Homework 6

due Wed Oct 26 at start of class

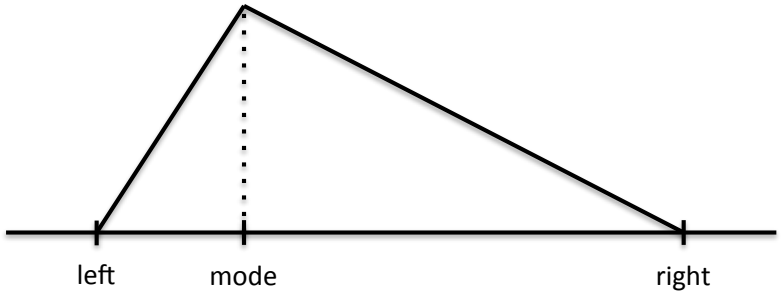
`Homework6.pdf`

(posted soon)

Homework 7

due Wed Nov 2 at start of class

`Homework7.pdf`



as noted last week, **Q1** will ask you to generate random numbers drawn from a triangular distribution
(a) asks you to plot the distribution, **(b)** asks you to draw samples from the built-in prng, **(c)** asks you to draw samples using rejection sampling, **(d)** asks you to draw samples using the Metropolis algorithm

download from Brightspace

`Visualization.ipynb`

Metropolis algorithm (symmetric proposal distribution)

Let $p(x)$ be the desired probability distribution, (for example, a triangular distribution); note that the Wikipedia page formalizes using $f(x)$, a function proportional to $p(x)$, which is the approach used, for example, to generate random numbers for a Bayesian posterior distribution.

1. Initialization:

- Choose an arbitrary point $x0$ to be the first sample. The first point in the chain xt will be equal to $x0$. If you know the range of the distribution, it makes sense to pick a first sample $x0$ that is within the range of the distribution.
- Choose a proposal distribution $g(xp|xt)$, where xt is the current point in the chain and xp is the proposed point in the chain. For the Metropolis algorithm, the proposal distribution needs to be symmetric (Metropolis-Hastings generalizes to allow for non-symmetric proposal distributions). The simplest choice for $g(xp|xt)$ is to assume that xp is drawn from a normal distribution with mean xt and standard deviation σ; σ should be small, but not too small. In python, to generate a random number with mean xt and standard deviation σ, you would use the following code:

```
xp = R.normal(xt, sigma)
```

2. For each iteration:

- Generate: Generate a candidate xp from $g(xp|xt)$ using the above.
- Calculate: Calculate the acceptance ratio $A = p(xp)/p(xt)$, which is used to decide whether to accept or reject the proposed candidate.
- Accept or Reject:
  - Generate a uniform random number U on [0,1) using `R.uniform()`.
  - If $U <= A$ accept the proposed candidate by setting $xt = xp$.
  - If $U > A$ reject the proposed candidate and keep xt at the same number.

Each new accepted xt produced on an iteration is a candidate random number produced by the Metropolis algorithm.

https://en.wikipedia.org/wiki/Metropolis–Hastings_algorithm

$$P_{move} = \min\left(\frac{p(\theta_{proposed})}{p(\theta_{current})}, 1\right)$$

Metropolis Algorithm (Metropolis et al., 1953)

One important use of MCMC techniques
is Bayesian statistical analysis.

Bayes is foundational to data science, statistics, biostatistics, machine learning, and lots of other applications.

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{p(D)}$$

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{\sum_i p(D \mid \theta_i)\, p(\theta_i)}$$

Bayes' Rule

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{p(D)}$$

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{\int p(D \mid \theta')\, p(\theta')\, d\theta'}$$

in many cases, these integrals are impossible to solve analytically and intractable to solve using numerical methods

Bayes' Rule

Metropolis Algorithm (Metropolis et al., 1953)

$$P_{move} = \min\left(\frac{p(\theta_{proposed})}{p(\theta_{current})}, 1\right)$$

accept with P_{move}
reject with $1\text{-}P_{move}$

Metropolis Algorithm (Metropolis et al., 1953)

$$P_{move} = \min\left(\frac{p(\theta_{proposed})}{p(\theta_{current})}, 1\right)$$

accept with P_{move}
reject with $1\text{-}P_{move}$

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{\int p(D \mid \theta')\, p(\theta')\, d\theta'}$$

you still have this pesky integral … or do you?

imagine you want to generate random samples from p(θ|D) using an MCMC approach like Metropolis

if you can generate random samples, you can calculate things like the expected value or do various statistical tests

Metropolis Algorithm (Metropolis et al., 1953)

$$P_{move} = \min\left(\frac{p(\theta_{proposed})}{p(\theta_{current})}, 1\right)$$

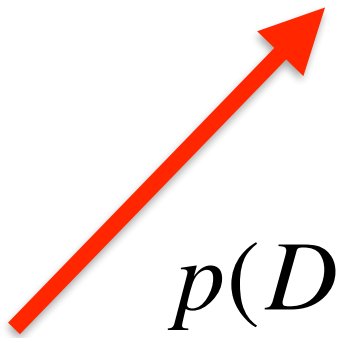
accept with P_{move}
reject with $1\text{-}P_{move}$

$$\frac{p(\theta_{proposed} \mid D)}{p(\theta_{current} \mid D)}$$

Metropolis Algorithm (Metropolis et al., 1953)

$$P_{move} = \min\left(\frac{p(\theta_{proposed})}{p(\theta_{current})}, 1\right)$$

accept with P_{move}
reject with $1\text{-}P_{move}$



$$\frac{p(\theta_{proposed} \mid D)}{p(\theta_{current} \mid D)} = \frac{\dfrac{p(D \mid \theta_{proposed})p(\theta_{proposed})}{\int p(D \mid \theta')d(\theta')}}{\dfrac{p(D \mid \theta_{current})p(\theta_{current})}{\int p(D \mid \theta')d(\theta')}}$$

Metropolis Algorithm (Metropolis et al., 1953)

$$P_{move} = \min\left(\frac{p(\theta_{proposed})}{p(\theta_{current})}, 1\right)$$

accept with P_{move}
reject with $1\text{-}P_{move}$

$$\frac{p(\theta_{proposed} \mid D)}{p(\theta_{current} \mid D)} = \frac{\dfrac{p(D \mid \theta_{proposed})\,p(\theta_{proposed})}{\int p(D \mid \theta')\,d(\theta')}}{\dfrac{p(D \mid \theta_{current})\,p(\theta_{current})}{\int p(D \mid \theta')\,d(\theta')}}$$

Metropolis Algorithm (Metropolis et al., 1953)

$$P_{move} = \min\left(\frac{p(\theta_{proposed})}{p(\theta_{current})}, 1\right)$$

accept with P_{move}
reject with $1\text{-}P_{move}$

$$\frac{p(\theta_{proposed} \mid D)}{p(\theta_{current} \mid D)} = \frac{p(D \mid \theta_{proposed})p(\theta_{proposed})}{p(D \mid \theta_{current})p(\theta_{current})}$$

Gibbs Sampling is more efficient that Metropolis (BUGS, JAGS)

Hamiltonian MCMC is more efficient than Gibbs Sampling (Stan)

Visualization in Python

matplotlib and related packages

Python Data Science Handbook, Jake VanderPlas
Chapter 4: Visualization With Matplotlib

Matplotlib Tutorials (review Usage Guide and Pyplot Tutorial)

https://matplotlib.org/stable/users/index.html

Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.

Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.

Paris, le 20 Novembre 1869.

Les nombres d'hommes présents sont représentés par les largeurs des zônes colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en travers des zônes. Le rouge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. — Les renseignements qui ont servi à dresser la carte ont été puisés dans les ouvrages de M.M. Thiers, de Ségur, de Fezensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre.

Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davoust qui avaient été détachés sur Minsk et Mobilow et ont rejoint vers Orscha et Witebsk, avaient toujours marché avec l'armée.

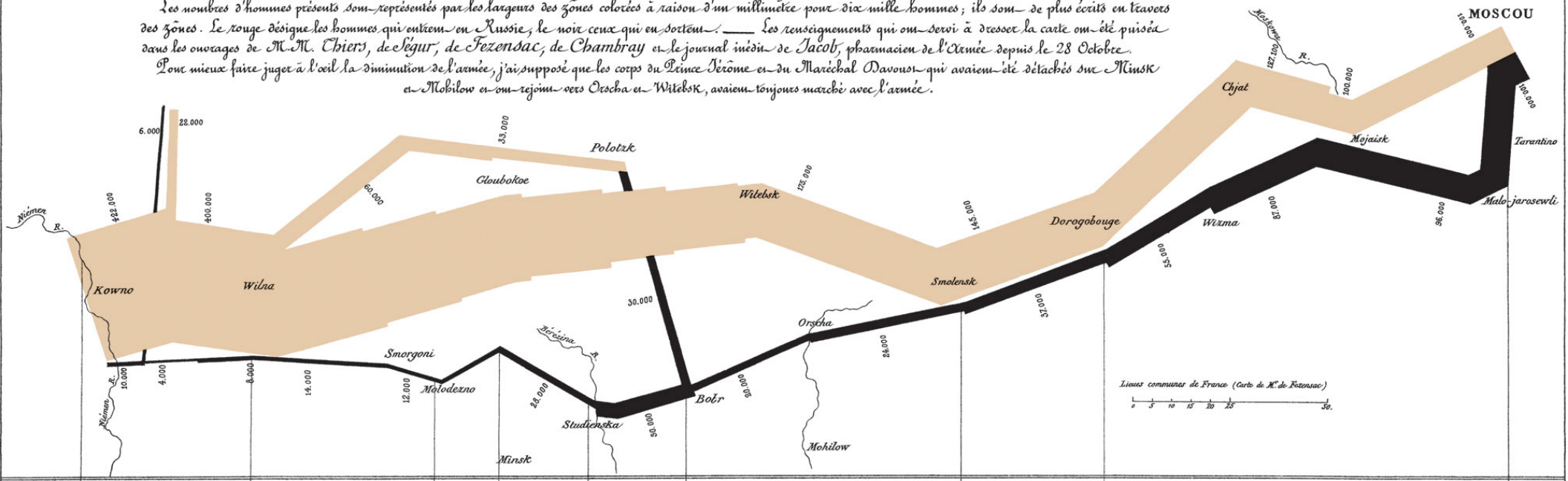
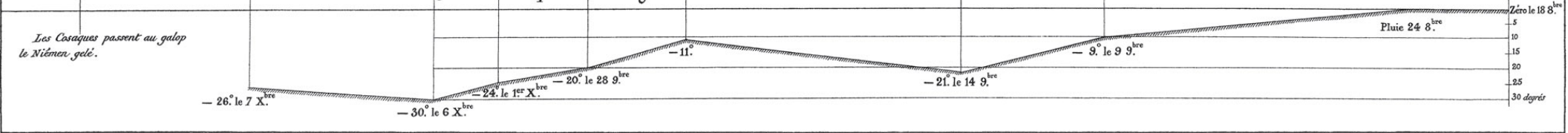


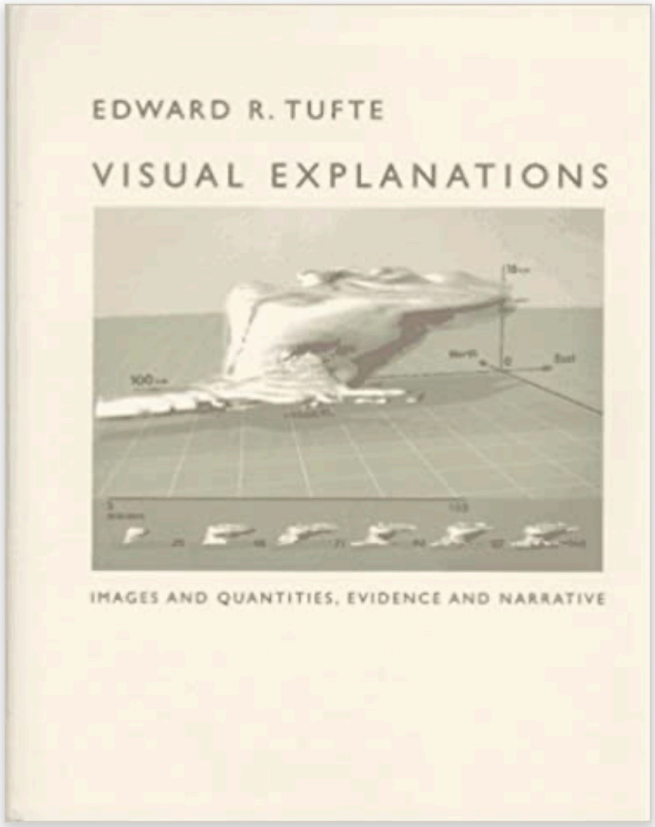
TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro.

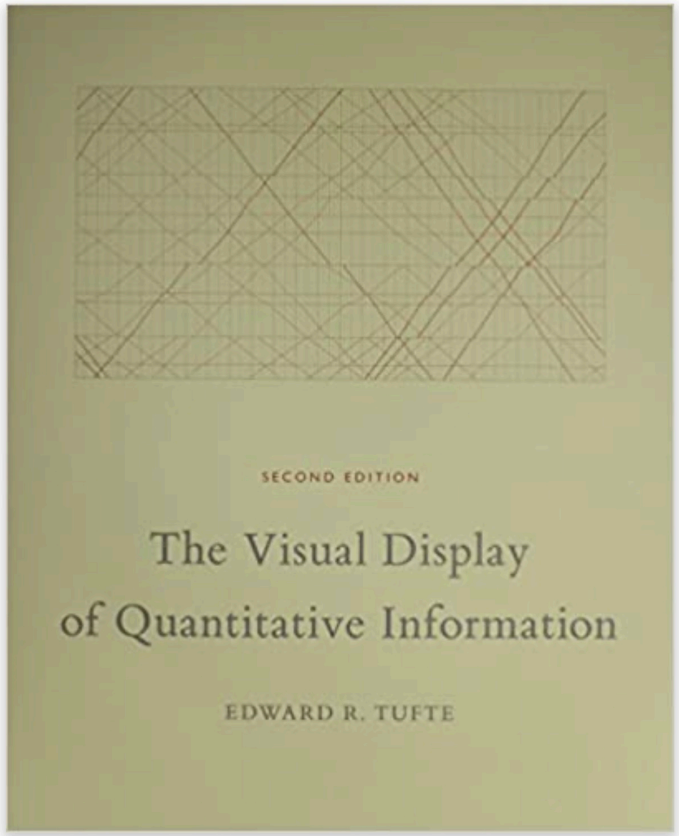


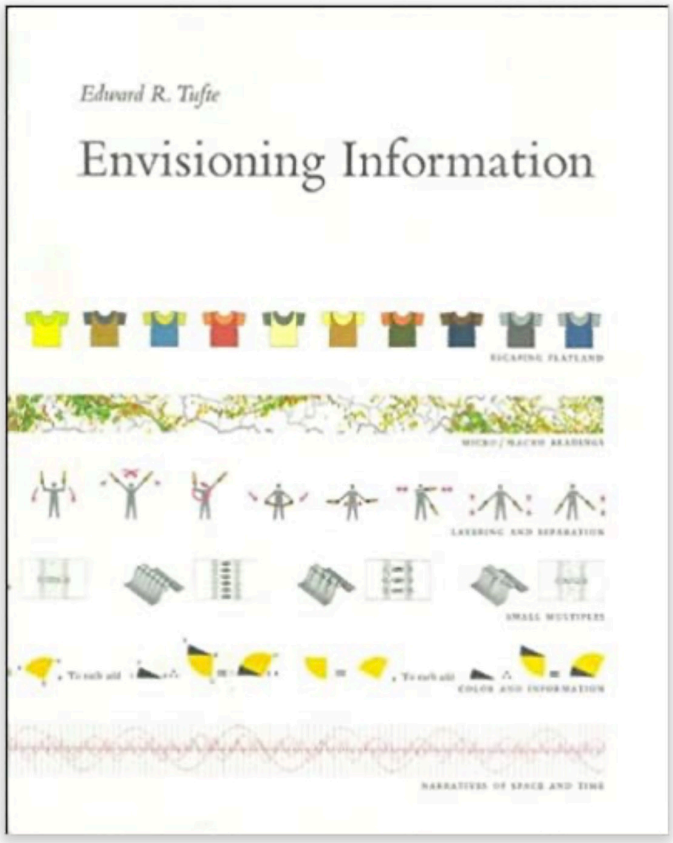
Autog. par Regnier, 8. Pas. Ste Marie Ste Cathe à Paris.

