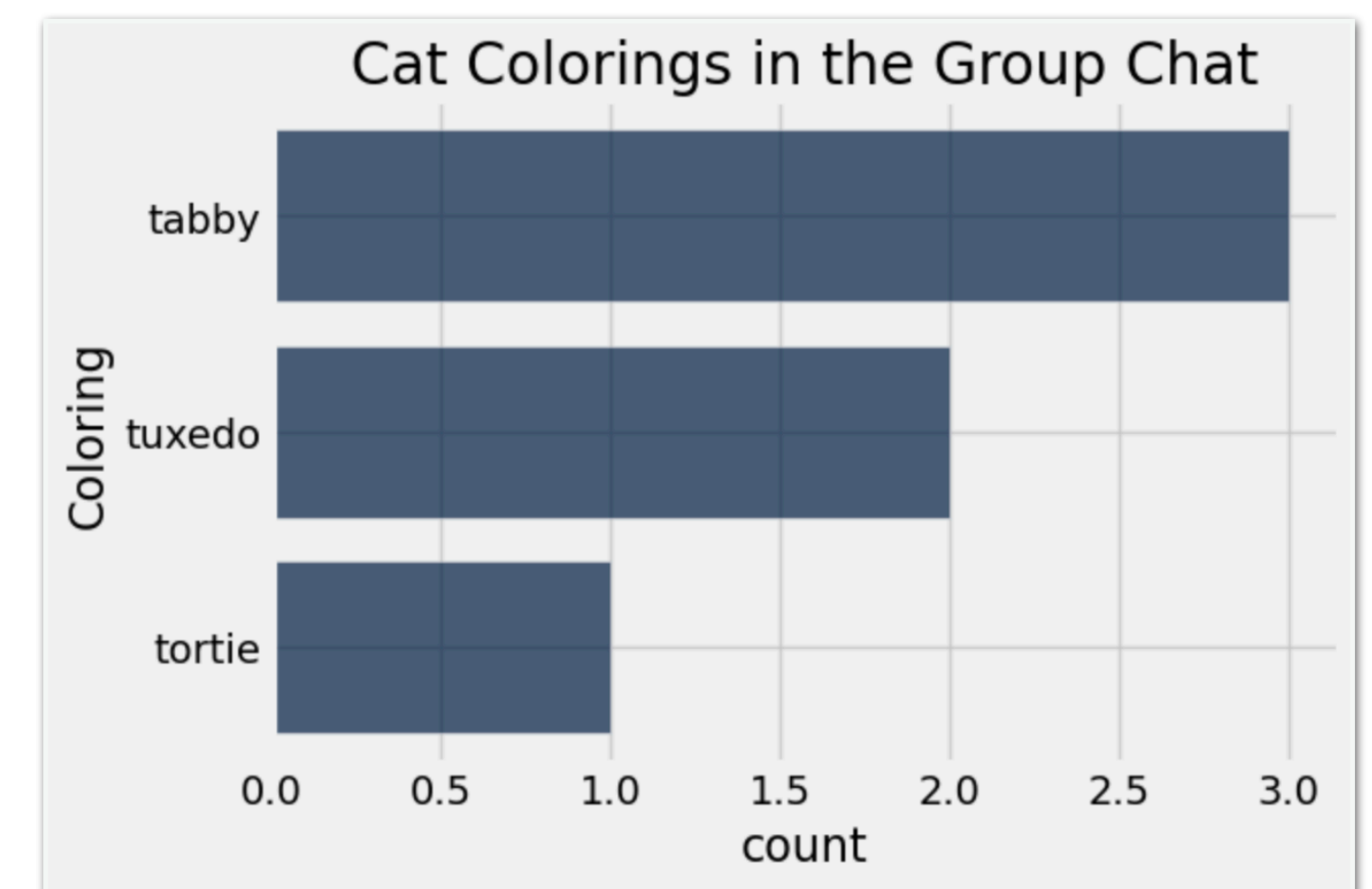
category ↓ frequency ↘

Coloring	count
tabby	3
tuxedo	2
tortie	1

```
tbl.barh(category_label, freq_label)
```

```
- cat_tbl.barh('Coloring', 'count')
```

```
- cat_tbl.barh('Coloring')
```



# Categorical Bar Chart Demo

# Creating Histograms for Numerical Distributions

# Visualizing Numerical Distributions

Let's say we have a data set containing grades students scored on an exam:

```
array([ 56,  83,  99,  87,  90,  73,  82,  88,  88,  90,  72,  77,  75,  
       85,  83,  88,  75,  93,  94,  86,  85,  87,  78,  63,  97,  96,  
       87,  66,  90,  91,  81,  81,  85,  70,  58,  77,  92,  66,  85,  
       93,  79,  85,  79,  90,  98,  75,  83,  76,  86,  82,  90,  67,  
       72,  90,  85,  91,  69,  94,  92,  99,  92,  92,  80,  72,  82,  
       91,  96,  90, 100,  90,  84,  80,  64,  71,  99,  92])
```

What if we want to know generally how students on the exam?

- Useful to sort these into groups (**bins**)
- Compare the size of these bins



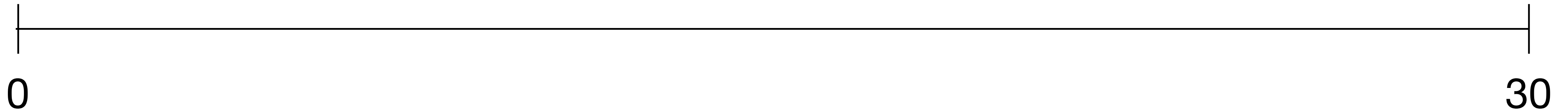
# Grouping Numerical Values into Bins

Binning is counting the number of numerical values that lie within a range (which is called bins)

- Bins are defined by their lower bounds (inclusive)
- The upper bound is the lower bound for the next bin)
  - Example: [10, 20)

# Binning: Example

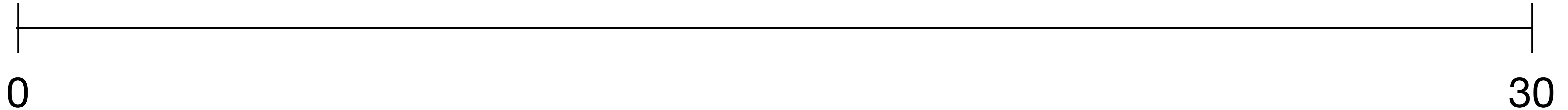
[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]



# Binning: Example

[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]

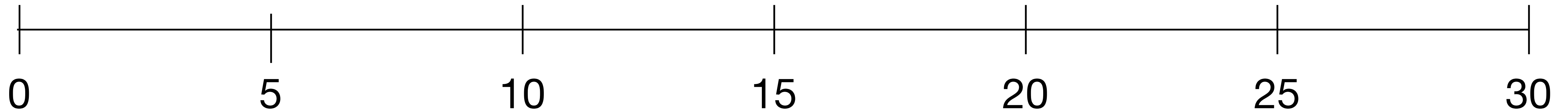
Let's say we decide to use a bin size of 5



# Binning: Example

[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]

Let's say we decide to use a bin size of 5

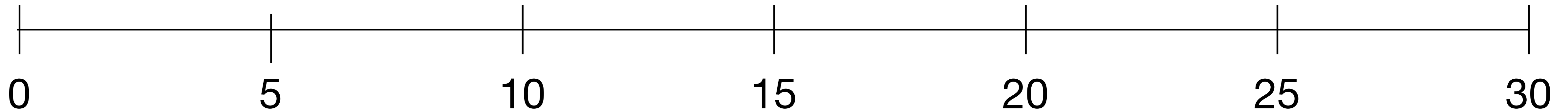




# Binning: Example

[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]

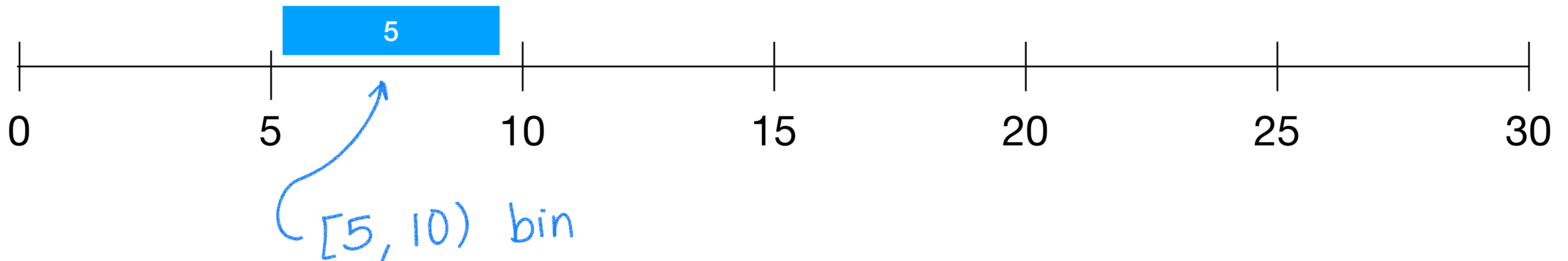
What bin would 5 fall into?



# Binning: Example

[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]

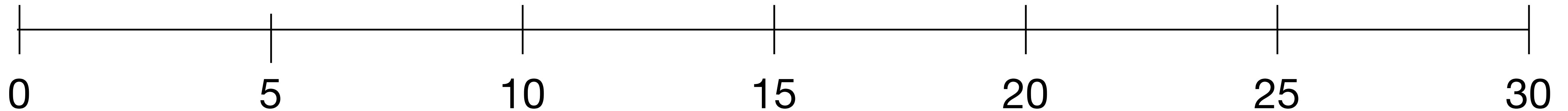
What bin would 5 fall into?



# Binning: Example

[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]

How many individuals fall into bin 15-20?

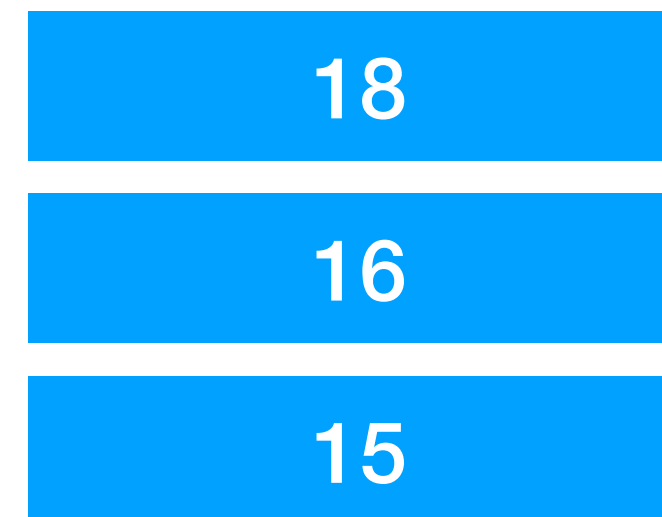


# Binning: Example

[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]

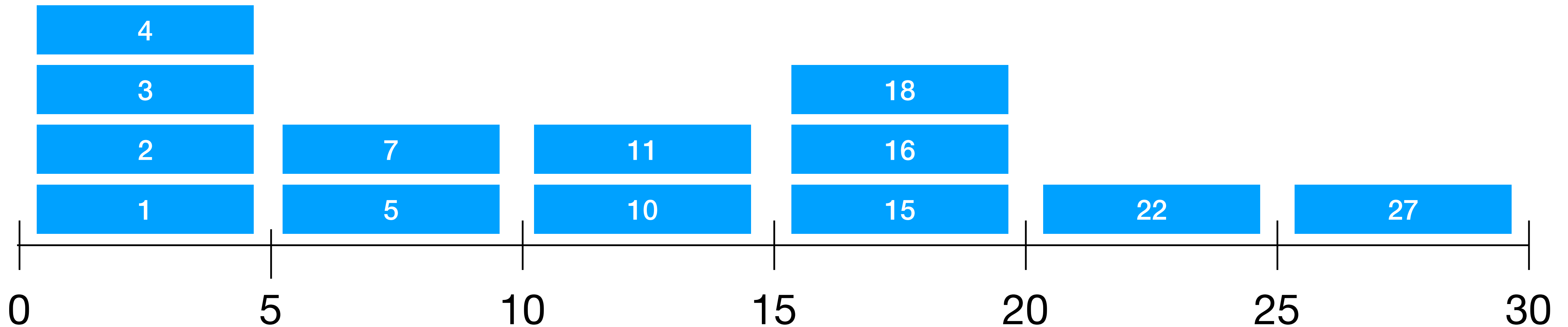
How many individuals fall into bin 15-20?

