# Programming for psychologists

**Lecture 4: Data visualization**

Matthias Nau

# Data viz in Python

# Why do good data viz skills matter?

### The obvious

- Summarizing complex information to make inferences.
- Finding patterns in data that may otherwise be missed (incl. outliers and anomalies).
- Communicating results to (non-expert) readers.
- Enhancing credibility of your research while reducing risk of misinterpretation.

### The less obvious (maybe)

- Plotting makes coding more fun: you see what your code is doing.
- Plotting data after each processing step is central for debugging.
- Good figures get people's attention, bad figures can make them turn away.
  You will feel the difference when attending a conference.

# Key libraries for data viz in Python

There are **many libraries in Python** with data visualization functionality

**Most relevant to research psychologists are the following:**

- **Matplotlib**: Foundational library for creating static and animated plots.
- **Seaborn**: High-level interface for beautiful graphics, built on top of Matplotlib
- **Pandas**: Built-in visualization functions for dataframes, built on top of Matplotlib
- **Plotly** & **Bokeh**: Interactive plots, including web-based applications (simple example: https://demo.bokeh.org/sliders)

# Matplotlib

# Matplotlib

The **most popular** Python library for creating **almost any type of figure**.

Comes with a large range of **customization options** (line color, width, etc).

Heavily inspired by plotting functionality of **MatLab**.

Check out their **Gallery**:
https://matplotlib.org/2.0.2/gallery.html

# Matplotlib – The basics

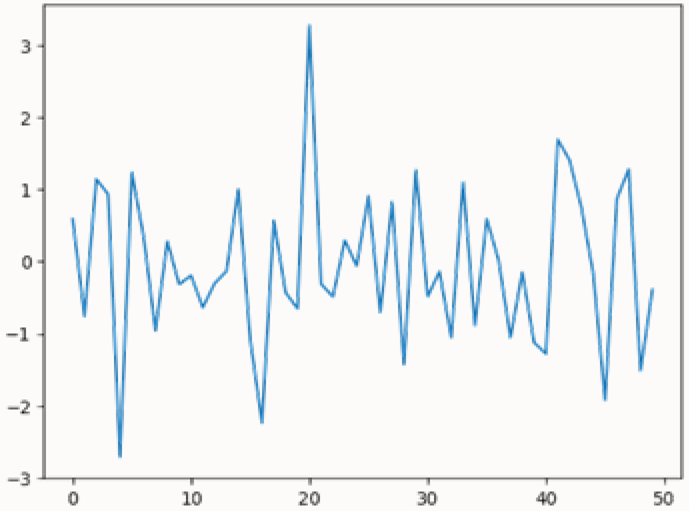
The **most basic way** of using Matplotlib is calling the **plot function**.

**Do this…**

```python
# import libraries
import matplotlib.pyplot as plt
import numpy as np

# plot
plt.plot(np.random.randn(50))
```

*Note: Community convention is to* import matplotlib.pyplot as plt

**…to get that**




*Note: Plotting a few random numbers here generated with NumPy*

This works in notebooks. If you are using scripts, add the **plt.show()** command underneath.

# Matplotlib – The basics

The **plot function** is great for quick visualization, but its customization options are limited. In most cases, what we actually want is a **Figure object**.

A figure object is essentially a **blank canvas** that waits for things to be plotted.

We create a figure (called fig1) with size 3x3 using the following command: **fig1 = plt.figure(figsize=(3,3))**

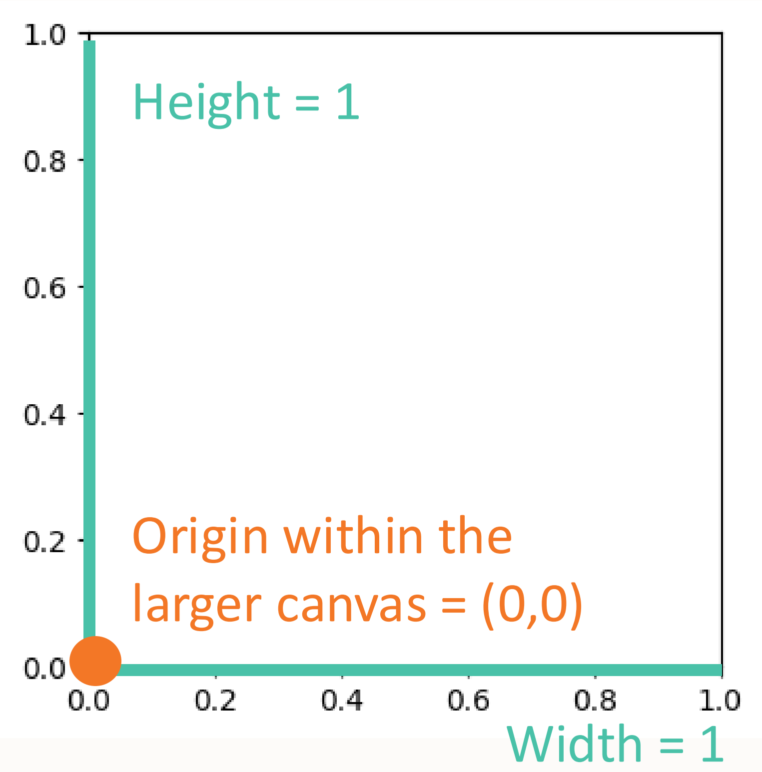
Now, we could plot into fig1 using the plot function, but really **figures** begin to shine after adding **Axes**!

**Nothing to see here yet**

# Matplotlib – The basics

Axes allow **controlling the size and position of the final plot** within the larger canvas

```python
# create figure
fig1 = plt.figure(figsize=(3,3))

# add axes
ax1  = fig1.add_axes([0, 0, 1, 1])
# [left, bottom, width, height]
plt.show()
```

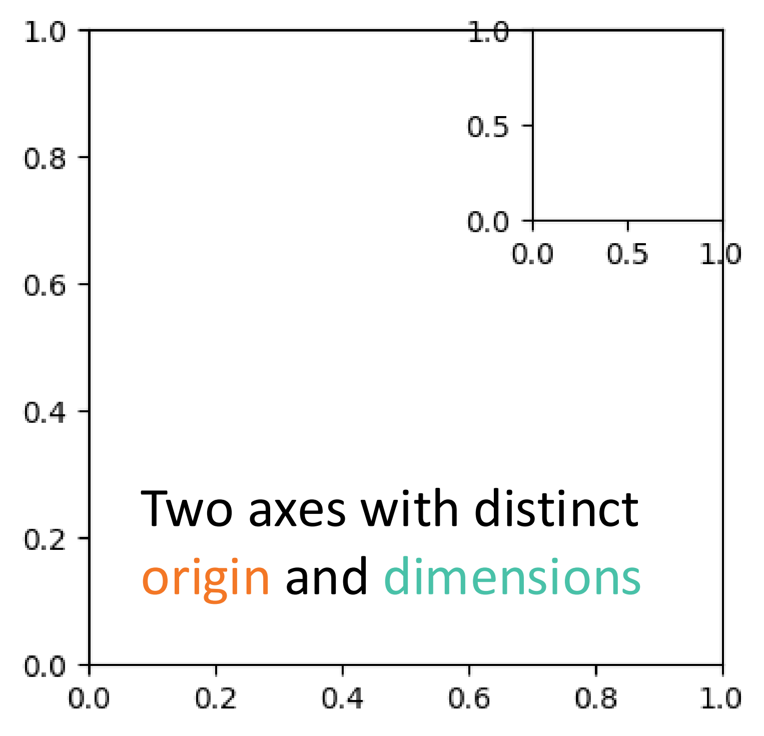



There can be **multiple axes** within the same canvas (i.e. multiple plots)!

