Graphing Best Practices

CSS 2 – Spring, 2022

Erik Brockbank

*content heavily borrowed from a graduate lecture by Edward Vul

What We'll Cover

What We'll Cover

1. Ways graphs can go wrong (*there are more than you think…*)

What We'll Cover

1. Ways graphs can go wrong (*there are more than you think…*)

2. How to avoid these pitfalls

What We'll Cover

1. Ways graphs can go wrong (*there are more than you think…*)

2. How to avoid these pitfalls

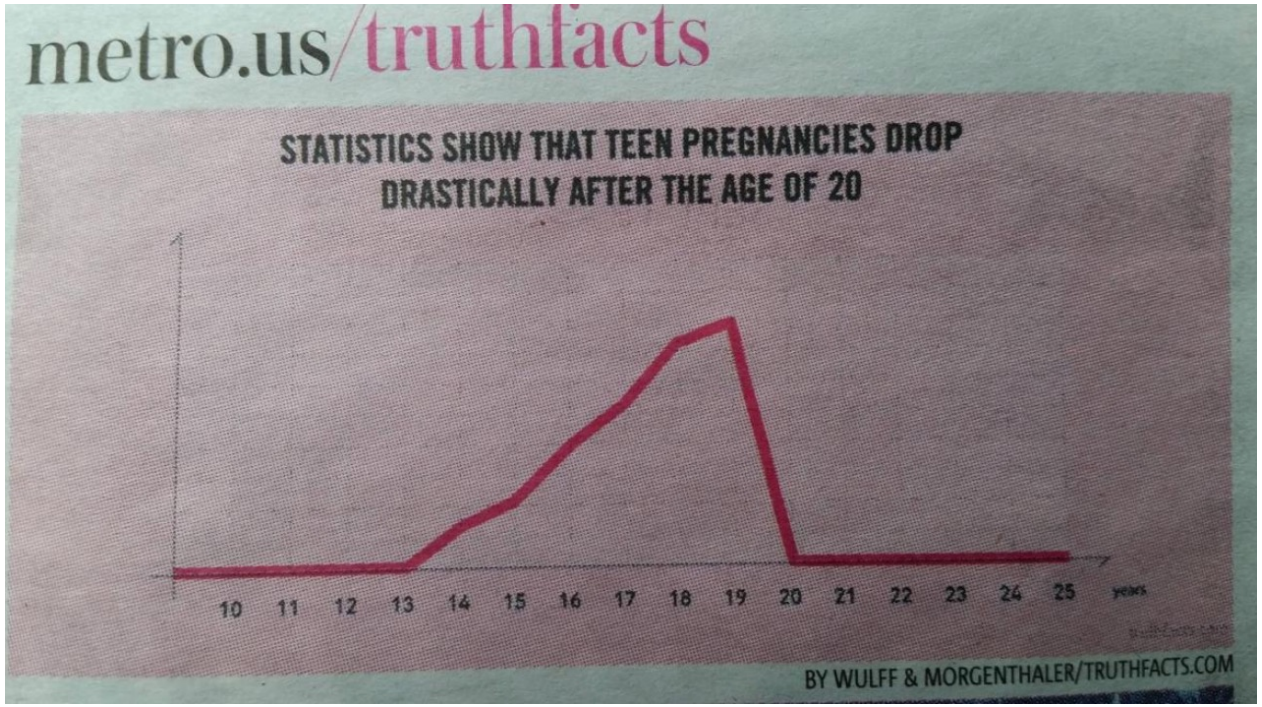
3. Choosing the right visualization

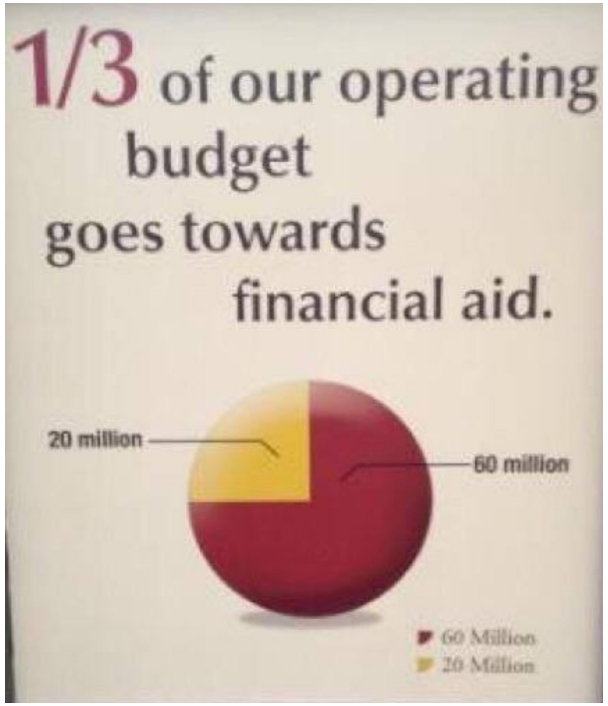
What We'll Cover

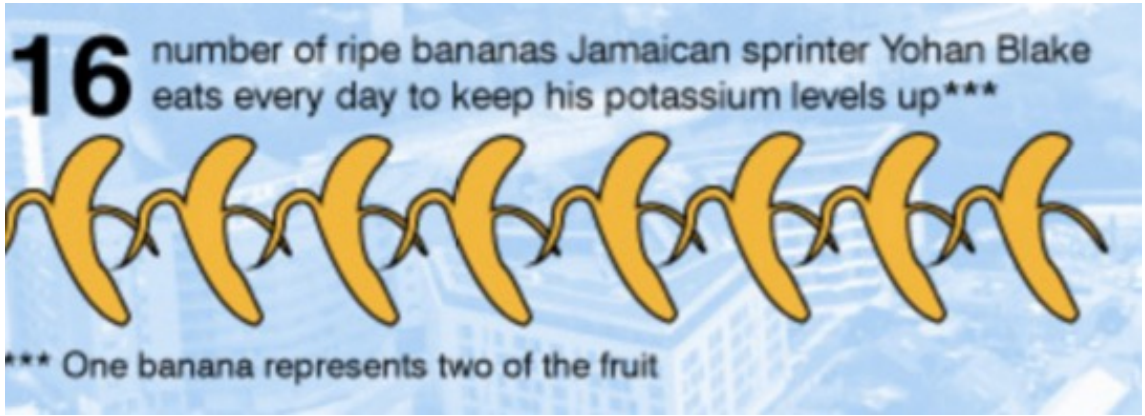
1. Ways graphs can go wrong (*there are more than you think…*)
2. How to avoid these pitfalls
3. Choosing the right visualization