Imp. Lith. Regnier et Dourdet.

Edward Tufte : one of the pioneers (along with Tukey and others)
of the field of data visualization





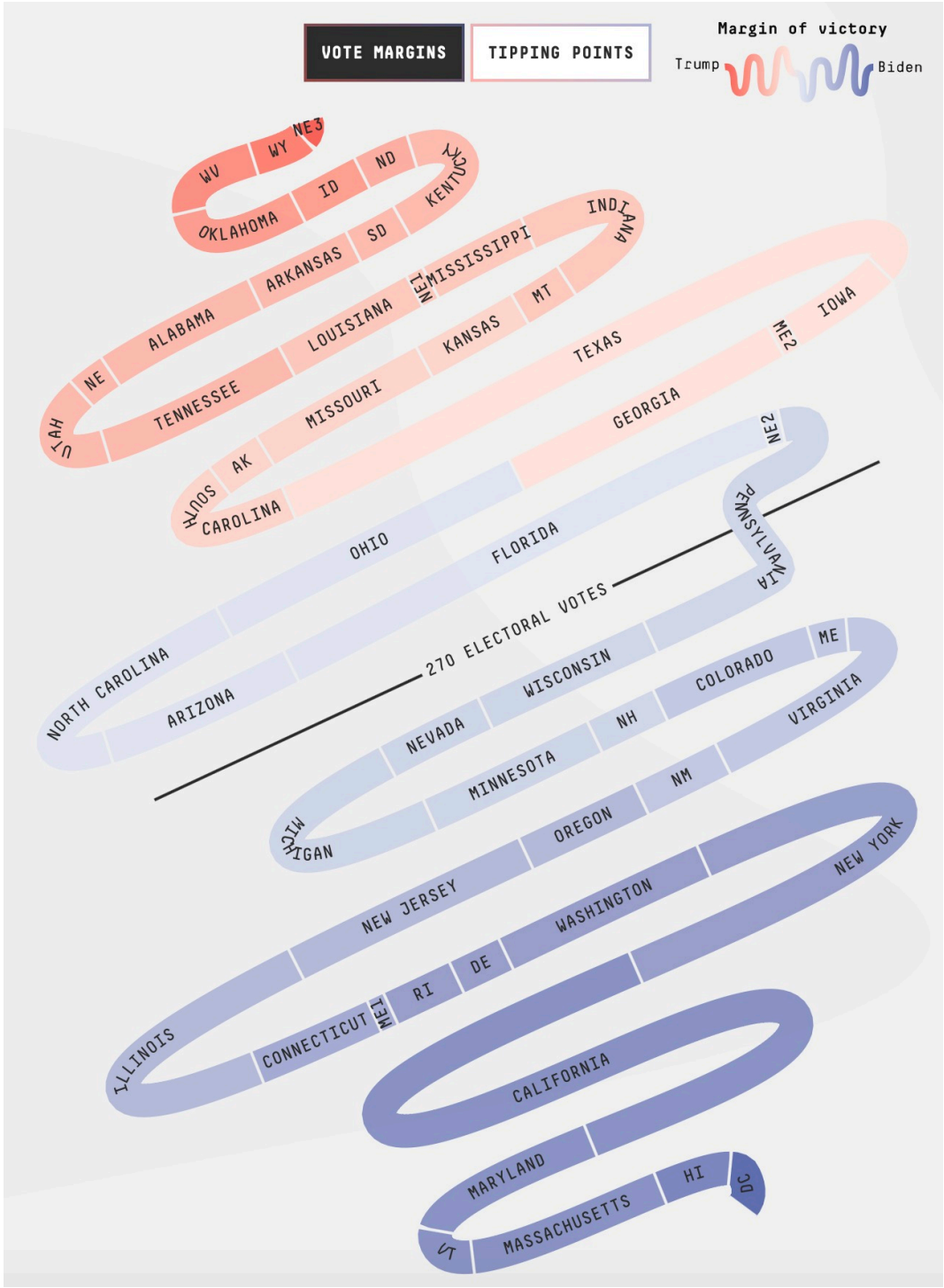


https://www.edwardtufte.com/tufte/
artist, statistician, Yale professor emeritus

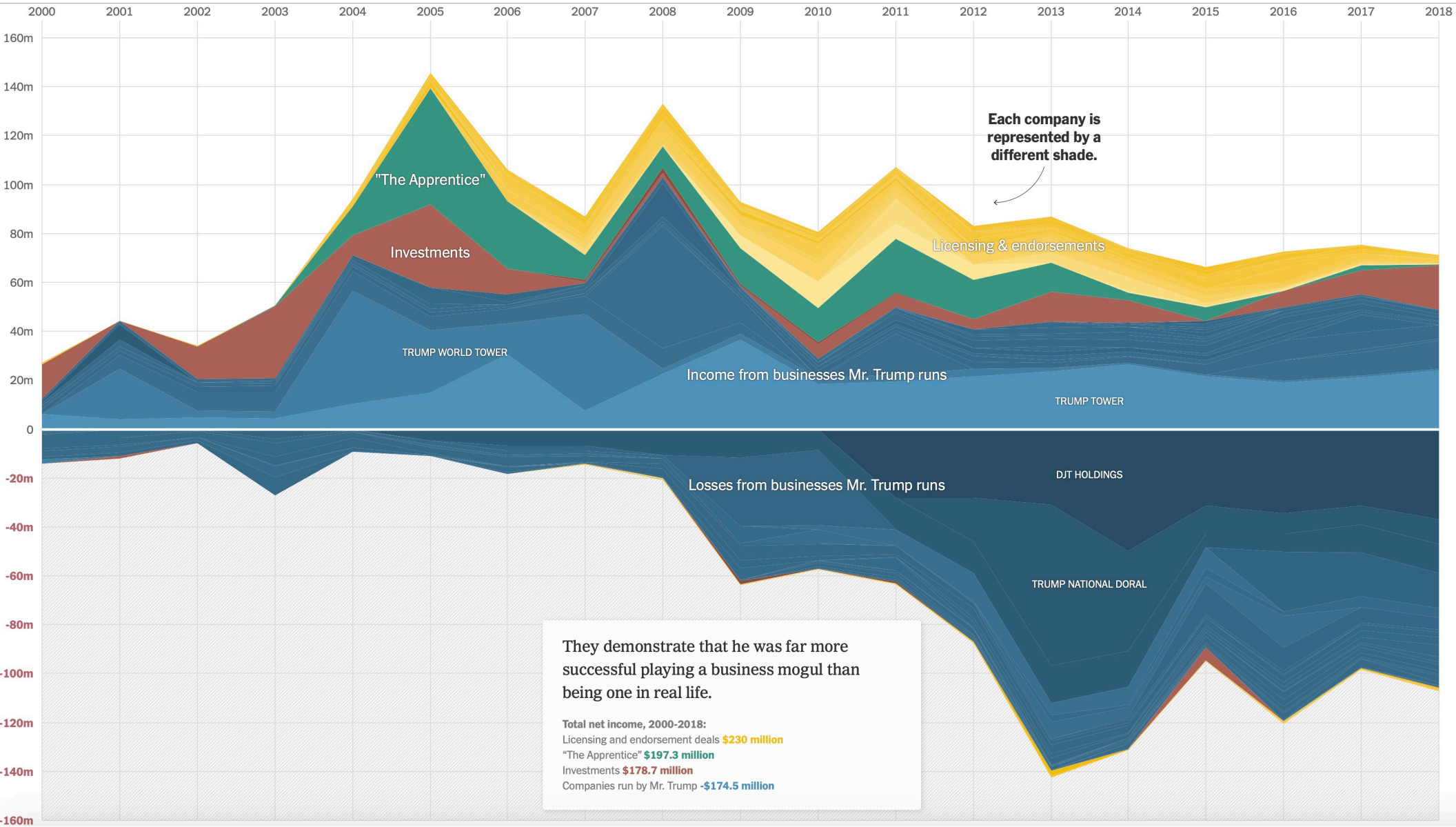
CS 3891 : Special Topics - Data Visualization

This is a course on visualization analysis and design. Topics covered include: perception, color design, interaction, multiple-view visualization design, and task analysis for visualization. The course will cover techniques for geographic data visualization, network visualization, visualizing hierarchical data, high-dimensional data visualization, and uncertainty visualization.

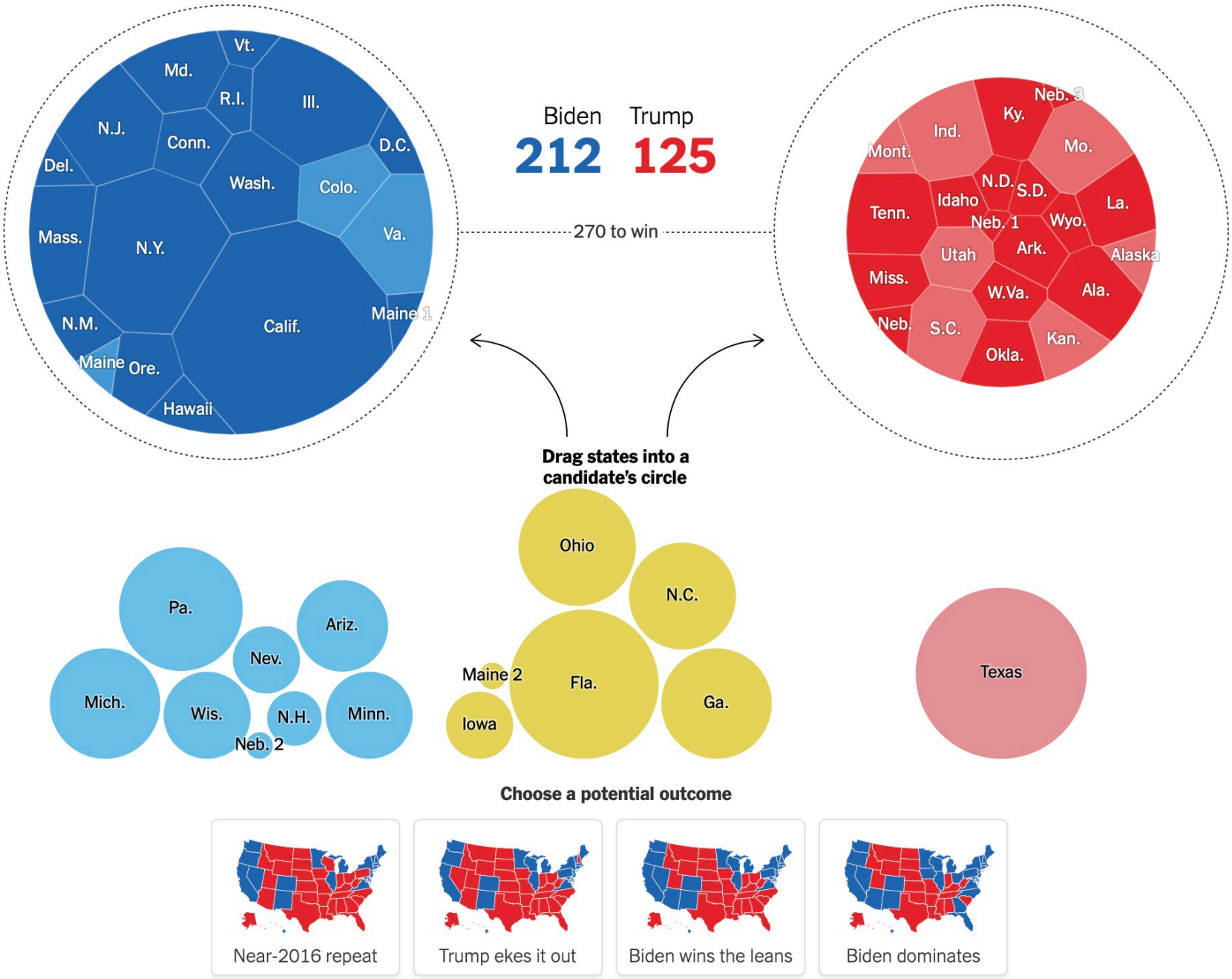
Instructor: Matthew Berger



https://projects.fivethirtyeight.com/2020-election-forecast/



https://www.nytimes.com/interactive/2020/09/27/us/donald-trump-taxes-timeline.html

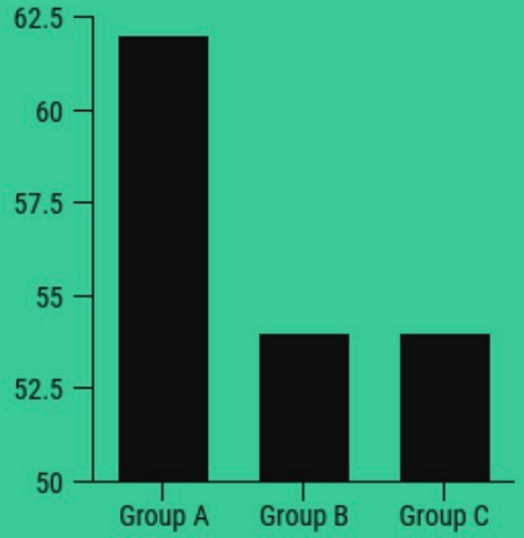


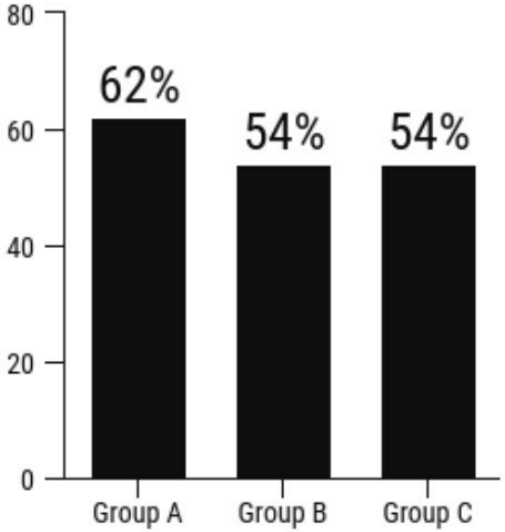
https://www.nytimes.com/interactive/2020/us/elections/election-states-biden-trump.html

some data visualization fundamentals

OMITTING THE BASELINE

In most cases, the baseline for a graph is 0. But writers can skew how data is perceived by making the baseline a different number. This is known as a “truncated graph”.





MISLEADING

- Starting the vertical axis at 50 makes a small difference between groups seem massive
- Group A looks much larger than Groups B and C

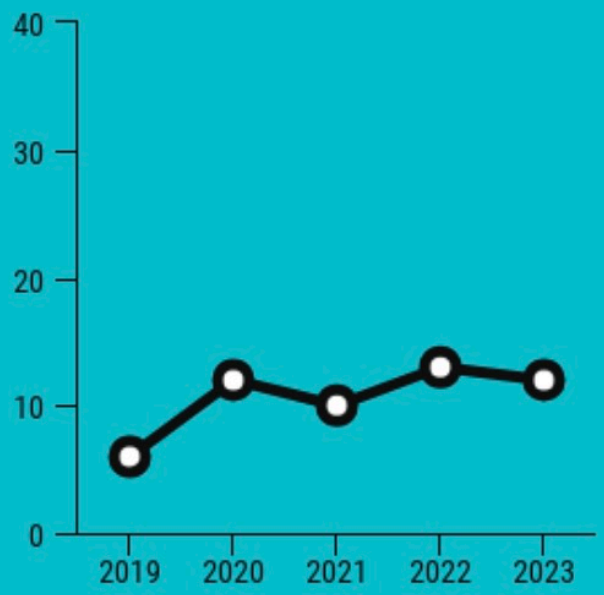
VS

ACCURATE

- Starting the vertical axis at 0 offers a more accurate depiction of the data
- The difference between the groups does not seem as dramatic

MANIPULATING THE Y-AXIS

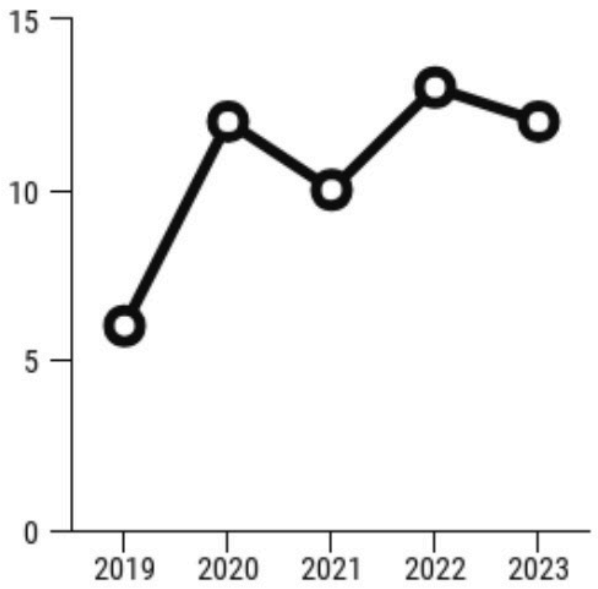
Expanding or compressing the scale on a graph can make changes in data seem more or less significant than they actually are.