3



# Binning: Example

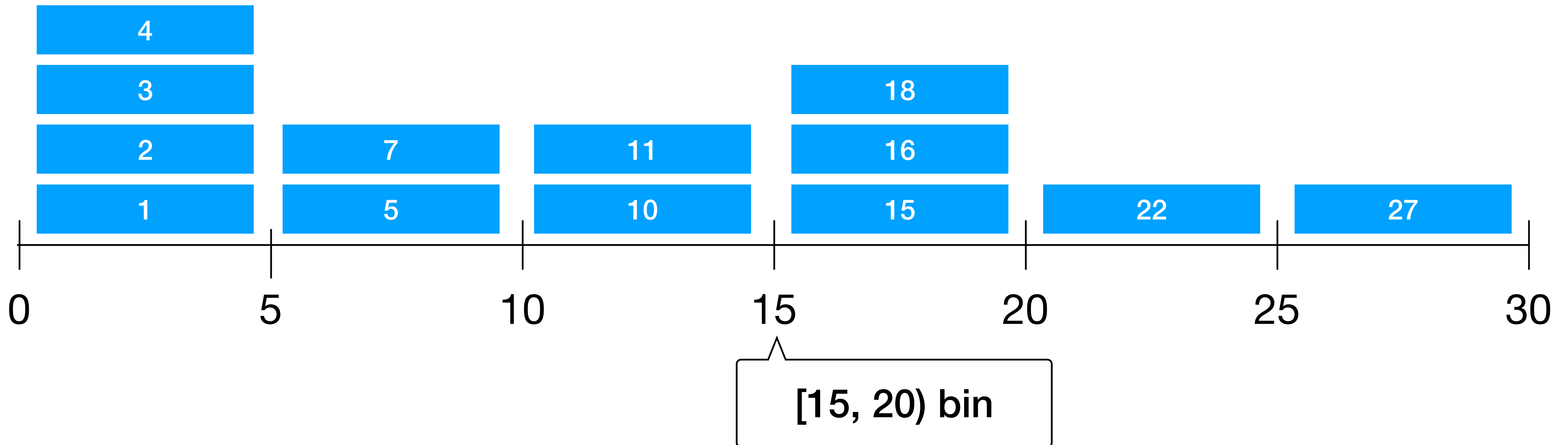
[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]





# Binning: Example

[1, 5, 7, 3, 2, 11, 18, 16, 15, 10, 22, 27, 4]



# Choosing Bin Size

Let's go back to our data from before:

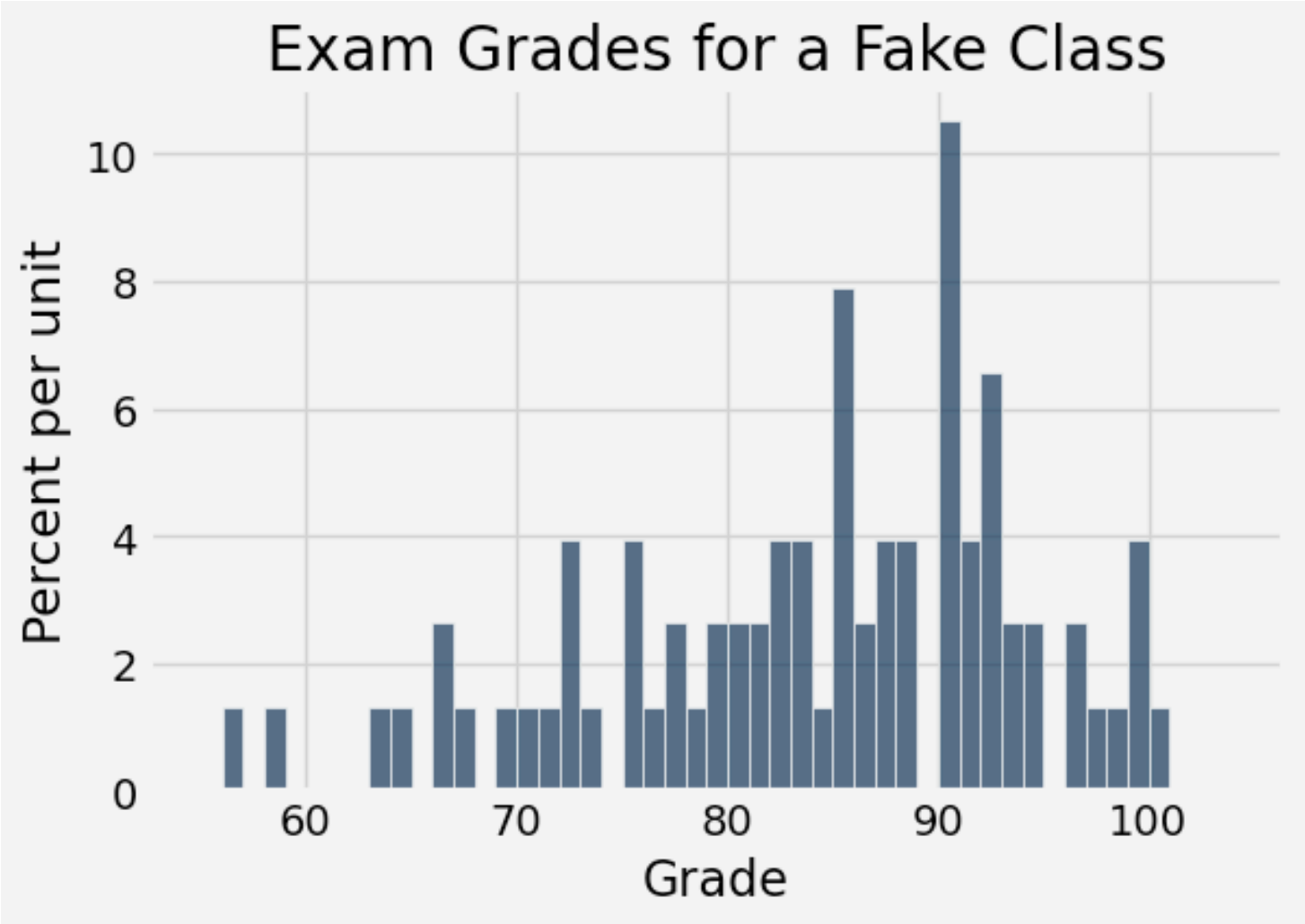
```
array([ 56,  83,  99,  87,  90,  73,  82,  88,  88,  90,  72,  77,  75,  
       85,  83,  88,  75,  93,  94,  86,  85,  87,  78,  63,  97,  96,  
       87,  66,  90,  91,  81,  81,  85,  70,  58,  77,  92,  66,  85,  
       93,  79,  85,  79,  90,  98,  75,  83,  76,  86,  82,  90,  67,  
       72,  90,  85,  91,  69,  94,  92,  99,  92,  92,  80,  72,  82,  
       91,  96,  90, 100,  90,  84,  80,  64,  71,  99,  92])
```

# Choosing Bin Size

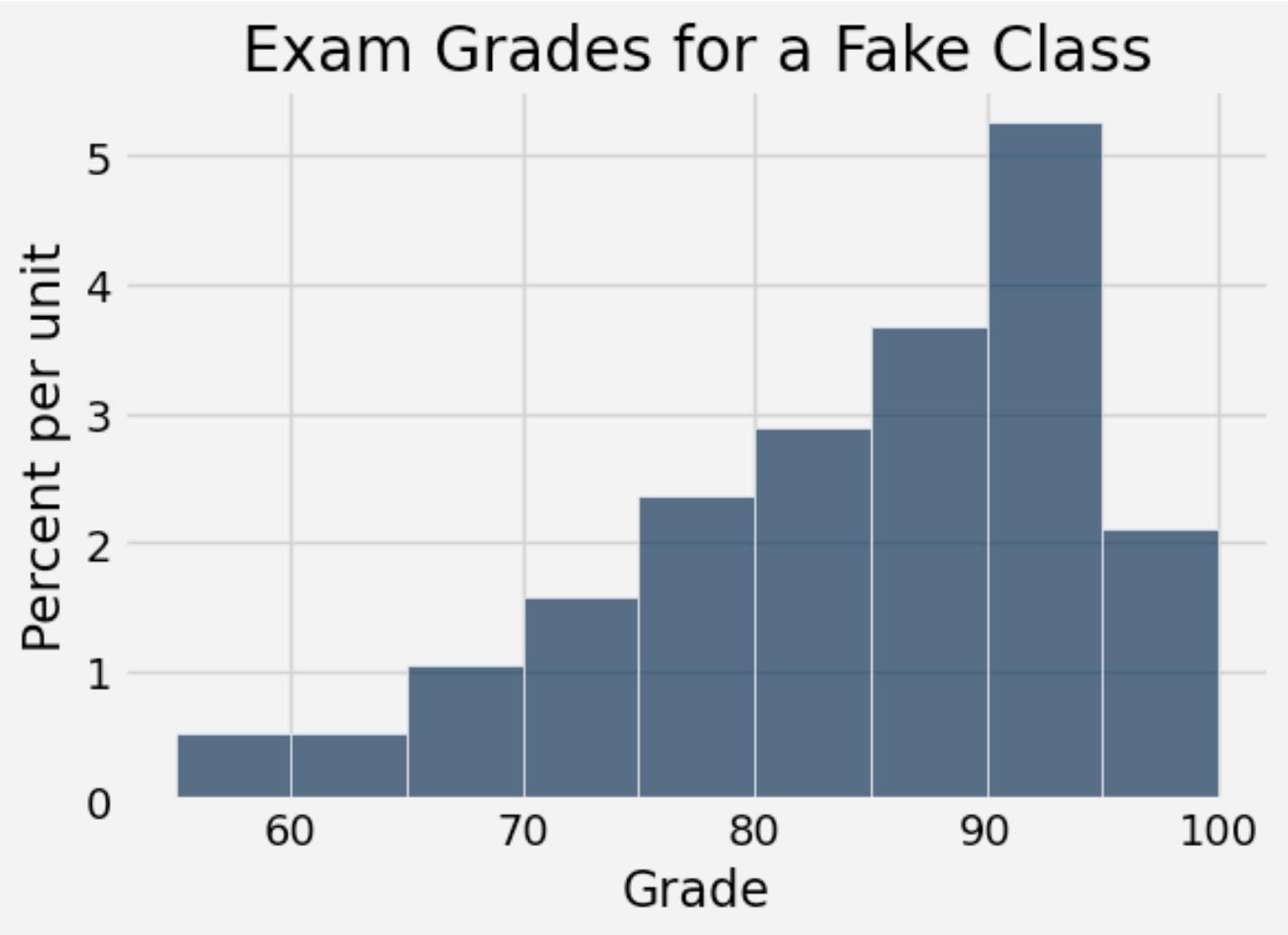
Let's go back to our data from before:

```
array([ 56,  83,  99,  87,  90,  73,  82,  88,  88,  90,  72,  77,  75,
        85,  83,  88,  75,  93,  94,  86,  85,  87,  78,  63,  97,  96,
        87,  66,  90,  91,  81,  81,  85,  70,  58,  77,  92,  66,  85,
        93,  79,  85,  79,  90,  98,  75,  83,  76,  86,  82,  90,  67,
        72,  90,  85,  91,  69,  94,  92,  99,  92,  92,  80,  72,  82,
        91,  96,  90, 100,  90,  84,  80,  64,  71,  99,  92])
```

