Computer Science

CSC580: Principles of Machine Learning

Data Analysis and Visualization

Prof. Jason Pacheco

Material from: Watkins, J. “Intro. to the Science of Statistics”

Final Project Report

- Posted on Gradescope

- Due Friday 12 / 8 @ 11:59pm

- 5-page limit

- Recommend (don’t require) NeurIPS Latex style
  - More useful for CSC580 students

Final Project Report

Introduction : Give background and motivation for your project. Why is it interesting?

Background : Provide any preliminary information needed to understand the project. This section can be more technically detailed than the introduction.

Methodology : Discuss how you approach the problem. Provide all the necessary technical details.

Results : Provide all of your results from the project. Negative results (i.e. things that didn't work) are also welcome and encouraged.

CSC580 Students should include Related Work section

Final Project Report

- Create a Github repository with all your code

- Include the Github link in your report (or share with me username: ‘pachecoj’ if you want to keep private)

Outline

- Data Visualization
- Data Summarization
- Python data tools

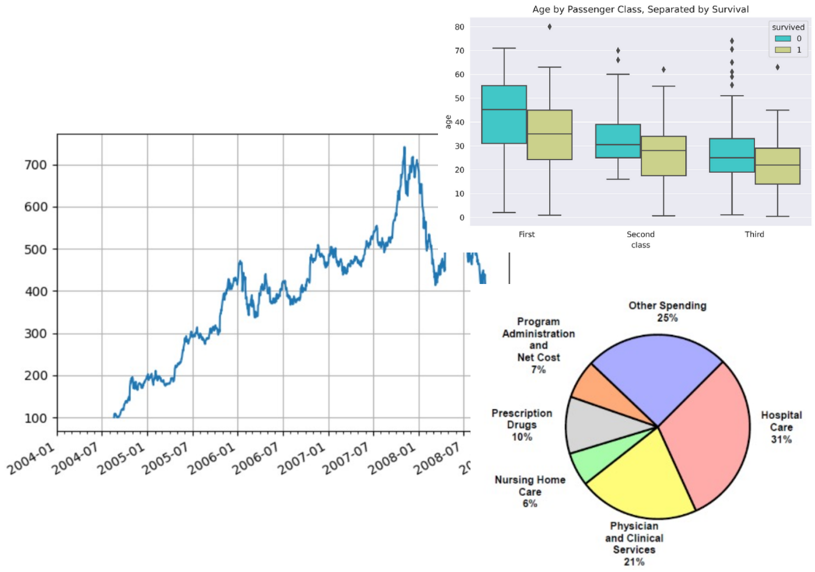
Outline

- Data Visualization
- Data Summarization
- Python data tools

Data Analysis, Exploration, and Visualization

141	137	134	134	132	130	129	129	131	135	130	128	129	126	128	128	130
138	136	134	134	135	133	131	129	132	139	133	128	130	128	127	129	131
135	135	134	133	133	132	130	128	132	136	134	130	131	131	132	132	133
133	134	133	132	131	130	130	131	131	129	134	134	130	134	137	134	134
134	134	134	134	133	132	134	138	136	127	135	137	132	136	140	135	139
137	135	136	138	137	135	137	143	142	132	136	138	135	137	138	138	142
139	135	135	138	138	134	135	141	143	133	133	134	135	135	133	138	140
136	137	137	138	141	143	142	144	140	143	142	137	137	139	137	135	136
137	138	136	136	138	140	141	143	140	144	143	139	139	140	138	137	139
137	139	137	136	136	136	137	140	143	146	143	140	141	142	142	143	143
137	140	141	139	138	136	135	137	143	144	142	139	142	144	145	147	146
140	144	144	143	141	137	135	137	139	139	139	139	143	145	146	147	147
145	148	147	145	143	140	139	141	136	138	140	142	147	147	146	147	149
146	148	147	144	143	141	140	143	137	139	142	145	146	145	145	148	147
145	147	146	143	142	140	140	143	138	140	143	143	143	141	143	148	142
145	145	144	144	143	141	141	142	142	145	146	145	144	141	143	150	144
144	143	142	143	143	142	142	144	143	144	143	144	148	144	142	147	145
146	145	144	143	143	143	144	146	144	144	141	146	157	154	144	143	148
149	148	145	144	143	143	144	145	144	146	142	149	167	169	155	146	151
150	149	147	145	142	142	143	143	145	147	143	147	166	175	164	151	152
150	150	149	147	145	145	145	145	147	148	143	142	154	165	160	148	150
152	152	152	150	149	150	150	149	151	151	150	147	146	152	153	147	151
152	153	153	152	151	151	151	150	152	152	156	155	148	149	155	153	152
152	152	152	152	152	151	151	151	152	152	152	153	152	151	152	153	154
152	152	152	152	152	152	151	151	152	152	152	153	152	151	151	152	154
153	153	153	153	153	153	153	153	154	154	153	153	152	152	150	152	154
153	153	153	153	154	154	154	154	154	154	153	153	153	153	152	153	155
153	153	152	153	154	154	154	154	153	154	154	153	153	153	153	154	157
153	152	152	152	154	155	155	155	153	155	155	154	152	152	152	154	159

Encoding



Iterate

Visual Perception

Understanding

There is a lot of advice about data visualization.

Avoid pie charts

Get it right in black and white

Data-Ink ratio should be high

Avoid word clouds

Overview first, zoom and filter, details on demand

Never truncate the y-axis

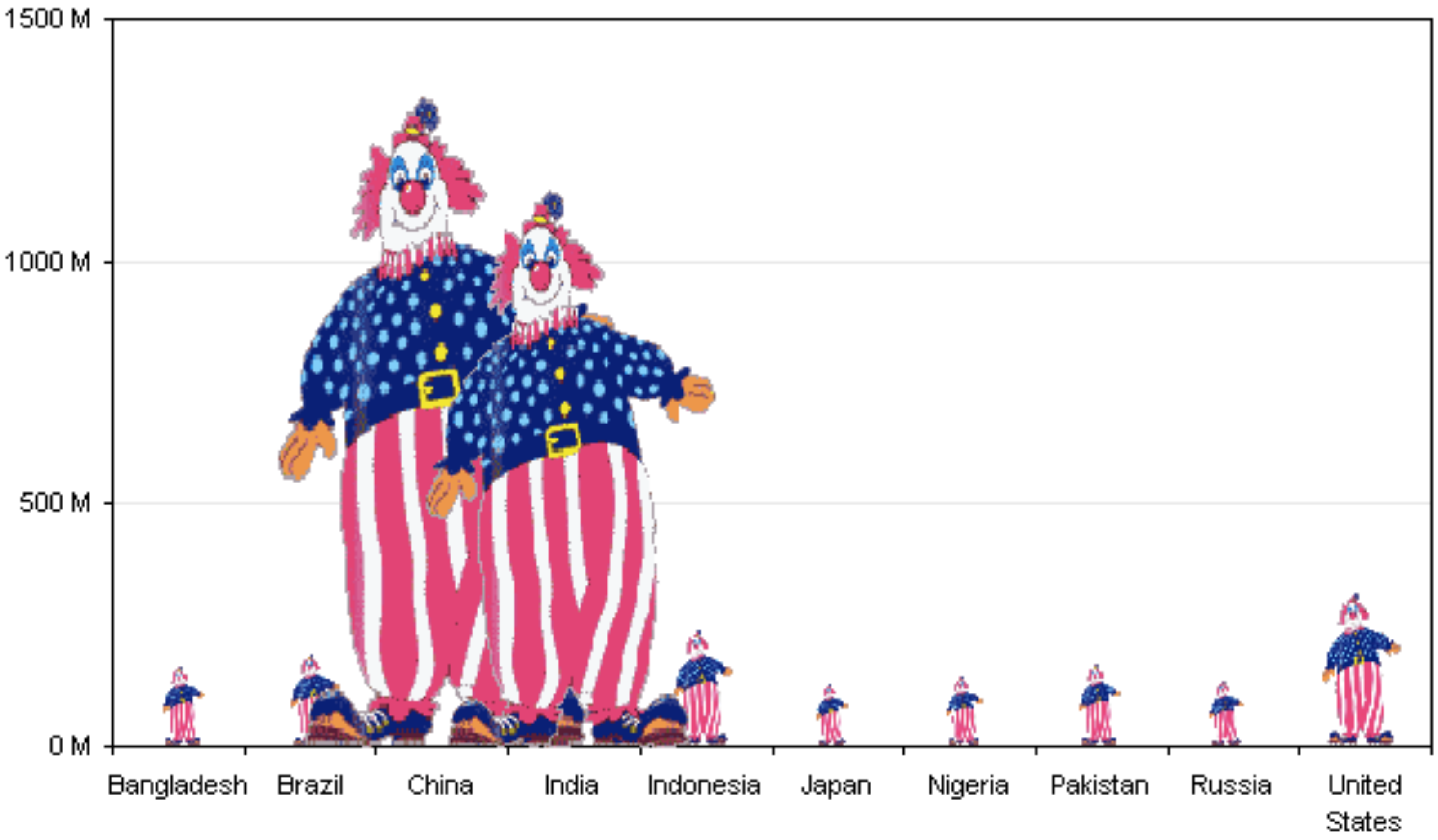
These can be considered design principles. Like all design principles, they shouldn't be adhered to dogmatically.

Be Careful with Area-as-Quantity

How many streams are there in November compared to December?

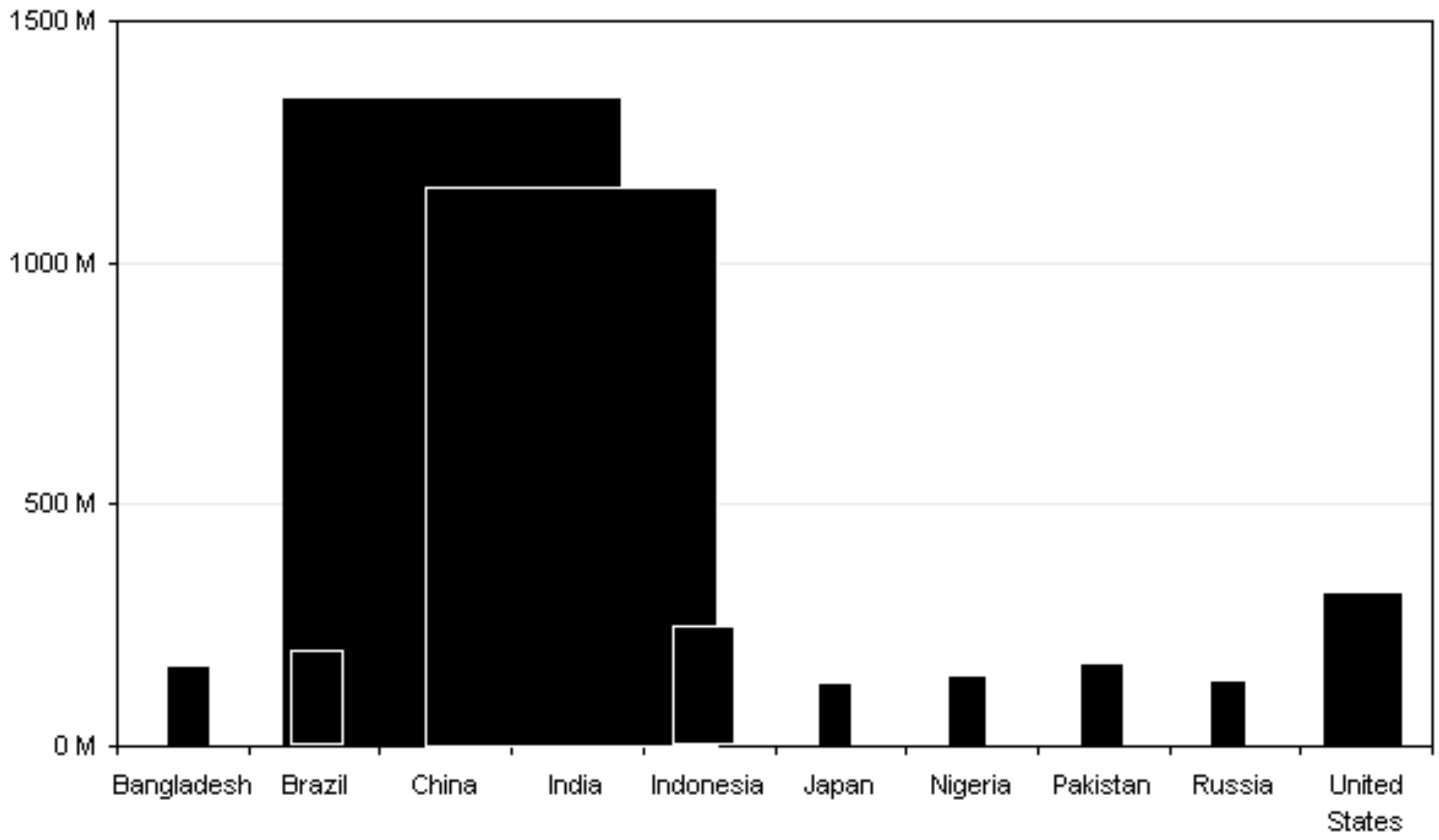
7.5 times as many streams!

Be careful of length vs. area for other marks



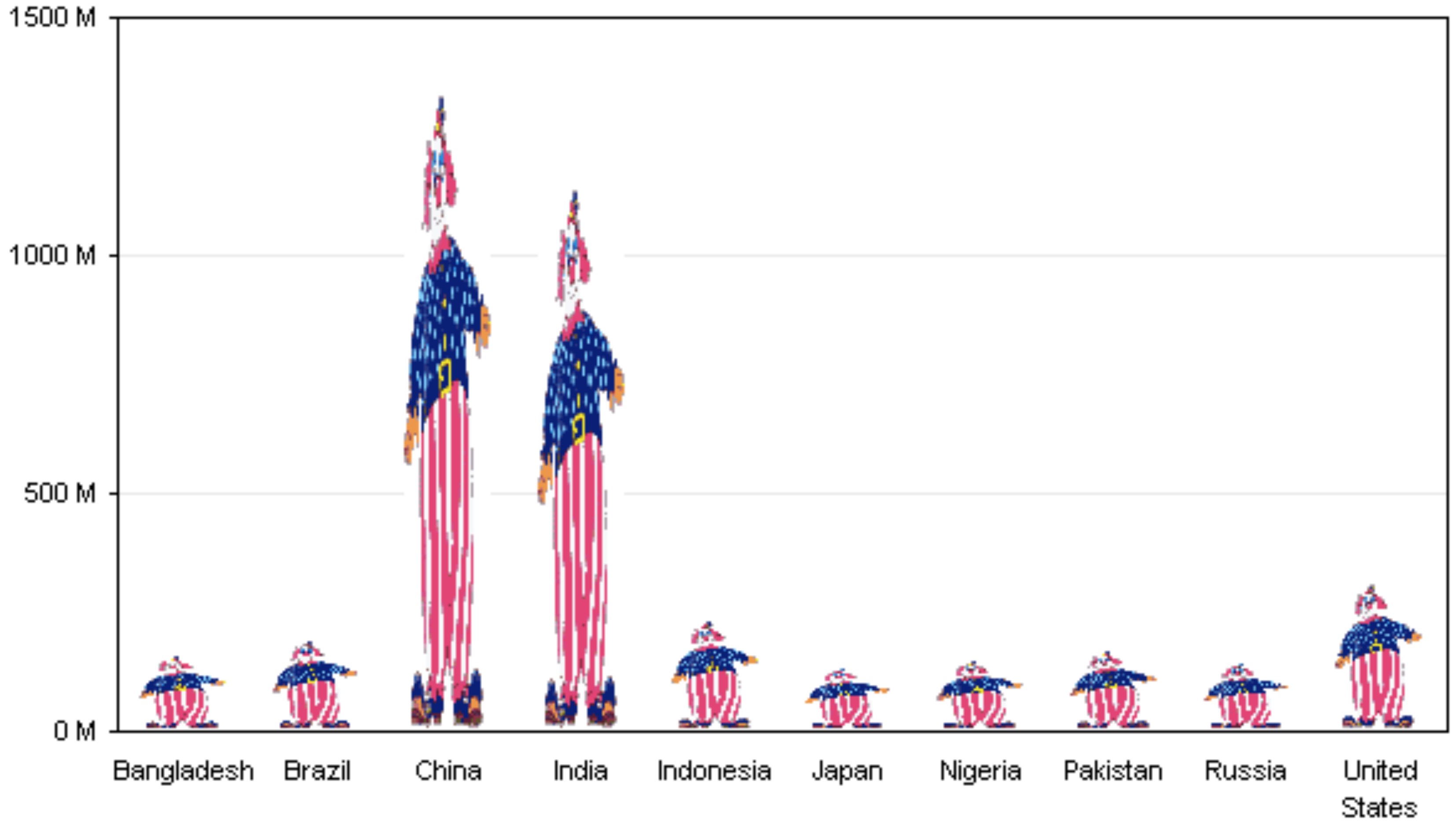
https://peltiertech.com/bad-bar-chart-practices-or-send-in-the-clowns/

What is being perceived?



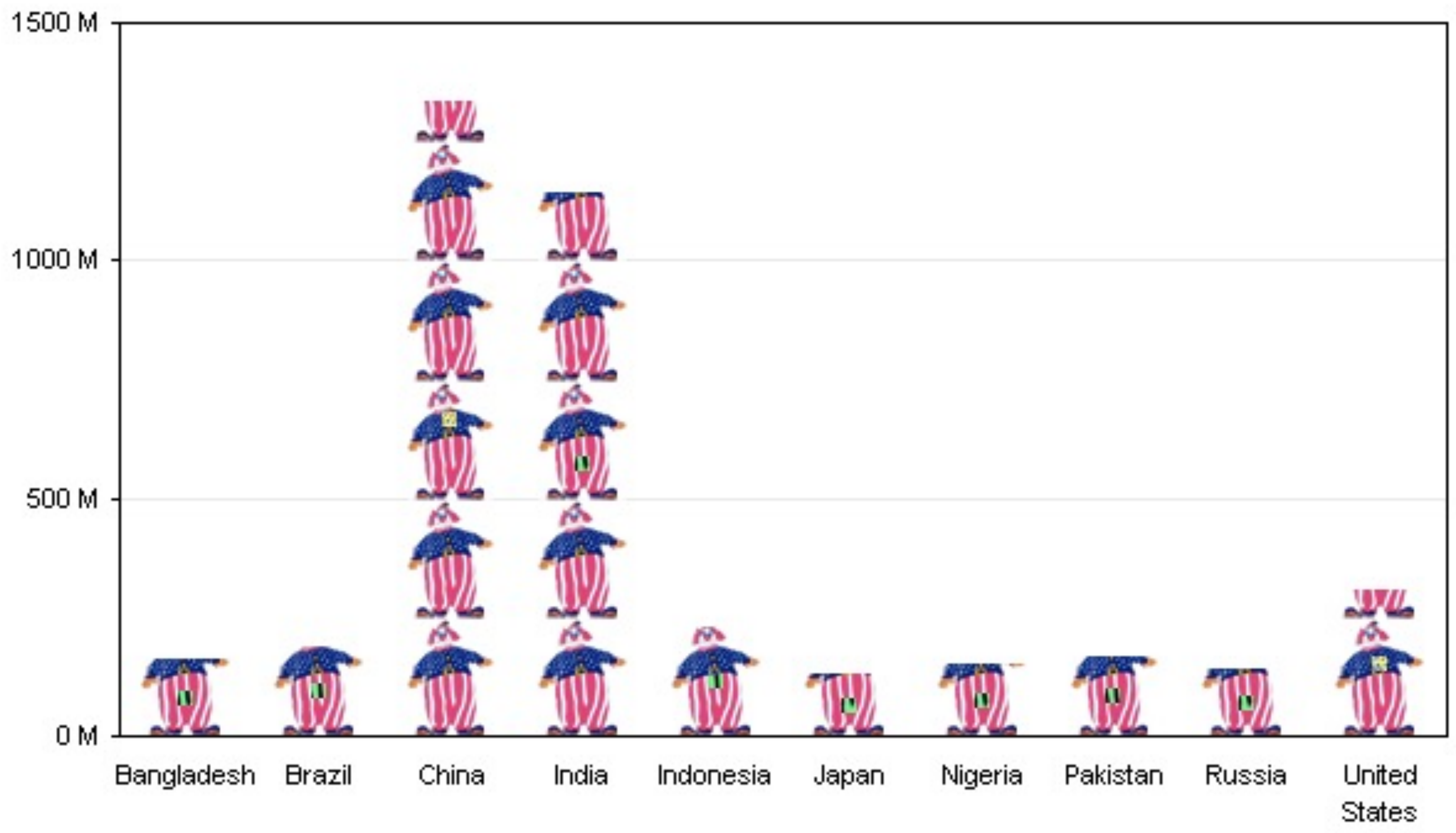
https://peltiertech.com/bad-bar-chart-practices-or-send-in-the-clowns/

Be careful of length vs. area for other marks



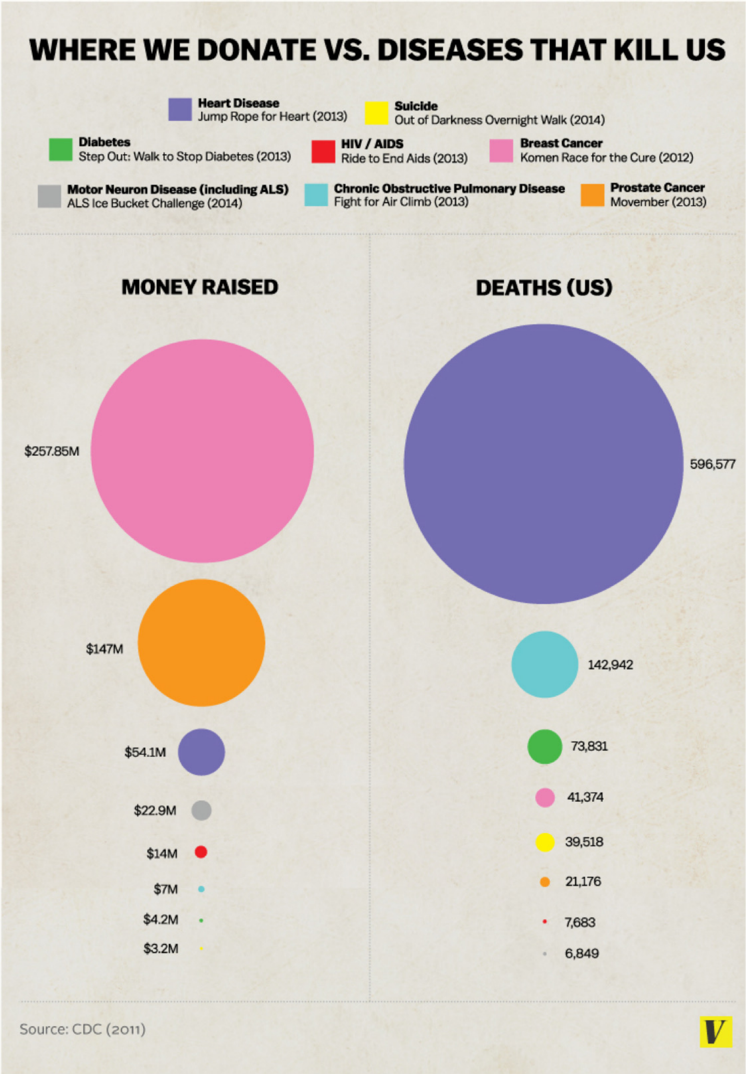
https://peltiertech.com/bad-bar-chart-practices-or-send-in-the-clowns/

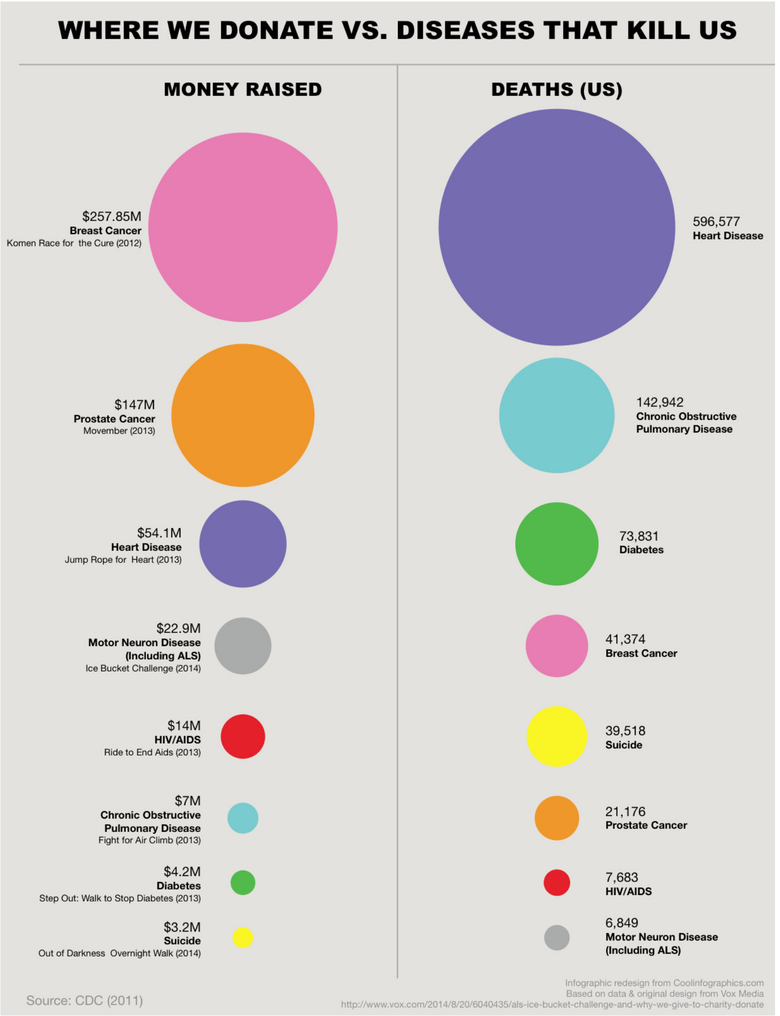
Consider using an Isotope Chart

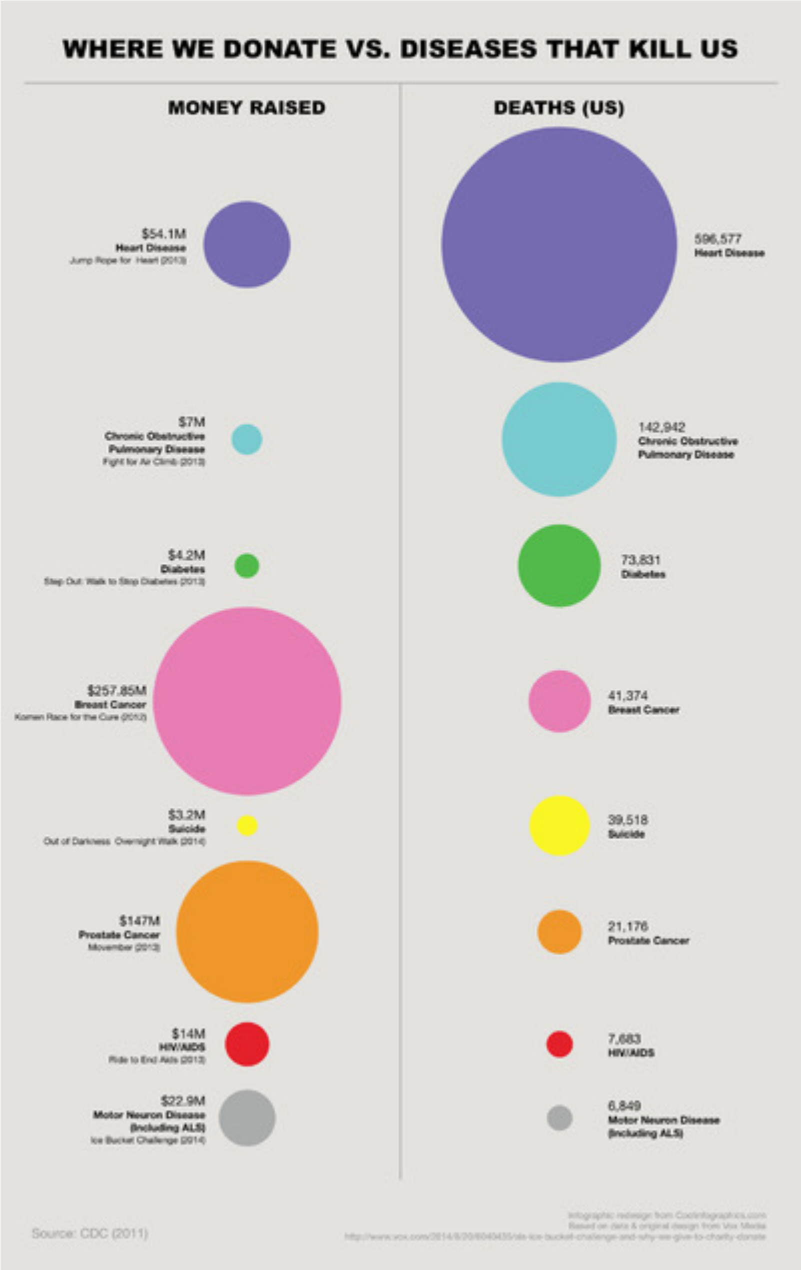


https://peltiertech.com/bad-bar-chart-practices-or-send-in-the-clowns/

Circles: Encode by Area not Radius







Images from Vox and
http://coolinfographics.com/blog/2014/8/29/false-visualizations-sizing-circles-in-infographics.html

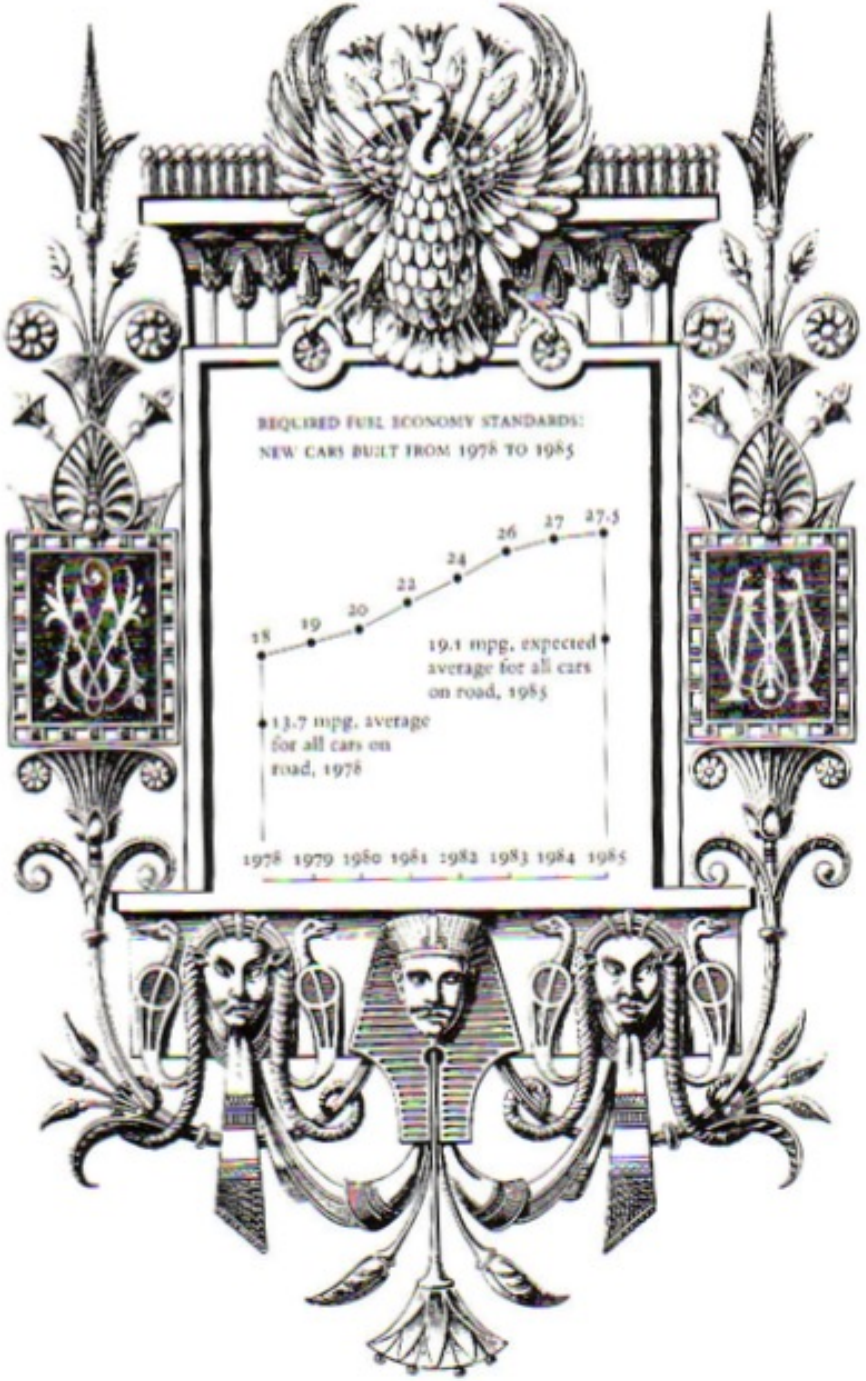
”Chart Junk”

Aside: Chart Junk

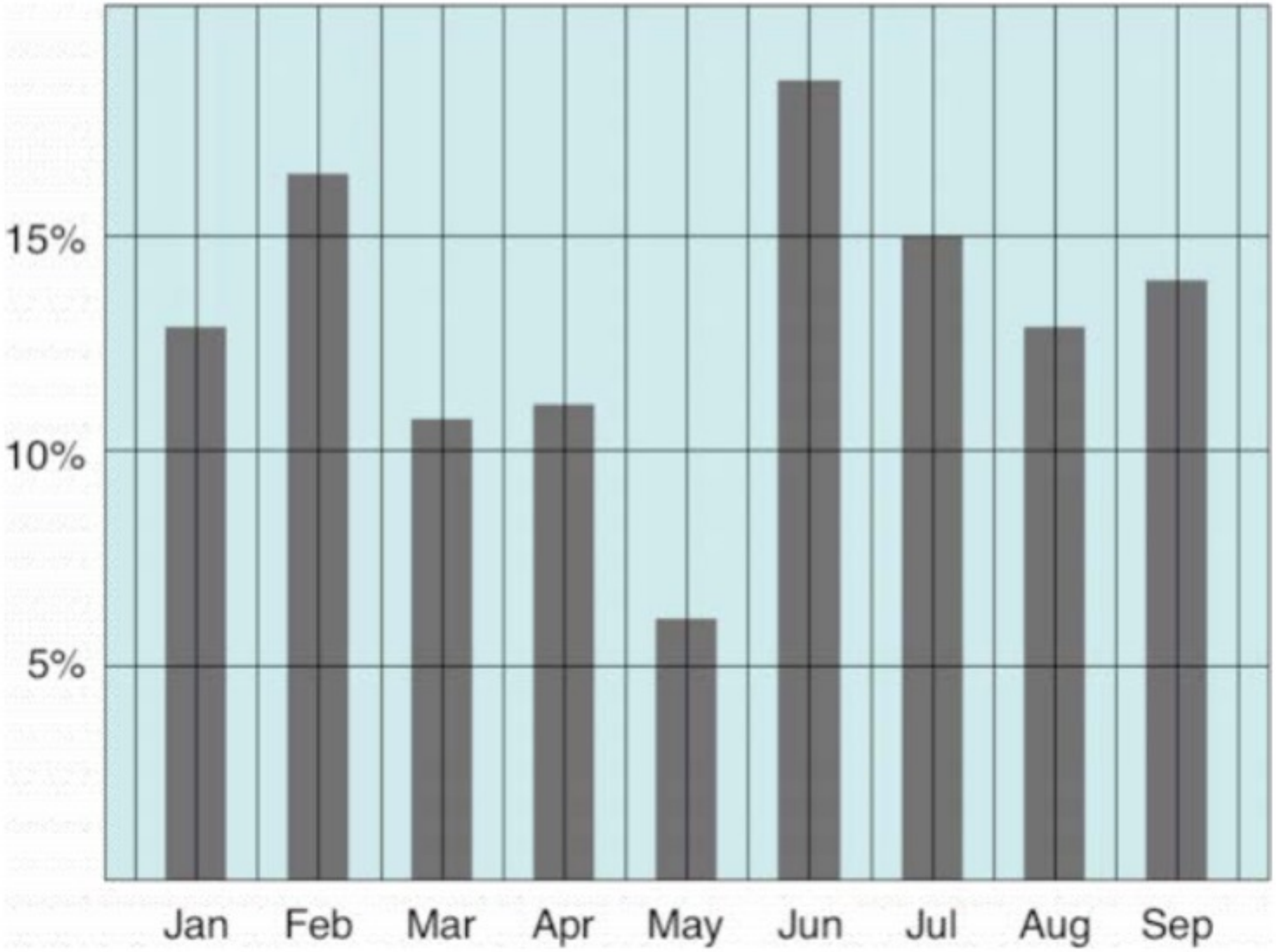
Chart Junk are embellishments to a chart that don't help communicate the data.