MISLEADING

- The scale is disproportionate to the data, making the change over time seem small

VS

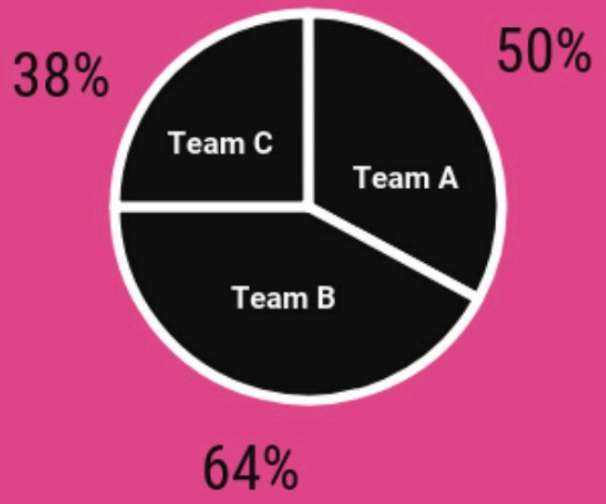


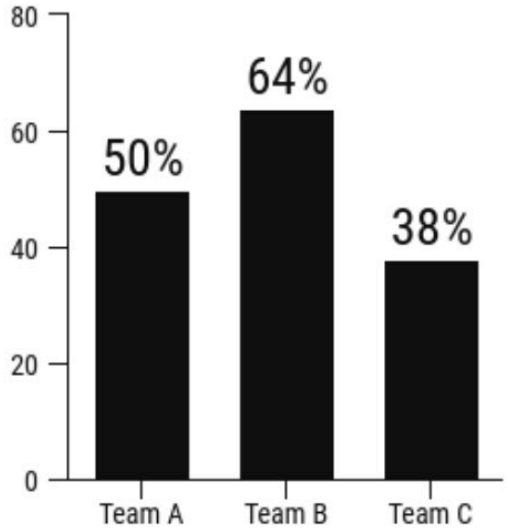
ACCURATE

- The scale is proportionate to the data, showing a greater change over time

USING THE WRONG GRAPH

The type of graph you use should depend on the type of data you want to visualize. Using the wrong type of graph can skew the data. Writers will sometimes use the wrong type of graph on purpose.





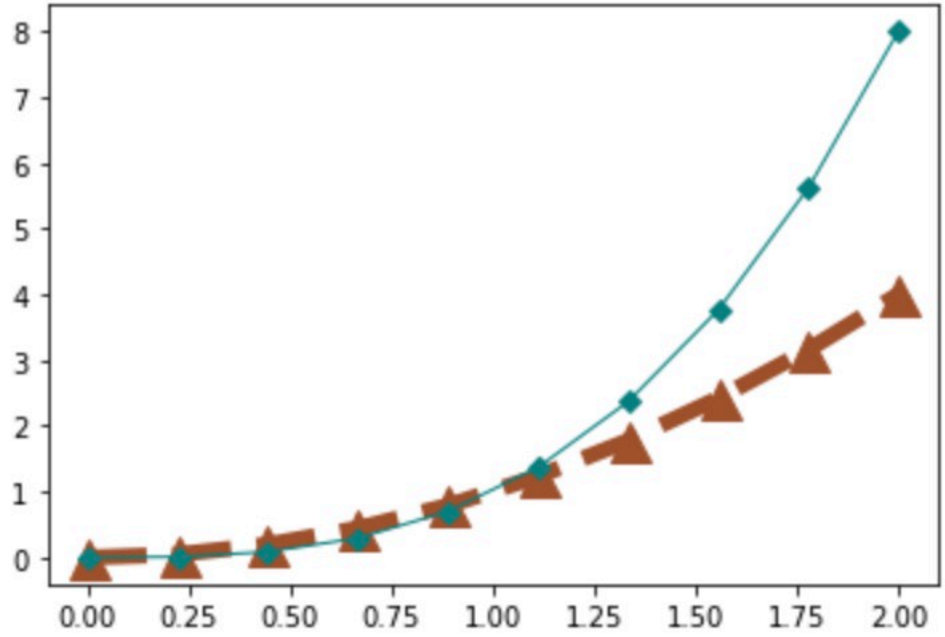
MISLEADING

- Pie charts are used to compare parts of a whole, not the difference between groups
- A different type of graph should be used to compare the three teams

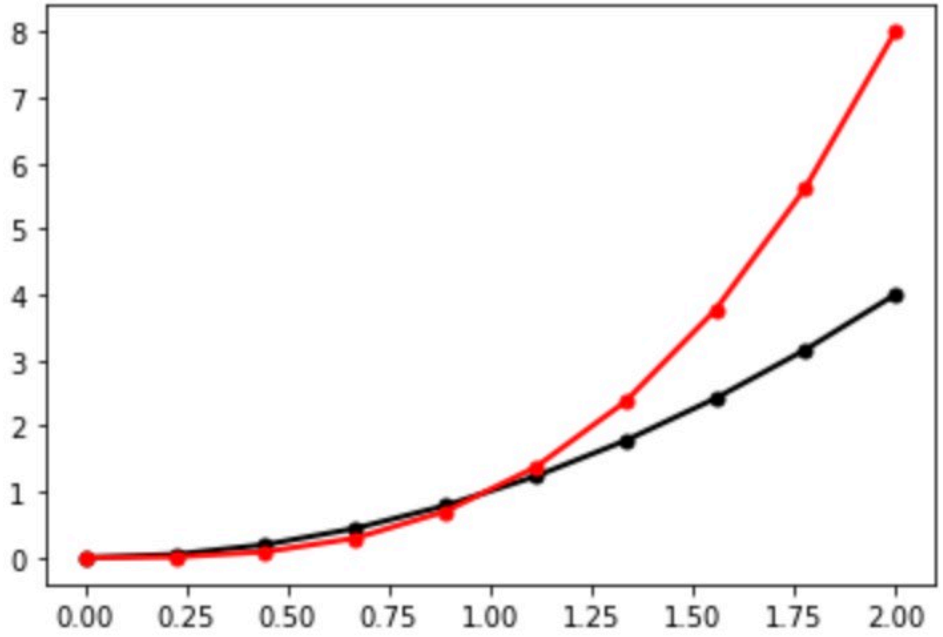
VS

ACCURATE

- Bar graphs are better for showing the differences between groups
- This chart is a better visualization of the data

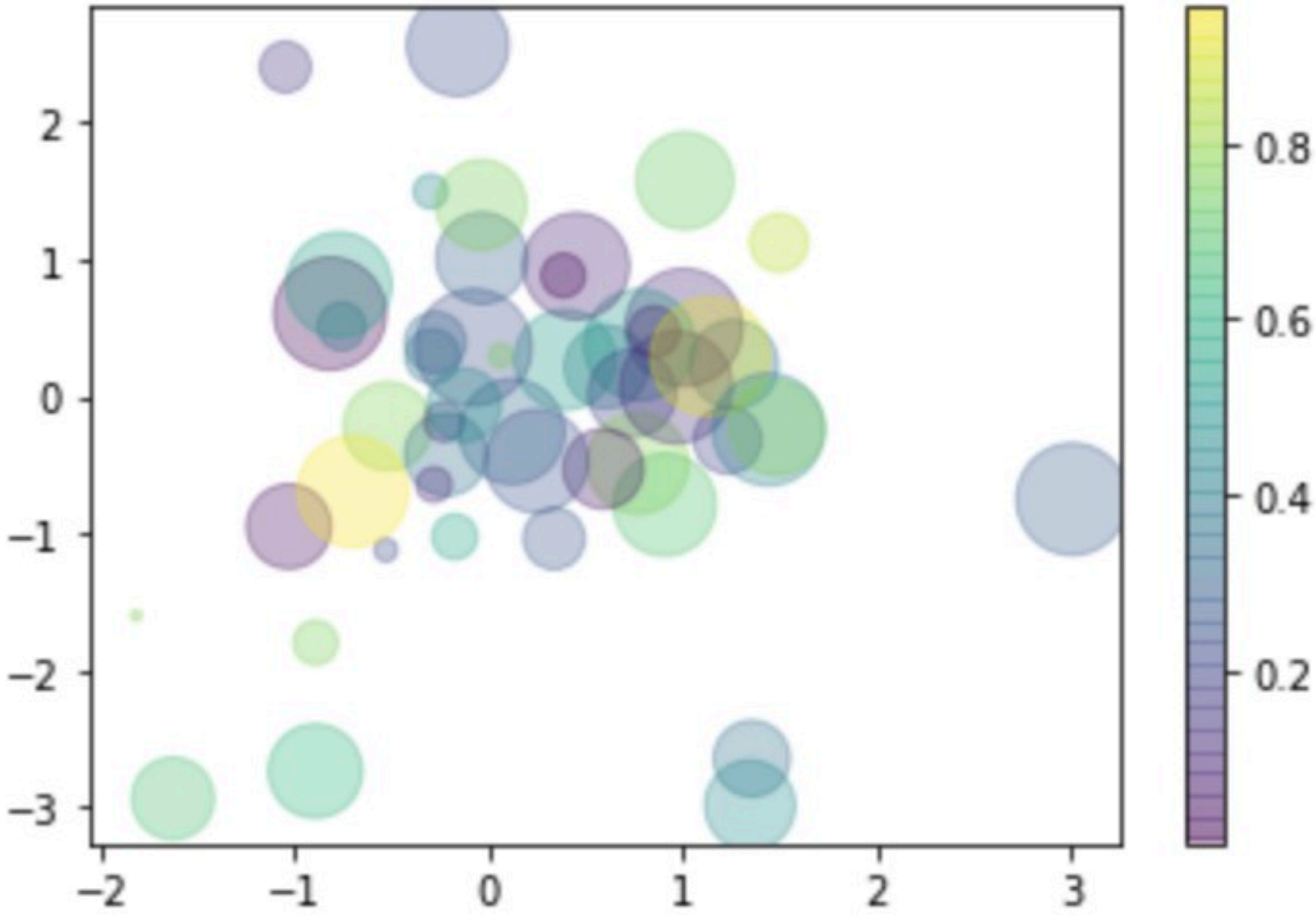


vs.



if there is only <u>one</u> difference between two sets of data shown in a plot
then
there should be only <u>one</u> visual difference between the lines in the plot

four dimensions of difference displayed
in a two-dimensional figure (good example of visualization)



unnecessary use of three dimensions
(bad example of visualization)

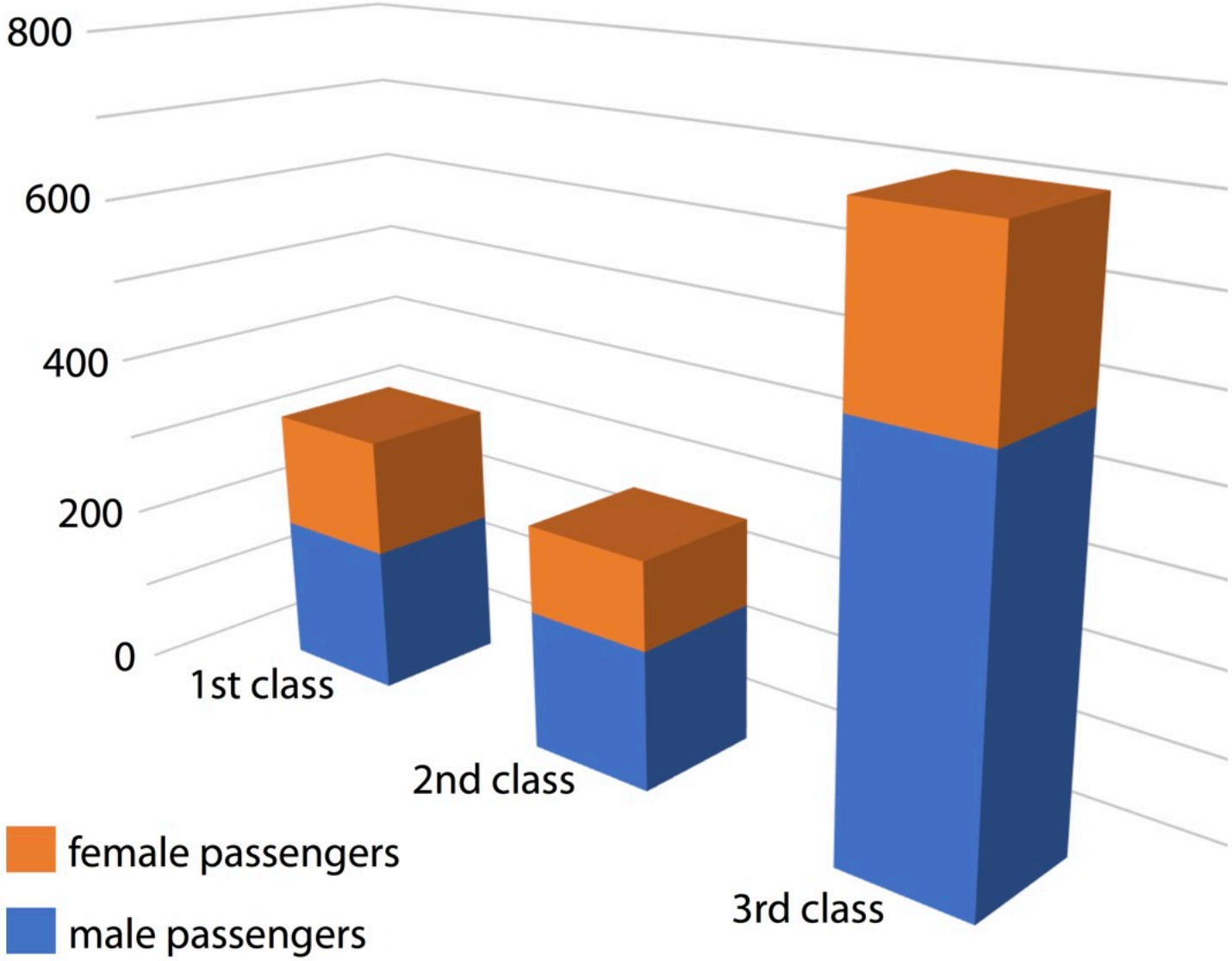
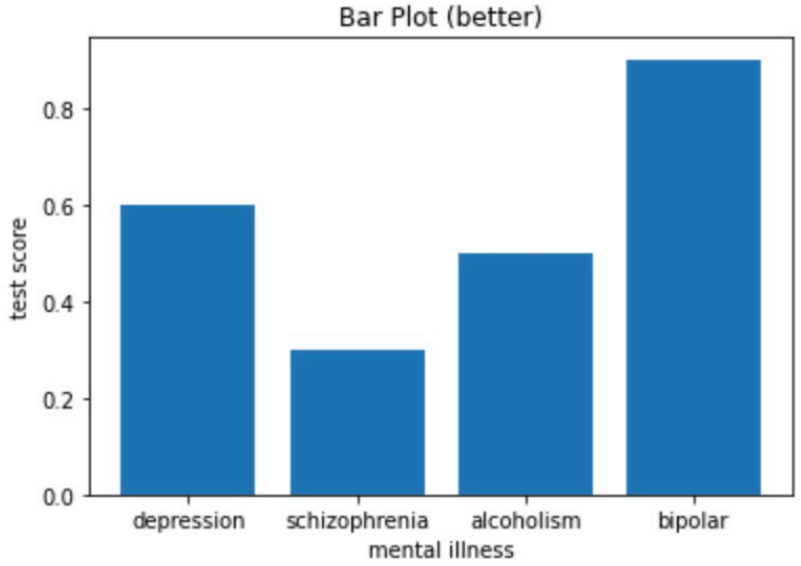