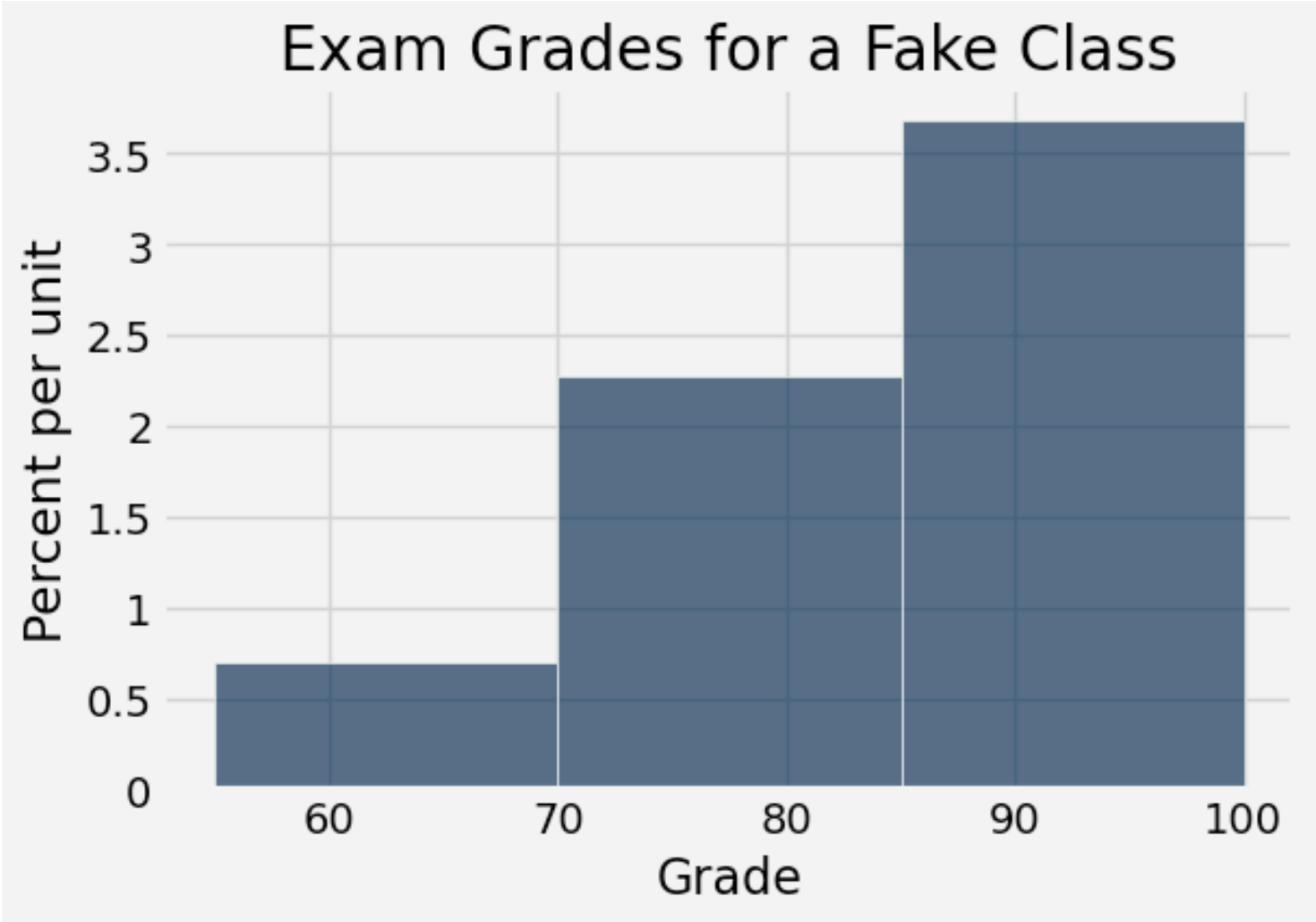
Bin size = 1



Bin size = 5

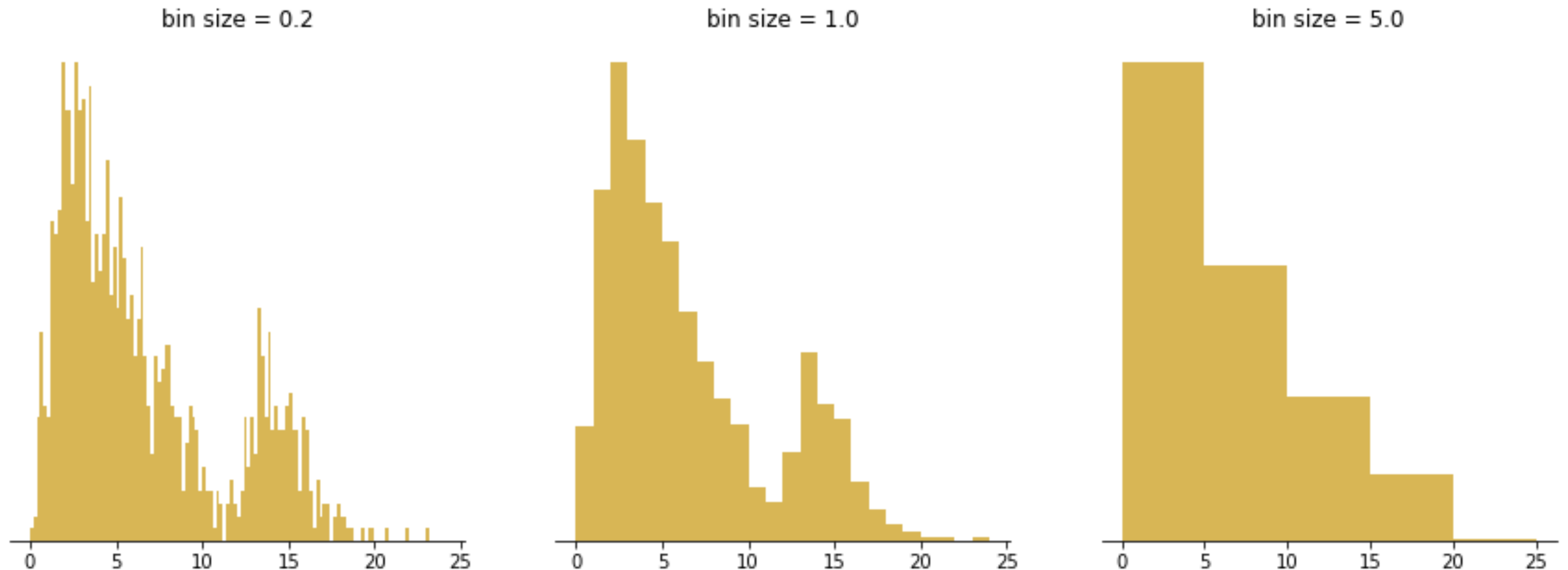


Bin size = 15



# Choosing Bin Size

Chose so that it's representative of your data



# bin

Group values in column `c` into 10 equally sized intervals:

- `tbl.bin(c)`

Create `n` equally wide bins:

- `tbl.bin(c, bins=n)`

Create bins of size `step` from `start` to `end`:

- `tbl.bin(c, bins=np.arange(start, end, step))`



# hist

Create a histogram of numerical values in column `c` with 10 equal bins:

- `tbl.hist(c)`

Create a histogram with `u` as the x-axis:

- `tbl.hist(c, unit=u)`

Create a histogram with specified bins:

- `tbl.hist(c, bins=np.arange(start, end, step))`

Create a histogram with x-axis `u` and specified bins:

- `tbl.hist(c, unit=u, bins=np.arange(start, end, step))`

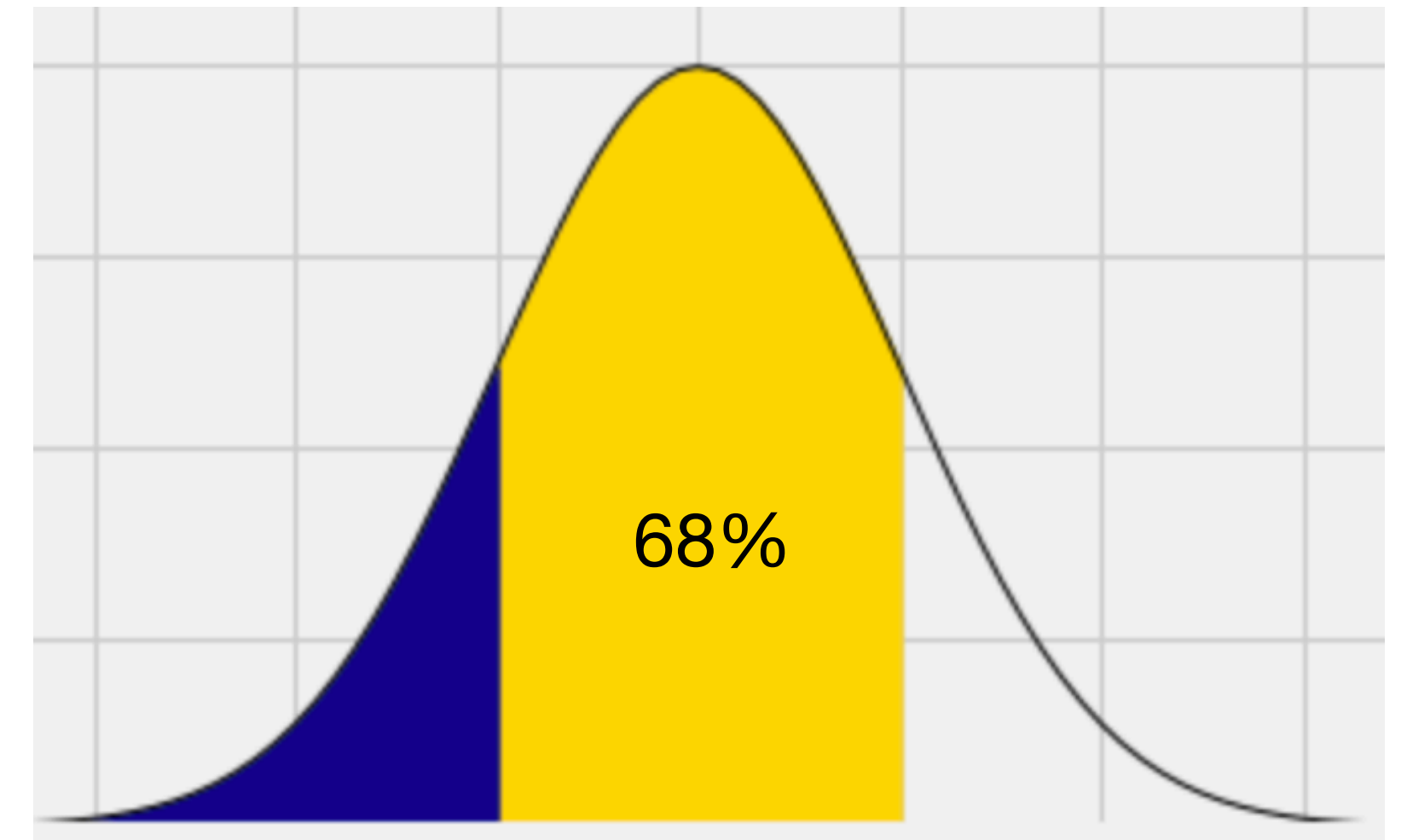
# Histogram Notebook Demo - Bins

# Area Principle

Areas should be *proportional* to the values they represent

In a histogram, the **area** of each bar is the **percent** of individuals in the corresponding bin

(Later on in the course, we will approximate histograms with smooth curves)