Removing such embellishments, whether completely decorative or embedded in the chart can help readability while conserving space.

However, sometimes embellishments can help with engagement and memorability, e.g., the clowns earlier

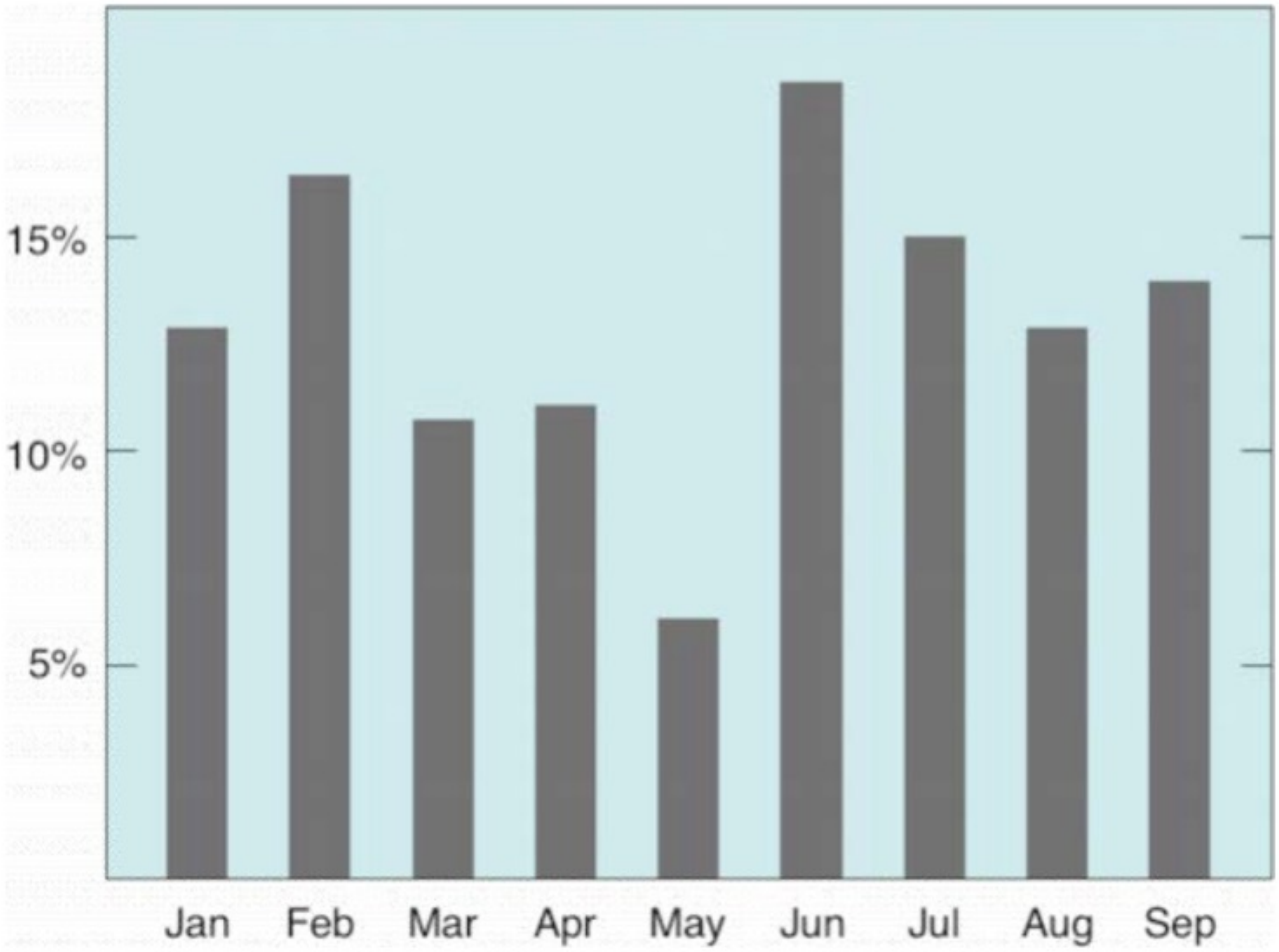


The added lines here help the chart look "charty" but don't help readability



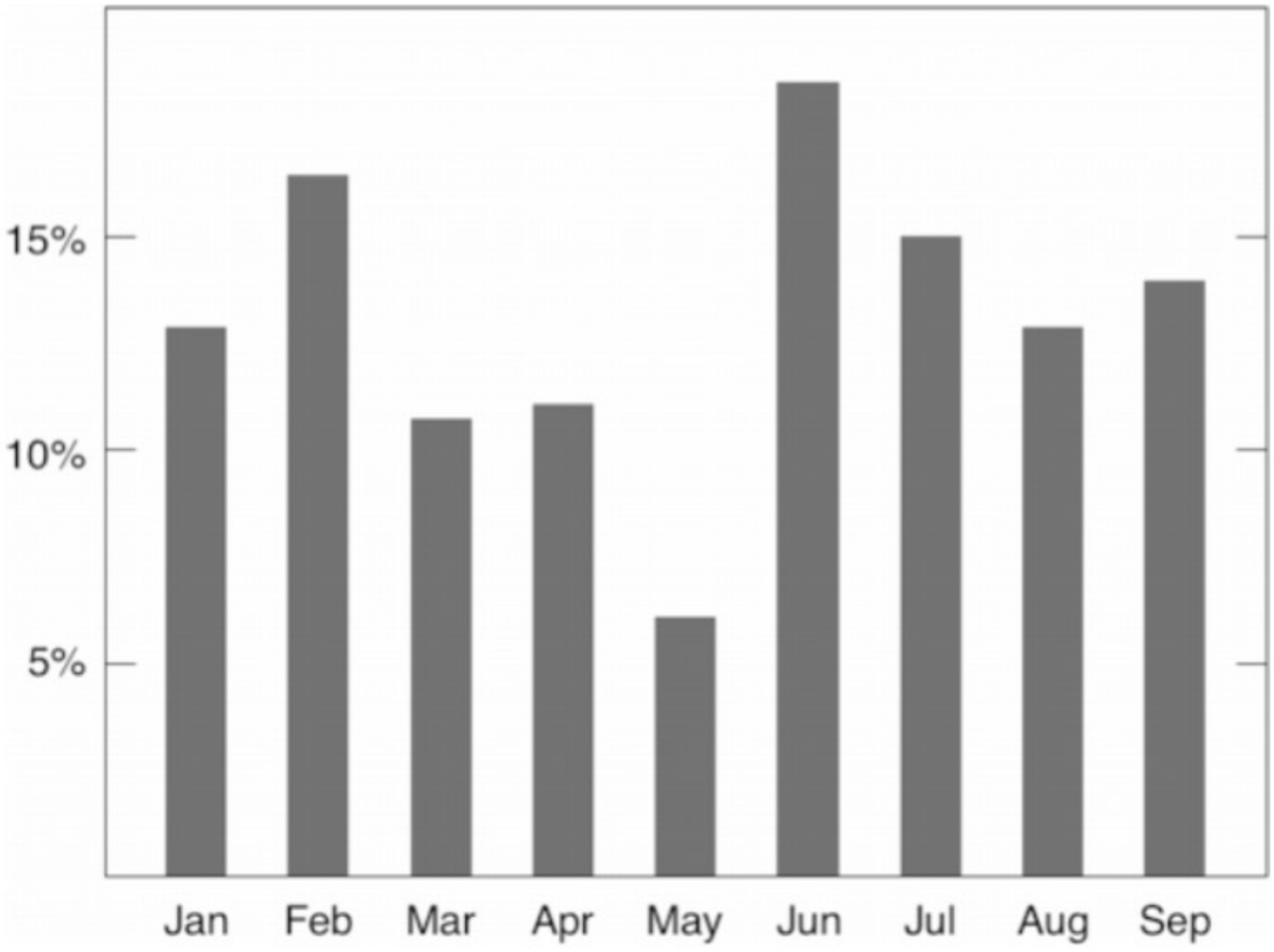
Images from http://www.tbray.org/ongoing/data-ink/di1

Remove or redesign grid lines



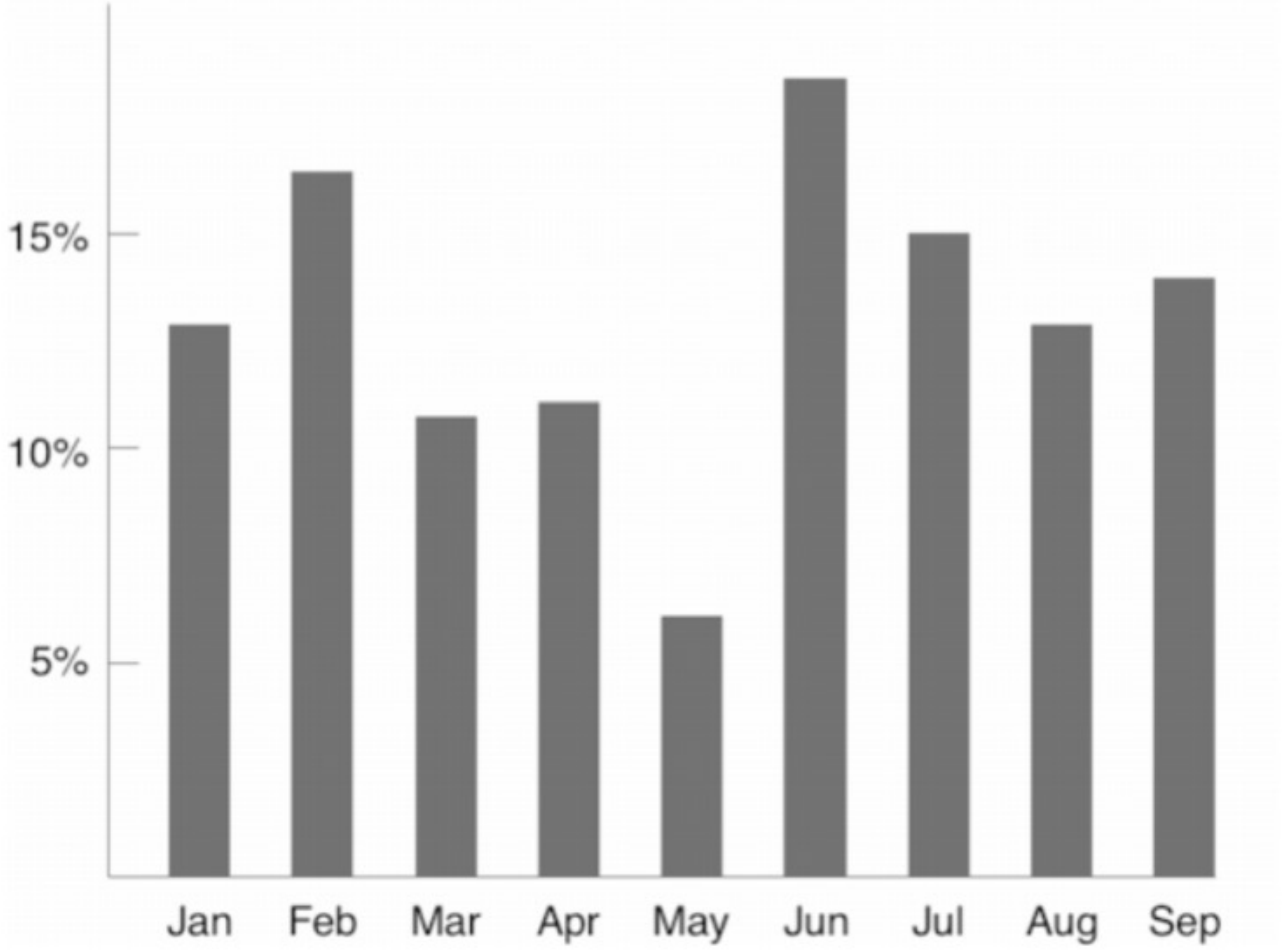
Images from http://www.tbray.org/ongoing/data-ink/di1

Remove color background, it did not help with contrast



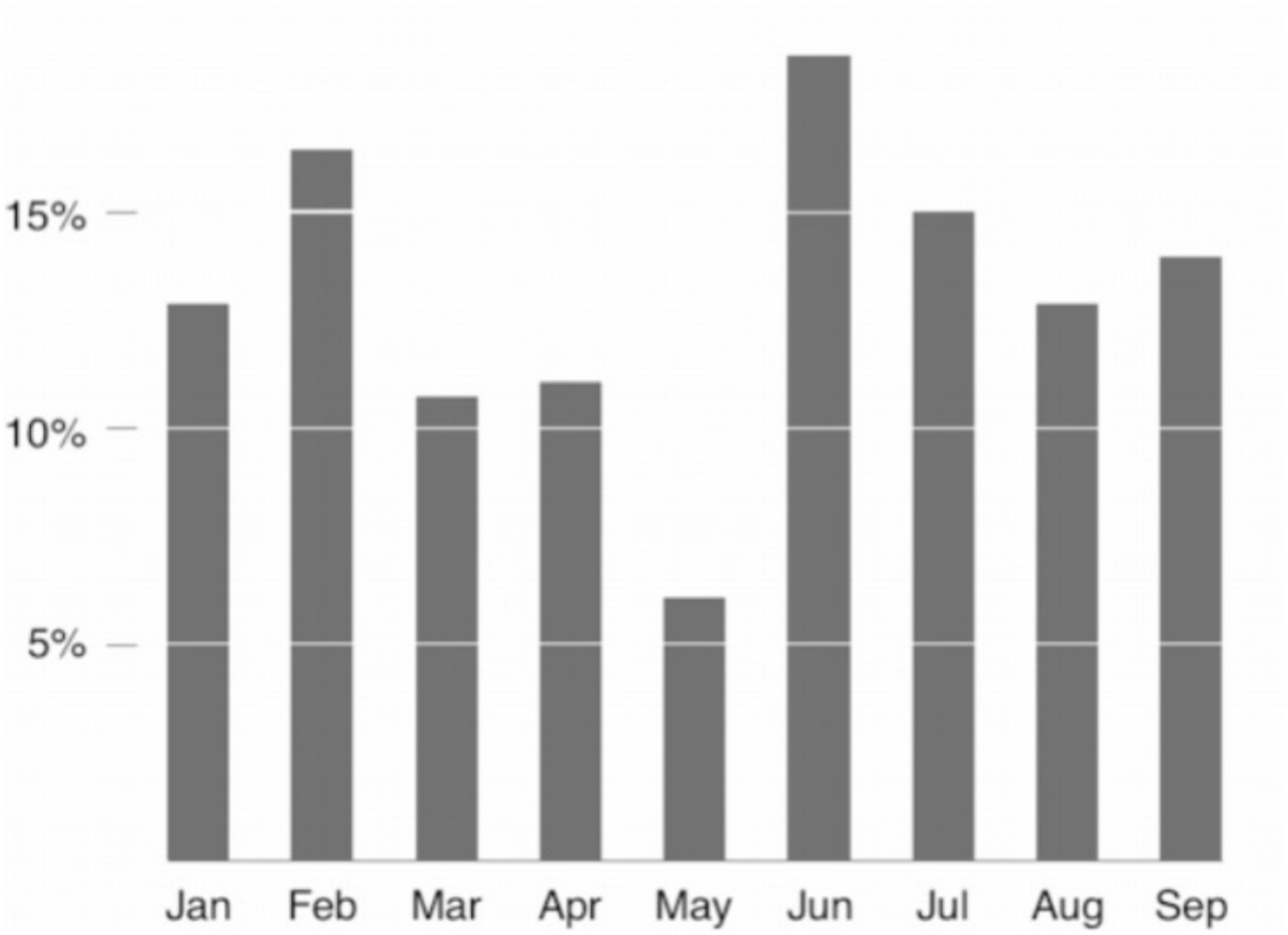
Images from http://www.tbray.org/ongoing/data-ink/di1

Remove non-axis lines, they did not bound data extents, just chart extents



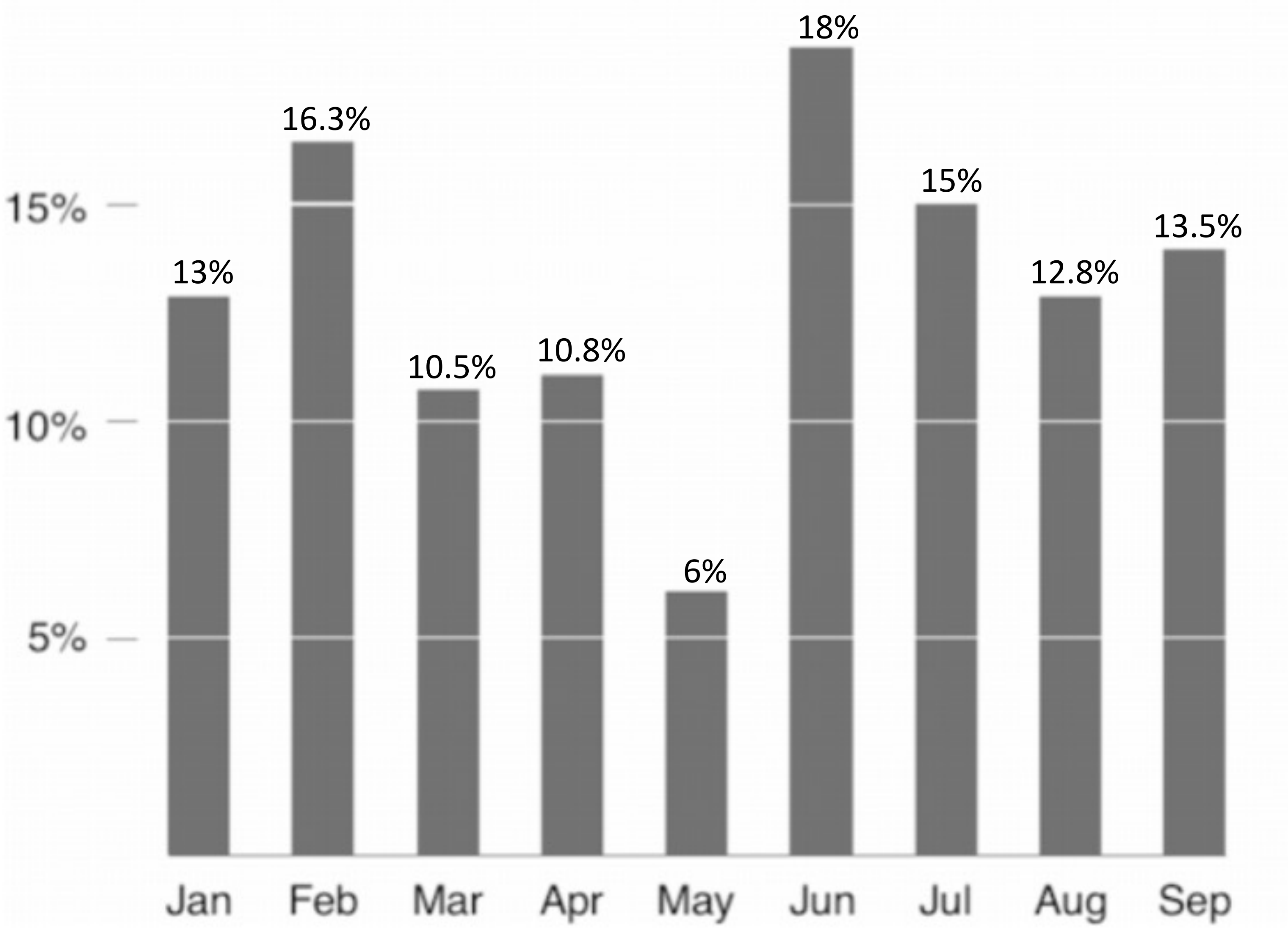
Images from http://www.tbray.org/ongoing/data-ink/di1

An alternative to the initial grid lines...



Images from http://www.tbray.org/ongoing/data-ink/di1

Direct Labeling



Images from http://www.tbray.org/ongoing/data-ink/dI1

Data-Ink Ratio & Caution

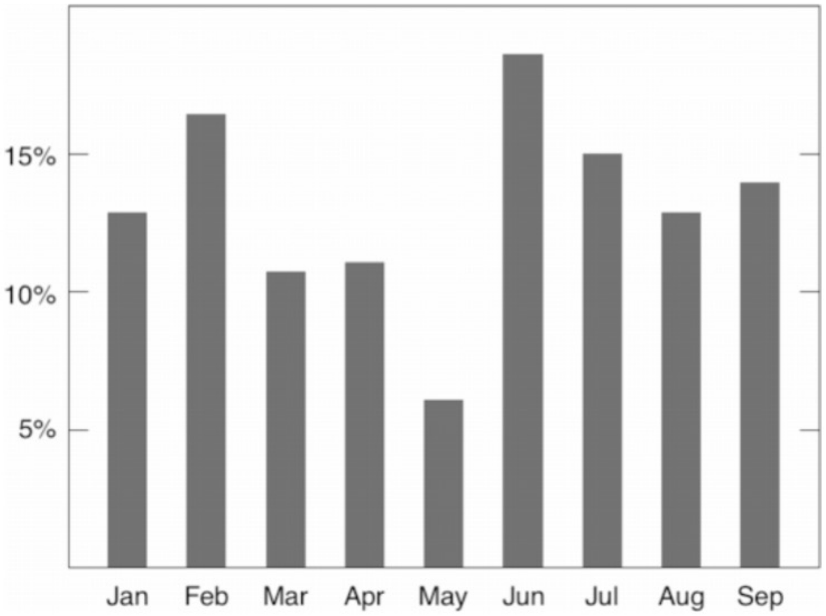
Data-Ink Ratio was coined by Edward Tufte in 1983:

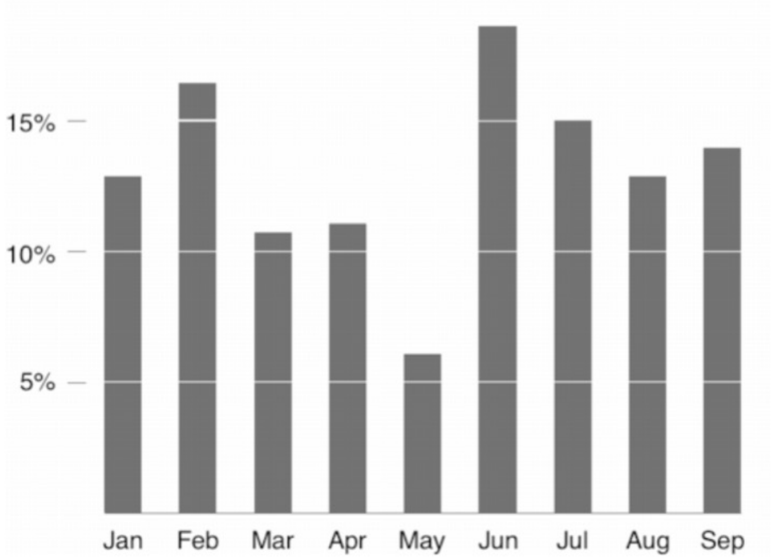
$$\frac{\text{Ink used to non-redundantly represent Data}}{\text{Ink used in the chart in Total}}$$

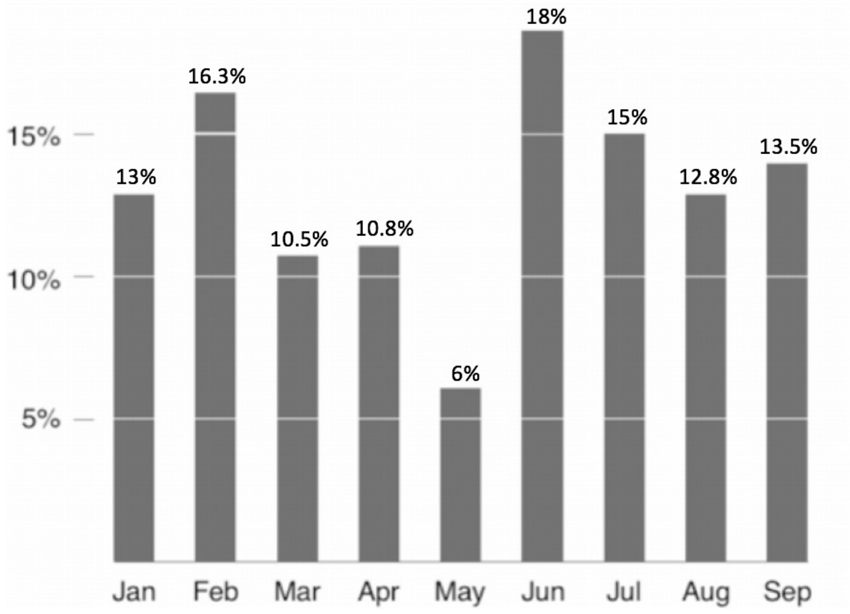
The goal is to have a high data-ink ratio.

Problems?

Which chart is better?







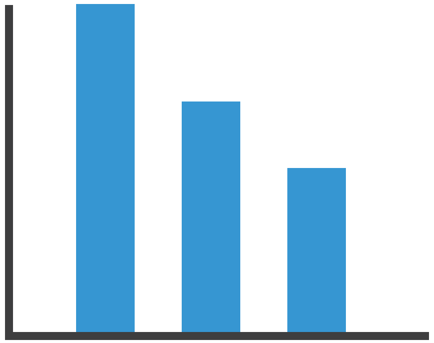
Images from http://www.tbray.org/ongoing/data-ink/di1

Truncating the y-axis

How much has Dickey’s knuckleball slowed?

Images https://www.huffingtonpost.com/raviparikh/lie-with-data-visualization_b_5169715.html

Where should the y-axis start?



Length
(aligned)

In bar charts, bar length is being compared. Therefore, starting the y-axis at an arbitrary position will work against the visualization task… often tricking the viewer.

This is referred to as "*truncating the y-axis*." Be careful when you start at something other than zero.

Images from Munzner, Visual Analysis and Design

What’s happening here?

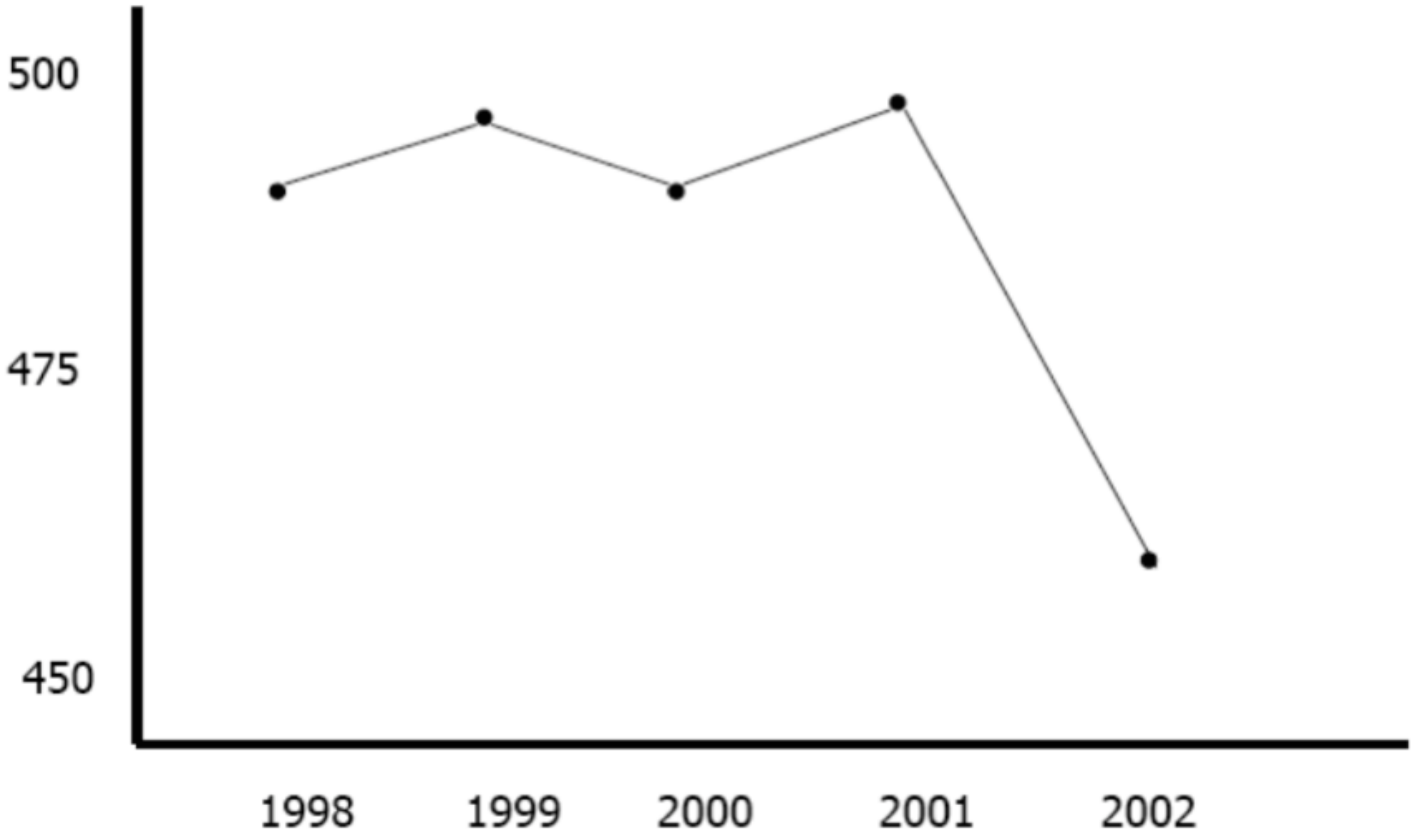


Image from slides of Josh Levine, adapted from John Stasko

What’s happening here?

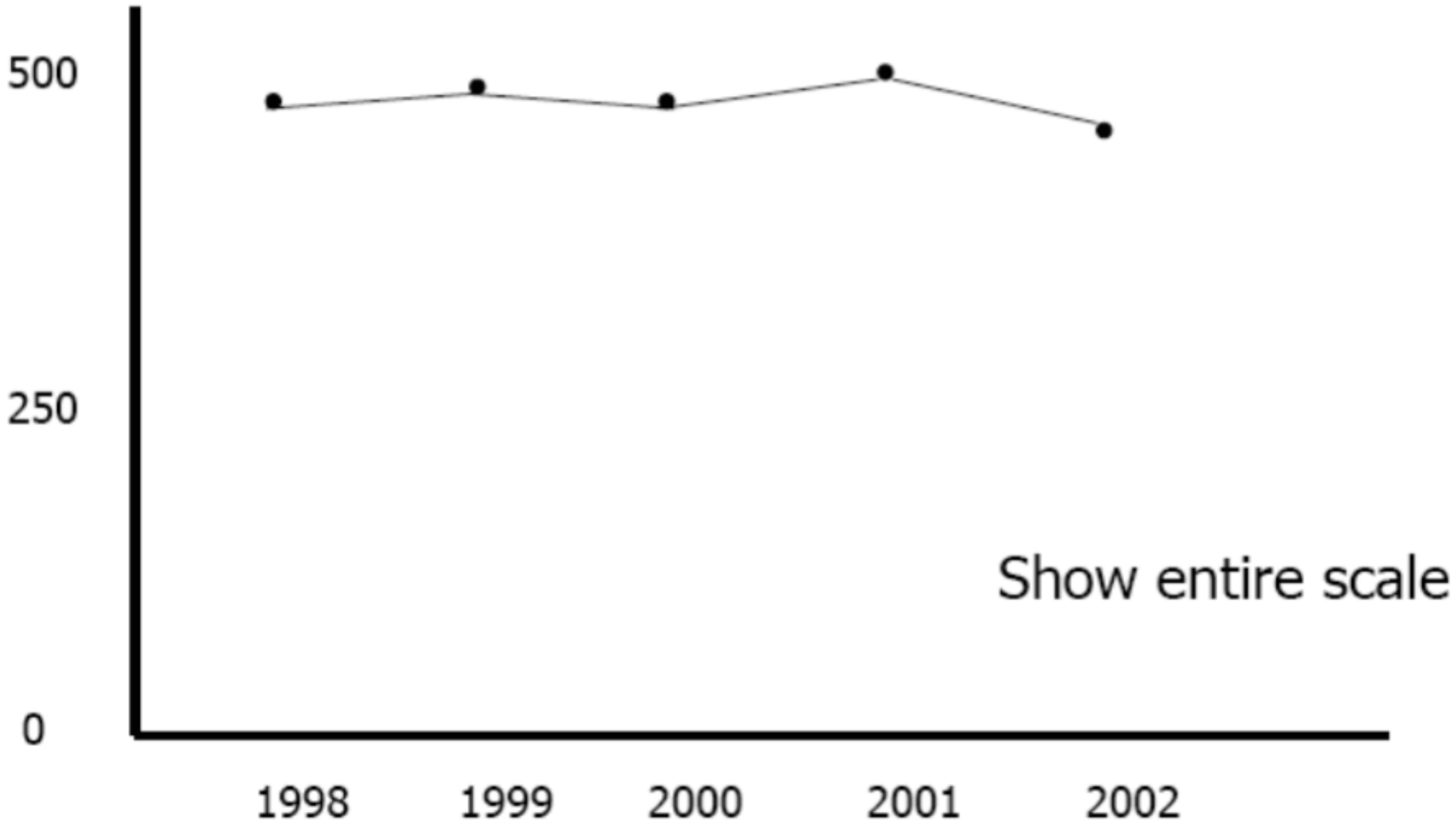


Image from slides of Josh Levine, adapted from John Stasko

What's happening here?

