# EEN1085/EEN1083
## Data analysis and machine learning I

Ali Intizar

Semester 1
2024/2025

Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

# Data summarization and visualization

Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

## Overview

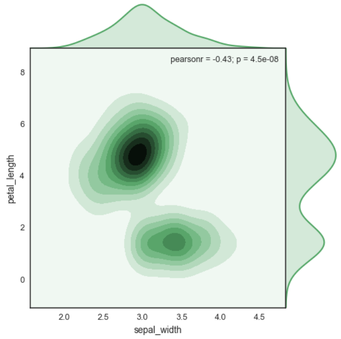
Σ

**Data summarization**

- Summary statistics
- Measures of central tendency
- Measures of variation
- Measures of correlation

**Data visualization**

- Why visualization is important
- Visualizing distributions
- Visualizing relationships
- Comparing quantities
- Visualizing composition
- Designing visualizations
- Non-statistical visualizations



$$\Sigma$$

## Data summarization

# Overview

Data **distributions**

Measures of **centrality**

- Mean, median, mode
- Sample mean vs population mean

Measures of **variation**

- Variance, standard deviation
- Range and Interquartile range

Sufficient statistics

Skewness and kurtosis

Conditioning on attributes

Measures of association

- Pearson
- Spearman

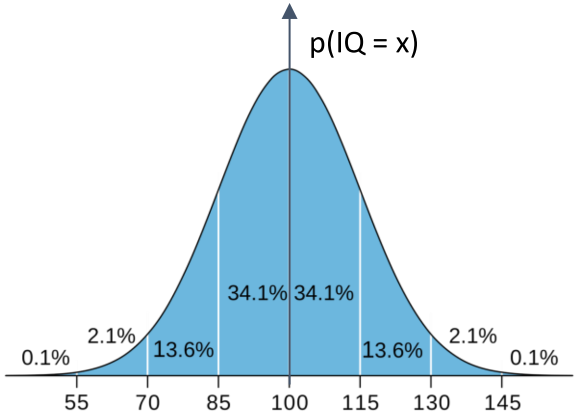
# Distributions

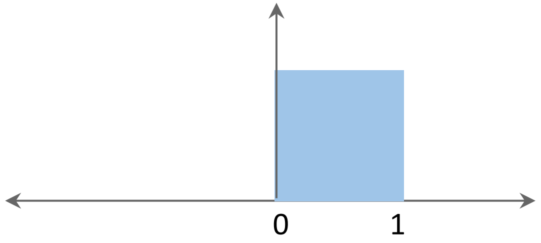
The data distribution describes the probability of a random variable taking a particular value.

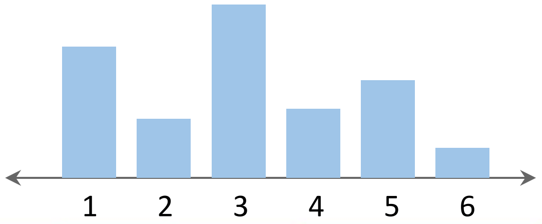
Example: IQ follows a **Normal (Gaussian) distribution.** ~70% of people have an IQ between 85 and 115, 95% between 70 and 130.

A random number generator (np.random.rand) will generate random numbers that follow a **uniform distribution** in [0, 1]

A roll of a biased dice has a **categorical distribution** (multinomial) over the values {1, 2, 3, 4, 5, 6}







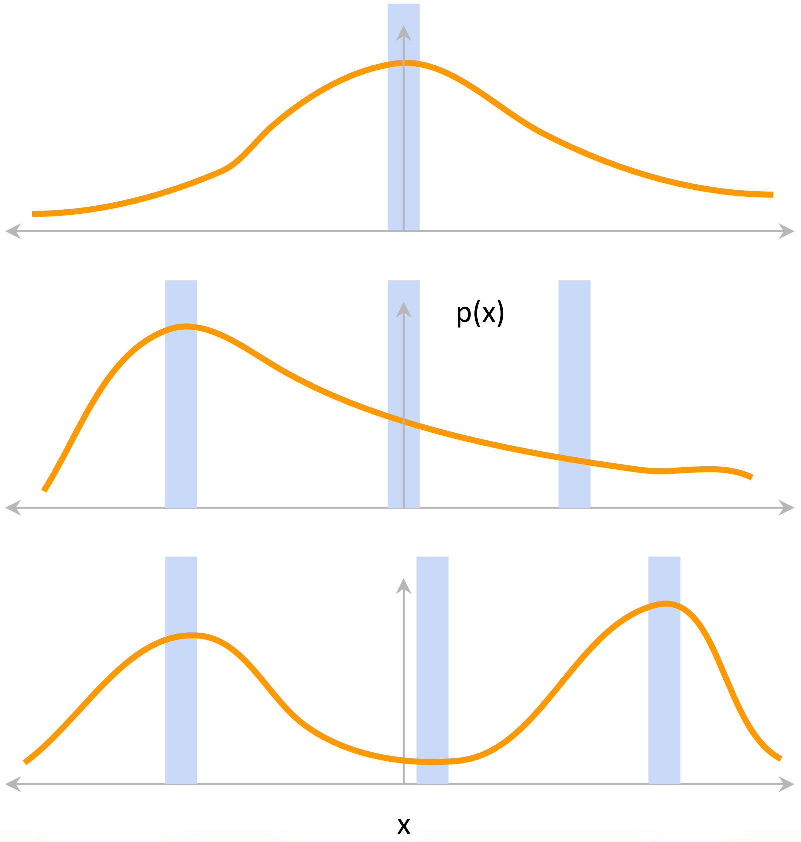
# Measures of central tendency

A **sample statistic** is a measurement on a sample from a distribution, calculated by applying a function to the sample.

Measures of **central tendency** are sample statistics that attempt to capture where the middle of a distribution is.

Many ways of measuring central tendency. Three common ones:

- Mean
- Median
- Mode



# Mean, median, mode

**Arithmetic mean**: sum of observations divided by number of observations

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

**Median**: the middle value that separates the higher half from the lower half of the data set

Sort data and take middle value.

Even number of observations: take arithmetic mean of middle two values

The median is a **robust statistic** (less sensitive to outliers)

**Mode**: the most frequent value in the dataset. Suitable measure of central tendency for nominal variables.

Easy to compute for discrete values: most frequent value.

Not straightforward for continuous distributions (some have no well-defined mode).

## Mean, median, mode

1, 2, 2, 3, 4, 7, 9

**Mean**

(1 + 2 + 2 + 3 + 4 + 7 + 9) / 7
= 4

**Median**

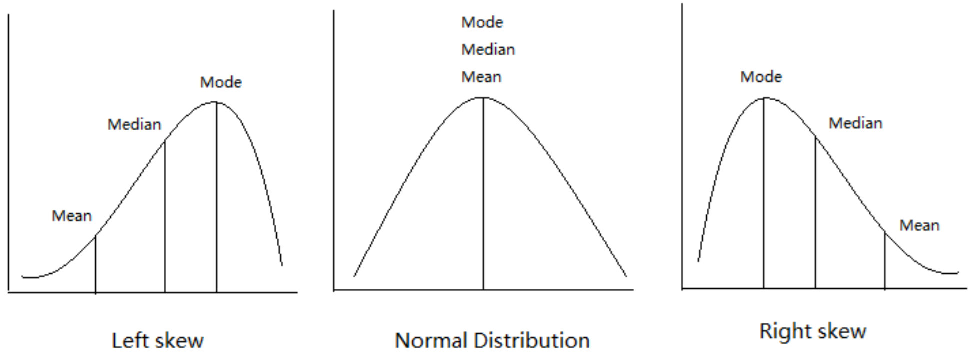
1, 2, 2, **3**, 4, 7, 9
= 3

**Mode**

1, **2**, **2**, 3, 4, 7, 9
= 2

# Mean, median, mode

| | Dimension | Type | Calculate |
|---|---|---|---|
| Mean | *N* | Quantitative | `np.mean` |
| Median | *1* | Ordinal | `np.median` |
| Mode | *N* | Nominal | From histogram |



# Sample mean vs population mean

The **population mean** is the true mean for the entire population.

$$\mu$$

The **sample mean** (average) is the calculated mean of a sample from a population.

$$\bar{x}$$

**E.g.** the population mean human height would require measuring the height of every person on earth. A sample mean 100 **randomly chosen** people.

As sample size increases the sample mean approaches the population mean.

$$\lim_{n \to \infty} \bar{x}_n = \mu$$

# Measures of statistical dispersion

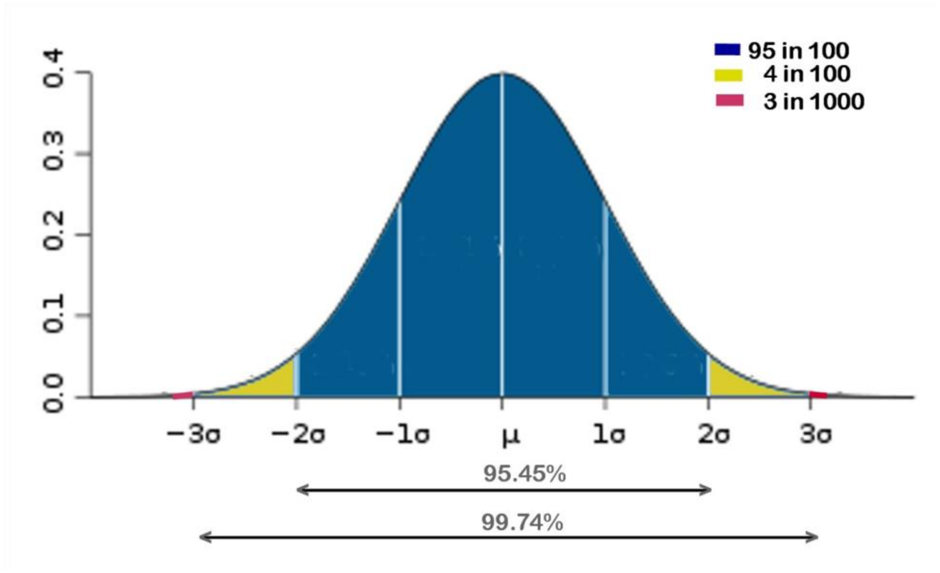
Measures of how stretched or squeezed a distribution is.

Most common methods are:

- Variance ($\sigma^2$)
- Standard deviation ($\sigma$)
- Interquartile range (IQR)

Again, can be defined for sample or population.

Normal PDF



# Variance, SD

**Variance**: the expected squared deviation from the mean.

$$\begin{aligned}\sigma^2 &= \mathbf{E}[(X - \mathbf{E}X)^2] \\ &= \mathbf{E}[(X - \mu)^2]\end{aligned}$$

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

Usually we don't know the population mean $\mu$, so we just use the sample mean:

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

For small samples, this is biased, so may want to use the bias corrected estimator:

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

The above is known as Bessel's correction.

The **standard deviation** is just the square root of the (possibly bias corrected) variance.

$$\sigma = \sqrt{\sigma^2}$$

# Range and interquartile range

The **range** of a sample is the difference between its minimum and maximum values.

The range is not a robust statistic!

The IQR is the range in which 50% of the data lies.

IQR = $Q_3 - Q_1$ = 119 − 31 = 88

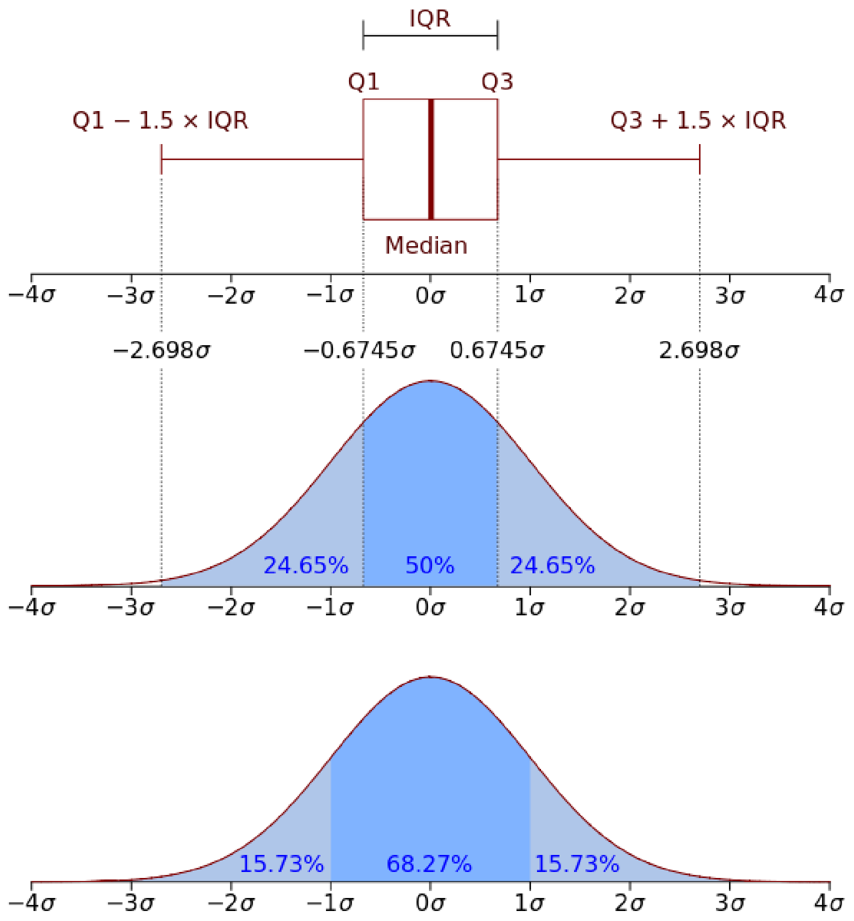
```
q1, q3 = np.percentile(x, [75 ,25])
iqr = q1 - q3
```

**Pandas**: `DataFrame.quantile()`

| | i | $x_i$ | Quartile | |
|---|---|---|---|---|
| | 1 | 7 | | |
| 25% | 2 | 7 | | |
| | 3 | 21 | | |
| | 4 | 31 | $Q_1$ (25% percentile) | |
| | 5 | 47 | | |
| | 6 | 75 | | |
| | 7 | 87 | $Q_2$ (median) | IQR |
| | 8 | 115 | | |
| | 9 | 116 | | |
| | 10 | 119 | $Q_3$ (75% percentile) | |
| | 11 | 119 | | |
| 25% | 12 | 155 | | |
| | 13 | 177 | | |

# Interquartile range

IQM can be used on ordinal variables and often makes more sense than the standard deviation.



## Sufficient statistics

In statistics, a statistic is **sufficient** with respect to a statistical model and its associated unknown parameter when no other statistic that can be calculated from the same sample provides any additional information as to the value of the parameter.

For the Normal distribution, sufficient statistics are the mean and standard deviation.

$$\theta = \{\mu, \sigma\}$$

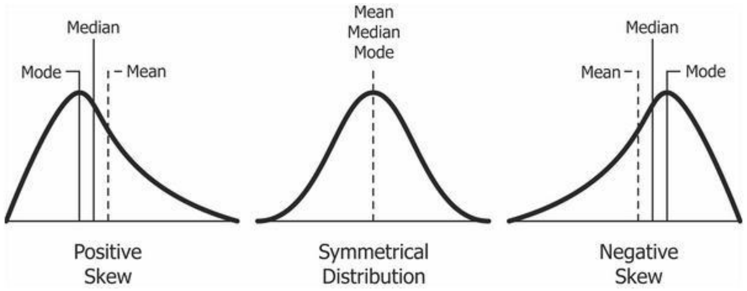
**Once you know the sufficient statistics, you know everything there is to know about the distribution of a variable.**

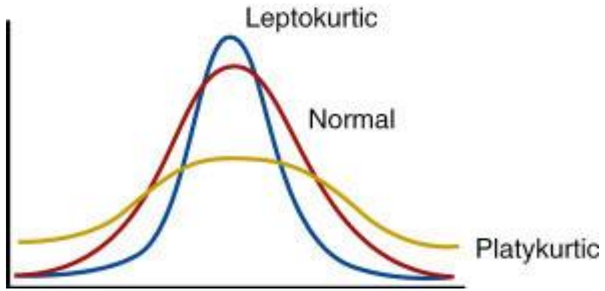
## Skewness and kurtosis

Most data are not normally distributed.

**Skewness** is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.

**Kurtosis** is a measure of how heavy the tails are of a probability distribution of a real-valued random variable.





# Skewness and kurtosis

**Pearson's moment coefficient of skewness**
Third standardized moment:

$$\gamma_1 = \operatorname{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$

**Pearson's moment coefficient of kurtosis**
Fourth standardized moment:

$$\beta_2 = \operatorname{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$$

Replace mu and sigma with sample statistics to calculate for a sample

`scipy.stats.skew(x)`

`scipy.stats.kurtosis(x)`

# Pandas describe

```
import seaborn as sns

tips = sns.load_dataset("tips")
tips.head()
```

| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| 0 | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| 1 | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| 2 | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| 3 | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| 4 | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

```
tips.describe()
```

| | total_bill | tip | size |
|---|---|---|---|
| count | 244.000000 | 244.000000 | 244.000000 |
| mean | 19.785943 | 2.998279 | 2.569672 |
| std | 8.902412 | 1.383638 | 0.951100 |
| min | 3.070000 | 1.000000 | 1.000000 |
| 25% | 13.347500 | 2.000000 | 2.000000 |
| 50% | 17.795000 | 2.900000 | 2.000000 |
| 75% | 24.127500 | 3.562500 | 3.000000 |
| max | 50.810000 | 10.000000 | 6.000000 |

# Conditioning on categorical variables

Compute statistics based on subsets of the sample.

Subsets selected by grouping on categorical variables.

These are statistics on the conditional distributions. E.g. **P(size | time = Lunch)**

**Pandas**:

```
tips.groupby('time').describe()

tips.groupby('smoker').mean()
```

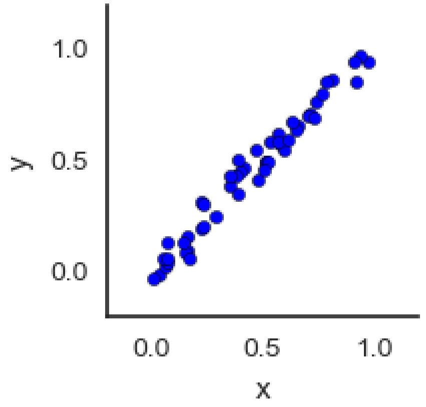
| | | size | tip | total_bill |
|---|---|---|---|---|
| Lunch | count | 68.000000 | 68.000000 | 68.000000 |
| | mean | 2.411765 | 2.728088 | 17.168676 |
| | std | 1.040024 | 1.205345 | 7.713882 |
| | min | 1.000000 | 1.250000 | 7.510000 |
| | 25% | 2.000000 | 2.000000 | 12.235000 |
| | 50% | 2.000000 | 2.250000 | 15.965000 |
| | 75% | 2.000000 | 3.287500 | 19.532500 |
| | max | 6.000000 | 6.700000 | 43.110000 |
| Dinner | count | 176.000000 | 176.000000 | 176.000000 |
| | mean | 2.630682 | 3.102670 | 20.797159 |
| | std | 0.910241 | 1.436243 | 9.142029 |
| | min | 1.000000 | 1.000000 | 3.070000 |
| | 25% | 2.000000 | 2.000000 | 14.437500 |
| | 50% | 2.000000 | 3.000000 | 18.390000 |
| | 75% | 3.000000 | 3.687500 | 25.282500 |
| | max | 6.000000 | 10.000000 | 50.810000 |

# Statistics of association between attributes

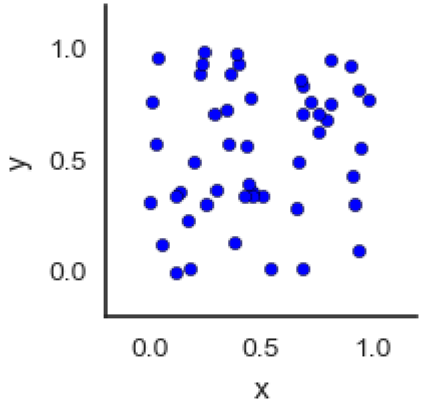
**Covariance:** measures how much two variables vary together.

$$\operatorname{cov}(X,Y) = \mathrm{E}[(X-\mu_X)(Y-\mu_Y)]$$

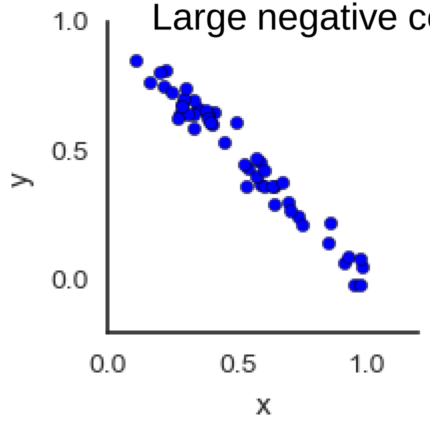
Large positive cov



Near zero cov



Large negative cov

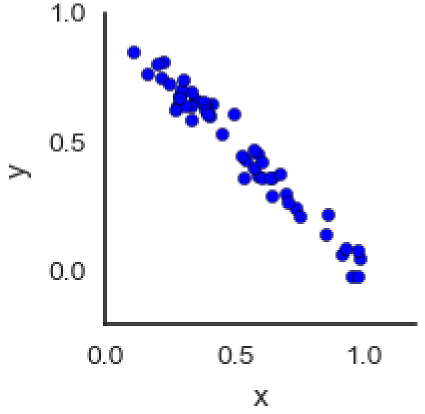


# Statistics of association between attributes

**Pearson product-moment correlation coefficient** (Pearson's rho): measure of the linear relationship between quantitative variables. Normalized version of the covariance:

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$$

rho = -1



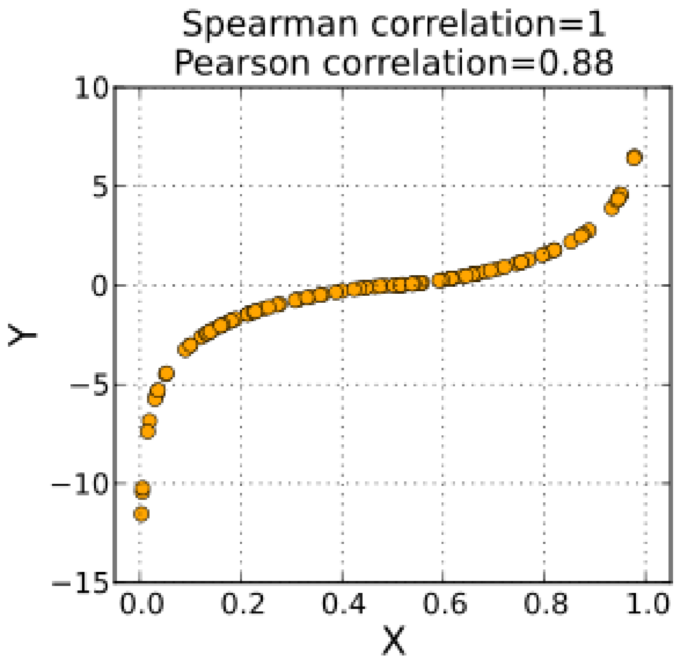
- Ranges from -1 to 1.
- > 0: positive correlation
- < 0: negative correlation
- +1 indicates a perfect linear relationship between X and Y
- -1 is perfect negative correlation

**Not suitable for ordinal variables!**

# Spearman correlation

**Spearman's rank correlation coefficient (Spearman's rho):** the Pearson correlation between the <u>rank values</u> of those two variables.

Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships (whether linear or not).