Graphs Gone Awry

Graphs Gone Awry

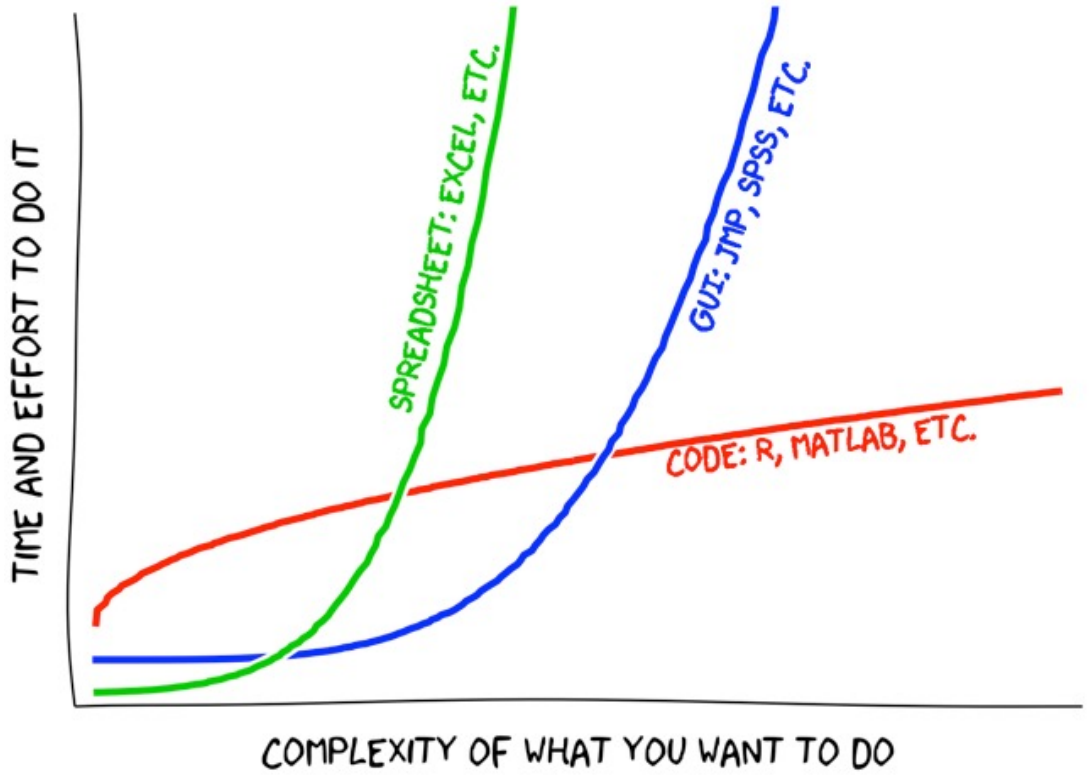






Graphs Gone Awry

Sometimes things that look like graphs are made up…



source: xkcd

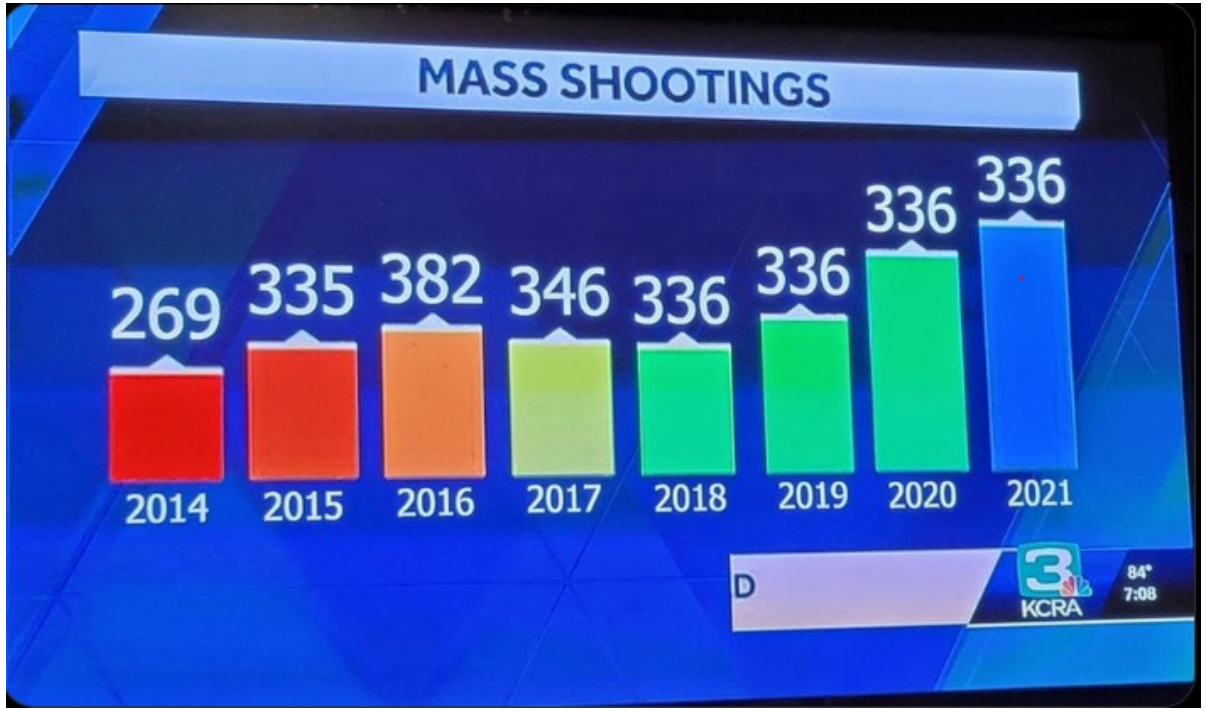
Graphs Gone Awry

Sometimes things that look like graphs are made up…

source: @GraphCrimes

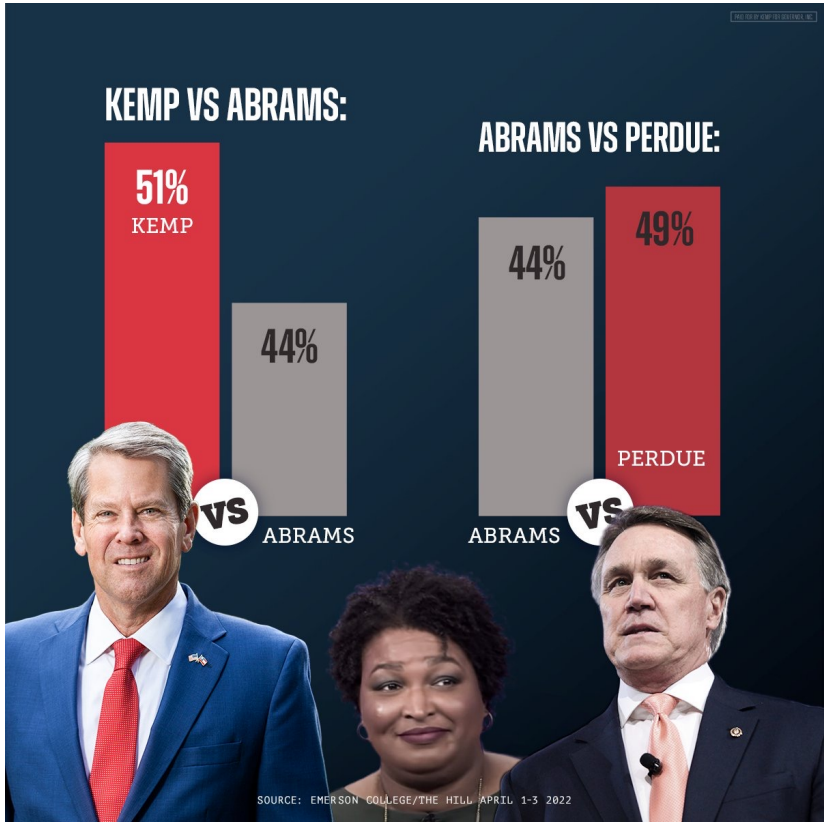
Graphs Gone Awry

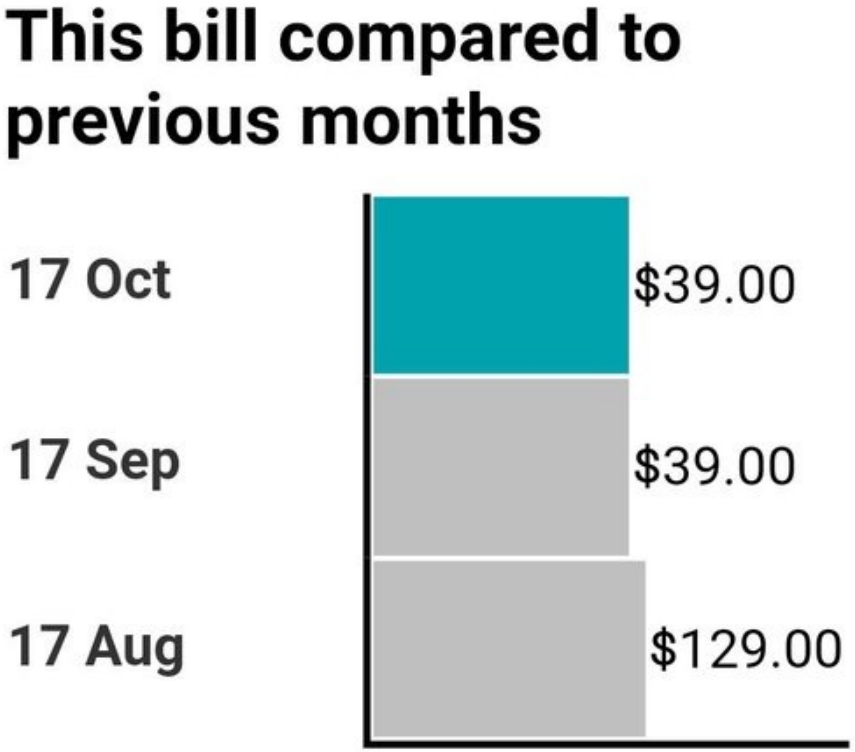
Sometimes it’s a real graph but it doesn’t match the data…