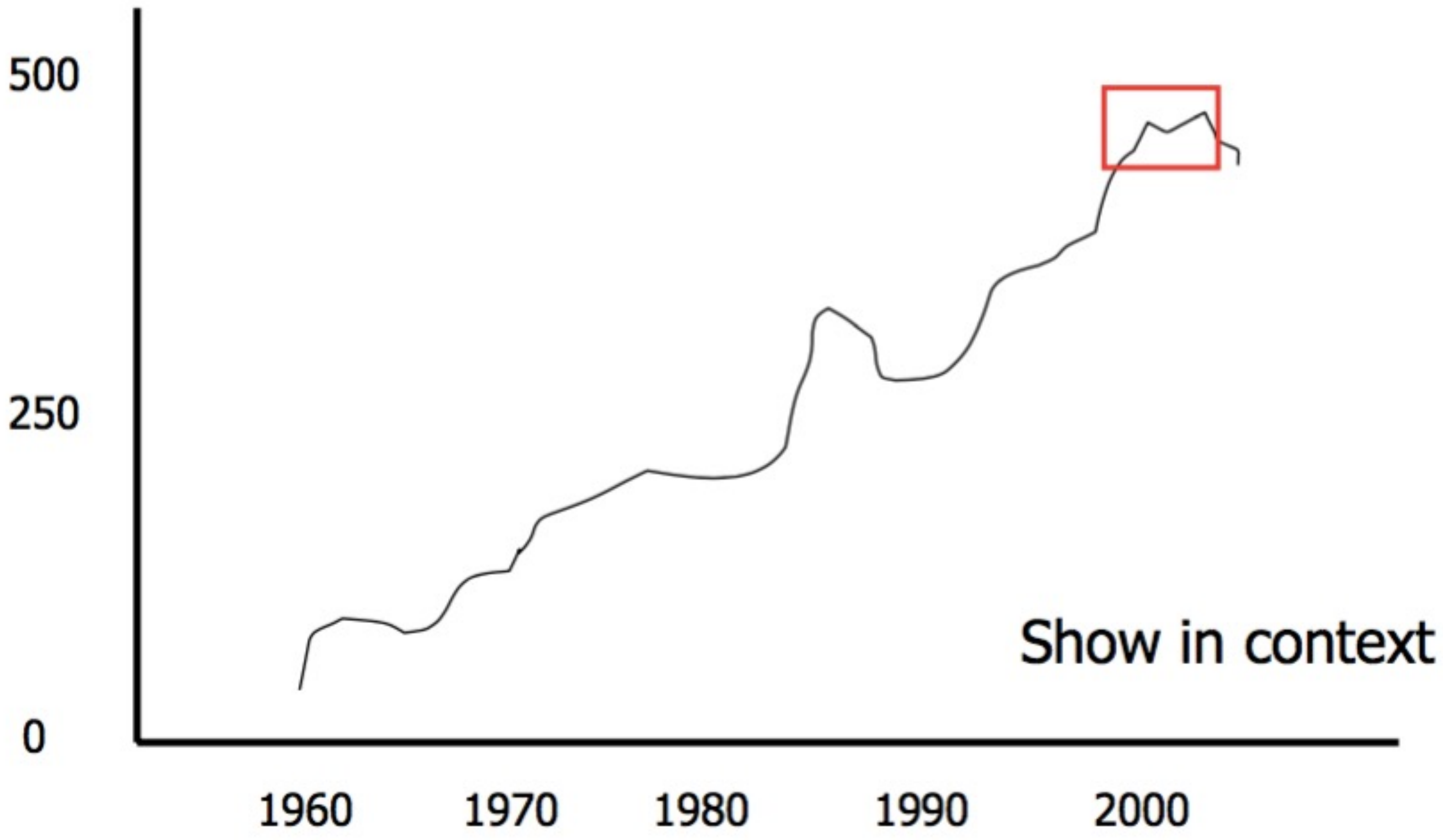
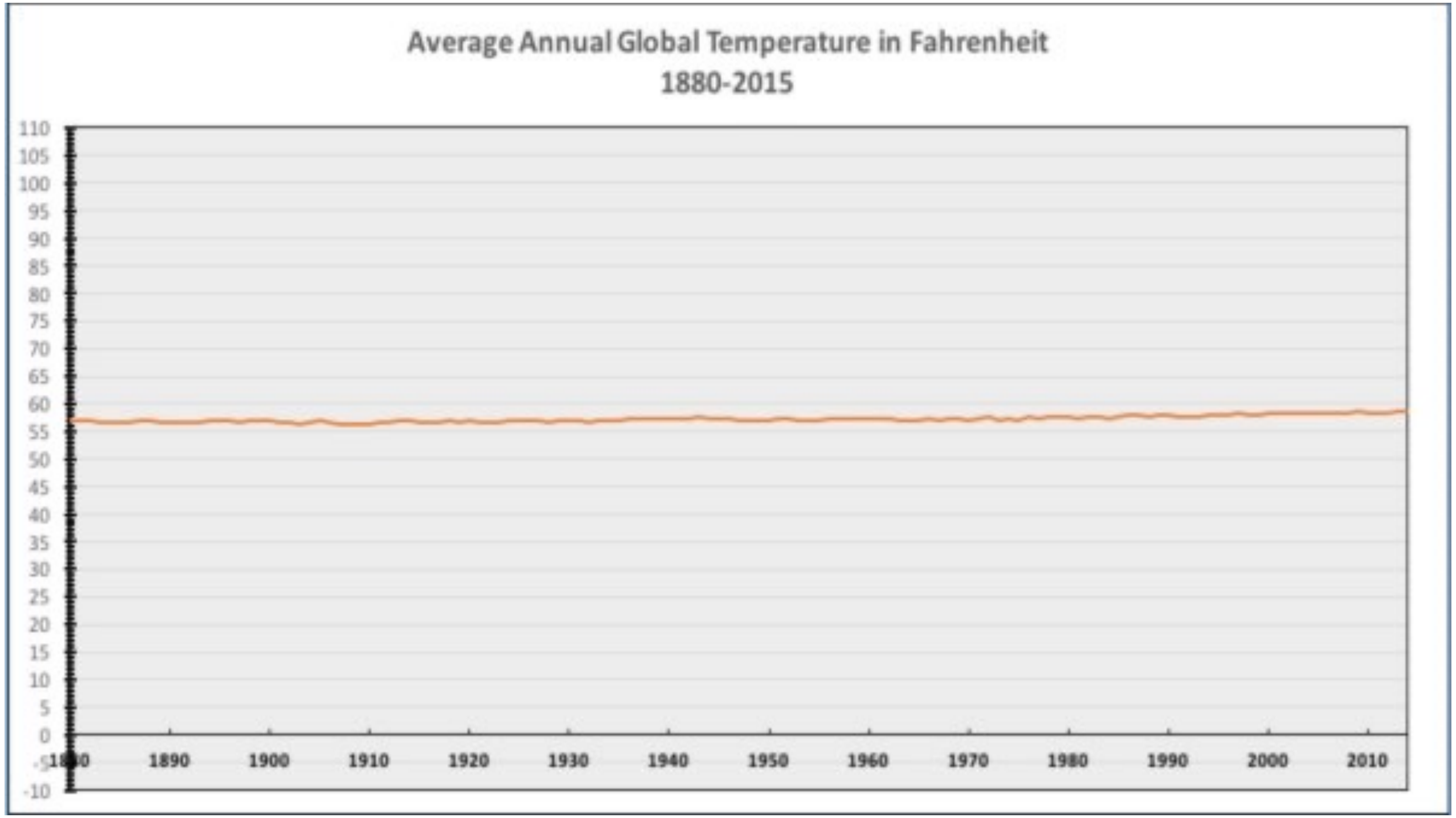


Image from slides of Josh Levine, adapted from John Stasko

Where should the y-axis start?



There are several possible "zero" points. Which one is the most natural?

Line graphs are generally used to analyze *change* in a range rather than absolute. The analysis task matters!

Image from the National Review

Meta-Psychology, 2019, vol 3, MP.2018.895,
https://doi.org/10.5626/MP.2018.895
Article type: Original Article

Open data: Yes
Open materials: Yes
Open and reproducible analysis: Yes
Open reviews and editorial process: Yes
Preregistration: No

Edited by: Felix Schönbrodt
Reviewed by: Julia Rohrer, Gordon Feld, Steve Haroz
Analysis reproduced by: Tobias Mühlmeister
All supplementary files can be accessed at the OSF project page: https://doi.org/10.17605/OSF.IO/HXK2U

Graph Construction: An Empirical Investigation on Setting the Range of the Y-Axis

Jessica K. Witt
Colorado State University

Graphs are an effective and compelling way to present scientific results. With few rigid guidelines, researchers have many degrees-of-freedom regarding graph construction. One such choice is the range of the y-axis. A range set just beyond the data will bias readers to see all effects as big. Conversely, a range set to the full range of options will bias readers to see all effects as small. Researchers should maximize congruence between visual size of an effect and the actual size of the effect. In the experiments presented here, participants viewed graphs with the y-axis set to the minimum range required for all the data to be visible, the full range from 0 to 100, and a range of approximately 1.5 standard deviations. The results showed that participants' sensitivity to the effect depicted in the graph was better when the y-axis range was between one to two standard deviations than with either the minimum range or the full range. In addition, bias was also smaller with the standardized axis range than the minimum or full axis ranges. To achieve congruency in scientific fields for which effects are standardized, the y-axis range should be no less than 1 standard deviations, and aim to be at least 1.5 standard deviations.

Keywords: Graph Design, Effect size, Sensitivity, Bias

One way to lie with statistics is to set the range of the y-axis to form a misleading impression of the data. A range set too narrow will exaggerate a small effect and can even make a non-significant trend appear to be a substantial effect (Pandey, Rall, Satterthwaite, Nov, & Bertini, 2015). Yet the default setting of many statistical and graphing software package-range set too wide also creates a misleading impression of the data by making effects seem smaller than they are. Here, I argue that for scientific fields that use standardized effect sizes and adopt Cohen's convention that an effect of $d = 0.8$ is big, the range of the y-axis should be approximately 1.5 standard deviations (SDs).

What range should I use?

Data range?
Start from zero?

Like all things,
depends on the task.

It’s not just the y-axis: aspect-ratio also affects perception

Changing the aspect-ratio changes the orientation of the lines.

A common rule is “banking to 45°,” for comparing slopes.

But where does this come from? And do you need to compare slopes?

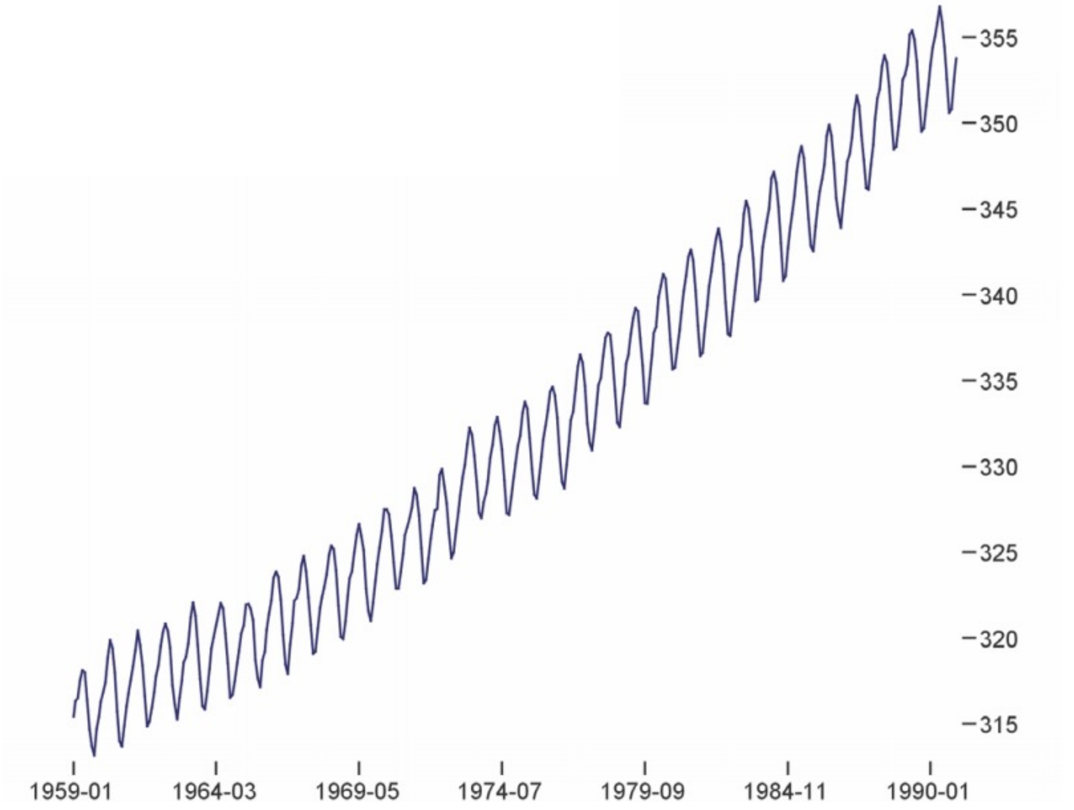


Figure 1a. CO_2 measurements, aspect ratio = 1.17. The x-axis shows time, in monthly increments, while the y-axis shows carbon dioxide measurements taken at the Mauna Loa observatory [2]. Note the bend in the trend of increasing values, suggesting an accelerating increase.

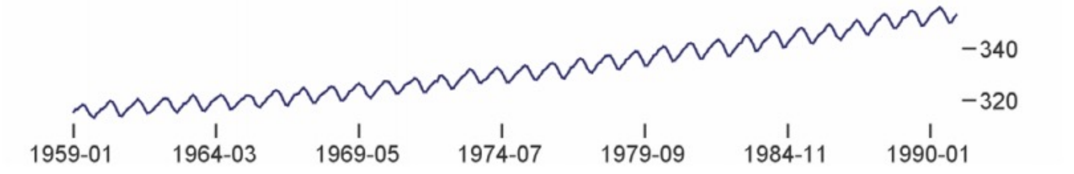
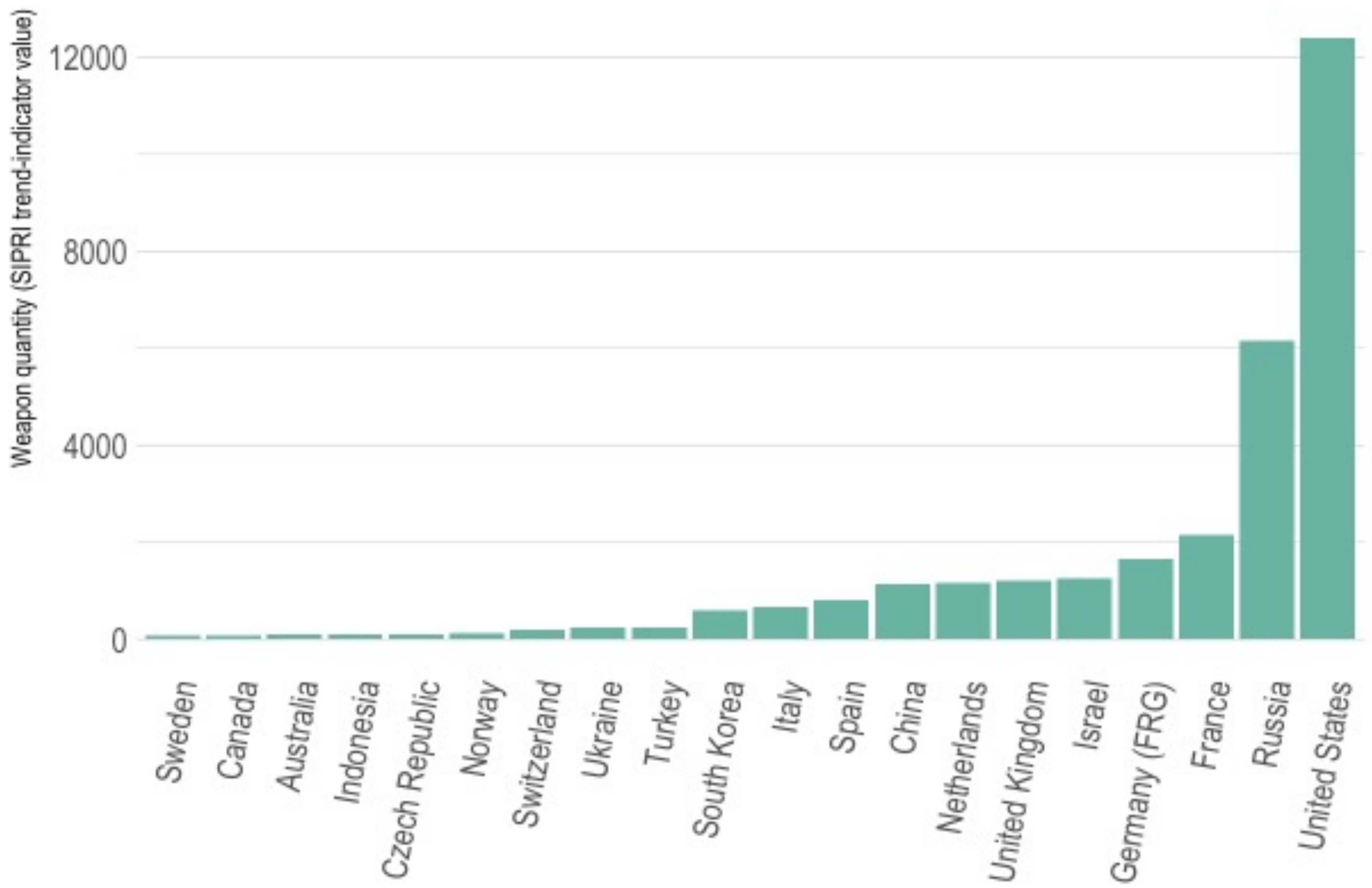


Figure 1b. CO_2 measurements, aspect ratio = 7.87. The wider aspect ratio enables the viewer to see that the ascent of each yearly cycle is more gradual than its decay. However, the bend in the lower-frequency trend is now difficult to see. The choice of aspect ratios for Figures 1a and 1b were automatically determined using multi-scale banking.

Image from Heer & Agrawala, *Multi-scale Banking to 45°*

Rotating for Readability

Be Aware of Text Angle

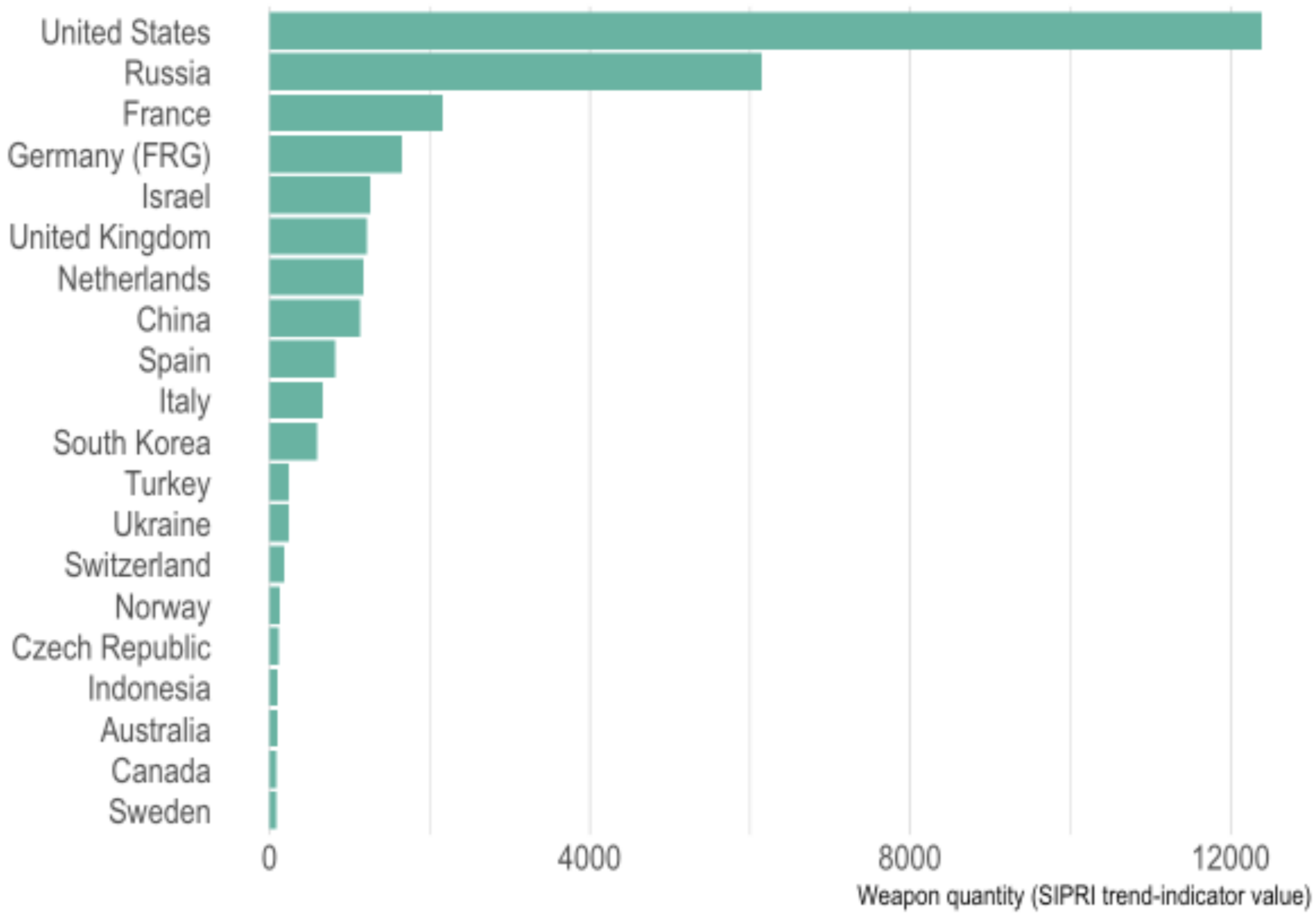


Users shouldn't feel the need to tilt their head to read labels.

Furthermore, while we can rotate fonts at any angle, they distort and become jagged.

Image from https://www.data-to-viz.com/caveat/hard_label.html

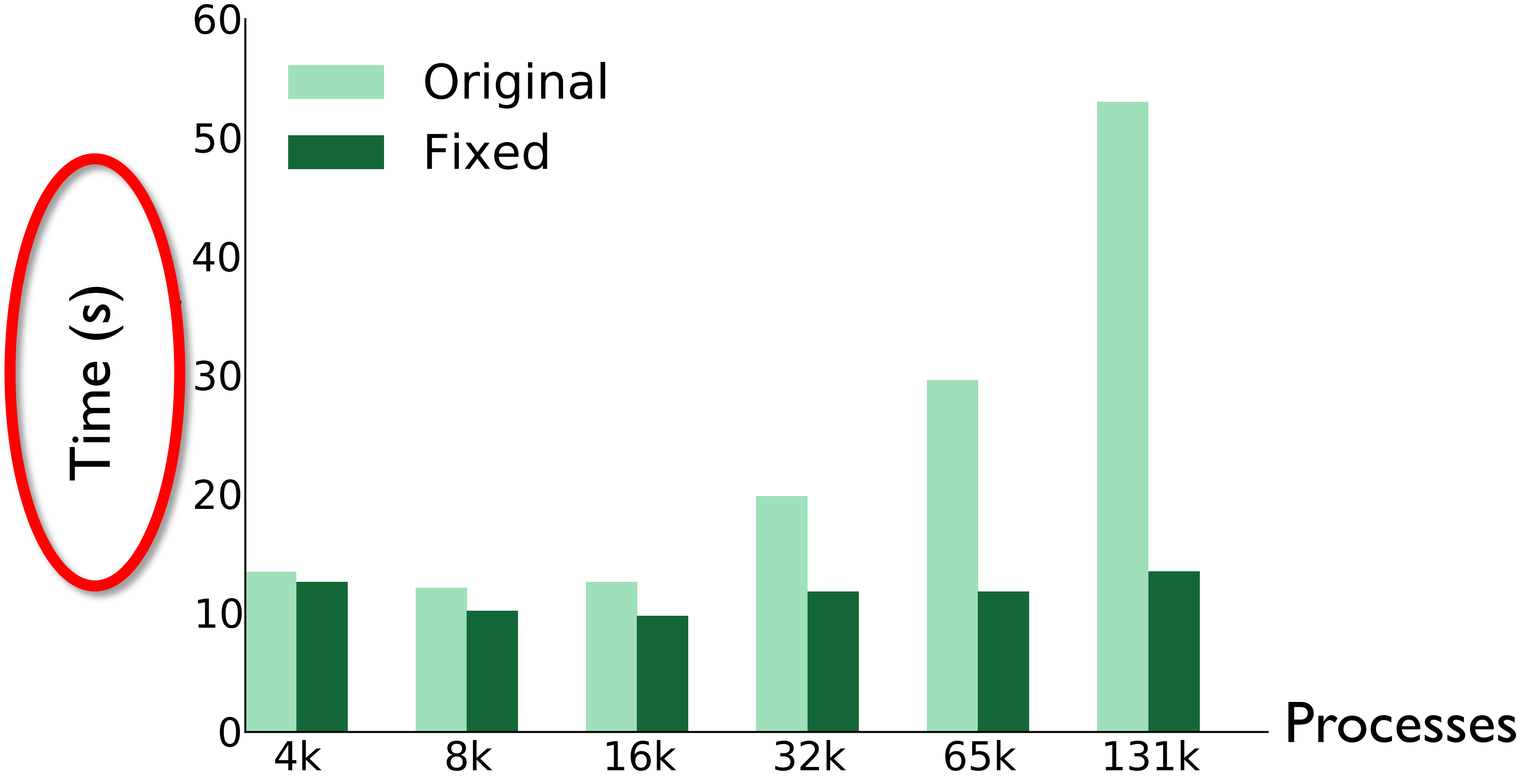
Consider rotating a bar chart



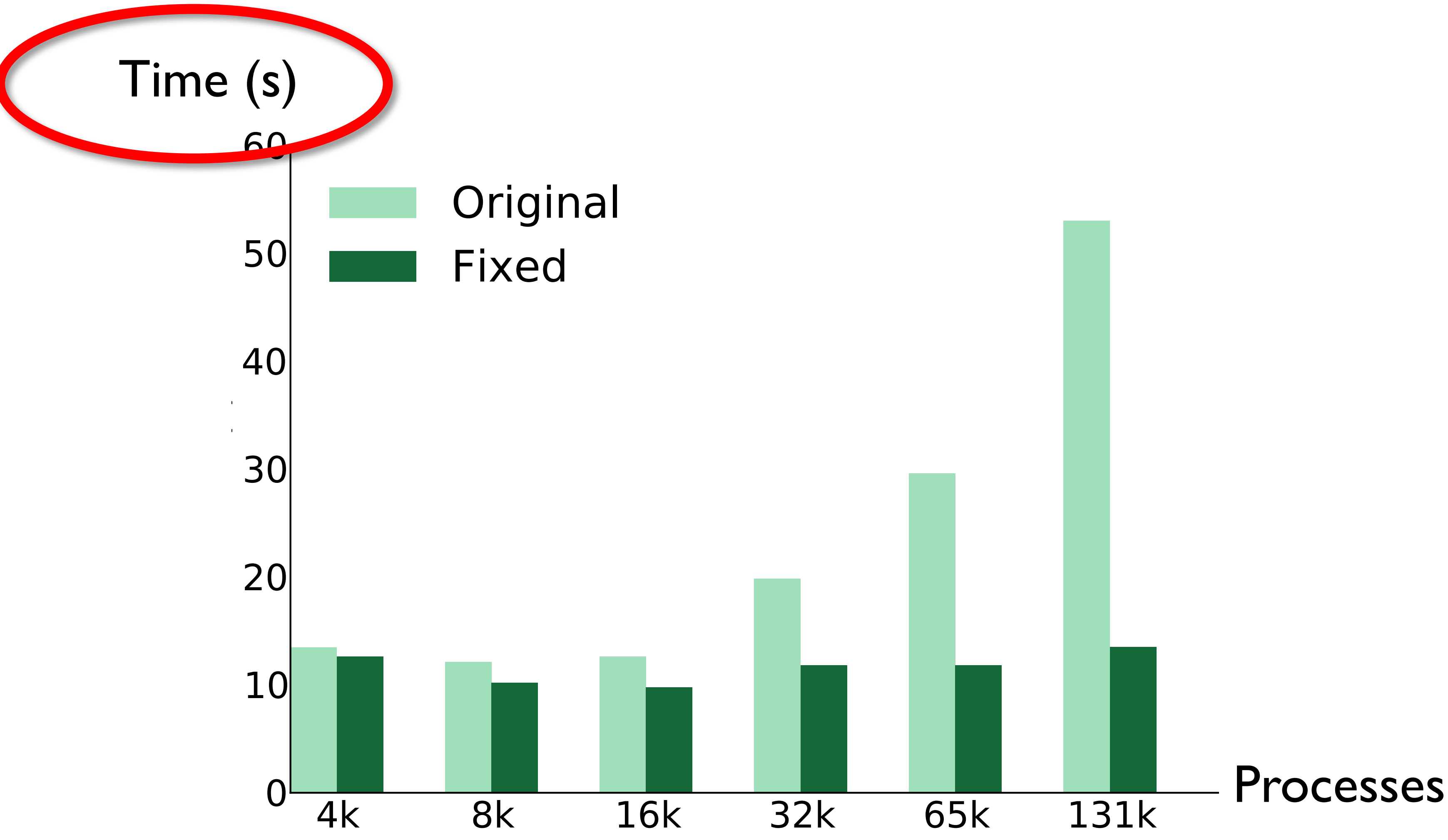
*Note however that some axes have a strong association with the x-axis (e.g., time). In that case, the design trade off may leave tilted text.

Image from https://www.data-to-viz.com/caveat/hard_label.html

Labels on the y-axis need not be vertical



Labels on the y-axis need not be vertical



When the rotation bucks convention, it may be misinterpreted

What is your initial reaction?

The designer's desire was to evoke blood running down a wall.

Takeaway: any design counter to well known conventions must be strongly justified.

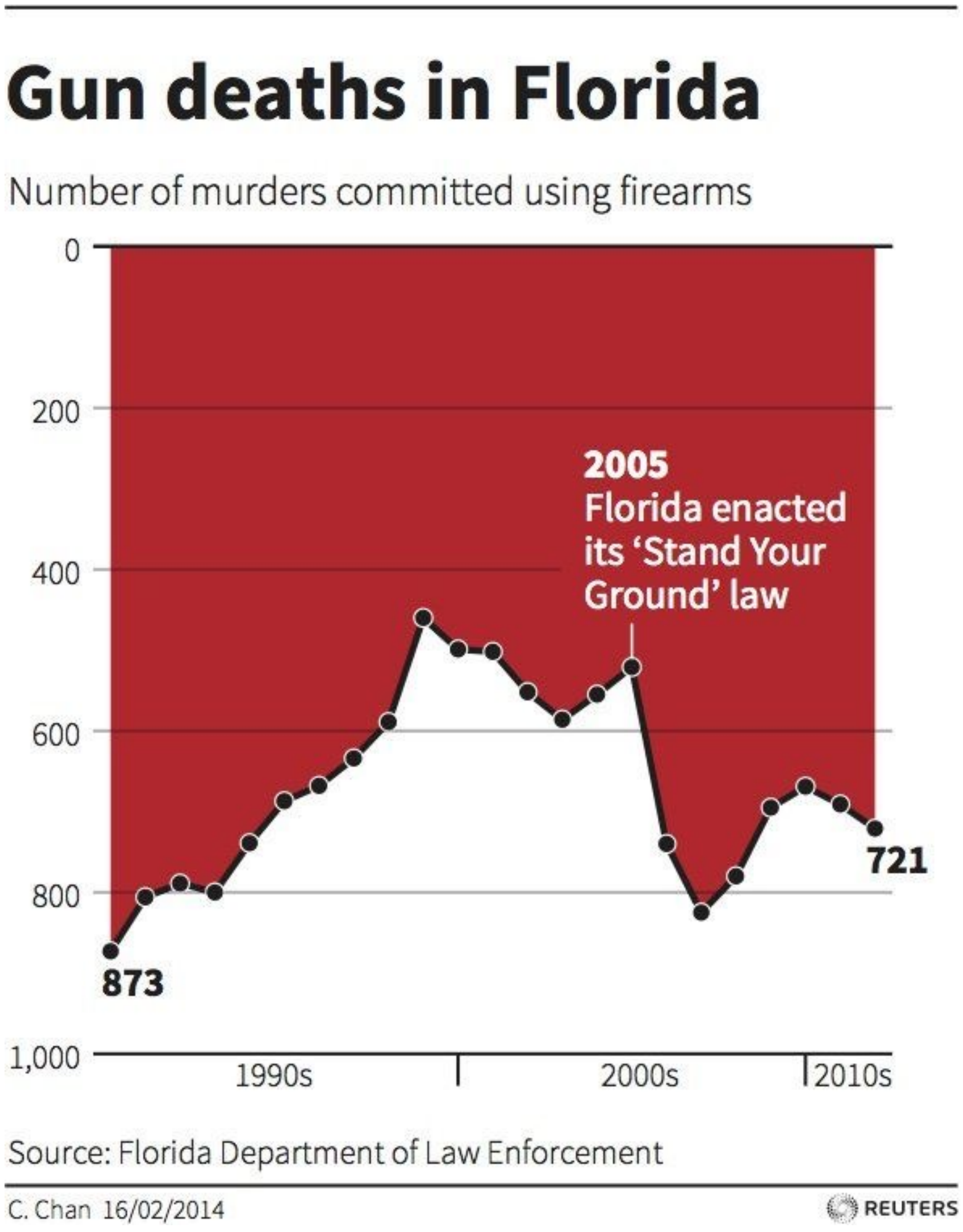
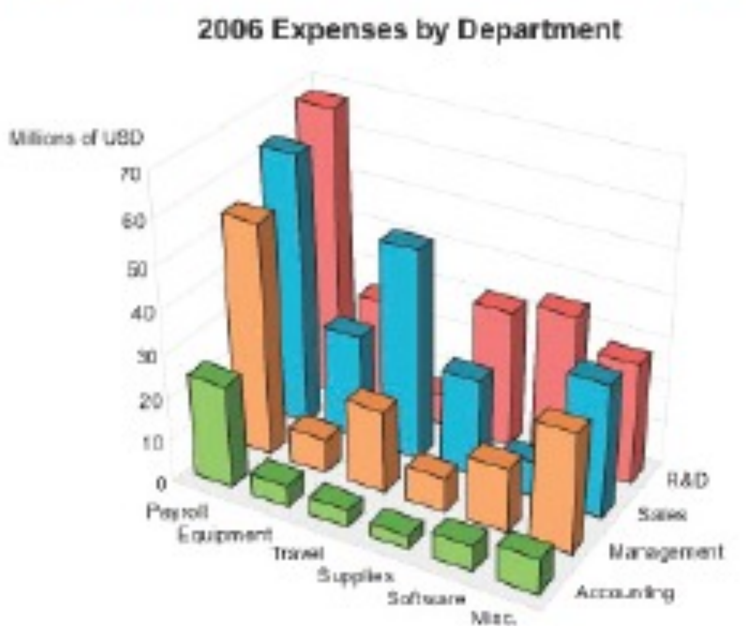


Image retrieved from https://www.livescience.com/45083-misleading-gun-death-chart.html

Using 3D

Justify the Use of 3D or Remove it

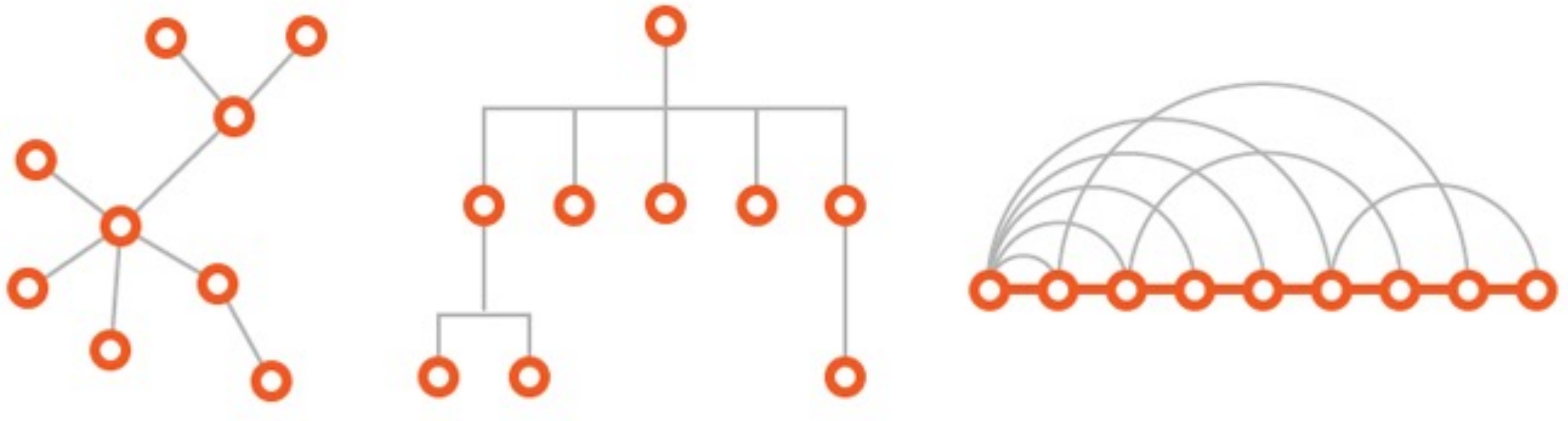


Evidence for the effectivenes of position and length apply to 2D only.

3D means occlusion and perceived depth can cause errors in judgement.

However, 3D excels for shape perception of truly 3D objects.