## Computing measures of association

**Covariance**

Numpy (covariance matrix)
```
np.cov(x, y)
```

**Spearman**

Scipy
```
scipy.stats.spearmanr(a, b)
```

Pandas
```
df.corr(method='spearman')
```

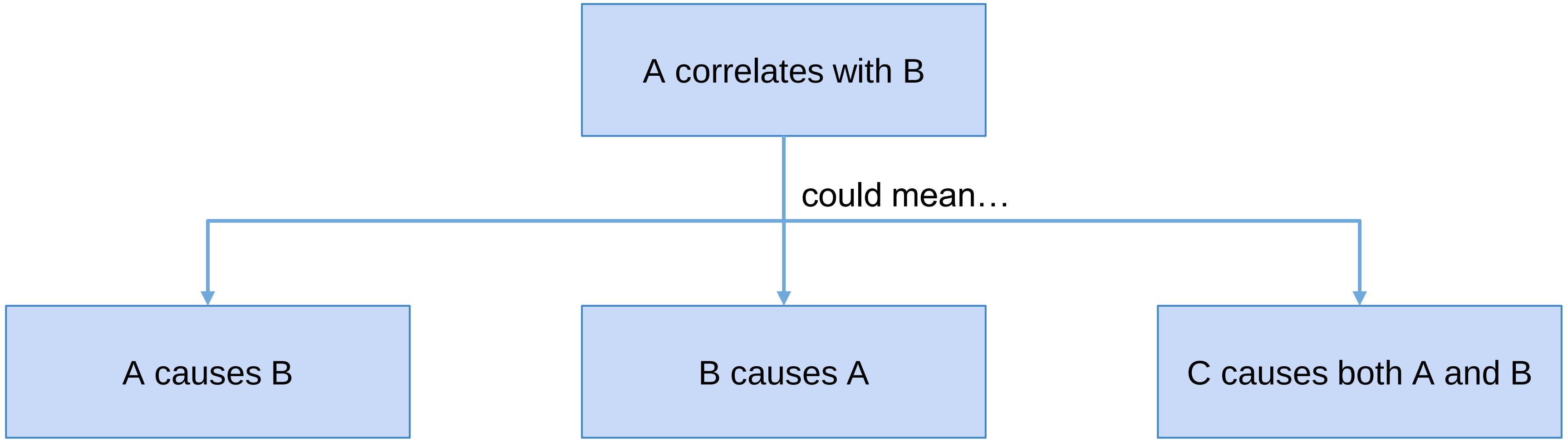
**Pearson**

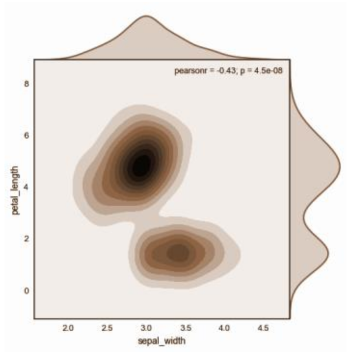
Scipy
```
scipy.stats.pearsonr(x, y)
```

Pandas
```
df.corr(method='pearson')
```

# On correlation vs causation

**Correlation does NOT imply causality!**





# Data visualization

## Why visualize data?

**Explore**

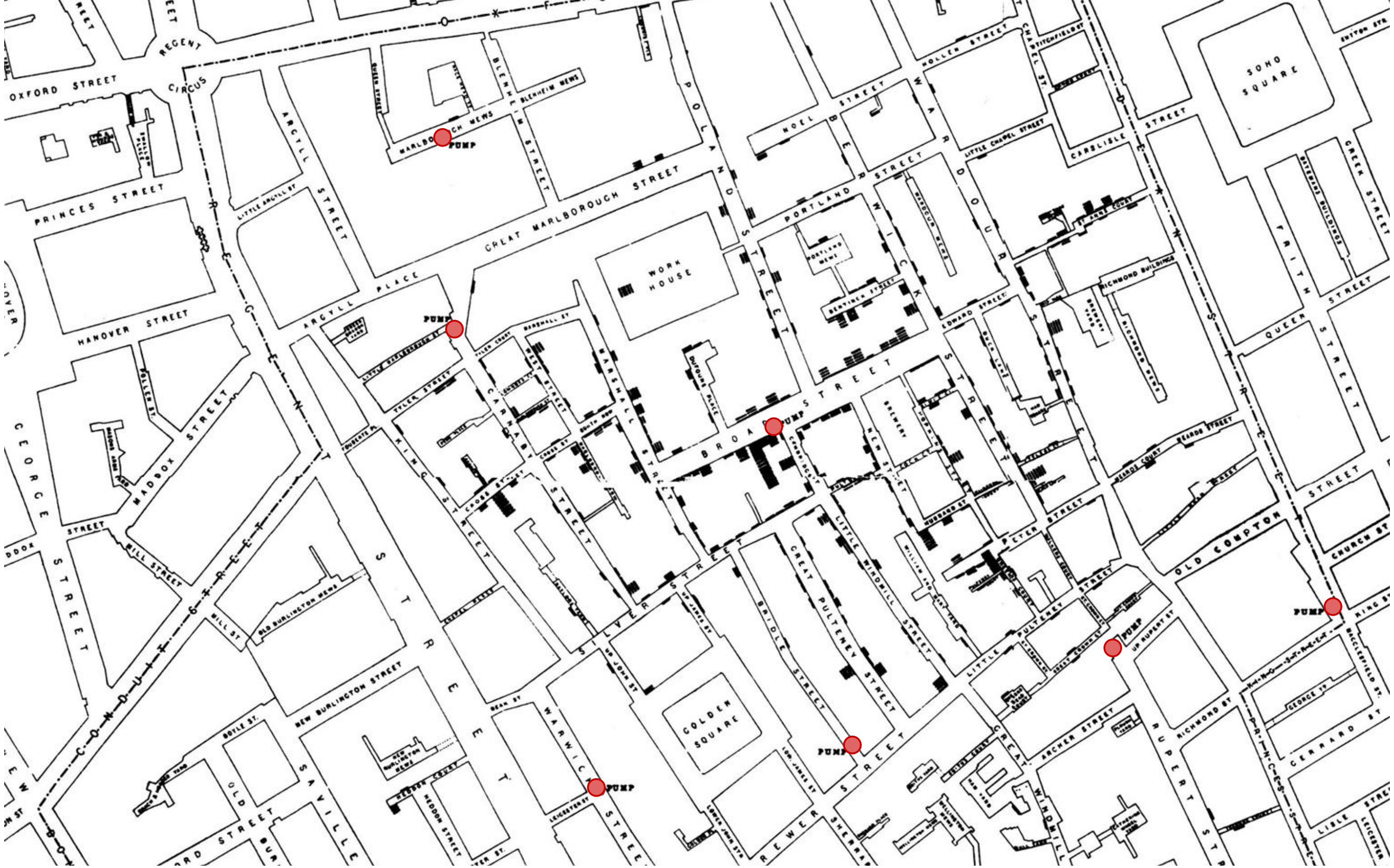
- Understand data and relationships between attributes
- Generate new hypotheses
- Detect errors and anomalies

**Confirm**

- Hypotheses already generated
- Check assumptions
- Verify conclusions

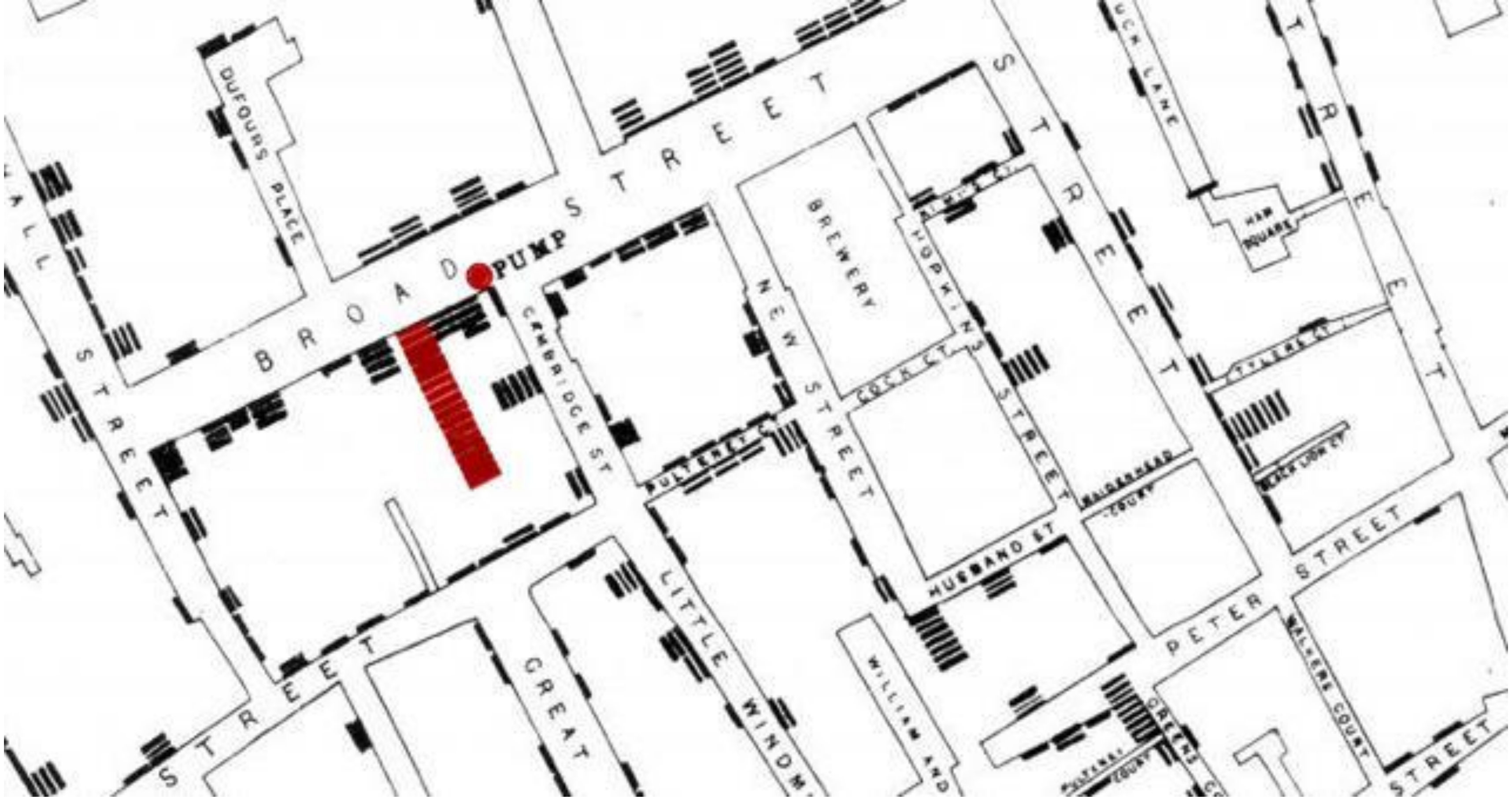
**Communicate**

- Analysis has been completed
- Conclusions verified
- Task: Present and communicate results
- Inform, persuade, educate, entertain

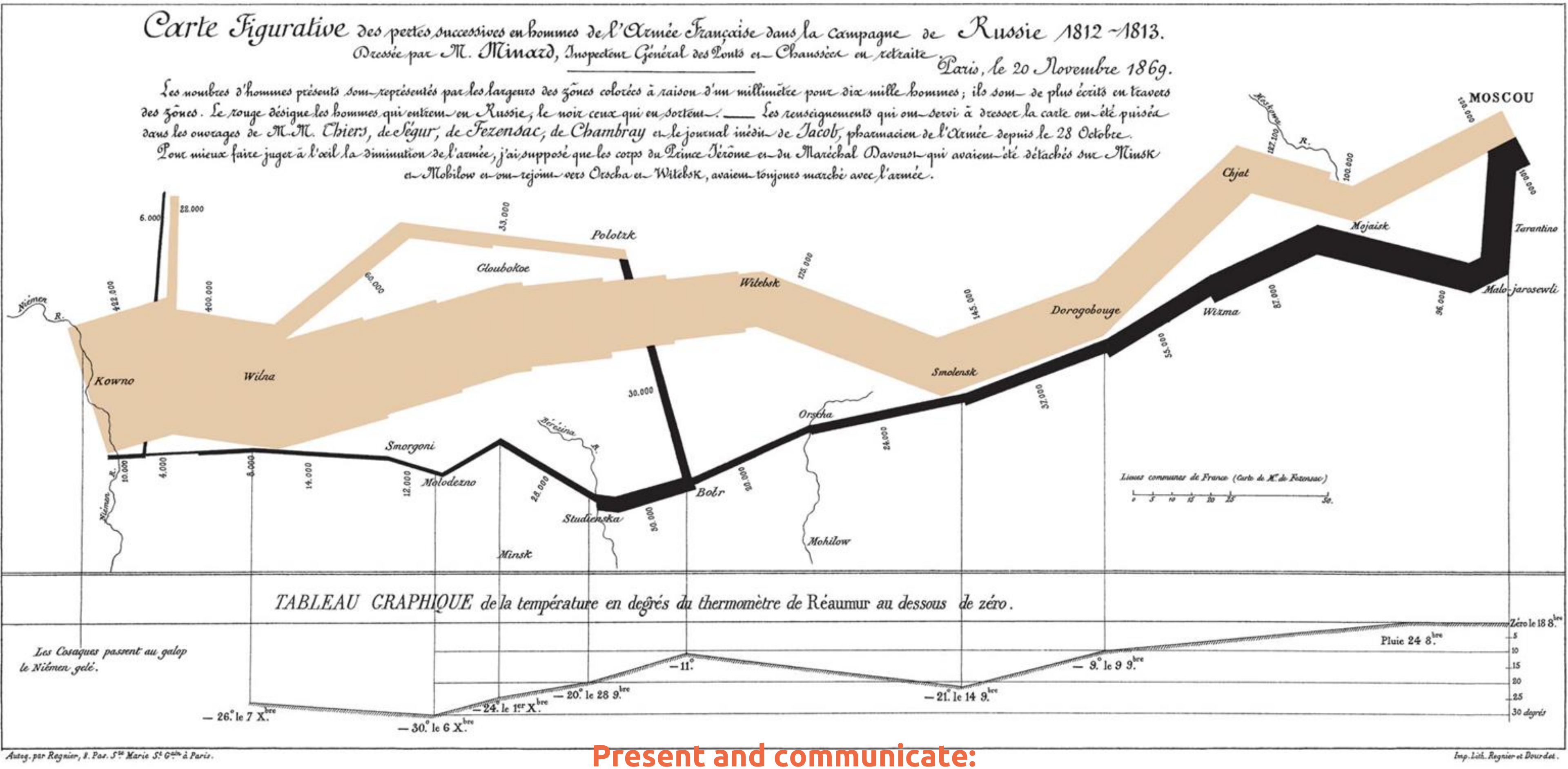


**Explore:**

John Snow's visualization of the London cholera outbreak of 1854



**Explore:**
John Snow's visualization of the London cholera outbreak of 1854


Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812–1813.
Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.
Paris, le 20 Novembre 1869.

Les nombres d'hommes présents sont représentés par les largeurs des zônes colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en travers des zônes. Le rouge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. —— Les renseignements qui ont servi à dresser la carte ont été puisés dans les ouvrages de MM. Thiers, de Ségur, de Fezensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre.
Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davoust qui avaient été détachés sur Minsk et Mobilow et ont rejoint vers Orscha et Witebsk, avaient toujours marché avec l'armée.

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

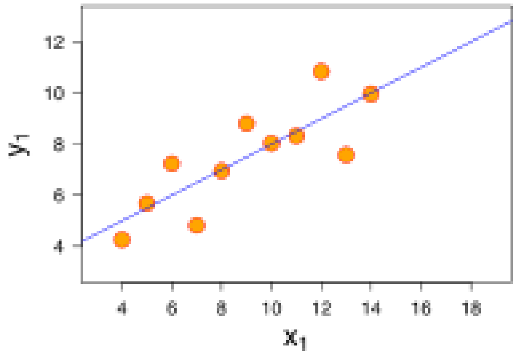
Les Cosaques passent au galop le Niémen gelé.

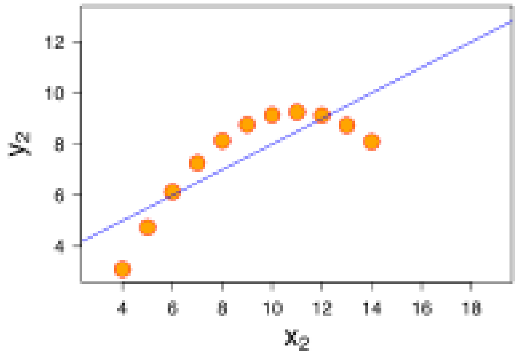
**Present and communicate:**

Charles Minard's map of Napoleon's disastrous Russian campaign of 1812.

## Importance of visualization: Anscombe's quartet

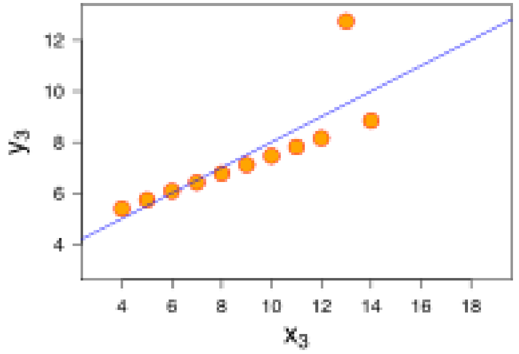
Linear + Gaussian Noise

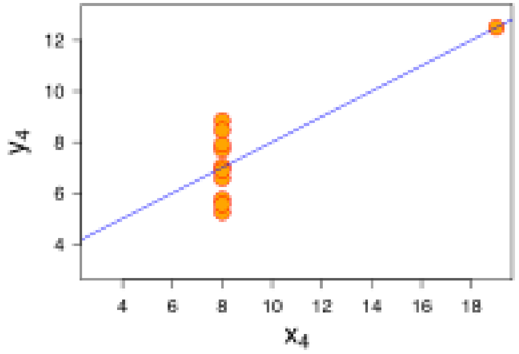




Nonlinear relationship

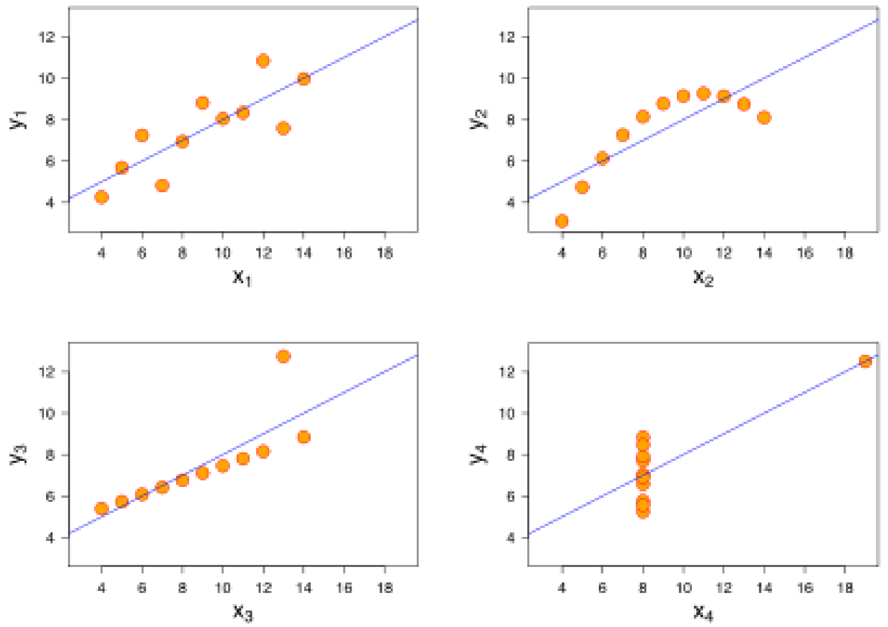
Linear + outlier





No relationship + outlier

# Importance of visualization: Anscombe's quartet



| | |
|---|---|
| Mean of x in each case | 9 |
| Sample variance of x in each case | 11 |
| Mean of y in each case | 7.50 |
| Sample variance of y in each case | 4.122 - 4.127 |
| Correlation between x and y in each case | 0.816 |
| Linear regression line in each case | y = 3.00 + 0.500x |

## Statistical data visualization

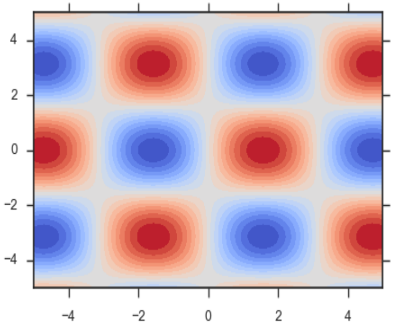
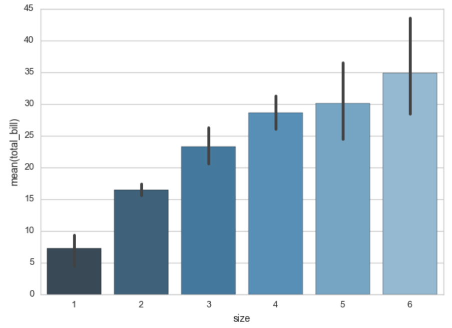
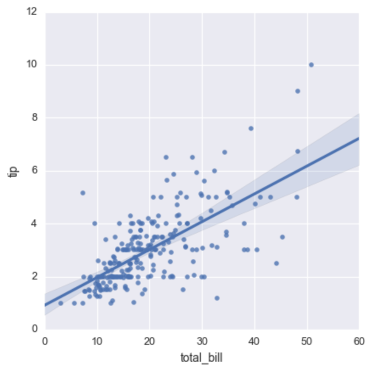
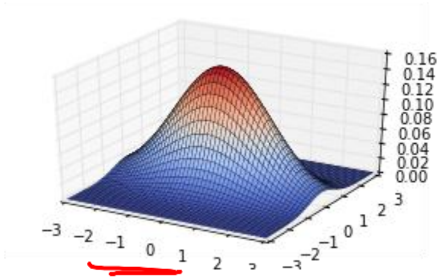
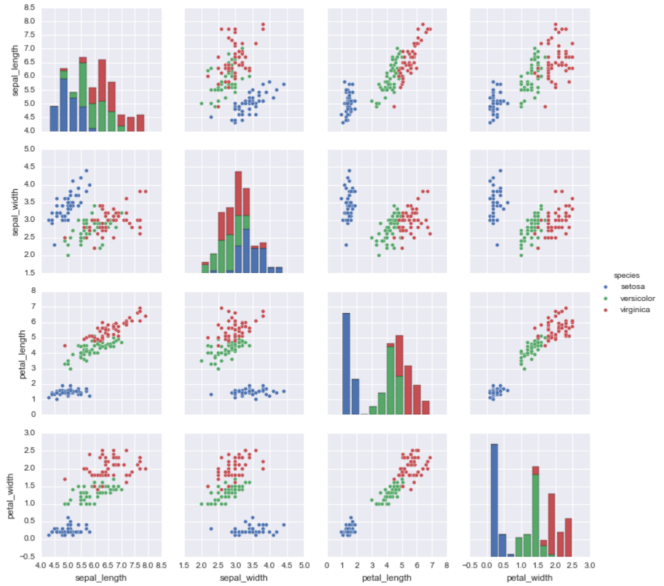
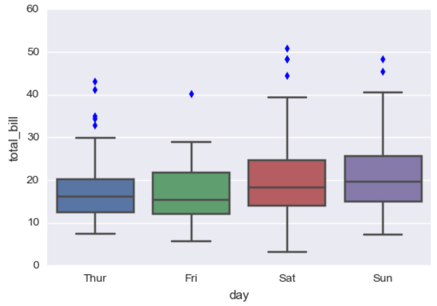
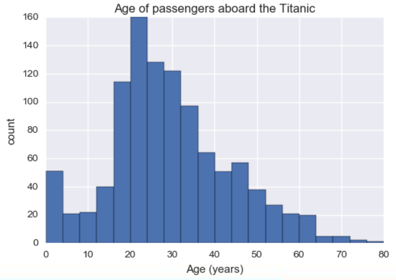
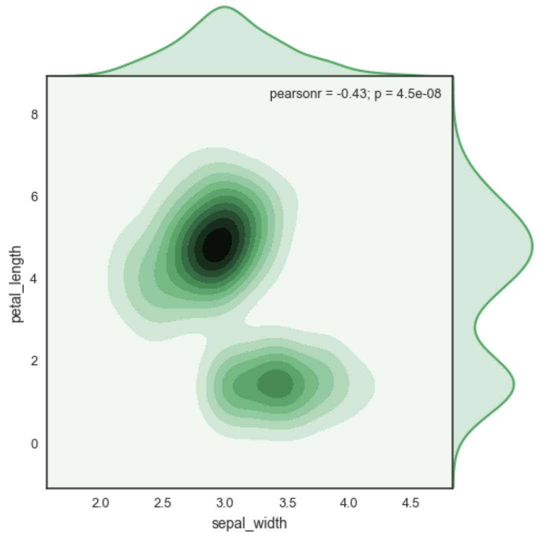
Visualizations like the ones just shown are extremely informative, but it take **significant time** to conceptualize and create.