# Matplotlib – The basics

Axes allow **controlling the size and position of the final plot** within the larger canvas

```
# create figure
fig1 = plt.figure(figsize=(3,3))

# add axes
ax1  = fig1.add_axes([0, 0, 1, 1])
ax2  = fig1.add_axes([0.7, 0.7, 0.3, 0.3])
plt.show()
```

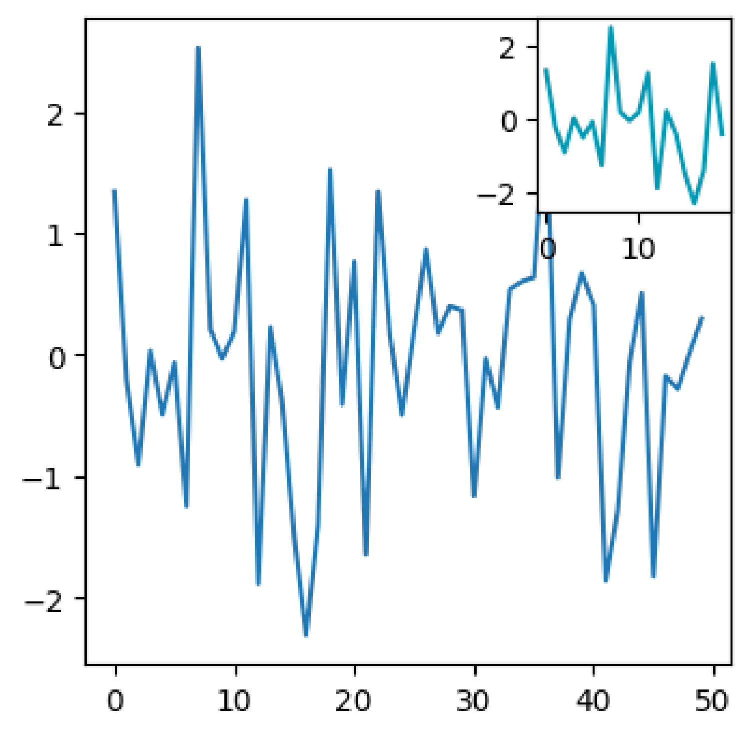



There can be **multiple axes** within the same canvas (i.e. multiple plots)!

# Matplotlib – The basics

Axes allow **controlling the size and position of the final plot** within the larger canvas

```python
# create figure
fig1 = plt.figure(figsize=(3,3))

# add axes
ax1  = fig1.add_axes([0, 0, 1, 1])
ax2  = fig1.add_axes([0.7, 0.7, 0.3, 0.3])

# plot some random data
x = np.random.randn(50)
ax1.plot(x)
ax2.plot(x[0:20], color=[0,0.6,0.7])
plt.show()
```




**Multiple plots within the same figure (e.g., great for inserts)**

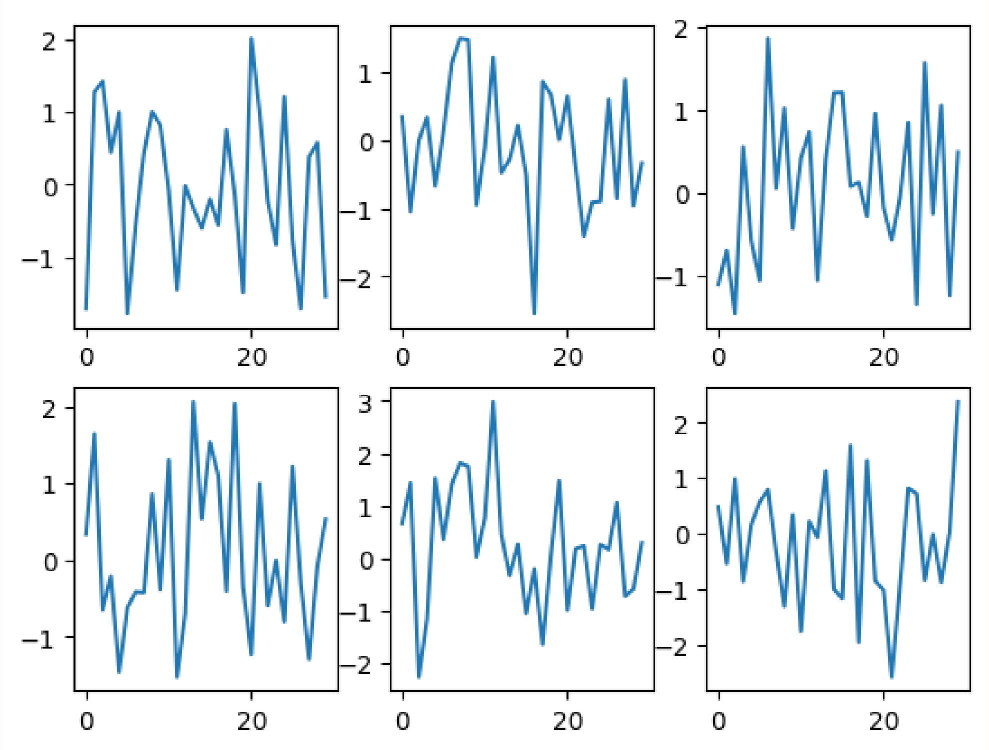
# Matplotlib – The basics

As the number of **subplots** increases, defining all origins and dimensons becomes infeasible.

Luckily, there is pre-written function called **plt.subplots()** that does this for us automatically.

It outputs the **figure object** and the **axes**.

```python
# create figure with subplots
fig, axes = plt.subplots(nrows=2, ncols=3)

# plot random numbers into each
for ax in axes.reshape(-1):
    ax.plot(np.random.randn(30))
```