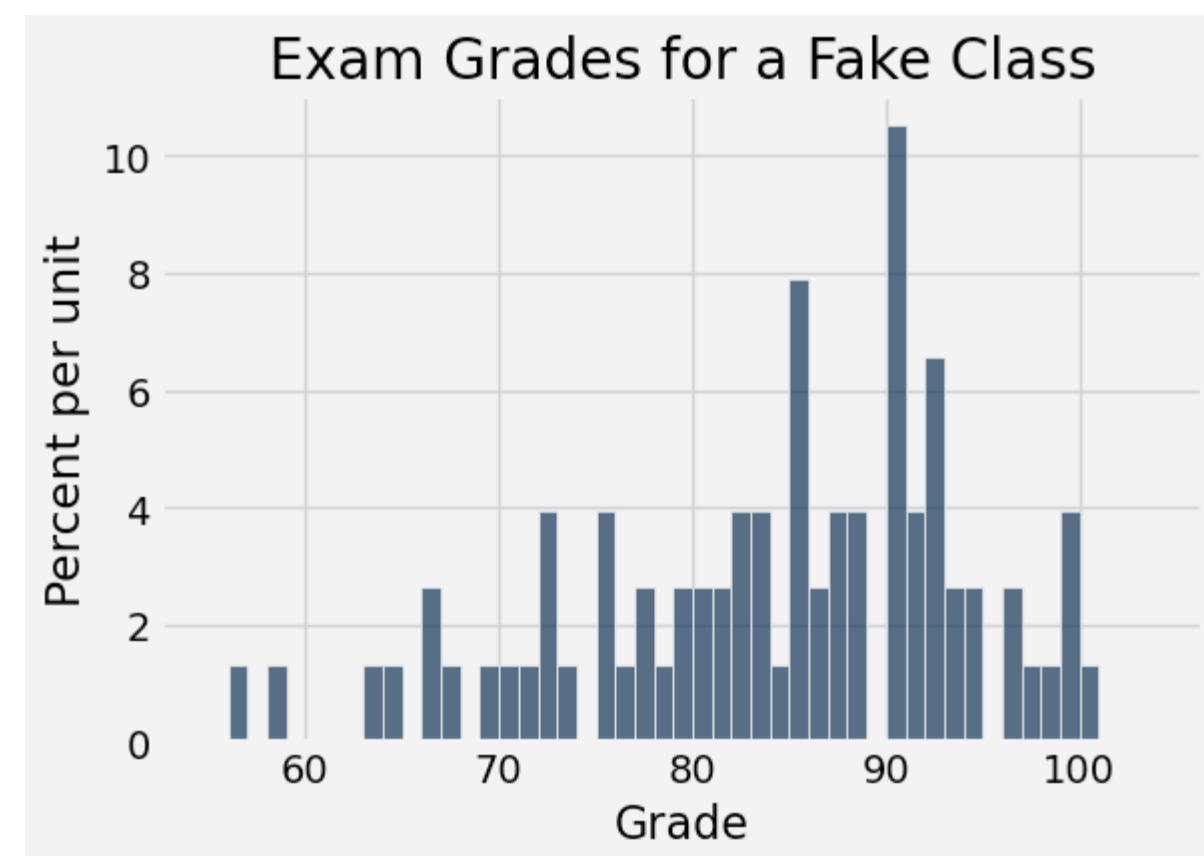
# Area Principle

area of bar = percent of entries in bin

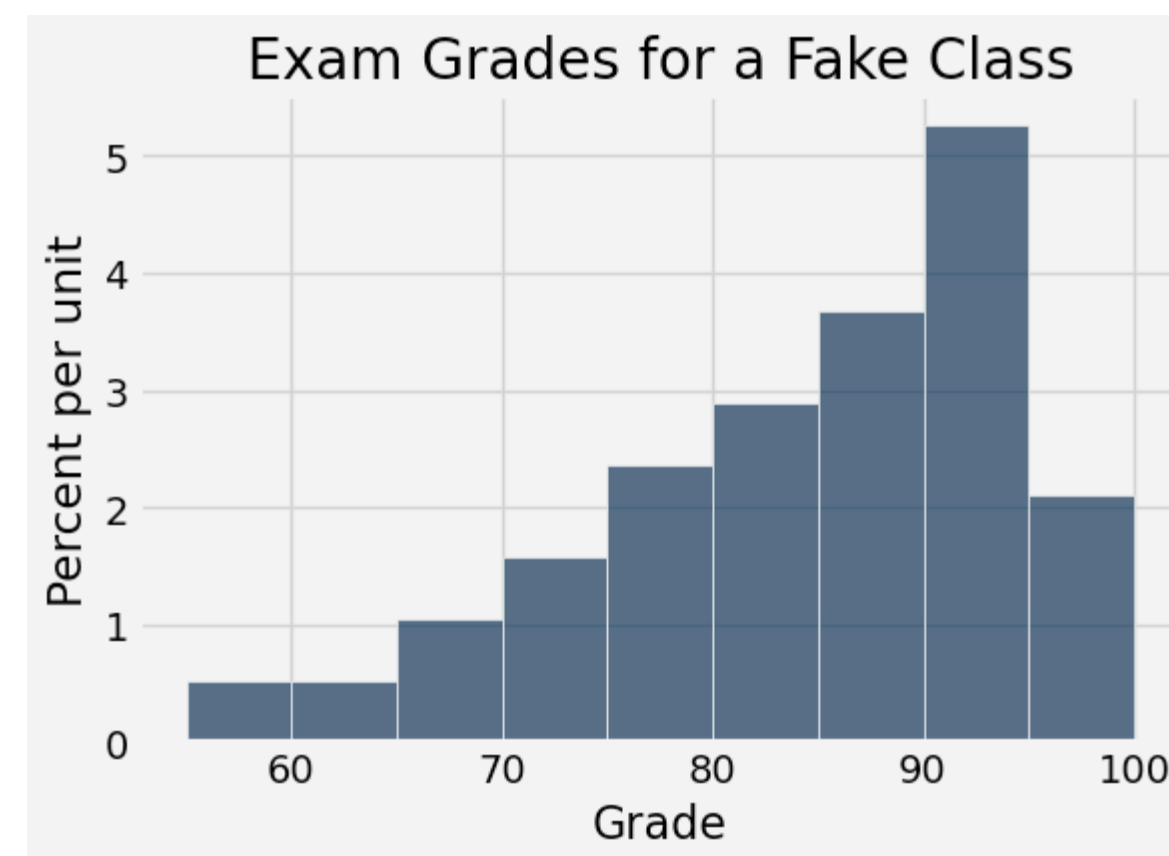
area of bar = (height of bar)  $\times$  (width of bin)

$$\text{height of bar} = \frac{\text{area of bar}}{\text{width of bin}} = \frac{\text{percent of entries in bin}}{\text{width of bin}}$$

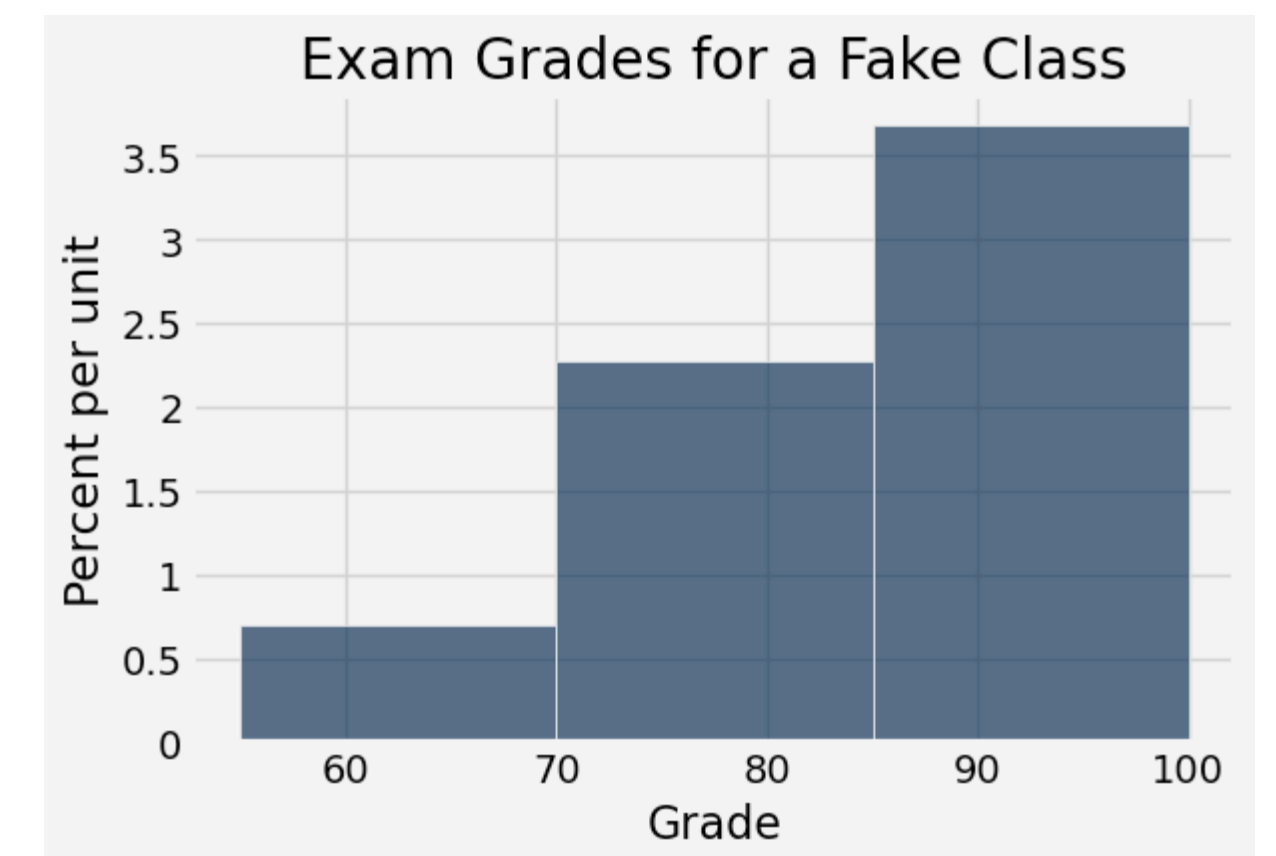
Bin size = 1



Bin size = 5



Bin size = 15



# Area Principle

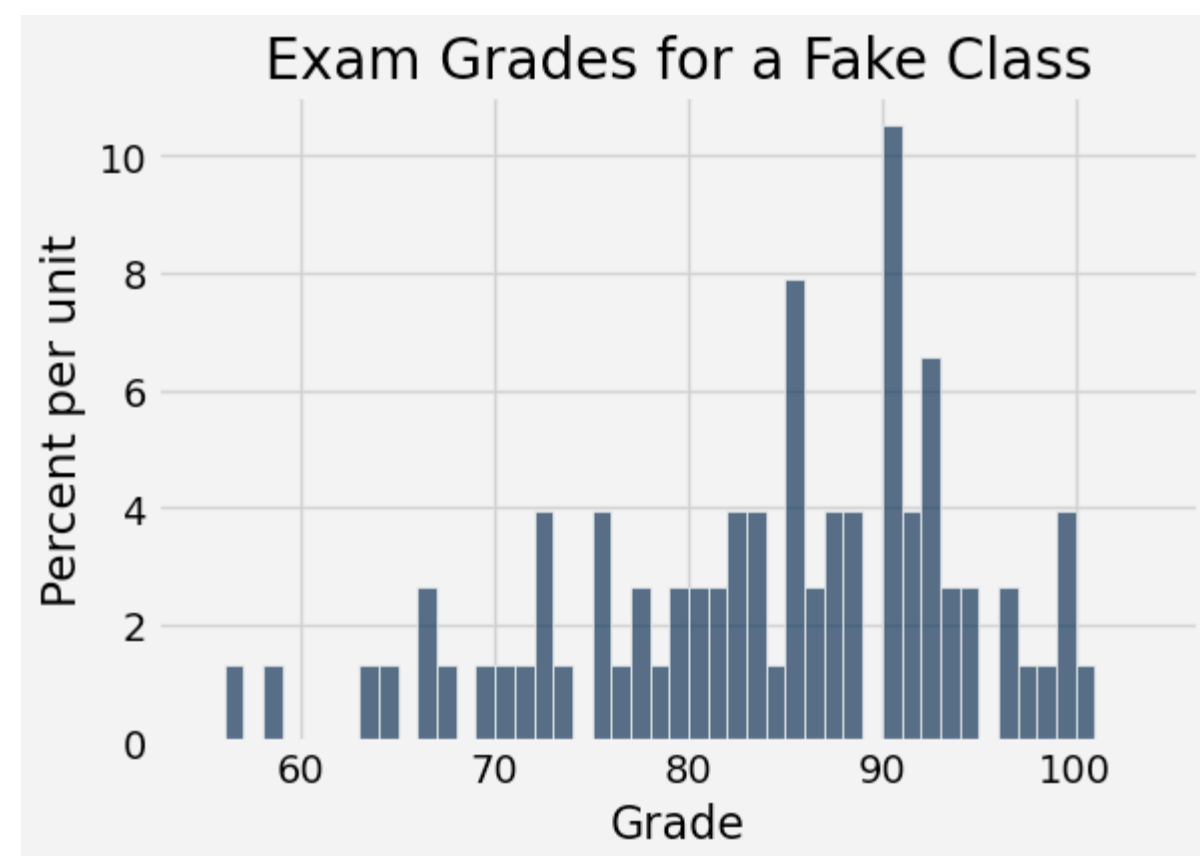
area of bar = percent of entries in bin

area of bar = (height of bar)  $\times$  (width of bin)

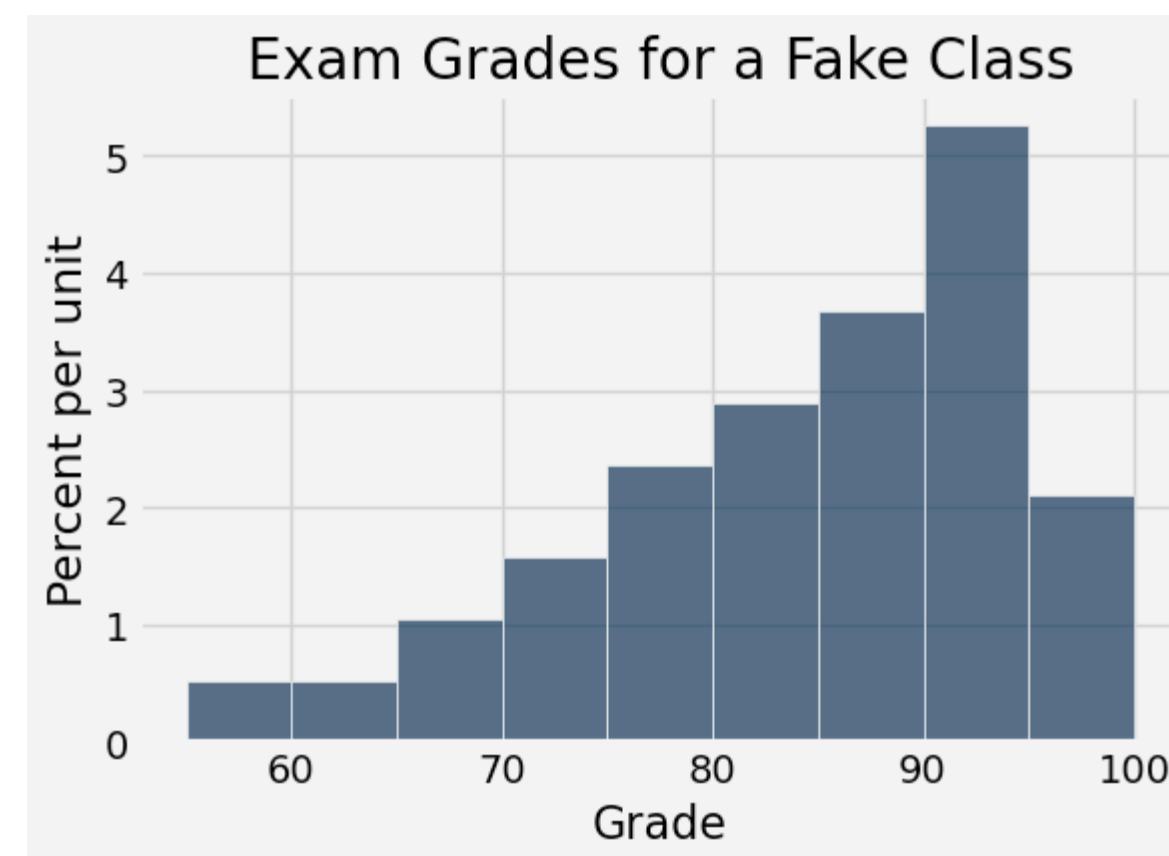
height of bar =  $\frac{\text{area of bar}}{\text{width of bin}} = \frac{\text{percent of entries in bin}}{\text{width of bin}}$

when bin size is 1:  
height =  $\frac{\text{area of bar}}{\text{width of bin}}$   
=  $\frac{\text{area of bar}}{1}$   
=  $\frac{\% \text{ of entries in bin}}{1}$

