Introduction to Data Science

Lecture 5.2 Visualization demo

Instructor: Sangryul Jeon

School of Computer Science and Engineering

srjeonn@pusan.ac.kr

Import packages

Let figures appear in your colab(jupyter) notebook.

```
from datascience import *
import pandas as pd
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
```

See the style sheets reference here
https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html

Load CSV file – actors.csv

- Read a CSV file and see what's in there

```
path_data = "https://raw.githubusercontent.com/mlee-pnu/IDS/main/FDS07/"
actors = Table.read_table(path_data + 'actors.csv')
actors
```

Variable
aka feature,
attribute

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

... (47 rows omitted)

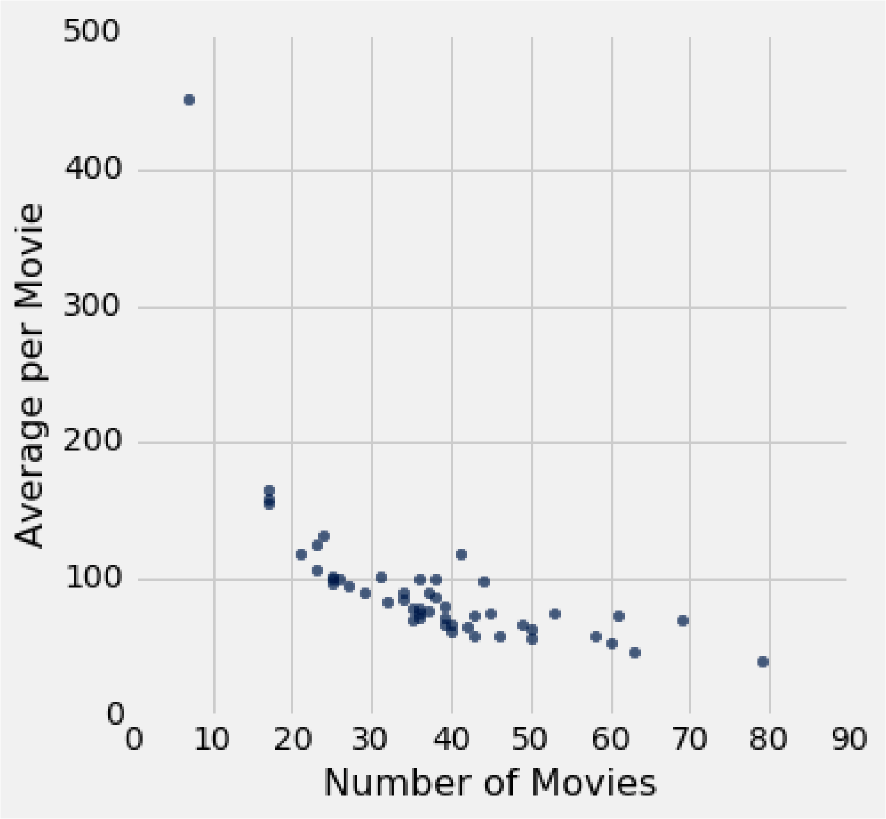
Categorical	Numerical	Numerical	Numerical	Categorical	Numerical

Data Description

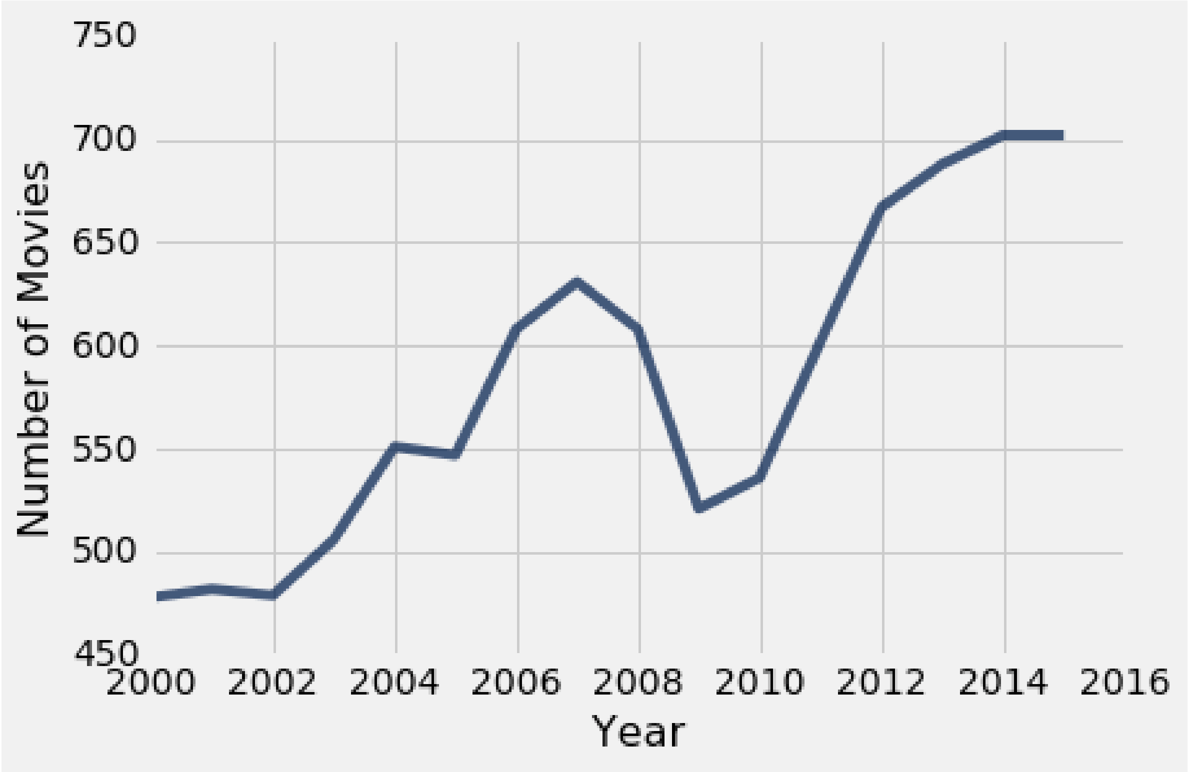
Column	Contents
Actor	Name of actor
Total Gross	Total gross domestic box office receipt, in millions of dollars, of all of the actor’s movies
Number of Movies	The number of movies the actor has been in
Average per Movie	Total gross divided by number of movies
#1 Movie	The highest grossing movie the actor has been in
Gross	Gross domestic box office receipt, in millions of dollars, of the actor’s #1 Movie

Plotting Two Numerical Variables

Scatter plot: `scatter`



Line graph: `plot`



Scatter Plot

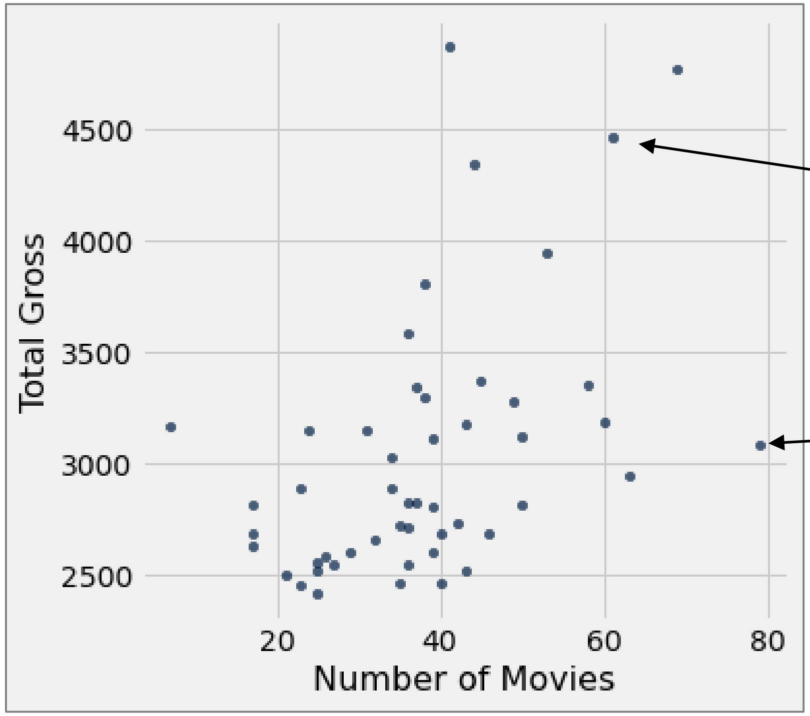
- A *scatter plot* displays the relation between two numerical variables.
- Use Table.scatter() method