There are many “off the shelf” visualizations that we can use for routine exploratory and visualization work.

**Statistical graphics** are used to explore the statistical properties of data in a visual way

Statistical data visualizations provide a set of standard visual tools for:

- Understanding the shape and distribution of data
- Understanding the relationships between attributes
- Understanding composition and part-whole relationships
- Comparing quantities



mean(total_bill)
size

tip
total_bill


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

total_bill
Thur
Fri
Sat
Sun
day

Age of passengers aboard the Titanic
count
Age (years)

pearsonr = -0.43; p = 4.5e-08
petal_length
sepal_width

# Software packages for visualization

**Python**

- **Matplotlib**: 2D plotting library with **simple** MATLAB style API
- **Seaborn**: Built on Matplotlib and provides a high-level interface for drawing attractive **statistical** graphics
- **Bokeh**: **interactive** visualization library that targets modern web browsers

**Others**

- ggplot2 (R language)
- d3js (Javascript)
- Processing (Java)

## Topics

- Visualizing the shape and distribution of data
- Visualizing relationships between attributes
- Comparing quantities
- Visualizing composition and part-whole relationships
- Design principles and visualization faux pas

# Visualizing the data distribution

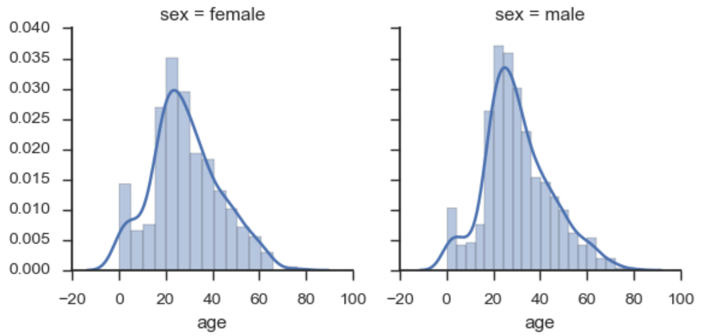
Data distribution: where are the areas in which most of the data lives.

**Shape**

- Is a variable normally distributed?
- Is it multi-modal?
- Are their long tails? Fat tails?
- Is there distribution skewed?

**Modes**

- Where are the modes of the distribution?
- Which values occur most frequently?



**Comparison**

- Are two distributions the same? Different?
- Does conditioning on a variable affect the distribution?

**Outliers**: are there obvious outliers that need to be removed/imputed?

# Histograms

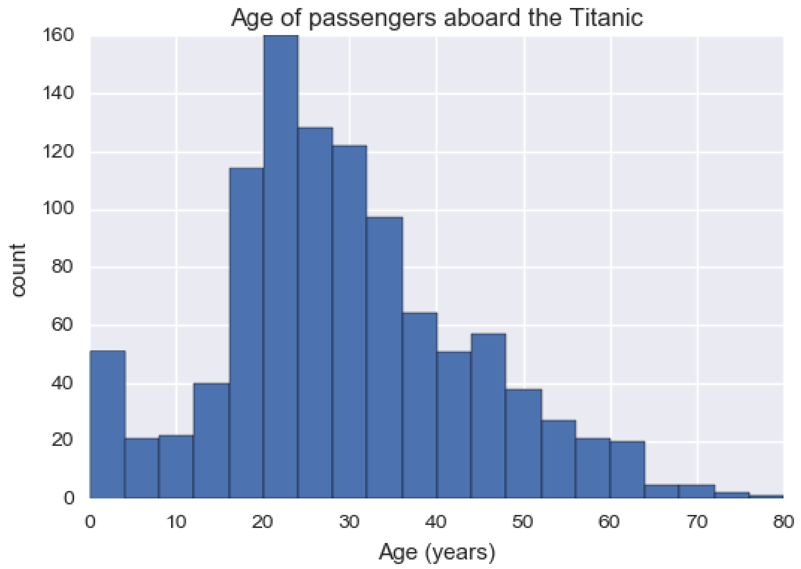
A histogram is a graphical representation of the distribution of numerical data.

Histograms allow you to look at the shape of the distribution of a single quantitative variable

**Construction**: divide the range of values into intervals (**bins**) and count how many values fall in each bin

**Matplotlib:**

```
ages = df['age'].dropna()
hist(ages, bins=20)
xlabel('Age (years)'); ylabel('count')
title('Age of passengers aboard the Titanic')
```

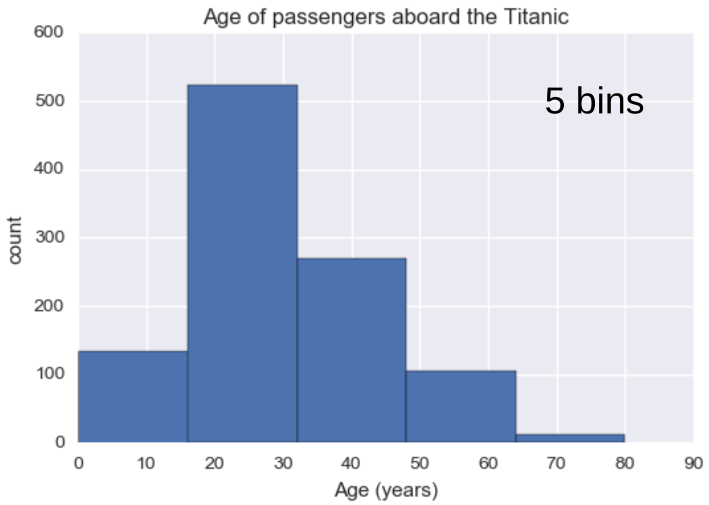


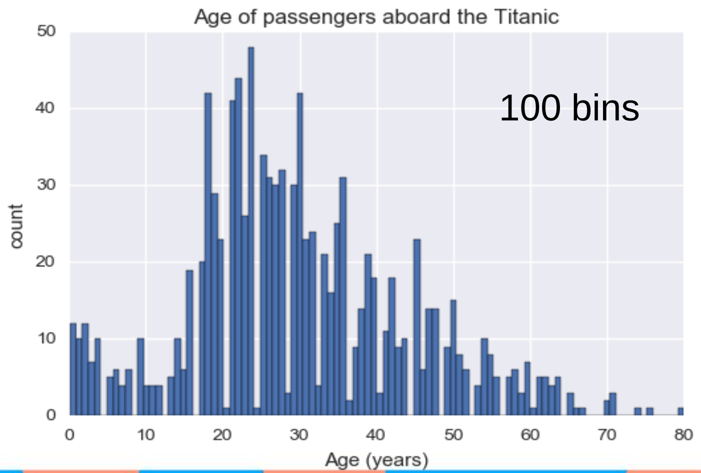
# Histograms

**Uses:**
- Get an idea for the shape of a distribution
- Determine if attribute is unimodal or multimodal
- See maximums and minimums
- Spot outliers

**Choosing the right number of bins is important!**
- Too few fails to capture details in shape
- Too many: lots of empty bins

Usually bins are of fixed width, but not necessarily.





# Kernel density estimators

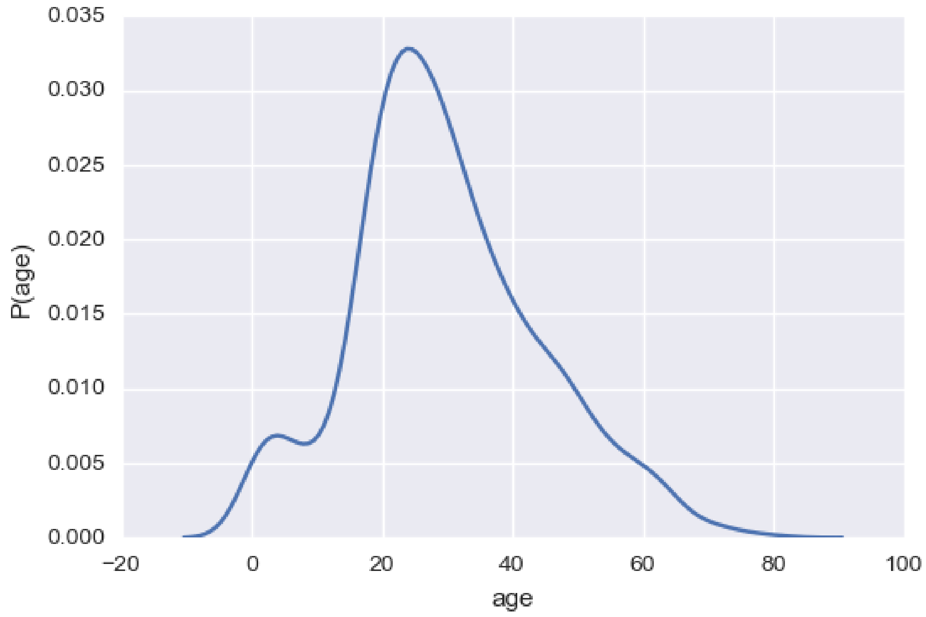
**Basic idea:** estimate the probability density at a point by counting points in local neighbourhood

$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right)$$

**K(.)** is a kernel function, *h* is a bandwidth parameter. K function is smaller for large values and integrates to 1.

**Seaborn**

```
sns.distplot(ages, hist=False)
xlabel('age'); ylabel('P(age)')
```

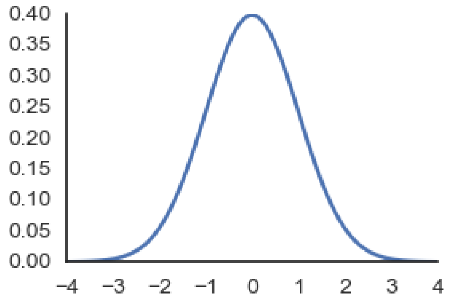


# Kernel density estimators

**Gaussian kernel:**

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp(-0.5x^2)$$

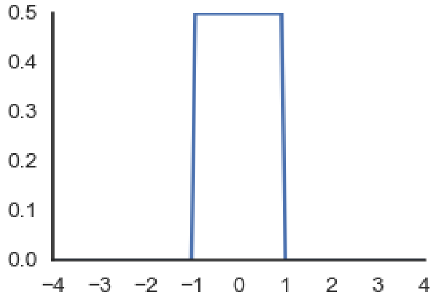
Produces smooth density plots



**Uniform kernel:**

$$K(x) = \frac{1}{2} I(|x| \leq 1)$$

Produces histograms!



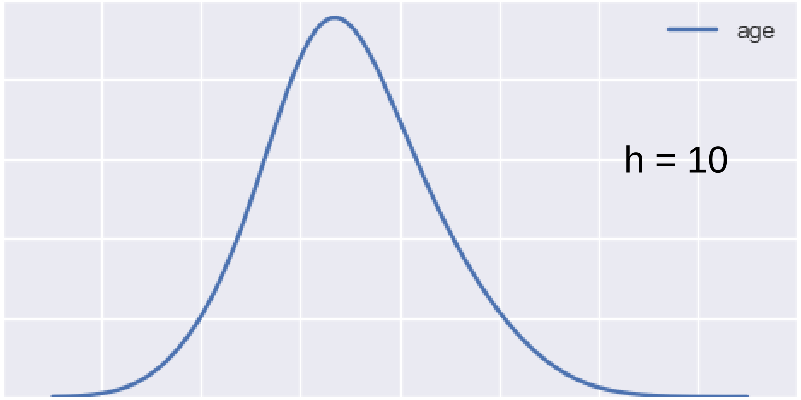
# Kernel density estimators

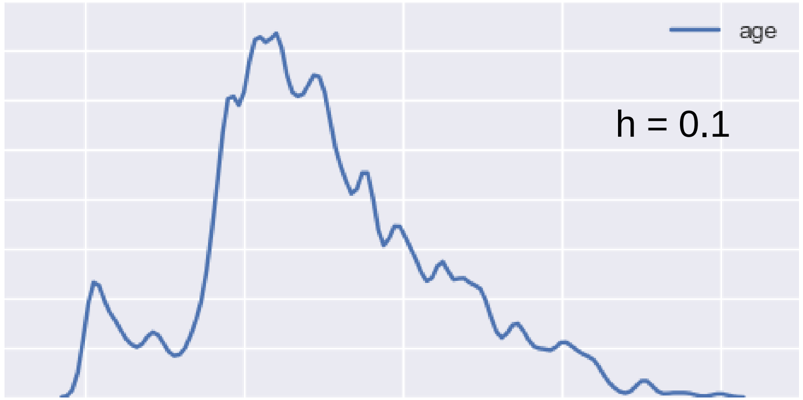
**Histograms:** important to choose number of bins appropriately

**Density plots:** important to choose bandwidth (*h*) properly

- Too large: over smoothing
- Too small: not enough smoothing

```
sns.kdeplot(ages, bw=10)
sns.kdeplot(ages, bw=0.1)
```





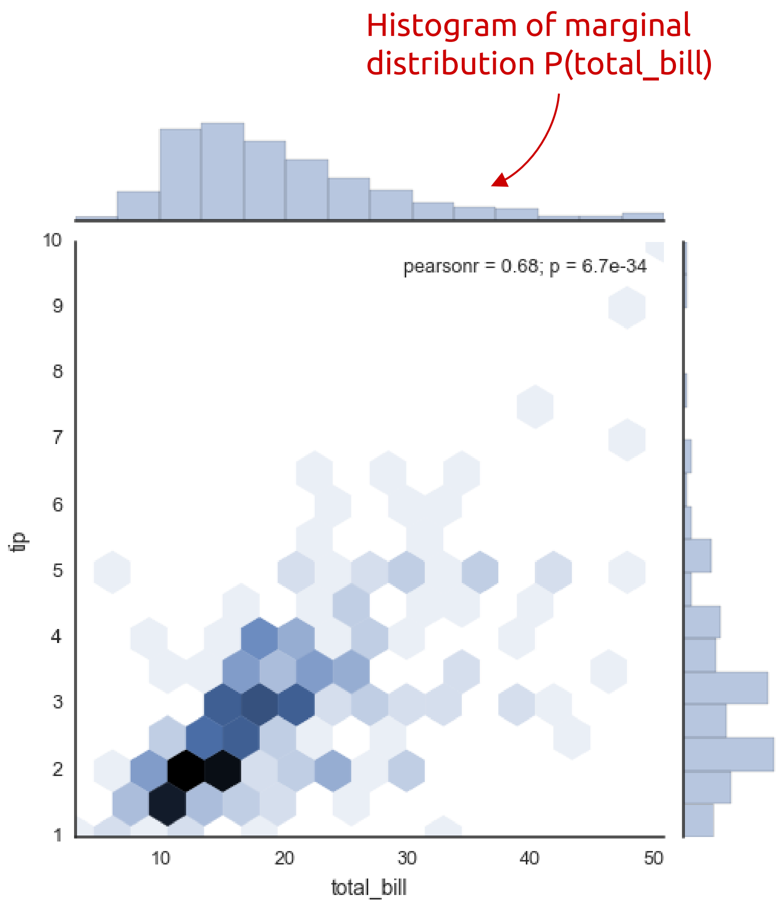
# Joint distributions

Standard histograms and density plots are fine for plotting distribution of single variable.

What if we want to visualize the **joint distribution** of two variables P(x, y)

**Hex plots** are 2D analogies to histograms.

```
tips = sns.load_dataset("tips")

sns.jointplot(
    "total_bill", "tip",
    data=tips, kind="hex")
```

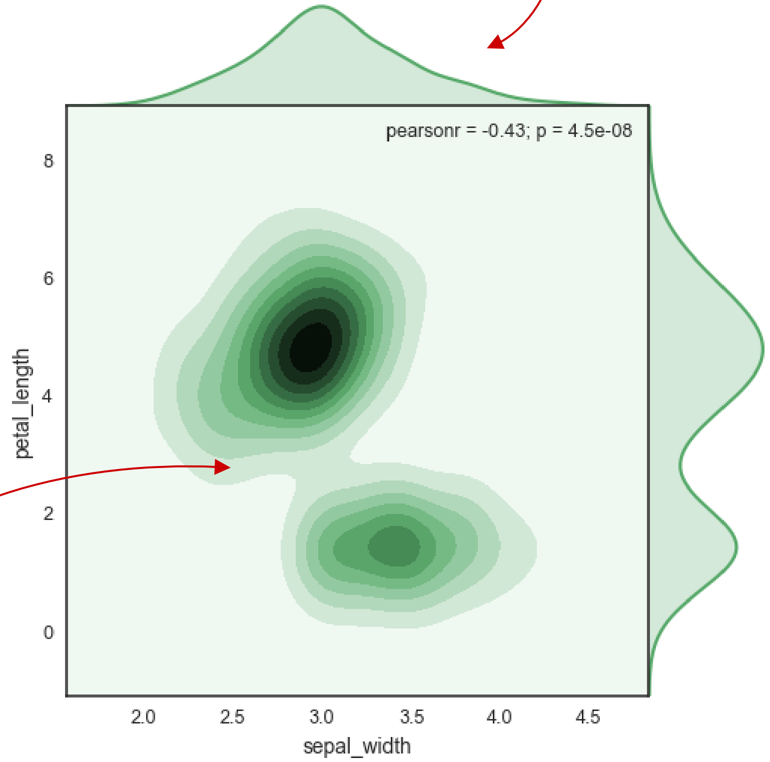


# Joint distributions

Kernel density estimators also possible in 2D

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


sns.jointplot(
    "sepal_width", "petal_length",
    data=iris, kind="kde",
    space=0, color="g")
```

KDE estimate of marginal distribution P(sepal_width)



This part is called a contour plot of P(x,y)

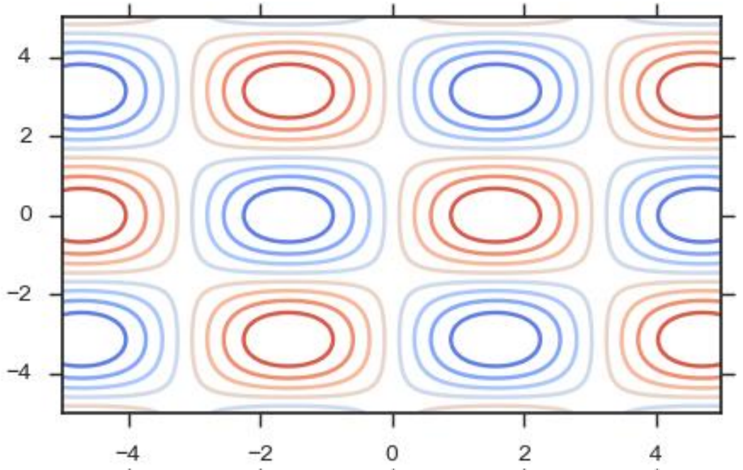
# Contour plots

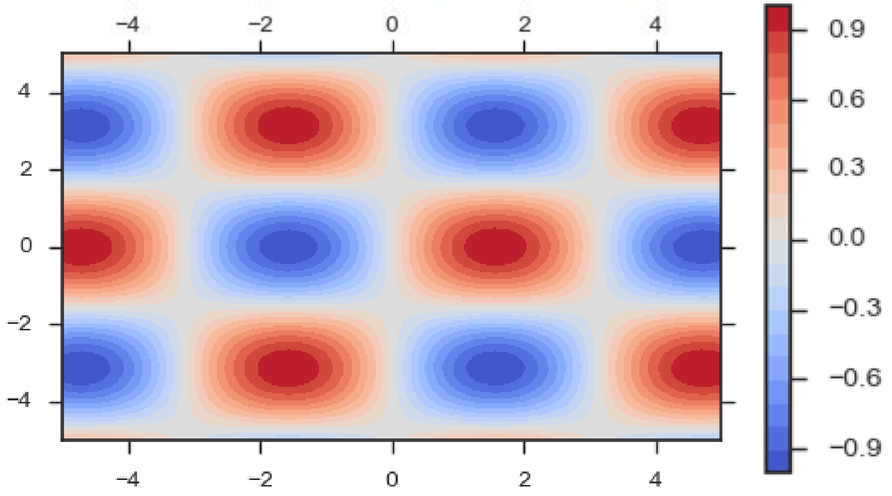
Incidentally, contour plots are not only for probability densities, but can be used to visualize any 2D function *f*(*x*, *y*) → R

E.g.

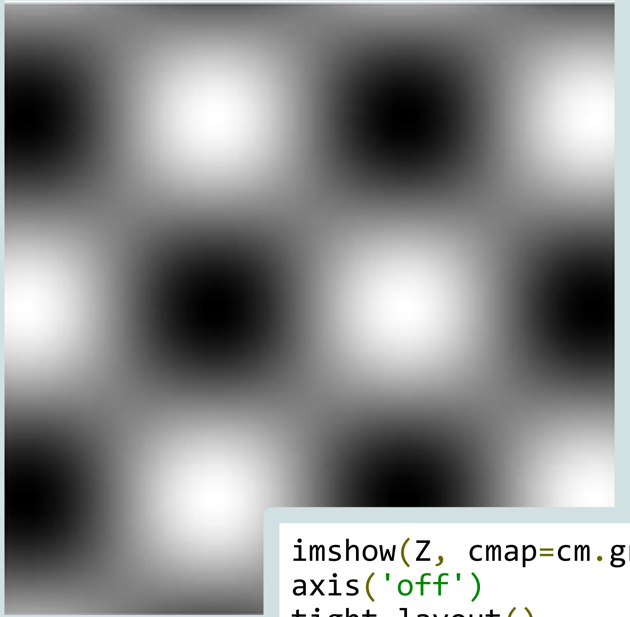
$$f(x, y) = \sin(x)\cos(y)$$

```
# create a grid of x, y values
xs = linspace(-5, 5, 100)
ys = linspace(-5, 5, 100)
X, Y = meshgrid(xs, ys)

# define function and evaluate on grid
f = lambda x, y: np.sin(x) * np.cos(y)
Z = f(X, Y)

# plot filled contours
contourf(X, Y, Z, levels=linspace(-1, 1, 20),
         cmap=cm.coolwarm)
```

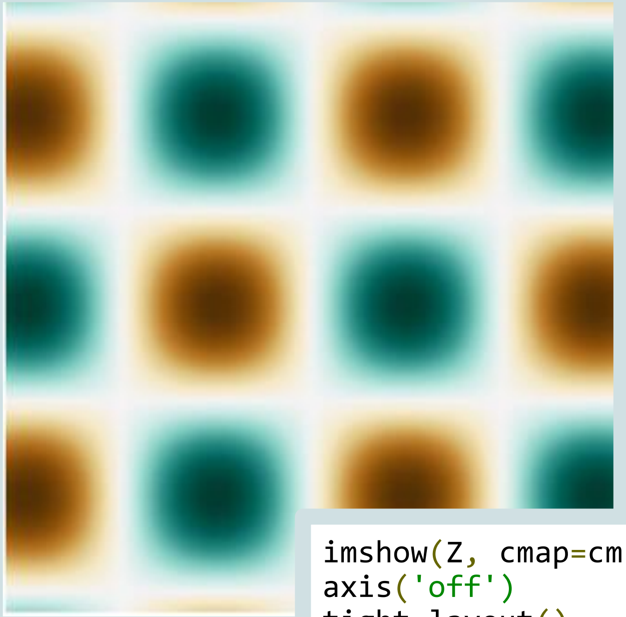




# Using images to visualize functions of two variables