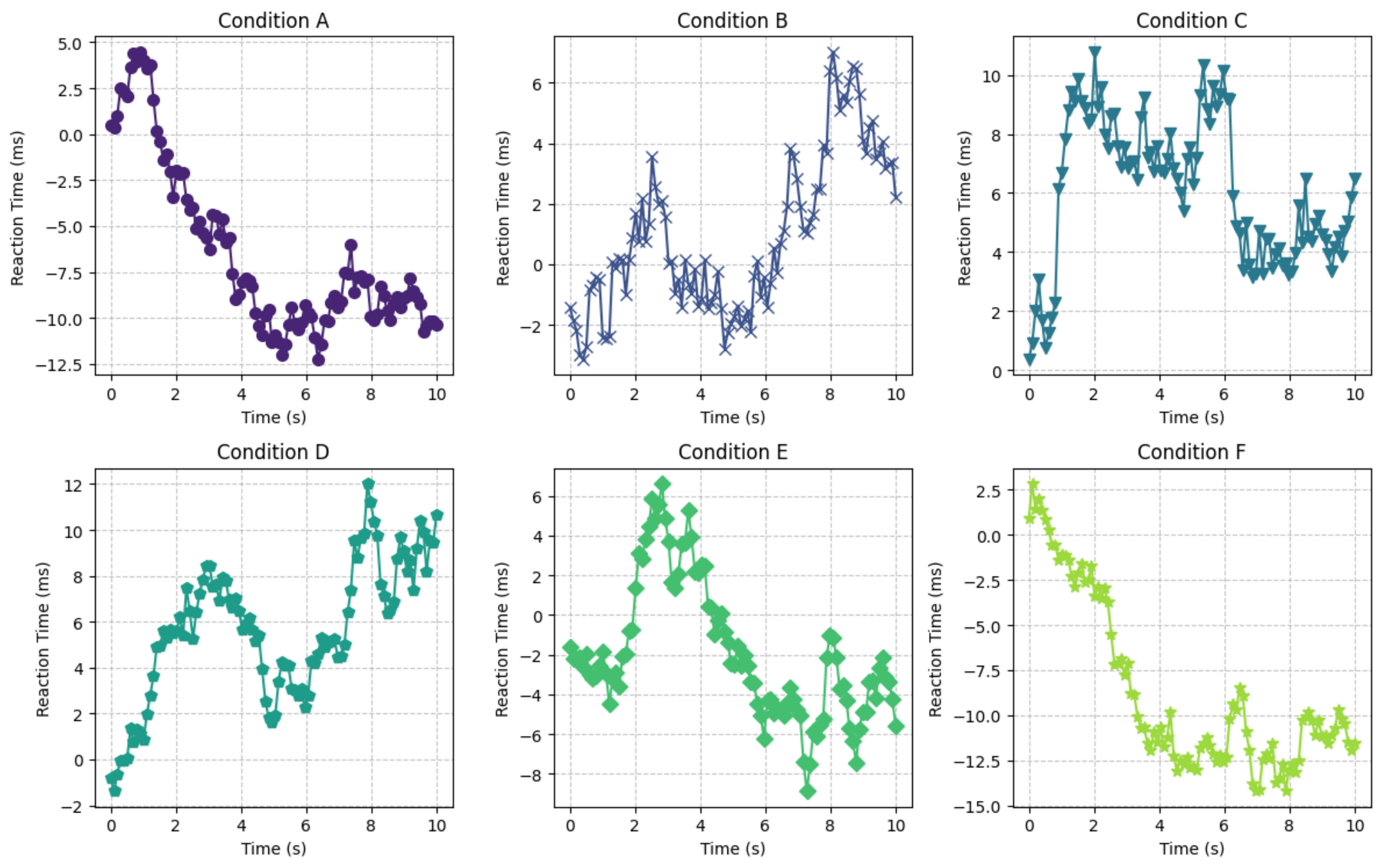
# Matplotlib – Styling & plot types

Matplotlib comes with a wide range of customization options (e.g., to change colors).

```
# fake data
np.random.seed(42)
data       = [np.random.randn(100).cumsum() for ii in range(6)]

# settings
colors     = plt.cm.viridis([0.1, 0.25, 0.4, 0.55, 0.7, 0.85])
conditions = ['Condition A', 'Condition B', 'Condition C', 'Condition D', 'Condition E', 'Condition F']
markers    = ['o', 'x', 'v', 'p', 'D', '*']

# Create 2x3 subplots, plot, and customize
fig, axs = plt.subplots(2, 3, figsize=(12, 8))
for i, ax in enumerate(axs.flat):
    ax.plot(np.linspace(0, 10, 100), data[i],
        color=colors[i], label=conditions[i], linestyle='-', marker=markers[i], markersize=7)
    ax.set_title(f'{conditions[i]}', fontsize=12)
    ax.set_xlabel('Time (s)', fontsize=10)
    ax.set_ylabel('Reaction Time (ms)', fontsize=10)
    ax.grid(True, linestyle='--', alpha=0.7)