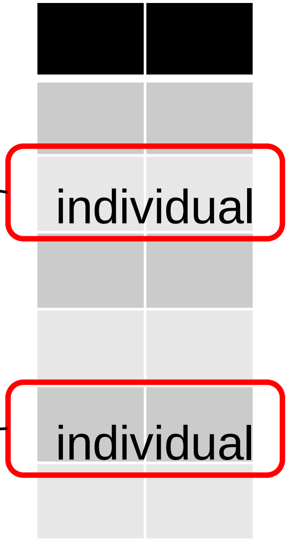
```
actors.scatter('Number of Movies', 'Total Gross')
```

x-axis y-axis

how many points?

any associations?



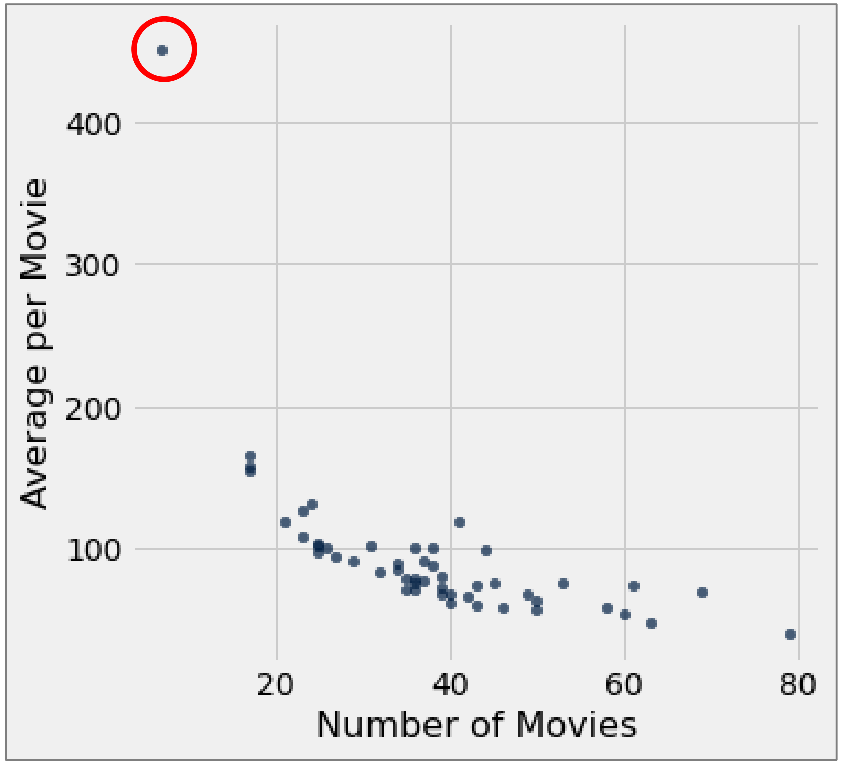


Scatter Plot (cont.)

- Number of Movies vs. Average per Movie

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

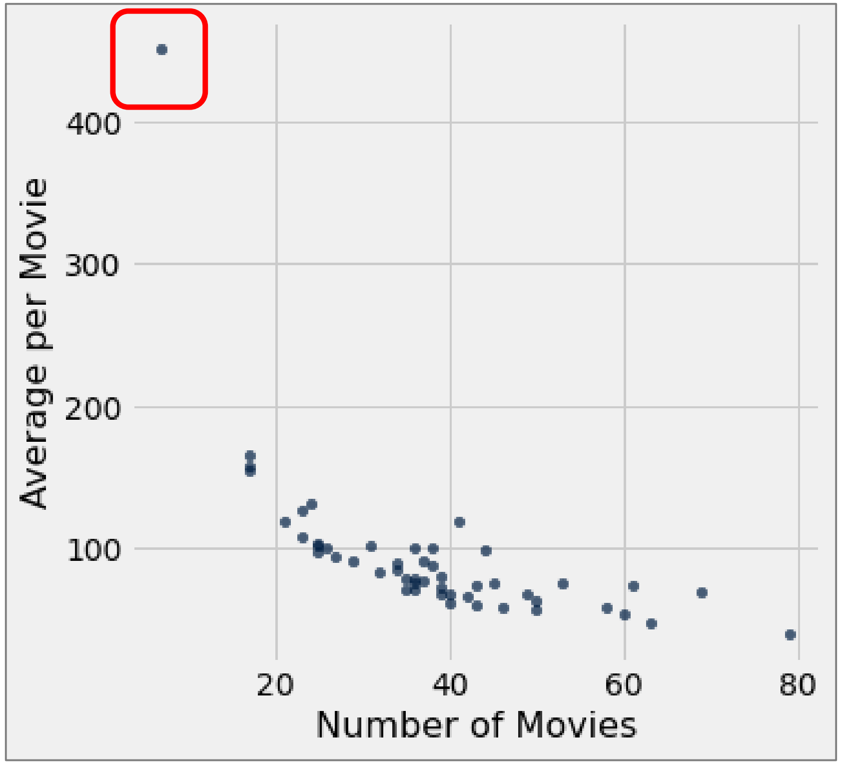
any associations?

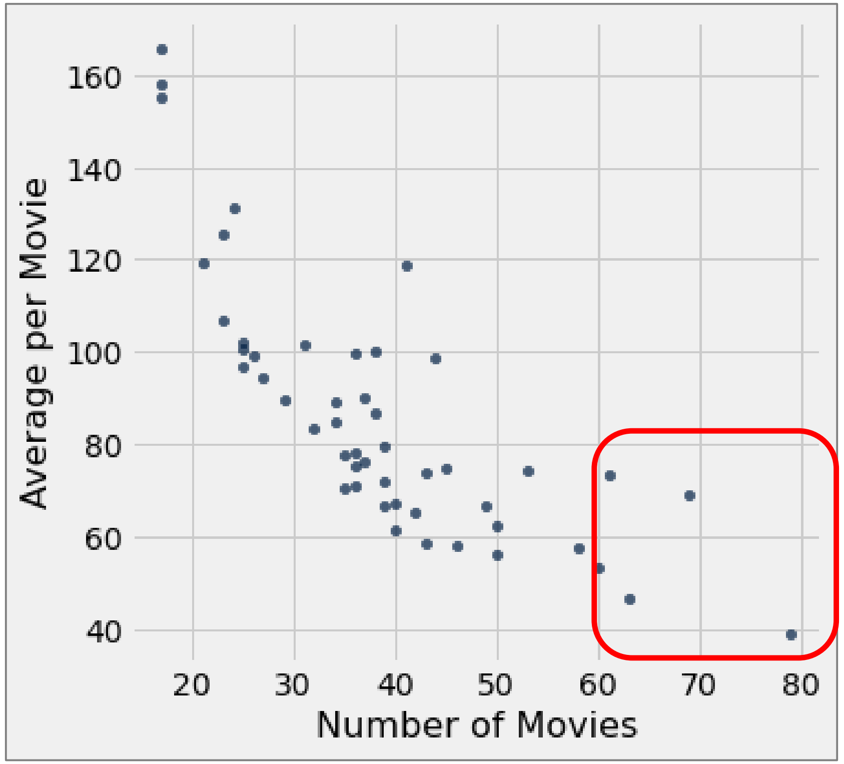


Scatter Plot (cont.)