```
imshow(Z, cmap=cm.gray)
axis('off')
tight_layout()
```



```
imshow(Z, cmap=cm.BrBG)
axis('off')
tight_layout()
```

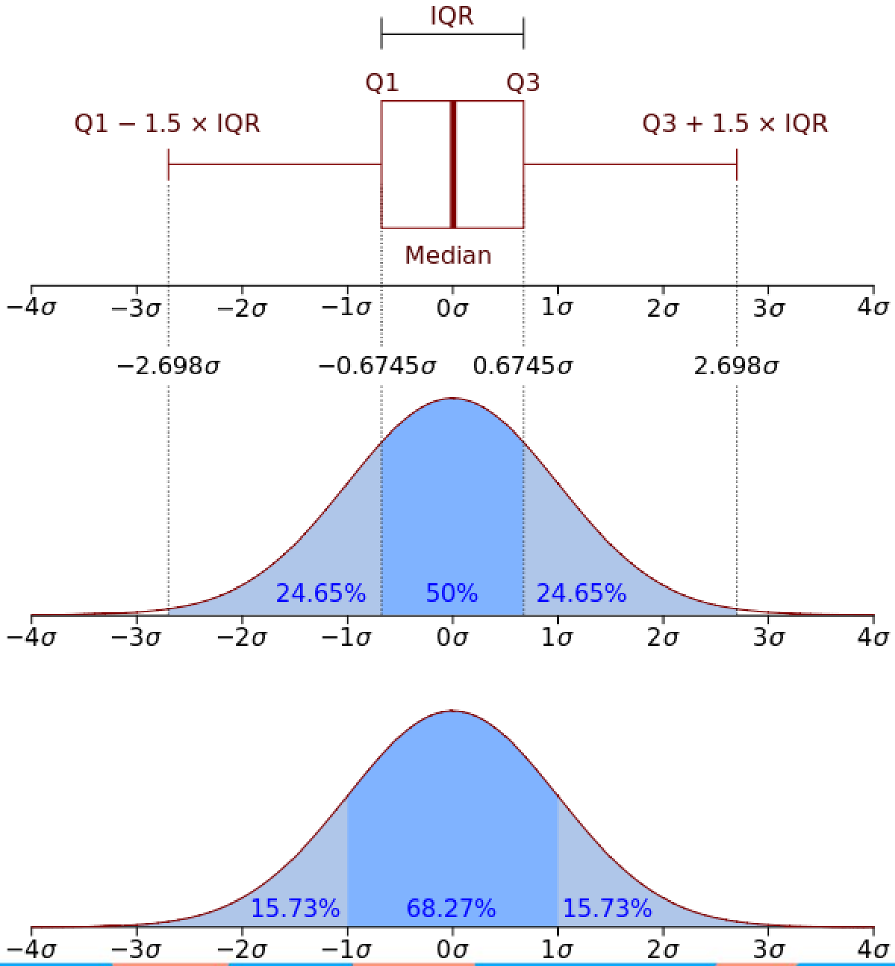
# Boxplots

A **boxplot** (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable.

The box shows the quartiles of the dataset

The whiskers extend to show the rest of the distribution, except for points that are determined to be outliers

The median is shown in the center of the box.

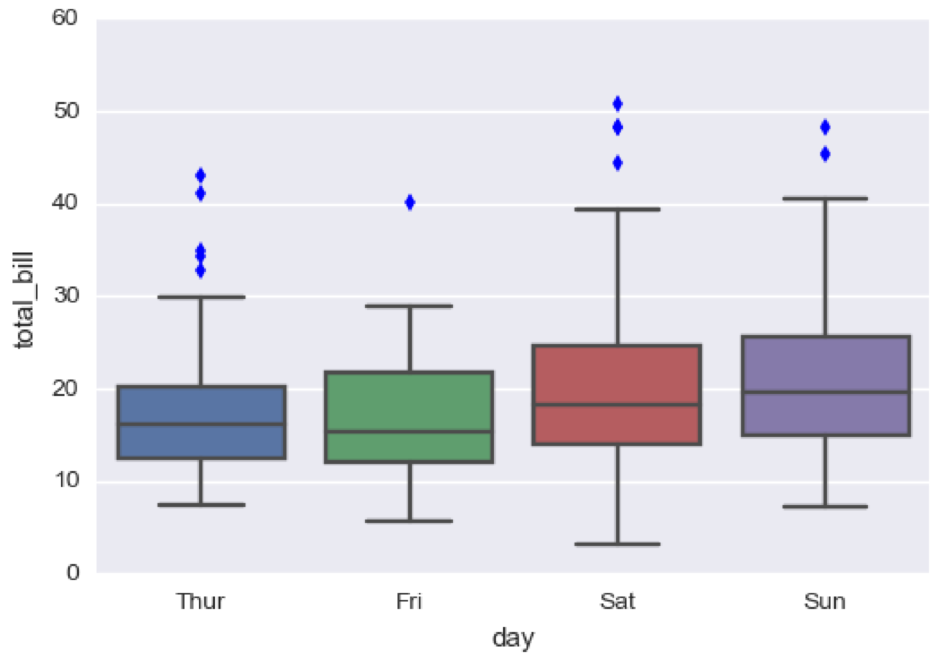


## Boxplots

Especially useful for

1. detecting **outliers**
2. **comparing** distributions conditioned on a categorical (nominal) variable.

```
tips = sns.load_dataset("tips")

sns.boxplot(
    x="day",
    y="total_bill",
    data=tips)
```



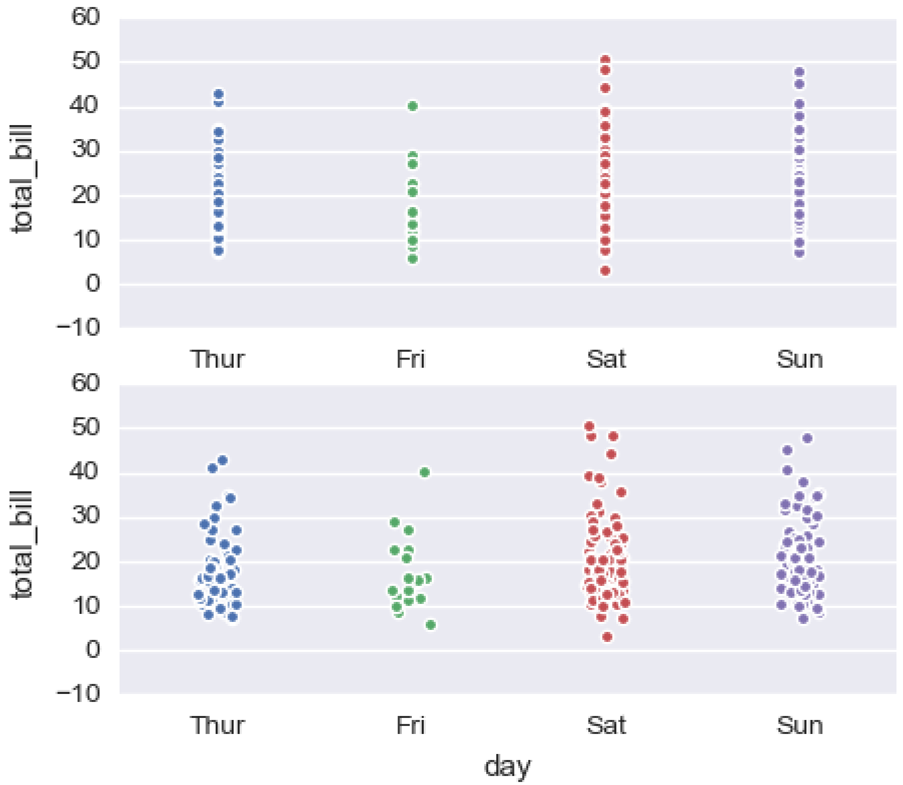
# Strip plots and jitter plots

It is often good to plot the raw data points to ensure that you are not being mislead by factors like the kernel density estimator or skewed distributions.

Plotting the data points on a 1D line often results in **overplotting.**

One technique to alleviate overplotting is to add some noise to the x-coordinate. This is called **jitter**.

```
sns.stripplot(
    x="day", y="total_bill",
    data=tips, jitter=True)
```



## Topics

- Visualizing the shape and distribution of data
- Visualizing relationships between attributes
- Comparing quantities
- Visualizing composition and part-whole relationships
- Design principles and visualization faux pas

# Visualizing relationships between variables

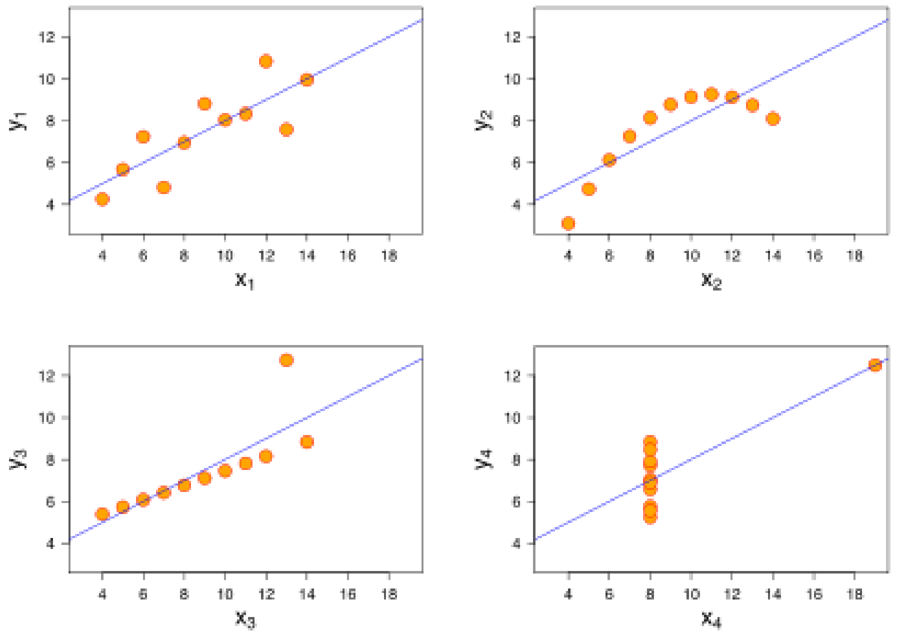
Pearson's and spearman's rho can be used to numerically measure linear and rank correlation.

Visualization is an even more powerful way to examine the relationship between variables:

- Non-linear relationships
- Non-rank order relationships

Some statistical graphics:

1. Scatter plots
2. Regression plots
3. Line plots



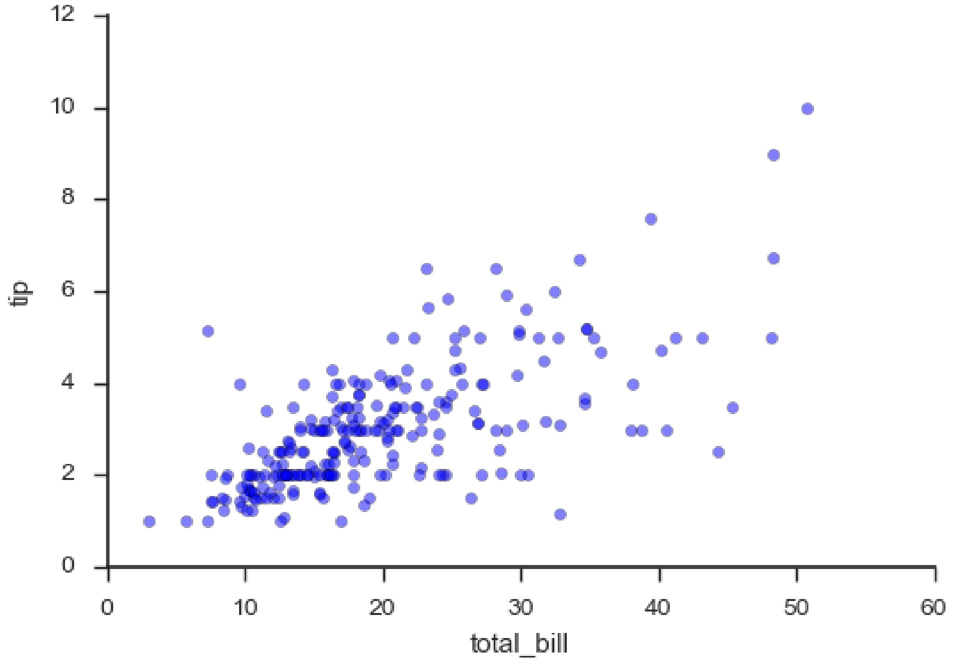
# Scatter plots

Scatter plots (**scattergrams**) show the relationship between two variables by plotting the values of one variable against the other.

If variables are **linearly correlated**, they should appear to cluster around a diagonal line.

If variables are **uncorrelated**, or statistically **independent**, they will appear randomly scattered across the graph.

Non-linear relationships may also be apparent in a scatterplot.

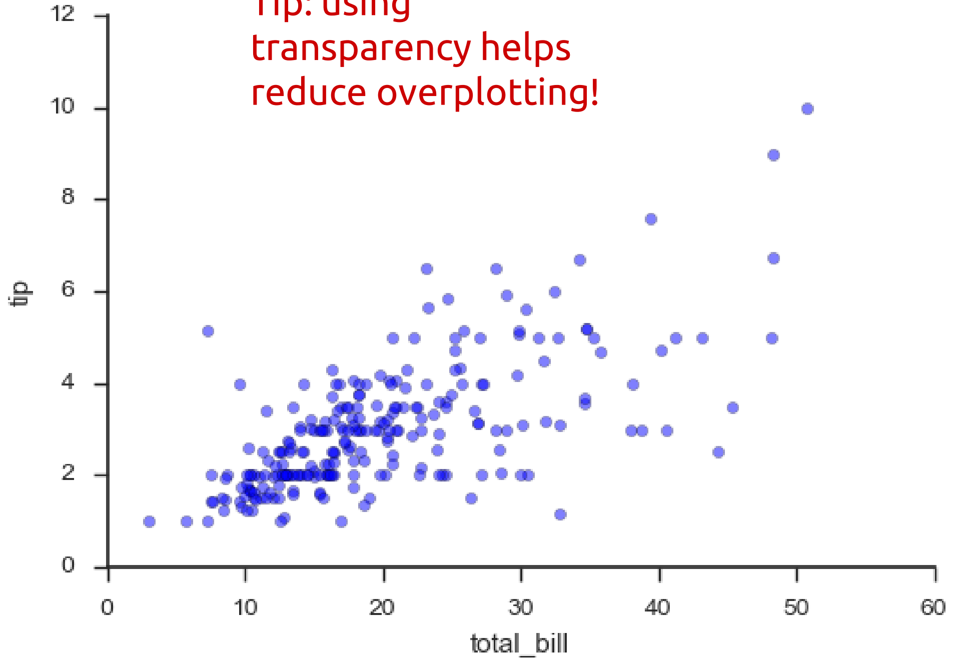


# Scatter plots

**Generating scatter plots with matplotlib (and seaborn for style):**

```
sns.set_style('ticks')
scatter(
   tips['total_bill'],
   tips['tip'], alpha=0.5)

xlabel('total_bill'); ylabel('tip')

# remove lines from box
sns.despine()
```

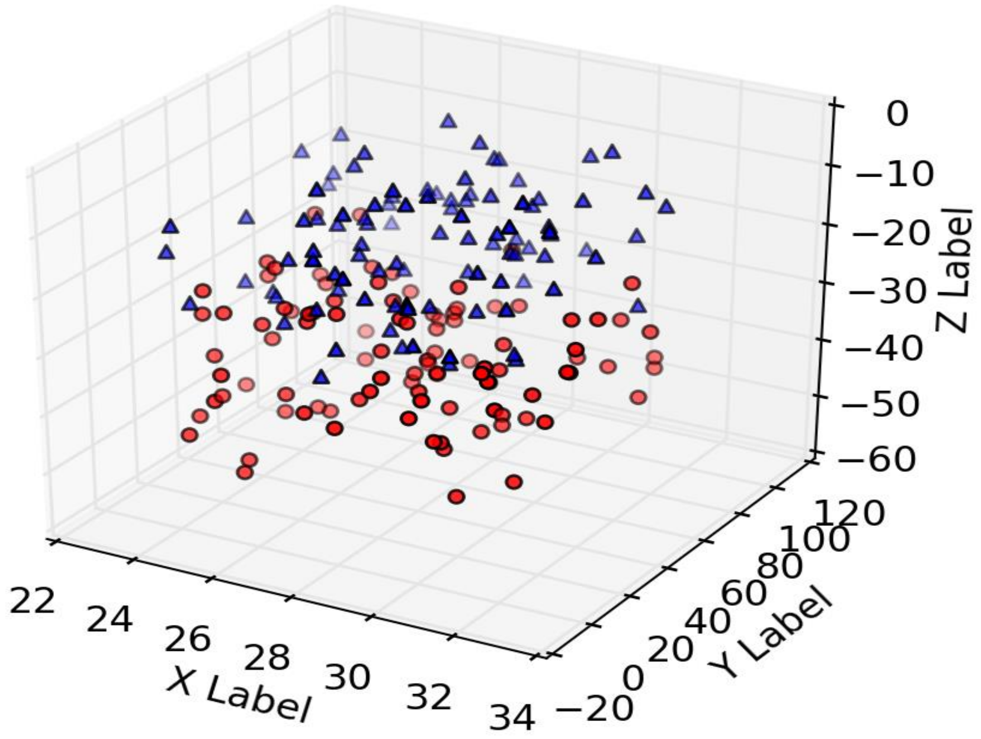
Tip: using transparency helps reduce overplotting!



# 3D scatterplots

Usually not very useful.

**Avoid**.

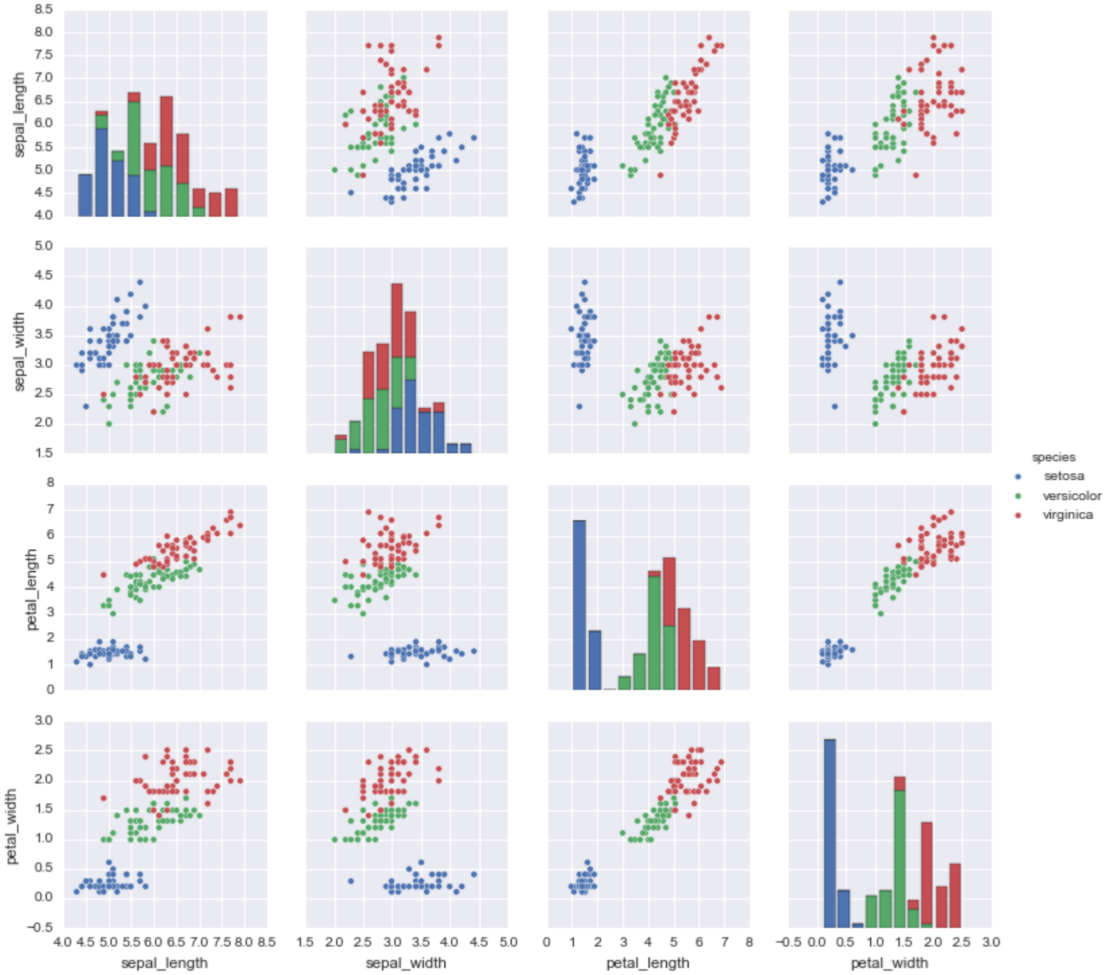


# Scatterplot matrix

If you want to look at the relationships between several pairs of attributes simultaneously, you can use a scatterplot matrix.

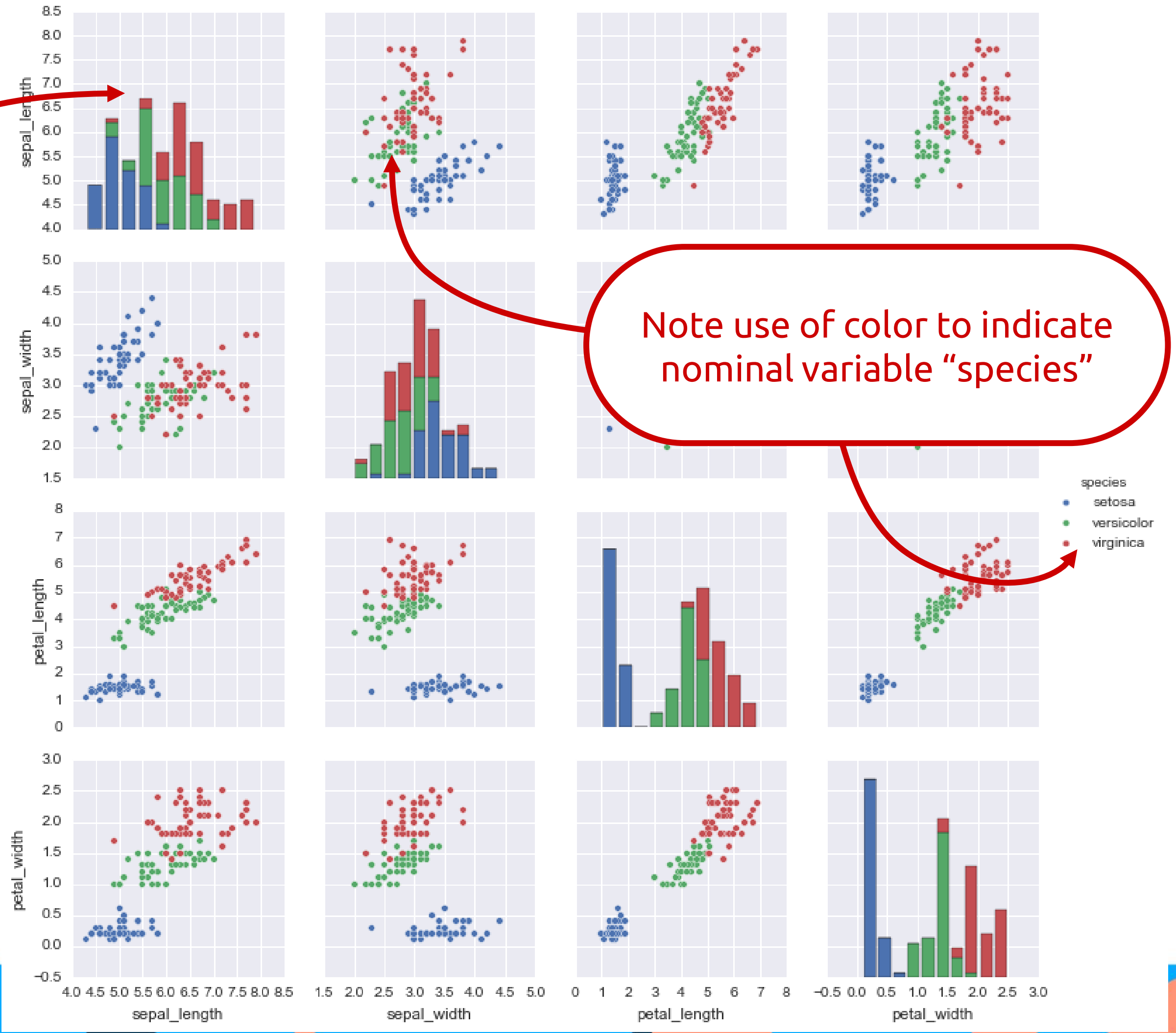
The relationship between variable 1 and 2 is shown in (1, 2) in the matrix. It’s transpose is in (2, 1).

The correlation between any variable and itself is always 1. Plotting packages often use the diagonal of the scatterplot matrix to show additional info.

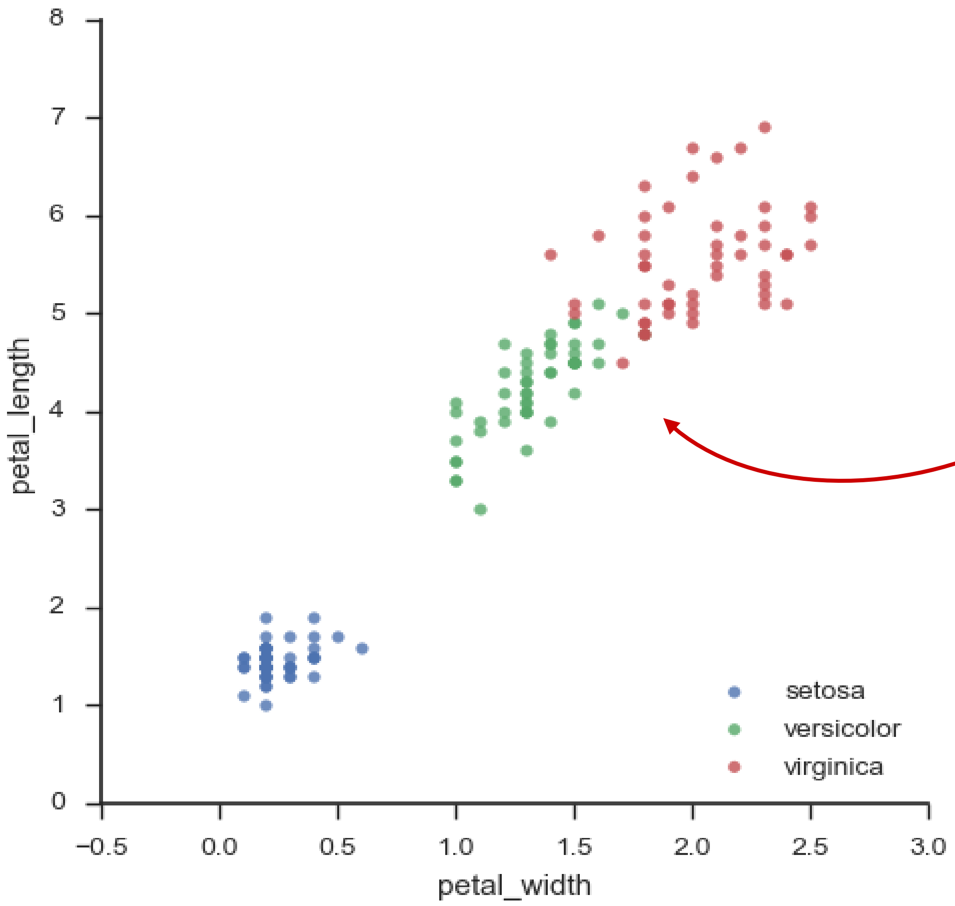


Seaborn shows univariate histograms of variables on the diagonal by default