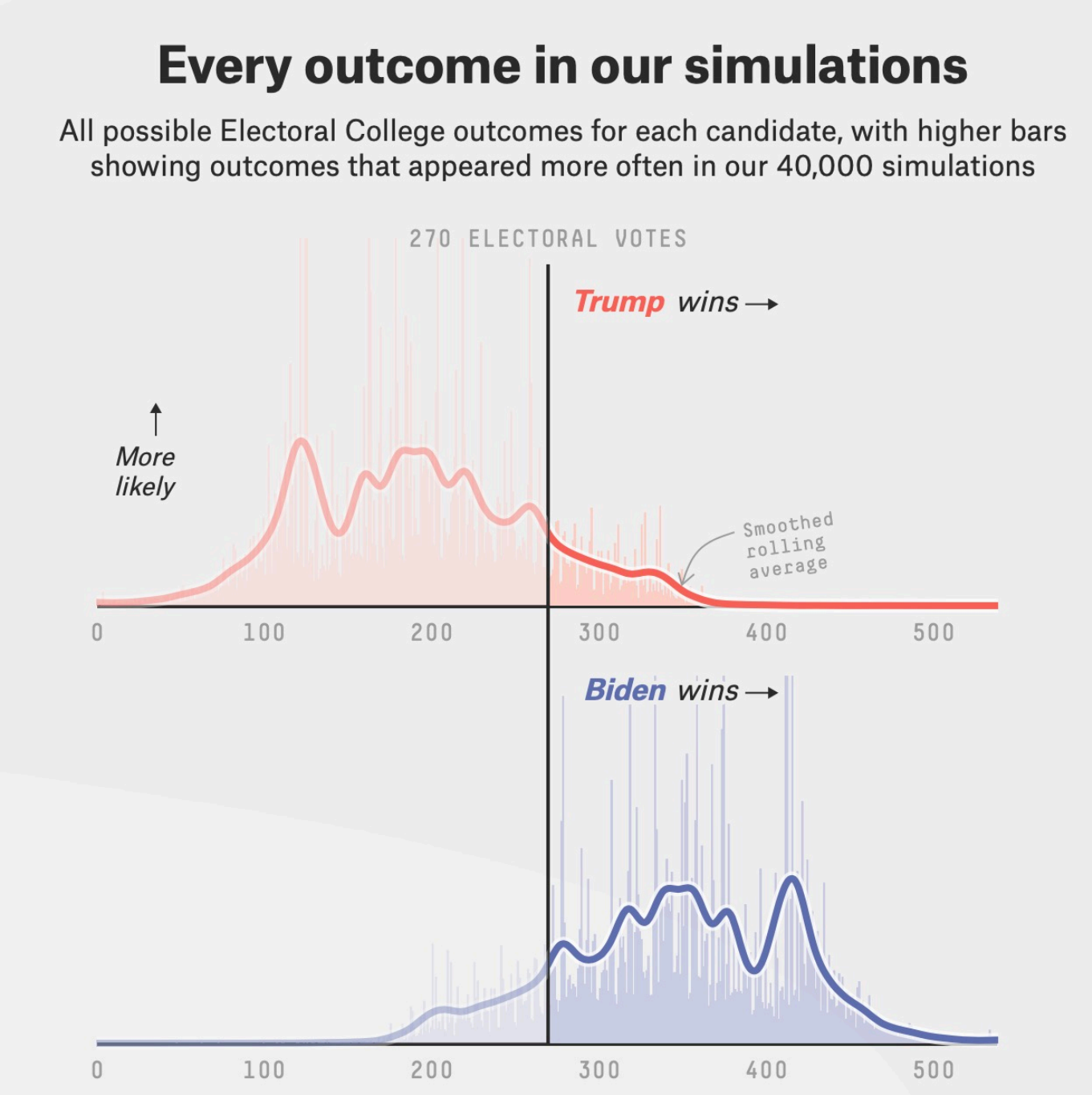
Figure 26.2: Numbers of female and male passengers on the Titanic traveling in 1st, 2nd, and 3rd class

do not use line plots for categorical differences

vs.

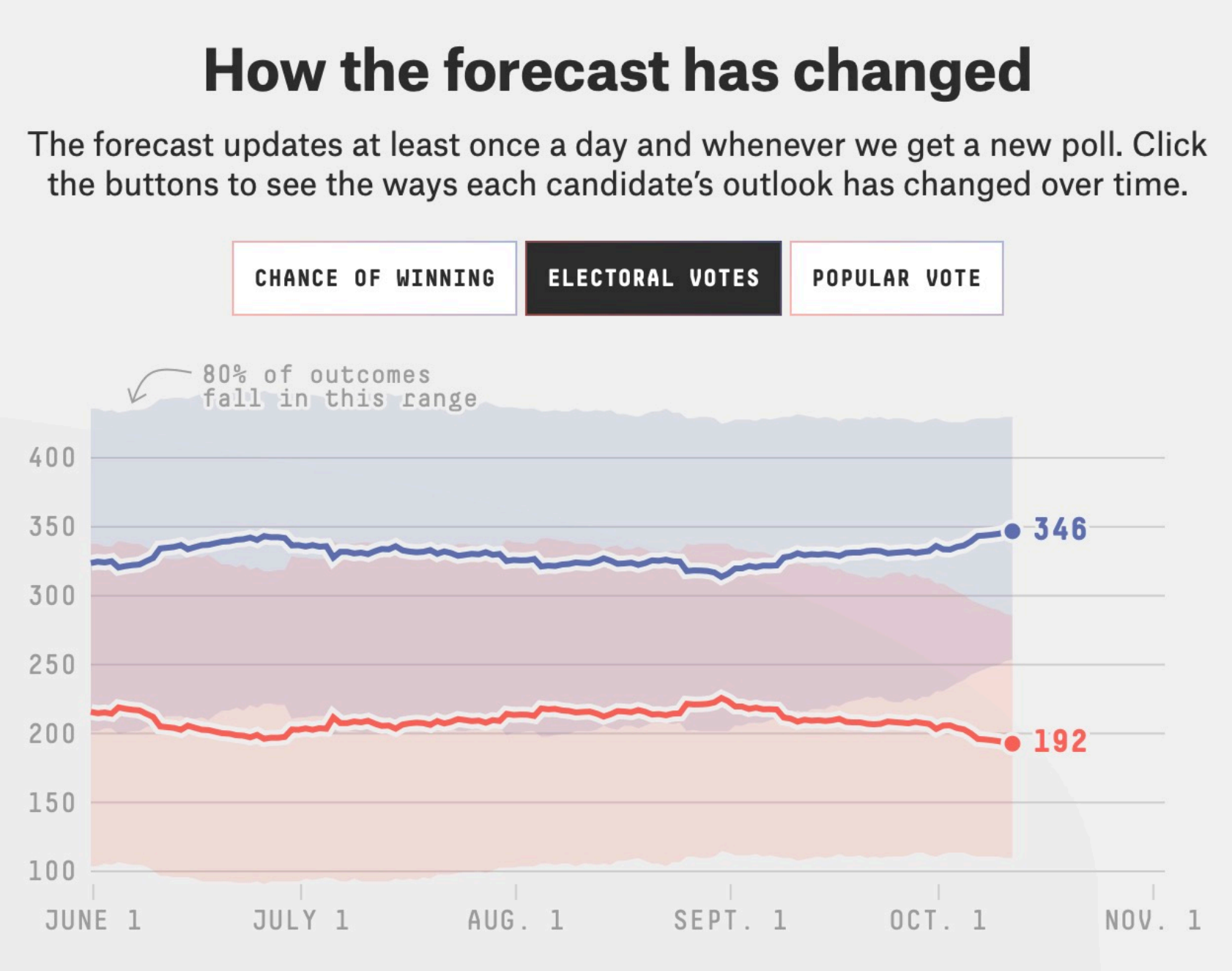


show variability and uncertainty



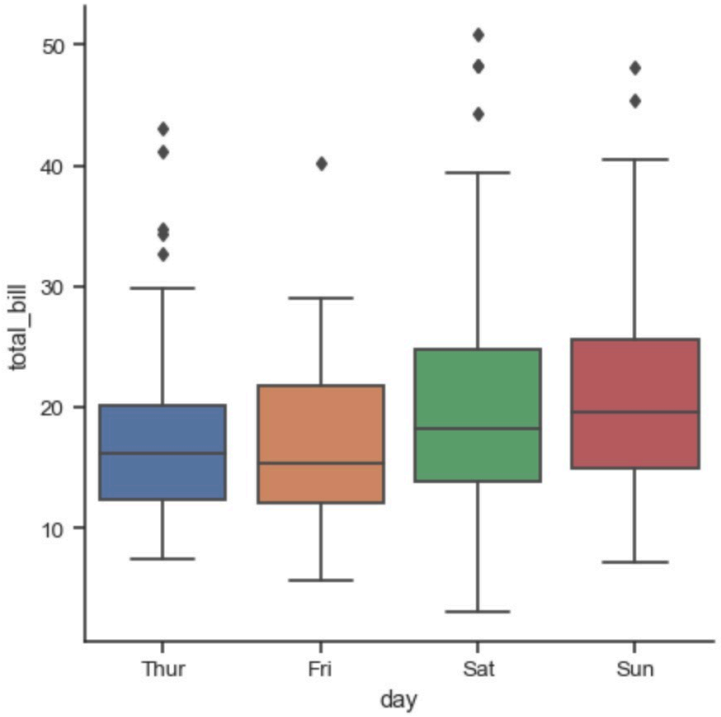
https://projects.fivethirtyeight.com/2020-election-forecast/

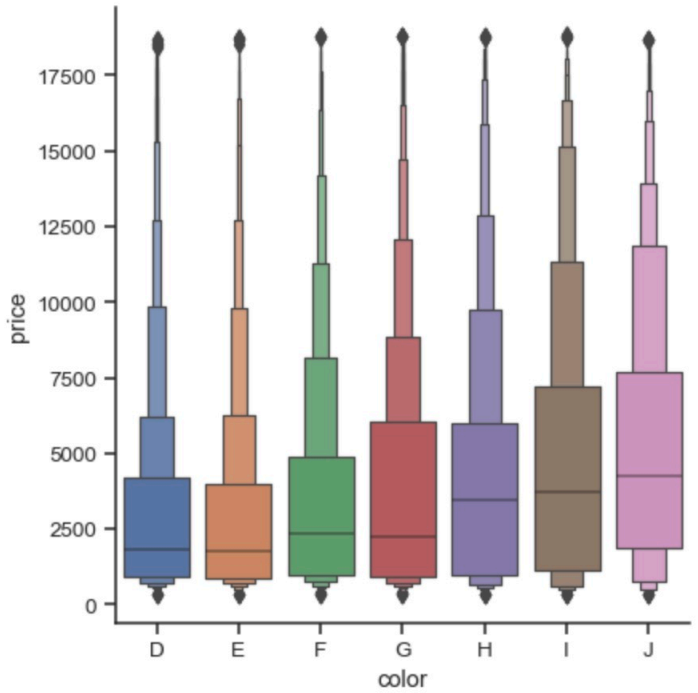
show variability and uncertainty

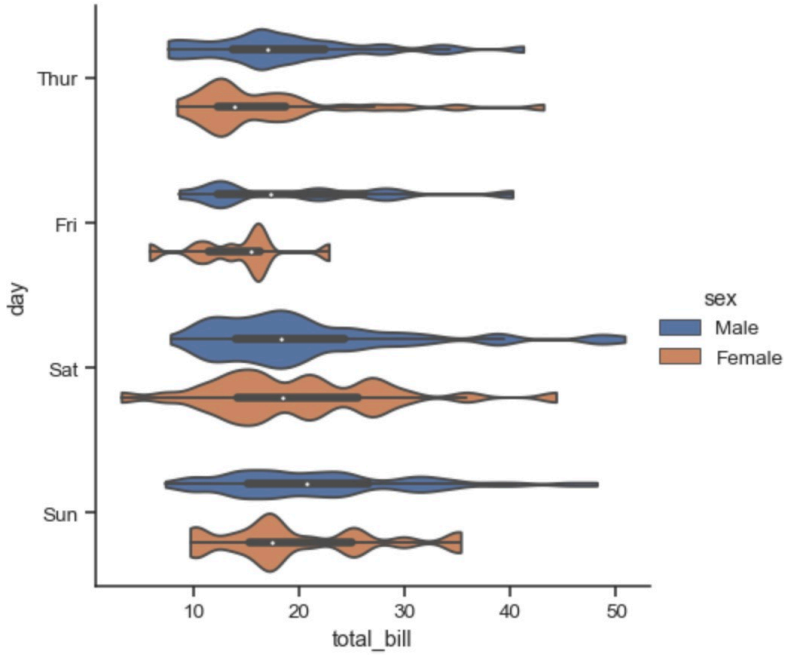


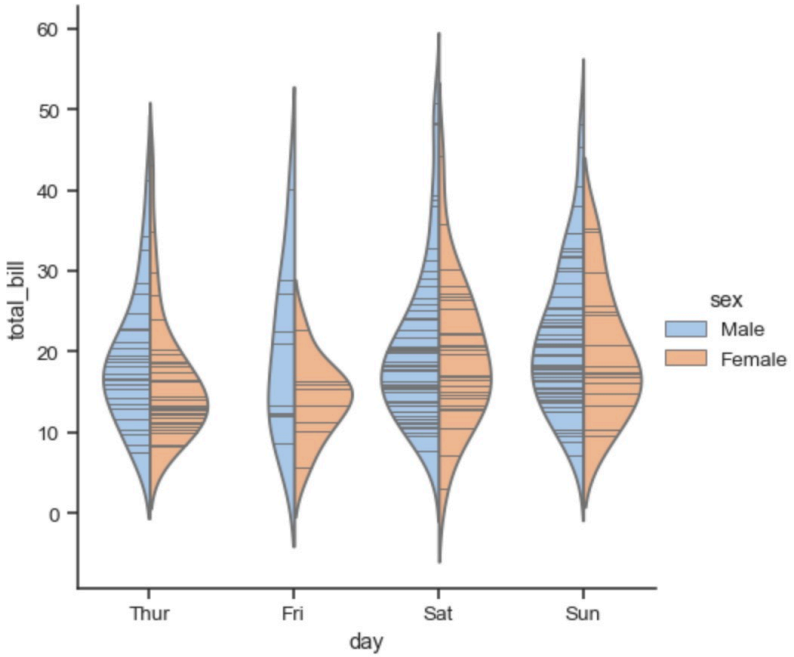
https://projects.fivethirtyeight.com/2020-election-forecast/