For Graphs, 2D Shows Topology



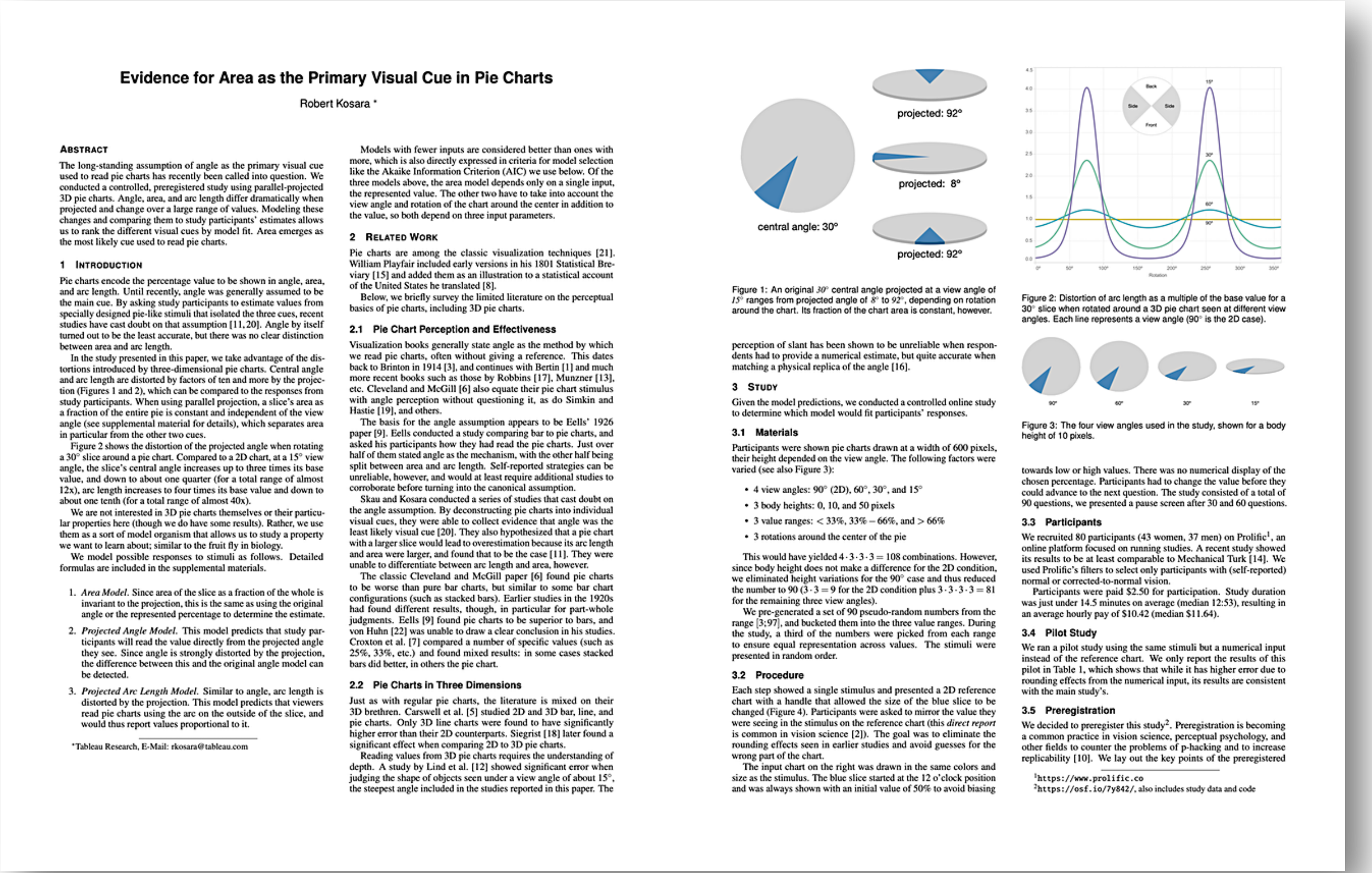
Justify your design choices for your tasks!

Pie Charts

Pie Charts are often maligned with the justification that length is a more effective channel than angle

…but, is angle what we're really perceiving?

There are still many things we don't understand about what is actually being perceived or measured in any given chart.



Evidence for Area as the Primary Visual Cue in Pie Charts

Robert Kosara *

Abstract

The long-standing assumption of angle as the primary visual cue used to read pie charts has recently been called into question. We conducted a controlled, preregistered study using parallel-projected 3D pie charts. Angle, area, and arc length differ dramatically when projected and change over a large range of values. Modeling these changes and comparing them to study participants' estimates allows us to rank the different visual cues by model fit. Area emerges as the most likely cue used to read pie charts.

1 Introduction

Pie charts encode the percentage value to be shown in angle, area, and arc length. Until recently, angle was generally assumed to be the main cue. By asking study participants to estimate values from specially designed pie-like stimuli that isolated the three cues, recent studies have cast doubt on that assumption [11,20]. Angle by itself turned out to be the least accurate, but there was no clear distinction between area and arc length.

In the study presented in this paper, we take advantage of the distortions introduced by three-dimensional pie charts. Central angle and arc length are distorted by factors of ten and more by the projection (Figures 1 and 2), which can be compared to the responses from study participants. When using parallel projection, a slice's area as a fraction of the entire pie is constant and independent of the view angle (see supplemental material for details), which separates area in particular from the other two cues.

Figure 2 shows the distortion of the projected angle when rotating a 30° slice around a pie chart. Compared to a 2D chart, at a 15° view angle, the slice's central angle increases up to three times its base value, and down to about one quarter (for a total range of almost 12x), arc length increases to four times its base value and down to about one tenth (for a total range of almost 40x).

We are not interested in 3D pie charts themselves or their particular properties here (though we do have some results). Rather, we use them as a sort of model organism that allows us to study a property we want to learn about; similar to the fruit fly in biology.

We model possible responses to stimuli as follows. Detailed formulas are included in the supplemental materials.

1. *Area Model*. Since area of the slice as a fraction of the whole is invariant to the projection, this is the same as using the original angle or the represented percentage to determine the estimate.
2. *Projected Angle Model*. This model predicts that study participants will read the value directly from the projected angle they see. Since angle is strongly distorted by the projection, the difference between this and the original angle model can be detected.
3. *Projected Arc Length Model*. Similar to angle, arc length is distorted by the projection. This model predicts that viewers read pie charts using the arc on the outside of the slice, and would thus report values proportional to it.

*Tableau Research, E-Mail: rkosara@tableau.com

Models with fewer inputs are considered better than ones with more, which is also directly expressed in criteria for model selection like the Akaike Information Criterion (AIC) we use below. Of the three models above, the area model depends only on a single input, the represented value. The other two have to take into account the view angle and rotation of the chart around the center in addition to the value, so both depend on three input parameters.

2 Related Work

Pie charts are among the classic visualization techniques [21]. William Playfair included early versions in his 1801 Statistical Breviary [15] and added them as an illustration to a statistical account of the United States he translated [8].

Below, we briefly survey the limited literature on the perceptual basics of pie charts, including 3D pie charts.

2.1 Pie Chart Perception and Effectiveness

Visualization books generally state angle as the method by which we read pie charts, often without giving a reference. This dates back to Brinton in 1914 [3], and continues with Bertin [1] and much more recent books such as those by Robbins [17], Munzner [13], etc. Cleveland and McGill [6] also equate their pie chart stimulus with angle perception without questioning it, as do Simkin and Hastie [19], and others.

The basis for the angle assumption appears to be Eells' 1926 paper [9]. Eells conducted a study comparing bar to pie charts, and asked his participants how they had read the pie charts. Just over half of them stated angle as the mechanism, with the other half being split between area and arc length. Self-reported strategies can be unreliable, however, and would at least require additional studies to corroborate before turning into the canonical assumption.

Skau and Kosara conducted a series of studies that cast doubt on the angle assumption. By deconstructing pie charts into individual visual cues, they were able to collect evidence that angle was the least likely visual cue [20]. They also hypothesized that a pie chart with a larger slice would lead to overestimation because its arc length and area were larger, and found that to be the case [11]. They were unable to differentiate between arc length and area, however.

The classic Cleveland and McGill paper [6] found pie charts to be worse than pure bar charts, but similar to some bar chart configurations (such as stacked bars). Earlier studies in the 1920s had found different results, though, in particular for part-whole judgments. Eells [9] found pie charts to be superior to bars, and von Huhn [22] was unable to draw a clear conclusion in his studies. Croxton et al. [7] compared a number of specific values (such as 25%, 33%, etc.) and found mixed results: in some cases stacked bars did better, in others the pie chart.

2.2 Pie Charts in Three Dimensions

Just as with regular pie charts, the literature is mixed on their 3D brethren. Carswell et al. [5] studied 2D and 3D bar, line, and pie charts. Only 3D line charts were found to have significantly higher error than their 2D counterparts. Siegrist [18] later found a significant effect when comparing 2D to 3D pie charts.

Reading values from 3D pie charts requires the understanding of depth. A study by Lind et al. [12] showed significant error when judging the shape of objects seen under a view angle of about 15°, the steepest angle included in the studies reported in this paper. The

projected: 92°
projected: 8°
central angle: 30°
projected: 92°

Figure 1: An original 30° central angle projected at a view angle of 15° ranges from projected angle of 8° to 92°, depending on rotation around the chart. Its fraction of the chart area is constant, however.

Figure 2: Distortion of arc length as a multiple of the base value for a 30° slice when rotated around a 3D pie chart seen at different view angles. Each line represents a view angle (90° is the 2D case).

90°　60°　30°　15°

Figure 3: The four view angles used in the study, shown for a body height of 10 pixels.

perception of slant has been shown to be unreliable when respondents had to provide a numerical estimate, but quite accurate when matching a physical replica of the angle [16].

3 Study

Given the model predictions, we conducted a controlled online study to determine which model would fit participants' responses.

3.1 Materials

Participants were shown pie charts drawn at a width of 600 pixels, their height depended on the view angle. The following factors were varied (see also Figure 3):

- 4 view angles: 90° (2D), 60°, 30°, and 15°
- 3 body heights: 0, 10, and 50 pixels
- 3 value ranges: < 33%, 33% – 66%, and > 66%
- 3 rotations around the center of the pie

This would have yielded $4 \cdot 3 \cdot 3 \cdot 3 = 108$ combinations. However, since body height does not make a difference for the 2D condition, we eliminated height variations for the 90° case and thus reduced the number to 90 ($3 \cdot 3 = 9$ for the 2D condition plus $3 \cdot 3 \cdot 3 \cdot 3 = 81$ for the remaining three view angles).

We pre-generated a set of 90 pseudo-random numbers from the range [3;97], and bucketed them into the three value ranges. During the study, a third of the numbers were picked from each range to ensure equal representation across values. The stimuli were presented in random order.

3.2 Procedure

Each step showed a single stimulus and presented a 2D reference chart with a handle that allowed the size of the blue slice to be changed (Figure 4). Participants were asked to mirror the value they were seeing in the stimulus on the reference chart (this *direct report* is common in vision science [2]). The goal was to eliminate the rounding effects seen in earlier studies and avoid guesses for the wrong part of the chart.

The input chart on the right was drawn in the same colors and size as the stimulus. The blue slice started at the 12 o'clock position and was always shown with an initial value of 50% to avoid biasing towards low or high values. There was no numerical display of the chosen percentage. Participants had to change the value before they could advance to the next question. The study consisted of a total of 90 questions, we presented a pause screen after 30 and 60 questions.

3.3 Participants

We recruited 80 participants (43 women, 37 men) on Prolific[1], an online platform focused on running studies. A recent study showed its results to be at least comparable to Mechanical Turk [14]. We used Prolific's filters to select only participants with (self-reported) normal or corrected-to-normal vision.

Participants were paid $2.50 for participation. Study duration was just under 14.5 minutes on average (median 12:53), resulting in an average hourly pay of $10.42 (median $11.64).

3.4 Pilot Study

We ran a pilot study using the same stimuli but a numerical input instead of the reference chart. We only report the results of this pilot in Table 1, which shows that while it has higher error due to rounding effects from the numerical input, its results are consistent with the main study's.

3.5 Preregistration

We decided to preregister this study[2]. Preregistration is becoming a common practice in vision science, perceptual psychology, and other fields to counter the problems of p-hacking and to increase replicability [10]. We lay out the key points of the preregistered

[1] https://www.prolific.co
[2] https://osf.io/7y842/, also includes study data and code

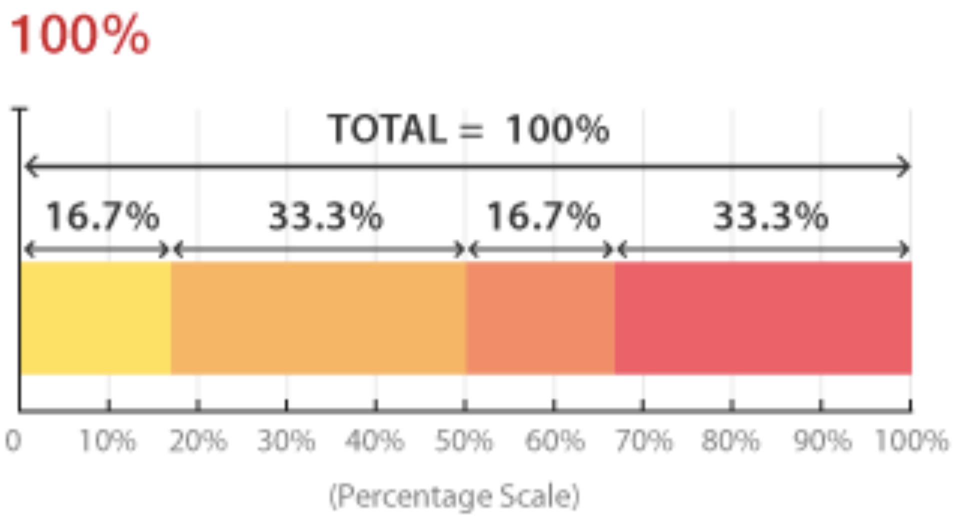
Images from EagerEyes.org

Length is still considered more effective than Area

Stacked bar charts are suggested because they encode with length.

Why might a stacked bar chart not be so good?

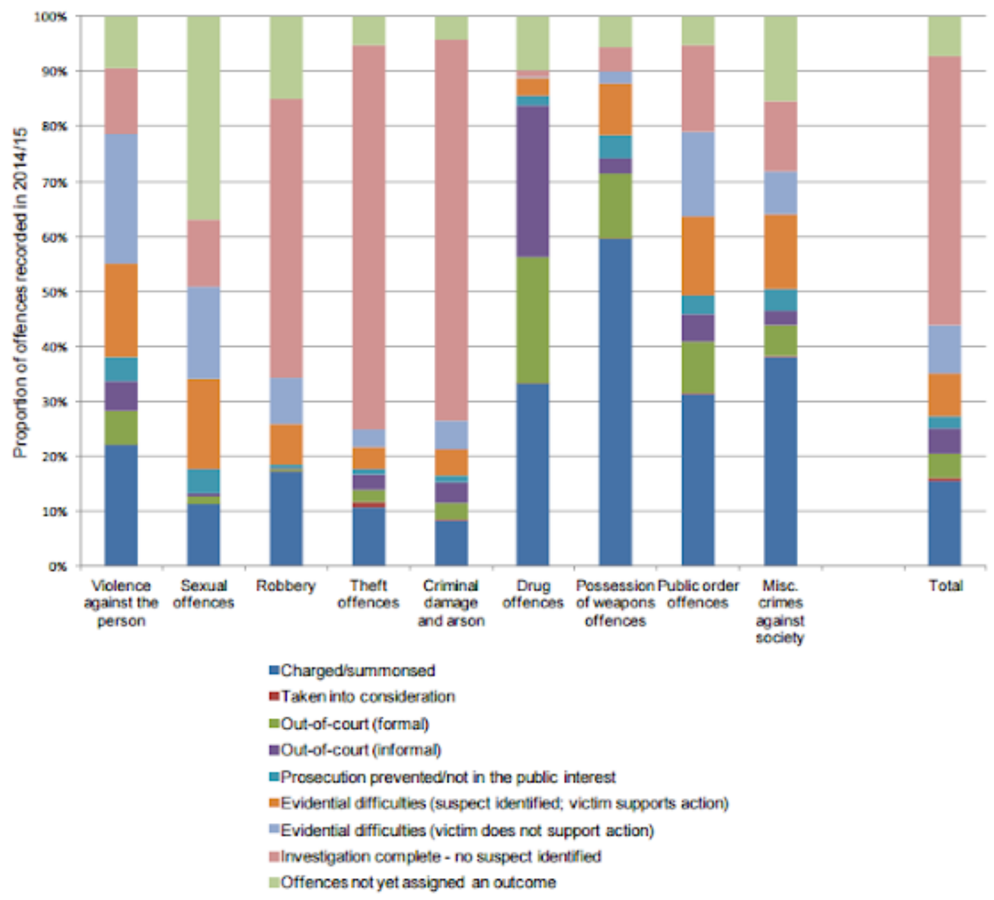
- Violates expressivity – can imply an order where there is none
- May be hard to compare disparate bars



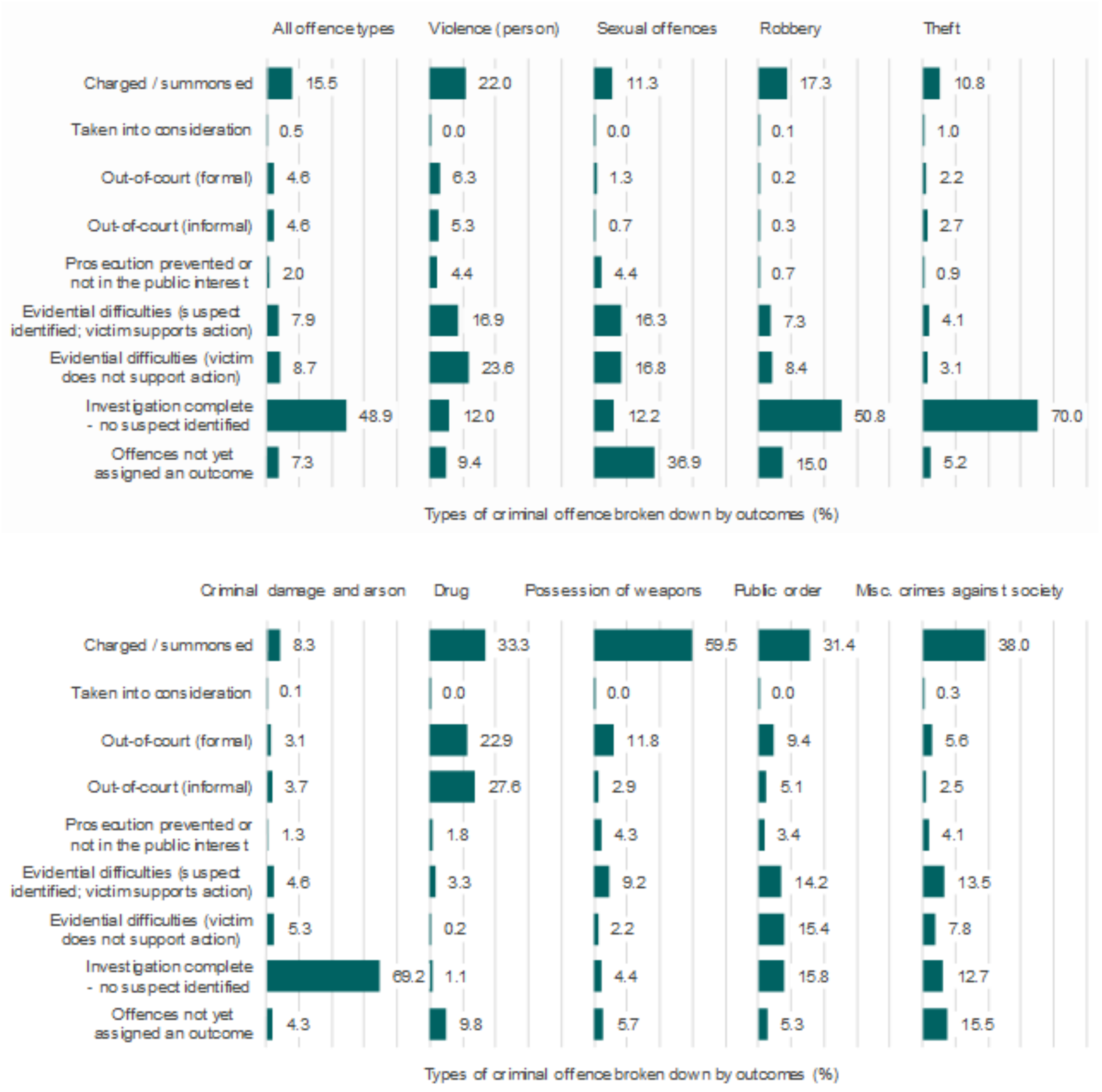
Images from datavizcatalogue

Stacked Bar Charts vs. Small Multiples

Figure 2.1: Outcomes assigned to offences recorded in 2014/15, by outcome group and offence group

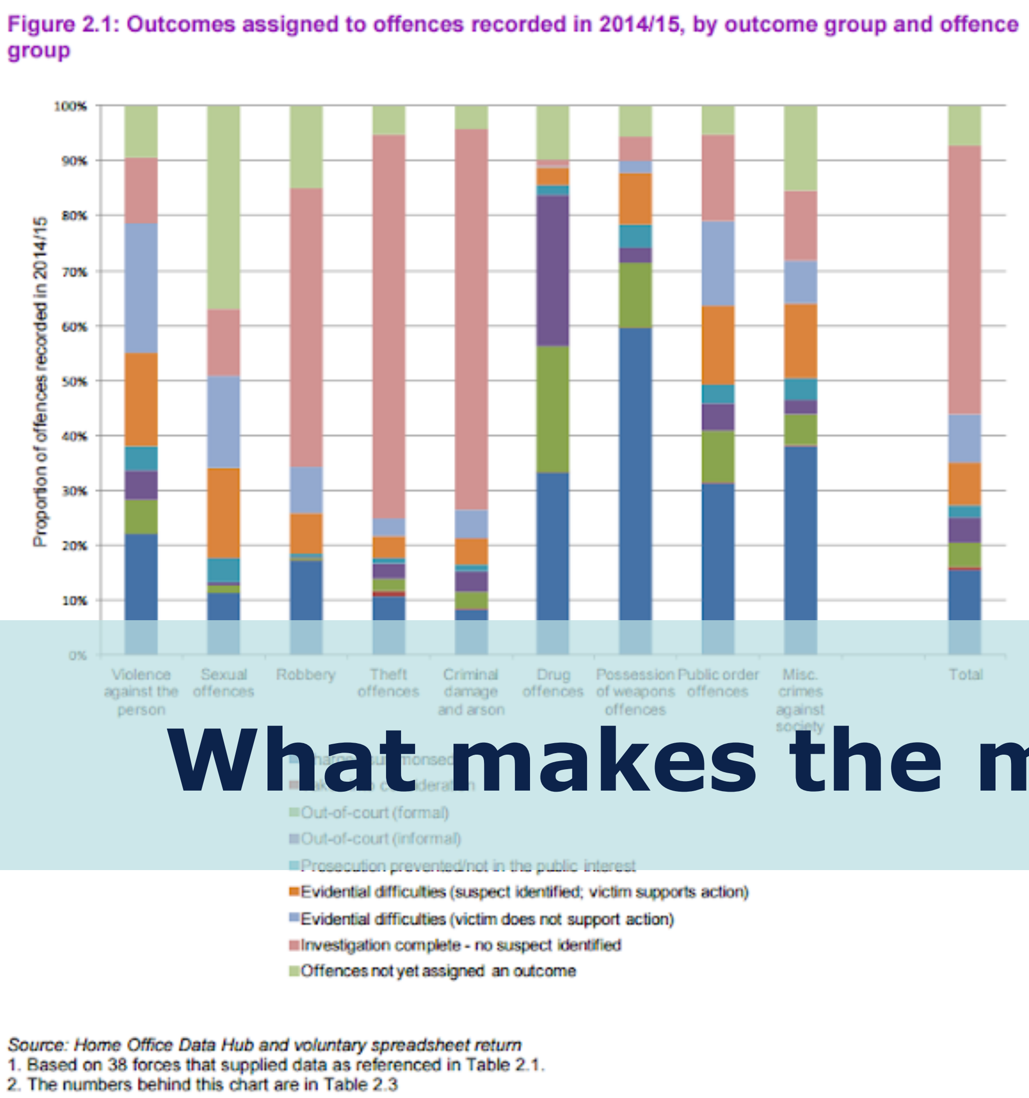


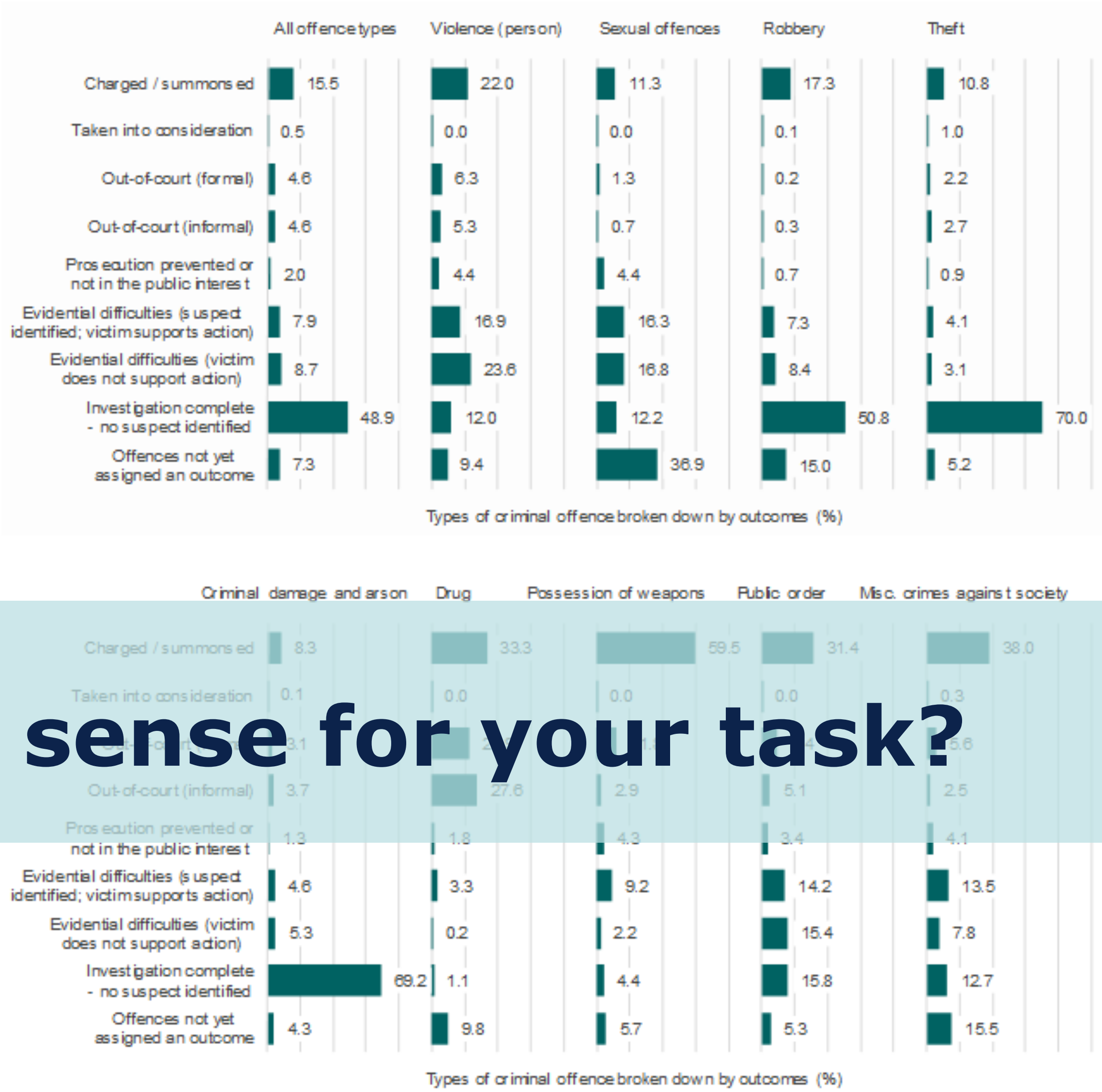
Source: Home Office Data Hub and voluntary spreadsheet return
1. Based on 38 forces that supplied data as referenced in Table 2.1.
2. The numbers behind this chart are in Table 2.3



Images from http://chartartistry.blogspot.com/2016/02/small-multiples-when-stack-bars-just.html

Stacked Bar Charts vs. Small Multiples

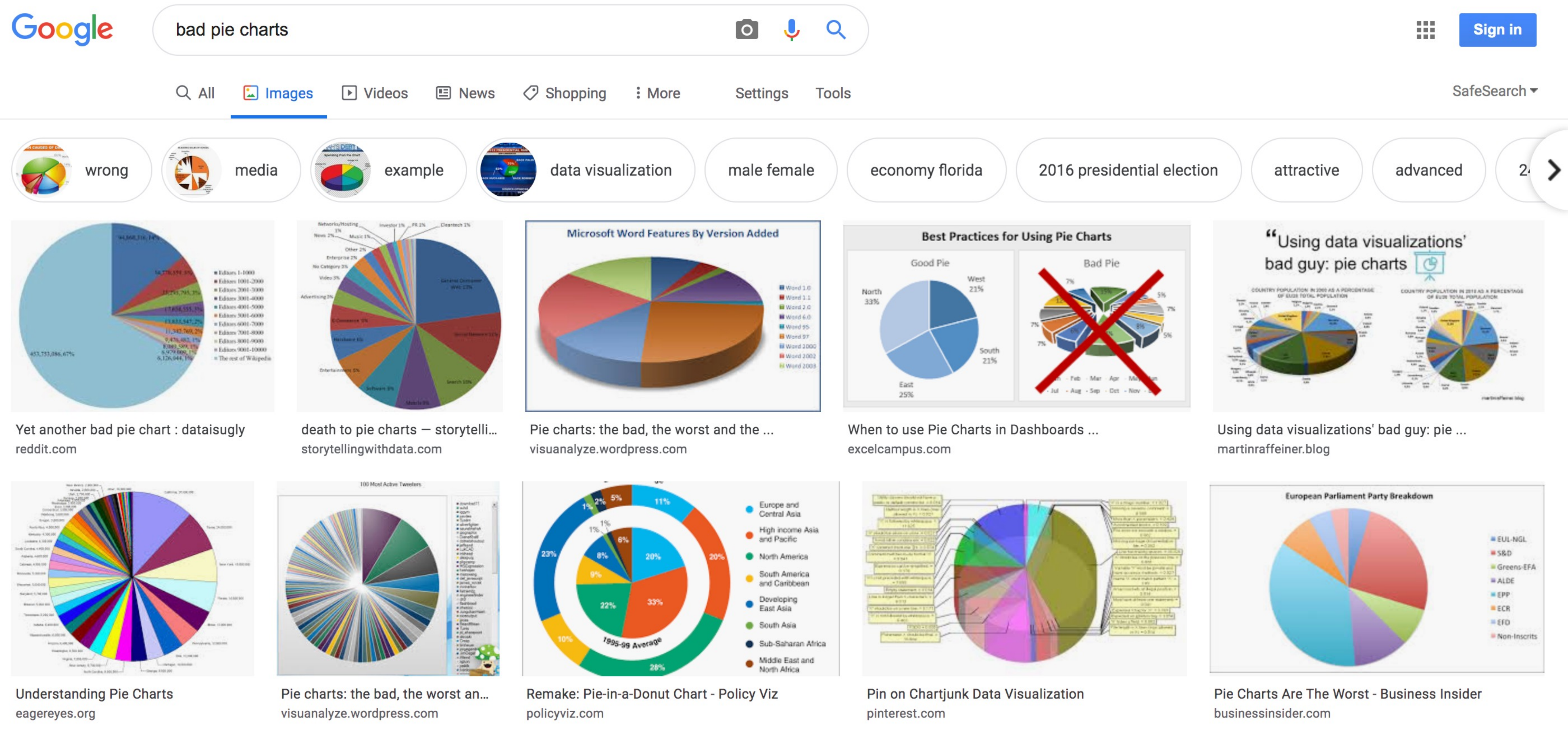




What makes the most sense for your task?

Images from http://chartartistry.blogspot.com/2016/02/small-multiples-when-stack-bars-just.html

Maybe the biggest problem with pie charts is that they have been so often done poorly...



Advice in Data Visualization is a starting point.

Avoid pie charts

Get it right in black and white

Avoid chart junk

Avoid word clouds

Overview first, zoom and filter, details on demand

Don’t truncate the y-axis

Recognize area comparisons

No unjustified 3D

Rotate for readability

Respect conventions

It is generally sufficient for simple visualizations, but harder analysis problems may require more thought.

...which leads us to the most important things in Data Visualization

Data

- Are you showing the right data for your goals?
- Do you understand the form of that data?

Tasks

- Does your representation match your tasks?
- Do your interactions match your tasks?

Design

- Know your data, your tasks, and start from principles, then iterate and evaluate.

Outline

- Data Visualization
- Data Summarization
- Python data tools

Data Summarization

- Raw data are hard to interpret
- Visualizations summarize important aspects of the data
- The *empirical distribution* estimates the distribution on data, but can be hard to interpret
- Summary statistics characterize aspects of the data distribution like:
  - Location / center
  - Scale / spread
  - Skew

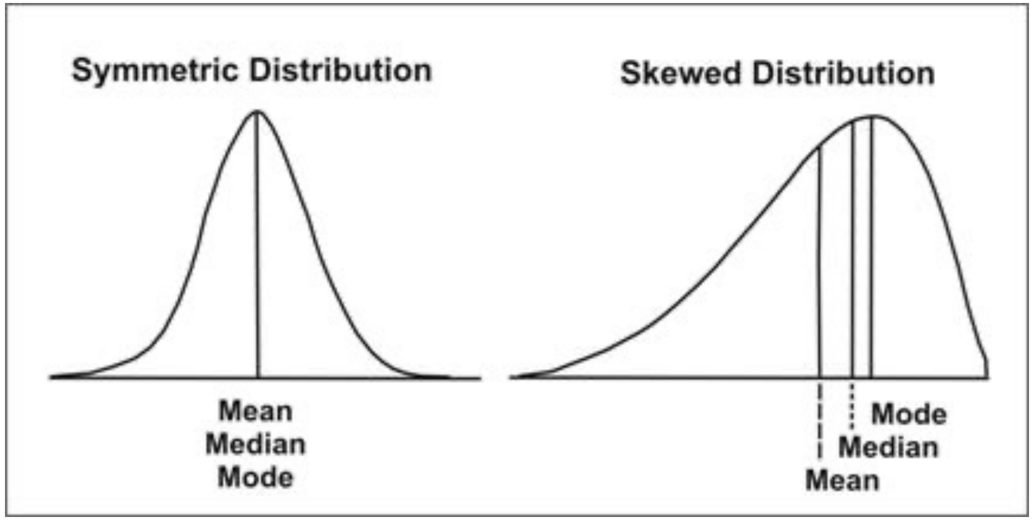
Measuring Location

Three common measures of the distribution location…

Mean Average (expected value) of the data distribution

Median Midpoint – 50% of the probability is below and 50% above

Mode Value of highest probability (mass or density)



…align with symmetric distributions, but diverge with asymmetry

Median

For data $x_1, x_2, \ldots, x_N$ sort the data,

$$x_{(1)}, x_{(2)}, \ldots, x_{(n)}$$

- Notation $x_{(i)}$ means the i-th *lowest* value, e.g. $x_{(i-1)} \leq x_{(i)} \leq x_{(i+1)}$
- $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ are called *order statistics*

If n is **odd** then find the middle datapoint,

$$\text{median}(x_1, \ldots, x_n) = x_{((n+1)/2)}$$

If n is **even** then average between both middle datapoints,

$$\text{median}(x_1, \ldots, x_n) = \frac{1}{2}\left(x_{(n/2)} + x_{(n/2+1)}\right)$$

Median

What is the median of the following data?

1, 2, 3, 4, 5, 6, 8, 9 **4.5**

What is the median of the following data?