```
iris =
sns.load_dataset('iris')
sns.pairplot(
    iris, hue='species')
```



Note use of color to indicate nominal variable “species”



Notice anything strange about the data in this graph?

Values are quantized!

```
iris = sns.load_dataset('iris')
sns.lmplot(
    "petal_width", "petal_length", data=iris,
    hue='species', fit_reg=False, legend=False)
legend(loc='lower right')
```

# On the use of color

Color can be an incredibly useful tool for including more dimensions in a 2D plot.

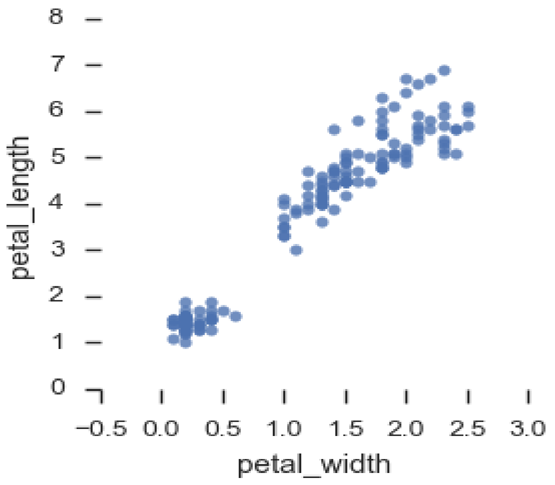
**Tufte**: color can be your greatest ally or worst enemy if used improperly

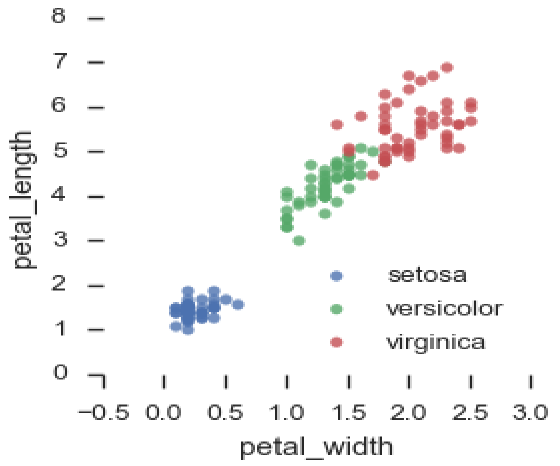
Example on right uses different colors to overlay the nominal variable species on the scatter plot.

The choice of color palette used is important, and should reflect the type of variable being indicated and what you want to show.

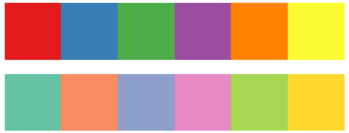
Recall Stephen's scales of measurement:

1. Nominal (categorical)
2. Ordinal (ordered)
3. Quantitative (ratio or interval)

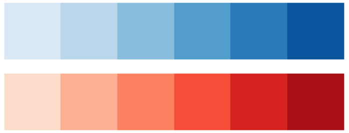




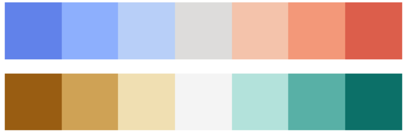
## Choosing a color palette



**Qualitative palettes**
Do not imply an inherent ordering. Suitable for nominal variables. E.g. species, gender, category, etc.

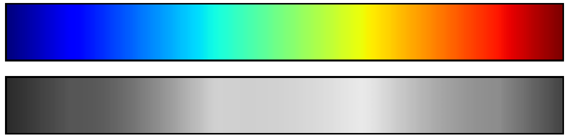


**Sequential palettes**
Imply an ordering. Appropriate when data range from relatively **low or uninteresting** values to relatively **high or interesting** values. Suitable for ordinal and quantitative variables. E.g. risk, crime rate



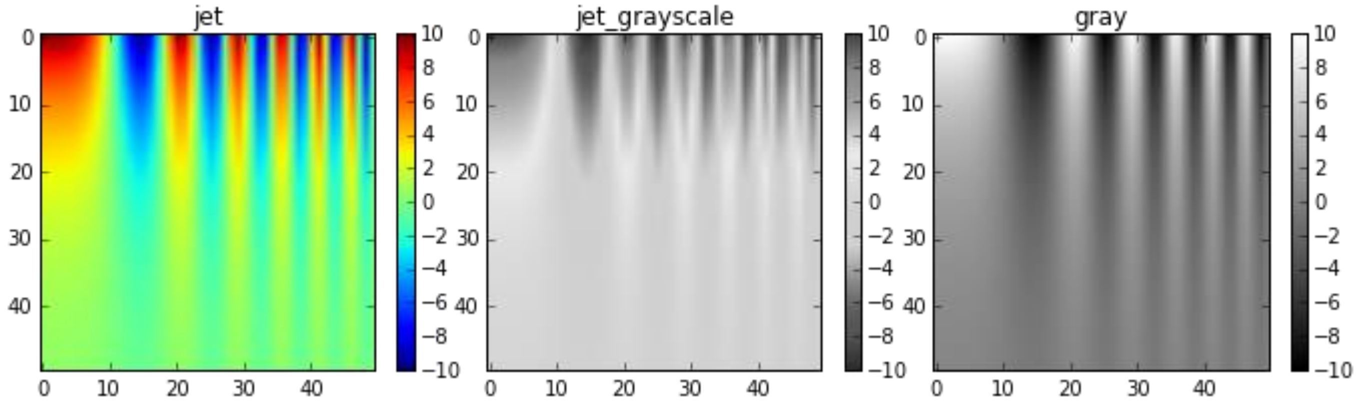
**Diverging palettes**
Also imply an ordering and suitable for ordinal and quantitative variables. Used for data where both l**arge low and high values are interesting**. Interpretable **midpoint** in the data. E.g. temperature, rainfall

# Avoid “jet” and “rainbow”

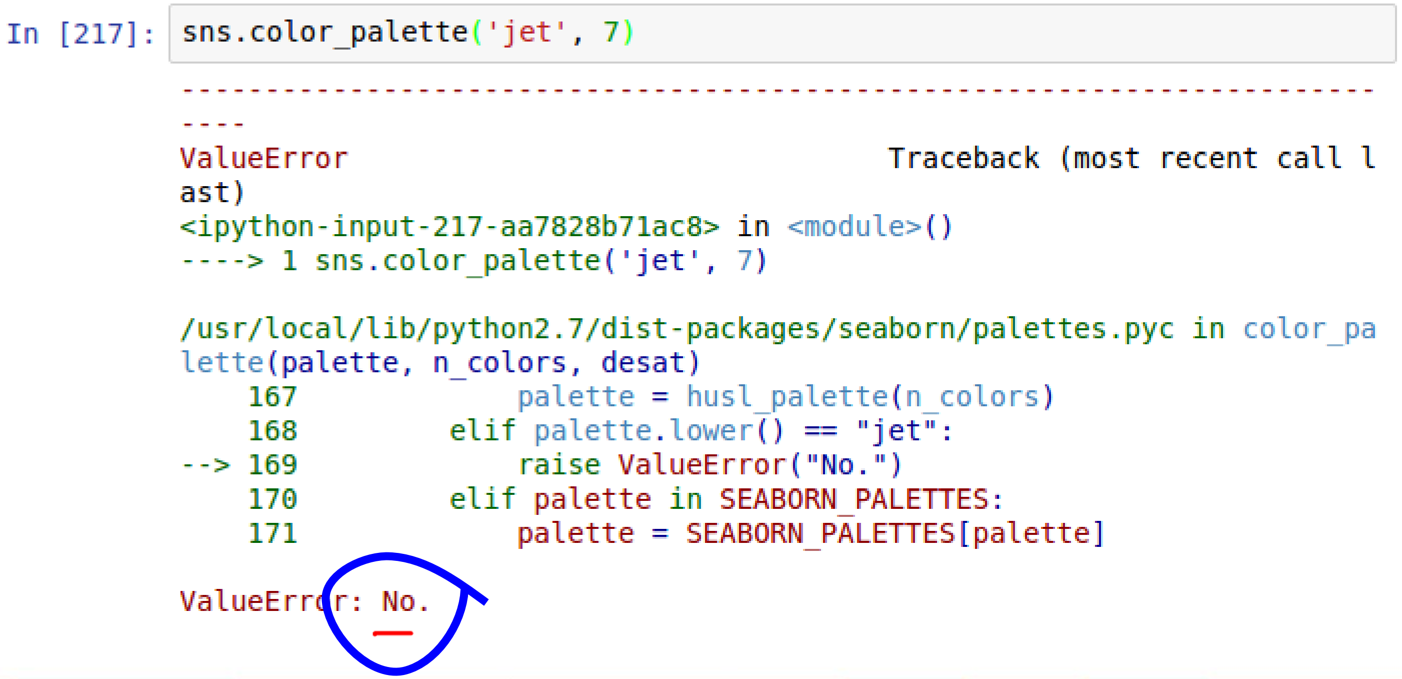


Jet is (or was) the default color bar originally used by MATLAB.

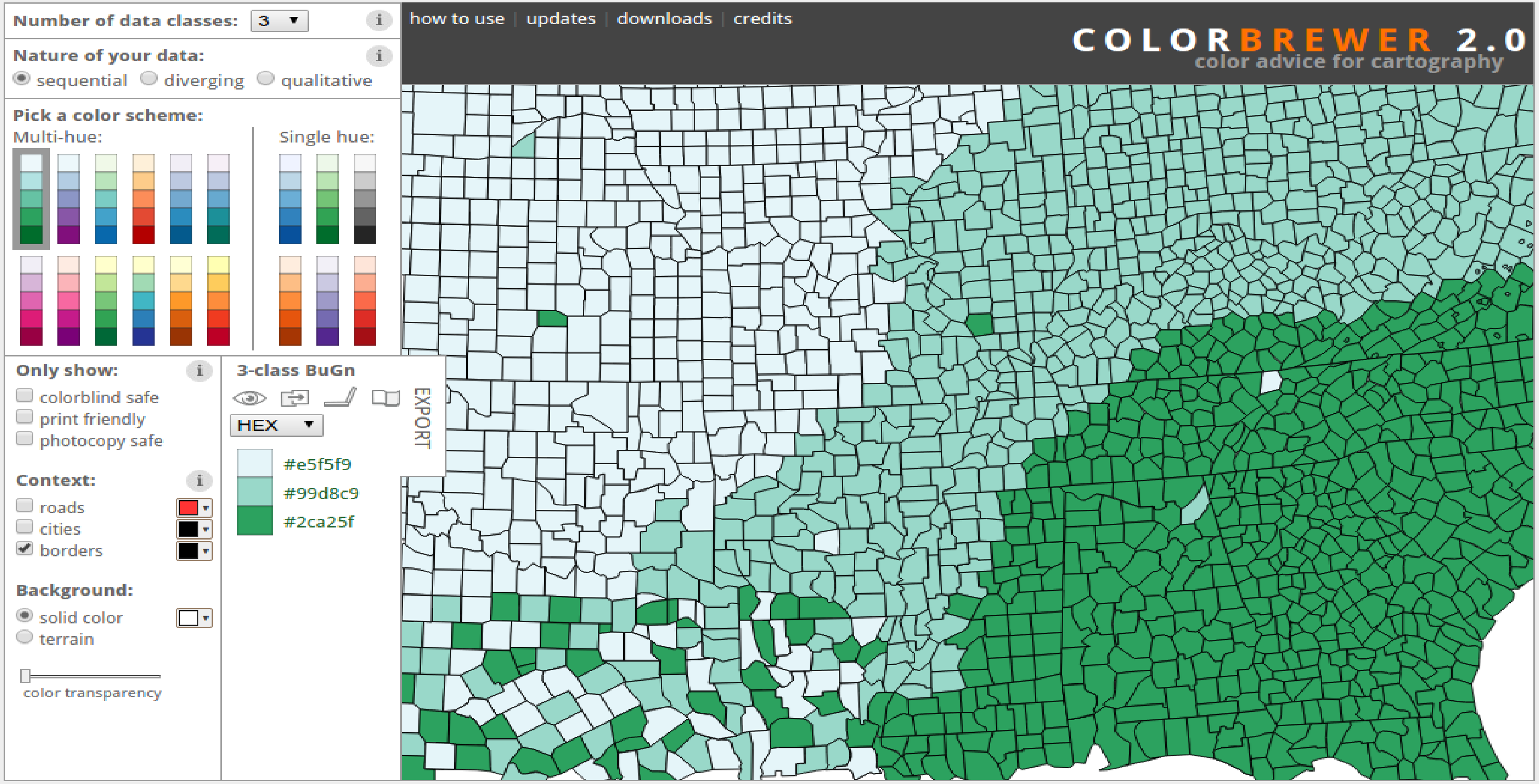
Complex luminance profile tends to introduce artifacts that are not really there! Highlights features that may not exist.



# Avoid “jet” and “rainbow”



```
In [217]: sns.color_palette('jet', 7)

          ---------------------------------------------------------------------------
          ----
          ValueError                                Traceback (most recent call l
          ast)
          <ipython-input-217-aa7828b71ac8> in <module>()
          ----> 1 sns.color_palette('jet', 7)

          /usr/local/lib/python2.7/dist-packages/seaborn/palettes.pyc in color_pa
          lette(palette, n_colors, desat)
              167             palette = husl_palette(n_colors)
              168         elif palette.lower() == "jet":
          --> 169             raise ValueError("No.")
              170         elif palette in SEABORN_PALETTES:
              171             palette = SEABORN_PALETTES[palette]

          ValueError: No.
```


how to use | updates | downloads | credits
COLORBREWER 2.0
color advice for cartography
EXPORT

Number of data classes: 3

Nature of your data:
[✓ sequential] diverging qualitative

Pick a color scheme:
Multi-hue: Single hue:

Only show:
☐ colorblind safe
☐ print friendly
☐ photocopy safe

Context:
☐ roads
☐ cities
☑ borders

Background:
[✓ solid color]
terrain

color transparency

3-class BuGn

HEX

| | |
|---|---|
| ⬜ | #e5f5f9 |
| 🟩 | #99d8c9 |
| 🟩 | #2ca25f |

# Regression lines

When we think there is a linear relationship between two variables (x, y), we may want to plot a regression line to illustrate it.

$$y = (ax + b) + \epsilon$$

- *y* is the **target** variable (aka. dependant variable)
- *x* is the **predictor** (feature, independent variable)
- *a, b* are **parameters**
- *epsilon* is **noise** (assume iid Gaussian)

To do this we can find the parameters (a, b) that minimize the sum of squared errors:

$$\sum_{i=1}^{N} (y_i - (ax_i + b))^2$$

The parameters can be found using a little calculus and algebra. Compute deriv wrt. (a, b) and set to zero and solve linear system.

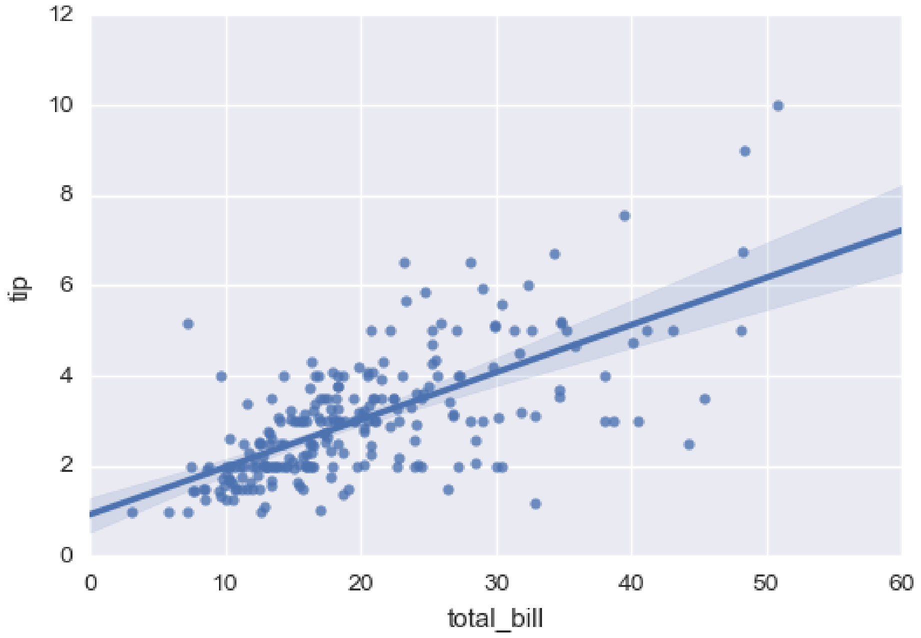
We'll see how to do this in more detail when we study **supervised learning** later...

# Regression lines

Seaborn can be used to plot regression lines.

