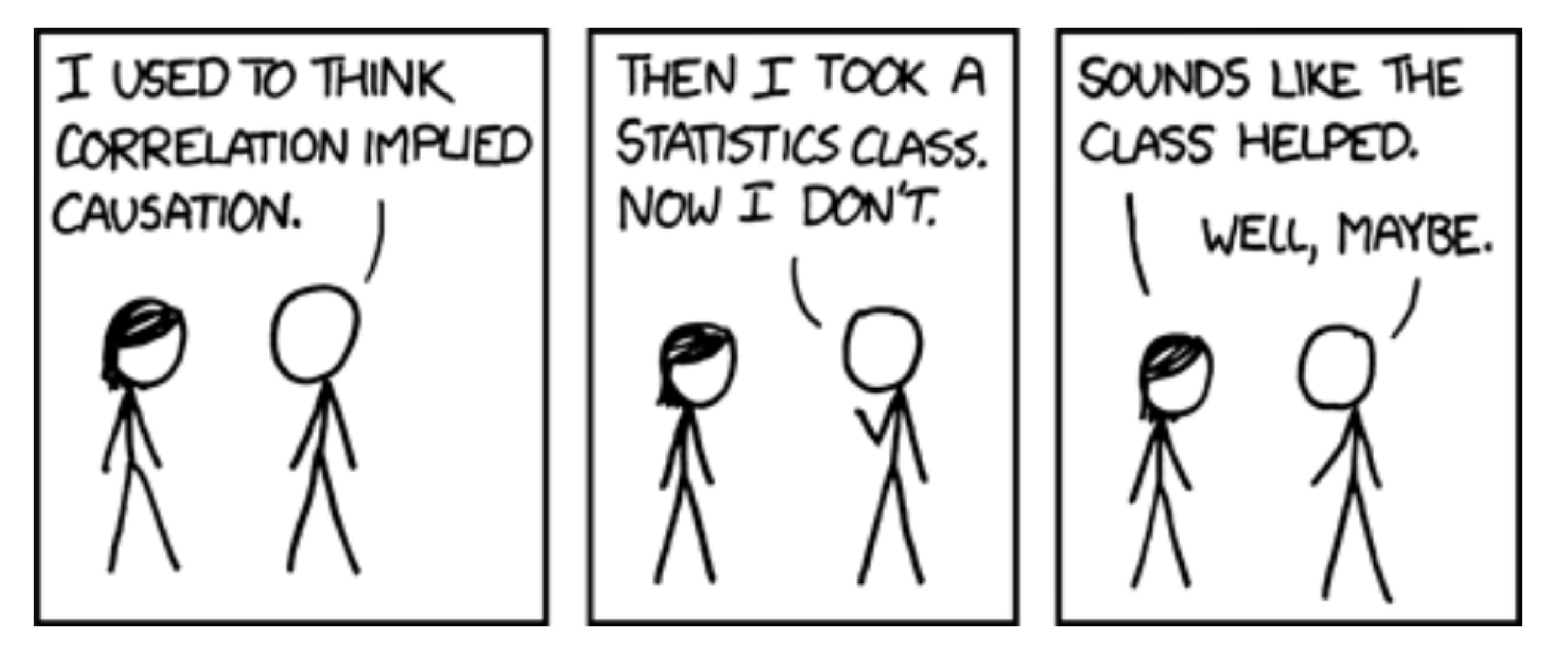
Week 3: Statistics and Exploratory Data Analysis

STARFISH SCHOOL 2020

Introduction to Basics in Statistics



https://xkcd.com/552/

Types of Data

Types of Data

Quantitative

- Continuous
  - Real or complex numbers

- Discrete
  - integers

Categorical

- Nominal
  - e.g., categories A, B, C, or I, II, III

- Ordinal
  - Ordering matters, e.g., a *Lykert Scale* used in a survey: 1,2,3,4,5

Types of Data

Quantitative

- Continuous
  - Real or complex numbers

- Discrete
  - integers

Categorical

- Nominal
  - e.g., categories A, B, C, or I, II, III

- Ordinal
  - Ordering matters, e.g., a *Likert Scale* used in a survey: 1,2,3,4,5

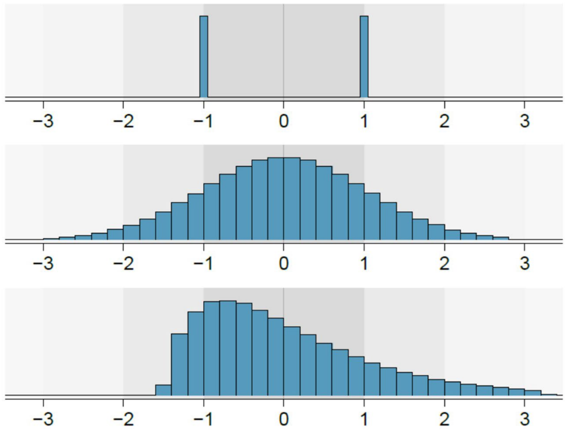
What astronomy examples can you think for each type?