Graphs Gone Awry

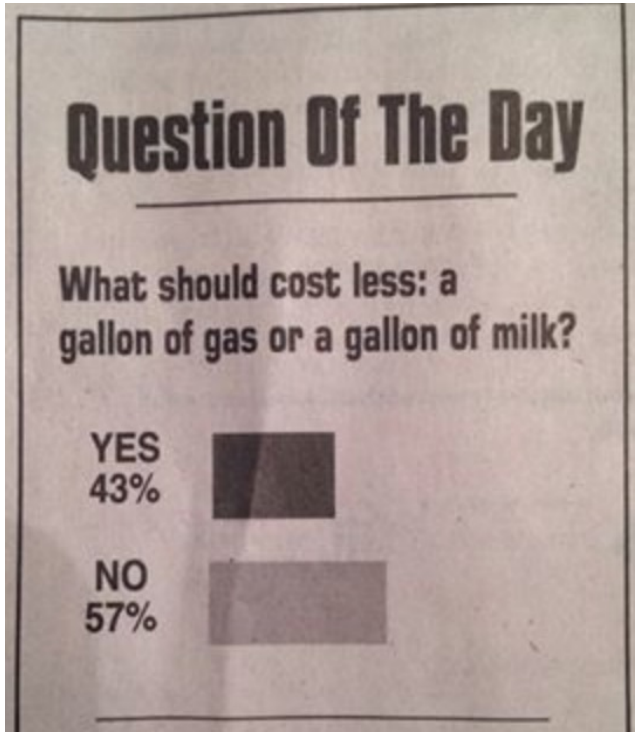
Axis scales can be misleading…

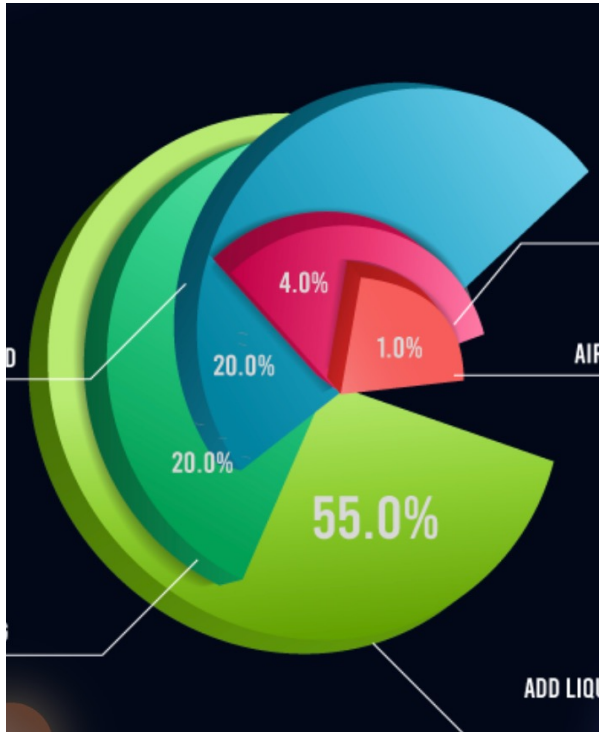




Graphs Gone Awry

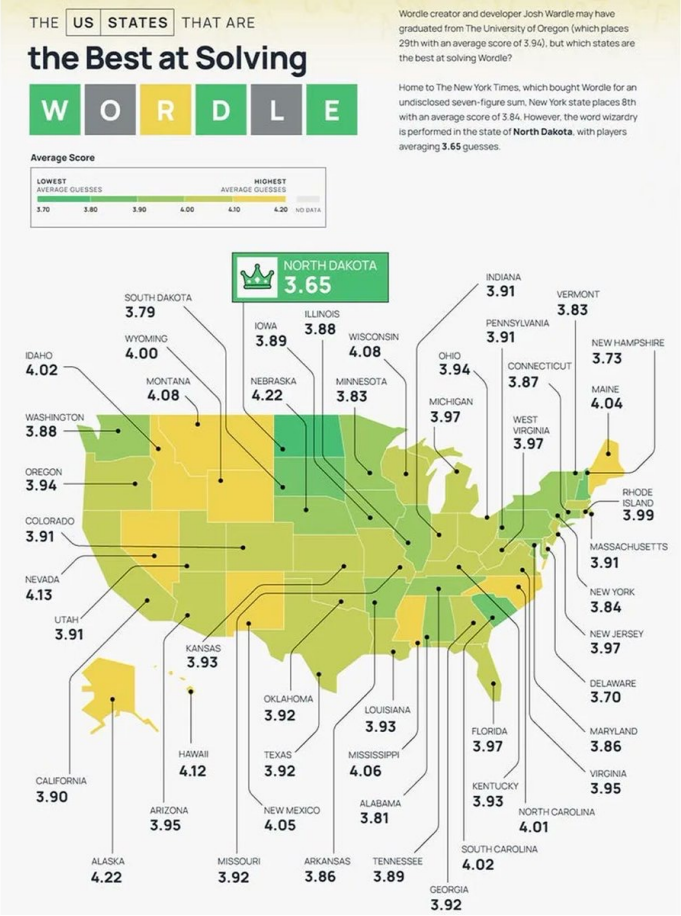
…or not there at all





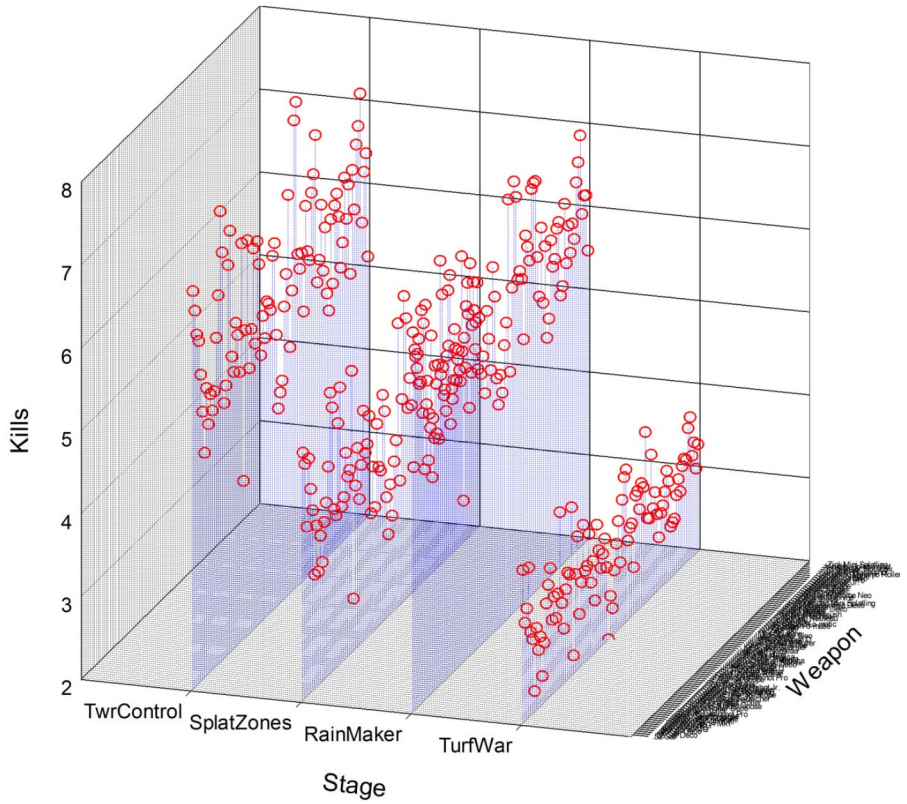
Graphs Gone Awry

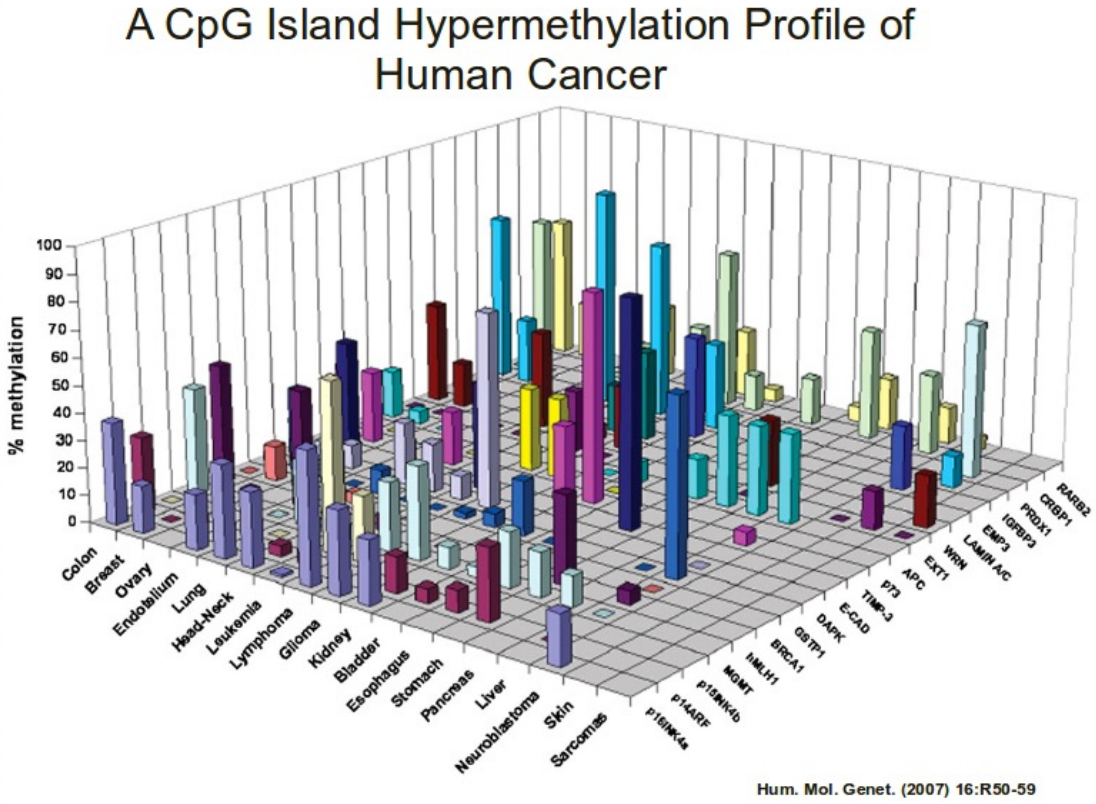
Be careful of bins or gradients that are kind of useless…



Graphs Gone Awry

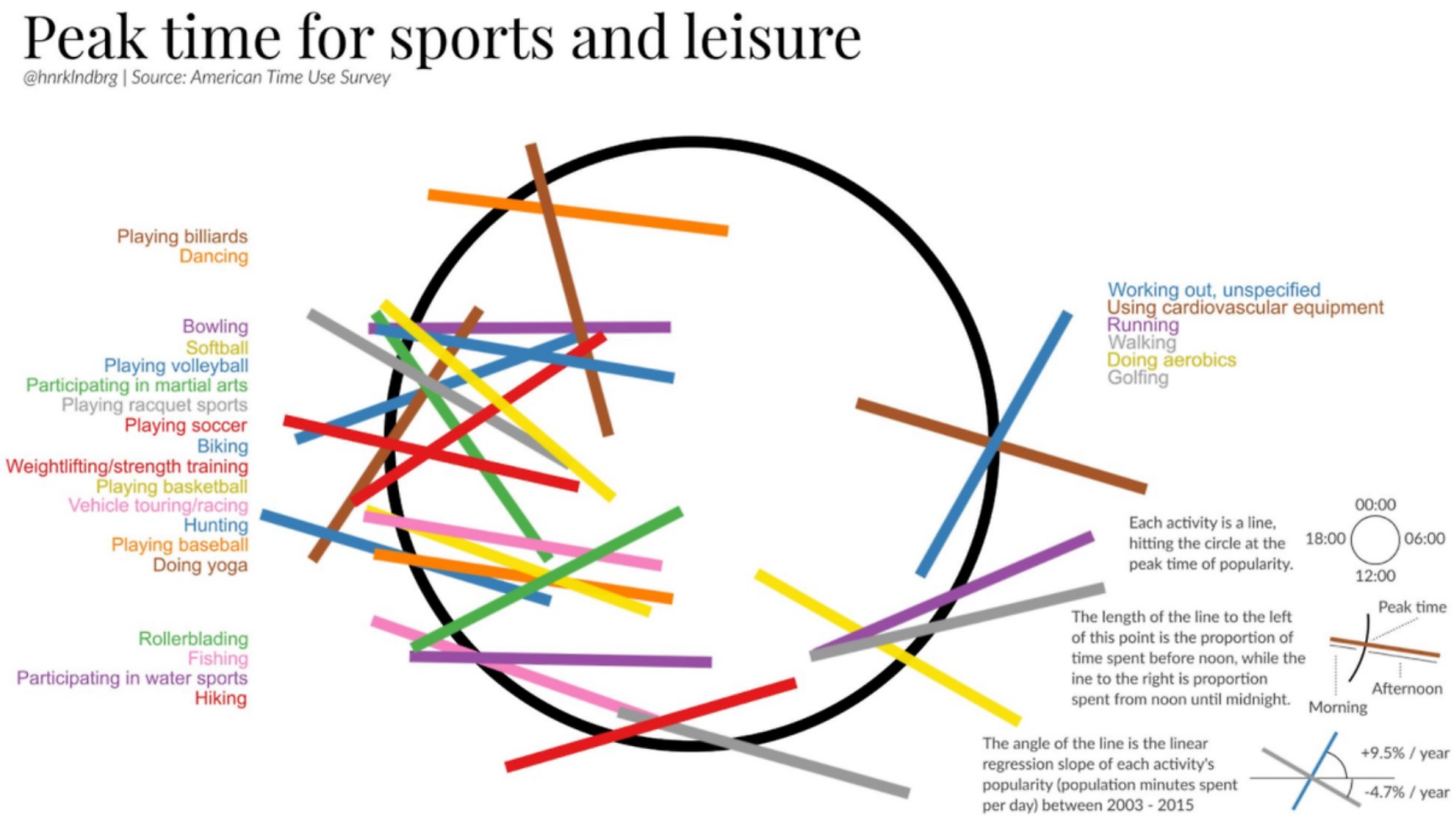
Sometimes, you'll see graphs that are mostly incomprehensible