- Let’s look at the portion that doesn’t have the outlier.

```
no_outlier = actors.where('Number of Movies', are.above(10))
no_outlier.scatter('Number of Movies', 'Average per Movie’)
```





Who are they?

Identifying actors

```
actors.where('Number of Movies', are.above(60))
```

```
actors.where('Number of Movies', are.below(10))
```

```
actors.where('Number of Movies', are.not_between_or_equal_to(10, 60))
```

Actor	Total Gross	Number of Movies	Average per Movie	#1 Movie	Gross
Samuel L. Jackson	4772.8	69	69.2	The Avengers	623.4
Morgan Freeman	4468.3	61	73.3	The Dark Knight	534.9
Anthony Daniels	3162.9	7	451.8	Star Wars: The Force Awakens	936.7
Robert DeNiro	3081.3	79	39	Meet the Fockers	279.3
Liam Neeson	2942.7	63	46.7	The Phantom Menace	474.5




Line Plot

- Line plots are often used to study chronological trends and patterns.
- Let's take a look at *movies_by_year* data

```
movies_by_year = Table.read_table(path_data + 'movies_by_year.csv')
movies_by_year.show(3)
```

Year	Total Gross	Number of Movies	#1 Movie
2015	11128.5	702	Star Wars: The Force Awakens
2014	10360.8	702	American Sniper
2013	10923.6	688	Catching Fire

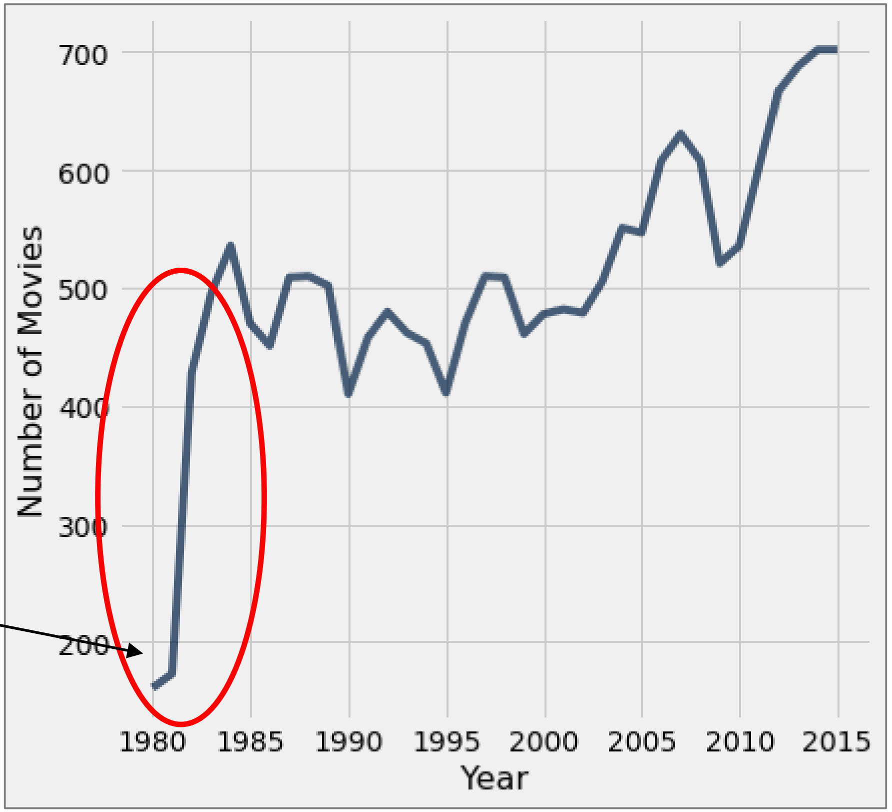
... (33 rows omitted)

Column	Content
Year	Year
Total Gross	Total domestic box office gross, in millions of dollars, of all movies released
Number of Movies	Number of movies released
#1 Movie	Highest grossing movie

Line Plot (cont.)

- Number of Movies by Year

```
movies_by_year.plot('Year', 'Number of Movies')
```

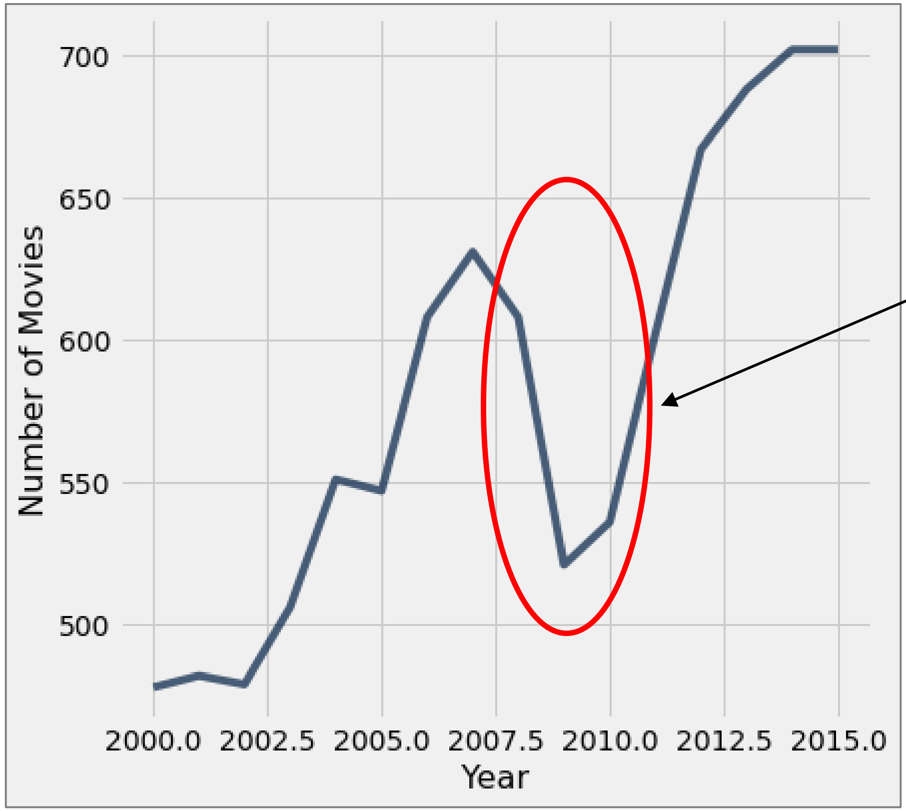


Production-driven
industry started

Line Plot (cont.)

- let's focus on the 21st century only

```
century_21 = movies_by_year.where('Year', are.above(1999))
century_21.plot('Year', 'Number of Movies')
```