# Adjust layout to prevent overlap
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

# Matplotlib – Styling & plot types

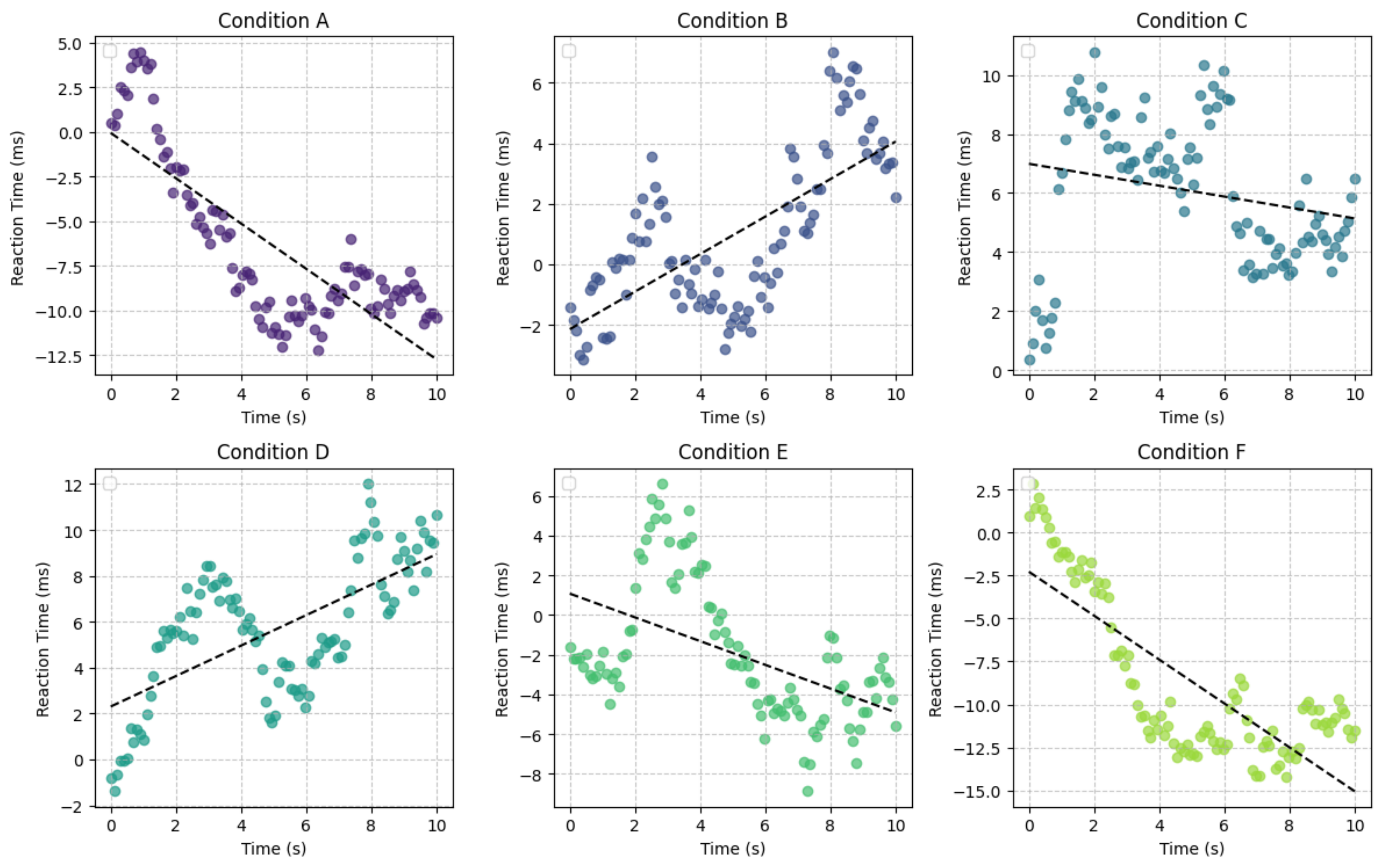
Matplotlib comes with a wide range of customization options (e.g., to change colors).




- Line plots
- Various marker types
- Various colors
- Various labels and titles

# Matplotlib – Styling & plot types

Matplotlib comes with a wide range of customization options (e.g., to change colors).

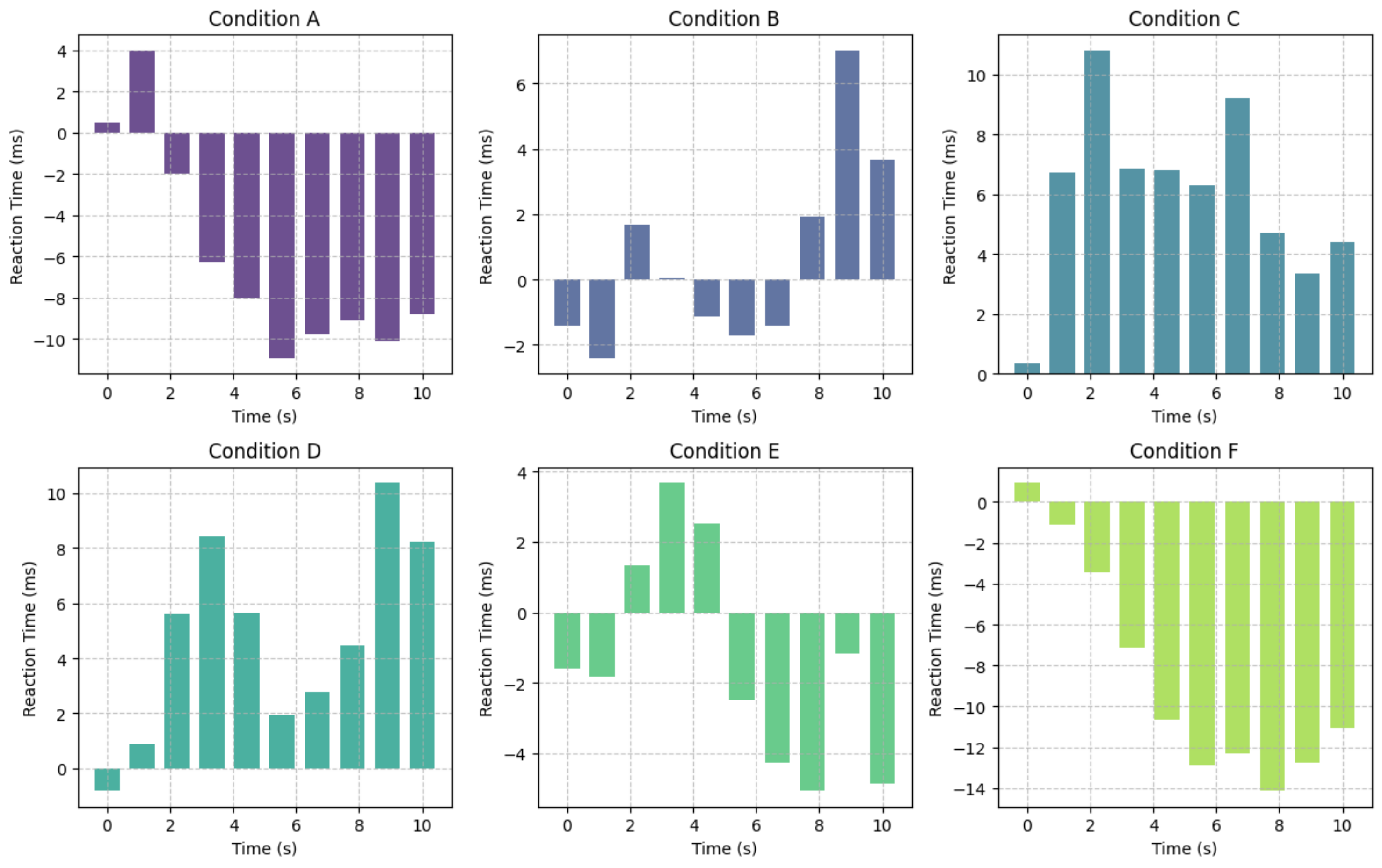



Scatter plots

Dashed trend line

# Matplotlib – Styling & plot types

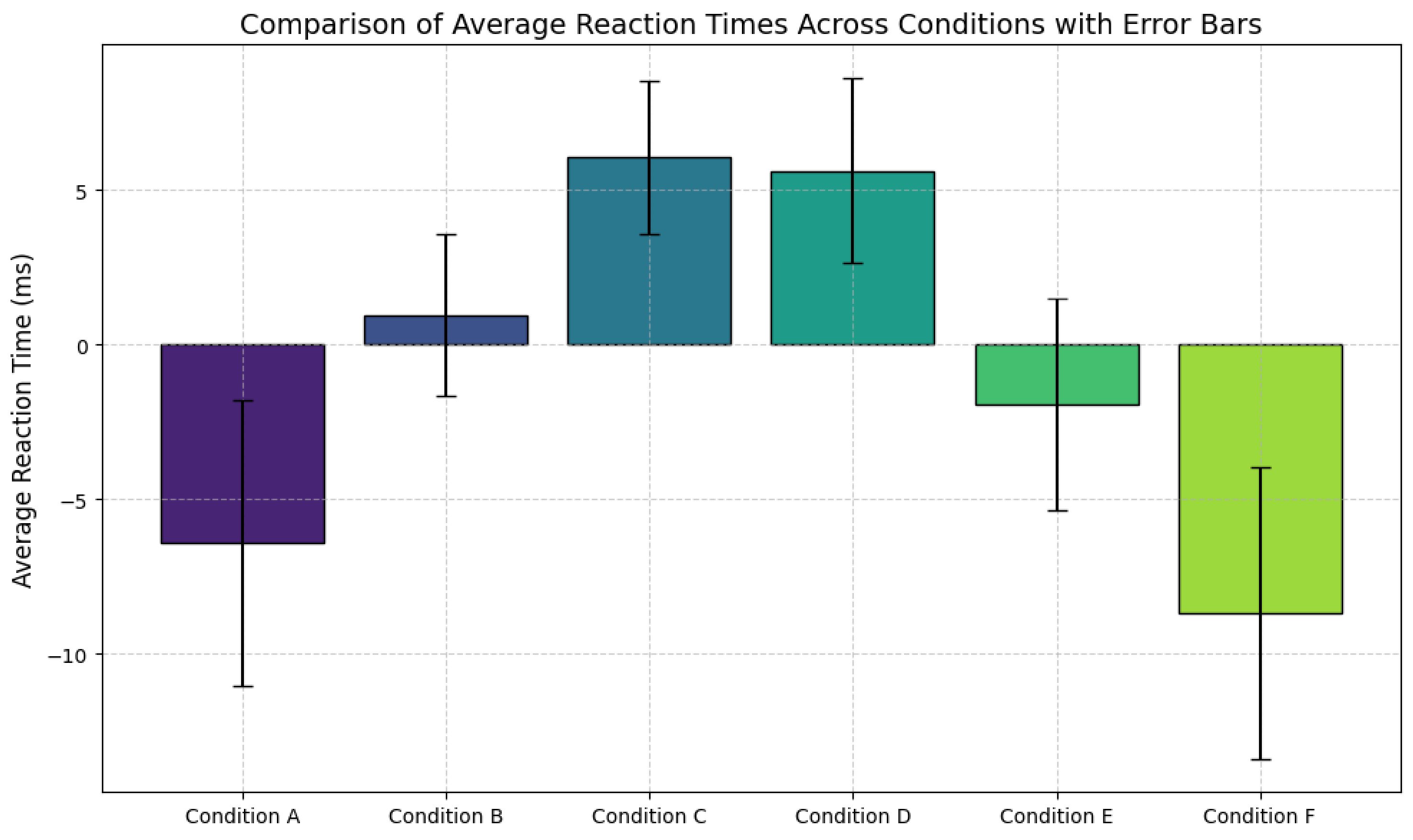
Matplotlib comes with a wide range of customization options (e.g., to change colors).




Bar plots

# Matplotlib – Styling & plot types

Matplotlib comes with a wide range of customization options (e.g., to change colors).

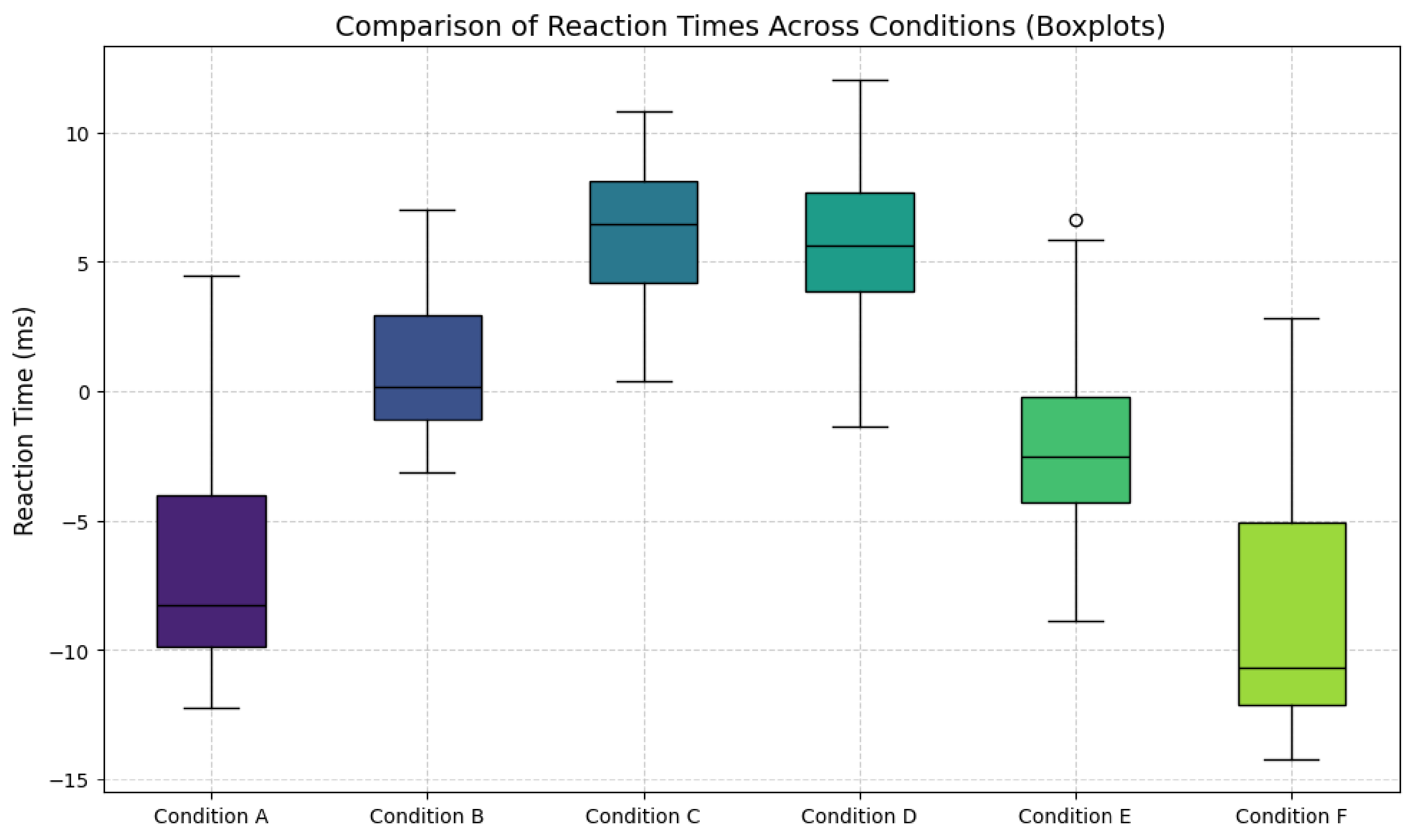



Bar plots

- Box: data mean
- Error bars: standard deviation