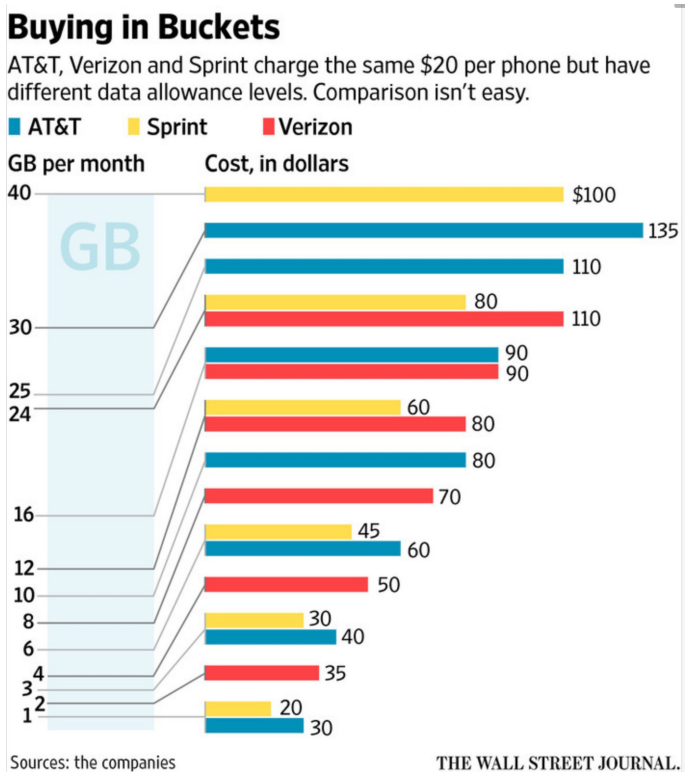
Graphs Gone Awry

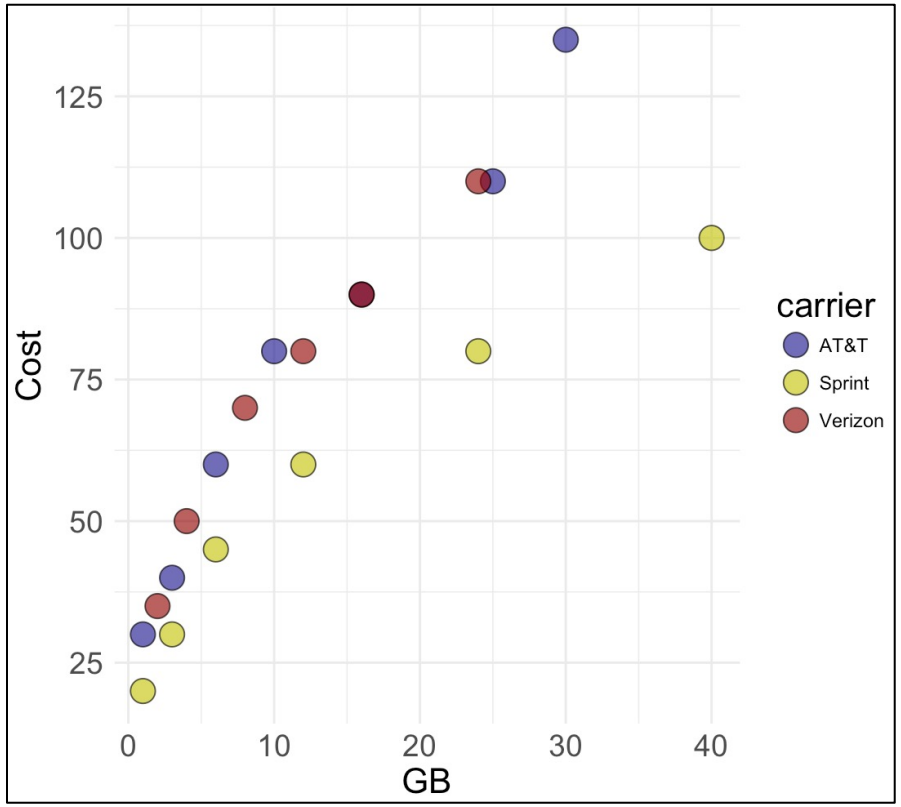
Or, the relationship between the underlying data and the visual presentation is *really stretched*



Graphs Gone Awry

Or, the relationship between the underlying data and the visual presentation is *really stretched*





Graphs Gone Awry

Graphs Gone Awry

In sum, there are a lot of ways graphs can go haywire.

Graphs Gone Awry

In sum, there are a lot of ways graphs can go haywire.

Recognizing this in the world around us is important, and avoiding it in our own work even more so!

What We'll Cover

1. Ways graphs can go wrong (*there are more than you think…*)
2. How to avoid these pitfalls
3. Choosing the right visualization

What We'll Cover

1. Ways graphs can go wrong (*there are more than you think…*)

2. How to avoid these pitfalls

3. Choosing the right visualization

Best Practices

Best Practices

1. Label your axes

Best Practices

1. Label your axes
2. Follow axis conventions

Best Practices

1. Label your axes
2. Follow axis conventions
   → Explanatory variable on x axis
   → Simple and intuitive y variable whenever possible