```
sns.regplot('total_bill', 'tip', data=tips)
```

By default, it will also plot **confidence intervals** too (estimated using bootstrap)

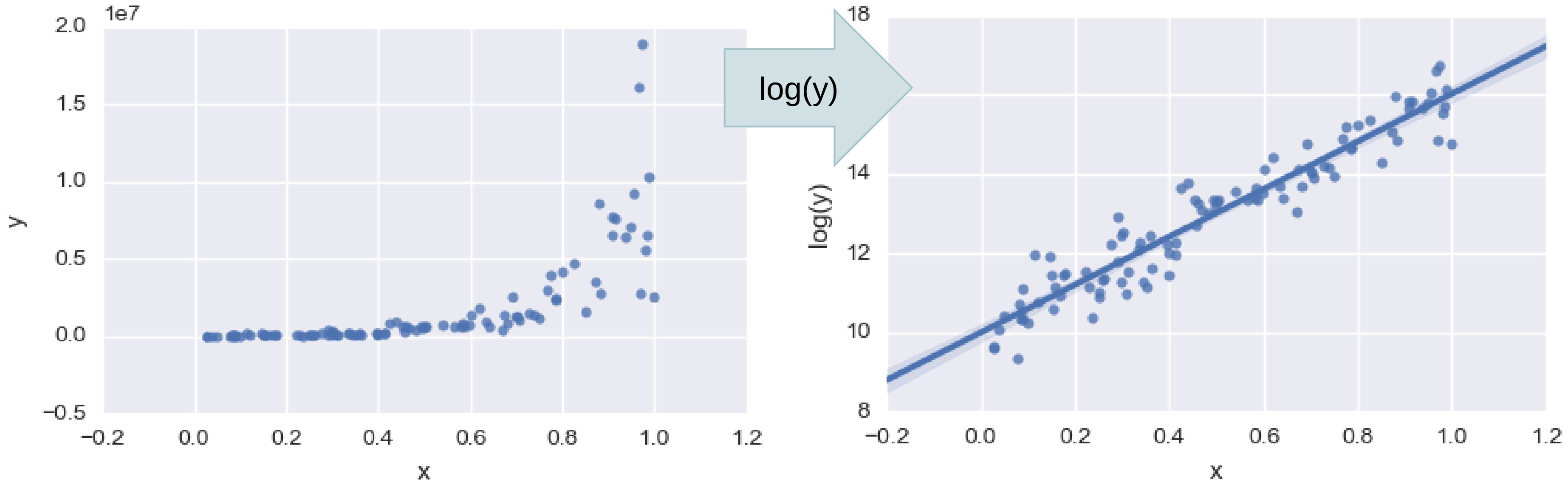


**The bootstrap** is a non-parametric way of estimating confidence intervals.

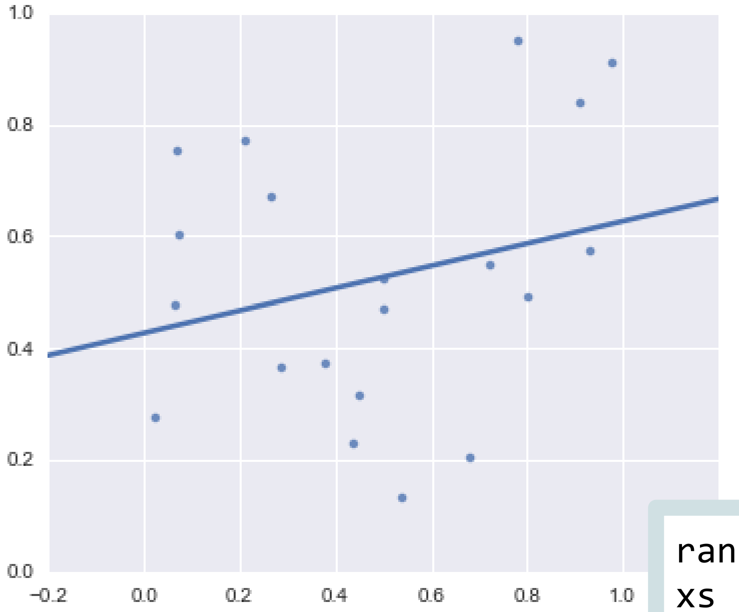
Idea: sample data with replacement 100 times. Each time fit a regression line. Look at variation in fits

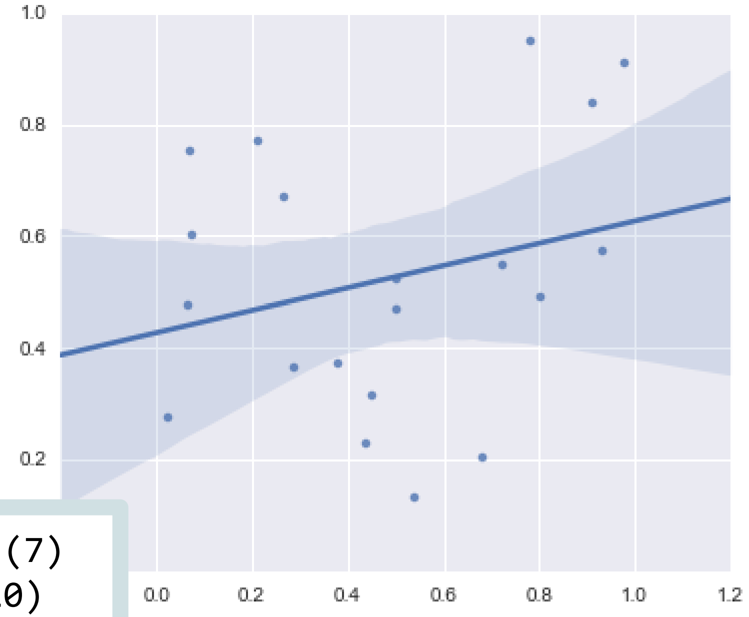
# Axes scales

Sometimes data appears correlated, but not linearly. If $y = \exp(ax + b)$ + noise, then changing the axes scale from linear to logarithmic allows a linear fit.



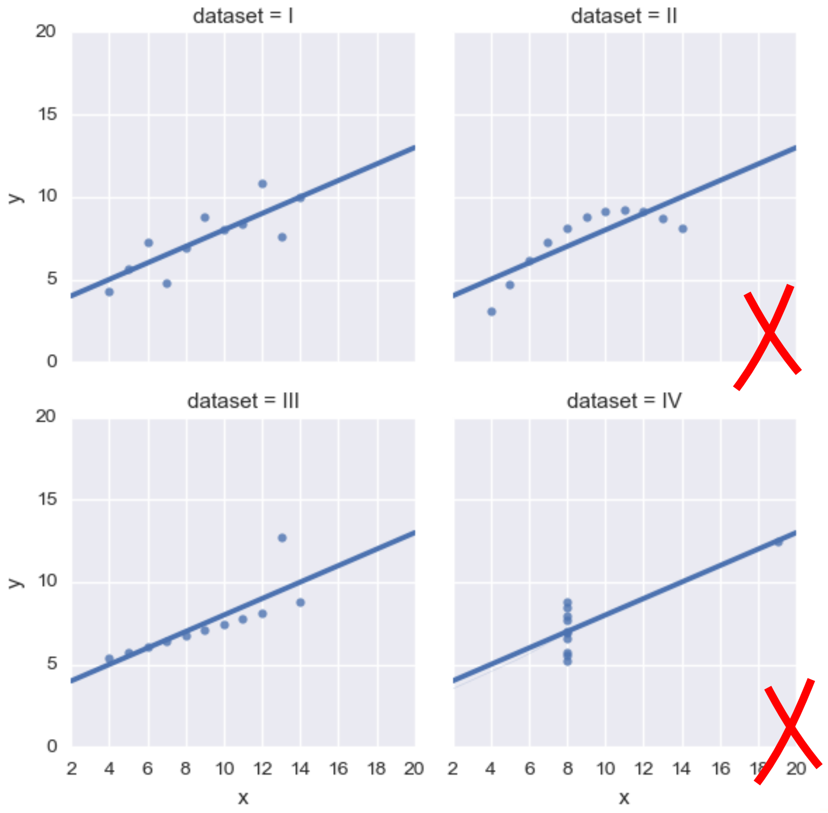
# Inappropriate uses of regression plots

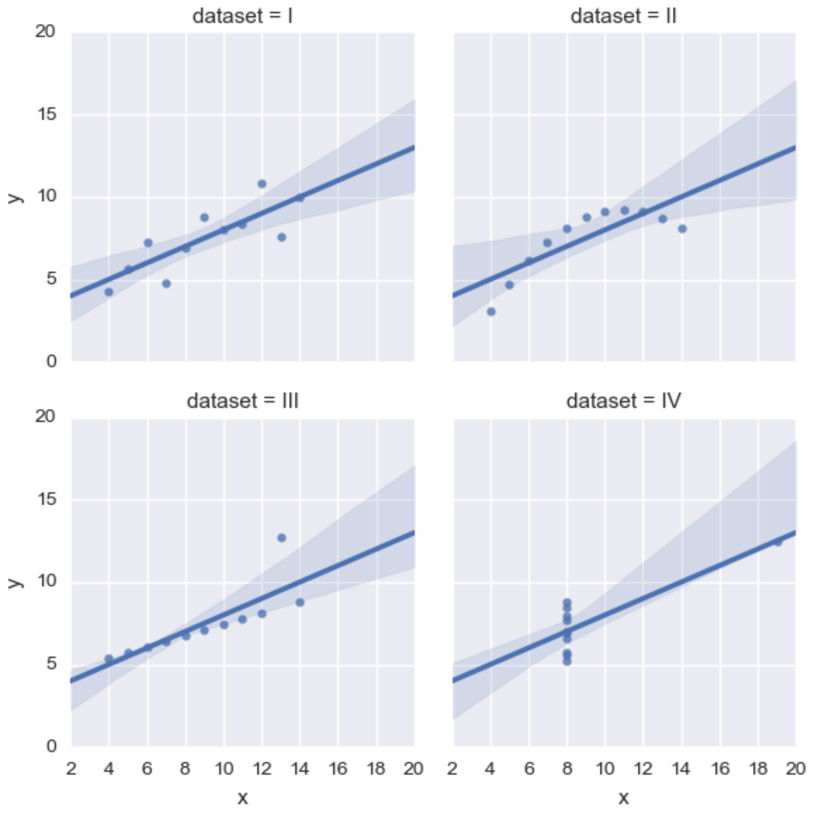




```
random.seed(7)
xs = rand(20)
ys = rand(20)
```

# Anscombe's quartet





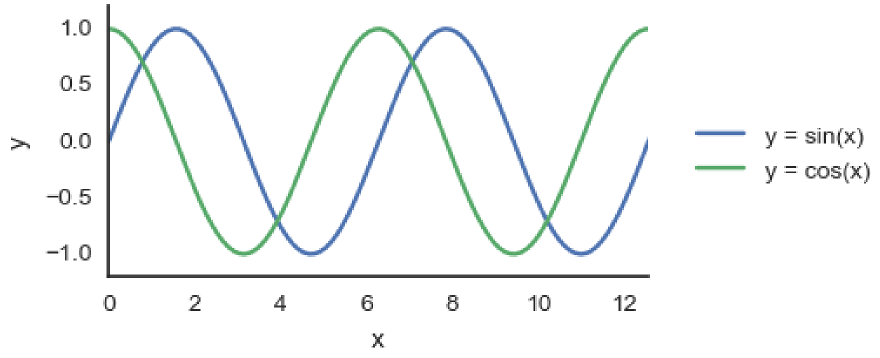
# Line graphs

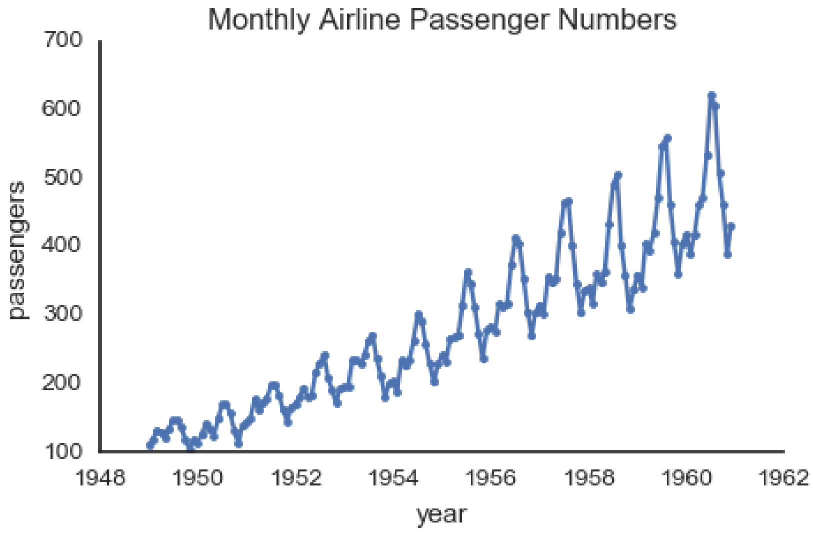
Line graphs are appropriate for plotting one variable as a function of another **continuous ordered** variable.

Often *x* is **time** and *y* is some quantity we wish to study over time. This is called a **time series**.

Data points are connected by lines. It’s good practice to include **markers** for the data points.

```
air = pd.read_csv('data/AirPassengers.csv')
figure(figsize=(5,3))
plot(air.time, air.AirPassengers, marker='.')
sns.despine()
xlabel('year')
ylabel('passengers')
title('Monthly Airline Passenger Numbers')
```





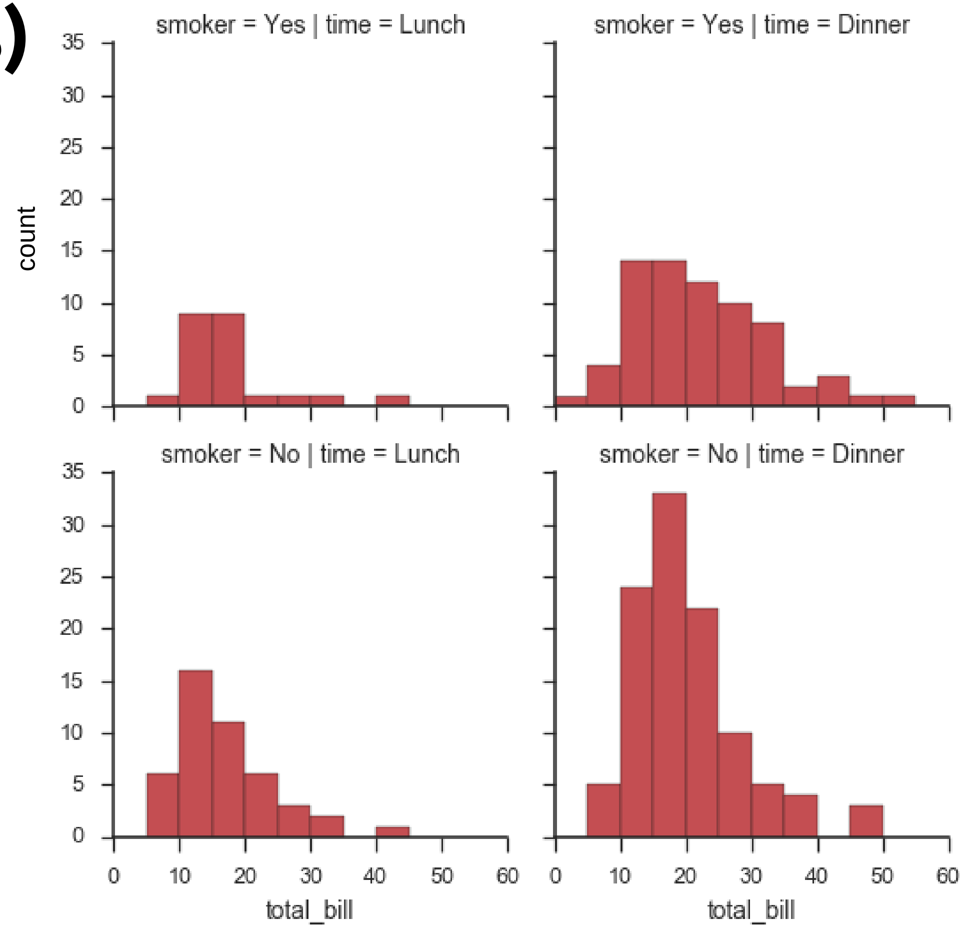
# Facet grids (lattice plots)

Visualize **multivariate relationships** by displaying relationship between one or two variables **conditioned on other nominal variables**.

I.e. group data by facets (nominal variables) and plot groups on a grid.

If variable being conditioned upon is not nominal, it can be quantized into intervals.

```
g = sns.FacetGrid(
    tips, col="time",  row="smoker")
g.map(plt.hist, "total_bill", bins=bins)
```



## Topics

- Visualizing the shape and distribution of data
- Visualizing relationships between attributes
- Comparing quantities
- Visualizing composition and part-whole relationships
- Design principles and visualization faux pas

## Comparing quantities

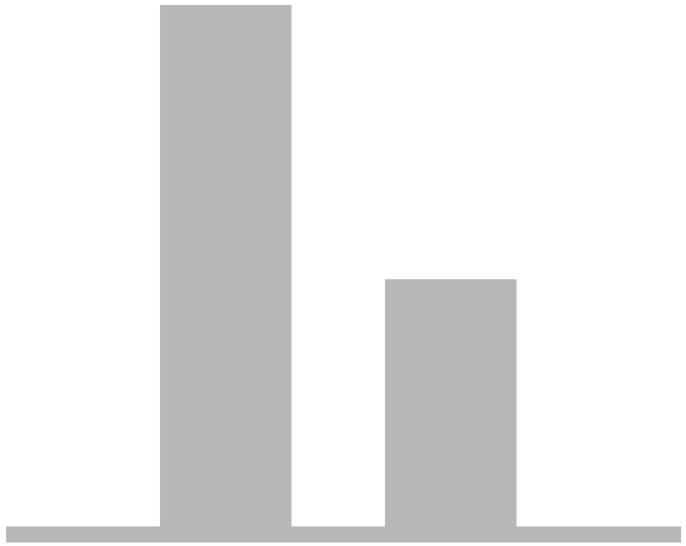
Compare quantities conditioned on another variable

Typical quantities:

- Mean
- Variance

E.g. accuracy given algorithm, height given gender, etc.

Time series: quantity given time.



## Bar charts

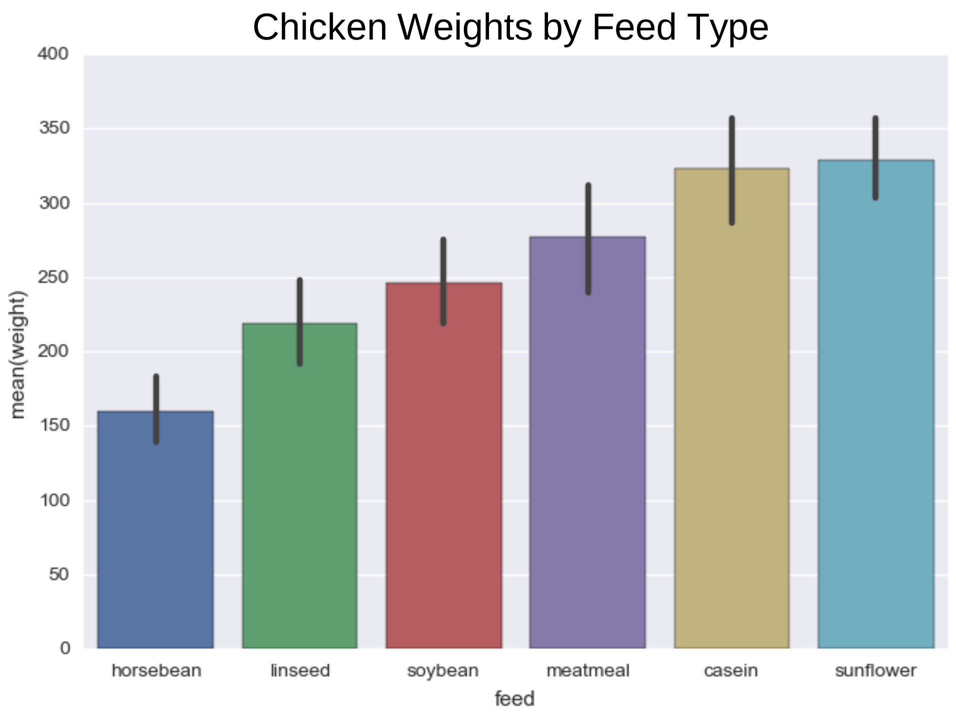
Use for comparing quantities conditioned on a categorical variable.

When quantities represent averages, confidence intervals can be also be plotted.

Seaborn uses bootstrapping to estimate confidence intervals.

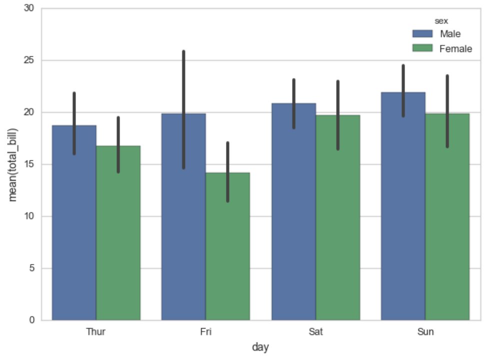
Bars be plotted horizontally or vertically. Horizontal bars better when there are many categories.

```
sns.barplot('feed', 'weight', data=chicken)
```



# Bars chart tips

- If it makes sense to **interpolate** between the bars (quantitative variables), use a line chart. If not, use a bar chart.
- Since categorical variables have no inherent order, **sort** categories by bar height.
- When there are only a few variables and confidence intervals don’t matter, prefer a small table to a bar chart.
- When conditioning on another variable, color can be used and bars can be grouped



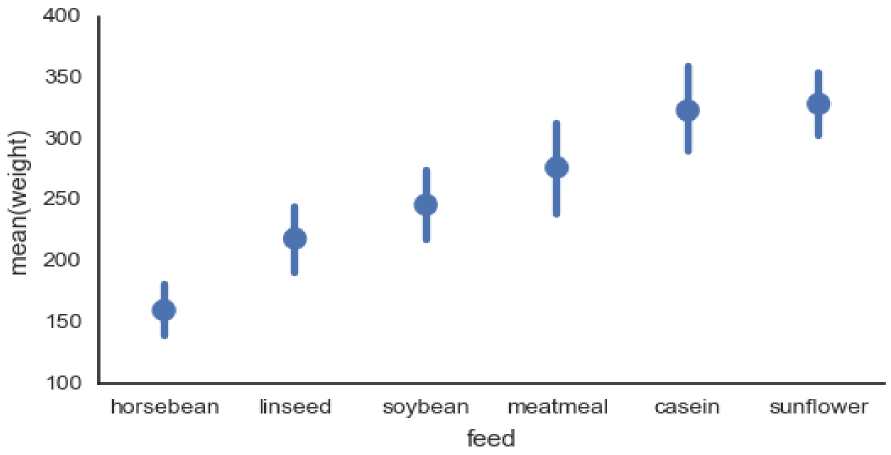
# Dot/point plots

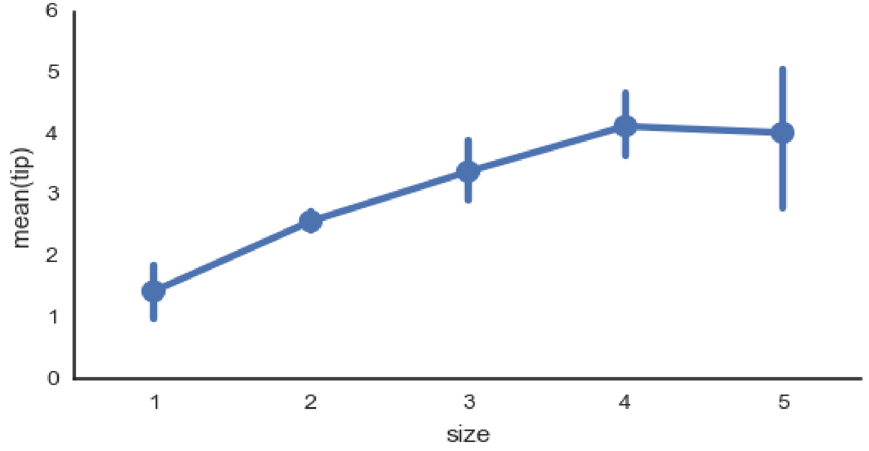
Similar use to bar charts. Save ink!

More appropriate when there is no meaningful zero (interval measurements)

Can include a line when it makes sense to interpolate between values on x-axis.