# Matplotlib – Styling & plot types

Matplotlib comes with a wide range of customization options (e.g., to change colors).

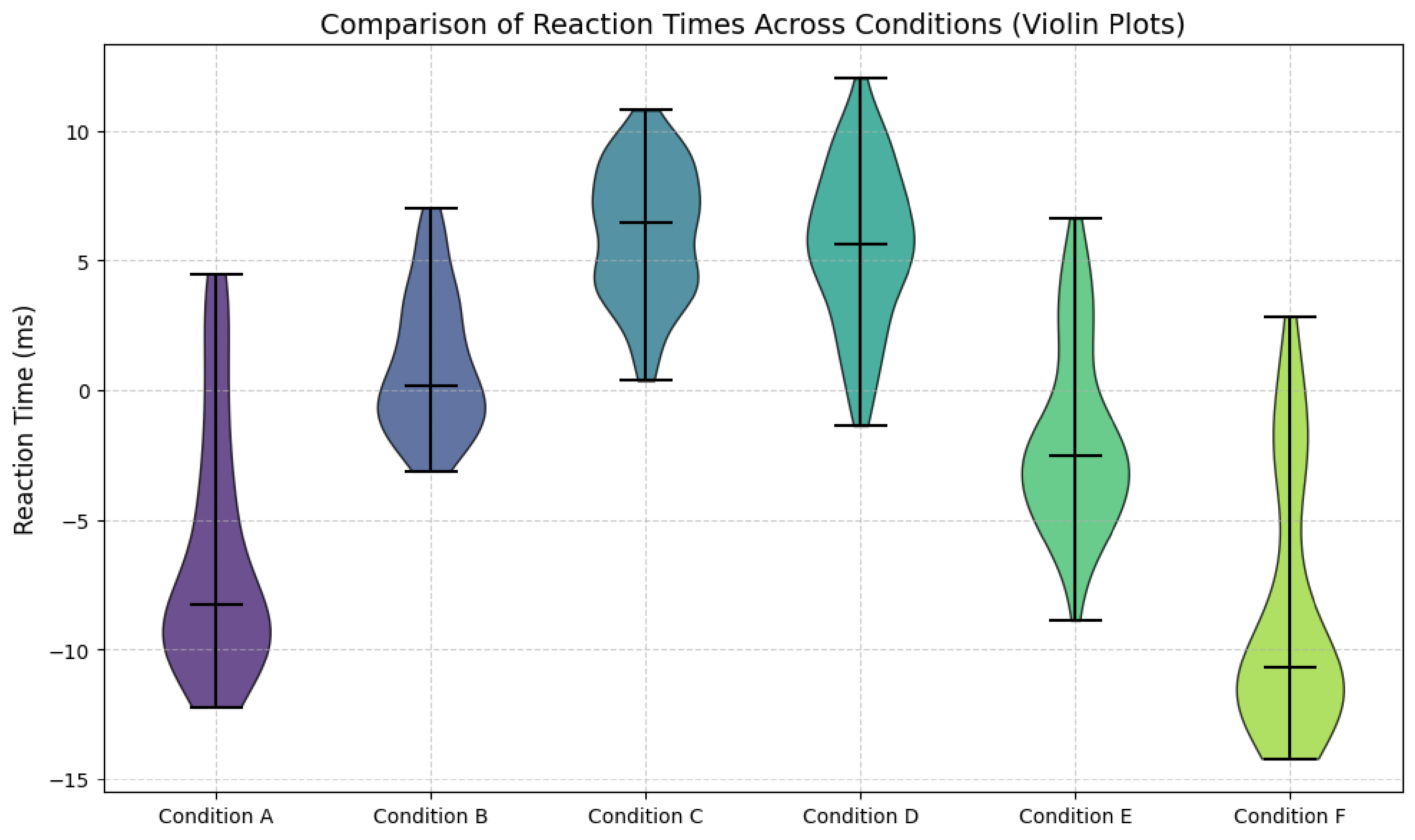



Box plots

- Black line: median
- Box: 50% of data
- Whiskers: 1.5 x interquartile range
- Dots: outliers

# Matplotlib – Styling & plot types

Matplotlib comes with a wide range of customization options (e.g., to change colors).

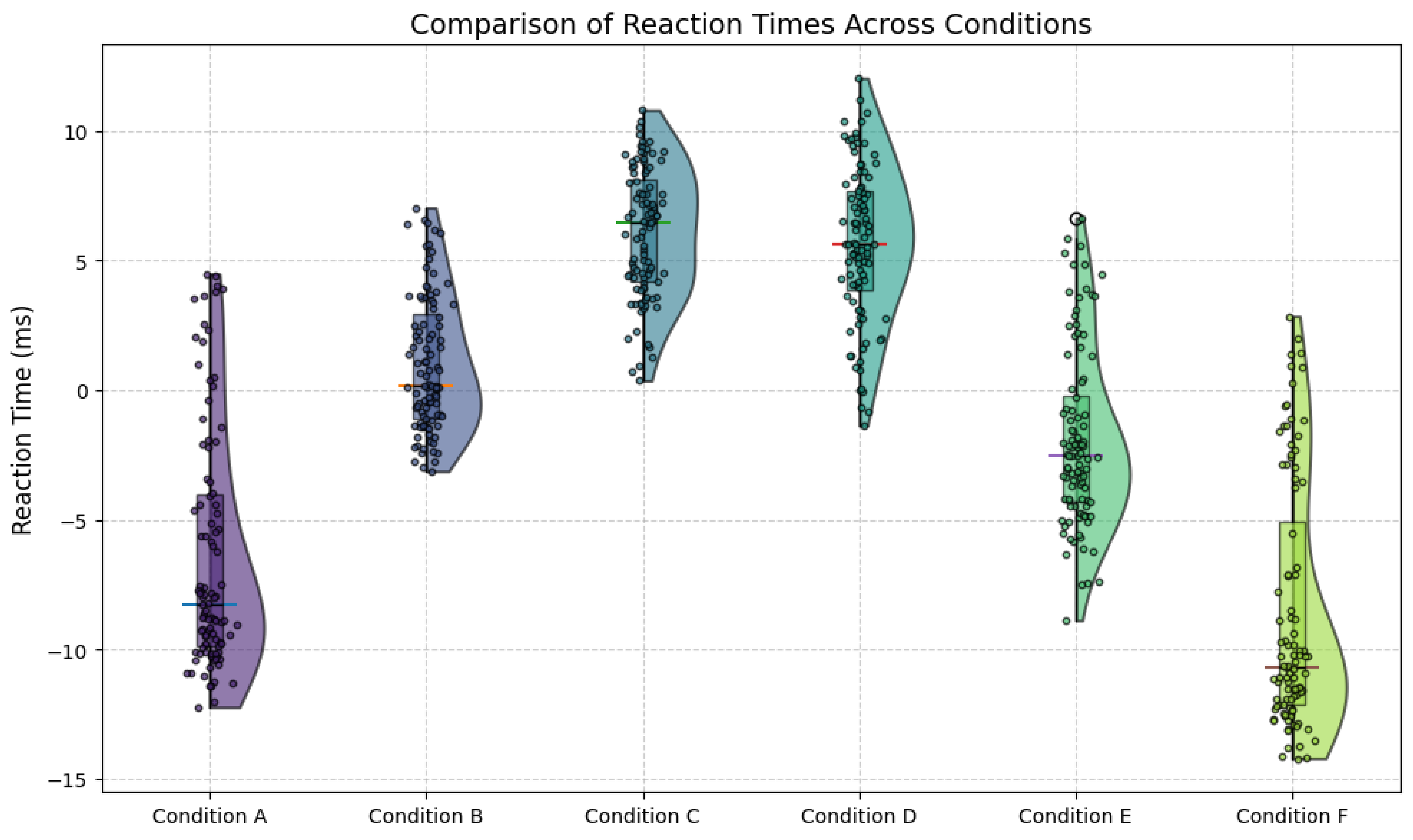



Violin plots

- Black line: median
- Smoother data distribution

# Matplotlib – Styling & plot types

Matplotlib comes with a wide range of customization options (e.g., to change colors).




Rain cloud plots

- Raw data, boxplots, and violin plots all together

**Many more options, check out matplotlib gallery!**
https://matplotlib.org

# Matplotlib – Color palettes

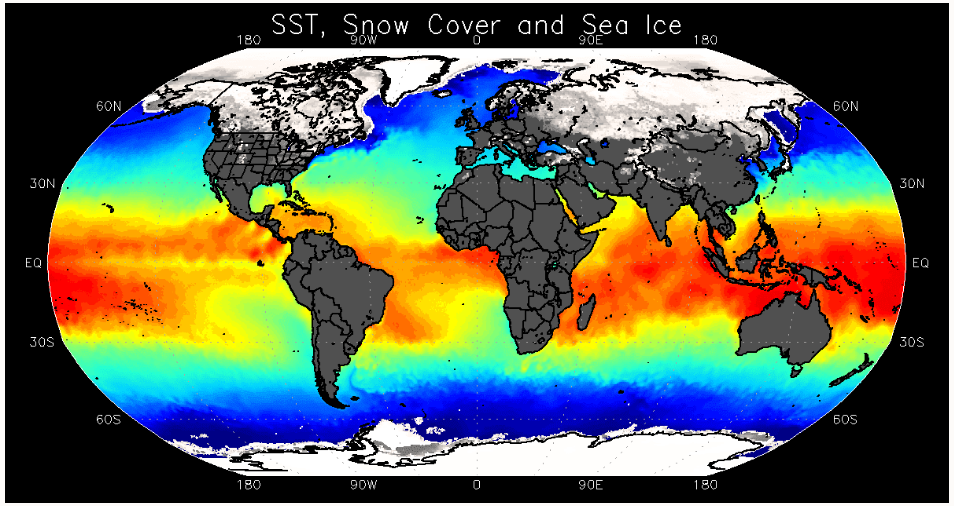
When picking a color palette, make sure they are appropriate for **color blindness**, which concerns ~5% of the population.