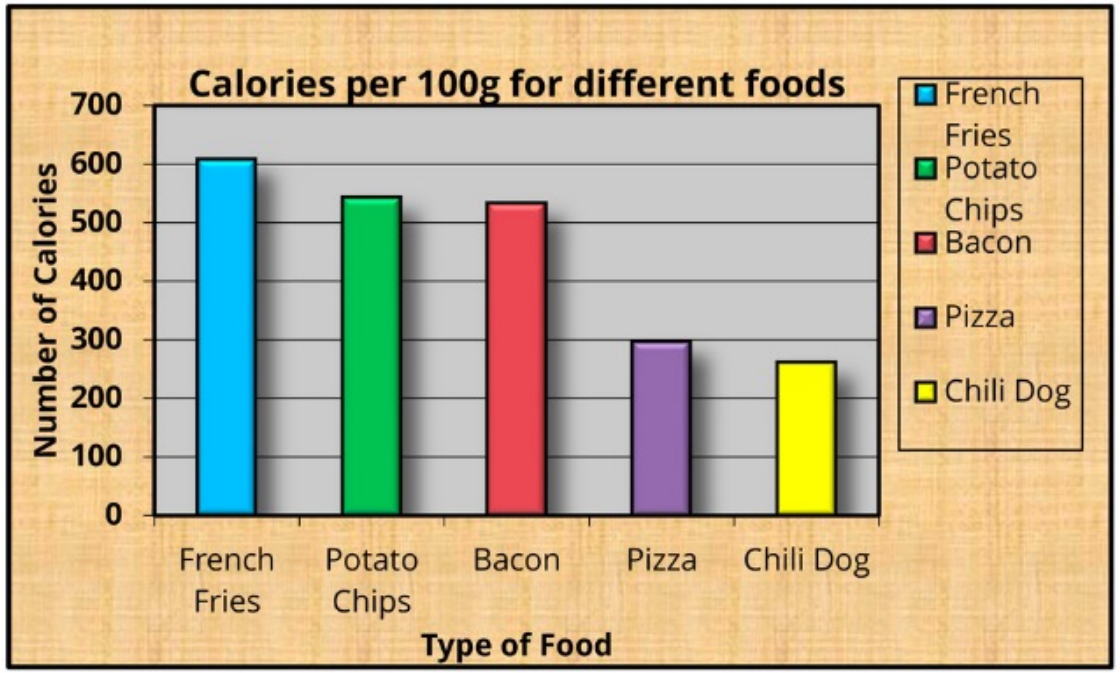
Best Practices

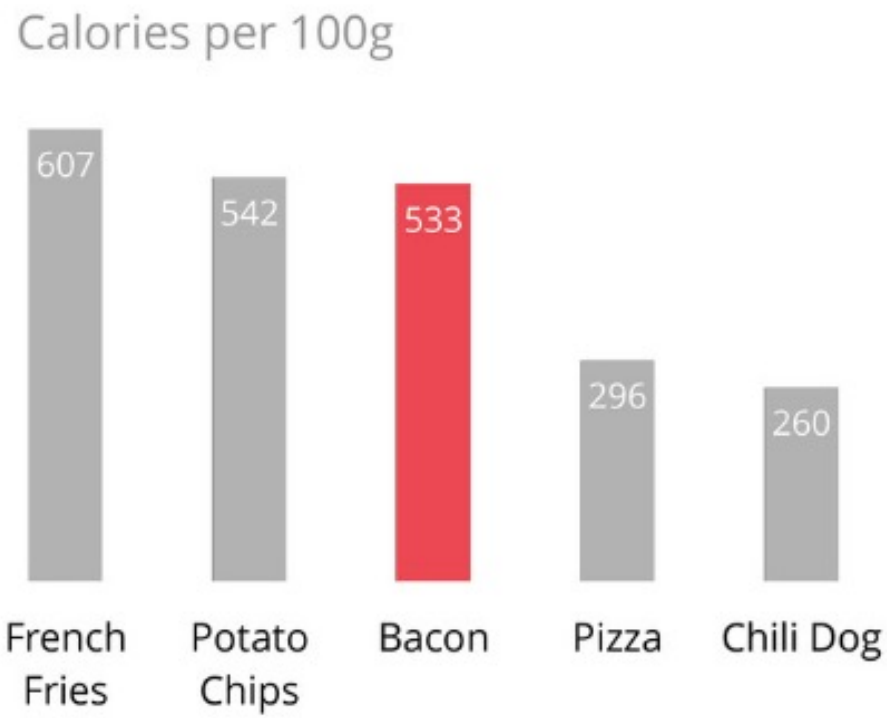
1. Label your axes
2. Follow axis conventions
   → Explanatory variable on x axis
   → Simple and intuitive y variable whenever possible
3. Include uncertainty (standard error, confidence intervals, or raw data itself)

Best Practices

1. Label your axes
2. Follow axis conventions
   → Explanatory variable on x axis
   → Simple and intuitive y variable whenever possible
3. Include uncertainty (standard error, confidence intervals, or raw data itself)
4. "Remove to improve"

Best Practices





Source: Darkhorse Analytics

Best Practices

1. Label your axes
2. Follow axis conventions
   → Explanatory variable on x axis
   → Simple and intuitive y variable whenever possible
3. Include uncertainty (standard error, confidence intervals, or raw data itself)
4. "Remove to improve"
5. Choose the right graph for the problem

How to know you’re doing it right

How to know you're doing it right

1. A viewer can easily recognize what's in your graph

How to know you're doing it right

1. A viewer can easily recognize what's in your graph
   → clear axis labels and legends, remove to improve

How to know you're doing it right

1. A viewer can easily recognize what's in your graph
   → clear axis labels and legends, remove to improve
2. A viewer can easily make the sorts of comparisons you want them to or identify the pattern you're trying to express

How to know you're doing it right

1. A viewer can easily recognize what's in your graph
   → clear axis labels and legends, remove to improve
2. A viewer can easily make the sorts of comparisons you want them to or identify the pattern you're trying to express
   → remove to improve, choose the right graph

How to know you're doing it right

1. A viewer can easily recognize what's in your graph
   → clear axis labels and legends, remove to improve
2. A viewer can easily make the sorts of comparisons you want them to or identify the pattern you're trying to express
   → remove to improve, choose the right graph
3. A viewer can easily figure out what the "point" of the graph is (i.e., what is the key result or finding; don't overdo it!)

How to know you're doing it right

1. A viewer can easily recognize what's in your graph
   → clear axis labels and legends, remove to improve
2. A viewer can easily make the sorts of comparisons you want them to or identify the pattern you're trying to express
   → remove to improve, choose the right graph
3. A viewer can easily figure out what the "point" of the graph is (i.e., what is the key result or finding; don't overdo it!)
   → Remove to improve, choose the right graph

How to know you're doing it right

1. A viewer can easily recognize what's in your graph
   → clear axis labels and legends, remove to improve
2. A viewer can easily make the sorts of comparisons you want them to or identify the pattern you're trying to express
   → remove to improve, choose the right graph
3. A viewer can easily figure out what the "point" of the graph is (i.e., what is the key result or finding; don't overdo it!)
   → Remove to improve, choose the right graph
4. A viewer can do all the above with the right level of precision or certainty

How to know you're doing it right

1. A viewer can easily recognize what's in your graph
   → clear axis labels and legends, remove to improve
2. A viewer can easily make the sorts of comparisons you want them to or identify the pattern you're trying to express
   → remove to improve, choose the right graph
3. A viewer can easily figure out what the "point" of the graph is (i.e., what is the key result or finding; don't overdo it!)
   → Remove to improve, choose the right graph
4. A viewer can do all the above with the right level of precision or certainty
   → Uncertainty or raw data

What We'll Cover

1. Ways graphs can go wrong (*there are more than you think…*)

2. How to avoid these pitfalls

3. Choosing the right visualization

What We'll Cover

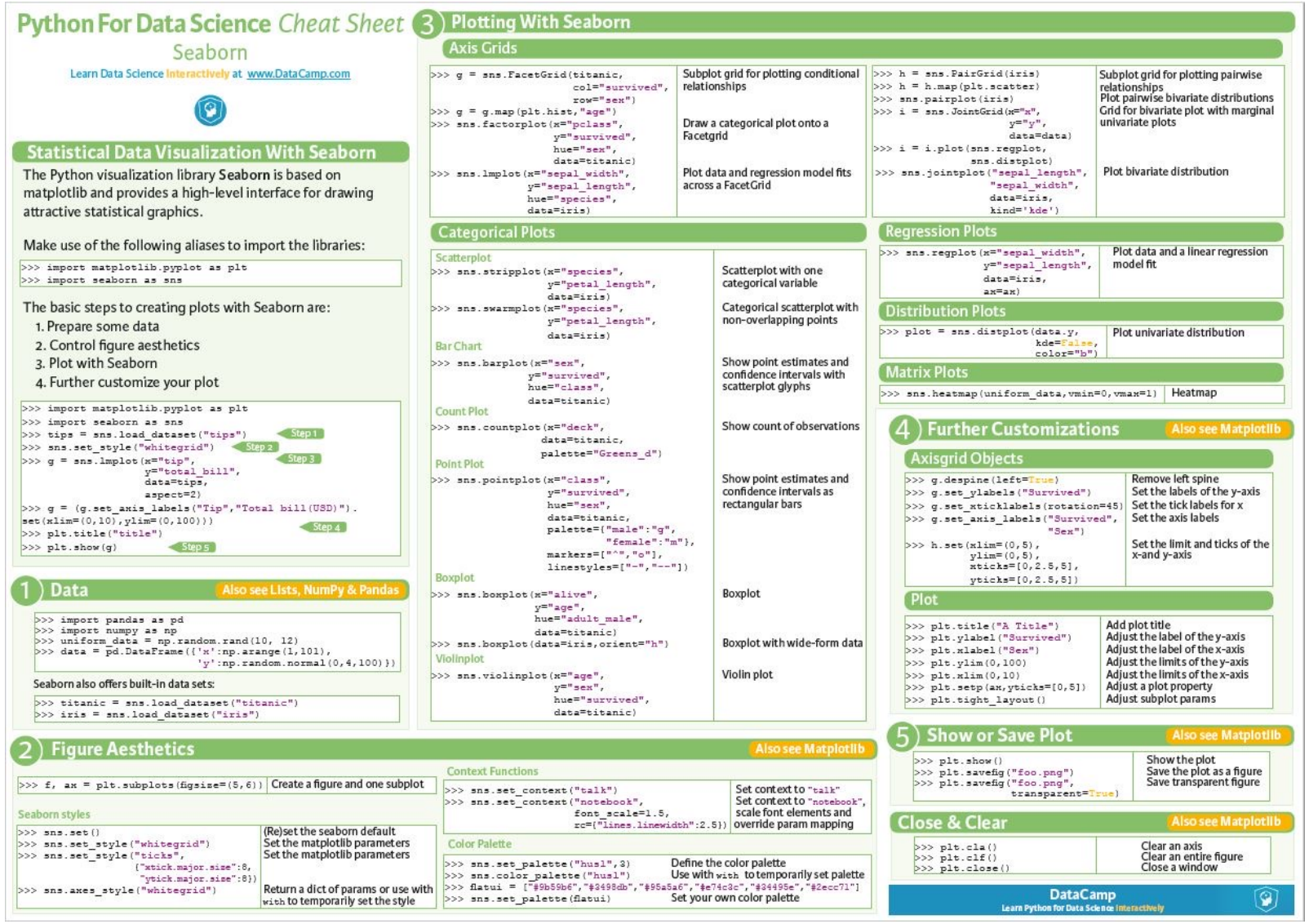
1. Ways graphs can go wrong (*there are more than you think…*)

2. How to avoid these pitfalls

3. Choosing the right visualization

Choosing the right graph for the job

Choosing the right graph for the job



Python For Data Science *Cheat Sheet*

Seaborn

Learn Data Science Interactively at www.DataCamp.com

Statistical Data Visualization With Seaborn

The Python visualization library **Seaborn** is based on matplotlib and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
```

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips")                    Step 1
>>> sns.set_style("whitegrid")                         Step 2
>>> g = sns.lmplot(x="tip",                            Step 3
                   y="total_bill",
                   data=tips,
                   aspect=2)
>>> g = (g.set_axis_labels("Tip","Total bill(USD)").
set(xlim=(0,10),ylim=(0,100)))
>>> plt.title("title")                                 Step 4
>>> plt.show(g)                                        Step 5
```

1 Data — Also see Lists, NumPy & Pandas

```
>>> import pandas as pd
>>> import numpy as np
>>> uniform_data = np.random.rand(10, 12)
>>> data = pd.DataFrame({'x':np.arange(1,101),
                         'y':np.random.normal(0,4,100)})
```

Seaborn also offers built-in data sets:

```
>>> titanic = sns.load_dataset("titanic")
>>> iris = sns.load_dataset("iris")
```