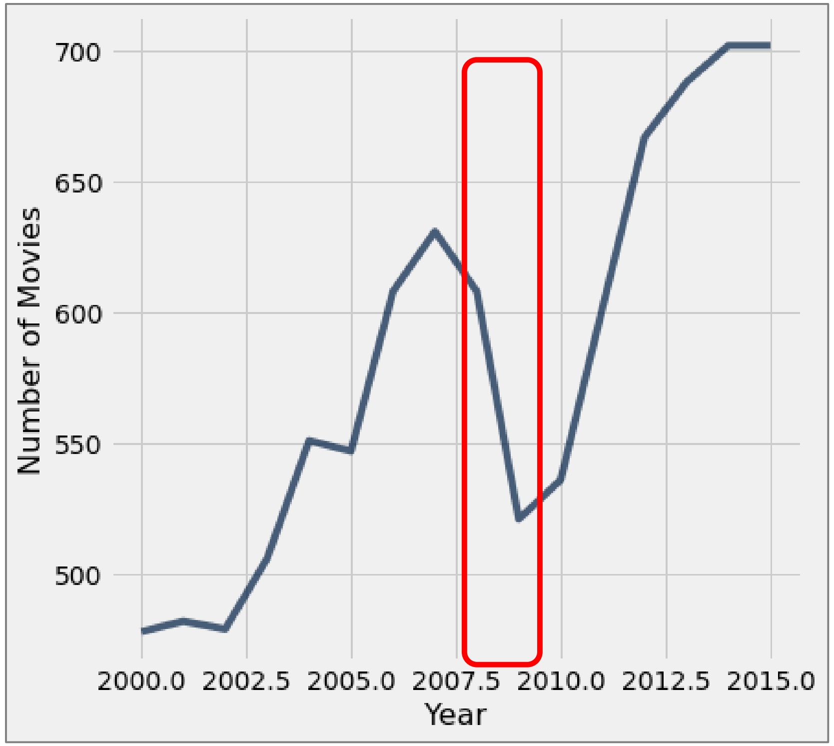


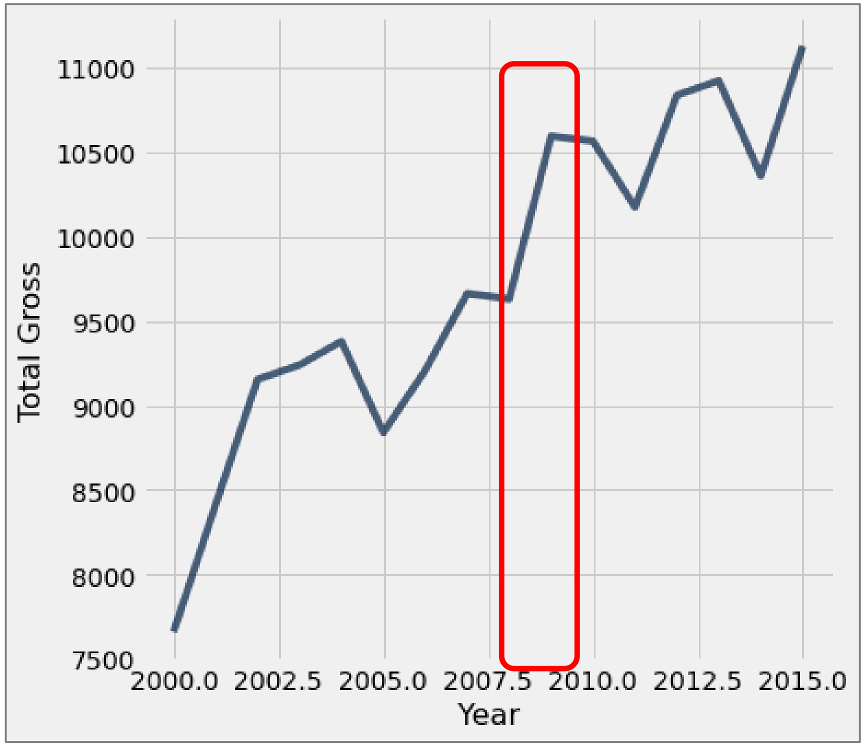
The global financial crisis of 2008

Line Plot (cont.)

- Plot the total gross by year and see if there're any unlikely patterns.

```
century_21.plot('Year', 'Total Gross')
```





why?

```
century_21.where('Year', are.equal_to(2009))
```

VISUALIZING CATEGORICAL DISTRIBUTIONS

Bar Chart

- Make a table with the number of cartoons of each flavor of ice cream.

```
icecream = pd.DataFrame({
     'Flavor': np.array(['Chocolate', 'Strawberry', 'Vanilla']),
     'Number of Cartons': np.array([16, 5, 9])
})

icecream
```

Flavor	Number of Cartons
Chocolate	16
Strawberry	5
Vanilla	9

Category

Frequency

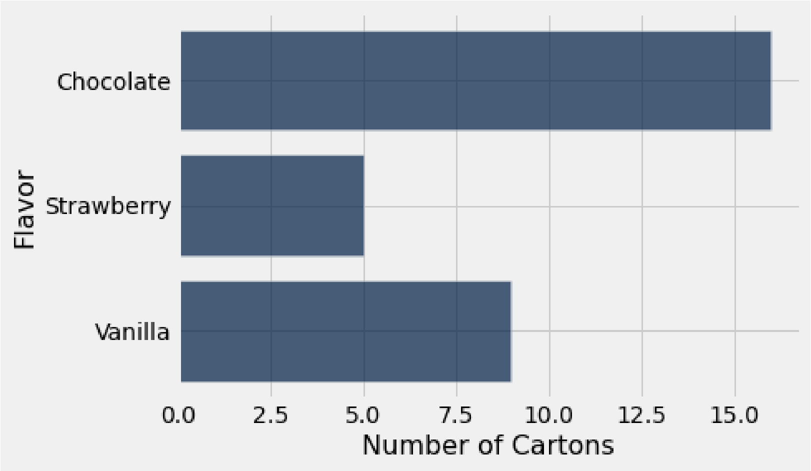
Bar Chart (cont.)

- The bar chart displays a bar for each category.
  - The bars are equally spaced and equally wide. The length of each bar is proportional to the frequency of the corresponding category.
  - Use plt.barh() for horizontal bar chart

```
icecream.barh('Flavor', 'Number of Cartons')
```

category frequency

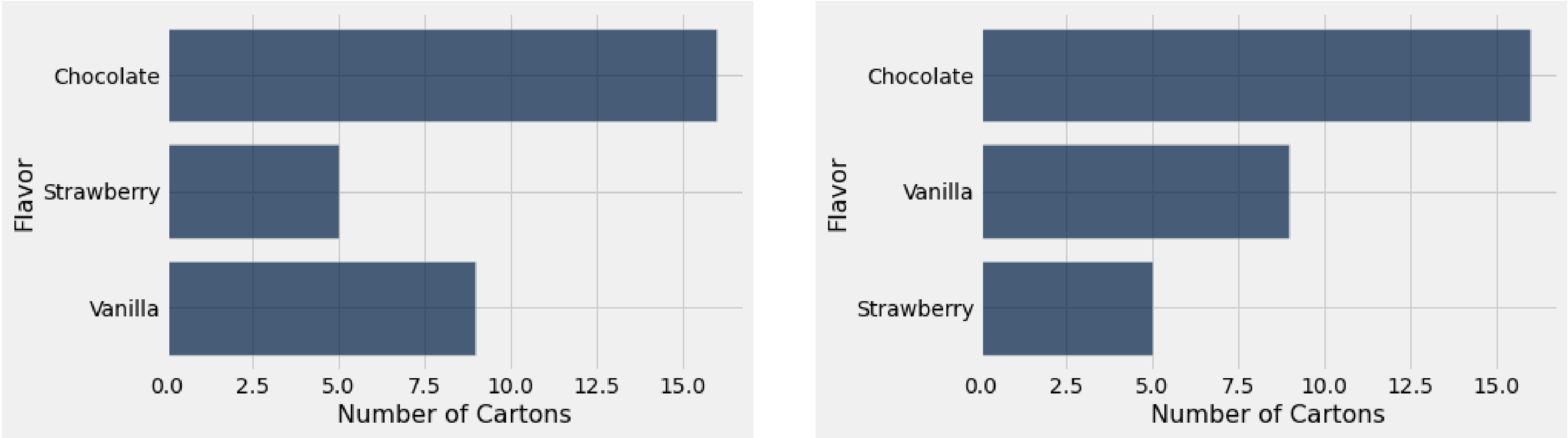
```
icecream.barh('Flavor')
```



Bar Chart (cont.)

- Scatter and line plots take two quantitative variables, while bar chart takes qualitative(categorical) and qualitative(numeric) variables.
- The width of each bar and the space between consecutive bars is entirely up to the person who is producing the graph
- The bars can be drawn in any order

```
icecream.sort_values(by=['Number of Cartons'])
```



Grouping Categorical Data

- top_movies_2017 data

```
top = Table.read_table(path_data + 'top_movies_2017.csv')
```

Title	Studio	Gross	Gross (Adjusted)	Year
Gone with the Wind	MGM	198676459	1796176700	1939
Star Wars	Fox	460998007	1583483200	1977
The Sound of Music	Fox	158671368	1266072700	1965
E.T.: The Extra-Terrestrial	Universal	435110554	1261085000	1982
Titanic	Paramount	658672302	1204368000	1997
The Ten Commandments	Paramount	65500000	1164590000	1956
Jaws	Universal	260000000	1138620700	1975
Doctor Zhivago	MGM	111721910	1103564200	1965
The Exorcist	Warner Brothers	232906145	983226600	1973
Snow White and the Seven Dwarves	Disney	184925486	969010000	1937

... (190 rows omitted)

Grouping Categorical Data (cont.)