```
sns.pointplot(
    'size', 'tip',
    data=tips, order=range(1,6))
```





## Topics

- Visualizing the shape and distribution of data
- Visualizing relationships between attributes
- Comparing quantities
- Visualizing composition and part-whole relationships
- Design principles and visualization faux pas

## Visualizing composition

Visualizing composition of some "whole" as a set of parts.

Very often **percentages.**

Several widely used types of plots, but often it's better just to use a bar plot!

## Pie charts

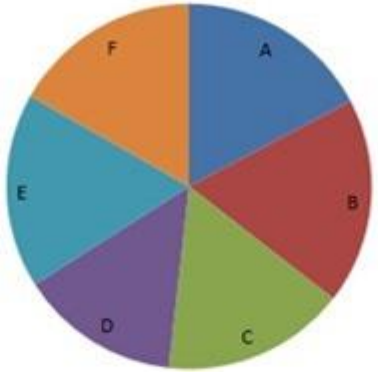
A pie chart is a statistical graphic that illustrates numerical proportion using the arc lengths of the slices.

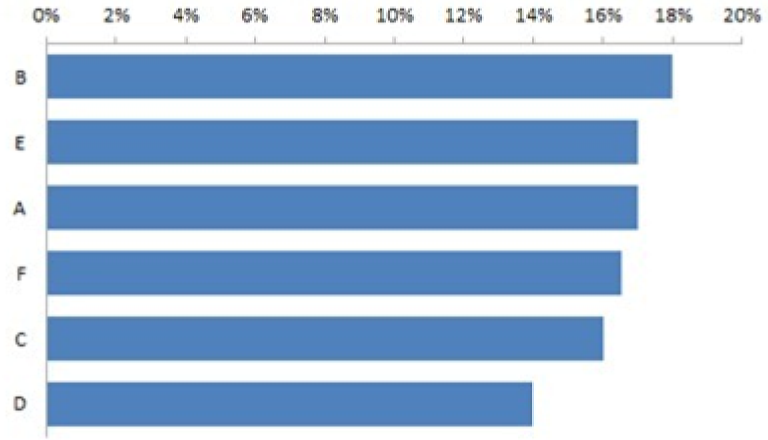
Widely used in business world and mass media.

**Disadvantages**:

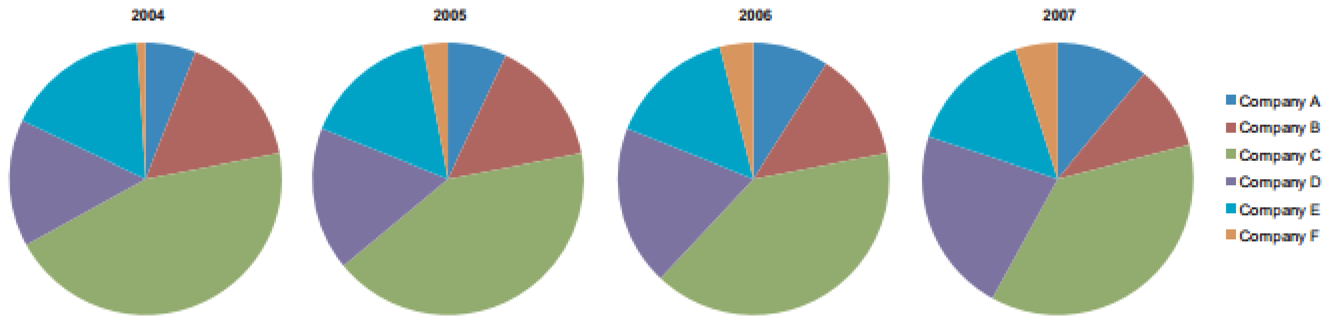
- Can only show small number of values
- Take up more space than bar charts
- **Humans are poor at comparing arc lengths and slice areas**

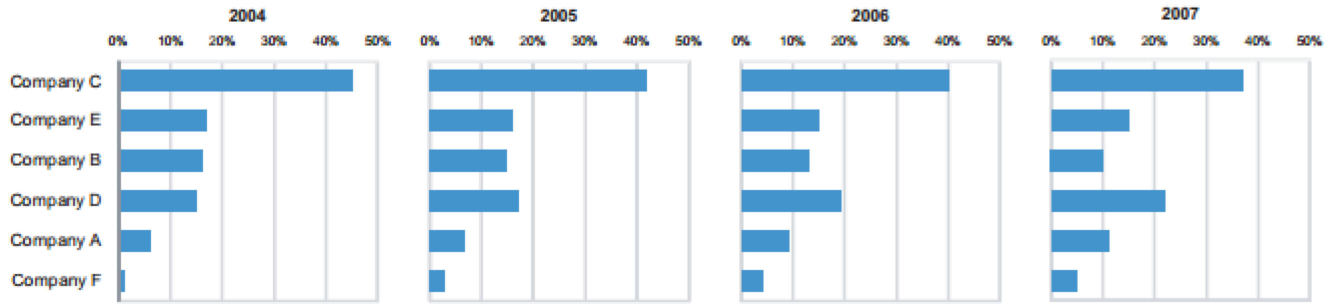
Prefer a sorted bar charts to pie charts!



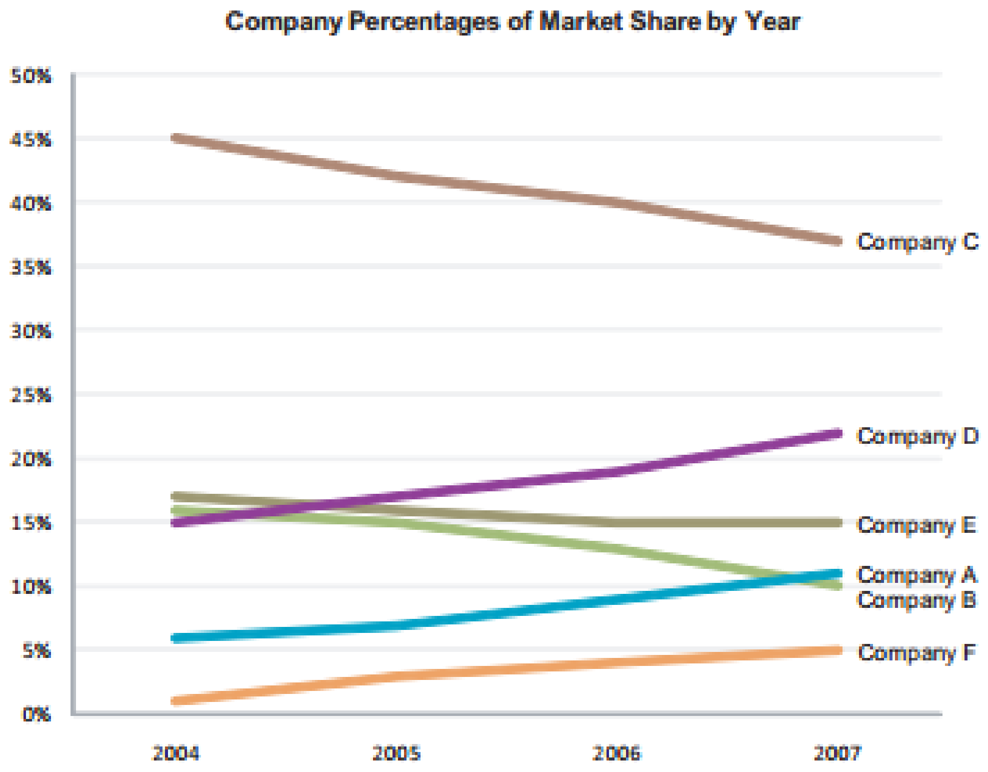


# Avoid pie charts





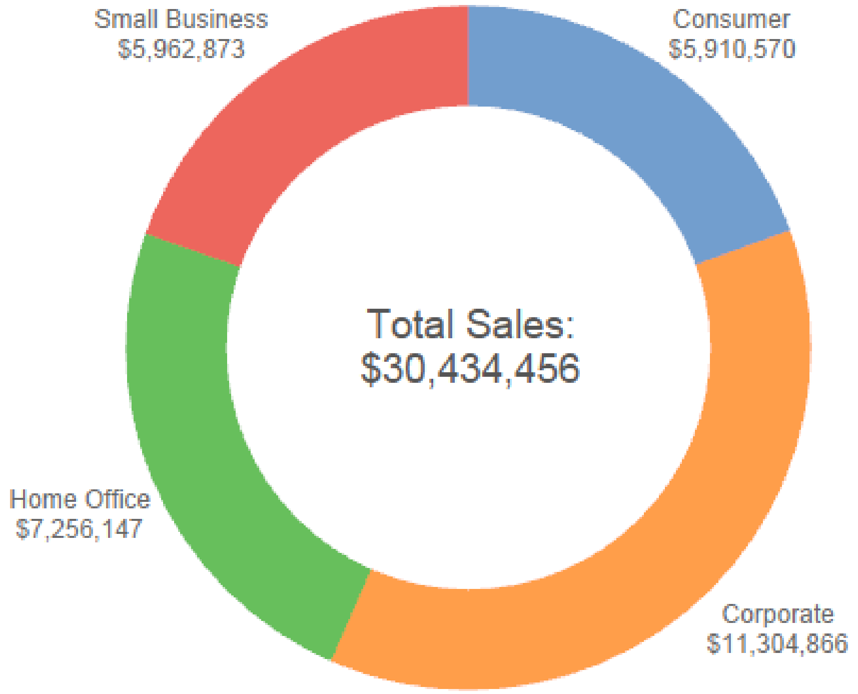
# Avoid pie charts



## What about donuts?

Look cooler, but have the same problems as pie charts. Avoid.

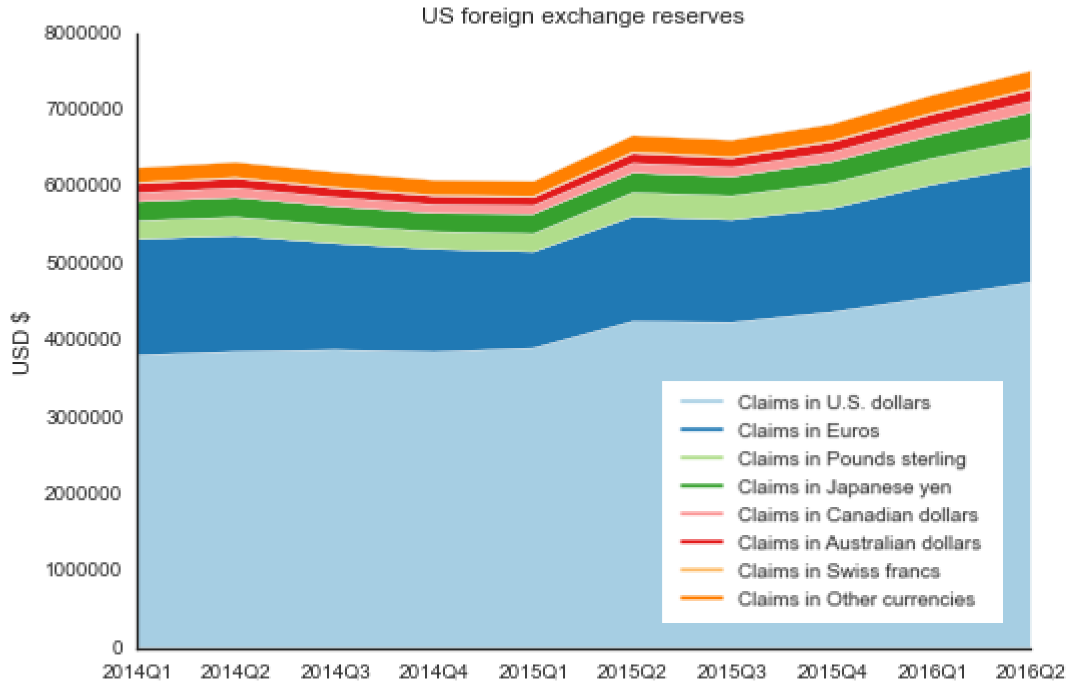
**Prefer bar charts to pies and donuts!**



# Stacked area charts

Useful for displaying composition conditioned on time (time series composition)

```
sns.set_style('white')

# trick to get the legend to work
colors = sns.color_palette(
    "Paired", len(fields))
for field, color in zip(fields, colors):
    plot([], [], label=field, color=color)

stackplot(range(10), data, colors=colors)
legend()

title('US foreign exchange reserves')
xticks(range(10), currency.index[1:])
ylabel('USD $')
sns.despine()
```



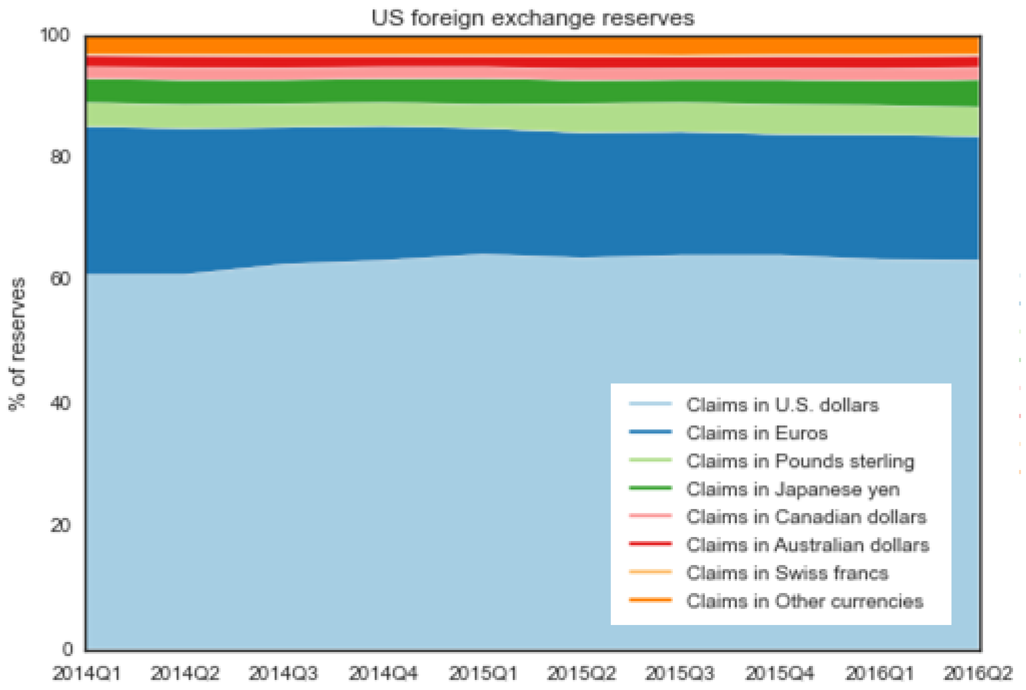
# Stacked area charts

To display percentages, simply convert the absolute values to percentages