show variability and uncertainty







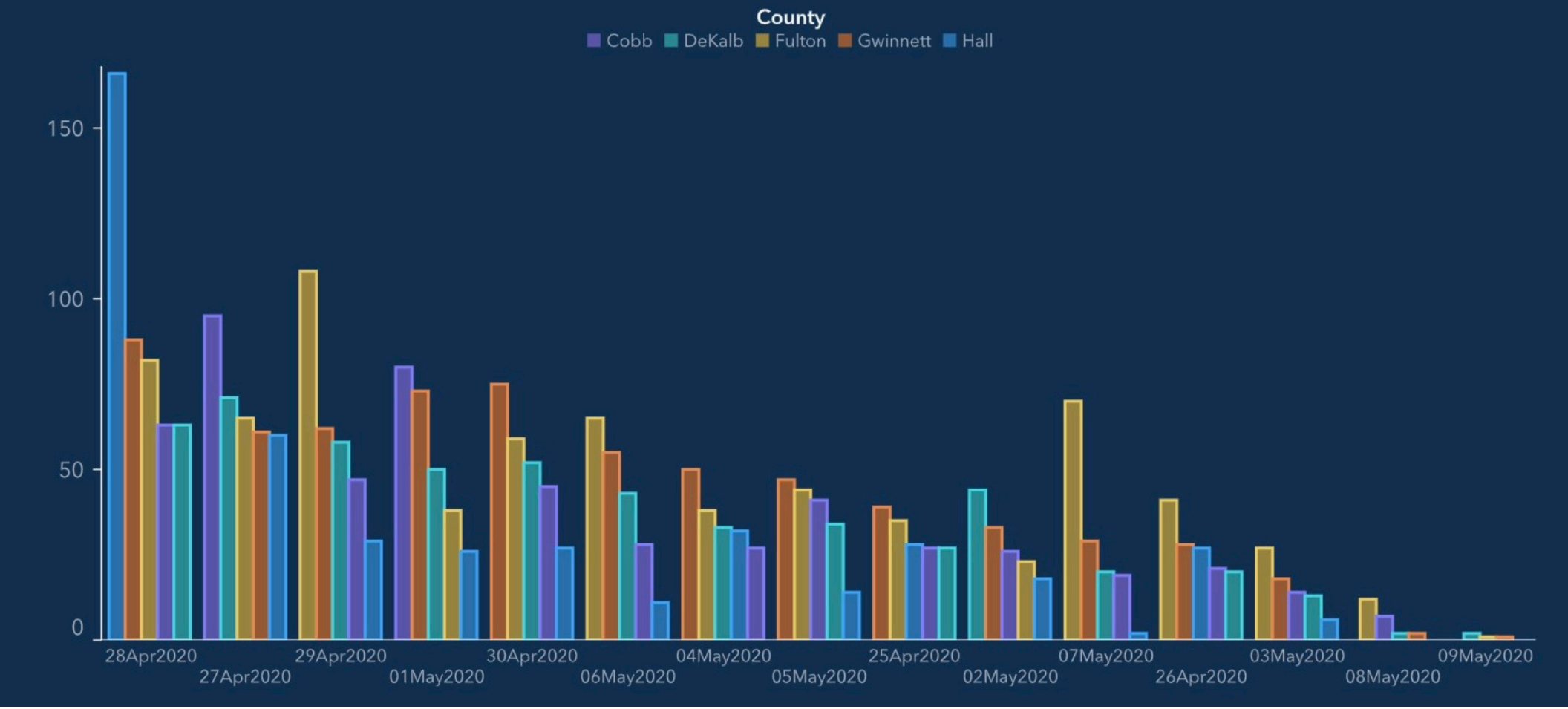


from https://seaborn.pydata.org/tutorial/categorical.html

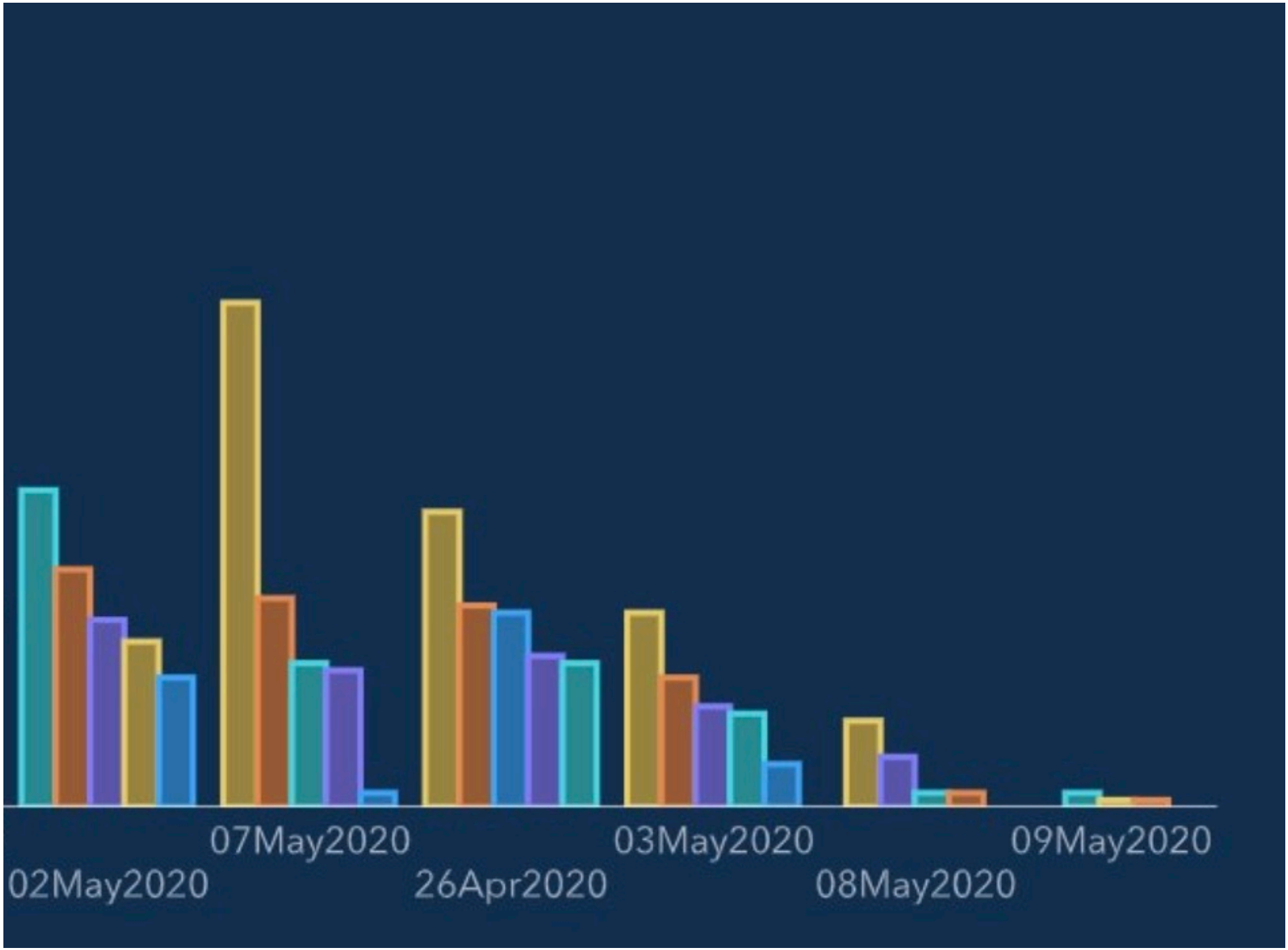
lessons from misleading COVID visuals

Top 5 Counties with the Greatest Number of Confirmed COVID-19 Cases

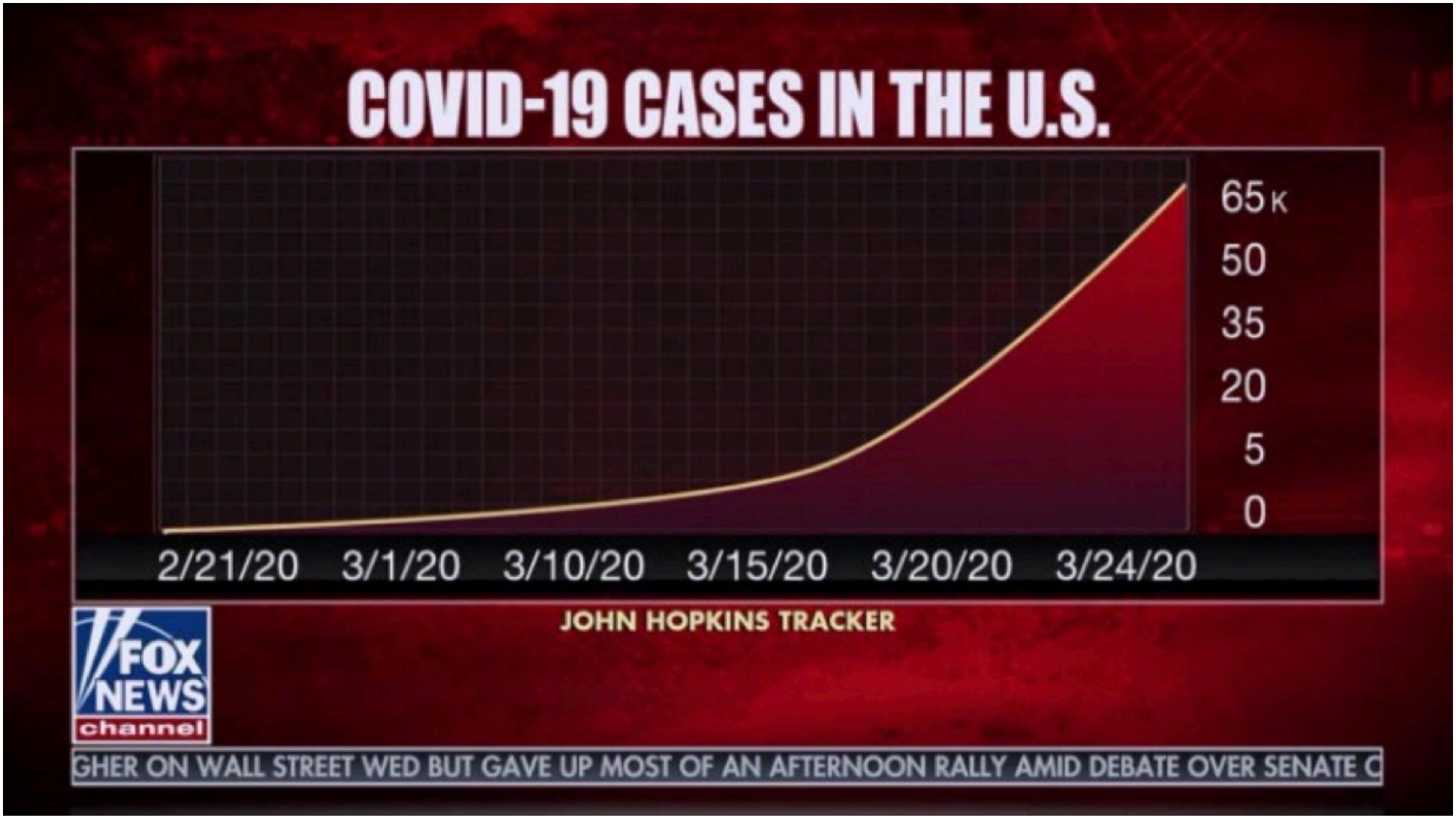
The chart below represents the most impacted counties over the past 15 days and the number of cases over time. The table below also represents the number of deaths and hospitalizations in each of those impacted counties.