- Aggregate the number of movies released by each studio

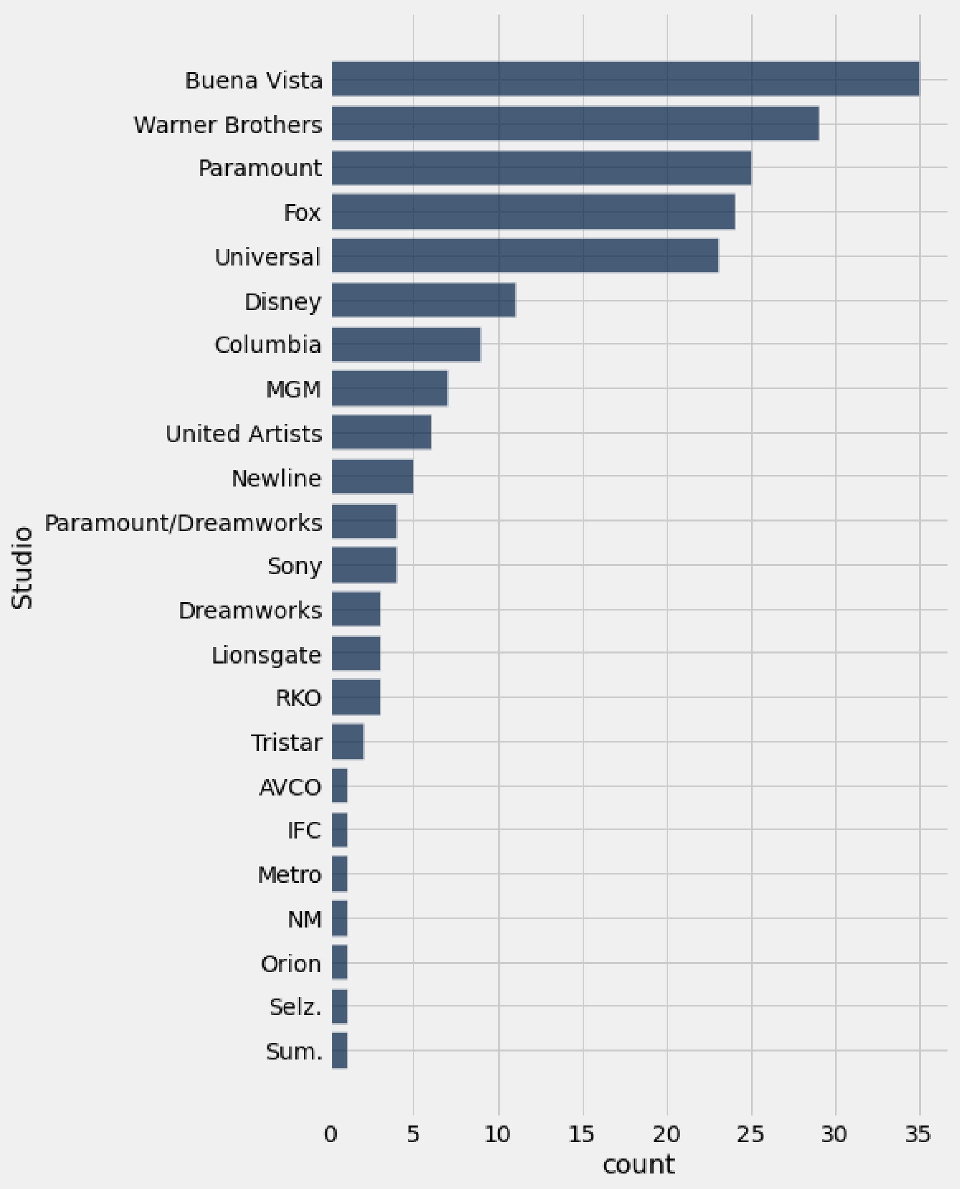
```
movies_and_studios = top.select('Title', 'Studio')
studio_distribution = movies_and_studios.group('Studio')
```

```
sum(studio_distribution.column('count'))
```

Studio	count
AVCO	1
Buena Vista	35
Columbia	9
Disney	11
Dreamworks	3
Fox	24
IFC	1
Lionsgate	3
MGM	7
Metro	1

... (13 rows omitted)

Grouping Categorical Data (cont.)



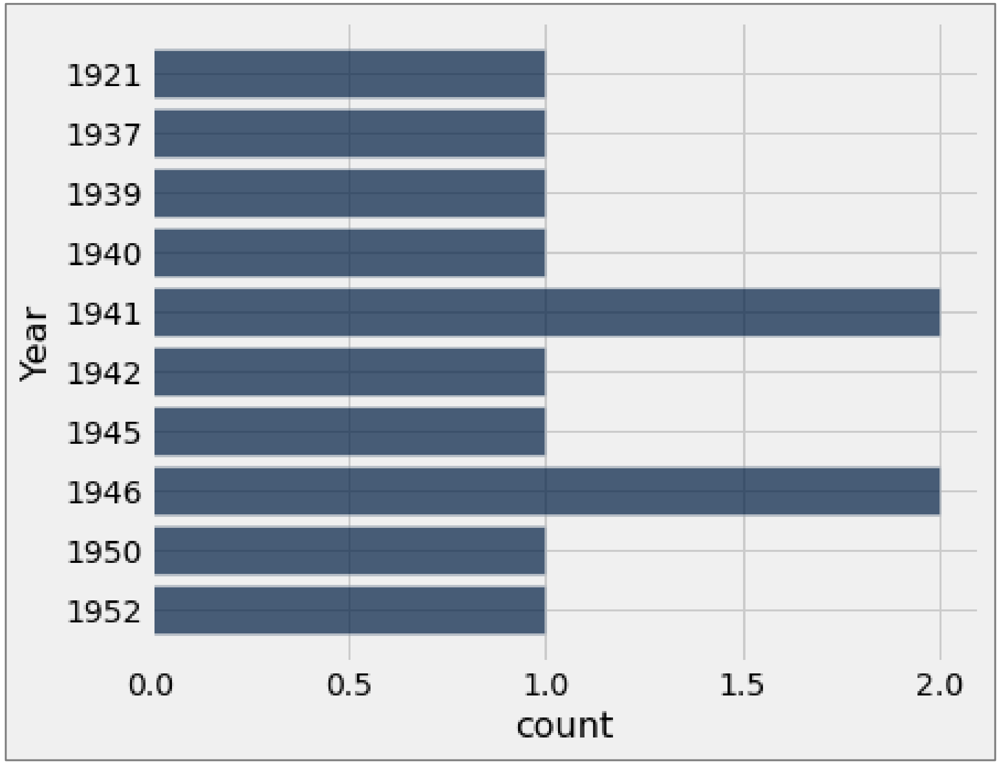
- Draw a bar chart in descending order

```
studio_distribution.sort('count',
descending=True).barh('Studio')
```

Grouping Numerical Data

- Can we aggregate the number of movies released by year?

```
movies_and_years = top.select('Title', 'Year')
movies_and_years.group('Year').take(np.arange(10)).barh('Year')
```