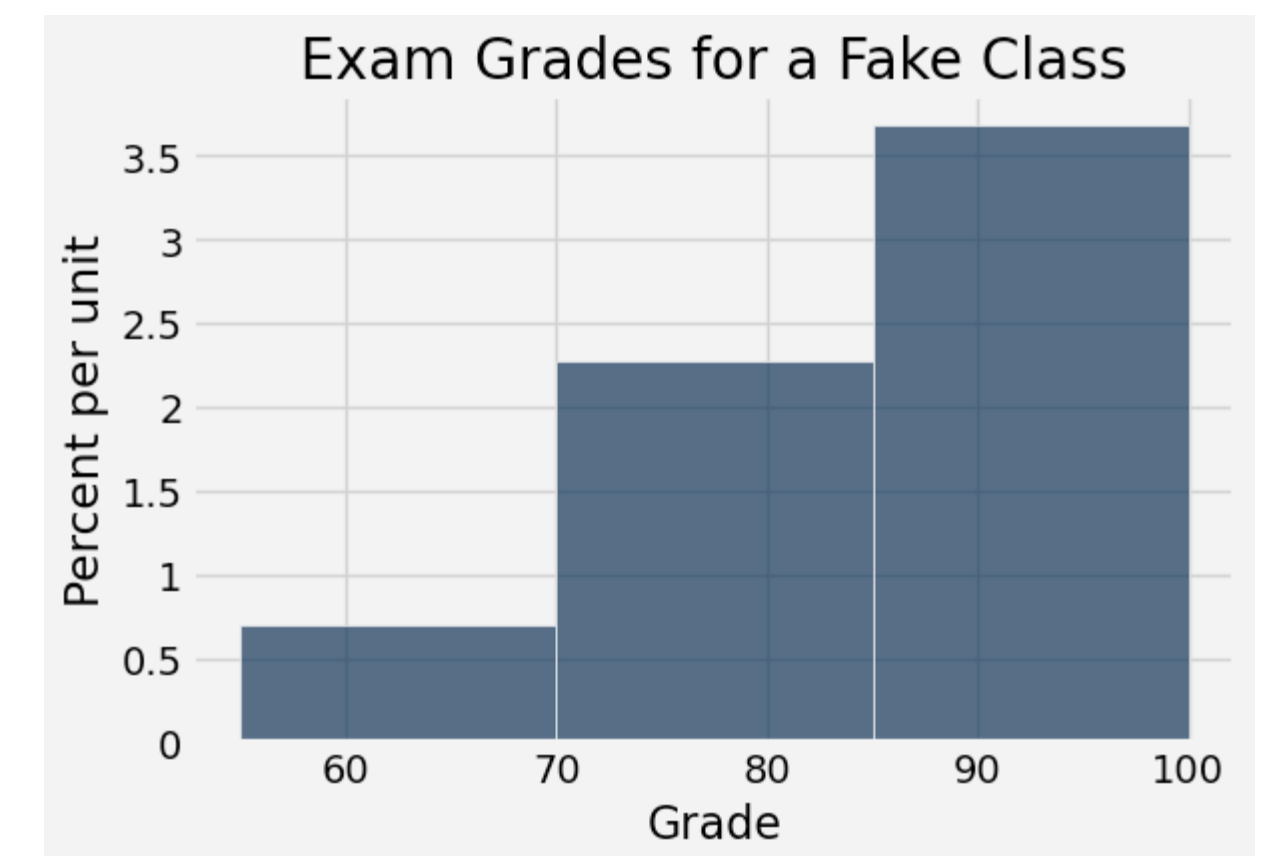
Bin size = 1



Bin size = 5



Bin size = 15



# Area Principle

area of bar = percent of entries in bin

area of bar = (height of bar)  $\times$  (width of bin)

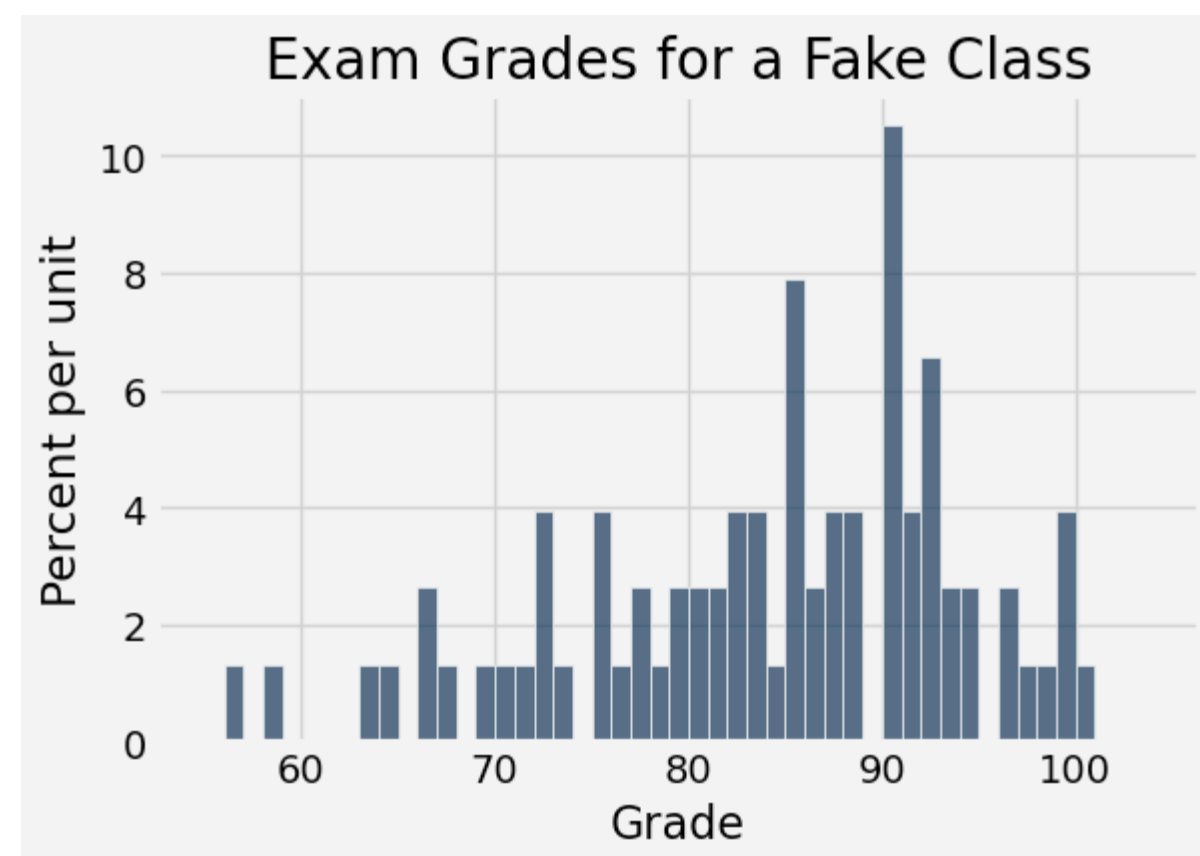
height of bar =  $\frac{\text{area of bar}}{\text{width of bin}} = \frac{\text{percent of entries in bin}}{\text{width of bin}}$

When bin size is 1:  
 $\text{height} = \frac{\text{area of bar}}{\text{width of bin}}$   
 $= \frac{\text{area of bar}}{1}$   
 $= \frac{\% \text{ of entries in bin}}{1}$

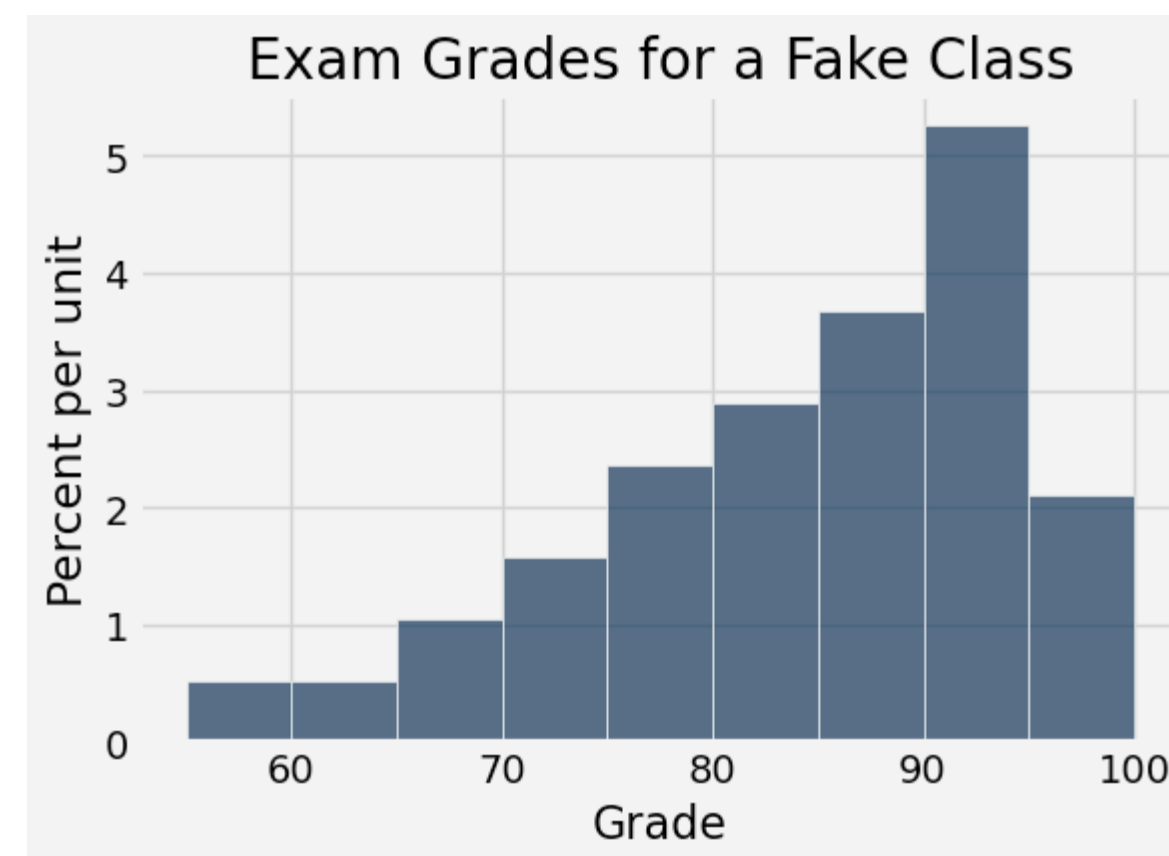
When bin size is 5:  
 $\text{height} = \frac{\% \text{ entries in bin}}{5}$