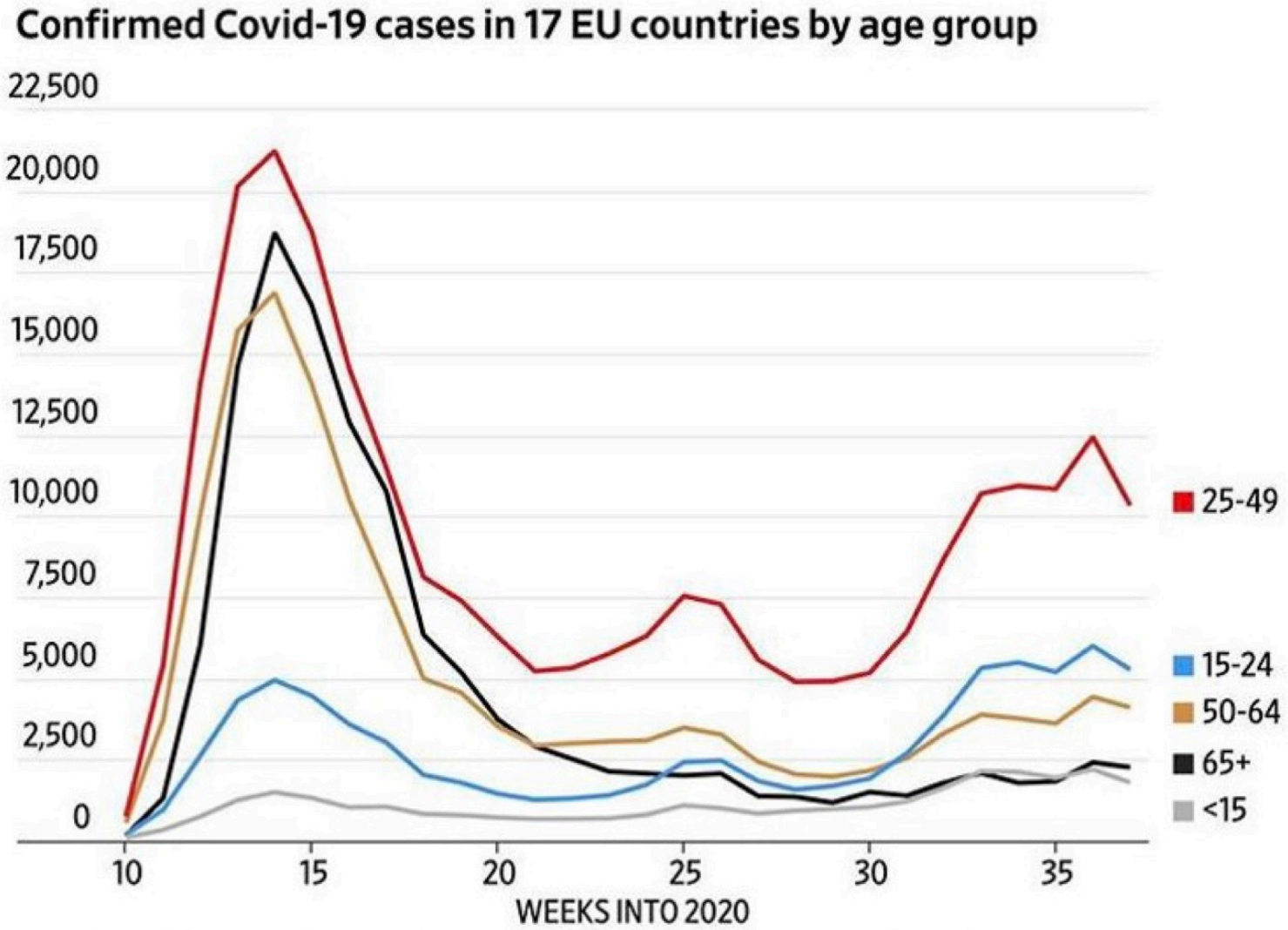
State of Georgia



State of Georgia

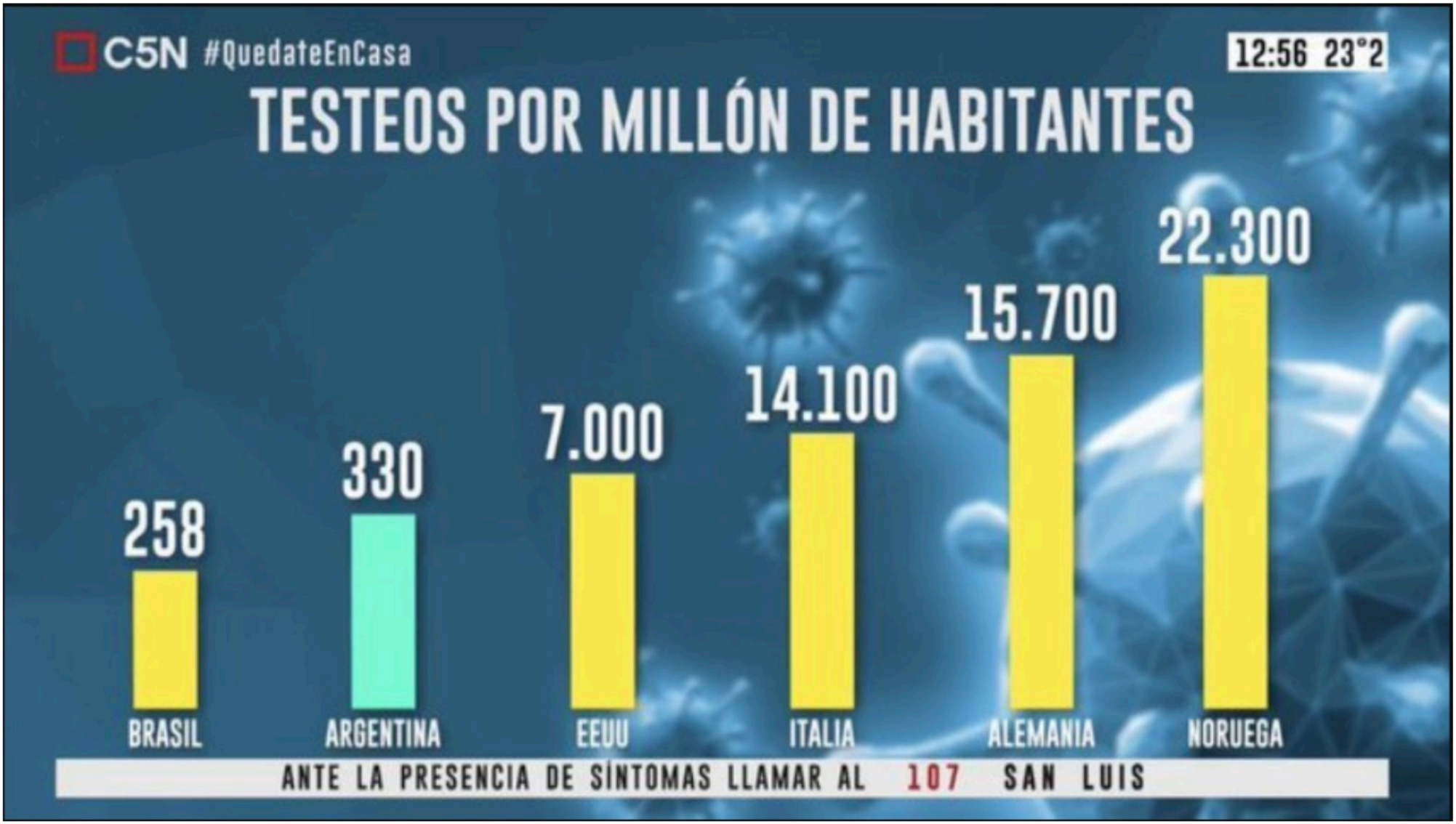

COVID-19 CASES IN THE U.S.
65K
50
35
20
5
0
2/21/20
3/1/20
3/10/20
3/15/20
3/20/20
3/24/20
JOHN HOPKINS TRACKER
FOX NEWS channel
GHER ON WALL STREET WED BUT GAVE UP MOST OF AN AFTERNOON RALLY AMID DEBATE OVER SENATE C

Younger People Are Driving the Current Surge in Covid-19 Cases Across Europe

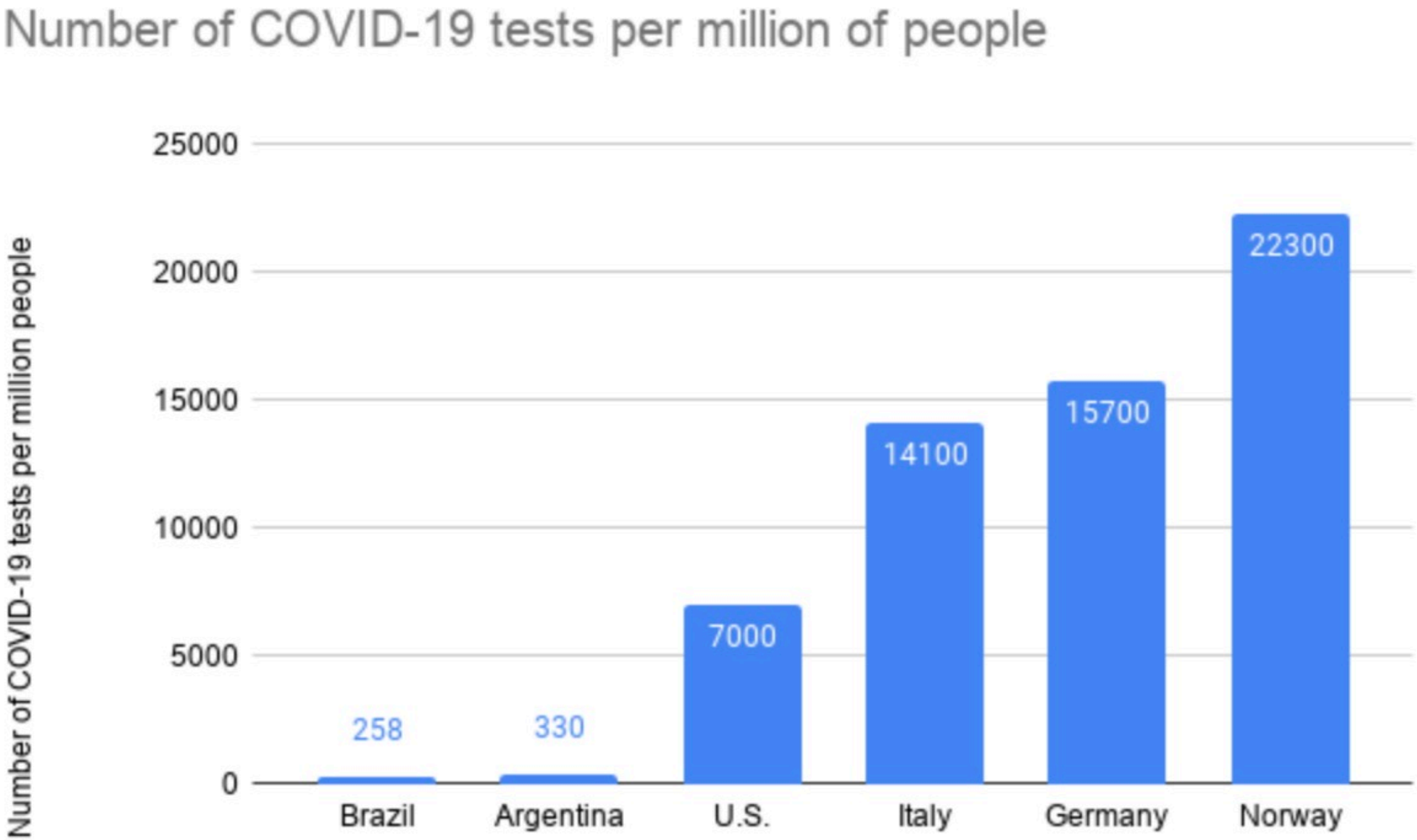


Note: Countries include Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Germany, Iceland, Ireland, Latvia, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal and Sweden

Source: European Surveillance System, part of the European Center for Disease Prevention and Control



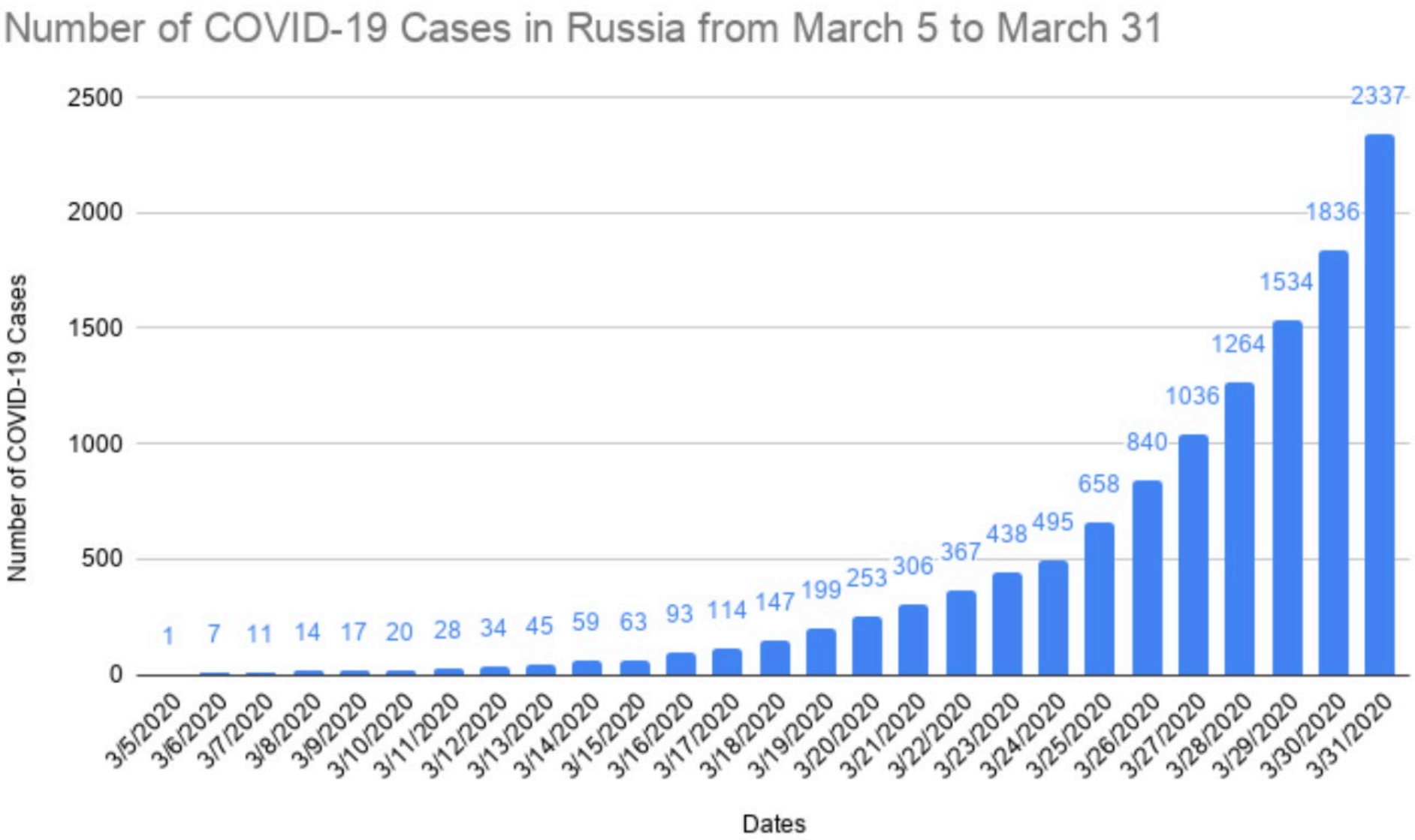
Argentinian TV



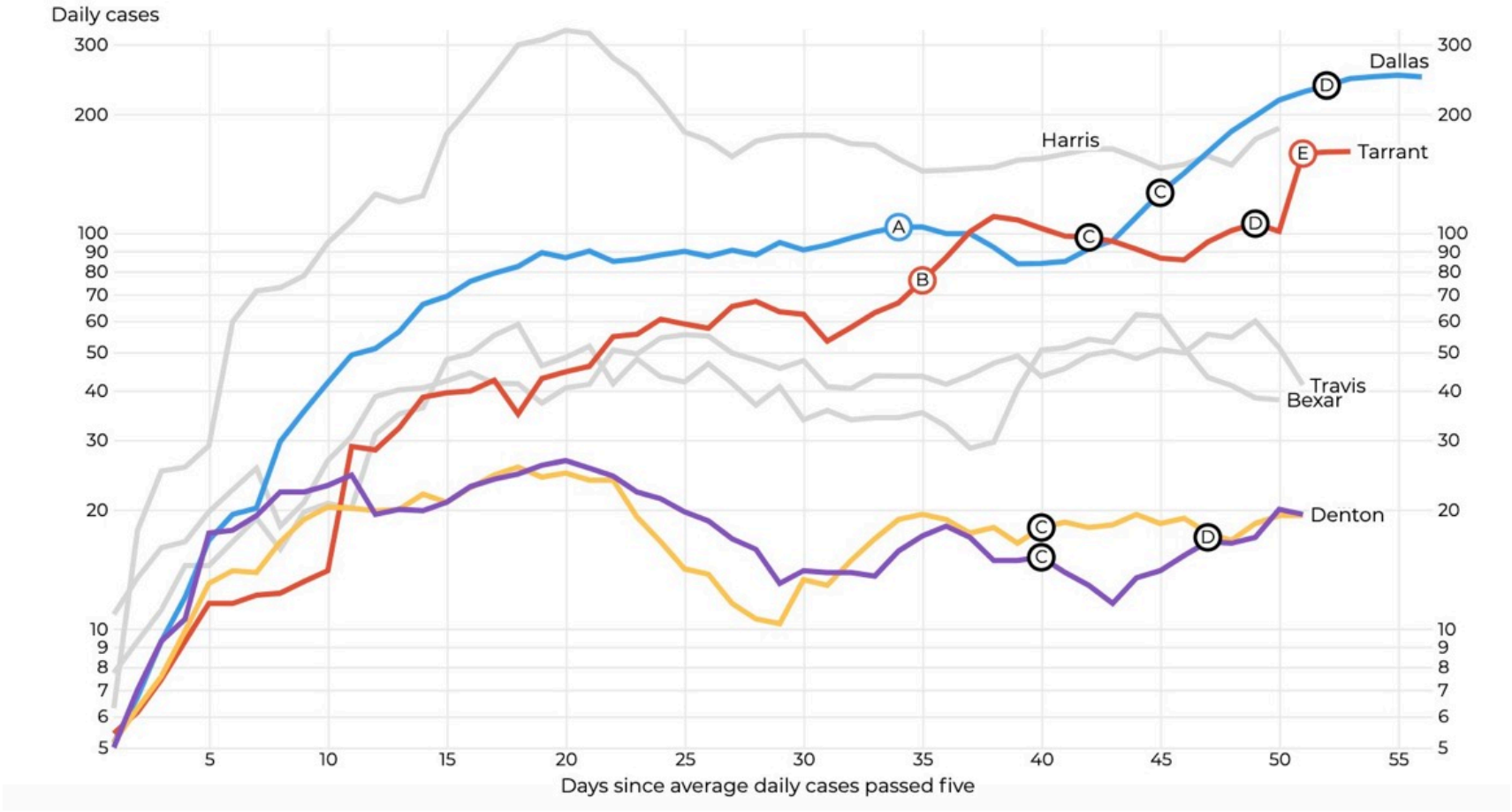
Argentinian TV
(replotted)

The number of COVID-19 cases in Russia from March 5 to March 31. Source: Reddit. Original Source: Russia Today