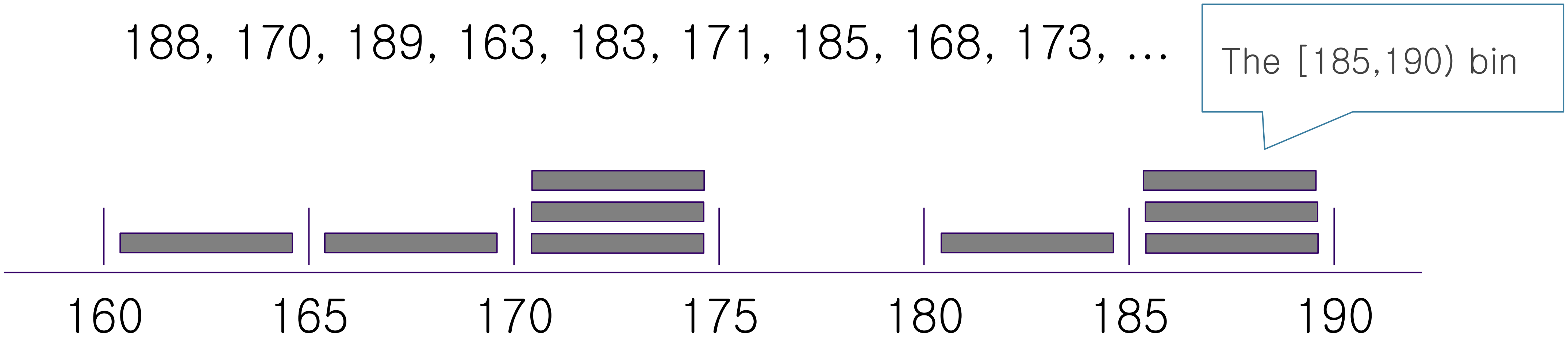


Any issues?
Any solutions?

Grouping Numerical Data (cont.)

- Binning is counting the number of numerical values that lie within rang es, called bins.
  - Bins are defined by their lower bounds (inclusive)
  - The upper bound is the lower bound of the next bin

188, 170, 189, 163, 183, 171, 185, 168, 173, …



Binning the Data

- USA's top-grossing movies, with adjusted gross to 2016 dollar value.

```
top = pd.read_csv(path_data+'top_movies_2017.csv')

millions = pd.DataFrame({'Title': top['Title'],
                         'Adjusted Gross': np.round(top['Gross Adjusted)']/1e6,
2)})
millions
```

Title	Studio	Gross	Gross (Adjusted)	Year
Gone with the Wind	MGM	198,676,459	1,796,176,700	1939
Star Wars	Fox	460,998,007	1,583,483,200	1977
The Sound of Music	Fox	158,671,368	1,266,072,700	1965
E.T.: The Extra-Terrestrial	Universal	435,110,554	1,261,085,000	1982
Titanic	Paramount	658,672,302	1,204,368,000	1997
The Ten Commandments	Paramount	65,500,000	1,164,590,000	1956
Jaws	Universal	260,000,000	1,138,620,700	1975
Doctor Zhivago	MGM	111,721,910	1,103,564,200	1965
The Exorcist	Warner Brothers	232,906,145	983,226,600	1973
Snow White and the Seven Dwarves	Disney	184,925,486	969,010,000	1937

... (190 rows omitted)

Title	Adjusted Gross
Gone with the Wind	1796.18
Star Wars	1583.48
The Sound of Music	1266.07
E.T.: The Extra-Terrestrial	1261.08
Titanic	1204.37
The Ten Commandments	1164.59
Jaws	1138.62
Doctor Zhivago	1103.56
The Exorcist	983.23
Snow White and the Seven Dwarves	969.01

... (190 rows omitted)

Binning the Data (cont.)

- Check the range of the data before making class intervals for the frequ ency distribution table.

```
adj_gross = millions['Adjusted Gross']
min(adj_gross), max(adj_gross)
```

- Set class intervals to encompass the range of the data.

```
bins = pd.cut(millions['Adjusted Gross'], bins_range,
right=False).value_counts().reset_index(name='Adjusted Gross Count')
```

Class interval aka bin width.

bin	Adjusted Gross count
300	68
400	60
500	32
600	15

... (14 rows omitted)

bin

bin

Frequency of data within the range [300,400)

Frequency distribution table

Binning the Data (cont.)

- Or you can specify the number of bins. Default value is 10.
- Note that the last class interval is [a, b], i.e., includes b.

```
bins = pd.cut(millions['Adjusted Gross'],
11).value_counts().reset_index(name='Adjusted Gross Count')
bins.rename(columns={'index': 'bins'}, inplace=True)
```

bin	Adjusted Gross count
338.41	177
702.852	15
1067.3	6
1431.74	2
1796.18	0

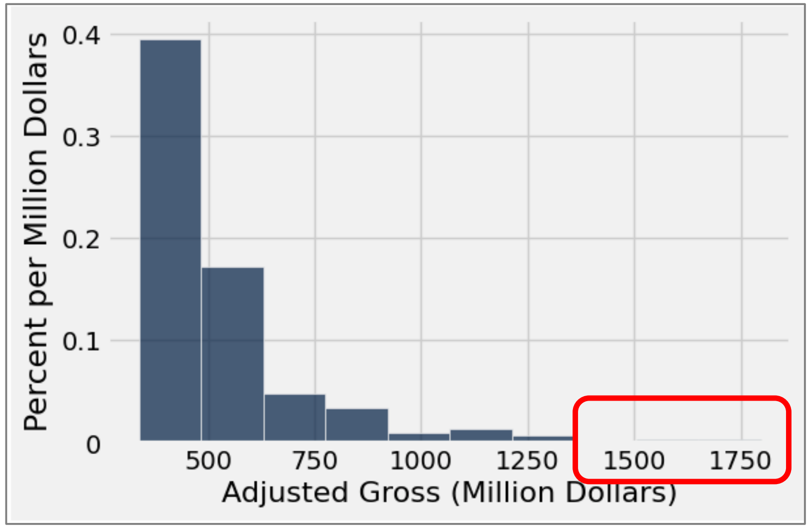
last class [1431.74, 1796.18]

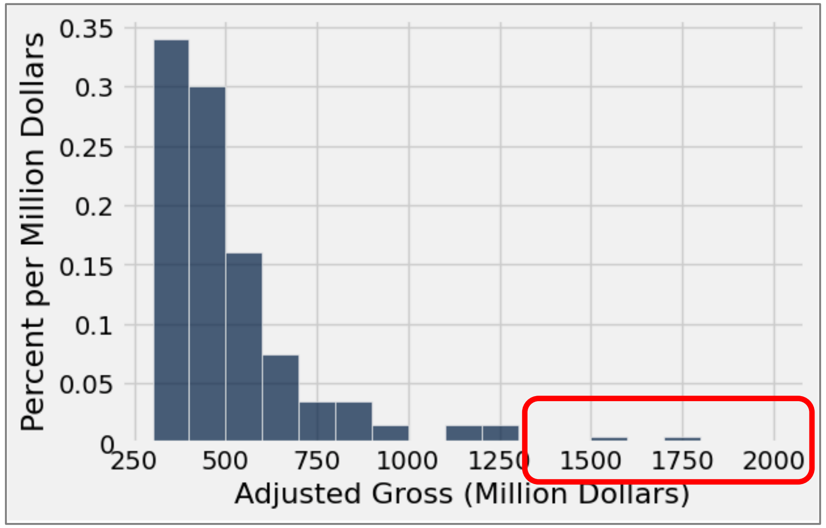
Histogram

- A *histogram* is a visualization of the distribution of a quantitative varia ble.

```
millions.hist('Adjusted Gross', unit="Million Dollars")
```

```
millions.hist('Adjusted Gross',
bins=np.arange(300,2001,100), unit="Million Dollars")
```





skewed to the right, or, less formally, having *a long right hand tail*

General Principles

- Histogram
  - The bins are drawn to scale and are contiguous (though some might be empty)
  - The **area** of each bar is proportional to the number of entries in the bin

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

  - Thus, $\text{height of bar} = \dfrac{\text{area of bar}}{\text{width of bin}} = \dfrac{\text{percent of entries in bin}}{\text{width of bin}}$

  - Density scale
    - The total area of all the bars in the histogram is 100%, or “sum to 1”