Matplotlib's default colormap is **Viridis**, which is perceptually uniform and suited color blindness

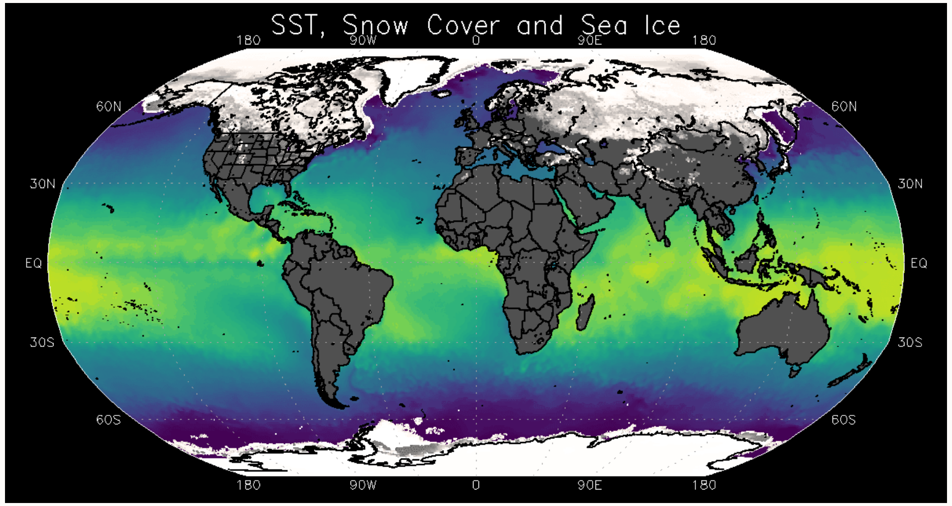
# Matplotlib – Color palettes

When picking a color palette, make sure it is **perceptually uniform,** meaning that a change in color values corresponds to an equal change in perceived color.

**Jet (non-uniform)**




**Viridis (uniform)**




These maps of water temperature show the same data, but temperature gradients look more extreme in the one on the left – falsely so!

Figures source: https://datascience.stanford.edu/news/fixing-non-uniform-colormaps-fixthejet

# Seaborn

# Seaborn - Examples

**Matplotlib** is great for controlling the appearance of your plots, especially highly customized visualizations. A downside is that it requires **many lines of code.**

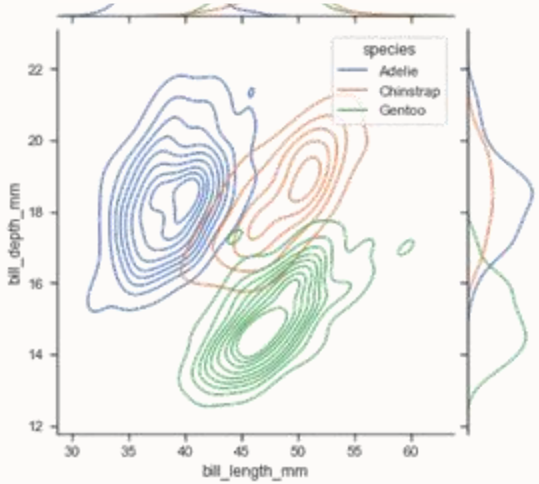