2 Figure Aesthetics — Also see Matplotlib

`>>> f, ax = plt.subplots(figsize=(5,6))`	Create a figure and one subplot

Seaborn styles

`>>> sns.set()`	(Re)set the seaborn default
`>>> sns.set_style("whitegrid")`	Set the matplotlib parameters
`>>> sns.set_style("ticks", {"xtick.major.size":8, "ytick.major.size":8})`	Set the matplotlib parameters
`>>> sns.axes_style("whitegrid")`	Return a dict of params or use with `with` to temporarily set the style

Context Functions

`>>> sns.set_context("talk")`	Set context to "talk"
`>>> sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth":2.5})`	Set context to "notebook", scale font elements and override param mapping

Color Palette

`>>> sns.set_palette("husl",3)`	Define the color palette
`>>> sns.color_palette("husl")`	Use with `with` to temporarily set palette
`>>> flatui = ["#9b59b6","#3498db","#95a5a6","#e74c3c","#34495e","#2ecc71"]`	
`>>> sns.set_palette(flatui)`	Set your own color palette

3 Plotting With Seaborn

Axis Grids

`>>> g = sns.FacetGrid(titanic, col="survived", row="sex")`	Subplot grid for plotting conditional relationships
`>>> g = g.map(plt.hist,"age")`	
`>>> sns.factorplot(x="pclass", y="survived", hue="sex", data=titanic)`	Draw a categorical plot onto a Facetgrid
`>>> sns.lmplot(x="sepal_width", y="sepal_length", hue="species", data=iris)`	Plot data and regression model fits across a FacetGrid
`>>> h = sns.PairGrid(iris)`	Subplot grid for plotting pairwise relationships
`>>> h = h.map(plt.scatter)`	Plot pairwise bivariate distributions
`>>> sns.pairplot(iris)`	Grid for bivariate plot with marginal univariate plots
`>>> i = sns.JointGrid(x="x", y="y", data=data)`	
`>>> i = i.plot(sns.regplot, sns.distplot)`	
`>>> sns.jointplot("sepal_length", "sepal_width", data=iris, kind='kde')`	Plot bivariate distribution

Categorical Plots

Scatterplot	
`>>> sns.stripplot(x="species", y="petal_length", data=iris)`	Scatterplot with one categorical variable
`>>> sns.swarmplot(x="species", y="petal_length", data=iris)`	Categorical scatterplot with non-overlapping points
Bar Chart	
`>>> sns.barplot(x="sex", y="survived", hue="class", data=titanic)`	Show point estimates and confidence intervals with scatterplot glyphs
Count Plot	
`>>> sns.countplot(x="deck", data=titanic, palette="Greens_d")`	Show count of observations
Point Plot	
`>>> sns.pointplot(x="class", y="survived", hue="sex", data=titanic, palette={"male":"g", "female":"m"}, markers=["^","o"], linestyles=["-","--"])`	Show point estimates and confidence intervals as rectangular bars
Boxplot	
`>>> sns.boxplot(x="alive", y="age", hue="adult_male", data=titanic)`	Boxplot
`>>> sns.boxplot(data=iris,orient="h")`	Boxplot with wide-form data
Violinplot	
`>>> sns.violinplot(x="age", y="sex", hue="survived", data=titanic)`	Violin plot

Regression Plots

`>>> sns.regplot(x="sepal_width", y="sepal_length", data=iris, ax=ax)`	Plot data and a linear regression model fit

Distribution Plots

`>>> plot = sns.distplot(data.y, kde=False, color="b")`	Plot univariate distribution

Matrix Plots

`>>> sns.heatmap(uniform_data,vmin=0,vmax=1)`	Heatmap

4 Further Customizations — Also see Matplotlib

Axisgrid Objects

`>>> g.despine(left=True)`	Remove left spine
`>>> g.set_ylabels("Survived")`	Set the labels of the y-axis
`>>> g.set_xticklabels(rotation=45)`	Set the tick labels for x
`>>> g.set_axis_labels("Survived", "Sex")`	Set the axis labels
`>>> h.set(xlim=(0,5), ylim=(0,5), xticks=[0,2.5,5], yticks=[0,2.5,5])`	Set the limit and ticks of the x-and y-axis

Plot

`>>> plt.title("A Title")`	Add plot title
`>>> plt.ylabel("Survived")`	Adjust the label of the y-axis
`>>> plt.xlabel("Sex")`	Adjust the label of the x-axis
`>>> plt.ylim(0,100)`	Adjust the limits of the y-axis
`>>> plt.xlim(0,10)`	Adjust the limits of the x-axis
`>>> plt.setp(ax,yticks=[0,5])`	Adjust a plot property
`>>> plt.tight_layout()`	Adjust subplot params

5 Show or Save Plot — Also see Matplotlib

`>>> plt.show()`	Show the plot
`>>> plt.savefig("foo.png")`	Save the plot as a figure
`>>> plt.savefig("foo.png", transparent=True)`	Save transparent figure

Close & Clear — Also see Matplotlib

`>>> plt.cla()`	Clear an axis
`>>> plt.clf()`	Clear an entire figure
`>>> plt.close()`	Close a window

DataCamp
Learn Python for Data Science Interactively

Source: https://www.kaggle.com/getting-started/126958

Choosing the right graph for the job

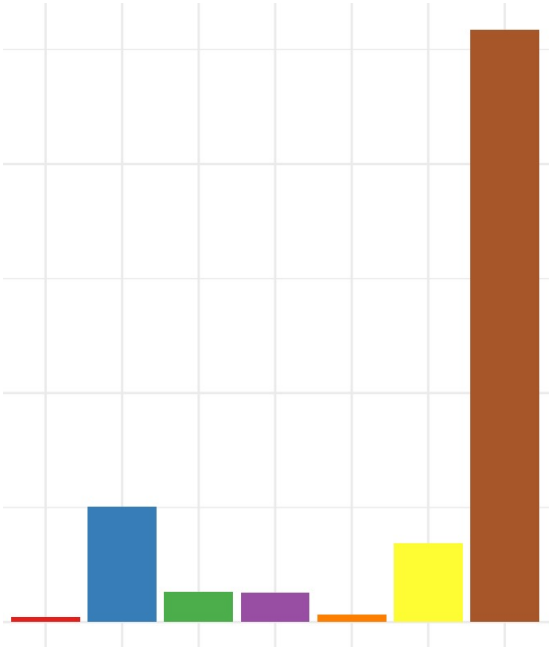
What kind of variables are we working with?

- Categorical / discrete
- Numerical / continuous

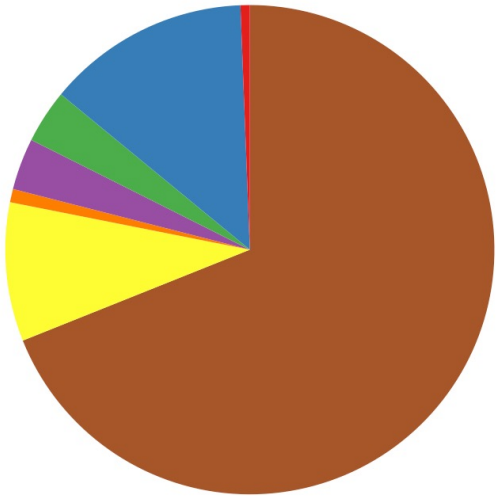
Which variable(s) are explanatory and which are being explained (response variable)?

Choosing the right graph for the job

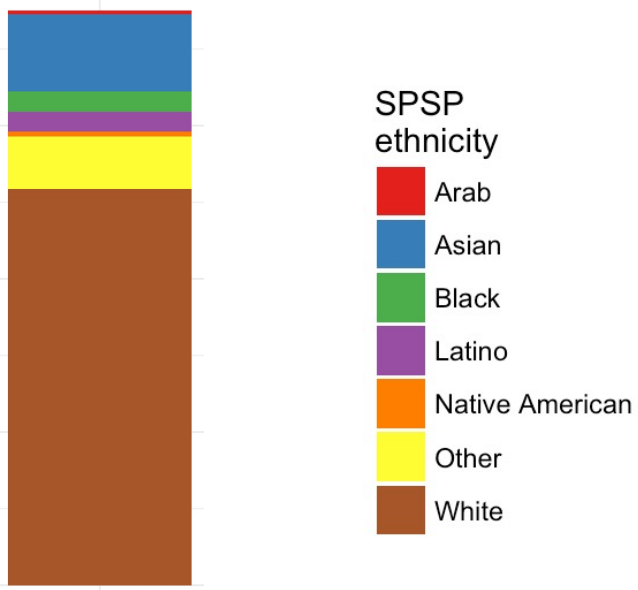
Categorical response variable, no explanatory variable



Histogram barplot
(counts or proportions)



Pie chart

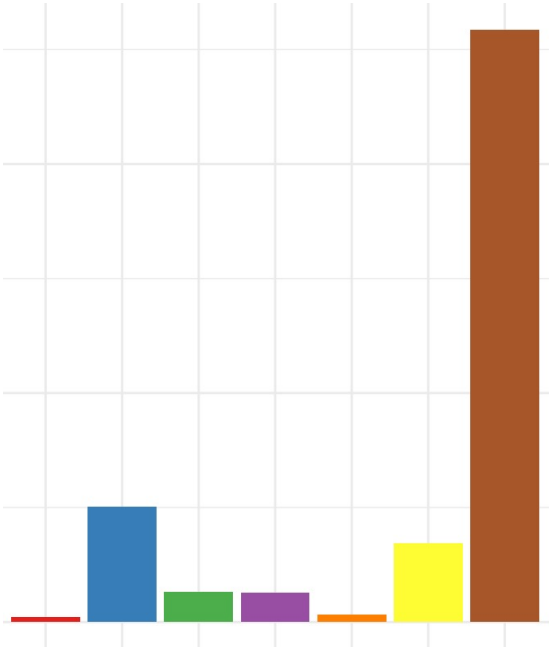


Stacked barplot

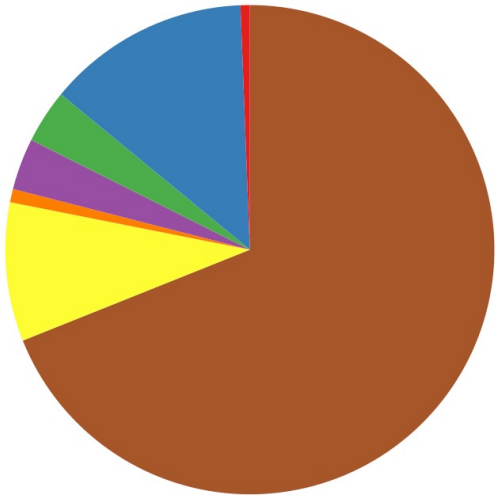
Data: http://vulstats.ucsd.edu/data/spsp.demographics.cleaned.csv