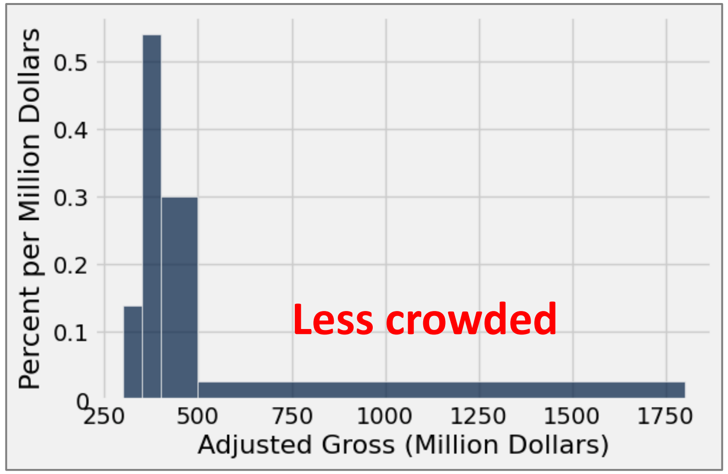
Density scale vs. counts

```
uneven = make_array(300, 350, 400, 500, 1800)
millions.hist('Adjusted Gross', bins=uneven, unit="Million Dollars")
```

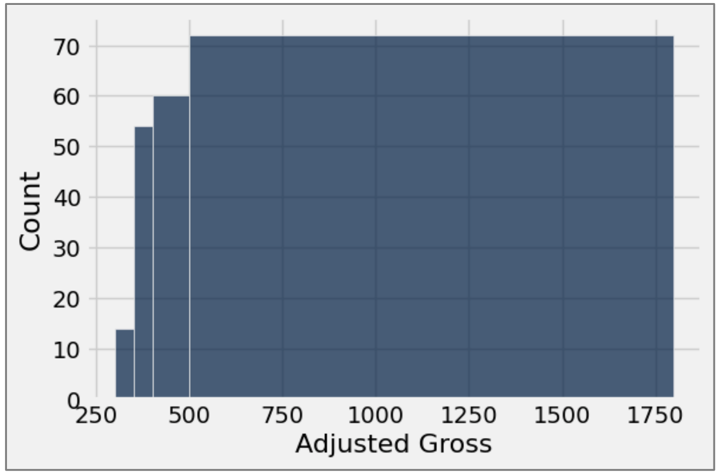
```
millions.hist('Adjusted Gross', bins=uneven, normed=False)
```

```
millions.bin('Adjusted Gross', bins=uneven)
```

bin	Adjusted Gross count
300	14
350	54
400	60
500	72
1800	0



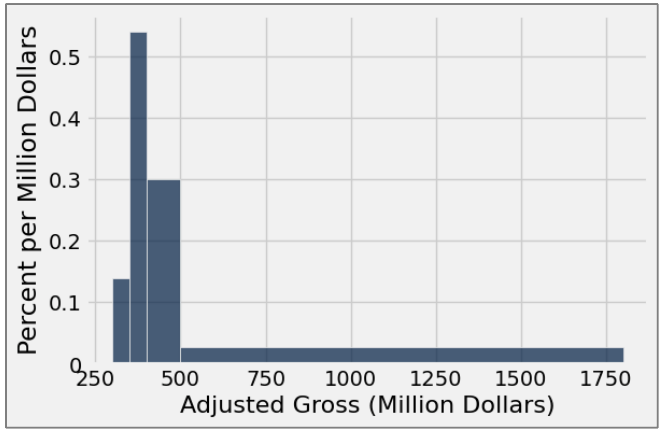
HISTOGRAM

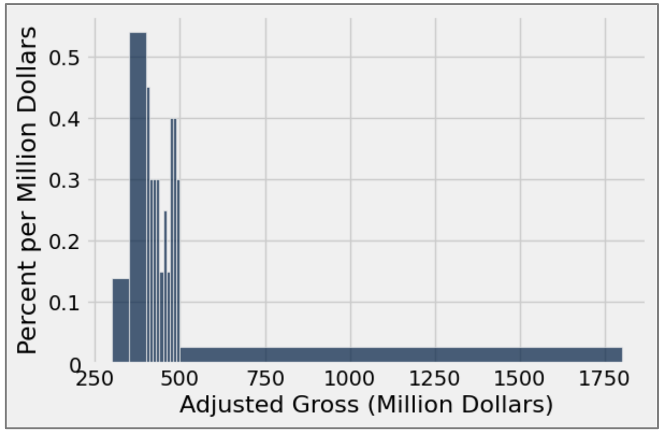


NOT A HISTOGRAM

Level of Detail

```
some_tiny_bins = make_array(
300, 350, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 1800)
millions.hist('Adjusted Gross', bins=some_tiny_bins, unit='Million Dollars')
```





- [400, 500) bin is $100M width and contain 30% of the data. Thus, 0.3% per $1M.
- a rough approximation for the hights of each of 100 skinny bins with $1M width in [400,500)

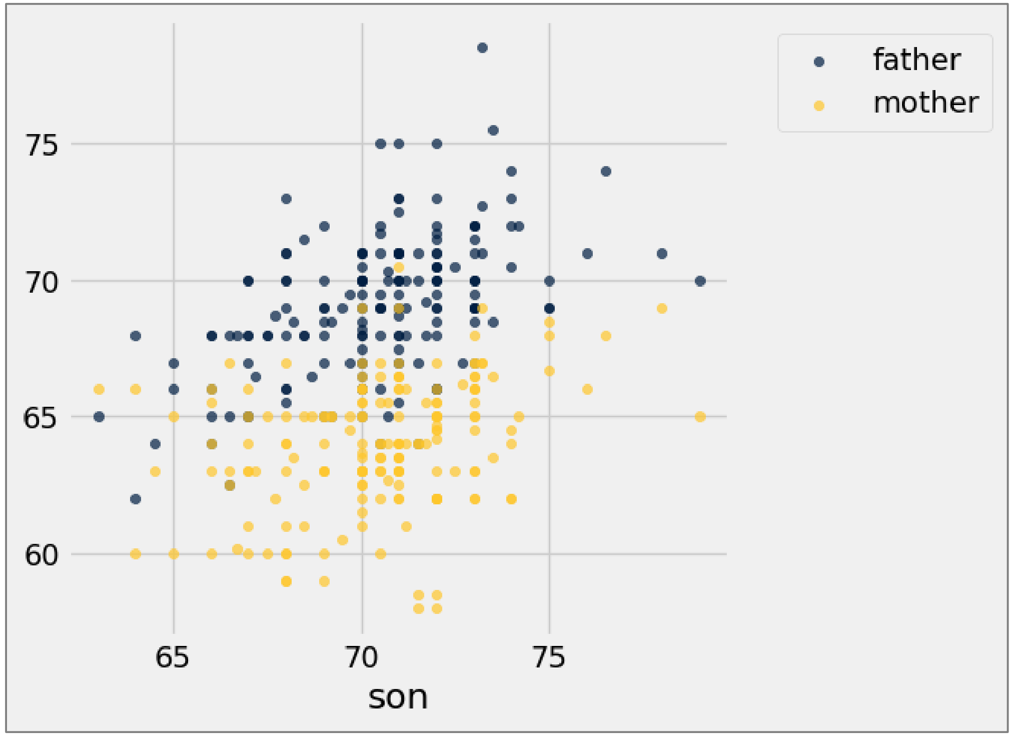
Overlaid Scatter Plot

```
heights = pd.read_csv(path_data+'sons_heights.csv')
plt.scatter(x,y, color='blue', alpha=0.5, label='father')
plt.scatter(x,z, color='yellow', alpha=0.5, label='mother')
```

father	mother	son
78.5	67	73.2
75.5	66.5	73.5
75	64	71
75	64	70.5

... (175 rows omitted)

Scale shared!