1, 2, 3, 4, 5, 6, 8, 100 **4.5**

Median is *robust* to outliers

Sample Mean

Empirical estimate of the true mean of the data distribution,

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Alternatively, if the value x occurs n(x) times in the data then,

$$\bar{x} = \frac{1}{N}\sum_{x} x n(x) = \sum_{x} x p(x) \quad \text{where} \quad \underbrace{p(x) = \frac{n(x)}{N}}_{\text{Empirical Distribution}}$$

Recall

- Law of Large Numbers says $\bar{x}$ goes to mean E[X]
- Central Limit Theorem says $\bar{x}$ has Normal distribution

Law of Large Numbers (LLN)

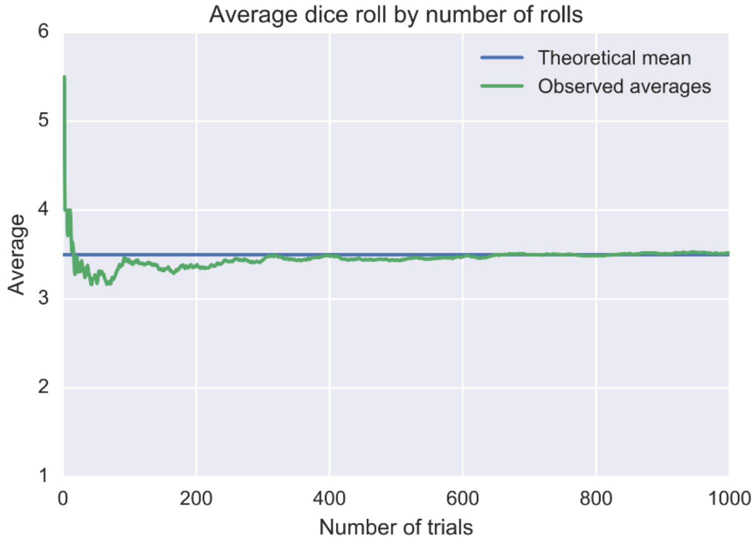
We now know the **sample mean** is an unbiased estimator, namely:

$$\mathbf{E}[\bar{X}_N] = \frac{1}{N}\sum_i \mathbf{E}[X_i] = \mathbf{E}[X]$$

But, expected value is not always high probability. Will we achieve the true mean?

$$\lim_{N\to\infty} \bar{X}_N \to \mathbf{E}[X]$$

Yes, with high probability



This is the **law of large numbers**

- Weak Law: Converges to mean with high probability
- Strong Law: Stronger notion of convergence (if variance is finite)

But what is the distribution of $\bar{X}_N$*?*

Central Limit Theorem (CLT)

Let $X_1, \ldots, X_N$ be iid with mean μ and variance σ^2 then $\bar{X}_N$ approaches a Normal distribution with mean μ and variance $\frac{\sigma^2}{N}$

$$\lim_{N \to \infty} \bar{X}_N \to \mathcal{N}\left(\mu, \frac{\sigma^2}{N}\right)$$

Alternatively written as,

$$\lim_{N \to \infty} \frac{\sqrt{N}}{\sigma}(\bar{X}_N - \mu) \to \mathcal{N}(0, 1)$$

Comments

- LLN says estimates $\bar{X}_N$ "pile up" near true mean, CLT says *how* they pile up
- Pretty remarkable since we make no assumption about how X_i are distributed
- Variance of X_i **must be finite**, i.e. $\sigma^2 < \infty$

Sample Mean

Example 2.1. *For the data set* $\{1, 2, 2, 2, 3, 3, 4, 4, 4, 5\}$, *we have* $n = 10$ *and the sum*

$$\begin{aligned} 1 + 2 + 2 + 2 + 3 + 3 + 4 + 4 + 4 + 5 &= 1n(1) + 2n(2) + 3n(3) + 4n(4) + 5n(5) \\ &= 1(1) + 2(3) + 3(2) + 4(3) + 5(1) = 30 \end{aligned}$$

Thus, $\bar{x} = 30/10 = 3$.

Sample Mean

Example 2.2. *For the data on the length in microns of wild type* Bacillus subtilis *data, we have*

length x	frequency $n(x)$	proportion $p(x)$	product $xp(x)$
1.5	18	0.090	0.135
2.0	71	0.355	0.710
2.5	48	0.240	0.600
3.0	37	0.185	0.555
3.5	16	0.080	0.280
4.0	6	0.030	0.120
4.5	4	0.020	0.090
sum	200	1	2.490

So the sample mean $\bar{x} = 2.49$.

Sample Mean

For any real-valued function h(x) we can compute the mean as,

$$\overline{h(x)} = \frac{1}{N}\sum_{i=1}^{N} h(x_i)$$

Note $\overline{h(x)} \neq h(\bar{x})$ in general.

Example Compute the average of the square of values,

{ 1, 2, 3, 4, 5, 5, 6 }

$$\overline{x^2} = \frac{1}{7}(1 + 2^2 + 3^3 + 4^2 + 2(5^2) + 6^2) \approx 16.57$$

Weighted Mean

In some cases we may weight data differently,

$$\sum_{i=1}^{N} w_i x_i \qquad \text{where} \qquad \sum_{i=1}^{N} w_i = 1 \qquad 0 \leq w_i \text{ for } i = 1, \ldots, N$$

For example, grades in a class:

$$\text{Grade} = 0.2 \cdot x_{\text{midterm}} + 0.2 \cdot x_{\text{final}} + 0.6 \cdot x_{\text{homework}}$$

Grading Breakdown

- Homework: 60%
- Midterm: 20%
- Final: 20%

Measuring Spread

We have seen estimates of spread via the sample variance,

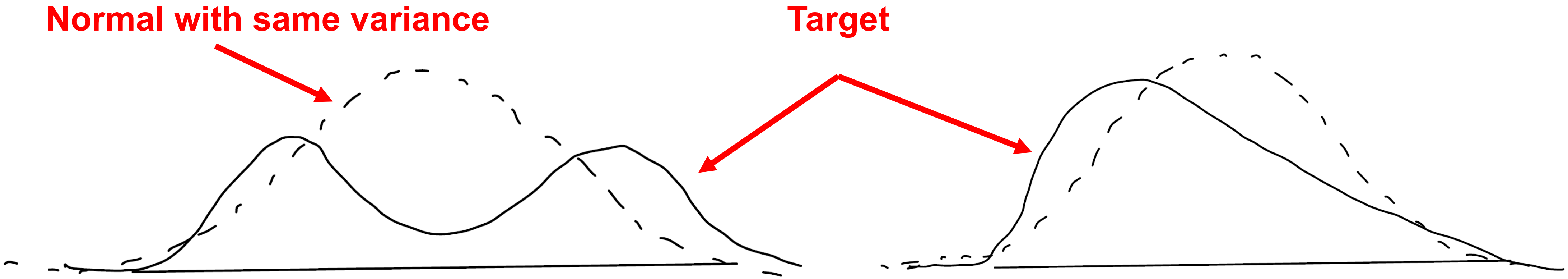
$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Biased

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Unbiased

Misleading estimate of spread for multimodal / skew distributions



Measuring Spread

Quartile divide data into 4 equally-sized bins,

- **1st Quartile** : Lowest 25% of data
- **2nd Quartile** : Median (lowest 50% of data)
- **3rd Quartile** : 75% of data is below 3rd quartile
- **4th Quartile** : All the data… not useful

Compute using `np.quantile()` :

```
x = np.random.rand(10) * 100
q = np.quantile(x, (0.25, 0.5, 0.75))
np.set_printoptions(precision=1)
print( "X: " , x )
print( "Q: " , q )
```

```
X:  [90.7 73.9 31.7  2.8 56.3 95.7 15.6 75.8  4.1 19.5]
Q:  [16.6 44.  75.3]
```

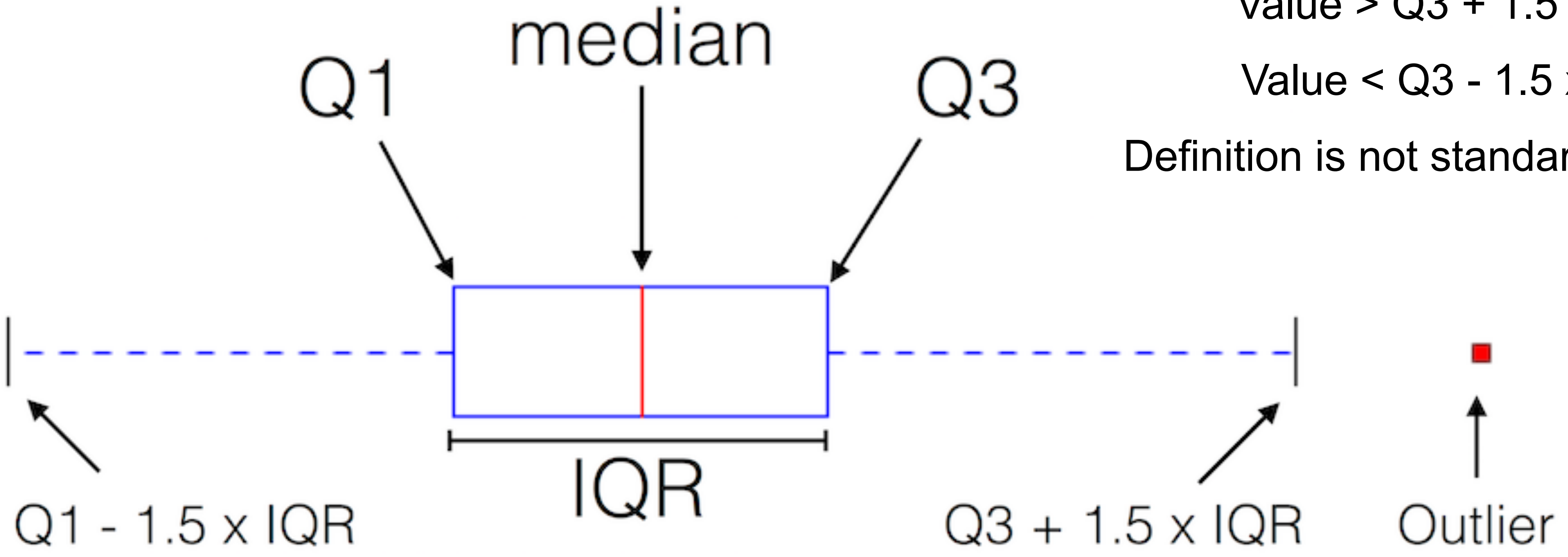
Box Plot

Outliers Are atypical data,

Value > Q3 + 1.5 x IQR

Value < Q3 - 1.5 x IQR

Definition is not standard.



Interquartile-Range (IQR) Measures interval containing 50% of data

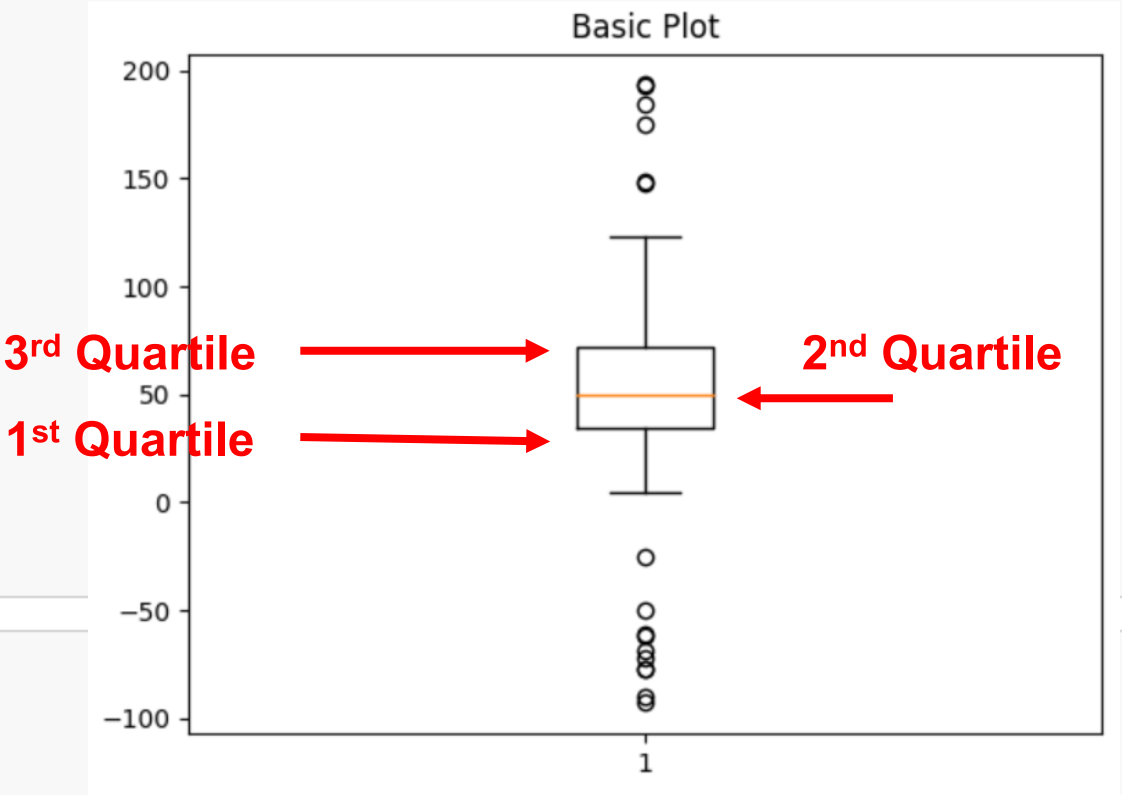
$$IQR = Q3 - Q1$$

Region of *typical* data

Box Plot

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

# Fixing random state for reproducibility
np.random.seed(19680801)

# fake up some data
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))
```

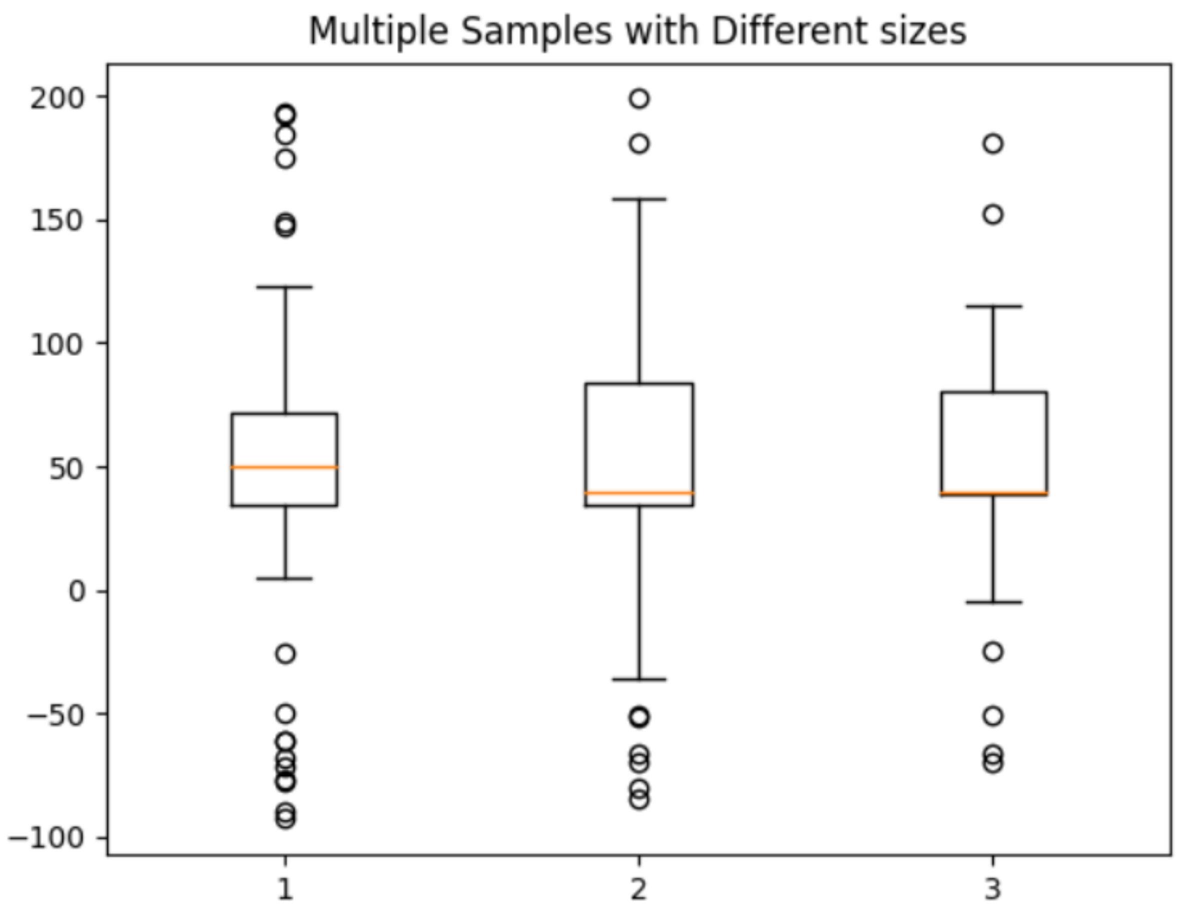
```python
fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)
```



Box Plot

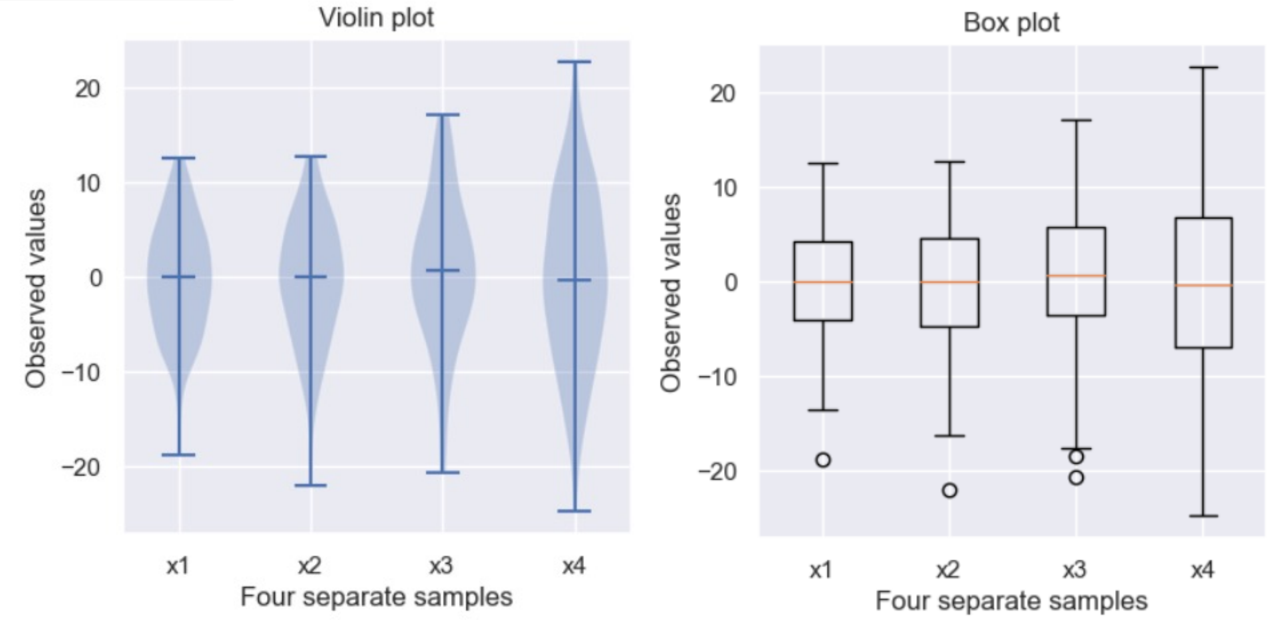
```
spread = np.random.rand(50) * 100
center = np.ones(25) * 40
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
d2 = np.concatenate((spread, center, flier_high, flier_low))
```

```
data = [data, d2, d2[::2]]
fig7, ax7 = plt.subplots()
ax7.set_title('Multiple Samples with Different sizes')
ax7.boxplot(data)

plt.show()
```



Violin Plot

```python
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(9, 4))

# Fixing random state for reproducibility
np.random.seed(19680801)

# generate some random test data
all_data = [np.random.normal(0, std, 100) for std in range(6, 10)]

# plot violin plot
axs[0].violinplot(all_data,
                  showmeans=False,
                  showmedians=True)
axs[0].set_title('Violin plot')

# plot box plot
axs[1].boxplot(all_data)
axs[1].set_title('Box plot')

# adding horizontal grid lines
for ax in axs:
    ax.yaxis.grid(True)
    ax.set_xticks([y + 1 for y in range(len(all_data))],
                  labels=['x1', 'x2', 'x3', 'x4'])
    ax.set_xlabel('Four separate samples')
    ax.set_ylabel('Observed values')
```



Measuring Spread

For nonnegative numbers we can look at the **coefficient of variation**,

$$\mathrm{cv} = \frac{s}{\bar{x}}$$

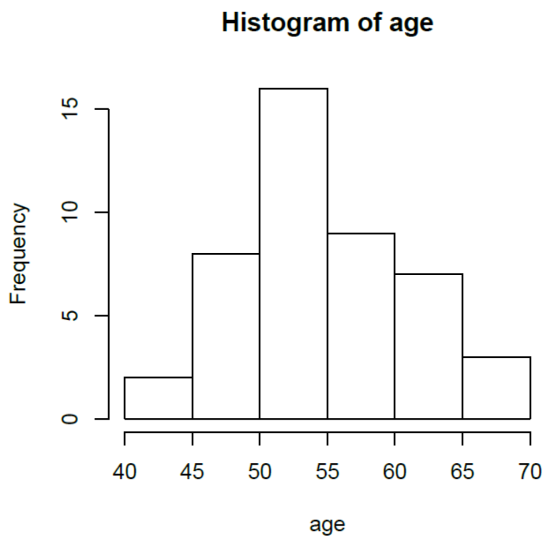
where $$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \quad \text{and} \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

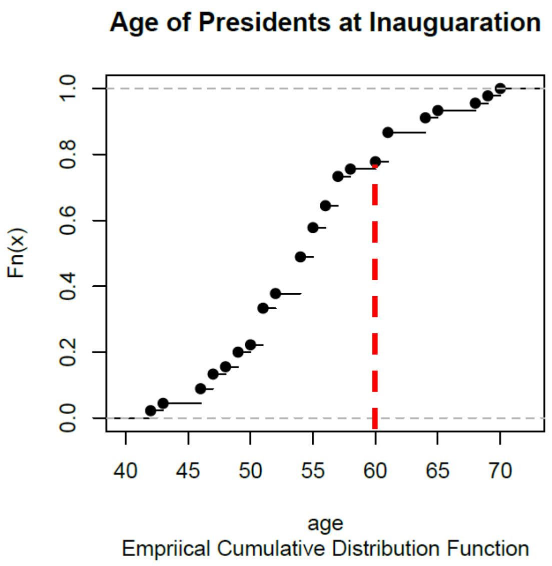
- It is a *pure number* – it has no units
- It represents spread relative to the mean

Question Why would we want to compute this?

Quantile / Percentile

Question Is 60yrs old for a US president? Why or why not?





Empirical CDF for each x gives P(X<x),

$$F_n(x) = \frac{1}{n}\#(\text{observations less than or equal to x})$$

Compute probability of being <60,

$$F_n(60) \approx 0.8$$

0.8 Quantile or 80th Percentile → About 80% of presidents younger than 60

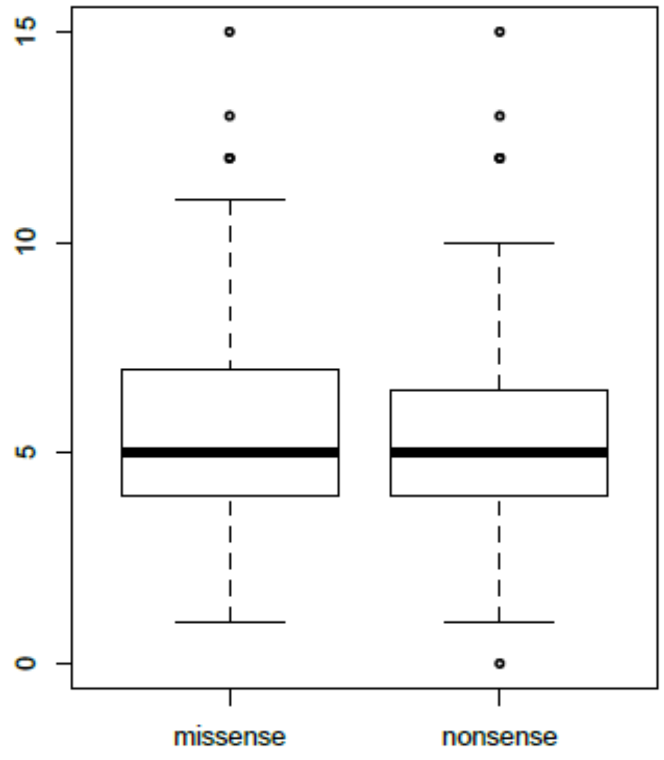
Quantile-Quantile Plot

Plot quantiles of two variables against each other…

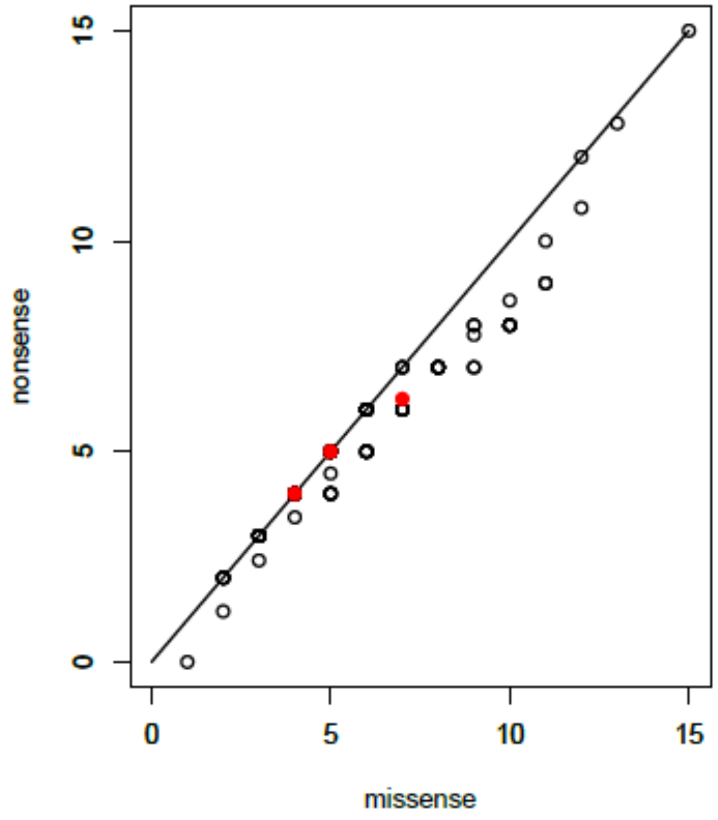
Variables with similar distributions will fall along a 45-degree (slope 1) line

In Q-Q plot correlation = distribution similarity

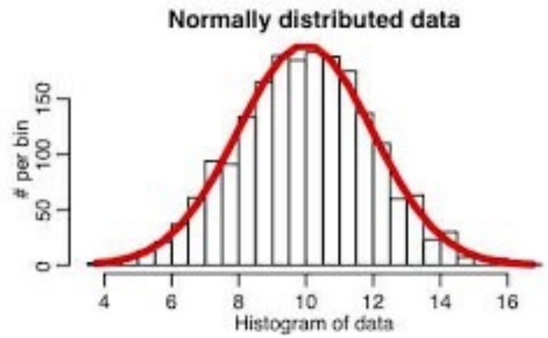
Box Plot

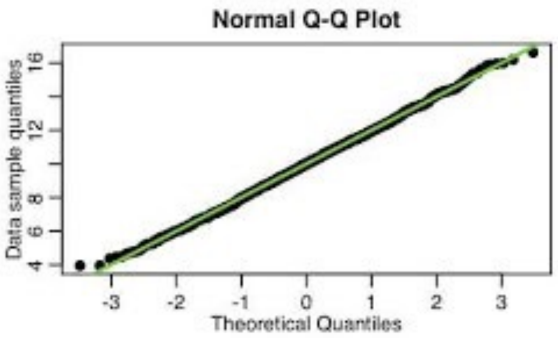


Q-Q Plot



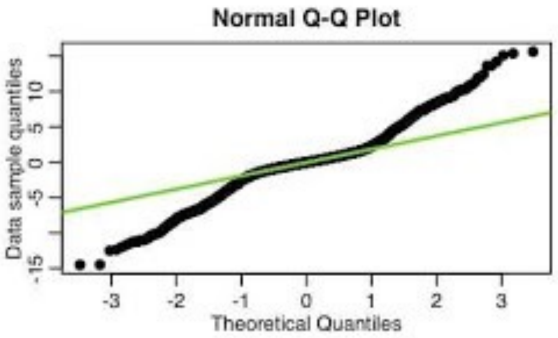
Interpreting Q-Q Plots

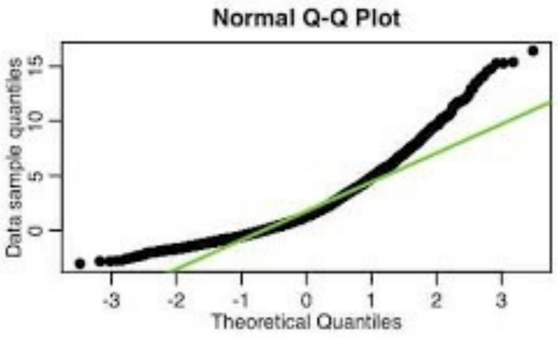




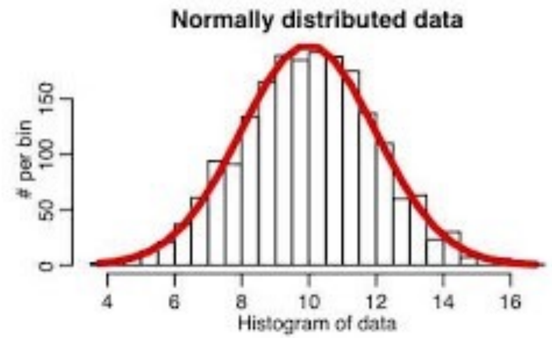
Plot against theoretical quantiles to check model fit

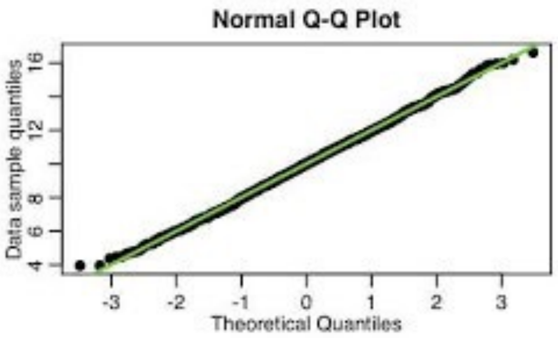
Good Fit





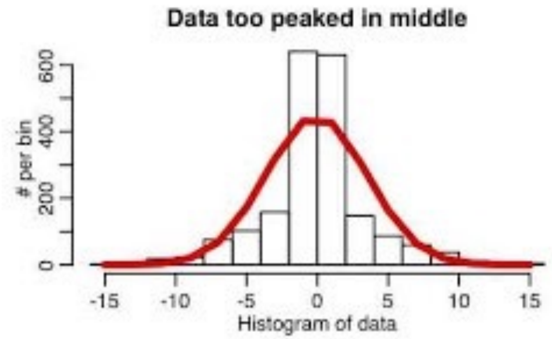
Interpreting Q-Q Plots

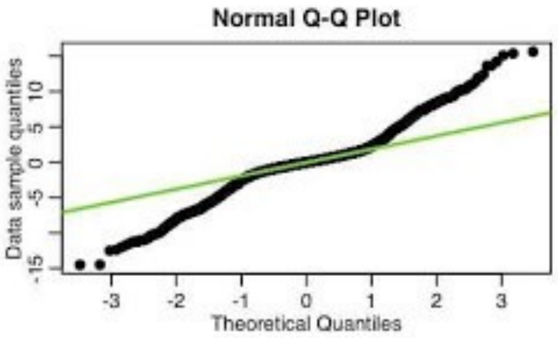




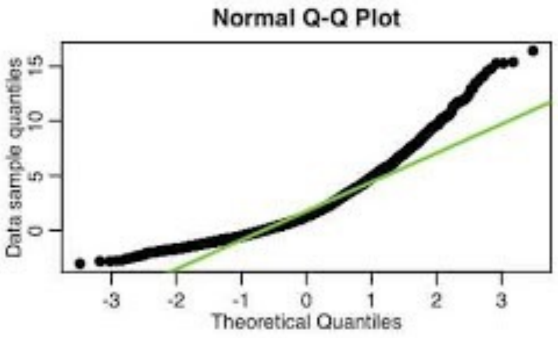
Plot against theoretical quantiles to check model fit

Good Fit

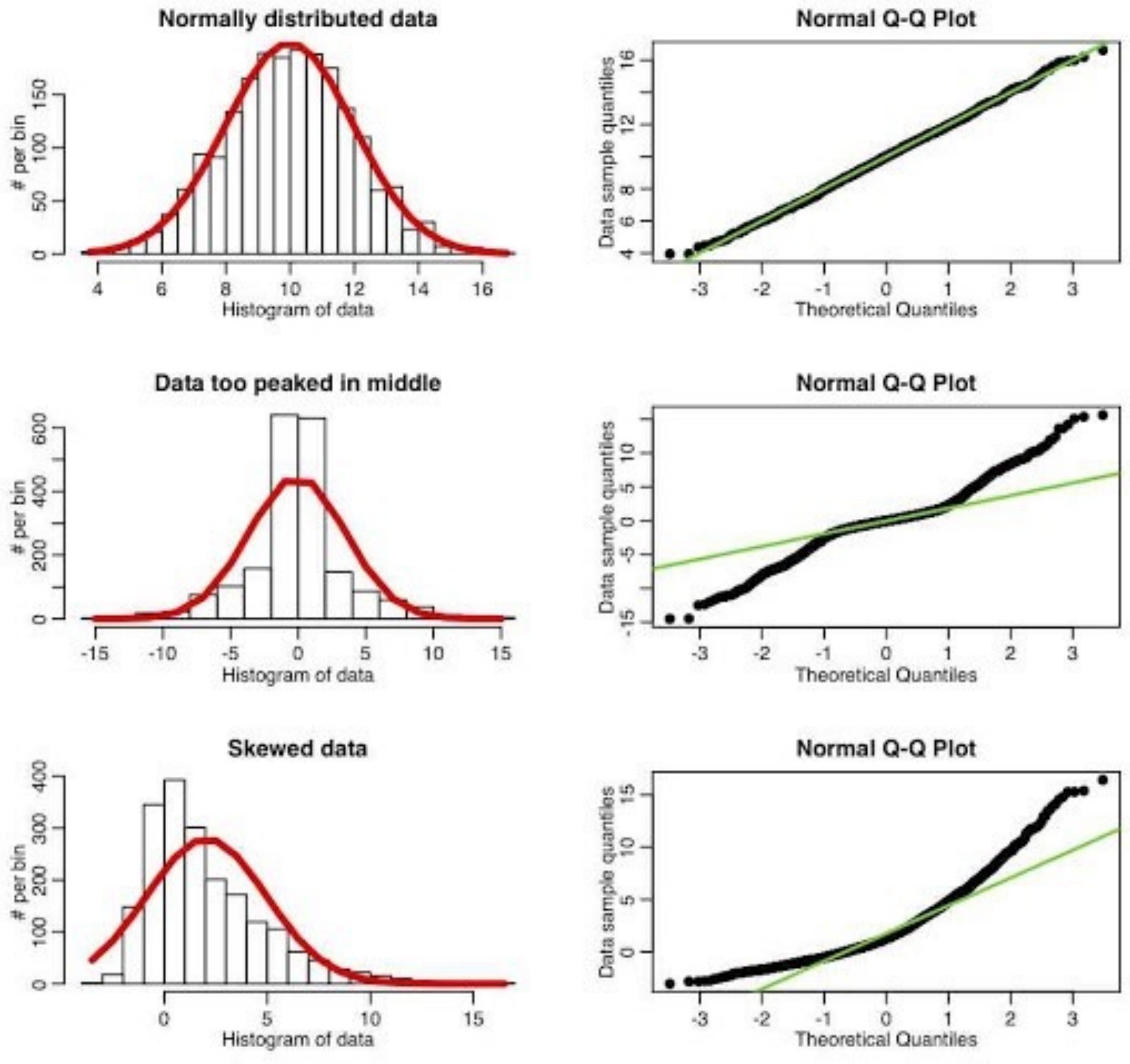




Fat Tails



Interpreting Q-Q Plots



Plot against theoretical quantiles to check model fit

Good Fit

Fat Tails

Positive Skew

Outline

- Data Visualization
- Data Summarization
- Python data tools

SciPy

Python-based ecosystem for math, science and engineering.

As usual, install with Anaconda:

```
> conda install scipy
```

Or with PyPI:

```
> pip install scipy
```

SciPy includes some libraries that you are already familiar with:

NumPy matplotlib

Other useful summary statistics:

moment(a[, moment, axis, nan_policy])	Calculate the nth moment about the mean for a sample.
trim_mean(a, proportiontocut[, axis])	Return mean of array after trimming distribution from both tails.
iqr(x[, axis, rng, scale, nan_policy, …])	Compute the interquartile range of the data along the specified axis.
bootstrap(data, statistic, *[, vectorized, …])	Compute a two-sided bootstrap confidence interval of a statistic.
variation(a[, axis, nan_policy, ddof])	Compute the coefficient of variation.

…

Trimmed Mean

Mean is not robust to outliers…

```
>>> x = np.arange(20)
>>> stats.trim_mean(x, 0.1)
9.5
>>> x2 = x.reshape(5, 4)
>>> x2
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
>>> stats.trim_mean(x2, 0.25)
array([  8.,   9.,  10.,  11.])
>>> stats.trim_mean(x2, 0.25, axis=1)
array([  1.5,   5.5,   9.5,  13.5,  17.5])
```

…trimmed mean "trims" % of either end of data (e.g. 0.1 → 10%) before computing the mean value

Anscomb's Quartet : The Data

Four distinct datasets of paired variables X and Y…

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

[Source: https://www.geeksforgeeks.org/anscombes-quartet/]

Anscomb’s Quartet : Summary Statistics

```python
# Import the csv file
df = pd.read_csv("anscombe.csv")

# Convert pandas dataframe into pandas series
list1 = df['x1']
list2 = df['y1']

# Calculating mean for x1
print('%.1f' % statistics.mean(list1))

# Calculating standard deviation for x1
print('%.2f' % statistics.stdev(list1))

# Calculating mean for y1
print('%.1f' % statistics.mean(list2))

# Calculating standard deviation for y1
print('%.2f' % statistics.stdev(list2))

# Calculating pearson correlation
corr, _ = pearsonr(list1, list2)
print('%.3f' % corr)

# Similarly calculate for the other 3 samples

# This code is contributed by Amiya Rout
```

Start by computing summary statistics, e.g. Dataset 1:

Mean X1: 9.0

STDEV X1: 3.32

Mean Y1: 7.5

STDEV Y1: 2.03

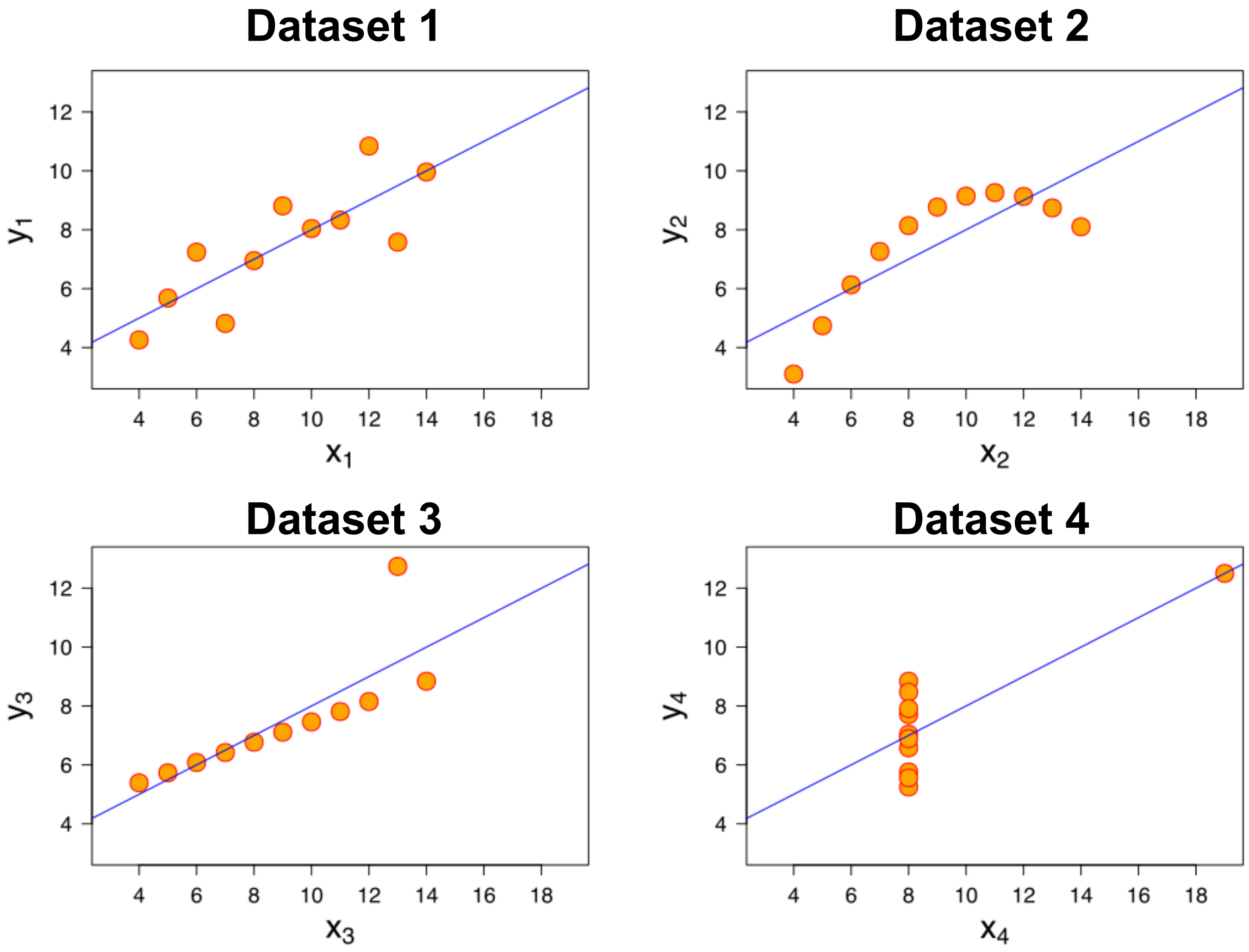
Correlation: 0.816

Actually, **all datasets have the same statistics…**

Question What can we conclude about these data? Are they the same?

[Source: https://www.geeksforgeeks.org/anscombes-quartet/]

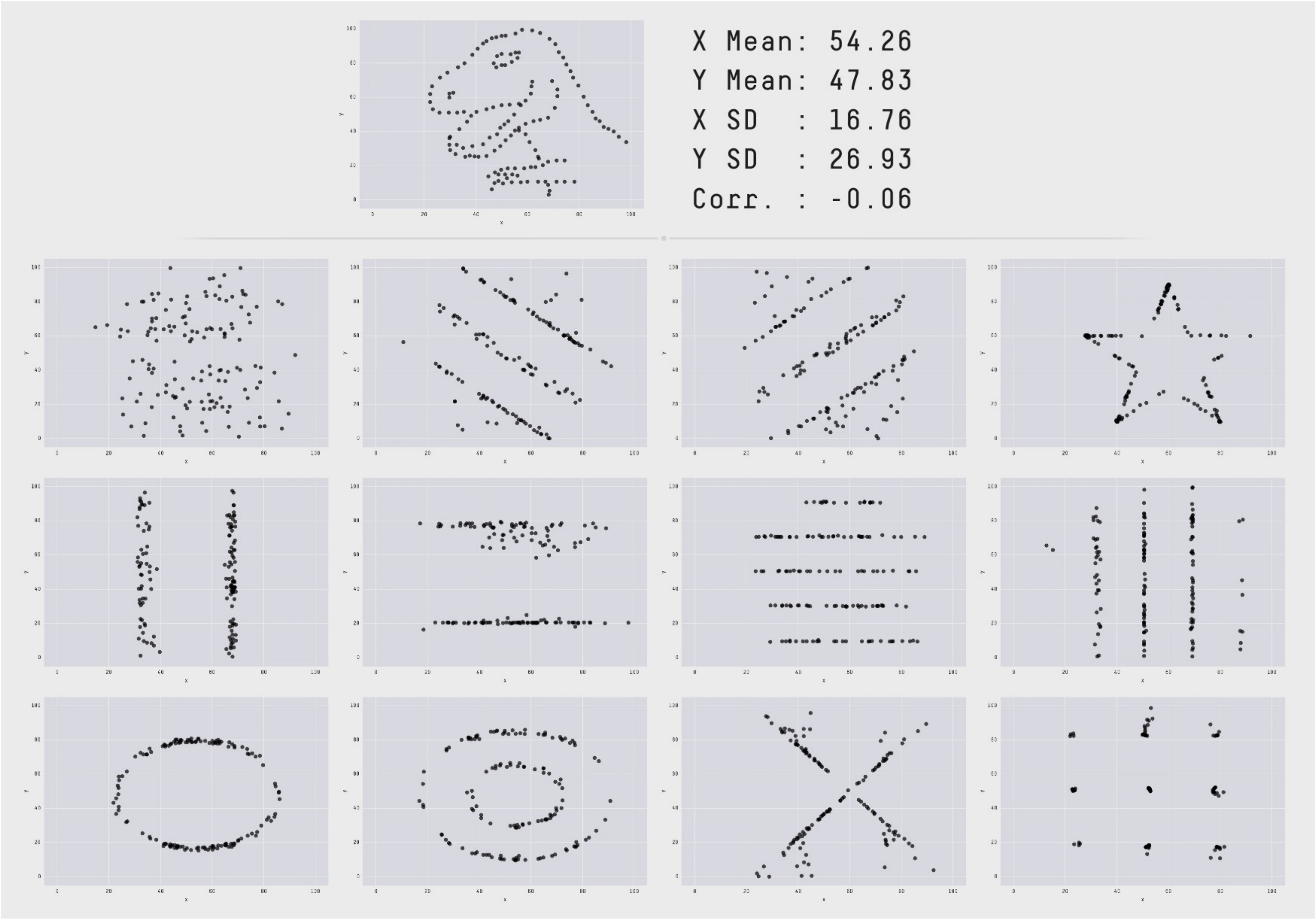
Anscomb's Quartet : Visualization



Visualizing data clearly indicates that these are *very different* datasets…

…this highlights the **importance of visualizing data**

Datasaurus



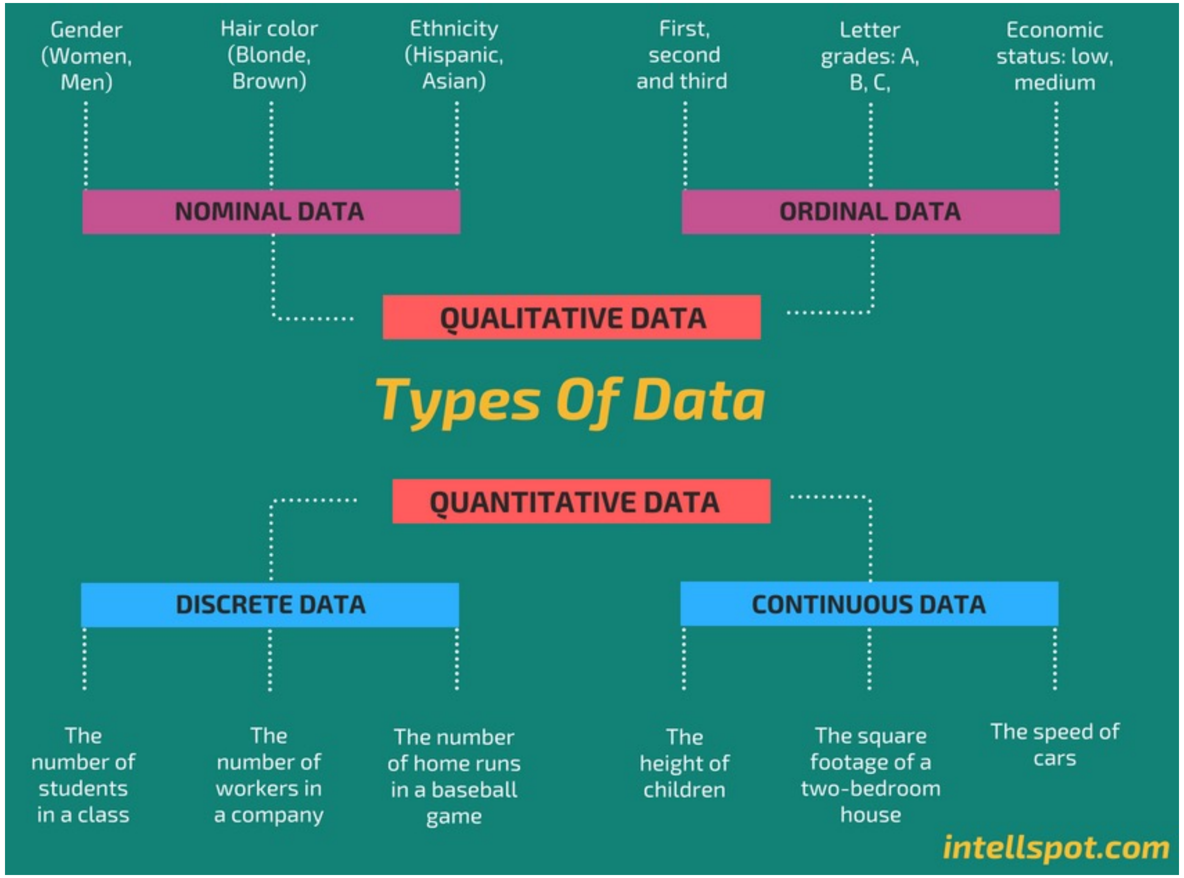
13 datasets that all have the same summary statistics, but look very different in simple visualizations

Can be very difficult to see differences in high dimensions, however

[Source: [Alberto Cairo]()]

Types of Data

Data come in many forms, each requiring different approaches & models

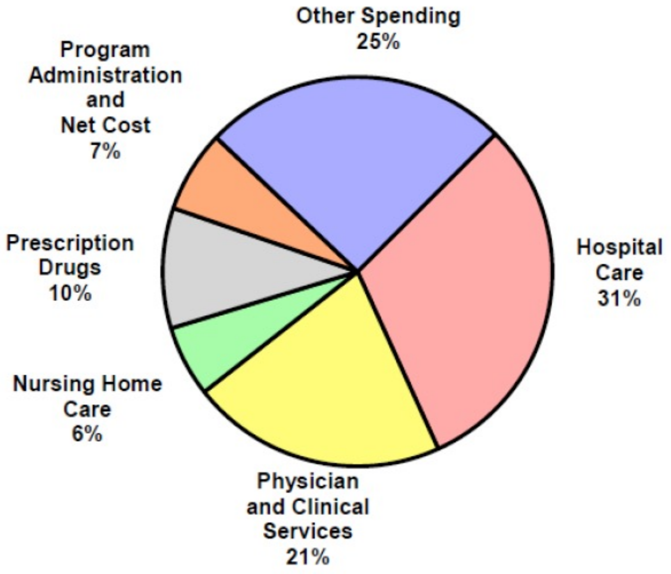


Qualitative or **categorical** : partition data into classes (flexible but imprecise)

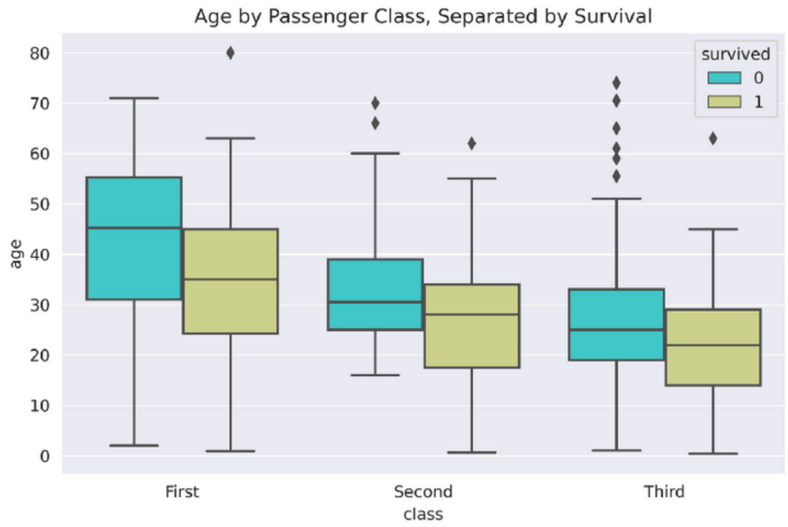
Quantitative : can perform mathematical operations (e.g. addition, subtraction, ordering)

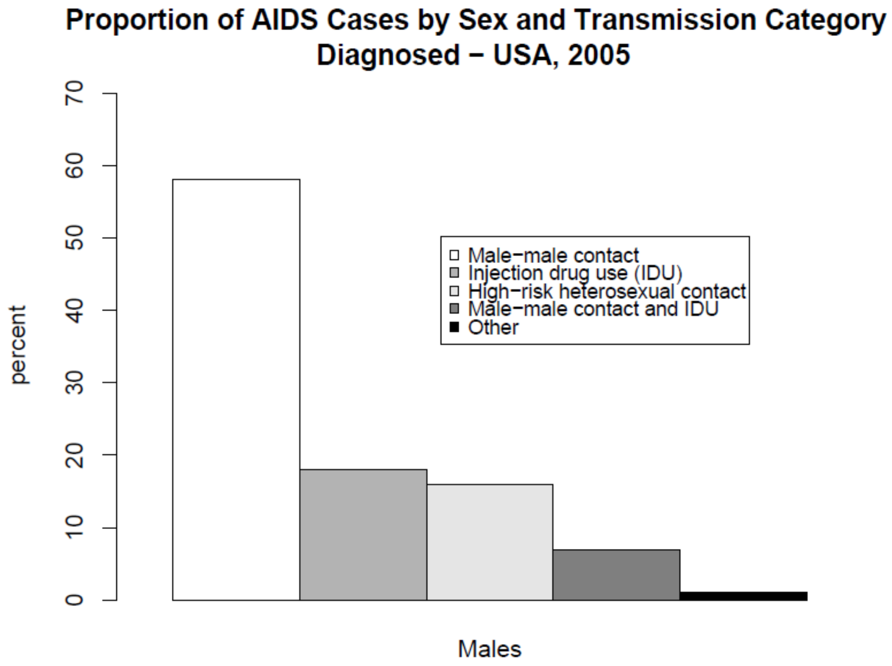
*We often refer to different types of data as **variables***

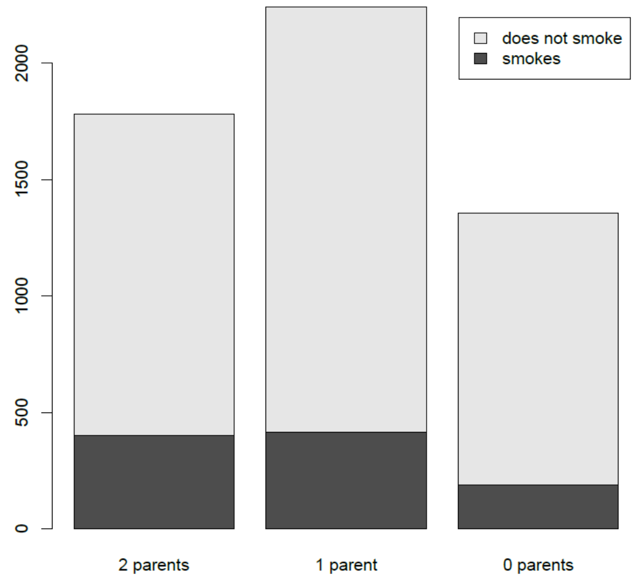
Visualizing Categorical Variables



	student smokes	student does not smoke	total
2 parents smoke	400	1380	1780
1 parent smokes	416	1823	2239
0 parents smoke	188	1168	1356
total	1004	4371	5375



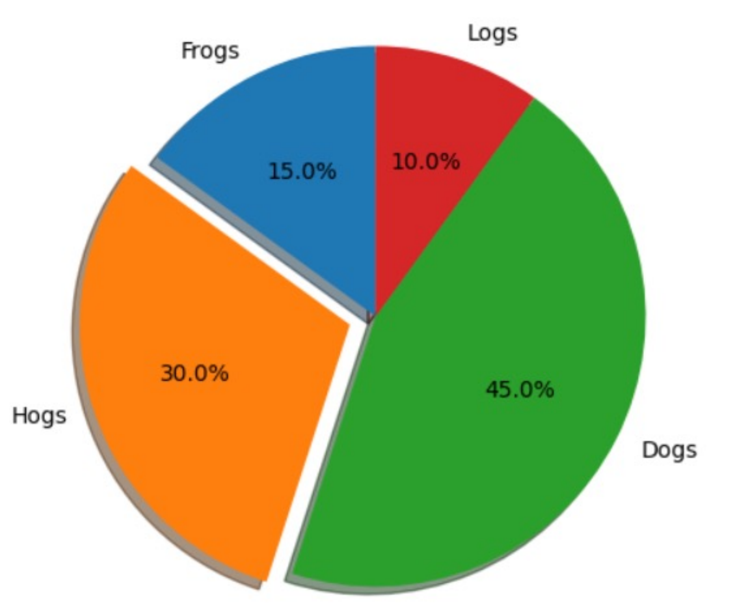




Pie Chart

Circular chart divided into sectors, illustrating relative magnitudes in frequencies or percent. In a pie chart, the area is proportional to the quantity it represents.

```python
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

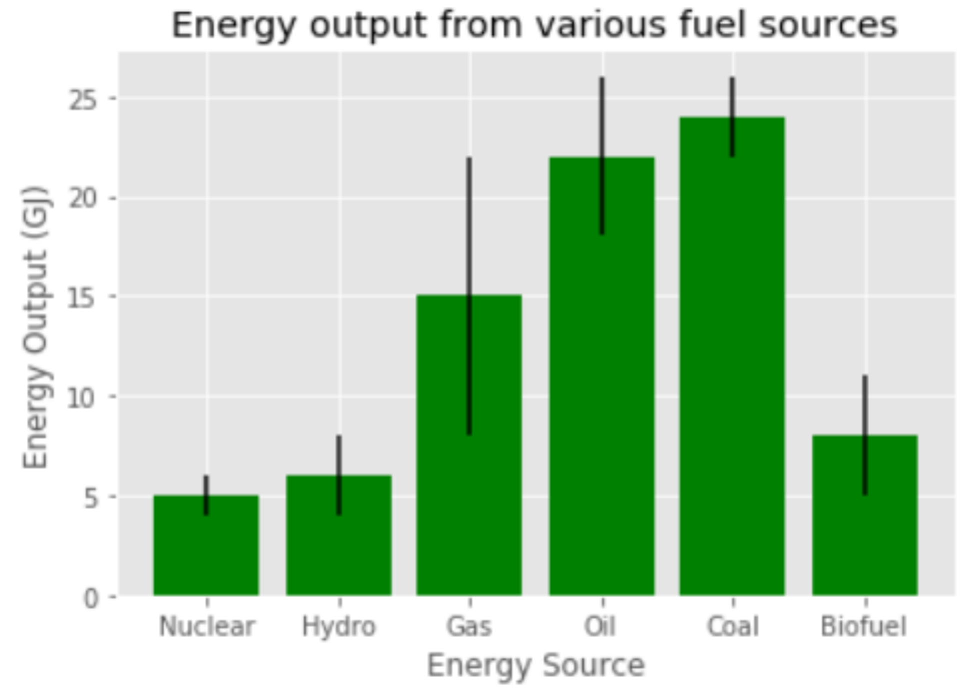


Bar Chart

We perceive differences in height / length better than area…

`plt.bar()`

```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
variance = [1, 2, 7, 4, 2, 3]

x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, energy, color='green', yerr=variance)
plt.xlabel("Energy Source")
plt.ylabel("Energy Output (GJ)")
plt.title("Energy output from various fuel sources")

plt.xticks(x_pos, x)

plt.show()
```



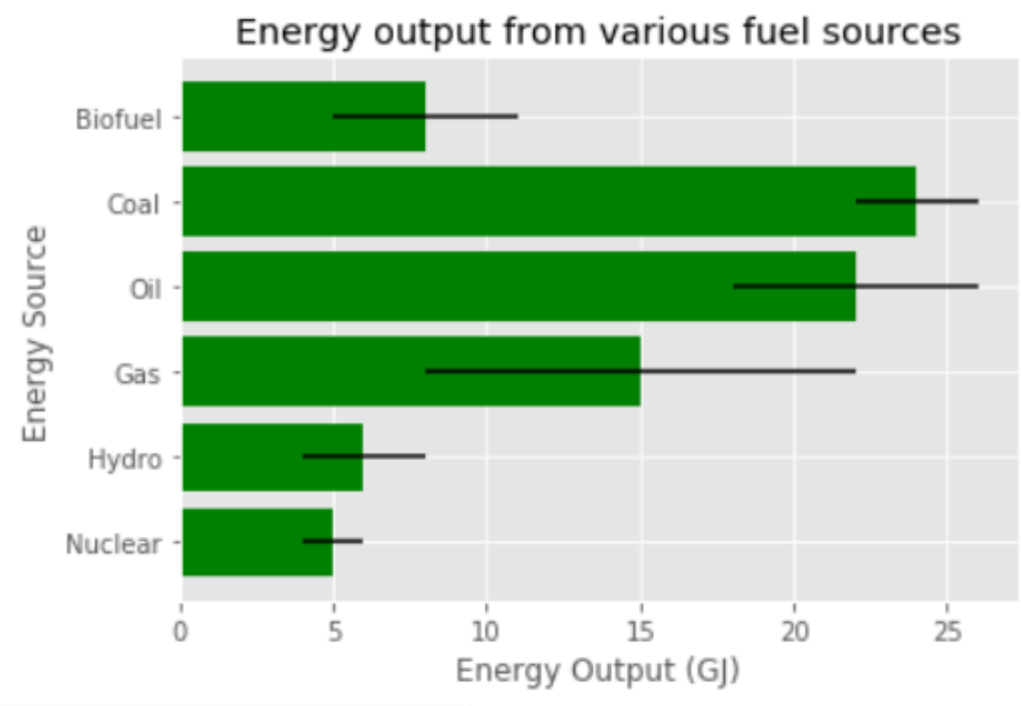
[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

Bar Chart

Don't make readers tilt their heads, consider rotating for readability…

`plt.barh()`

```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
variance = [1, 2, 7, 4, 2, 3]

x_pos = [i for i, _ in enumerate(x)]

plt.barh(x_pos, energy, color='green', xerr=variance)
plt.ylabel("Energy Source")
plt.xlabel("Energy Output (GJ)")
plt.title("Energy output from various fuel sources")

plt.yticks(x_pos, x)

plt.show()
```



[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

Bar Chart

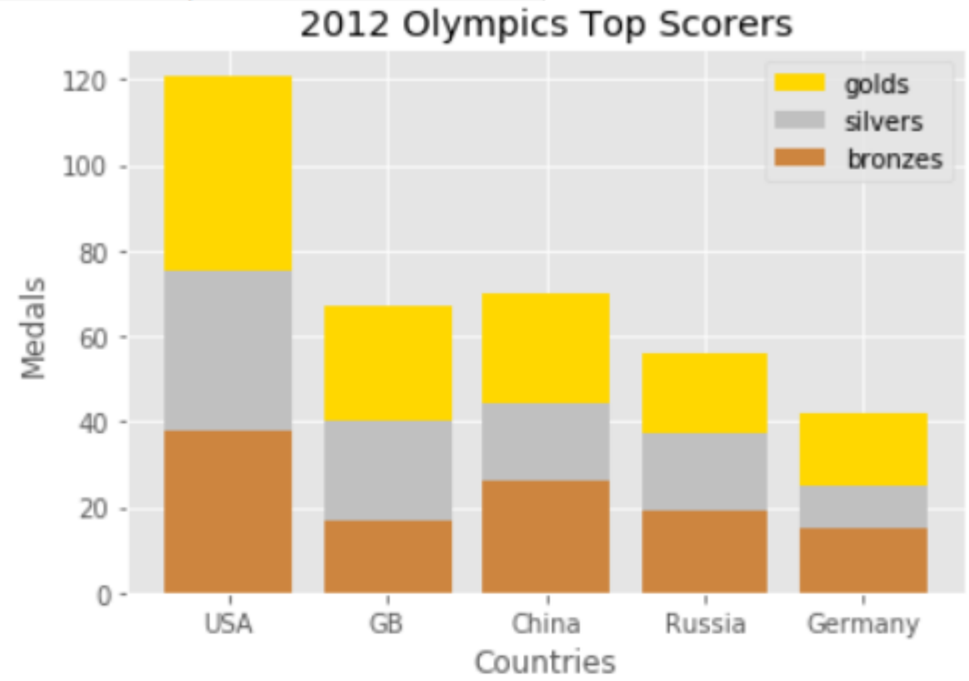
Multiple groups of bars…

```
import numpy as np

N = 5
men_means = (20, 35, 30, 35, 27)
women_means = (25, 32, 34, 20, 25)

ind = np.arange(N)
width = 0.35
plt.bar(ind, men_means, width, label='Men')
plt.bar(ind + width, women_means, width,
    label='Women')

plt.ylabel('Scores')
plt.title('Scores by group and gender')

plt.xticks(ind + width / 2, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.legend(loc='best')
plt.show()
```

[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

Stacked Bar Chart

```
countries = ['USA', 'GB', 'China', 'Russia', 'Germany']
bronzes = np.array([38, 17, 26, 19, 15])
silvers = np.array([37, 23, 18, 18, 10])
golds = np.array([46, 27, 26, 19, 17])
ind = [x for x, _ in enumerate(countries)]

plt.bar(ind, golds, width=0.8, label='golds', color='gold', bottom=silvers+bronzes)
plt.bar(ind, silvers, width=0.8, label='silvers', color='silver', bottom=bronzes)
plt.bar(ind, bronzes, width=0.8, label='bronzes', color='#CD853F')

plt.xticks(ind, countries)
plt.ylabel("Medals")
plt.xlabel("Countries")
plt.legend(loc="upper right")
plt.title("2012 Olympics Top Scorers")

plt.show()
```



[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

Two-Way Table

Also called contingency table or cross tabulation table…

Frequency

	student smokes	student does not smoke	total
2 parents smoke	400	1380	1780
1 parent smokes	416	1823	2239
0 parents smoke	188	1168	1356
total	1004	4371	5375

Two-Way Table

Also called <u>contingency table</u> or <u>cross tabulation table</u>…

Relative Frequency

Column Variable

	student smokes	student does not smoke	total
2 parents smoke	7.4%	25.7%	33.1%
1 parent smokes	7.7%	33.9%	41.7%
0 parents smoke	3.5%	21.8%	25.2%
total	18.7%	81.3%	100%

Marginal Distribution Of Row Variable

Row Variable

Marginal Distribution Of Column Variable

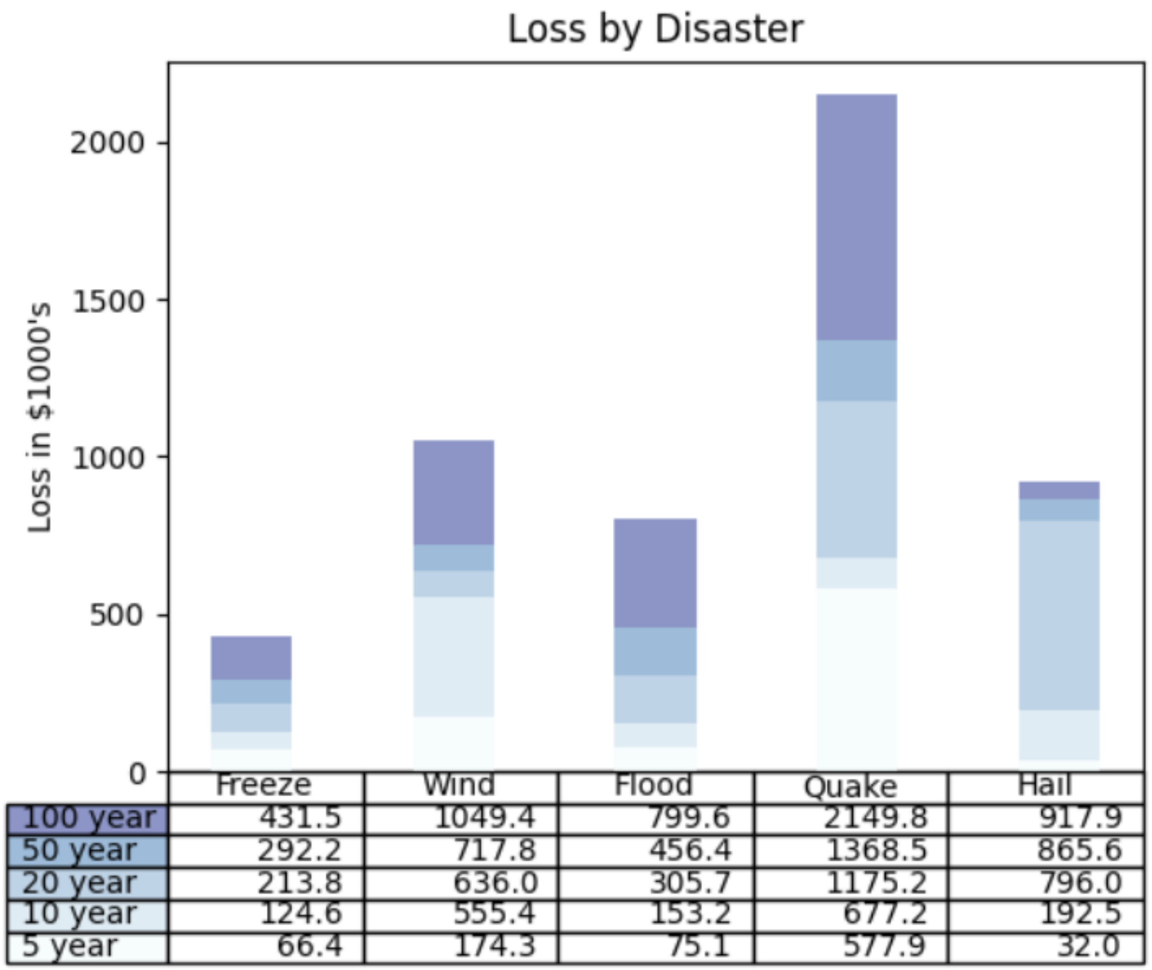
Joint Distribution

Two-Way Table

```python
data = [[ 66386, 174296,  75131, 577908,  32015],
        [ 58230, 381139,  78045,  99308, 160454],
        [ 89135,  80552, 152558, 497981, 603535],
        [ 78415,  81858, 150656, 193263,  69638],
        [139361, 331509, 343164, 781380,  52269]]

columns = ('Freeze', 'Wind', 'Flood', 'Quake', 'Hail')
rows = ['%d year' % x for x in (100, 50, 20, 10, 5)]
colors = plt.cm.BuPu(np.linspace(0, 0.5, len(rows)))
```

```python
the_table = plt.table(cellText=cell_text,
                      rowLabels=rows,
                      rowColours=colors,
                      colLabels=columns,
                      loc='bottom')
```