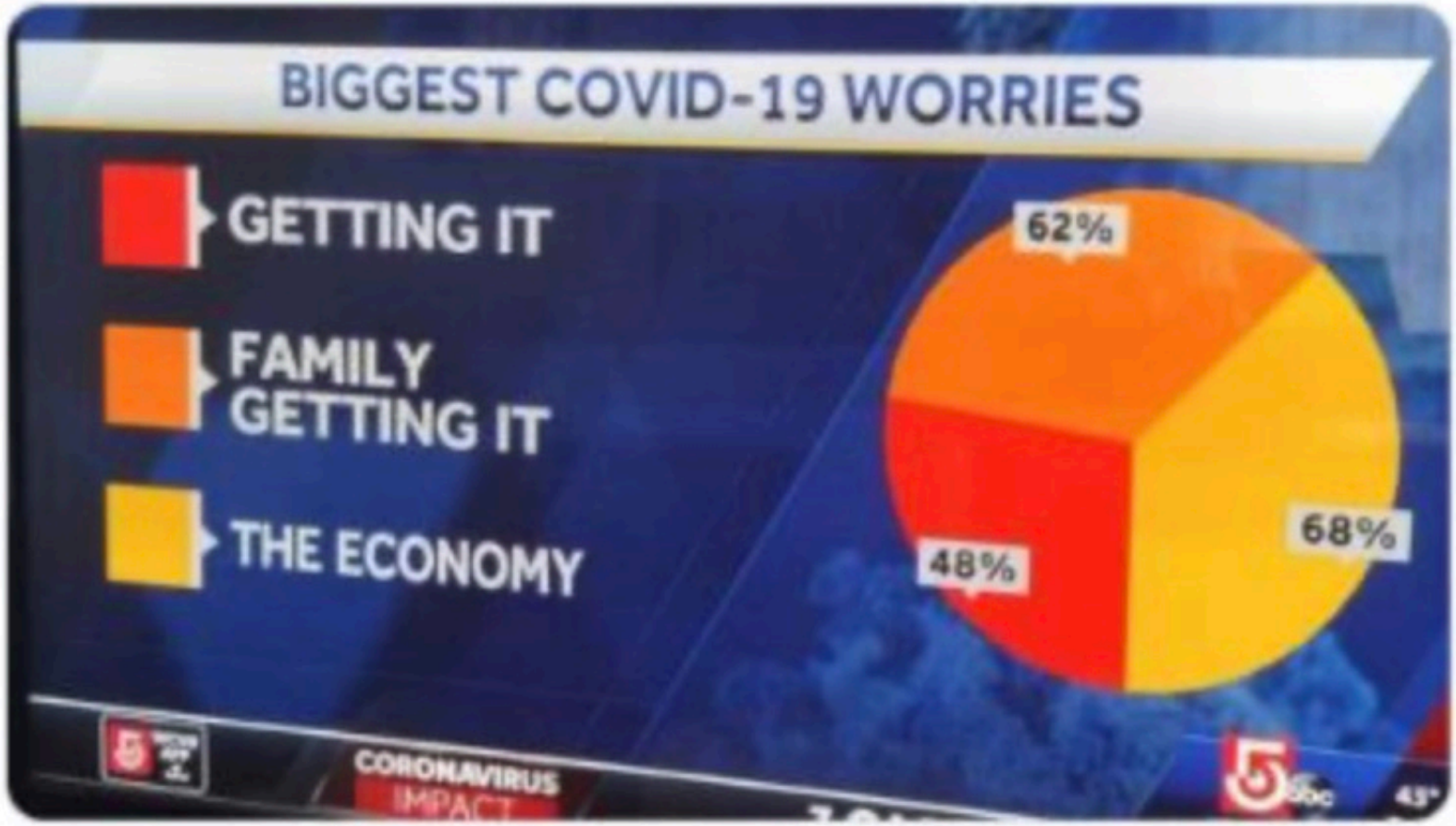
Russia Today

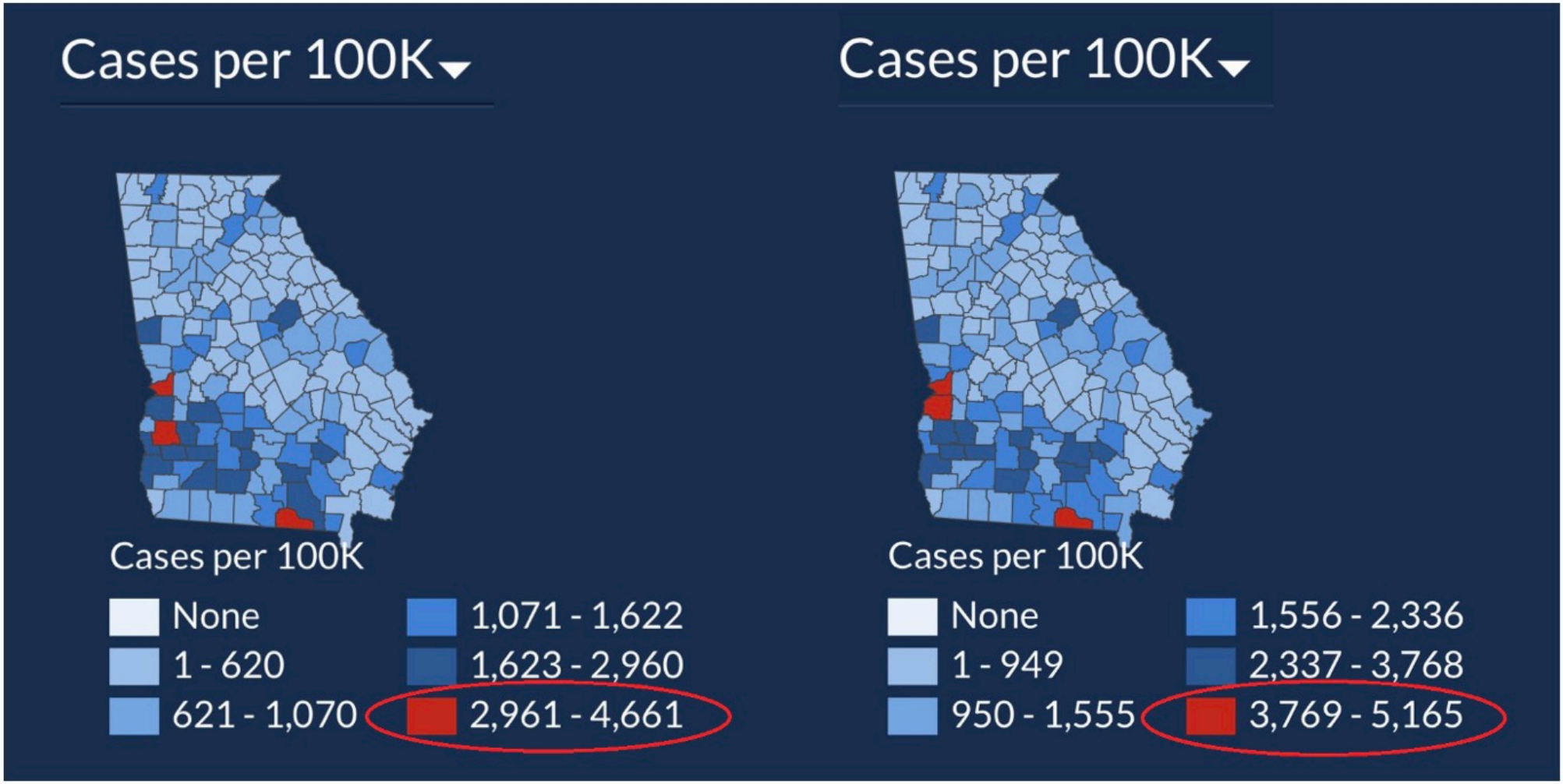


Russia Today
(replotted)



Dallas Morning News

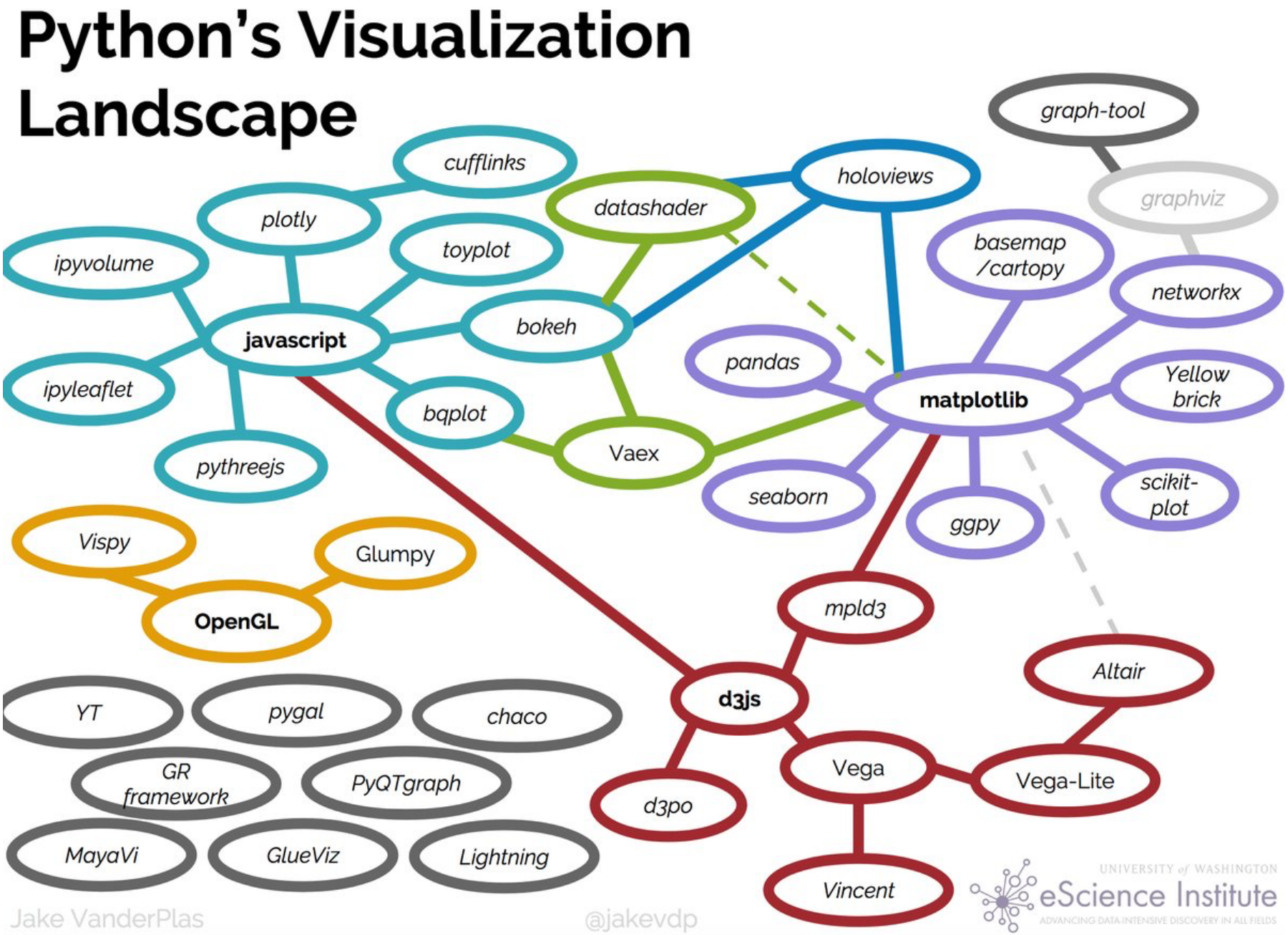

BIGGEST COVID-19 WORRIES
GETTING IT
FAMILY GETTING IT
THE ECONOMY
62%
68%
48%
CORONAVIRUS
IMPACT



State of Georgia

matplotlib

we'll only cover a tiny fraction of all the graphing options available in Python



we've already seen a bunch of matplotlib

```
import matplotlib.pyplot as plt
```

line plot

```
plt.plot(x, p)
plt.xlabel("x")
plt.ylabel("p(x)")
plt.title(f"Normal Distribution (m={m}, s={s})");
```

bar chart

```
plt.bar(hb[:len(hb)-1], h, width=(xmax-xmin)/nbins,
                           align='edge')
plt.xlabel('x')
plt.ylabel('count')
plt.title(f'randn()\nN = {N:,}')
plt.xlim((xmin, xmax));
```

matplotlib

developed in 2002 as an implementation in Python of graphing/plotting techniques used in Matlab

`pylab` (old implementation) - do not use

`pyplot` (current implementation)

`seaborn` and other tools built on `matplotlib.pyplot`

we will only discuss a tiny fraction of `pyplot` options

matplotlib expects numpy arrays for the data to be plotted

`plt.show()`

not necessary in a Jupyter Notebook

required in a Python `.py` script

while in a Jupyter Notebook, the whole purpose is to display graphs/plots and outputs in the notebook, in a script you do not necessarily need to actually display graphs/plots, but can create them and save to files

you might run a `.py` script in batch mode (ACCRE or Amazon AWS)

saving graphs to a file with `plt.savefig()`

save figure in various types with a specified resolution (dpi)

```
plt.savefig('lnplot.pdf', dpi=300, format='pdf')
plt.savefig('lnplot.jpg', dpi=300, format='jpg')
plt.savefig('lnplot200.tif', dpi=200, format='tif')
plt.savefig('lnplot600.tif', dpi=600, format='tif')
```

fname **dpi** **jpg**

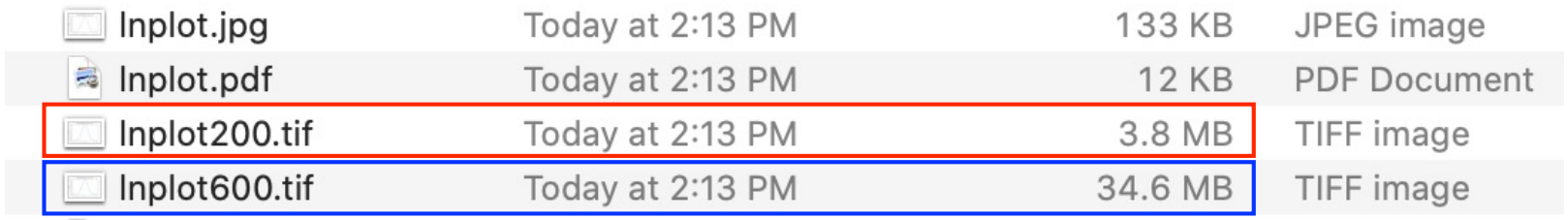
```
{'ps': 'Postscript',
 'eps': 'Encapsulated Postscript',
 'pdf': 'Portable Document Format',
 'pgf': 'PGF code for LaTeX',
 'png': 'Portable Network Graphics',
 'raw': 'Raw RGBA bitmap',
 'rgba': 'Raw RGBA bitmap',
 'svg': 'Scalable Vector Graphics',
 'svgz': 'Scalable Vector Graphics',
 'jpg': 'Joint Photographic Experts Group',
 'jpeg': 'Joint Photographic Experts Group',
 'tif': 'Tagged Image File Format',
 'tiff': 'Tagged Image File Format'}
```

```
fig = plt.figure()
fig.canvas.get_supported_filetypes()
```

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.savefig.html

saving graphs to a file with `plt.savefig()`

```
plt.savefig('lnplot200.tif', dpi=200, format='tif')
plt.savefig('lnplot600.tif', dpi=600, format='tif')
```

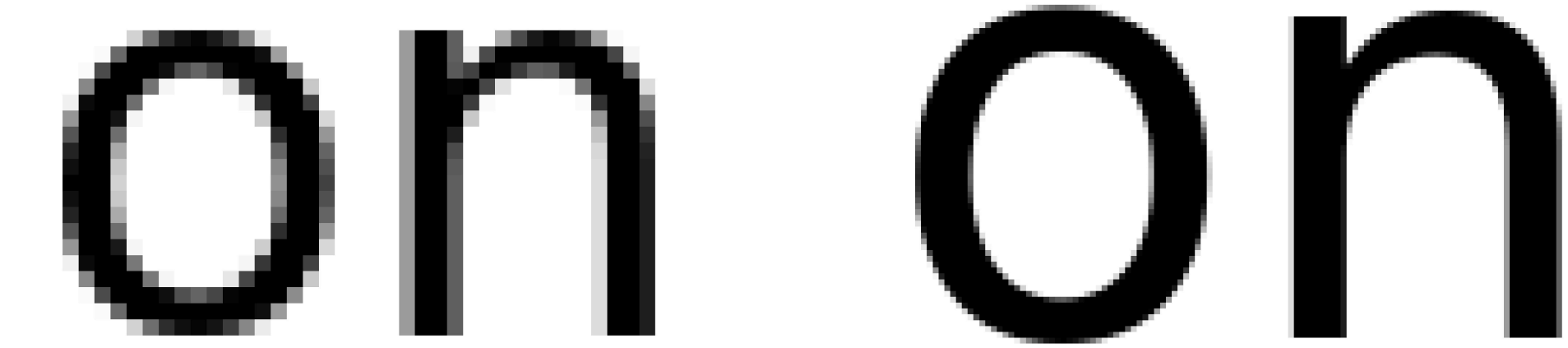


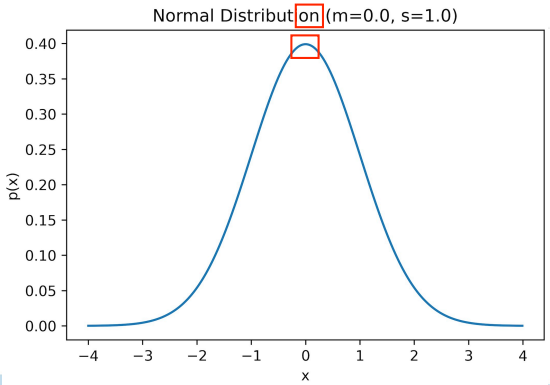
TIF or TIFF (Tagged Image File Format) are often required for figures in manuscripts accepted for publication

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.savefig.html

200 dpi TIF

600 dpi TIF





journals typically require high-resolution TIF images

What will this do? As a single Jupyter Notebook cell or as a piece of code in a `.py` script?

```
# pdf
plt.plot(x, p)
plt.xlabel("x")
plt.ylabel("p(x)")
plt.title(f"Normal Distribution (m={m}, s={s})");

# histogram
plt.bar(hb[:len(hb)-1], h, width=(xmax-xmin)/nbins, align='edge')
plt.xlabel('x')
plt.ylabel('count')
plt.title(f'randn()\nN = {N:,}')
plt.xlim((xmin, xmax));
```

What will this do? As a single Jupyter Notebook cell or as a piece of code in a `.py` script?