**Seaborn** is an alternative that creates attractive, informative statistical plots quickly, with minimal effort and **fewer lines of code** than Matplotlib.

Built on-top of Matplotlib, Seaborn was designed to work with **Pandas DataFrames**, a super useful DataType that you will want to read up on in the future!

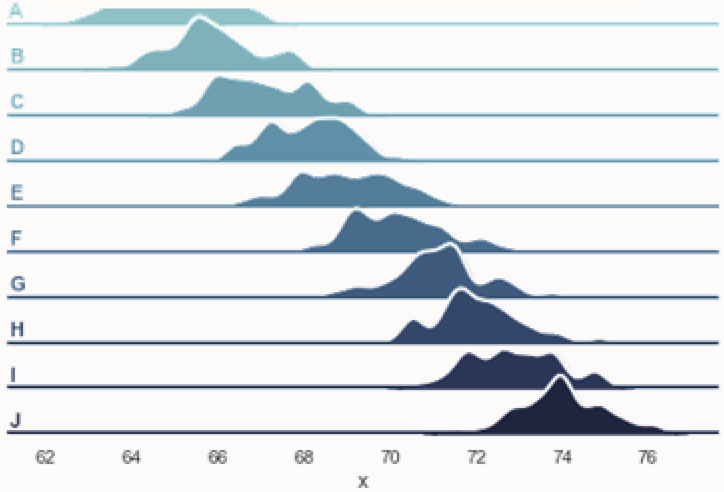
**Check out the Seaborn introduction and tutorial online**
https://seaborn.pydata.org/tutorial/introduction.html

# Seaborn - Examples

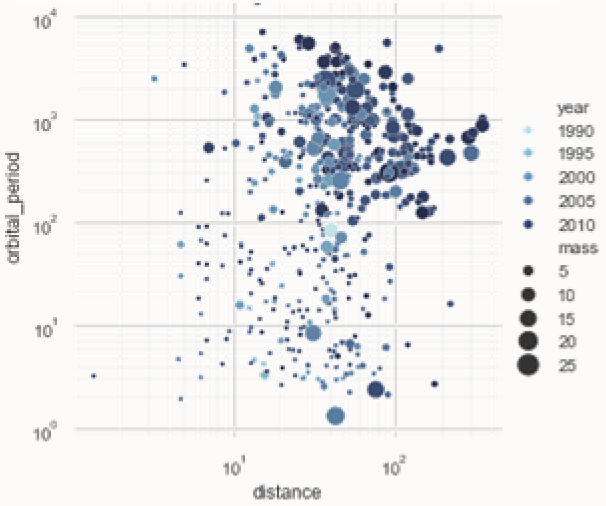
## Joint plots



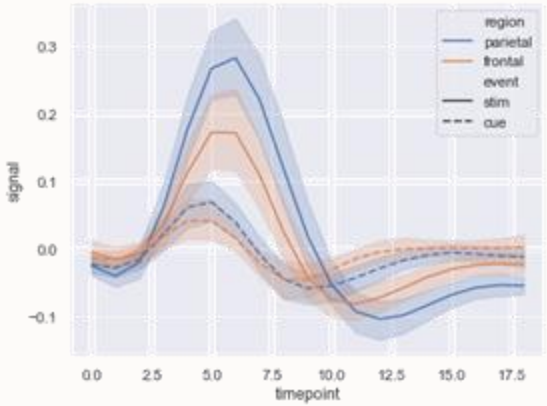

## "Joy Division" plots



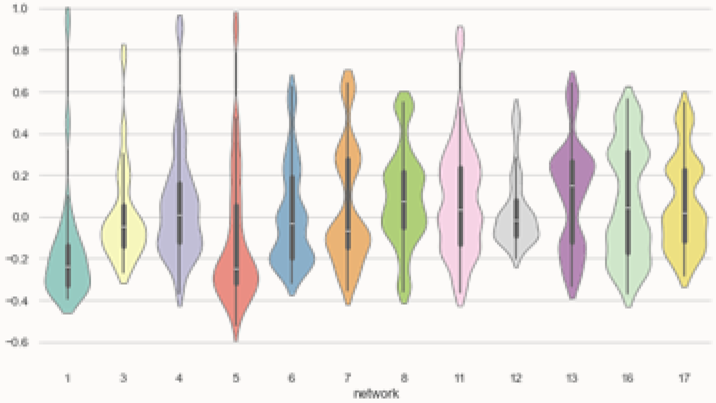

## Great scatter plots




## Shaded error bars




## Simple violinplots



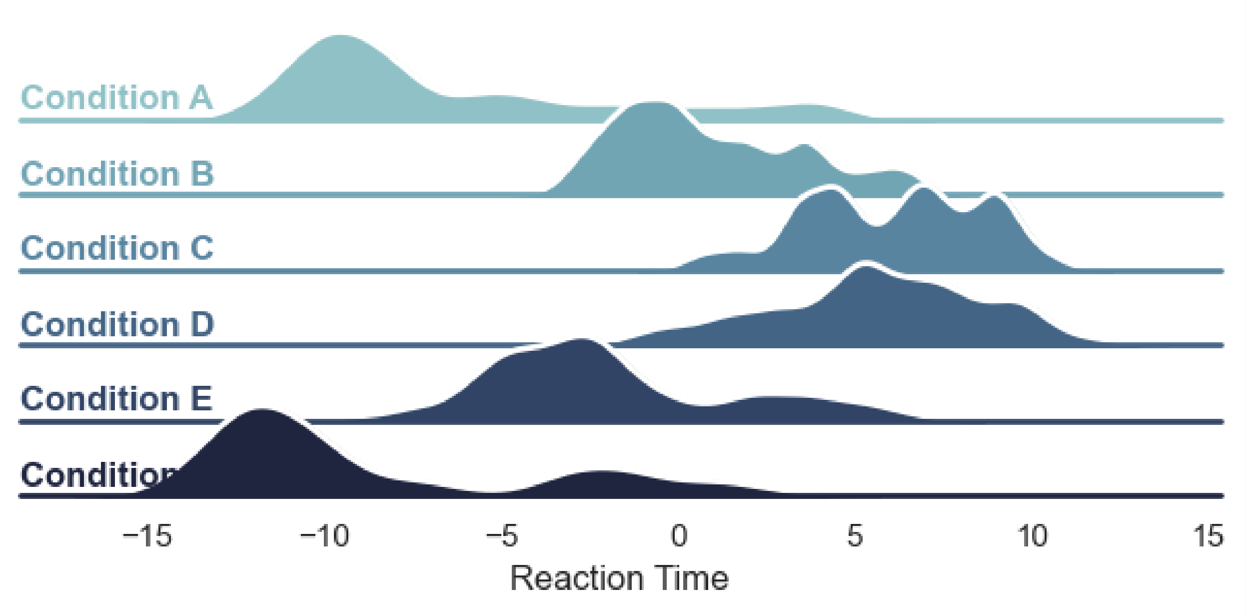

## Annotated heatmaps


Many more!

# Seaborn - Examples

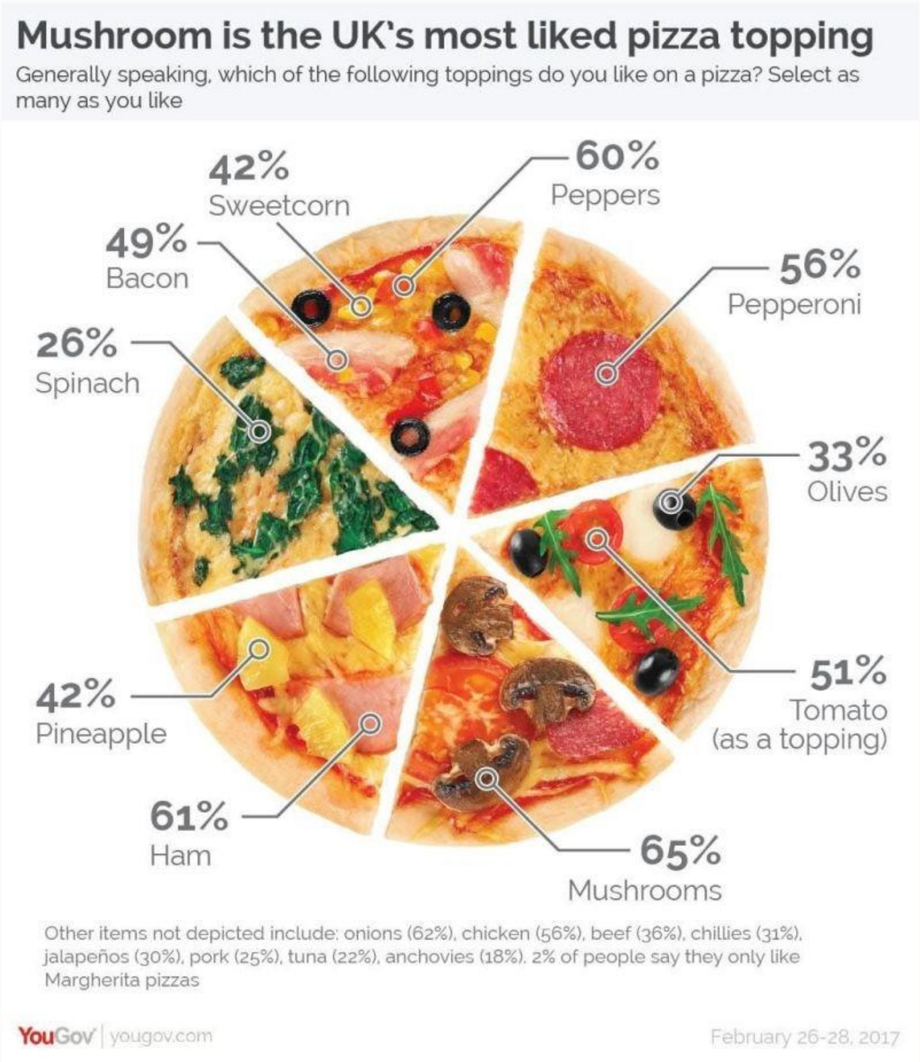
```
# Create DataFrame
df = pd.DataFrame({'Reaction Time': np.concatenate(data),'Condition': np.repeat(conditions, 100)})