Choosing the right graph for the job

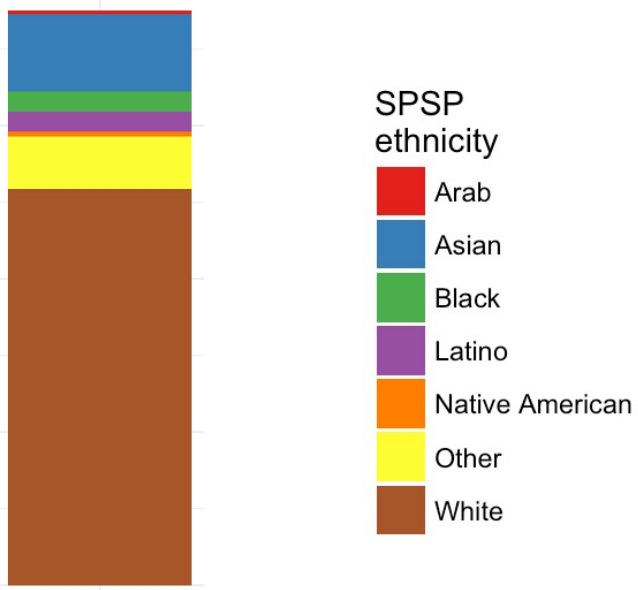
Categorical response variable, no explanatory variable



Histogram barplot
(counts or proportions)



Pie chart

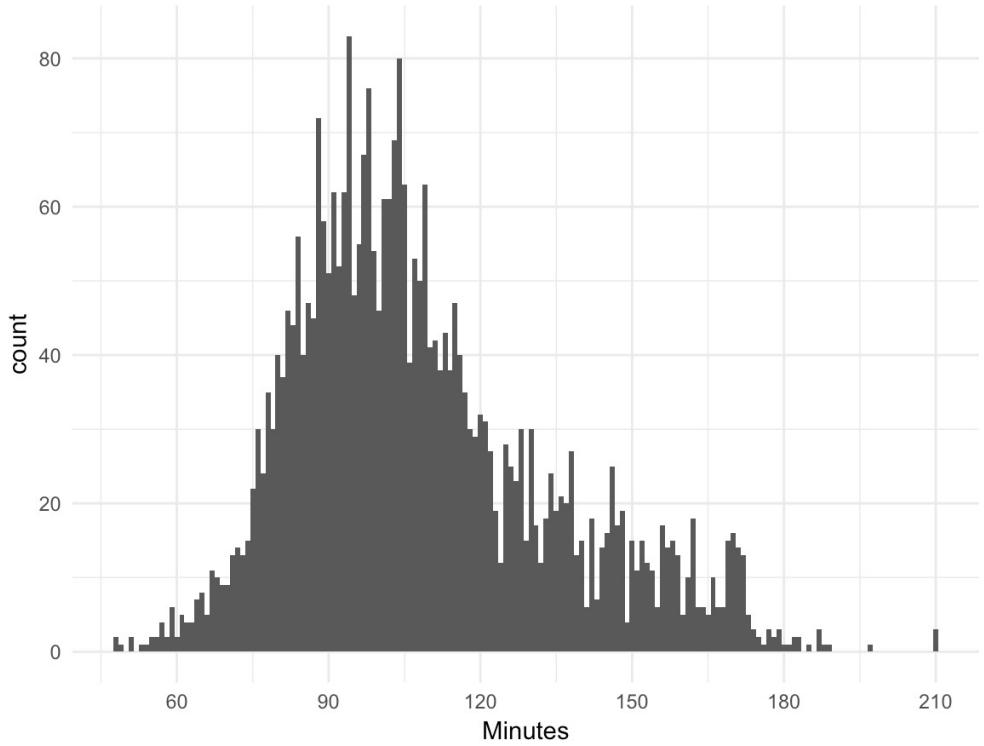


Stacked barplot

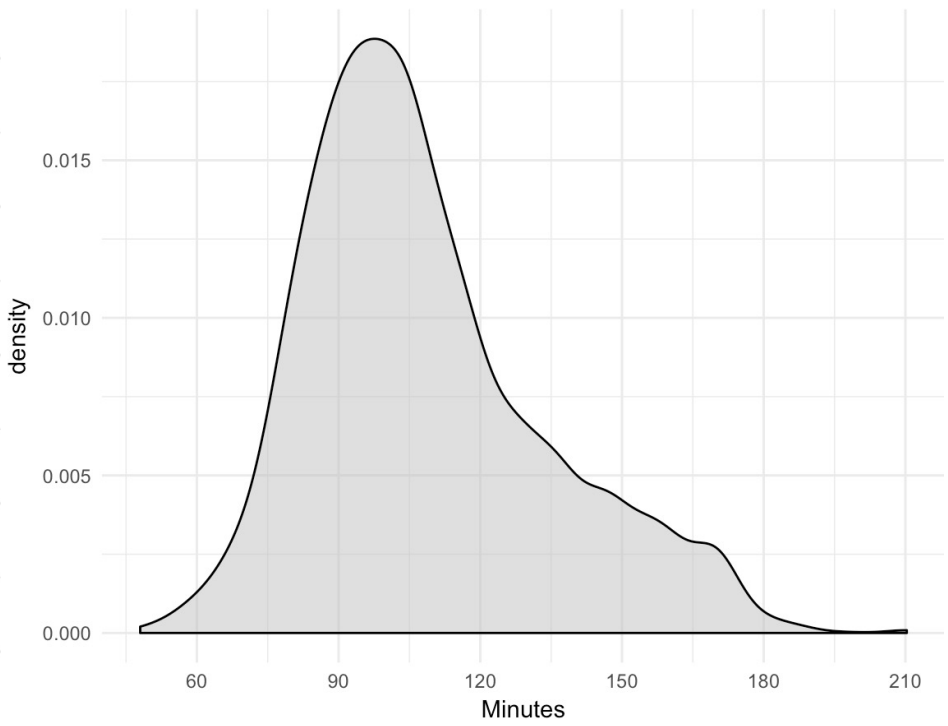
Data: http://vulstats.ucsd.edu/data/spsp.demographics.cleaned.csv

Choosing the right graph for the job

Numerical response variable, no explanatory variable



Histogram!

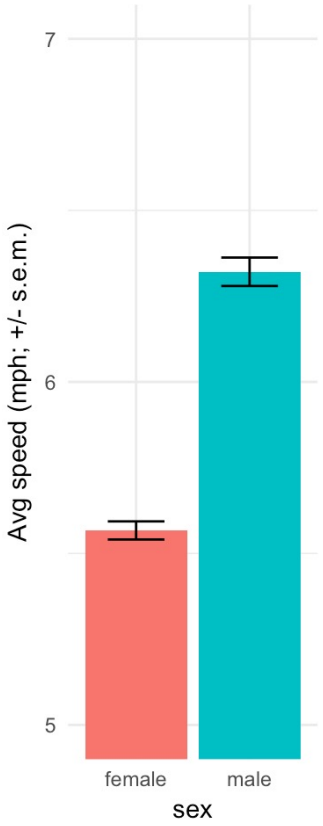


Smoothed density

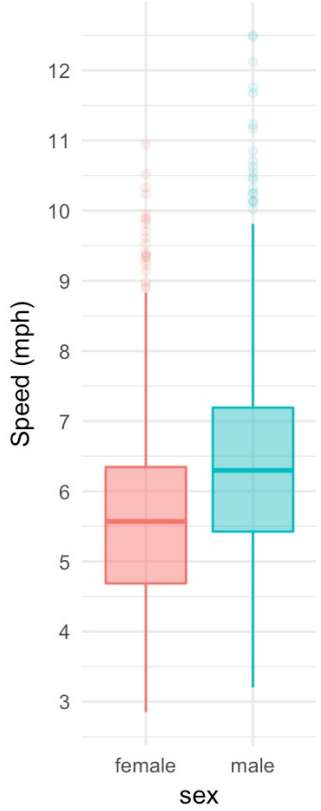
Data: http://vulstats.ucsd.edu/data/cal1020.cleaned.Rdata

Choosing the right graph for the job

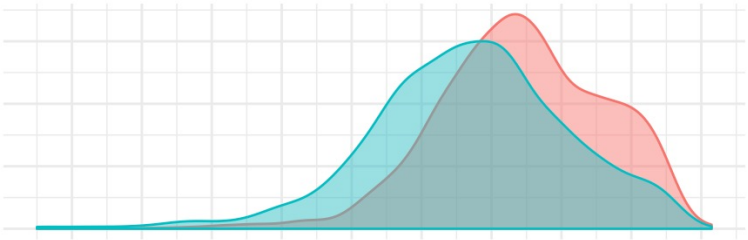
Numerical response variable, **categorical** explanatory variable