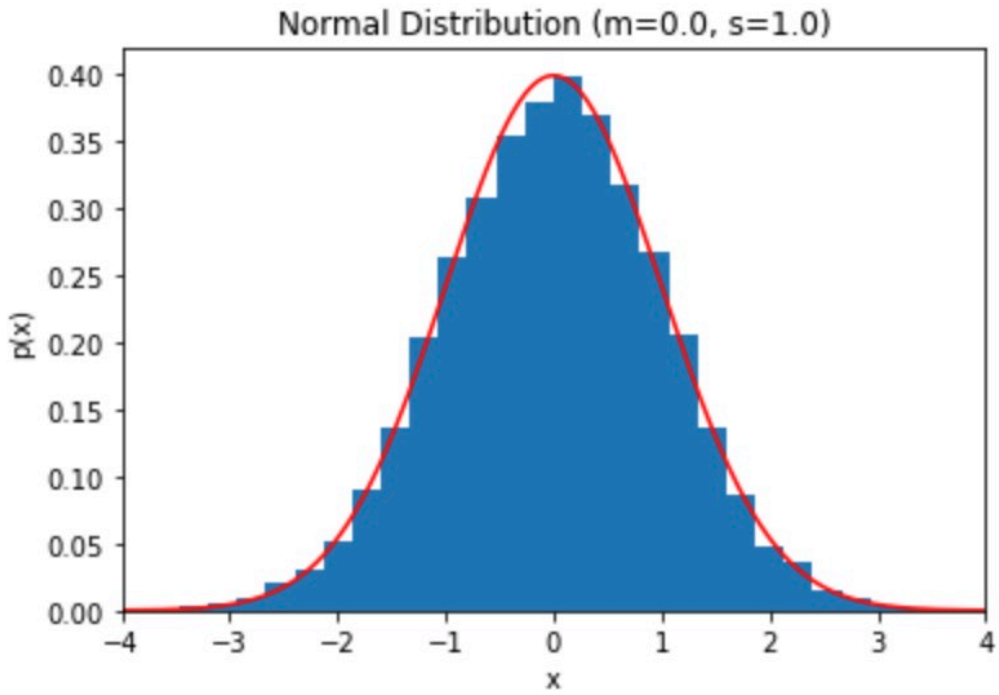
```
# pdf
plt.plot(x, p)
plt.xlabel("x")
plt.ylabel("p(x)")
plt.title(f"Normal Distribution (m={m}, s={s})");

# histogram
plt.bar(hb[:len(hb)-1], h, width=(xmax-xmin)/nbins, align='edge')
plt.xlabel('x')
plt.ylabel('count')
plt.title(f'randn()\nN = {N:,}')
plt.xlim((xmin, xmax));
```

both the line plot ...

... and the bar plot are on the same graph

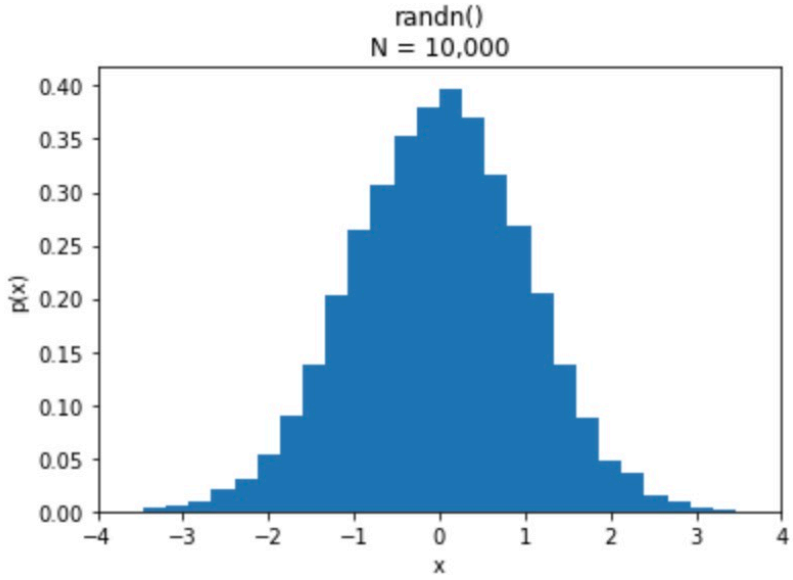
overwrites the labels and titles

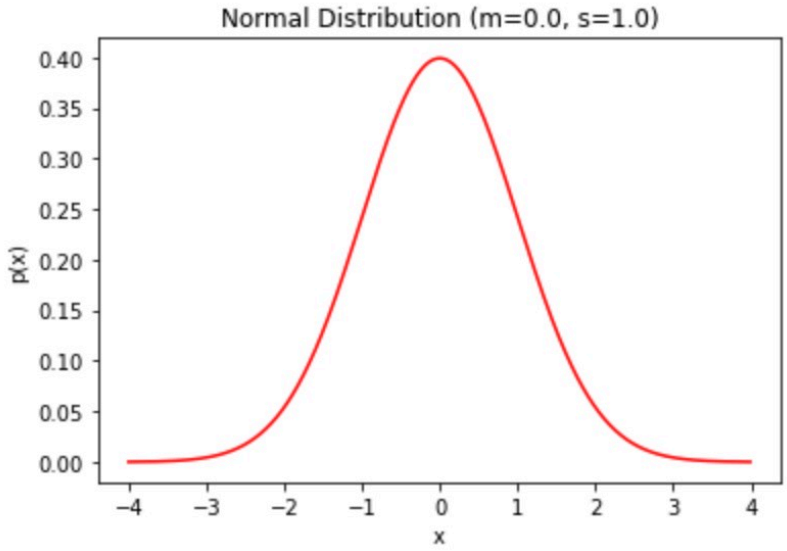


this forces two separate figures to be generated

```
# pdf
plt.plot(x, p)
plt.xlabel("x")
plt.ylabel("p(x)")
plt.title(f"Normal Distribution (m={m}, s={
plt.show()

# histogram
plt.bar(hb[:len(hb)-1], h, width=(xmax-xmir
plt.xlabel('x')
plt.ylabel('count')
plt.title(f'randn()\nN = {N:,}')
plt.xlim((xmin, xmax));
plt.show()
```

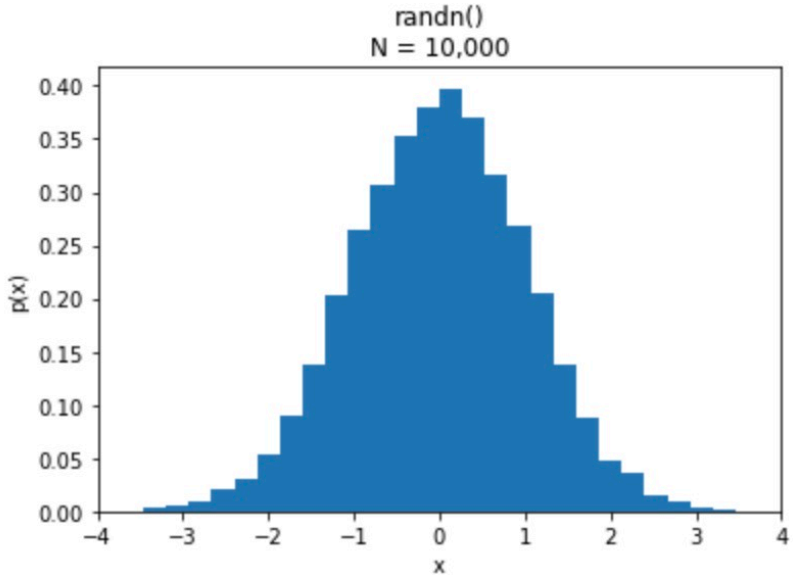


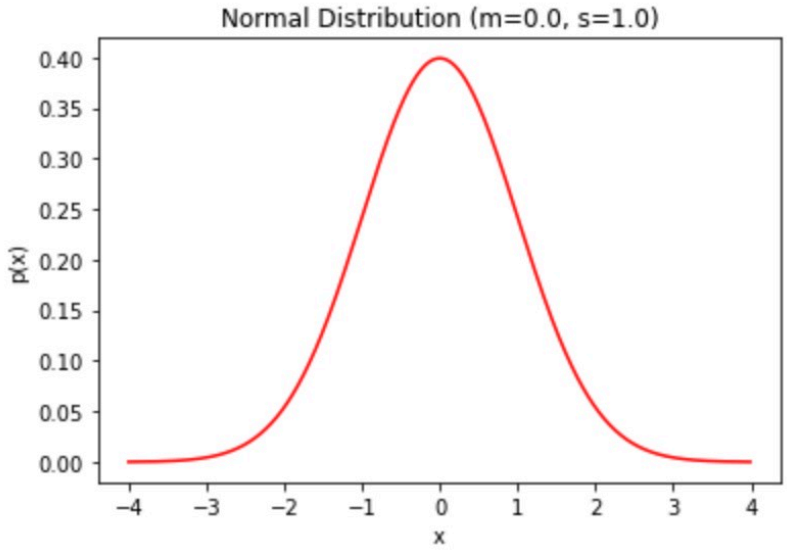


this forces two separate figures to be generated

```
# pdf
plt.figure()                    create a "figure"
plt.plot(x, p)                  draw it
plt.xlabel("x")
plt.ylabel("p(x)")
plt.title(f"Normal Distribution (m={m}, s={

# histogram
plt.figure()                    create a "figure"
plt.bar(hb[:len(hb)-1], h, width=(xmax-xmir
plt.xlabel('x')                 draw it
plt.ylabel('count')
plt.title(f'randn()\nN = {N:,}')
plt.xlim((xmin, xmax));
```





Matlab style of figure generation (non-Pythonic)

```
# pdf
plt.plot(x, p)
plt.xlabel("x")
plt.ylabel("p(x)")
plt.title(f"Normal Distribution (m={m}, s={s})");
plt.show()
```

Matlab-style matplotlib interface is "stateful"

it keeps track of (via an internal state) the current figure and axes to draw/plot to

https://matplotlib.org/stable/users/explain/api_interfaces.html

Object-oriented style of figure generation (Pythonic)

declare figure and axes (objects)

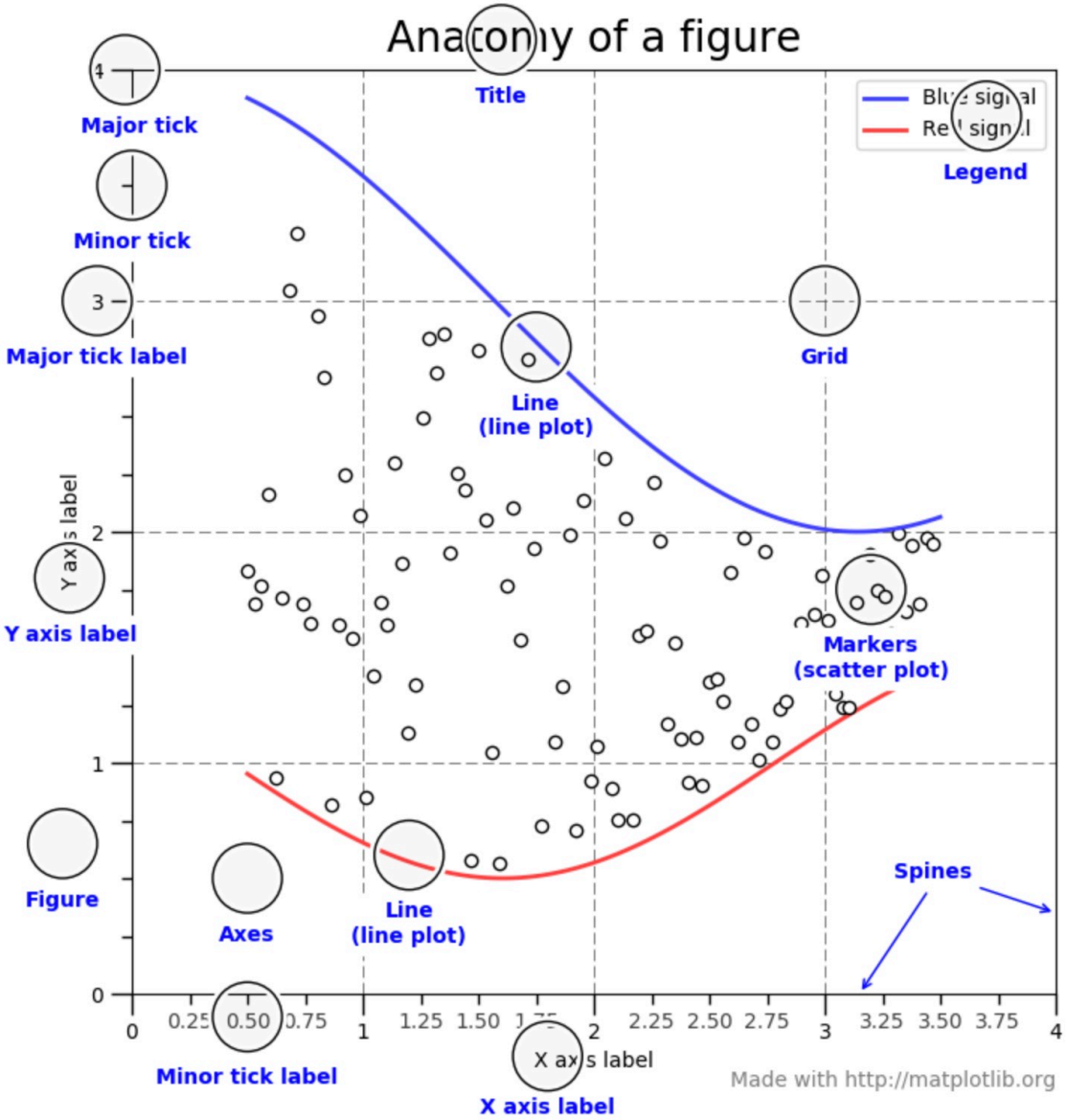
```
fig, ax = plt.subplots()
```

can hold multiple plots

can hold multiple axes

can do this

```
fig = plt.figure()
ax = plt.axes()
```



```
fig = plt.figure()
```

arguments and methods associated with figure
https://matplotlib.org/api/_as_gen/matplotlib.figure.Figure.html

```
ax = plt.axes()
```

arguments and methods associated with axes
https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.axes.html

Matlab style

```
plt.plot(x, p, 'r-')
plt.xlabel('x')
plt.ylabel('p(x)')
plt.title(f'Normal Distribution');
```

object-oriented style

```
ax2.plot(x, p, 'r-')
ax2.set_xlabel('x')
ax2.set_ylabel('p(x)')
ax2.set_title(f'Normal Distribution');
```

- `plt.xlabel()` → `ax.set_xlabel()`
- `plt.ylabel()` → `ax.set_ylabel()`
- `plt.xlim()` → `ax.set_xlim()`
- `plt.ylim()` → `ax.set_ylim()`
- `plt.title()` → `ax.set_title()`

plt.subplots()

creates multiple plots (axes) on the same figure

```
fig, axs = plt.subplots(2, 2)
x = np.linspace(0, 10, 1000)
axs[0,0].plot(x, np.sin(x), 'r-')
axs[0,1].plot(x, np.cos(x), 'g-')
axs[1,0].plot(x, np.sqrt(x), 'b-')
axs[1,1].plot(x, np.log(x+1), 'b-')
```

multiple plots on the same axes

```
fig = plt.figure(figsize=[10,3])
ax = plt.axes(polar=False)

ax.plot(x, x, label='linear')
ax.plot(x, x**2, label='quadratic')
ax.plot(x, x**3, label='cubic')
```

unlike Matlab, where you need to do "hold on" to avoid overwriting