When bin size is 15:  
 $\text{height} = \frac{\% \text{ entries in bin}}{15}$

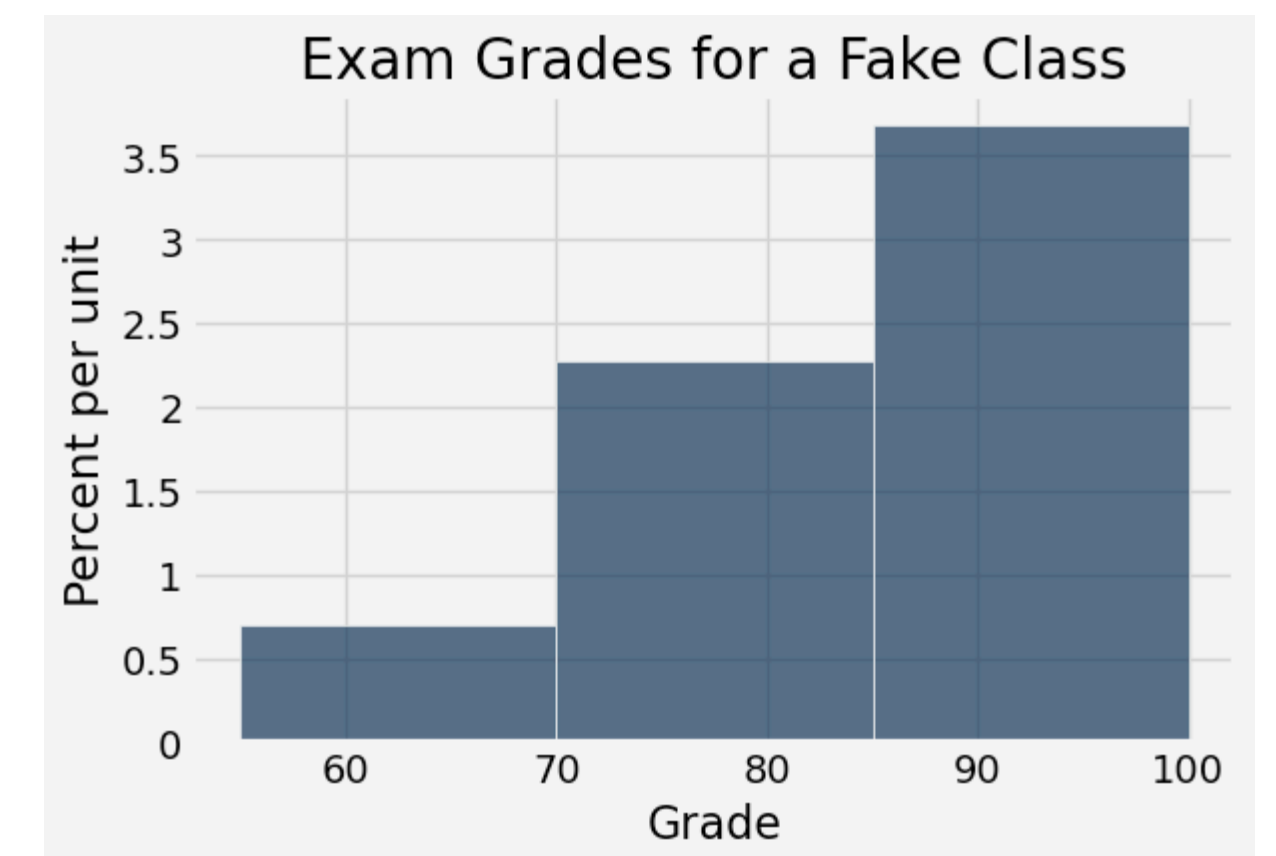
Bin size = 1



Bin size = 5

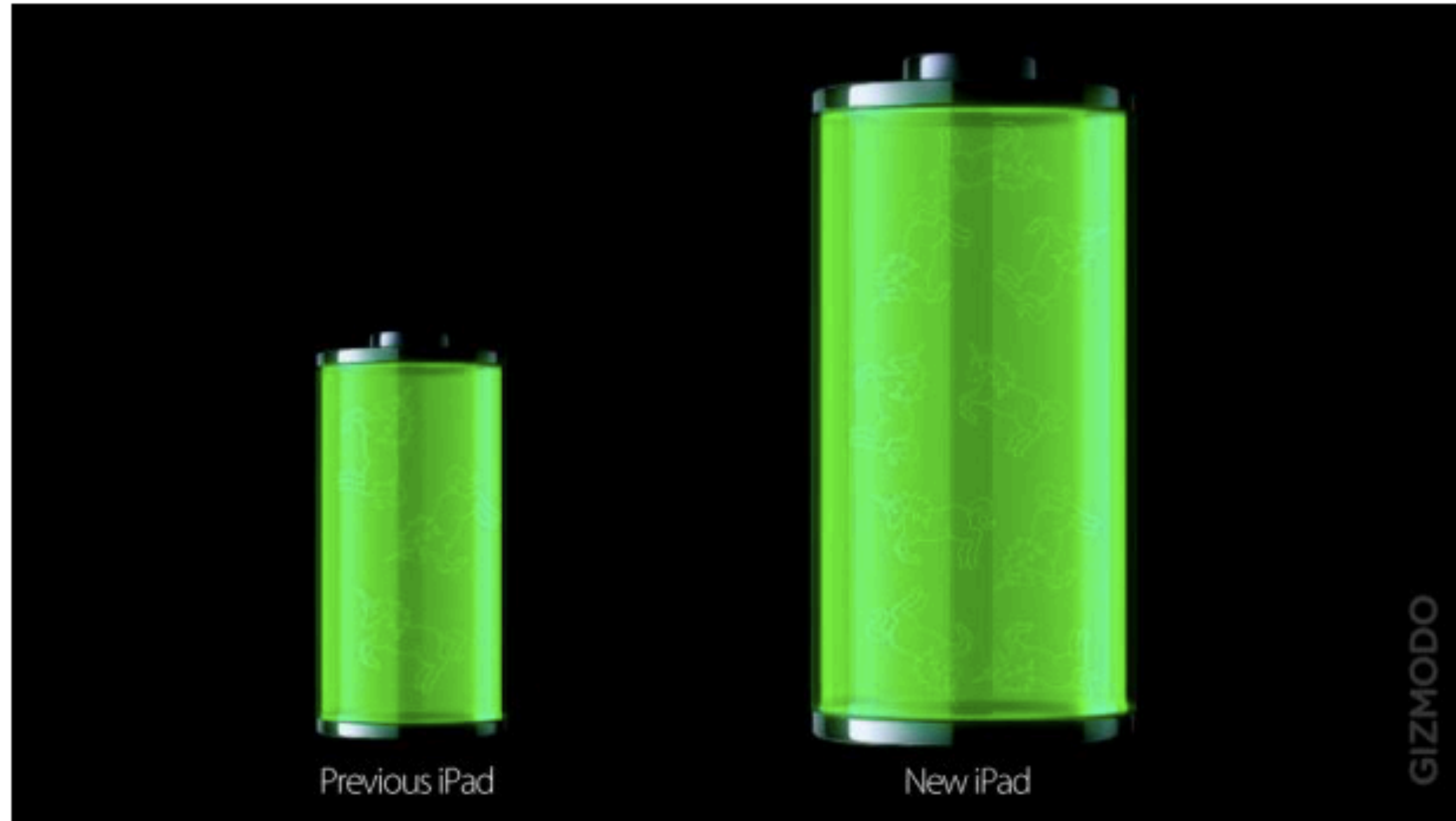


Bin size = 15



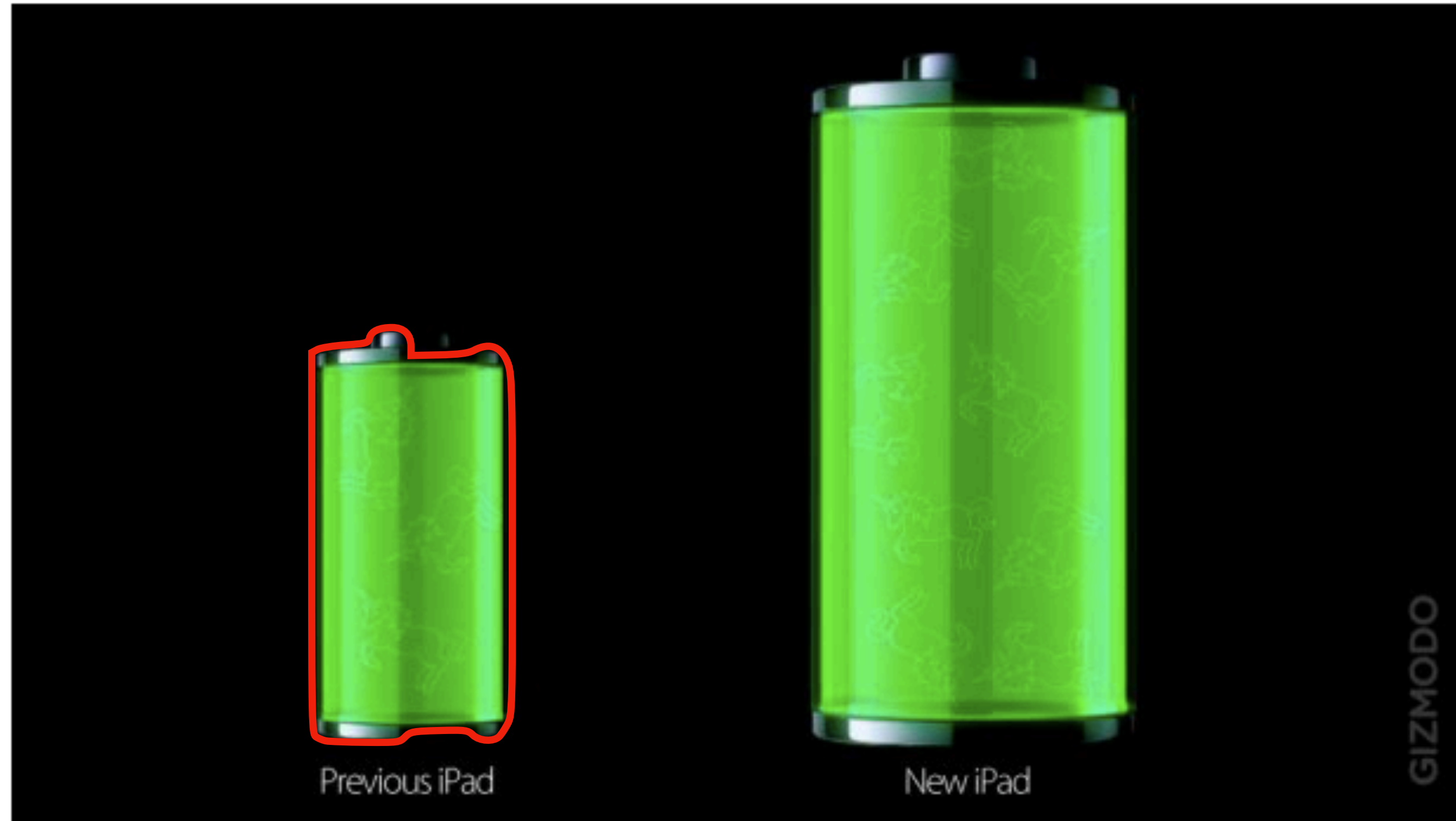


# Area Principle



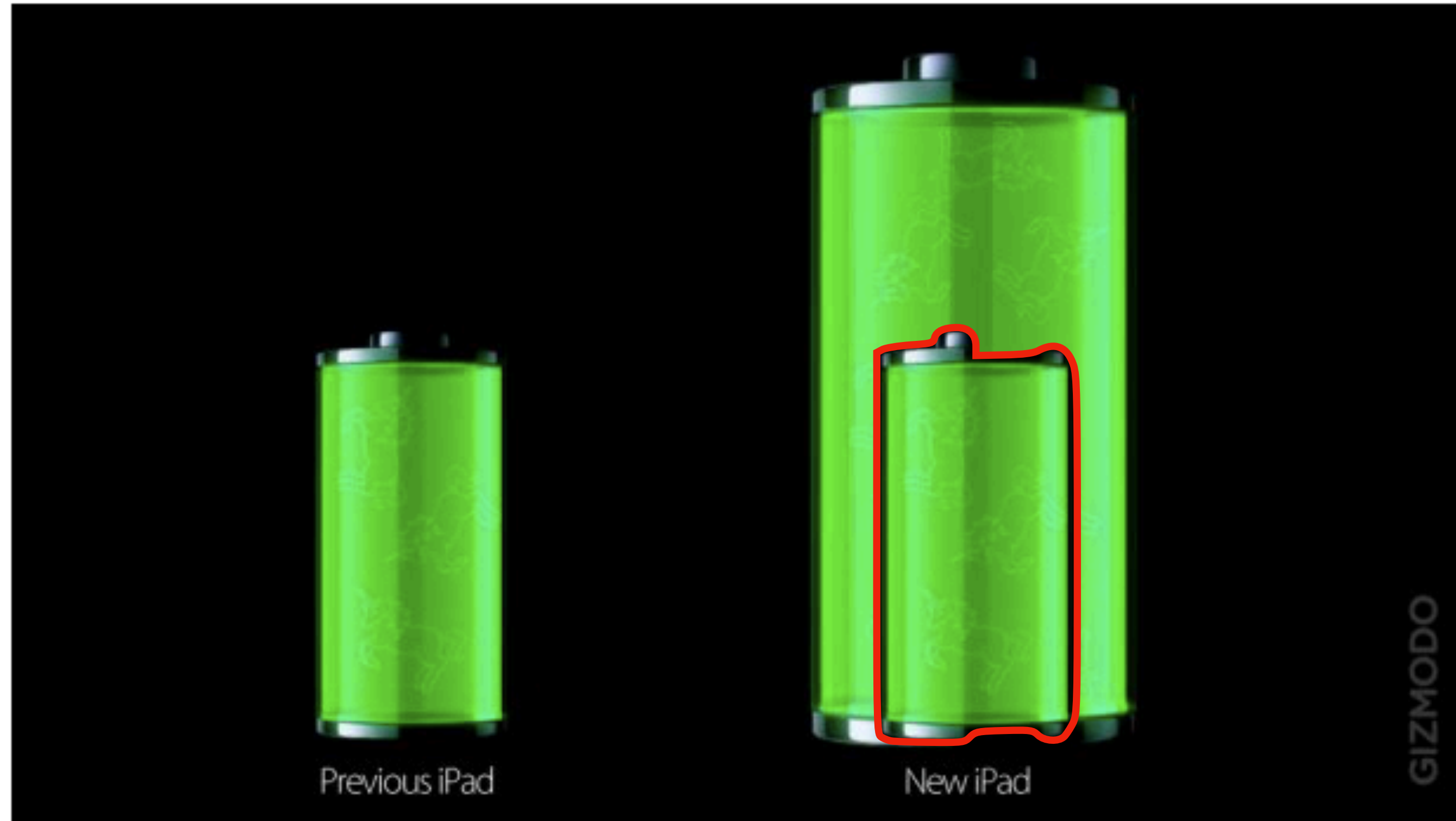
From [Gizmodo](#), this shows battery size in the new iPad versus that of the iPad 2. The battery in the former is 70 percent bigger than that of the latter. Something's not right here.