```
percent = data / data.sum(axis=0) * 100
```

Which version is best to use depends on what you are trying to show.

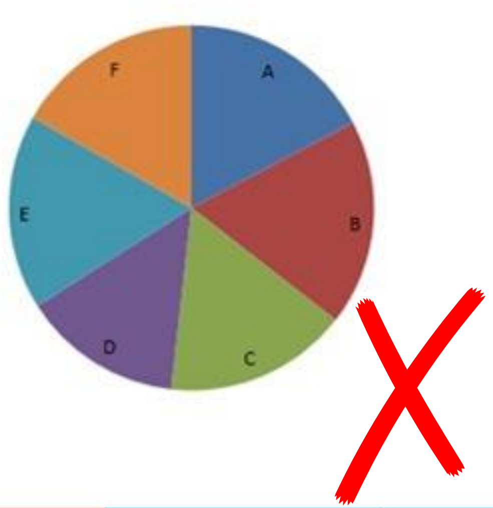
Percentages show less information but highlight changes in composition.

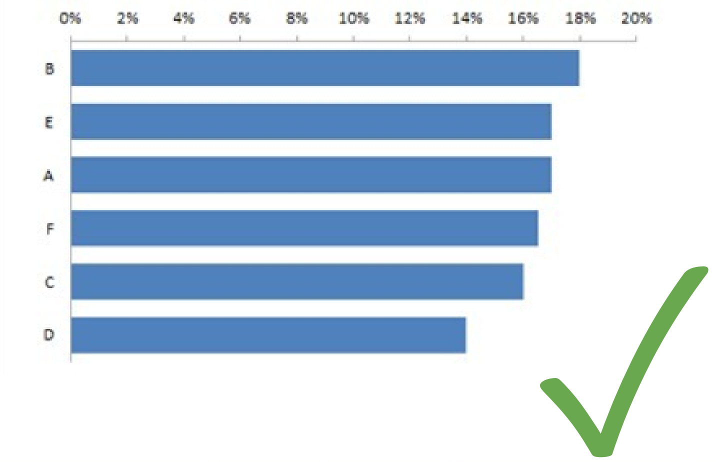


## Summary of visualizing composition

Studies show human beings are not very good at judging areas.

In most cases, just use a bar or line chart to show composition!

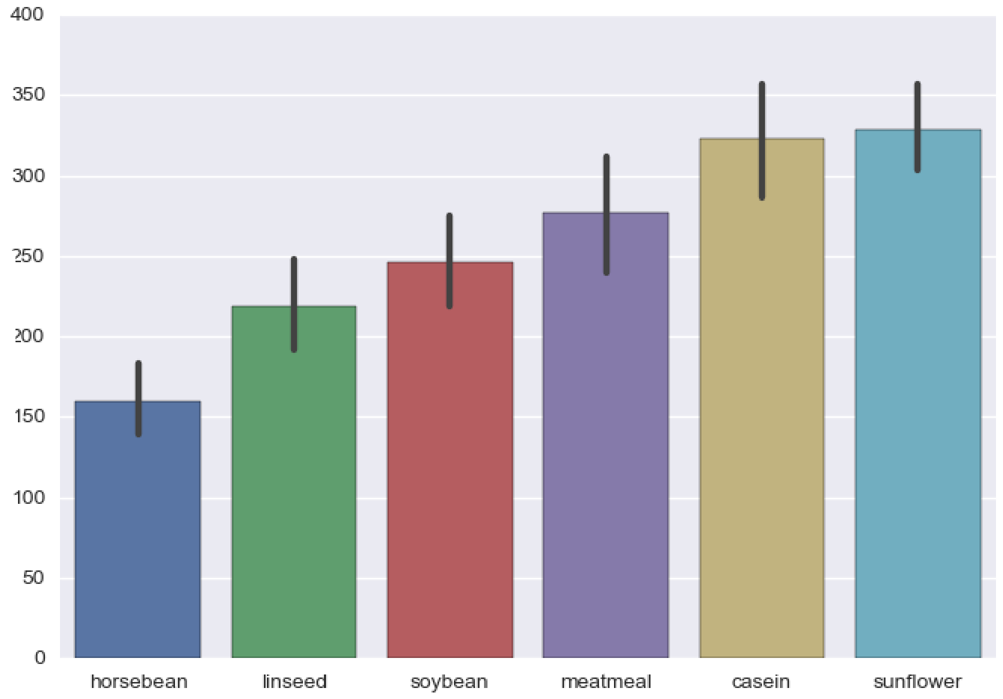




## Topics

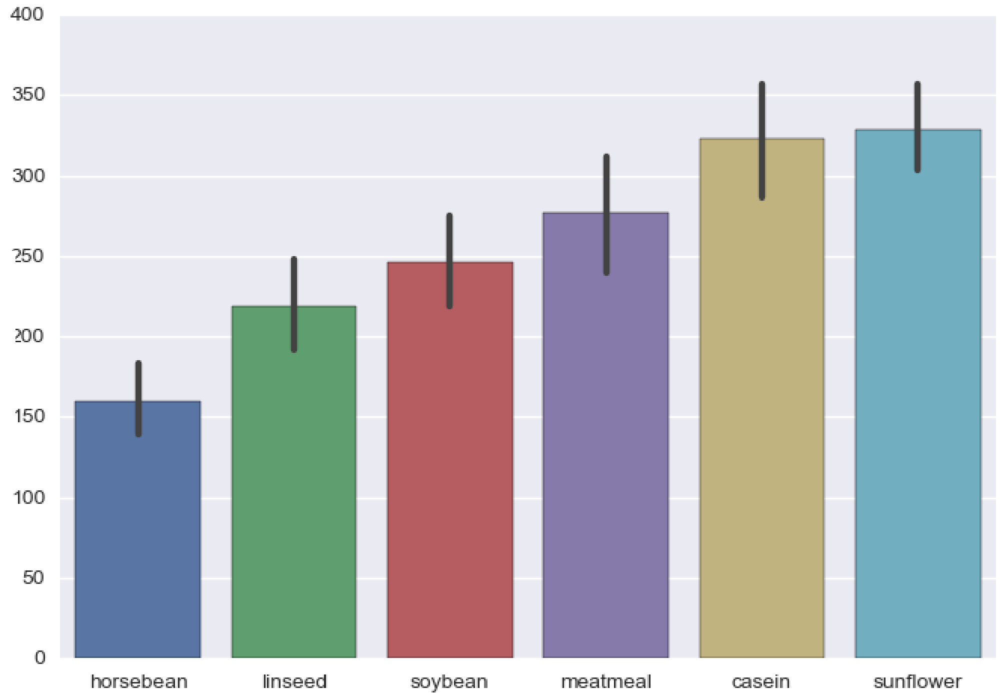
- Visualizing the shape and distribution of data
- Visualizing relationships between attributes
- Comparing quantities
- Visualizing composition and part-whole relationships
- Design principles and visualization faux pas

## Faux pas

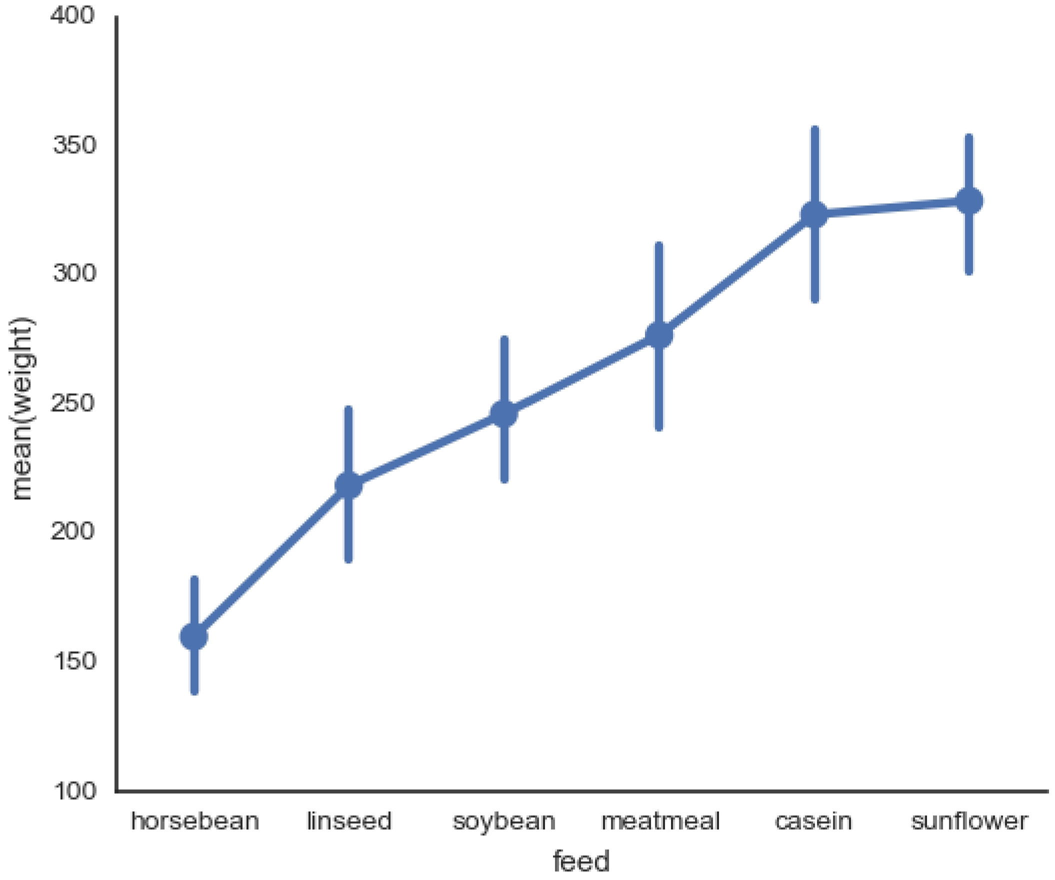

400
350
300
250
200
150
100
50
0
horsebean
linseed
soybean
meatmeal
casein
sunflower

# Faux pas

Forgetting to label the axes (when it's important)

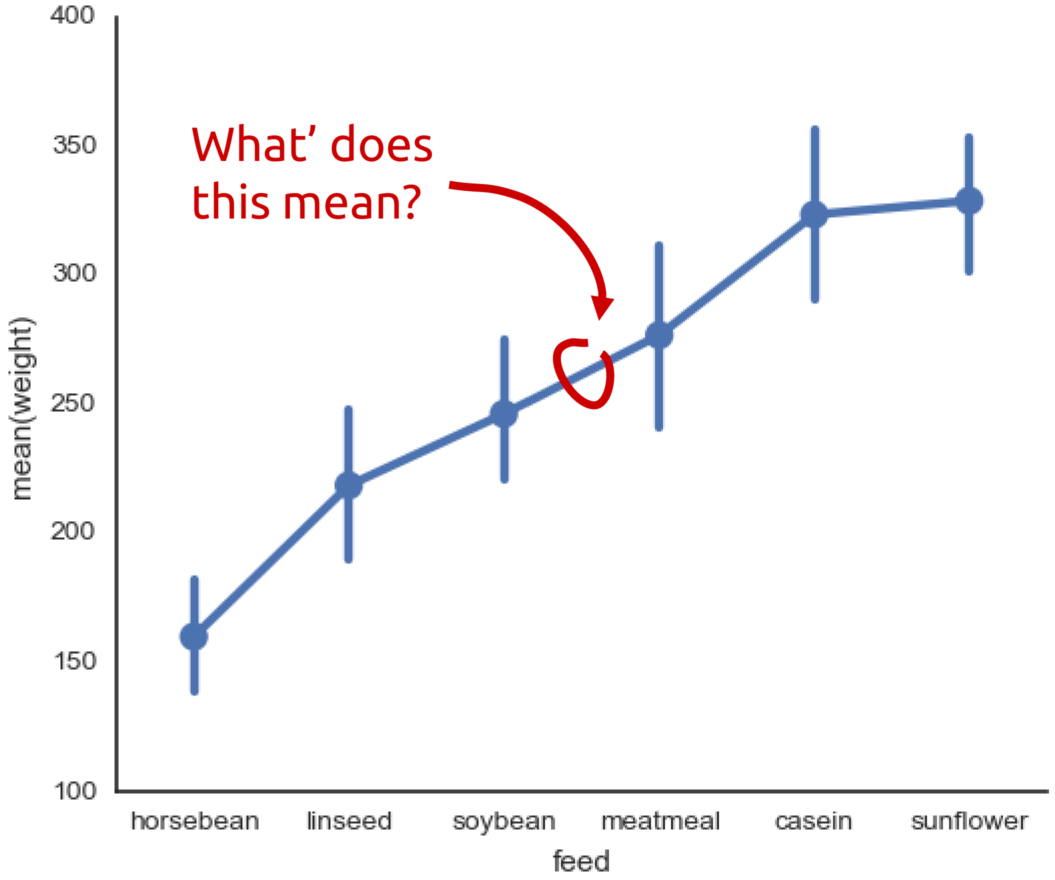


## Faux pas


400
350
300
250
200
150
100
mean(weight)
horsebean
linseed
soybean
meatmeal
casein
sunflower
feed

## Faux pas

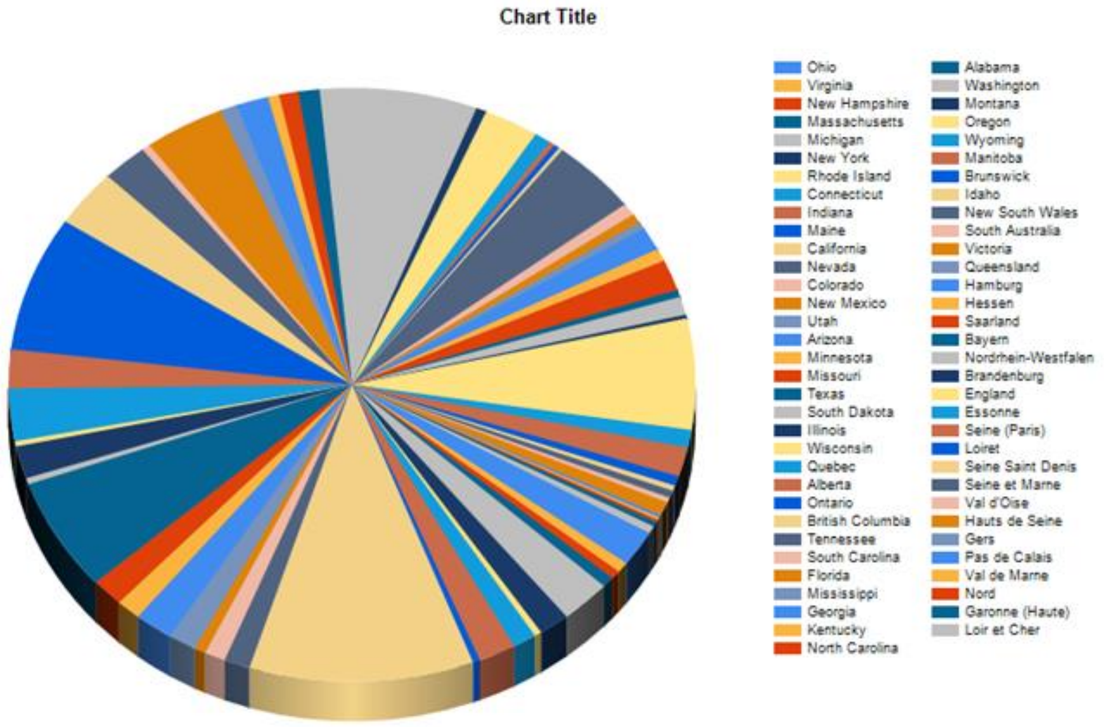
Line plot where bar is more appropriate



## Faux pas

Pie charts with too many sectors

- 2-4 sectors max, only when differences are large.
- Or avoid altogether!



## General design principles

Seen lots of statistical graphics plots

Easy to design ugly, misleading visualizations even using standard plots

Several useful principles for designing visualizations:

- Tufte principles
    - Graphical excellence
    - Graphical integrity
    - Data-ink ratio
    - Chartjunk
- Methods for representing attributes visually
- Avoiding overplotting

# Tufte's principles

**Graphical excellence**

Excellence in statistical graphics consists of complex ideas communicated with clarity, precision, and efficiency

- **show the data**
- induce viewer to think about substance, rather than form
- **avoid distorting the data**
- present many numbers in small space
- make large datasets coherent
- reveal data on several levels of detail

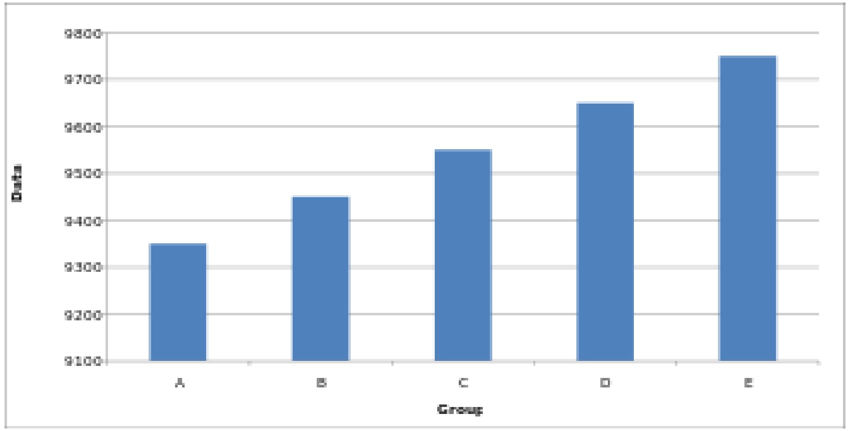
**Graphical integrity**

Data graphics are no different to words: any means of communication can be used to deceive

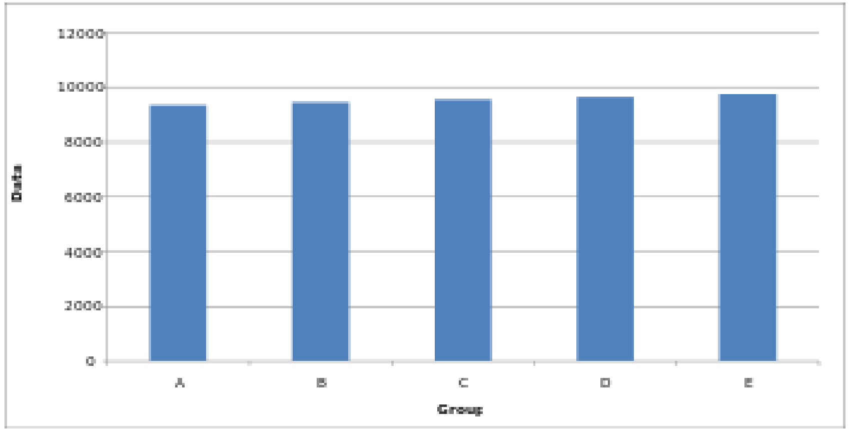
- visual representation should be consistent with numerical

# Integrity: scale distortions

Amazing improvement!

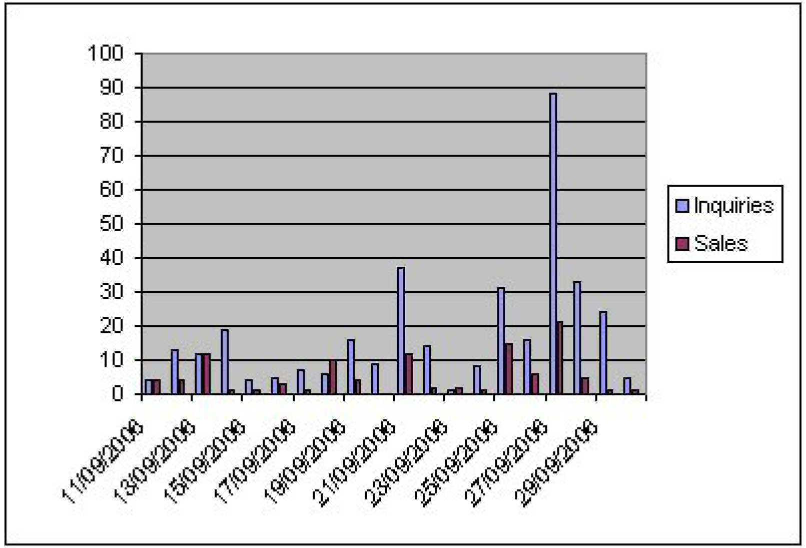


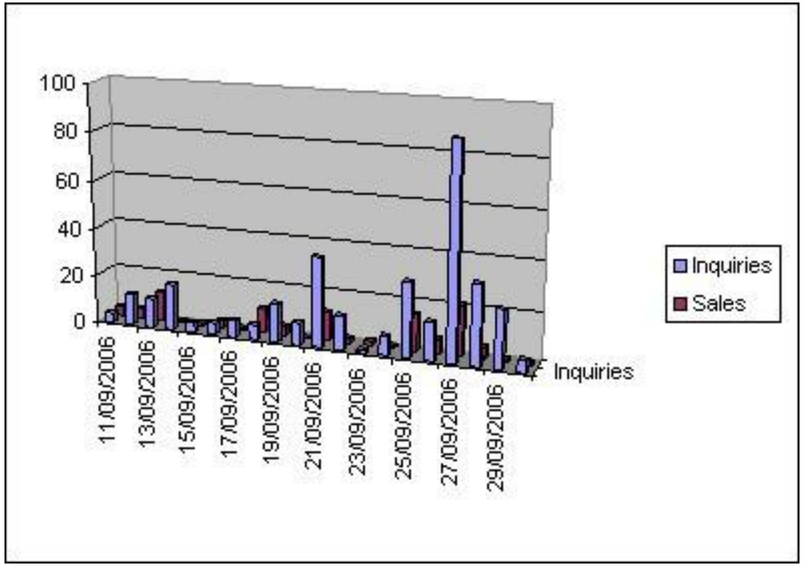
Not so much...



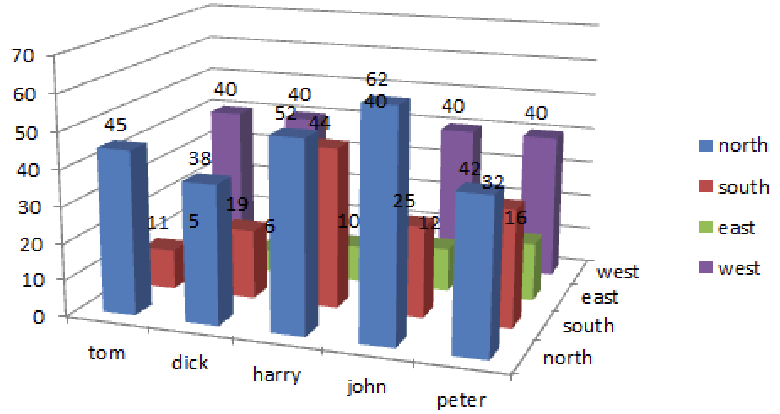
Many more examples:
https://en.wikipedia.org/wiki/Misleading_graph

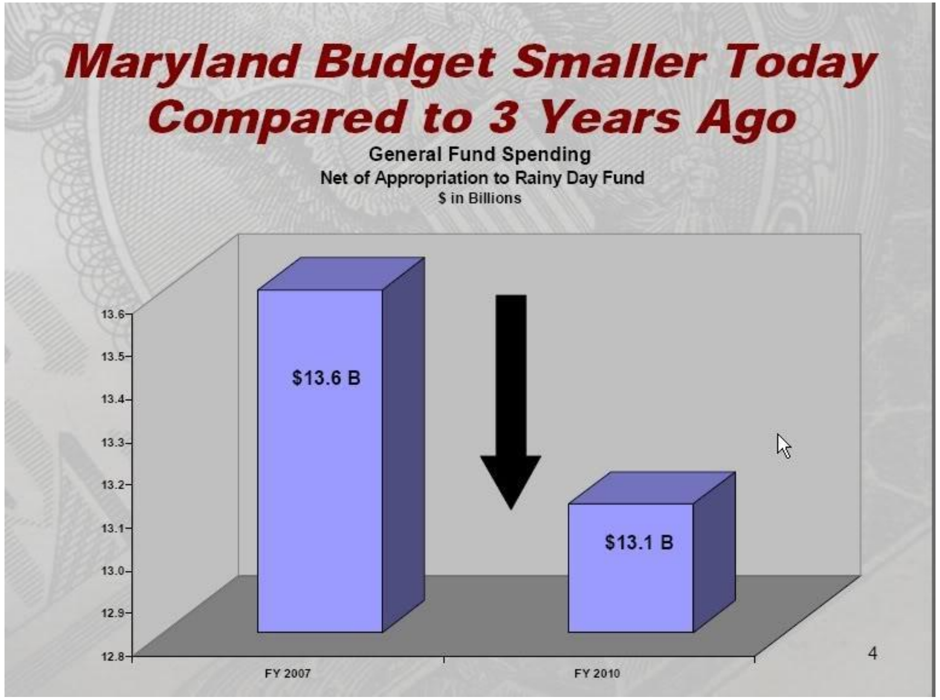
# Chartjunk


100
90
80
70
60
50
40
30
20
10
0
11/09/2006
13/09/2006
15/09/2006
17/09/2006
19/09/2006
21/09/2006
23/09/2006
25/09/2006
27/09/2006
29/09/2006
Inquiries
Sales


100
80
60
40
20
0
11/09/2006
13/09/2006
15/09/2006
17/09/2006
19/09/2006
21/09/2006
23/09/2006
25/09/2006
27/09/2006
29/09/2006
Inquiries
Inquiries
Sales

# Chartjunk


70
60
50
40
30
20
10
0
45
11
38
5
19
40
6
52
40
44
10
62
40
25
12
40
42
32
16
40
tom
dick
harry
john
peter
west
east
south
north
north
south
east
west


Maryland Budget Smaller Today Compared to 3 Years Ago
General Fund Spending
Net of Appropriation to Rainy Day Fund
$ in Billions
13.6
13.5
13.4
13.3
13.2
13.1
13.0
12.9
12.8
$13.6 B
$13.1 B
FY 2007
FY 2010
4

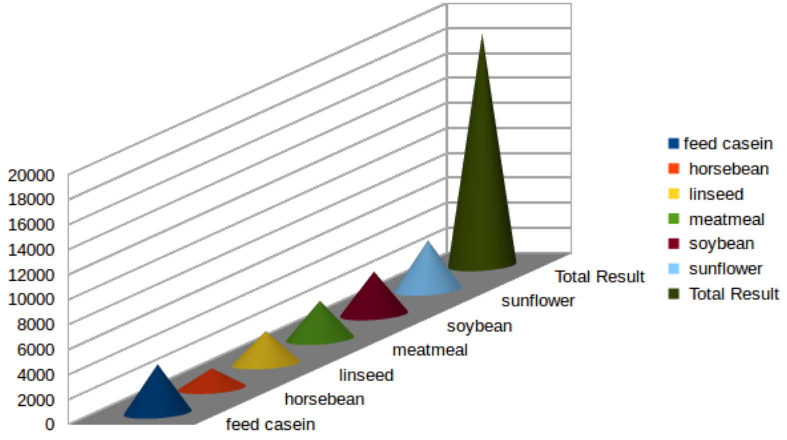
# Tufte's data-ink ratio

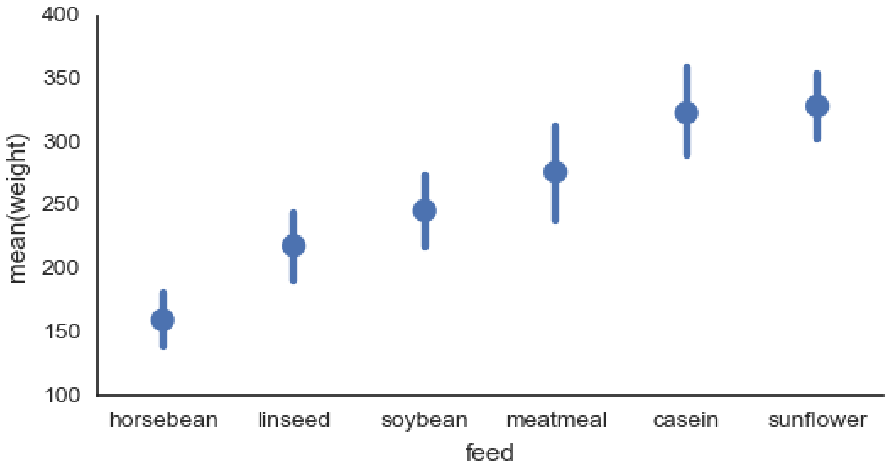
**The data-ink**: the non-erasable core of the graphic

**Maximize the data-ink ratio (within reason).**

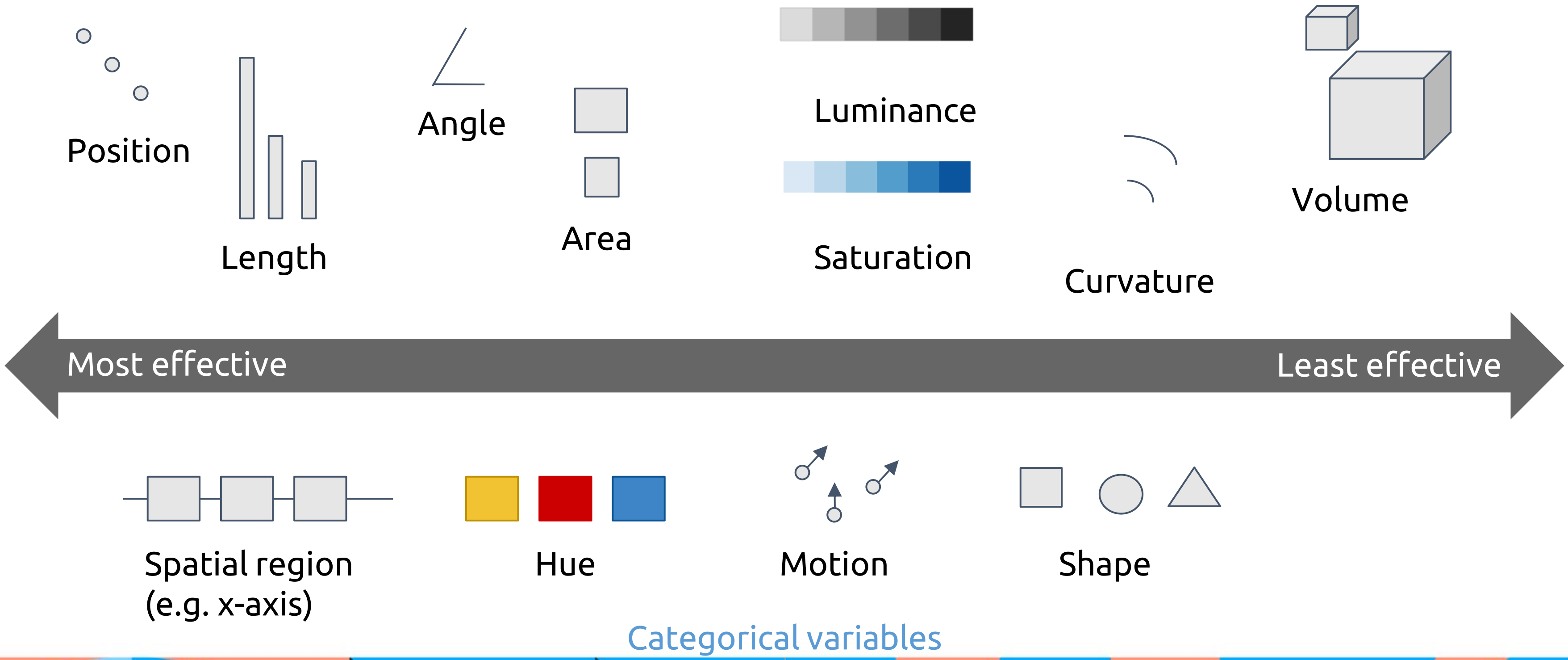
**Two principles:**

1. Increase data ink (add more data)
2. Erase non-data ink





# Representing attributes visually



## Overplotting

Reduce size of points

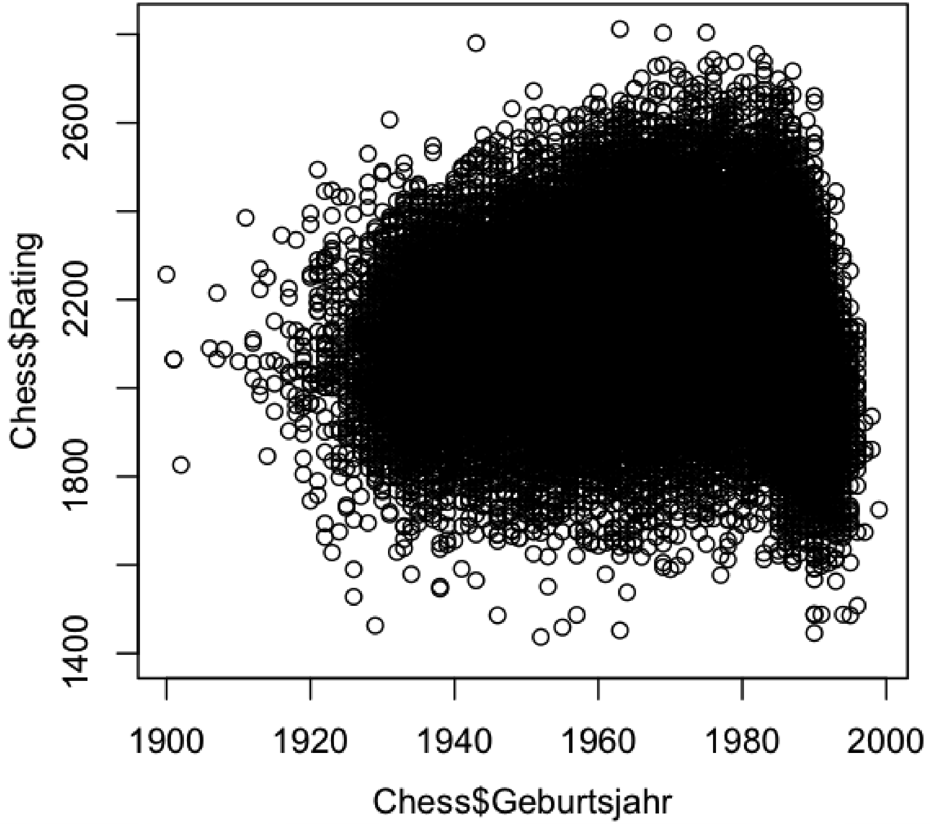
Increase transparency

Jitter

Use a sample

Split data by facets

https://www.perceptualedge.com/articles/visual_business_intelligence/over-plotting_in_graphs.pdf



## Non-statistical visualizations

Statistical graphics great out of the box tools for fast data exploration

Various other types of visualizations to consider for communication and presentation of results:

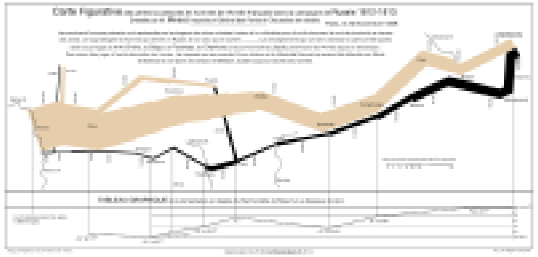
- Custom graphics
- Animations
- Infographics
- Interactive visualizations

# Infographics

Graphic visual representations of information, data or knowledge intended to present information quickly and clearly.

They can improve cognition by using graphics to enhance the human visual system's ability to see patterns and trends.

Can be extremely effective but be wary of distorting the data and use of excessive design junk.



# Interactive visualizations

**Interactive statistical graphics**

Bokeh can be used to create interactive statistical graphics:

- Uses browser technology backend (Javascript, SVG, canvas)
- Interaction is a good tool for data exploration
- Publication export more problematic

**Interactive custom graphics**

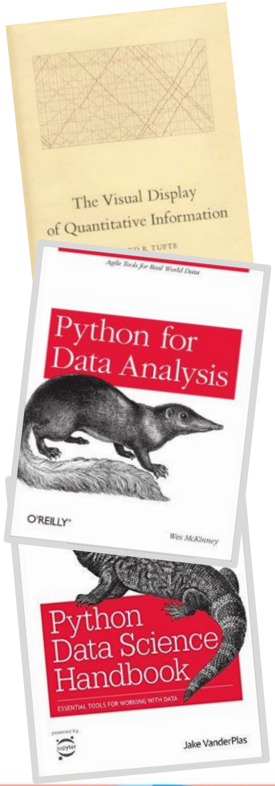
Create powerful novel visualizations

- d3js (Javascript)
- Processing (Java)

Can be time consuming to design these visualizations.

## Questions?

# Further reading



Edward R. Tufte, **The Visual Display of Quantitative Information** (DCU library)

Wes McKinney, **Python for Data Analysis** (DCU library)

- Chapter 8: Plotting and visualization
- Chapter 9: Data aggregation and group operations

Jake Vanderplas, **Python for Data Science Handbook**

- Chapter 4: Visualization with Matplotlib

**Seaborn documentation**: http://seaborn.pydata.org/index.html