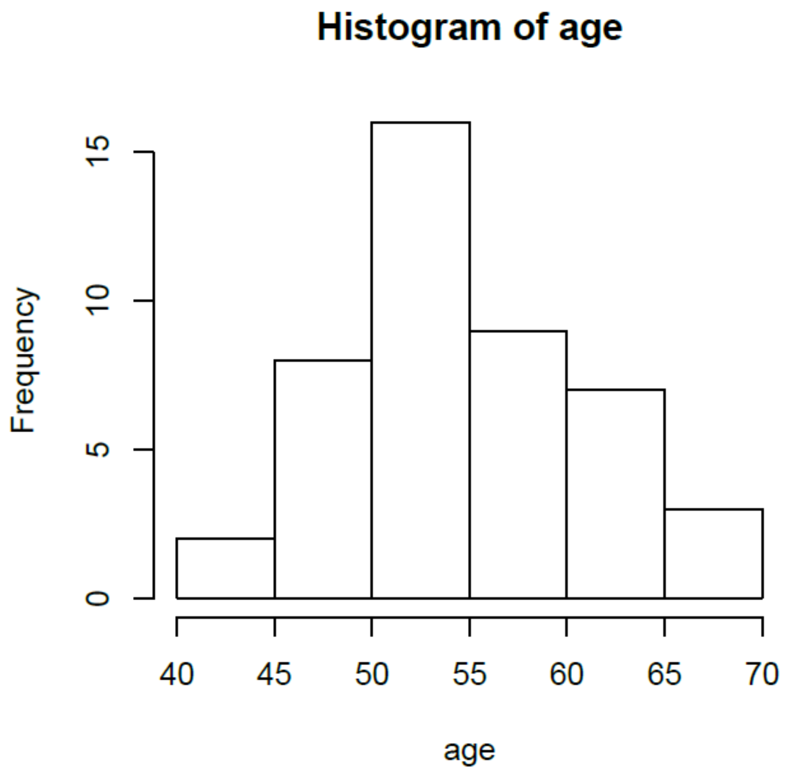
Adding stacked bars requires more steps, full code here: *https://matplotlib.org/stable/gallery/misc/table_demo.html*

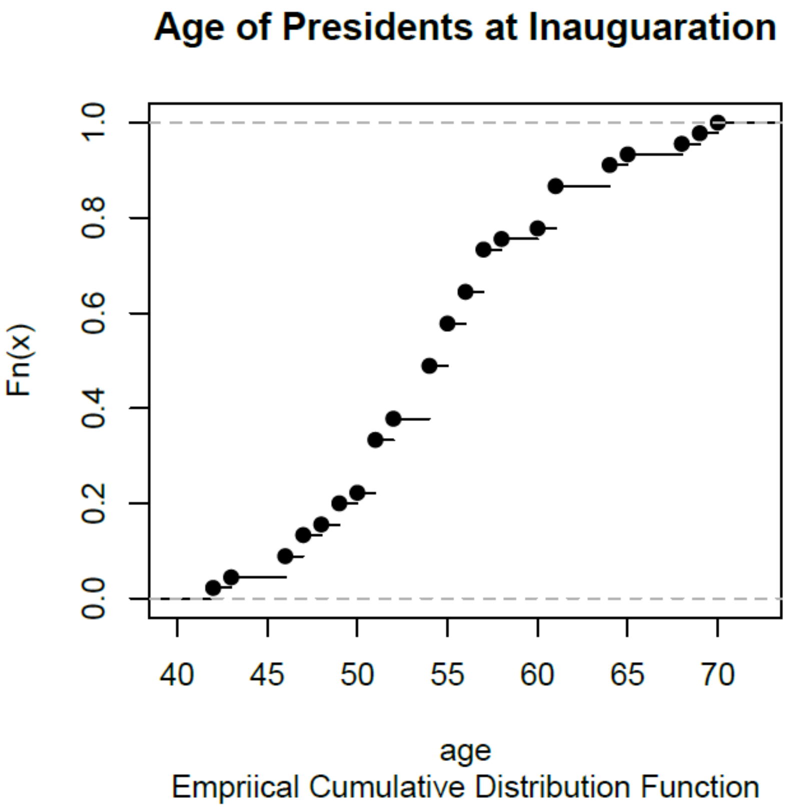


	Freeze	Wind	Flood	Quake	Hail
100 year	431.5	1049.4	799.6	2149.8	917.9
50 year	292.2	717.8	456.4	1368.5	865.6
20 year	213.8	636.0	305.7	1175.2	796.0
10 year	124.6	555.4	153.2	677.2	192.5
5 year	66.4	174.3	75.1	577.9	32.0

Histogram

Empirical approximation of (quantitative) data generating distribution



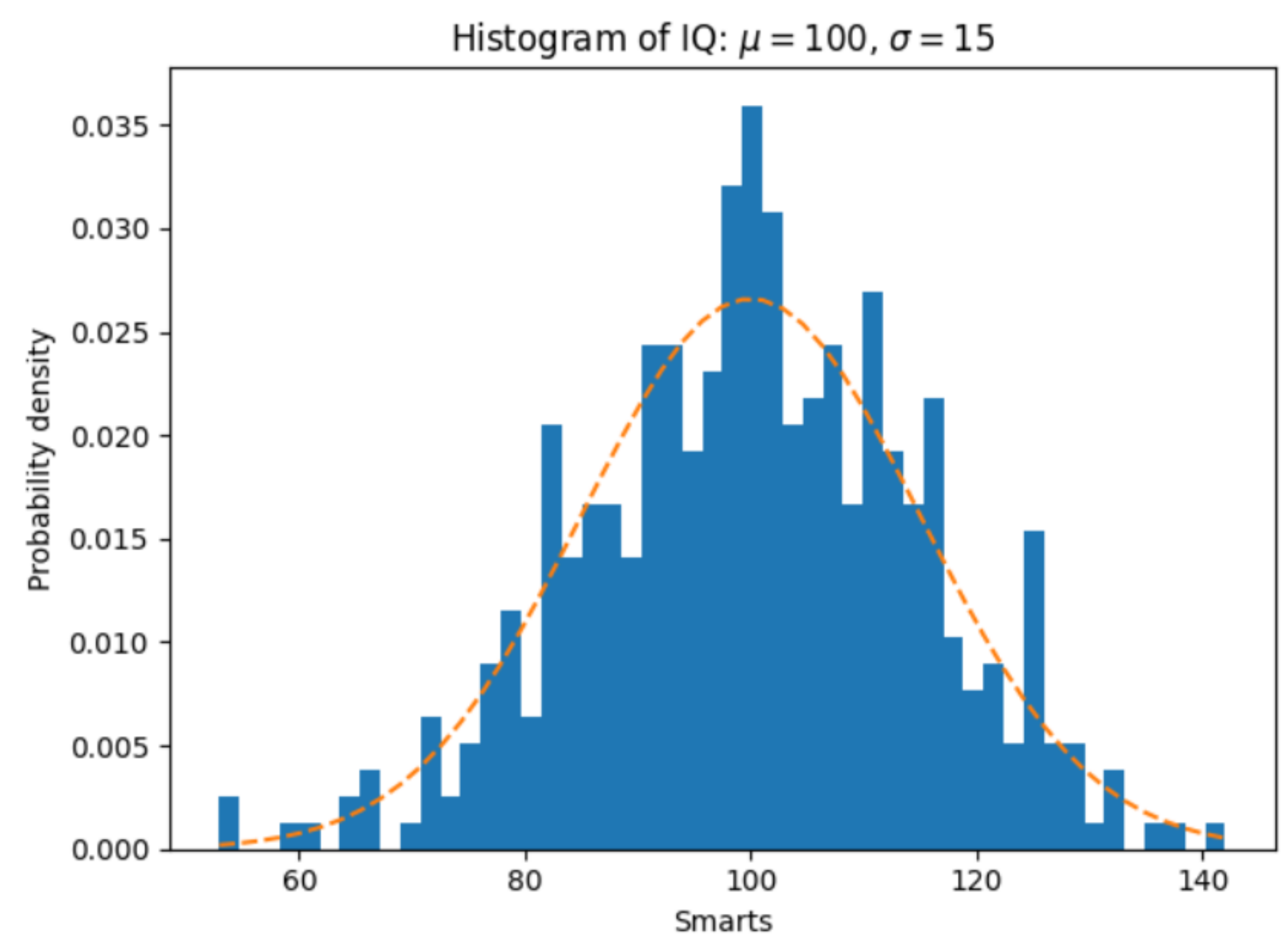


Empirical CDF for each x gives P(X<x),

$$F_n(x) = \frac{1}{n}\#(\text{observations less than or equal to x})$$

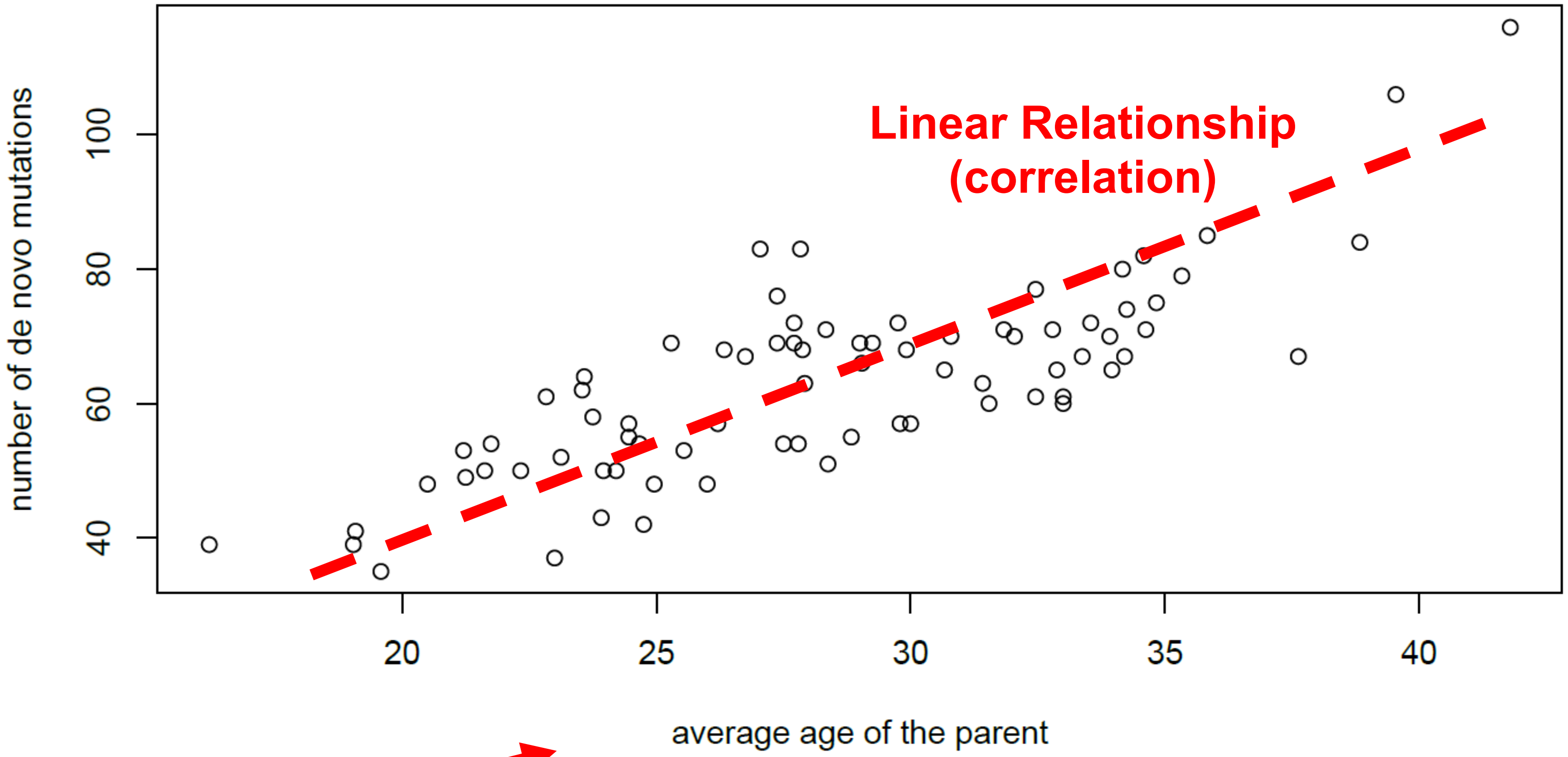
Histogram

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(19680801)

# example data
mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(437)

num_bins = 50

fig, ax = plt.subplots()

# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=True)

# add a 'best fit' line
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
     np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
ax.set_title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```



Scatterplot

Compares relationship between two quantitative variables…



Target (or response) Variable

Explanatory Variable

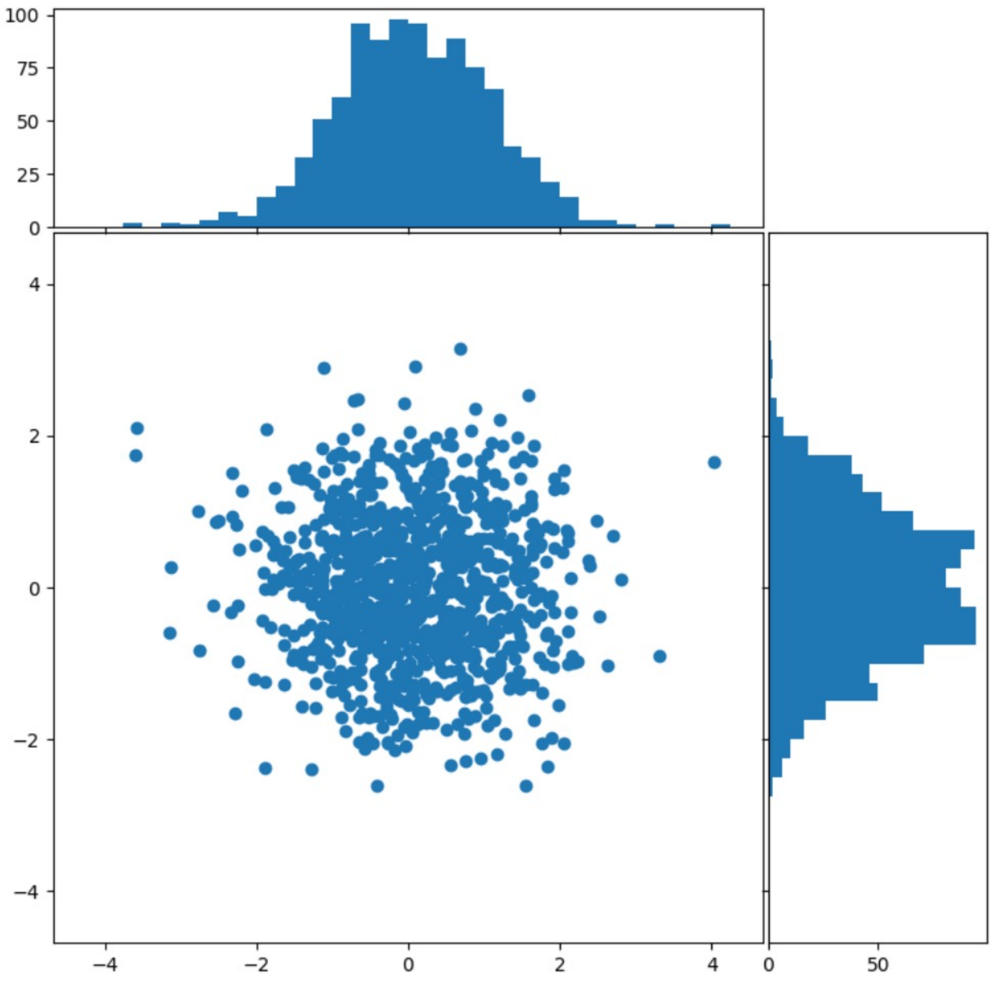
Also: predictor, descriptor, input

Relationship can also be:

- Nonlinear (e.g. “curvy”)
- Clustered or grouped

Scatterplot

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

# Fixing random state for reproducibility
np.random.seed(19680801)

# some random data
x = np.random.randn(1000)
y = np.random.randn(1000)


def scatter_hist(x, y, ax, ax_histx, ax_histy):
    # no labels
    ax_histx.tick_params(axis="x", labelbottom=False)
    ax_histy.tick_params(axis="y", labelleft=False)

    # the scatter plot:
    ax.scatter(x, y)

    # now determine nice limits by hand:
    binwidth = 0.25
    xymax = max(np.max(np.abs(x)), np.max(np.abs(y)))
    lim = (int(xymax/binwidth) + 1) * binwidth

    bins = np.arange(-lim, lim + binwidth, binwidth)
    ax_histx.hist(x, bins=bins)
    ax_histy.hist(y, bins=bins, orientation='horizontal')
```

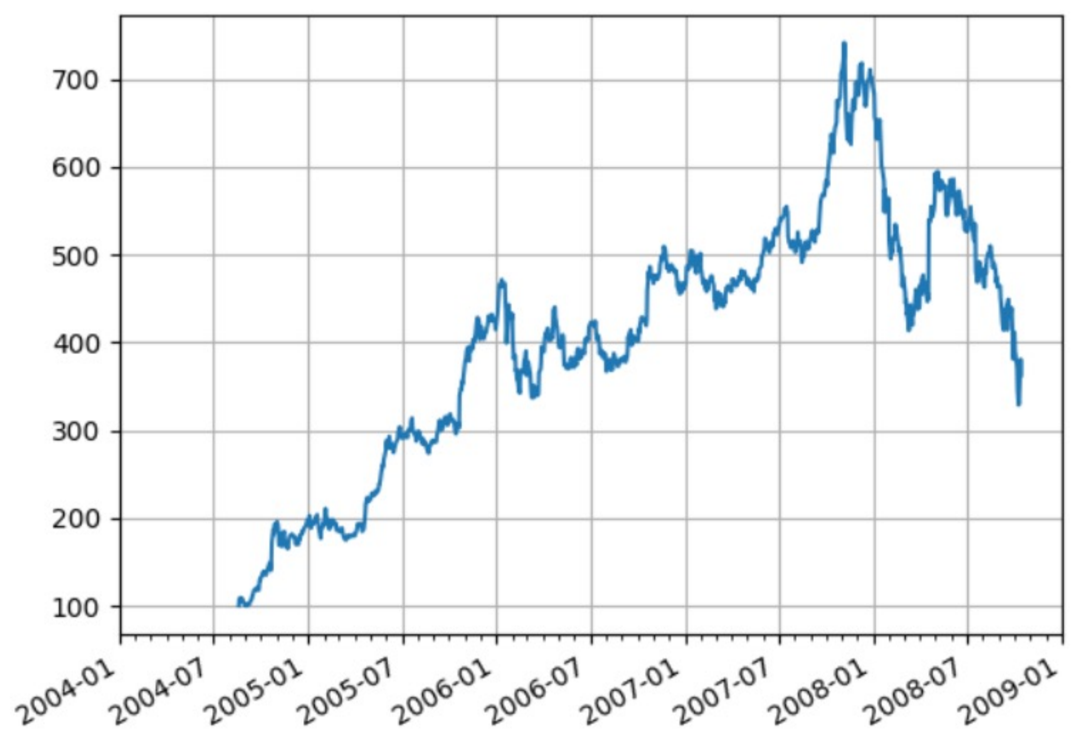


Full Code:
https://matplotlib.org/stable/gallery/lines_bars_and_markers/scatter_hist.html

Timeseries

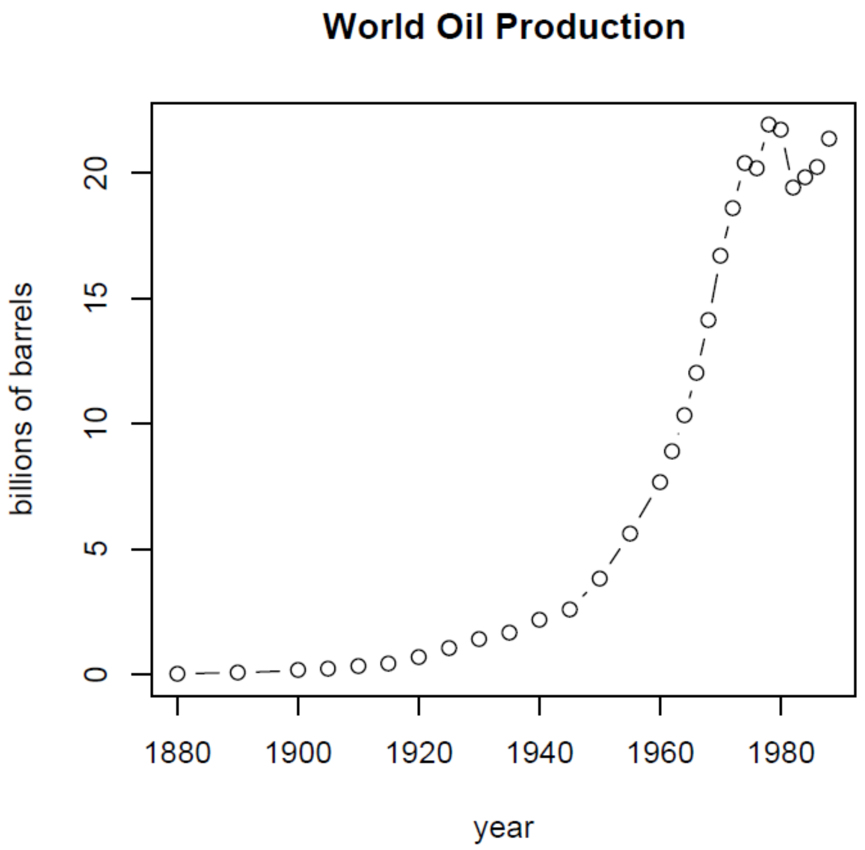
```python
fig, ax = plt.subplots()
ax.plot('date', 'adj_close', data=data)

# Major ticks every 6 months.
fmt_half_year = mdates.MonthLocator(interval=6)
ax.xaxis.set_major_locator(fmt_half_year)

# Minor ticks every month.
fmt_month = mdates.MonthLocator()
ax.xaxis.set_minor_locator(fmt_month)

# Text in the x axis will be displayed in 'YYYY-mm' format.
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))

# Round to nearest years.
datemin = np.datetime64(data['date'][0], 'Y')
datemax = np.datetime64(data['date'][-1], 'Y') + np.timedelta64(1, 'Y')
ax.set_xlim(datemin, datemax)

# Format the coords message box, i.e. the numbers displayed as the cursor moves
# across the axes within the interactive GUI.
ax.format_xdata = mdates.DateFormatter('%Y-%m')
ax.format_ydata = lambda x: f'${x:.2f}'  # Format the price.
ax.grid(True)

# Rotates and right aligns the x labels, and moves the bottom of the
# axes up to make room for them.
fig.autofmt_xdate()

plt.show()
```

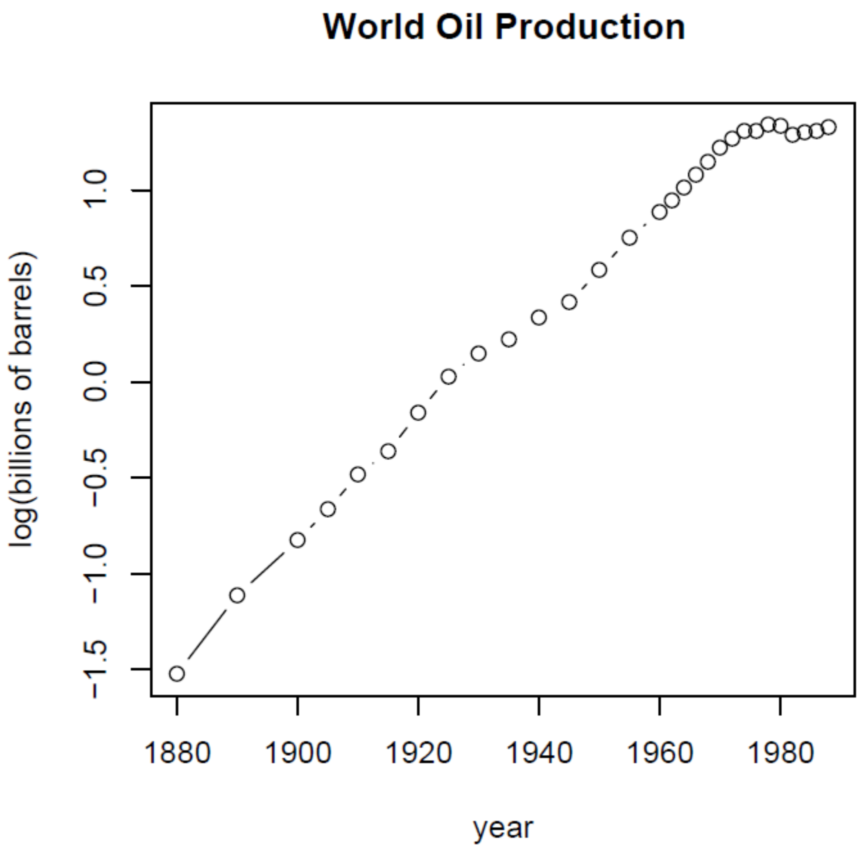
Data follow an explicit ordering



Logarithm Scale

Changing limits and base of y-scale highlights different aspects…

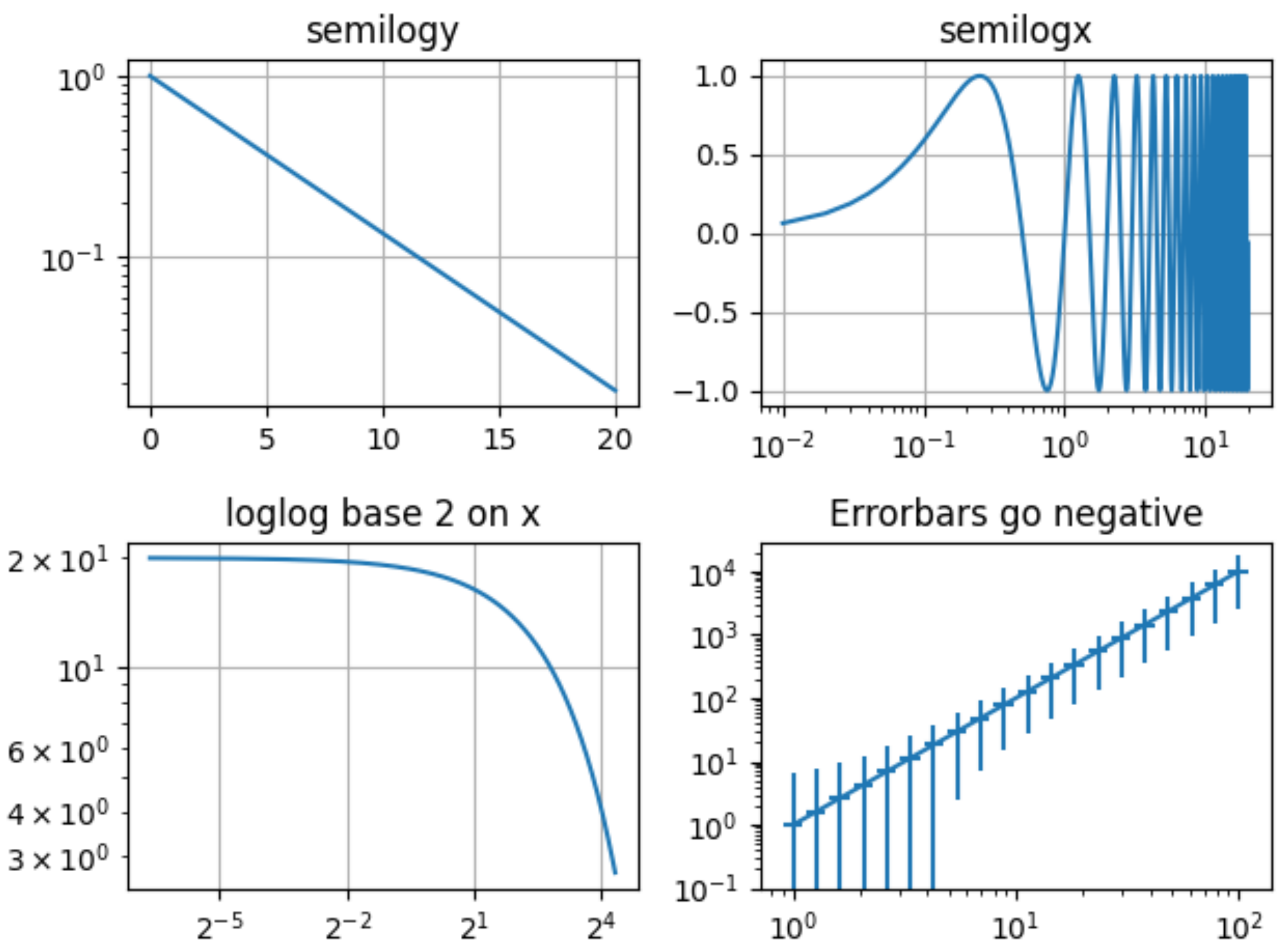




…log-scale emphasizes relative changes in smaller quantities

Line Plots in Log-Domain

```python
# Data for plotting
t = np.arange(0.01, 20.0, 0.01)

# Create figure
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)

# log y axis
ax1.semilogy(t, np.exp(-t / 5.0))
ax1.set(title='semilogy')
ax1.grid()

# log x axis
ax2.semilogx(t, np.sin(2 * np.pi * t))
ax2.set(title='semilogx')
ax2.grid()

# log x and y axis
ax3.loglog(t, 20 * np.exp(-t / 10.0))
ax3.set_xscale('log', base=2)
ax3.set(title='loglog base 2 on x')
ax3.grid()

# With errorbars: clip non-positive values
# Use new data for plotting
x = 10.0**np.linspace(0.0, 2.0, 20)
y = x**2.0

ax4.set_xscale("log", nonpositive='clip')
ax4.set_yscale("log", nonpositive='clip')
ax4.set(title='Errorbars go negative')
ax4.errorbar(x, y, xerr=0.1 * x, yerr=5.0 + 0.75 * y)
# ylim must be set after errorbar to allow errorbar to autoscale limits
ax4.set_ylim(bottom=0.1)
```



Pandas

Open source library for data handling and manipulation in high-performance environments.

Installation If you are using Anaconda package manager,

```
conda install pandas
```

Or if you are using PyPi (pip) package manager,

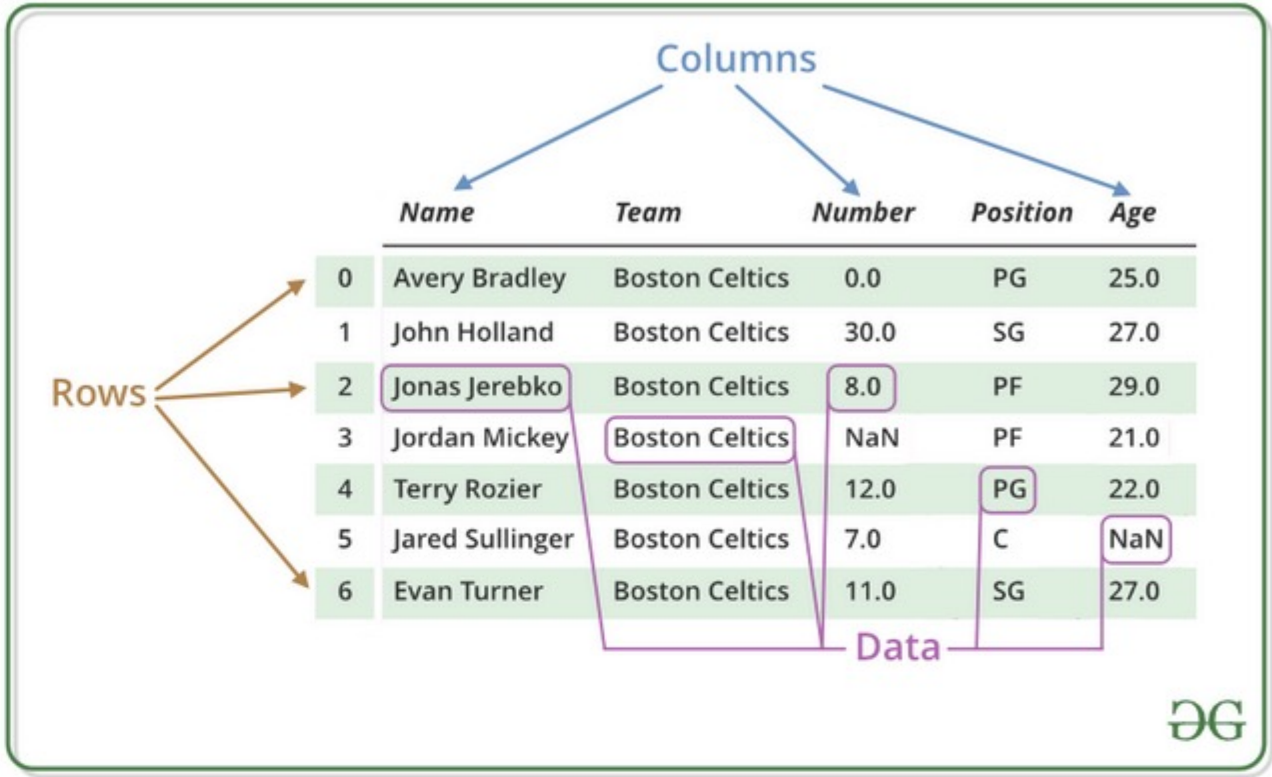
```
pip install pandas
```

See Pandas documentation for more detailed instructions

https://pandas.pydata.org/docs/getting_started/install.html

DataFrame

Primary data structure : Essentially a table



[Source: https://www.geeksforgeeks.org/python-pandas-dataframe/]

DataFrame Example

Create and print an entire DataFrame

```python
# import pandas as pd
import pandas as pd

# list of strings
lst = ['Geeks', 'For', 'Geeks', 'is',
            'portal', 'for', 'Geeks']

# Calling DataFrame constructor on list
df = pd.DataFrame(lst)
print(df)
```

	0
0	Geeks
1	For
2	Geeks
3	is
4	portal
5	for
6	Geeks

[Source: https://www.geeksforgeeks.org/python-pandas-dataframe/]

DataFrame Example

Can create named columns using dictionary

```python
import pandas as pd

# intialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}

# Create DataFrame
df = pd.DataFrame(data)

# Print the output.
print(df)
```

	Name	Age
0	Tom	20
1	nick	21
2	krish	19
3	jack	18

[Source: https://www.geeksforgeeks.org/python-pandas-dataframe/]

DataFrame : Selecting Columns

Select columns to print by name,

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# select two columns
print(df[['Name', 'Qualification']])
```

	Name	Qualification
0	Jai	Msc
1	Princi	MA
2	Gaurav	MCA
3	Anuj	Phd

[Source: https://www.geeksforgeeks.org/python-pandas-dataframe/]

DataFrame : Selecting Rows

```python
import pandas as pd
import numpy as np

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Print rows 1 & 2
row = df.loc[1:2]
print(row)
```

Output

```
     Name  Age    Address Qualification
1  Princi   24     Kanpur            MA
2  Gaurav   22  Allahabad           MCA
```

DataFrame : Selecting Rows

`head()` and `tail()` select rows from beginning / end

```
import pandas as pd
import numpy as np

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Print first / last rows
first2 = df.head(2)
last2 = df.tail(2)
print(first2)
print('\n', last2)
```

Output

```
     Name  Age Address Qualification
0     Jai   27   Delhi           Msc
1  Princi   24  Kanpur            MA

     Name  Age    Address Qualification
2  Gaurav   22  Allahabad           MCA
3    Anuj   32    Kannauj           Phd
```

Reading Data from Files

Easy reading / writing of standard formats,

```
df = pd.read_json("data.json")
print(df)
df.to_csv("data.csv", index=False)
df_csv = pd.read_csv("data.csv")
print(df_csv.head(2))
```

Output

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.4
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
   Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
```

Data Structure Conversions

Working with DataFrames outside of Pandas can be tricky,

```
df['Duration']
```

```
0      60
1      60
2      60
3      45
4      45
       ..
164    60
165    60
166    60
167    75
168    75
Name: Duration, Length: 169, dtype: int64
```

We can easily convert to built-in types, for example to a list (e.g. to use in Numpy or whatever),