# Area Principle



From [Gizmodo](#), this shows battery size in the new iPad versus that of the iPad 2. The battery in the former is 70 percent bigger than that of the latter. Something's not right here.

# Area Principle

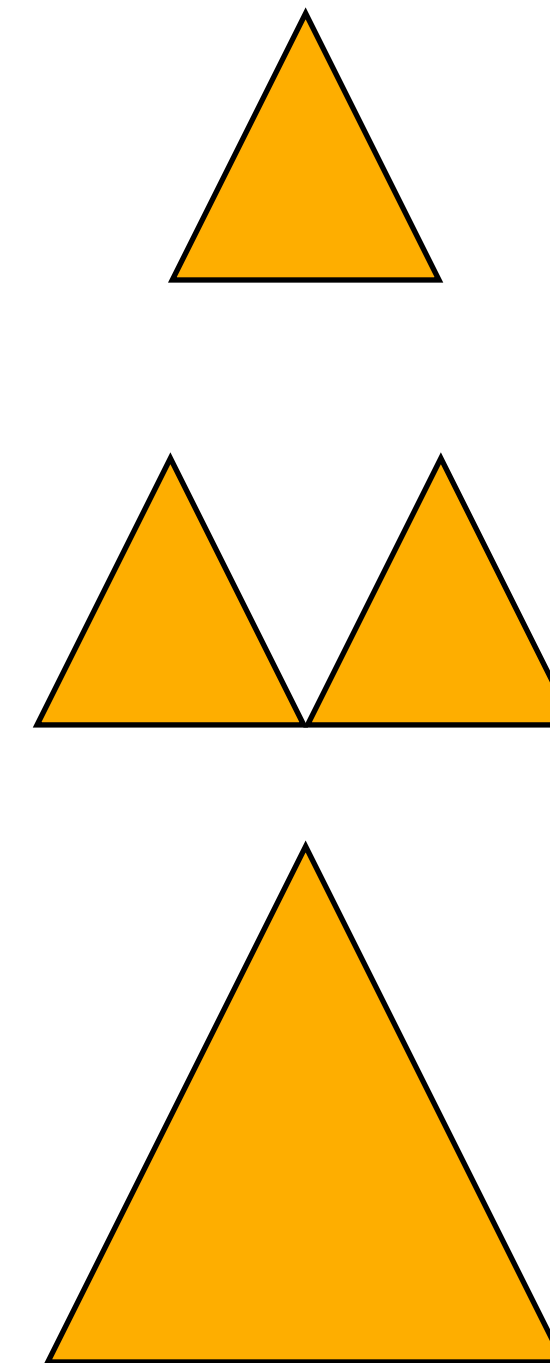


From [Gizmodo](#), this shows battery size in the new iPad versus that of the iPad 2. The battery in the former is 70 percent bigger than that of the latter. Something's not right here.

# Area Principle

Areas should be proportional to the values they represent

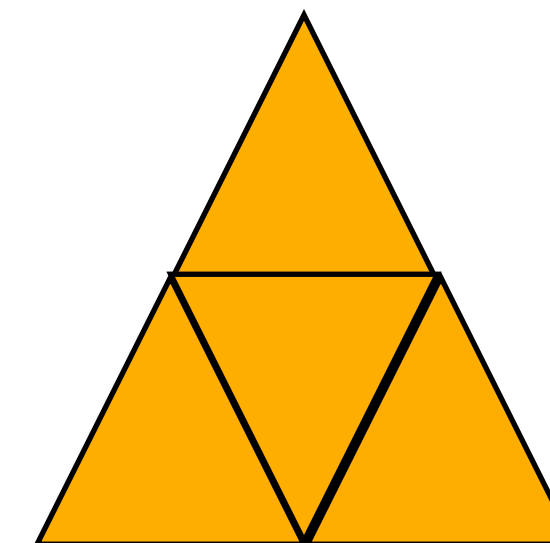
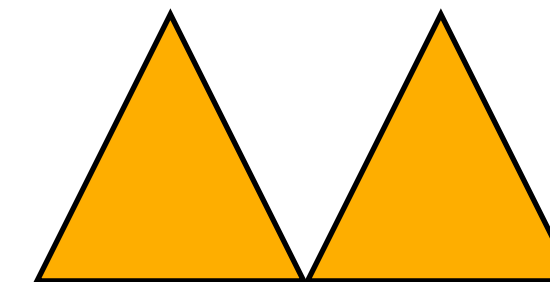
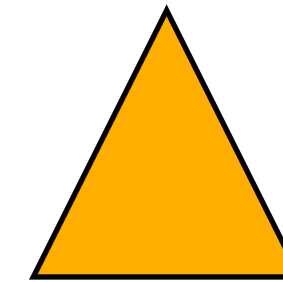
- For example
  - If you represent 20% of a population by:
  - Then 40% can be represented by:
  - But not by:



# Area Principle

Areas should be proportional to the values they represent

- For example
  - If you represent 20% of a population by:
  - Then 40% can be represented by:
  - But not by:

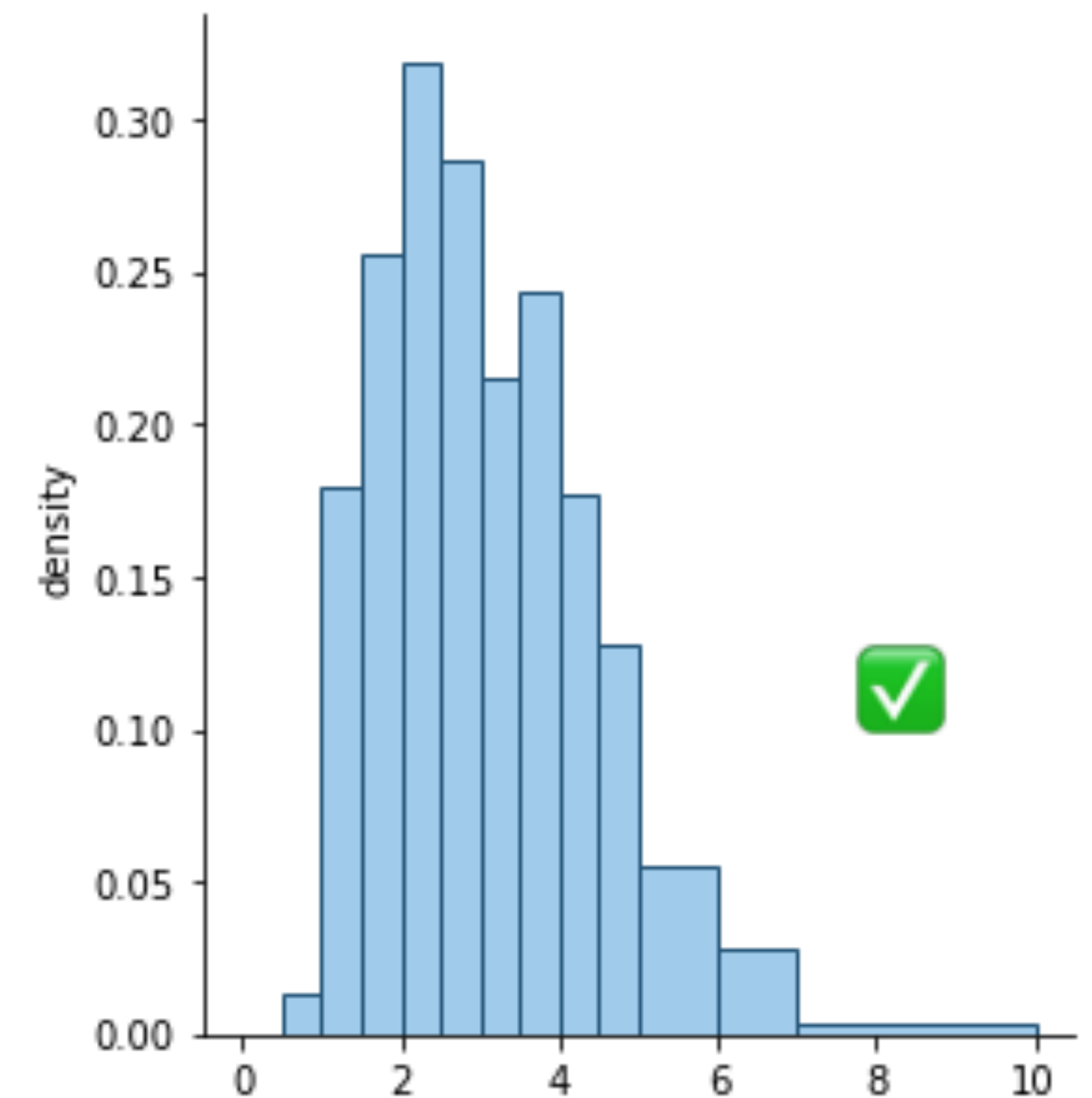
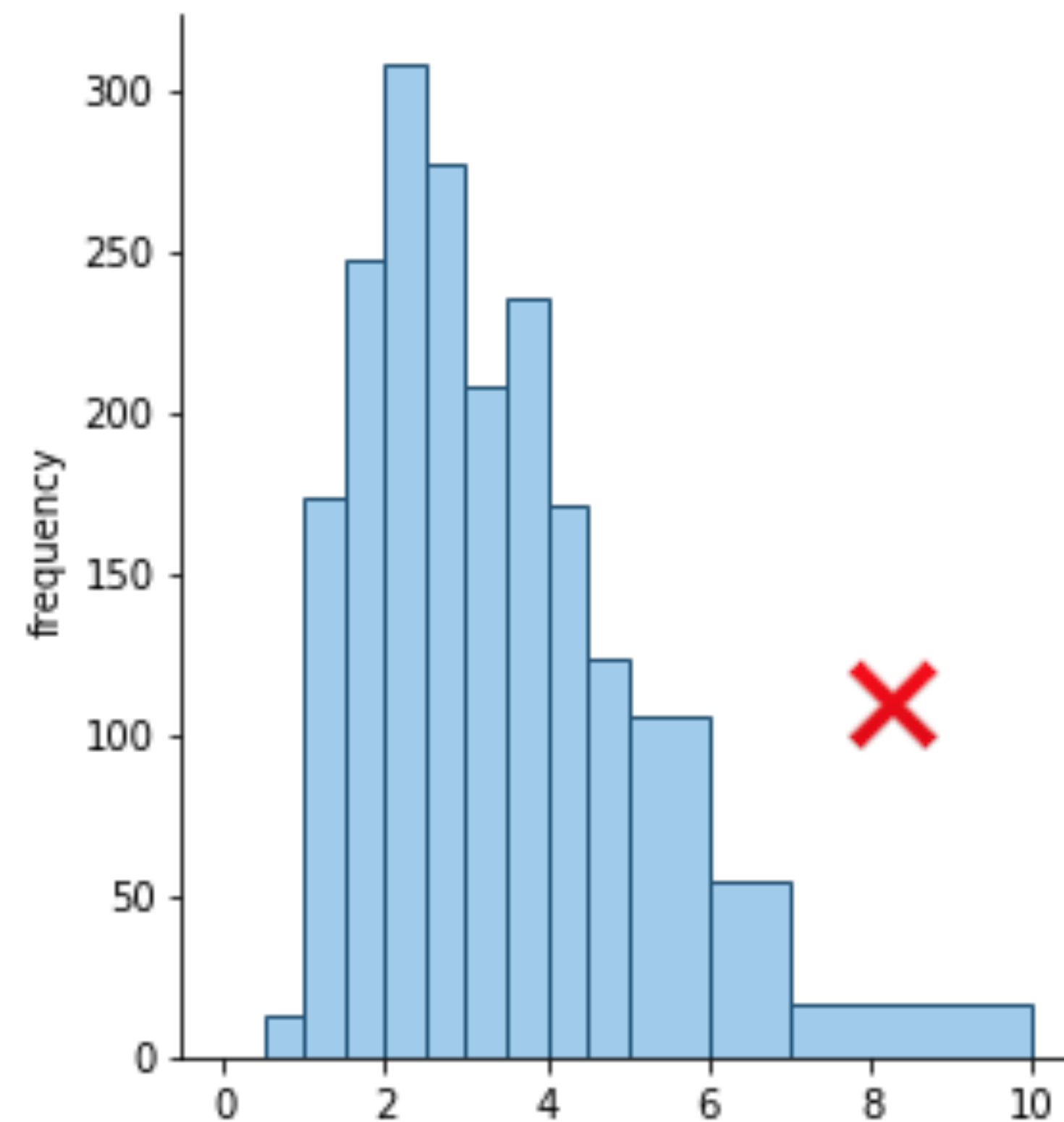
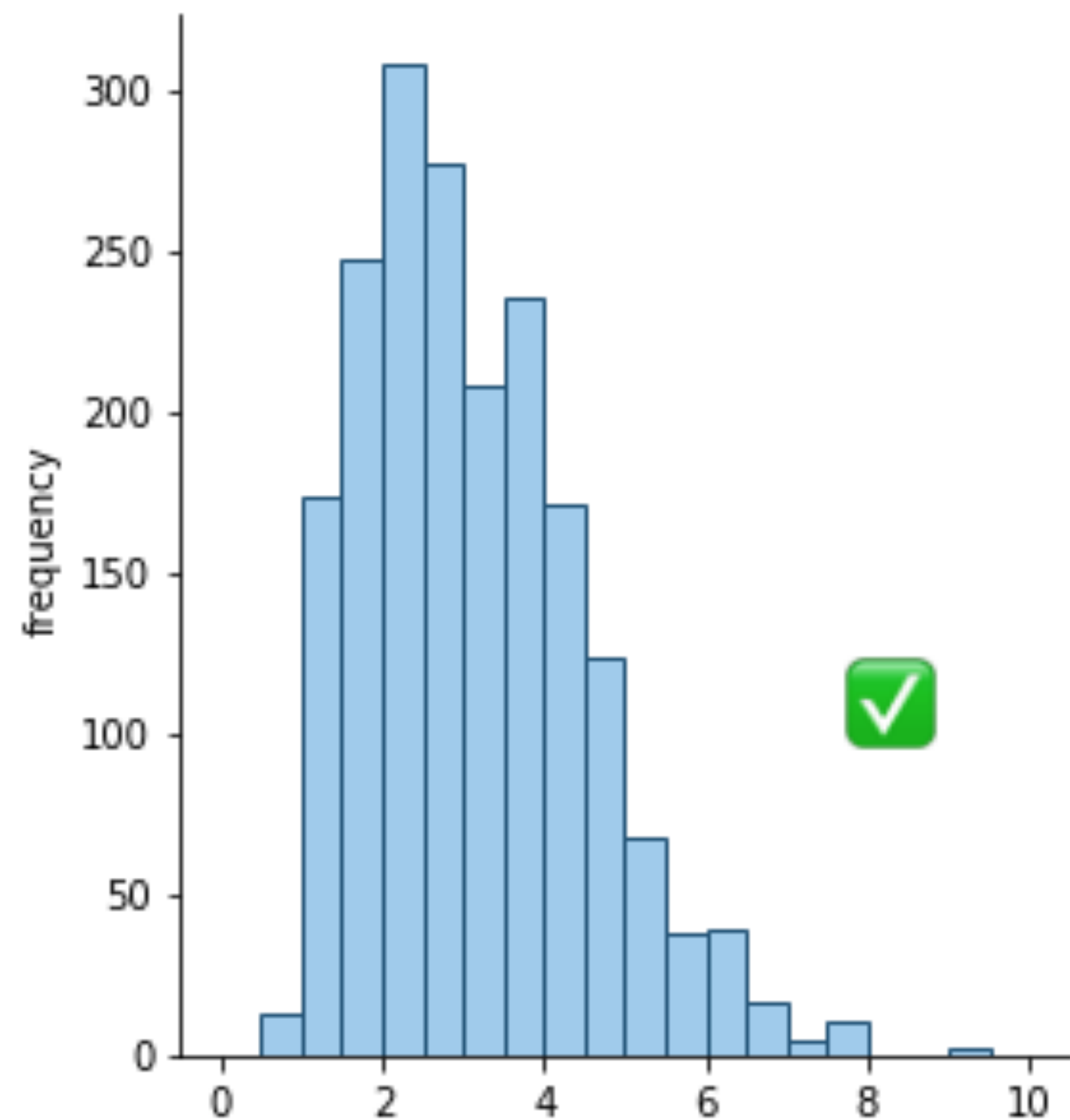