Overlaid Line Plot

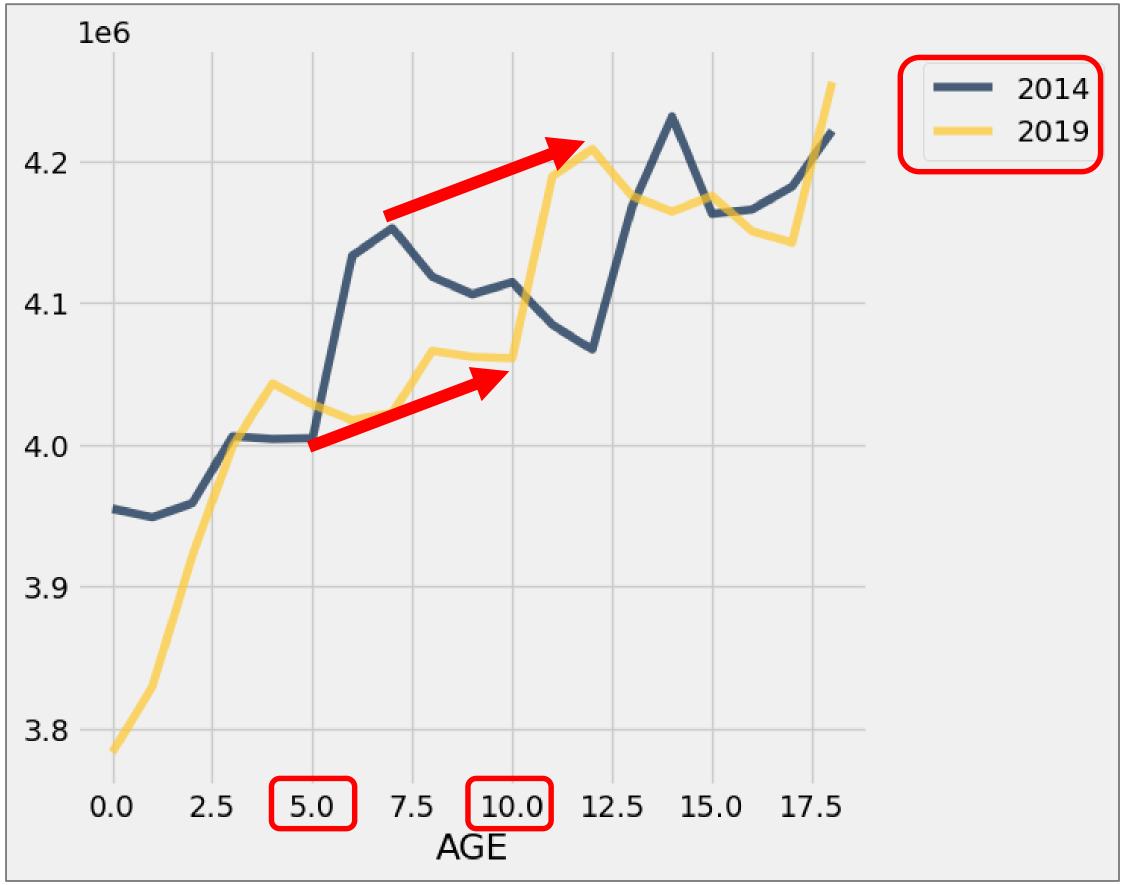
```
# Read the full Census table
data = 'http://www2.census.gov/programs-surveys/popest/technical-documentation/file-layouts/2010-2019/nc-est2019-agesex-res.csv'
full_census_table = pd.read_csv(data)

# Extract four columns from full_census_table.
partial_census_table = full_census_table[['SEX', 'AGE', 'POPESTIMATE2019',
'POPESTIMATE2014']]

# Rename two columns
us_pop = partial_census_table.rename(columns={'POPESTIMATE2019': '2019',
'POPESTIMATE2014': '2014'}, inplace=False)

# Access the rows corresponding to all children, ages 0-18, sex 0 (male & female)
filter1 = us_pop['AGE'] <= 18
filter2 = us_pop['SEX'] == 0
children = us_pop.loc[filter1 & filter2]

# Drop column 'SEX'
children.drop(columns=['SEX'], inplace=True)
children
```

0: all, 1: male, 2: female

Overlaid Line Plot (cont.)

AGE	2014	2019
0	3954787	3783052
1	3948891	3829599
2	3958711	3922044
3	4005928	3998665
4	4004032	4043323

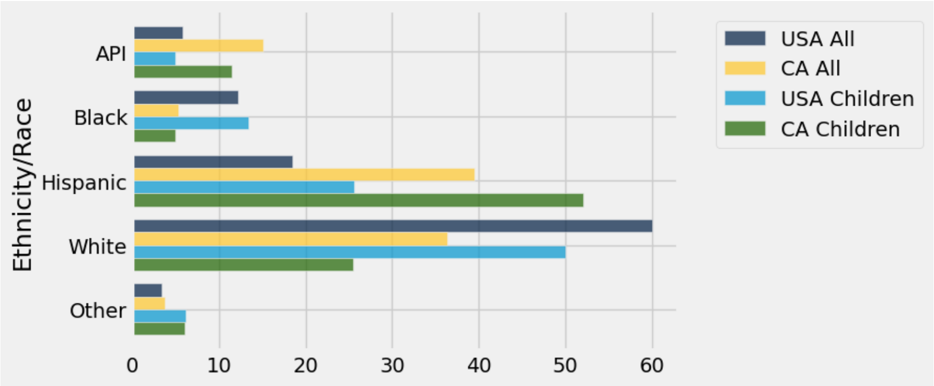
... (14 rows omitted)



Grouped Bar Charts

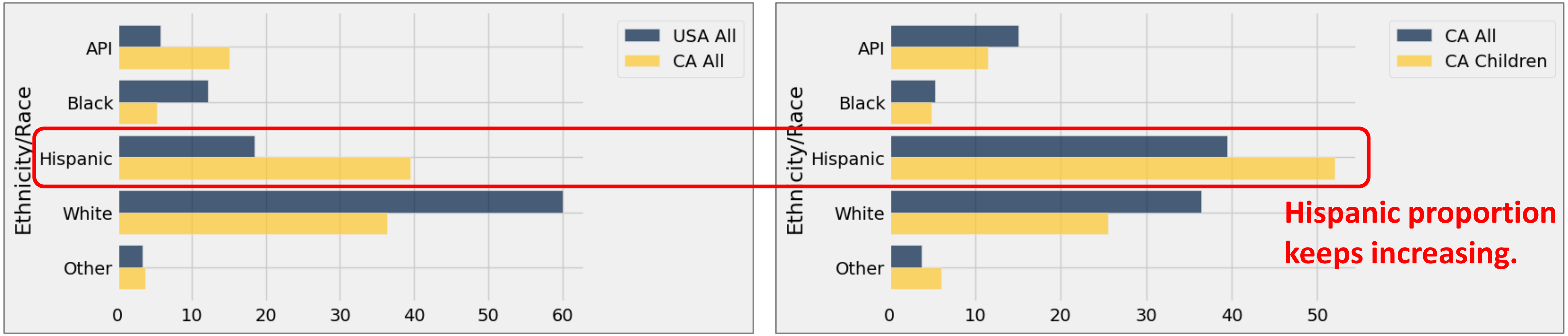
```
usa_ca = pd.read_csv(path_data+'usa_ca_2019.csv')
```

Ethnicity/Race	USA All	CA All	USA Children	CA Children
API	5.8	15.1	4.9	11.5
Black	12.2	5.3	13.4	4.9
Hispanic	18.5	39.5	25.6	52.1
White	60.1	36.4	50	25.5
Other	3.4	3.7	6.1	6




```
new_usa_ca = usa_ca.set_index(keys=['Ethnicity/Race'], inplace=False)
```

```
new_usa_ca = new_usa_ca[['USA All','CA All']]
```



Hispanic proportion keeps increasing.

Q&A

Prof. Jeon, Sangryul

Computer Vision Lab.

Pusan National University, Korea

Tel: +82-51-510-2423

Web: http://sr-jeon.github.io/

E-mail: srjeonn@pusan.ac.kr