# Initialize FacetGrid object
sns.set_theme(style="white", rc={"axes.facecolor": (0, 0, 0, 0)})
pal = sns.cubehelix_palette(6, rot=-.25, light=.7)  # Use a color palette
g   = sns.FacetGrid(df, row="Condition", hue="Condition", aspect=15, height=.5, palette=pal)

# Draw densities
g.map(sns.kdeplot, "Reaction Time", bw_adjust=.5, clip_on=False, fill=True, alpha=1, linewidth=1.5)
g.map(sns.kdeplot, "Reaction Time", clip_on=False, color="w", lw=2, bw_adjust=.5)
g.refline(y=0, linewidth=2, linestyle="-", color=None, clip_on=False)

# label axes coordinates
def label(x, color, label):
    ax = plt.gca()
    ax.text(0, .2, label, fontweight="bold", color=color,
            ha="left", va="center", transform=ax.transAxes)
g.map(label, "Reaction Time")

# adjust figure overlap
g.figure.subplots_adjust(hspace=-.25)
g.set_titles("")
g.set(yticks=[], ylabel="")
g.despine(bottom=True, left=True)
plt.show()
```

The code looks wild, but trust me, creating this figure in Matplotlib would be even wilder!

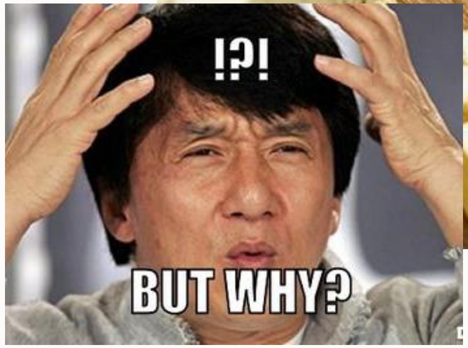


# Data viz fails

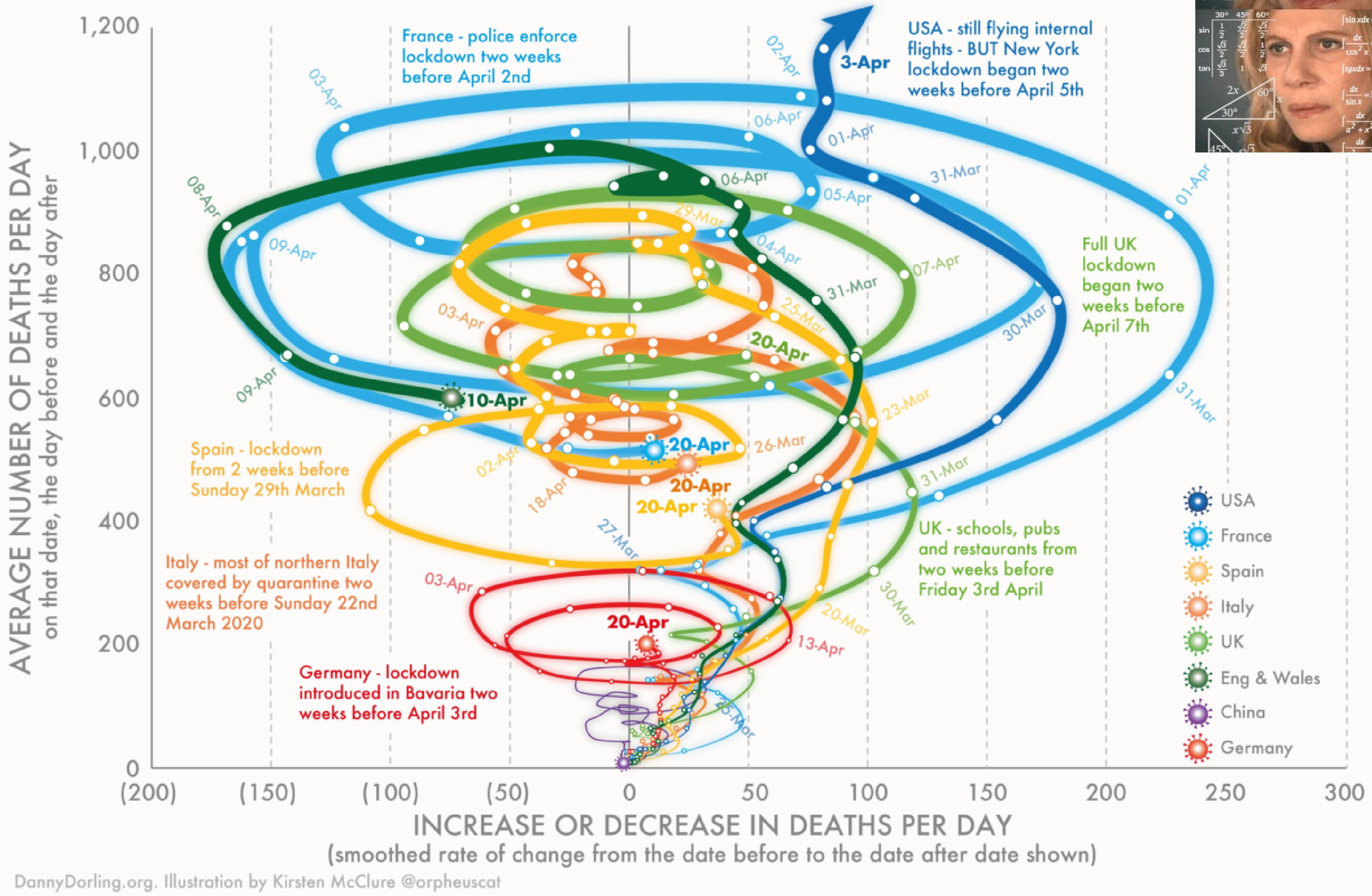
# Data viz fails

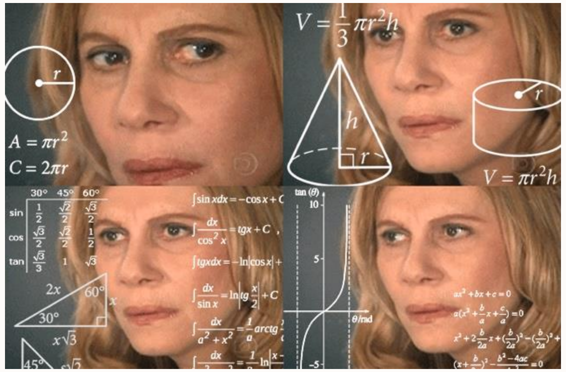




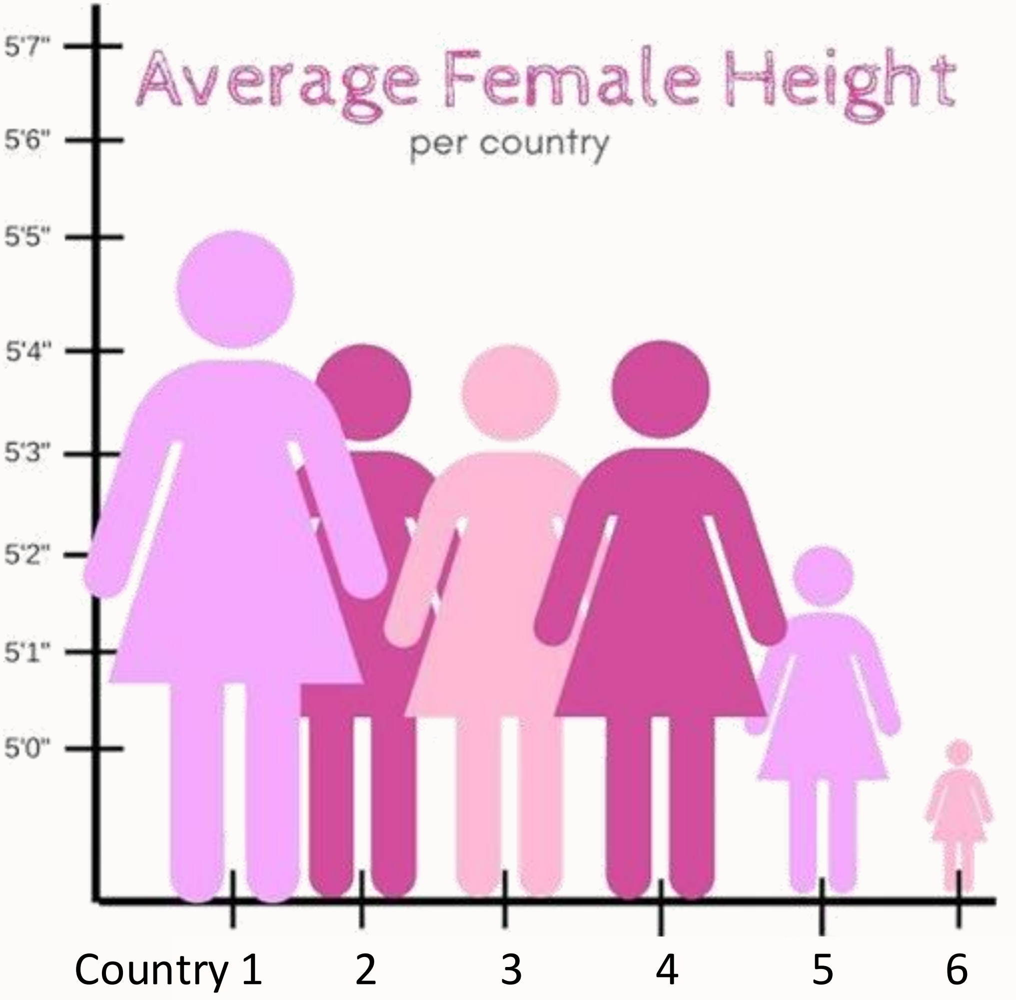



# Data viz fails



# Data viz fails





Tuesday, March 16, 2021

k Nebras-
road condi-
osures.
ult to ask
patient
look out onto
front of
say, 'Hey. This
an and dry,'
the road it
rifted," Hoev-

got crews out.
l crews out
time. We've
f equipment,
ist a function
ing that much
heavy and
Everybody's
t now trying to
ur own side-
d it takes a lot
ment trying
en some of
Our trucks
now,

**Reader Poll**

**Results:** Do you think Nebraska should legalize marijuana?



**New question:** Do you think the City of Gering and Gering Schools should build new tennis courts?

starherald.com/opinion/polls

**Lottery Numbers**

(I checked. It's still illegal!)

## Complete the data survey

1) Go to Canvas, Module 4
2) Click on "Data_survey"
3) Answer the questions
4) Have fun!

# Before the next practical, go through these slides again!

**Do you know what the following terms mean?**

- Matplotlib
- Seaborn
- Plotting
- Figure
- Axes
- Subplots
- Figure styling
- Line plots
- Scatter plots
- Bar plots
- Boxplots
- Violin plots
- Rain cloud plots
- Heatmaps
- Color palette
- Perceptual uniformity

Thanks - See you next week!