# Unequal Bin Sizes

Bin sizes don't need to be equal

- unequal bin size is often used for better representing tails

For unequal bin sizes - vertical axis now represents ***density*** rather than frequency



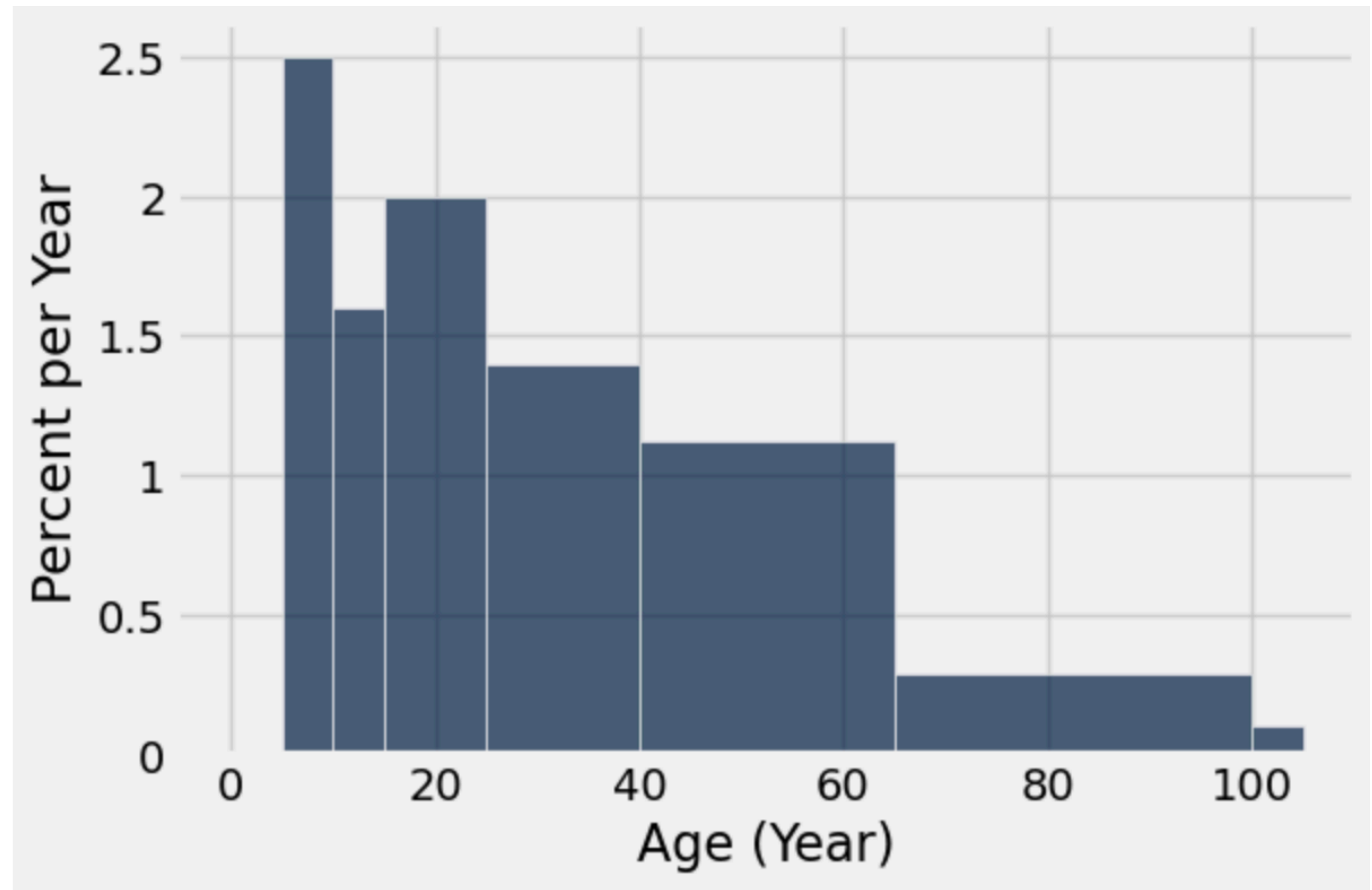


# Histograms

The **area** of each bar is a **percentage** of the whole

The **horizontal axis** is a numerical distribution - the bins don't need to be of equal size

The **vertical axis** is a rate (e.g., percent/year) - density

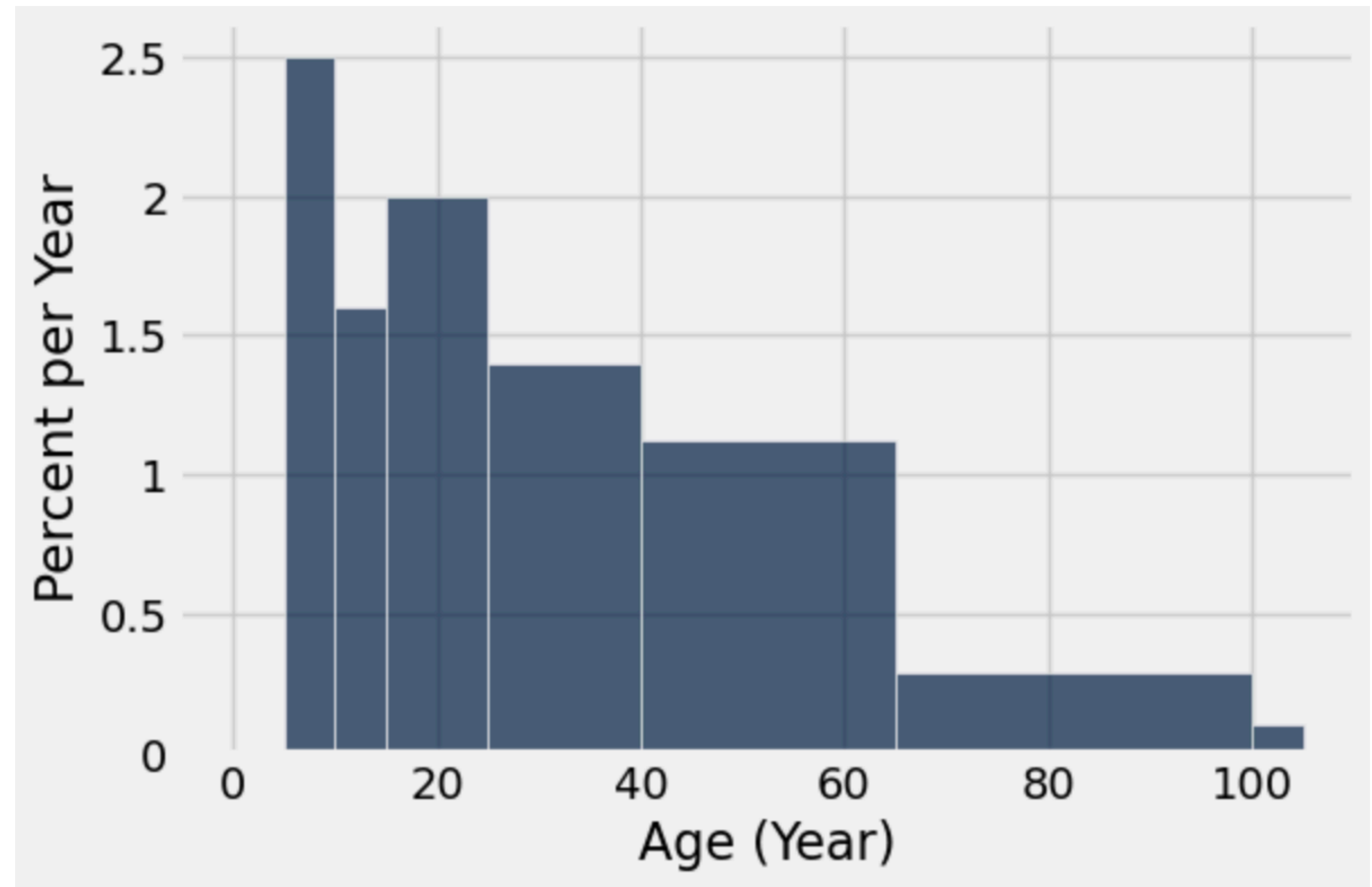


# Calculating Heights

The [40, 65) bin contains  
56/200 items

- The bin is 28% (56/200) of the whole
- The bin width is 65-40 = 25 years

- Height = 
$$\frac{28 \text{ percent}}{25 \text{ years}}$$
$$= 1.12\% \text{ per year}$$



# Area Notebook Demo

# Bar Chart vs Histogram

## Bar Chart

- Distribution of **categorical** variable
- Length of bars is proportional to the frequency / percent of individuals

## Histogram

- Distribution of **numerical** variable
- Horizontal axis is numerical, bins can be unequal
- **Area** of bars is proportional of percent of individuals, **height** measures density

# Next Class

- More charts