```
L = df['Duration'].to_list()
print(L)
```

```
[60, 60, 60, 45, 45, 60, 60, 45, 30, 60, 60, 60, 60, 60, 60, 60, 60, 45, 60, 45, 60, 45, 60, 45, 60, 60, 60, 60, 60,
60, 60, 45, 60, 60, 60, 60, 60, 60, 60, 45, 45, 60, 60, 60, 60, 60, 60, 45, 45, 60, 60, 80, 60, 60, 30, 60, 60, 45, 2
0, 45, 210, 160, 160, 45, 20, 180, 150, 150, 20, 300, 150, 60, 90, 150, 45, 90, 45, 45, 120, 270, 30, 45, 30, 120, 4
5, 30, 45, 120, 45, 20, 180, 45, 30, 15, 20, 20, 30, 25, 30, 90, 20, 90, 90, 90, 30, 30, 180, 30, 90, 210, 60, 45, 1
5, 45, 60, 60, 60, 60, 60, 60, 30, 45, 60, 60, 60, 60, 60, 60, 90, 60, 60, 60, 60, 60, 60, 20, 45, 45, 45, 20, 60, 6
0, 45, 45, 60, 45, 60, 60, 30, 60, 60, 60, 60, 30, 60, 60, 60, 60, 60, 30, 30, 45, 45, 45, 60, 60, 60, 75, 75]
```

Summary Statistics

Easily compute summary statistics on data

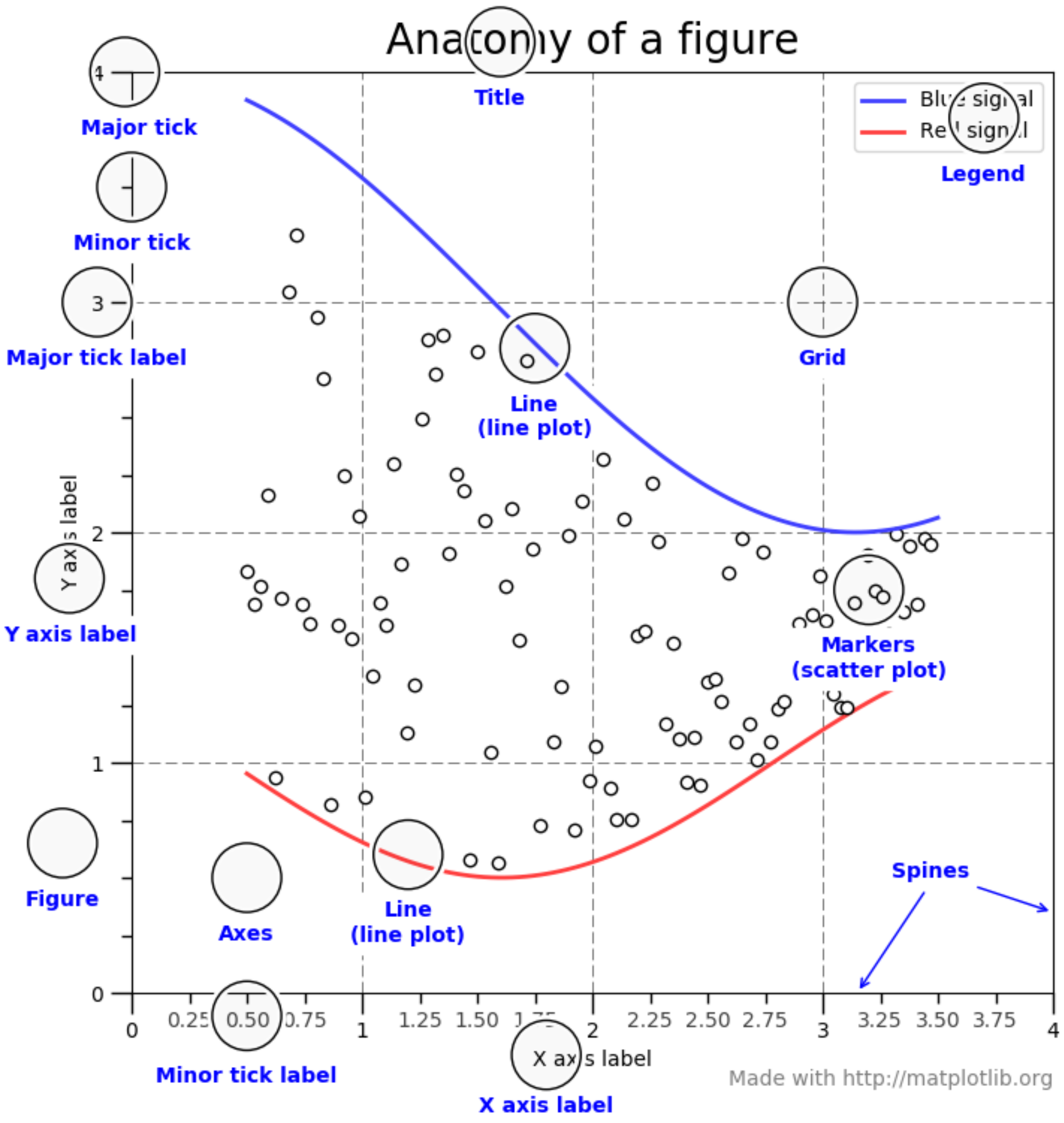
```
print('Min: ', df['Duration'].min())
print('Max: ', df['Duration'].max())
print('Median: ', df['Duration'].median())
```

```
Min:   15
Max:   300
Median:   60.0
```

Can also count occurrences of unique values,

```
df['Duration'].value_counts()
```

```
60     79
45     35
30     16
20      9
90      8
150     4
120     3
180     3
15      2
75      2
160     2
210     2
270     1
25      1
300     1
80      1
Name: Duration, dtype: int64
```



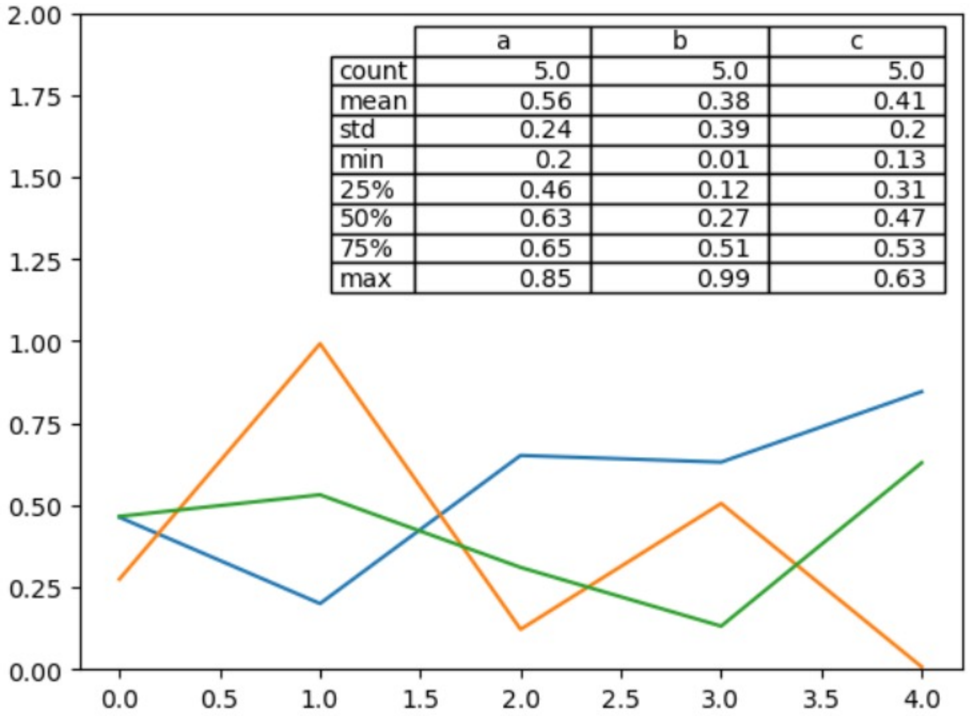
matplotlib

Documentation + tutorials:

https://matplotlib.org/

Pandas / Matplotlib Integration

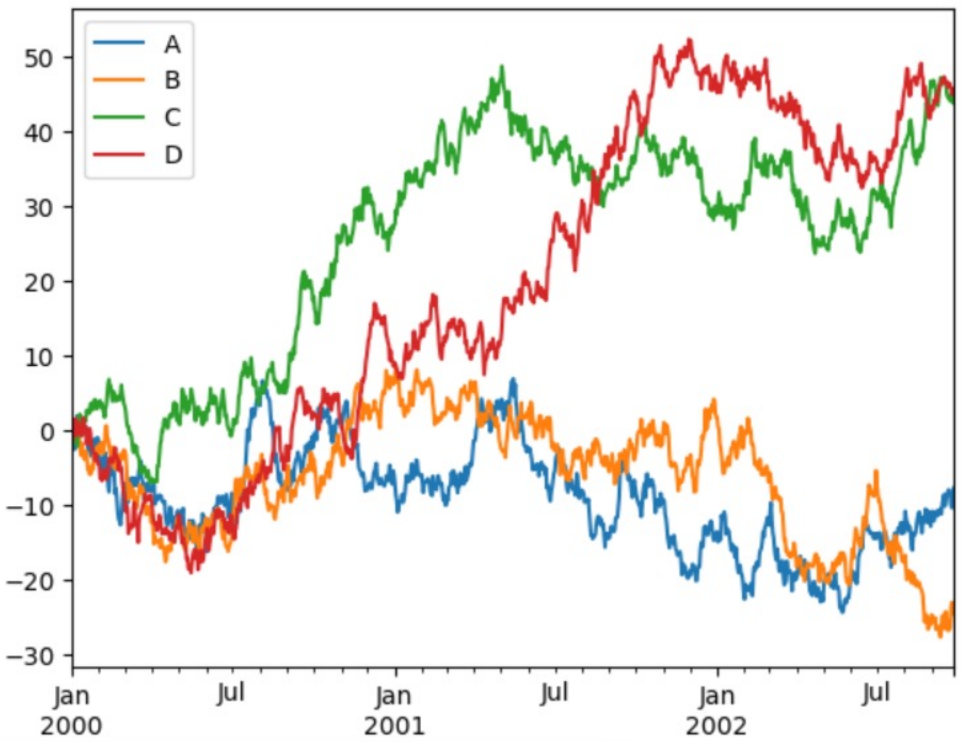
```python
from pandas.plotting import table
fig, ax = plt.subplots(1, 1)
df = pd.DataFrame(np.random.rand(5, 3), columns=["a", "b", "c"])
table(ax, np.round(df.describe(), 2), loc="upper right", colWidths=[0.2, 0.2, 0.2]);
df.plot(ax=ax, ylim=(0, 2), legend=None);
```



DataFrame.plot() wraps Matplotlib’s plot() function

Pandas / Matplotlib Integration

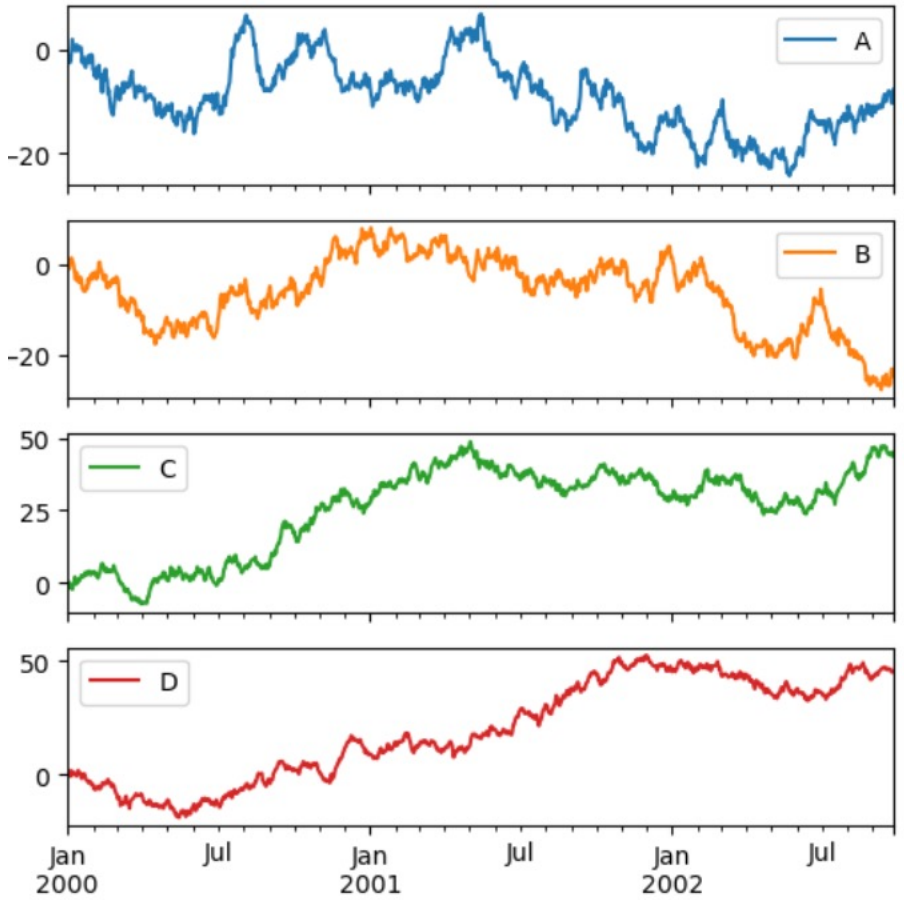
```
df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index, columns=list("ABCD"))
df = df.cumsum()
plt.figure();
df.plot();
```

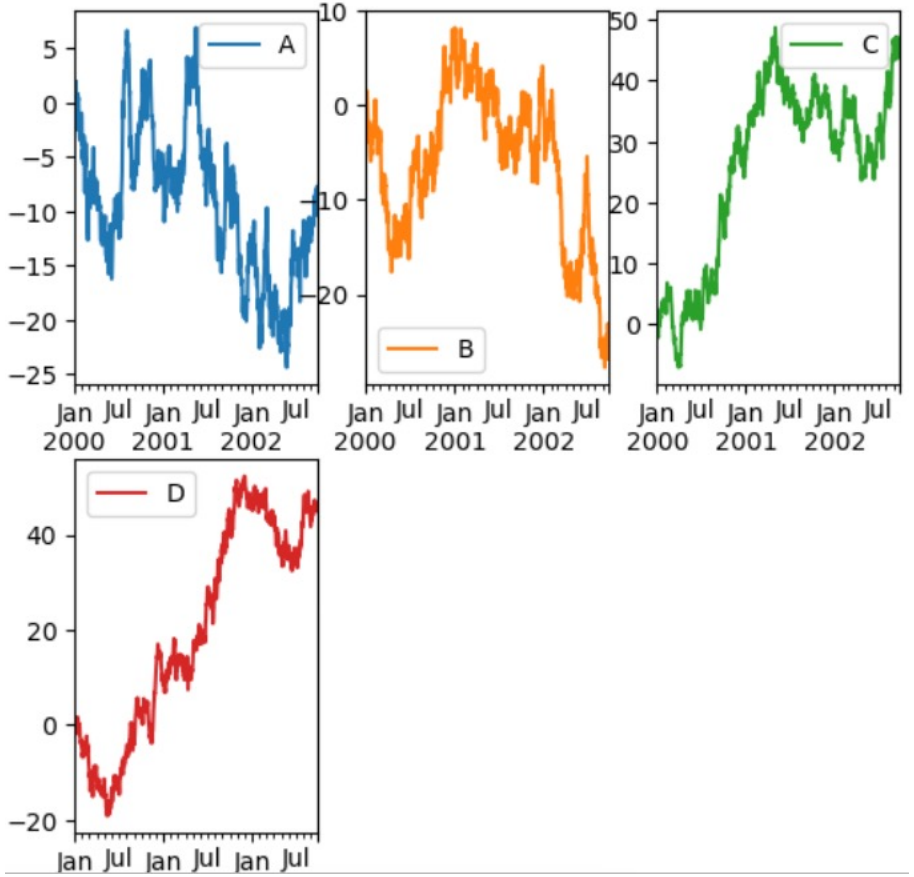


Specifies reasonable defaults for colors, legend, etc.

Pandas / Matplotlib Integration

```
df.plot(subplots=True, figsize=(6, 6));
```

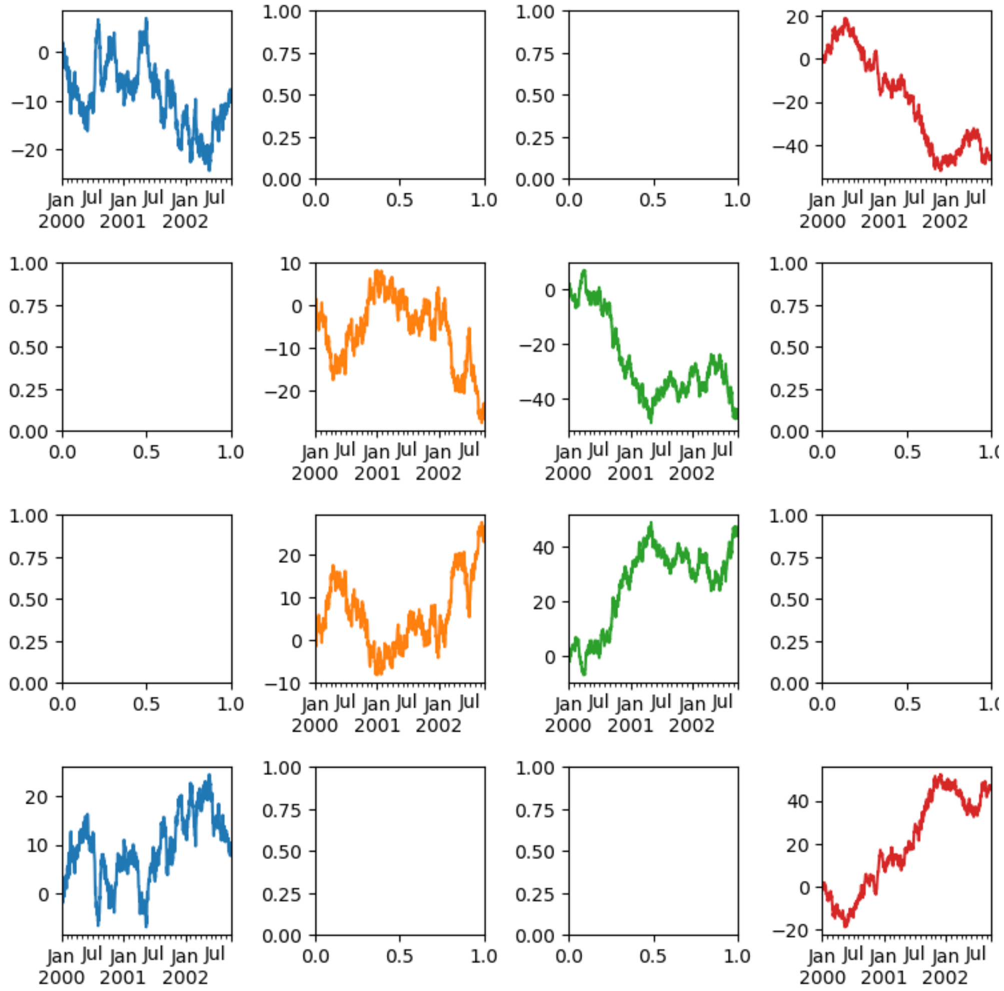





```
df.plot(subplots=True, layout=(2, 3), figsize=(6, 6), sharex=False);
```

Pandas / Matplotlib Integration

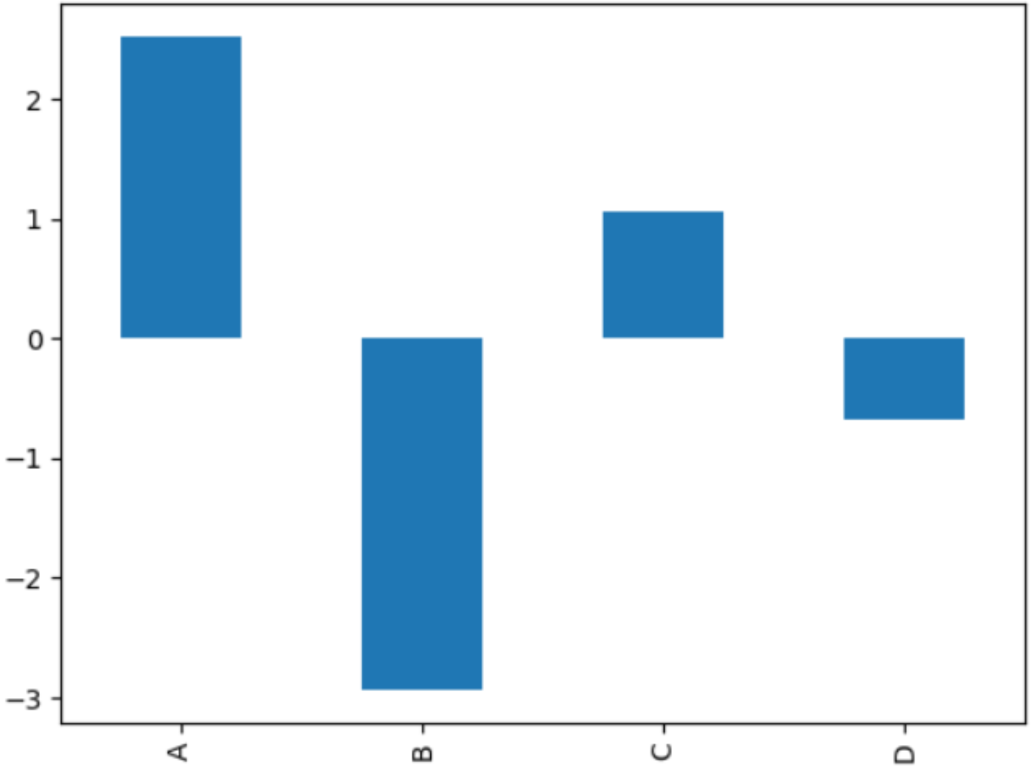
```
fig, axes = plt.subplots(4, 4, figsize=(9, 9))
plt.subplots_adjust(wspace=0.5, hspace=0.5)
target1 = [axes[0][0], axes[1][1], axes[2][2], axes[3][3]]
target2 = [axes[3][0], axes[2][1], axes[1][2], axes[0][3]]
df.plot(subplots=True, ax=target1, legend=False, sharex=False, sharey=False);
(-df).plot(subplots=True, ax=target2, legend=False, sharex=False, sharey=False);
```

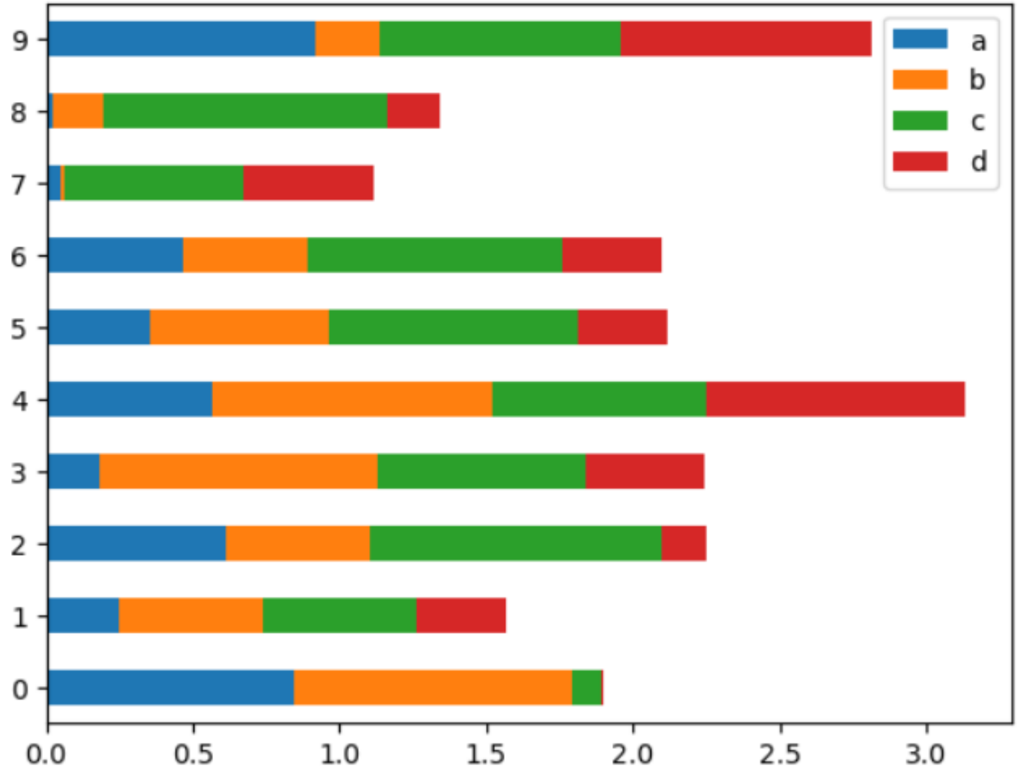


Easy control over subplot placement & handling

Pandas / Matplotlib Integration

```
plt.figure();
df.iloc[5].plot(kind="bar");
```

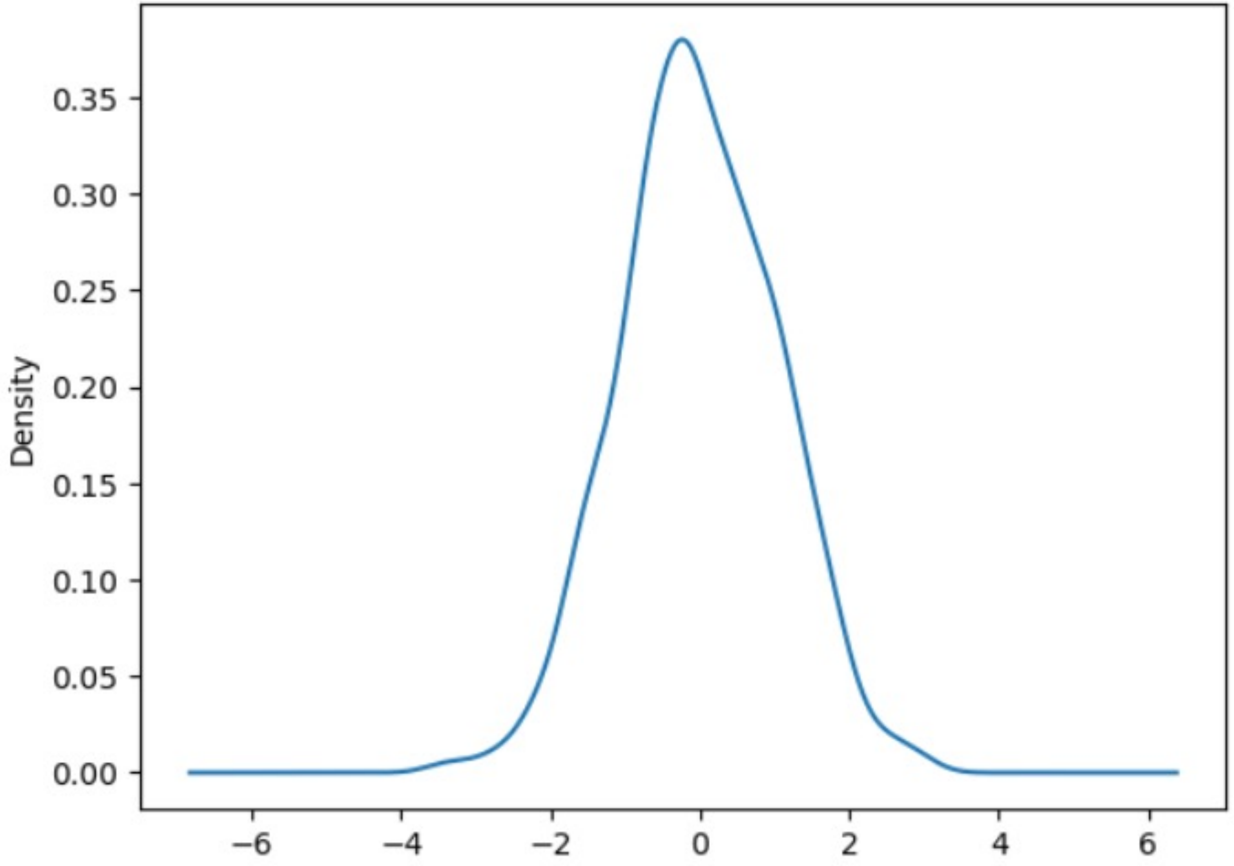





```
df2 = pd.DataFrame(np.random.rand(10, 4), columns=["a", "b", "c", "d"])
df2.plot.barh(stacked="True");
```

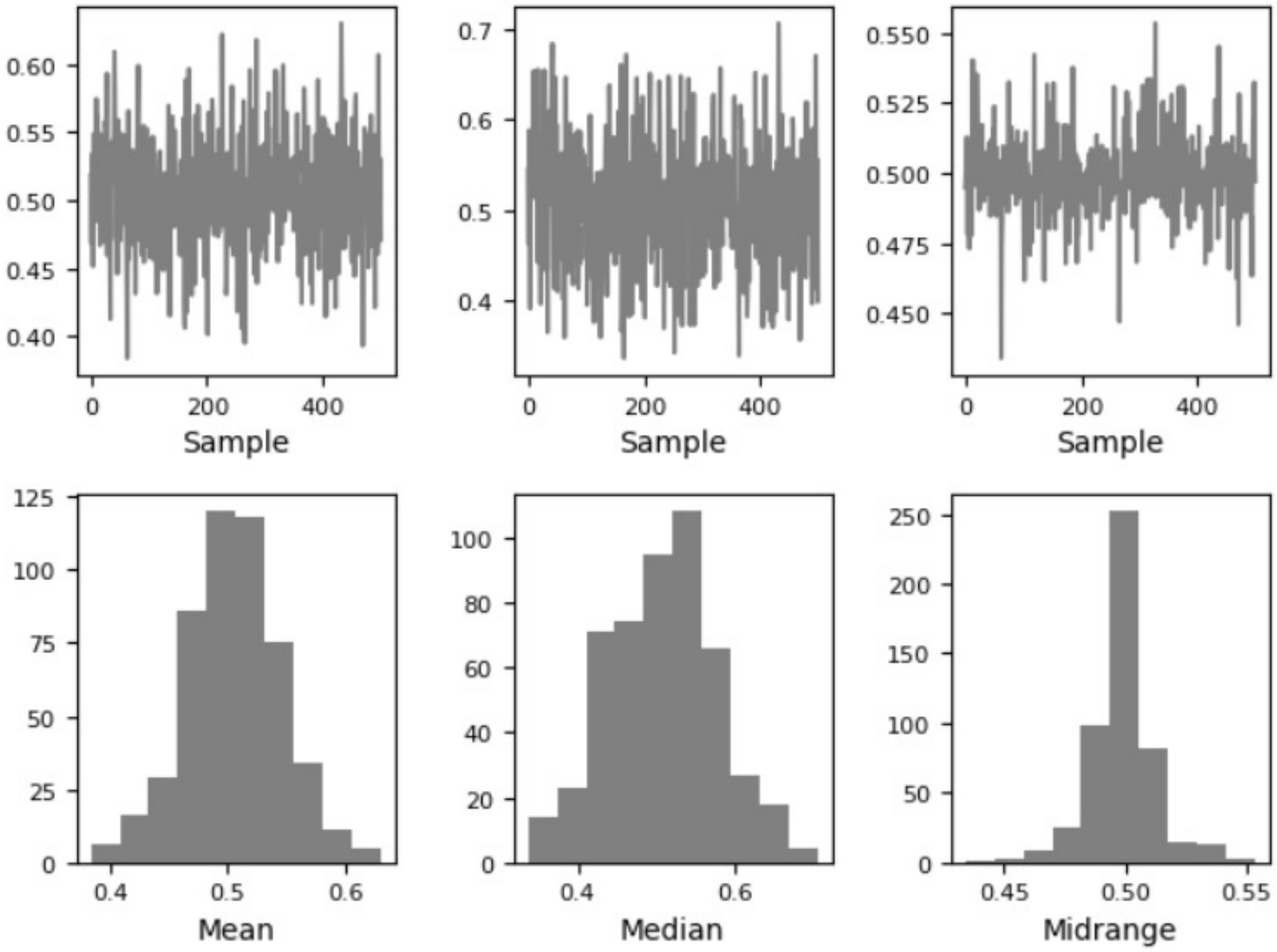
Pandas / Matplotlib Integration

```
ser = pd.Series(np.random.randn(1000))
ser.plot(kind="density");
```



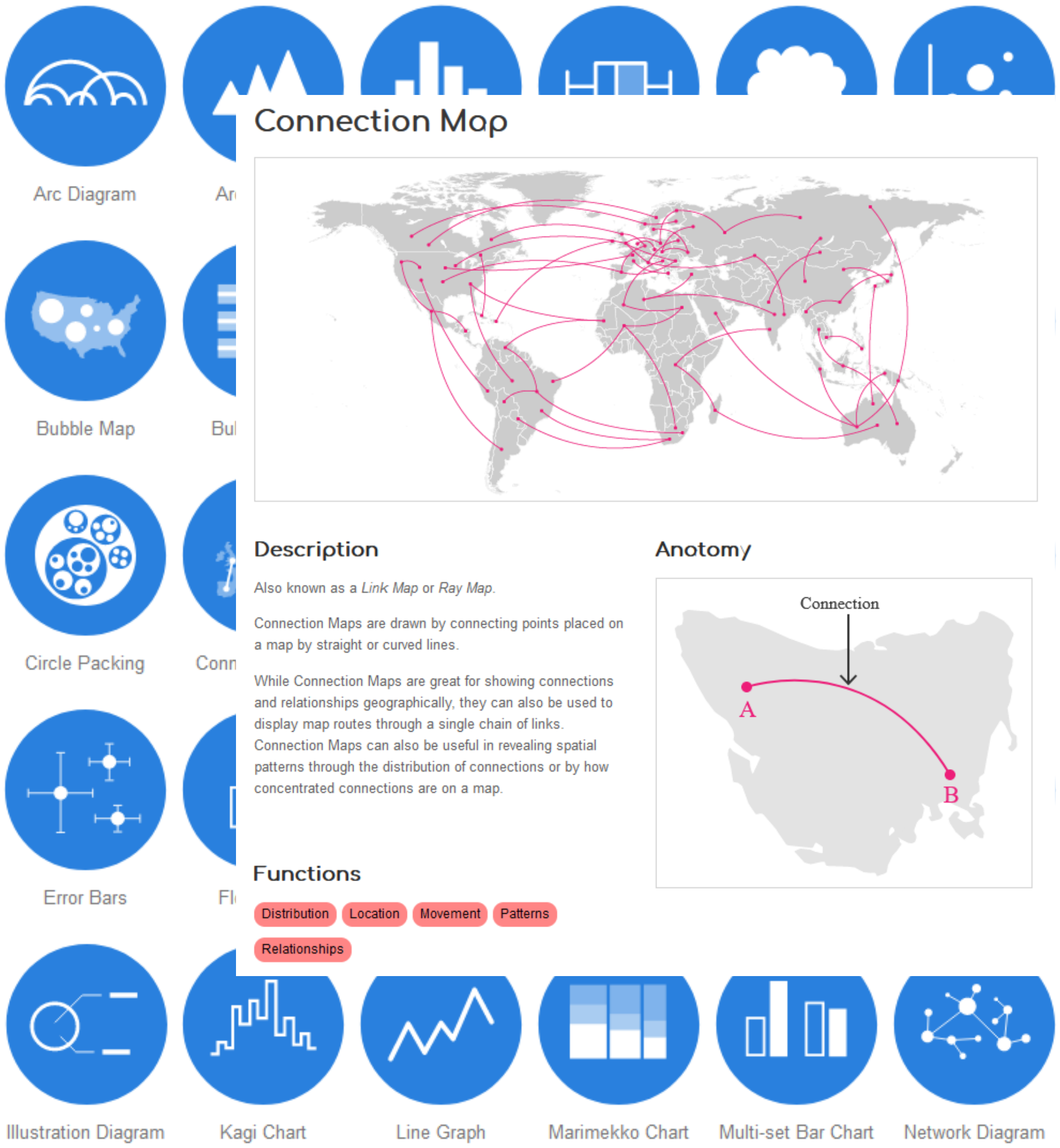
Pandas / Matplotlib Integration

```
from pandas.plotting import bootstrap_plot
data = pd.Series(np.random.rand(1000))
bootstrap_plot(data, size=50, samples=500, color="grey");
```



More Visualization Resources

datavizcatalogue.com



matplotlib.org

scikit-learn.org