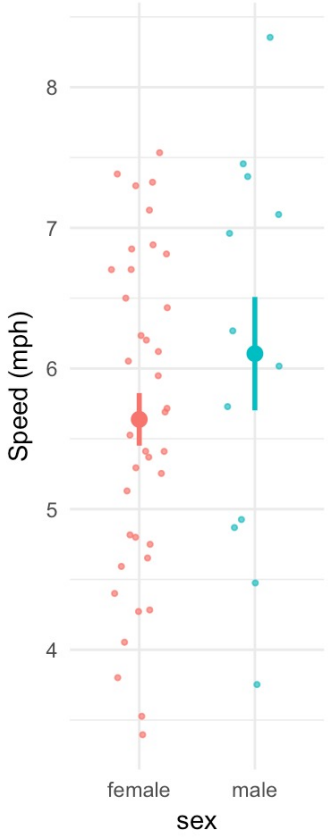
Mean + error



Boxplot



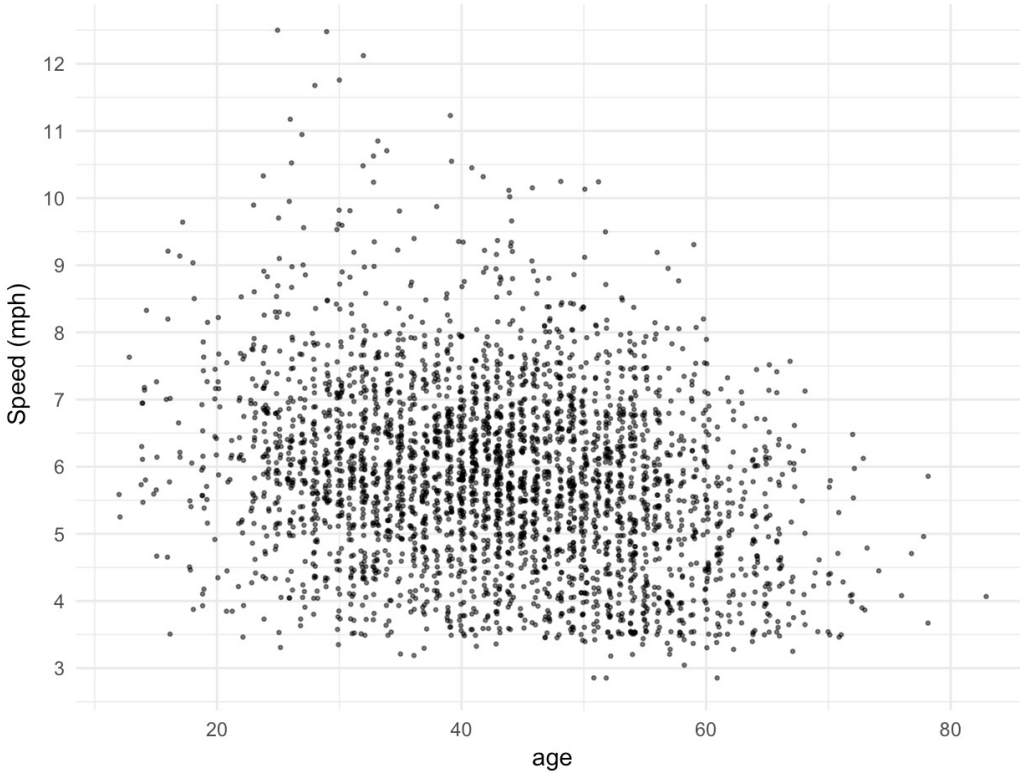
Densities (or histograms)



Data: http://vulstats.ucsd.edu/data/cal1020.cleaned.Rdata

Choosing the right graph for the job

Numerical response variable, **numerical** explanatory variable

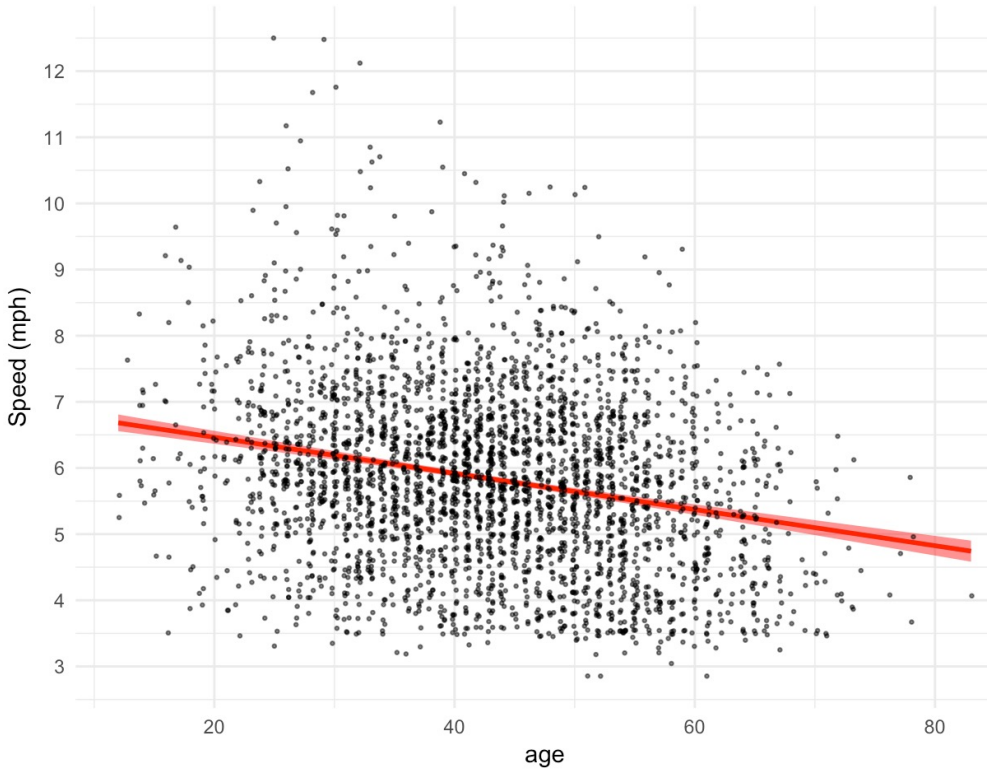


Scatterplot

Data: http://vulstats.ucsd.edu/data/cal1020.cleaned.Rdata

Choosing the right graph for the job

Numerical response variable, **numerical** explanatory variable

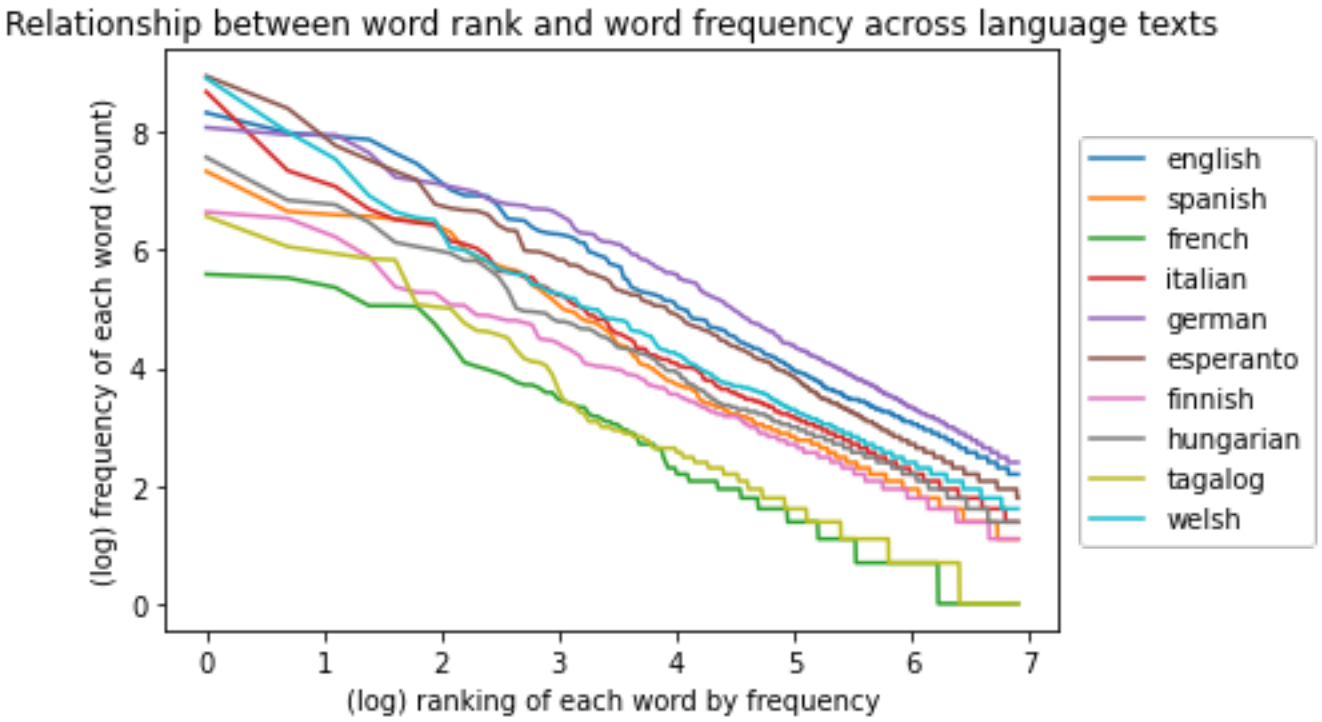


Include line of best fit!

Data: http://vulstats.ucsd.edu/data/cal1020.cleaned.Rdata

Choosing the right graph for the job

Numerical response variable, **numerical** *and* **categorical** explanatory variables



More options here, but this is usually a good start!

Choosing the right graph for the job

- This obviously isn't **everything** when it comes to visualization options

- But this should get you on the right path most of the time!

What We'll Cover

1. Ways graphs can go wrong (*there are more than you think…*)

2. How to avoid these pitfalls

3. Choosing the right visualization