Distributions

What exactly is a distribution?

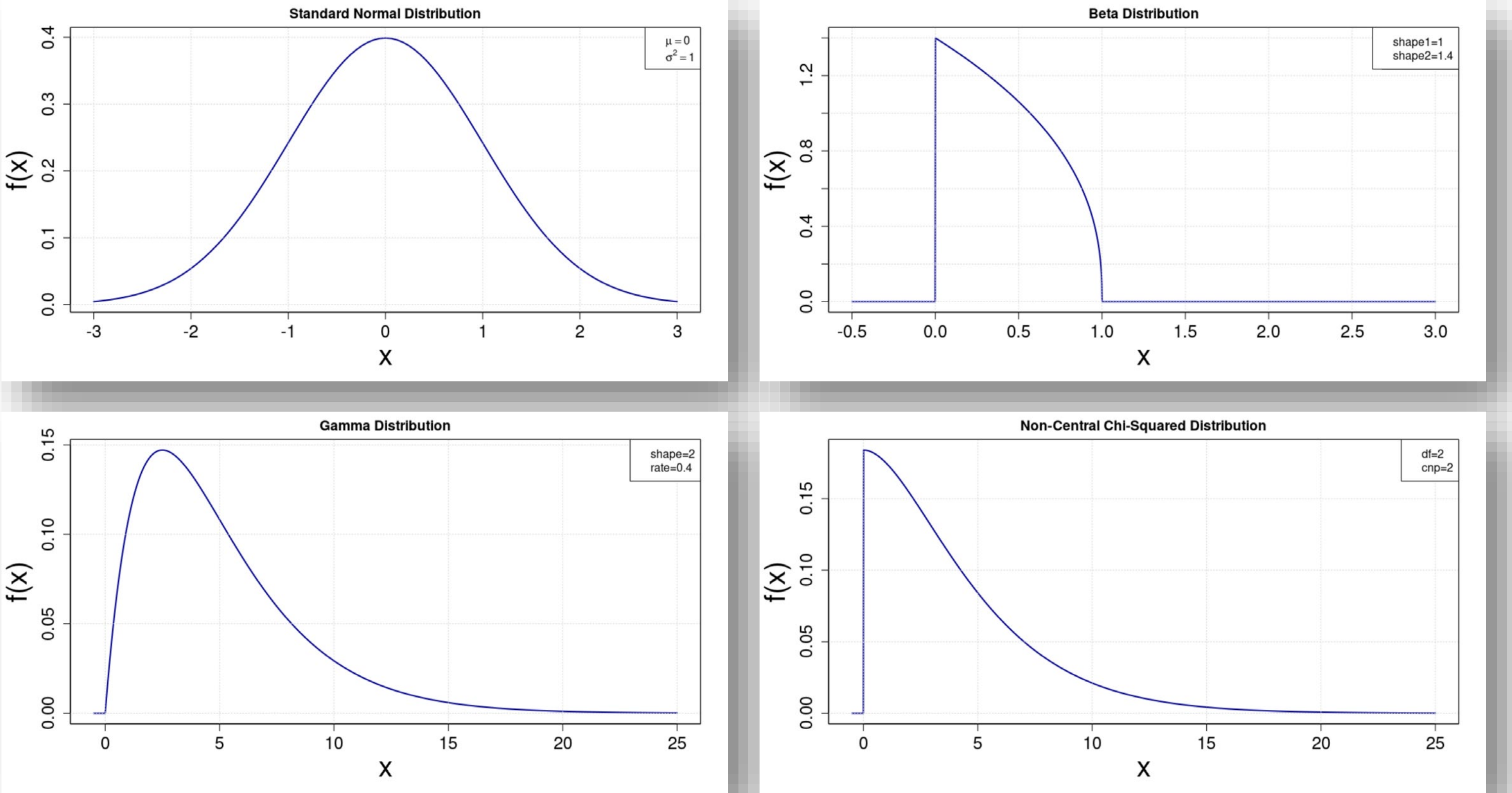
A distribution...

- Tells you the frequency or relative frequency of each possible value/event, or of some data that was collected
- **Could be empirical or analytic**
- Can be useful for modelling a population of objects
- Is often a foundation of statistical reasoning
- Can be continuous or discrete
- That is analytic has parameters that define its shape
- Can be univariate or multivariate



Example histograms (figure from Open Intro Statistics 4th ed.)

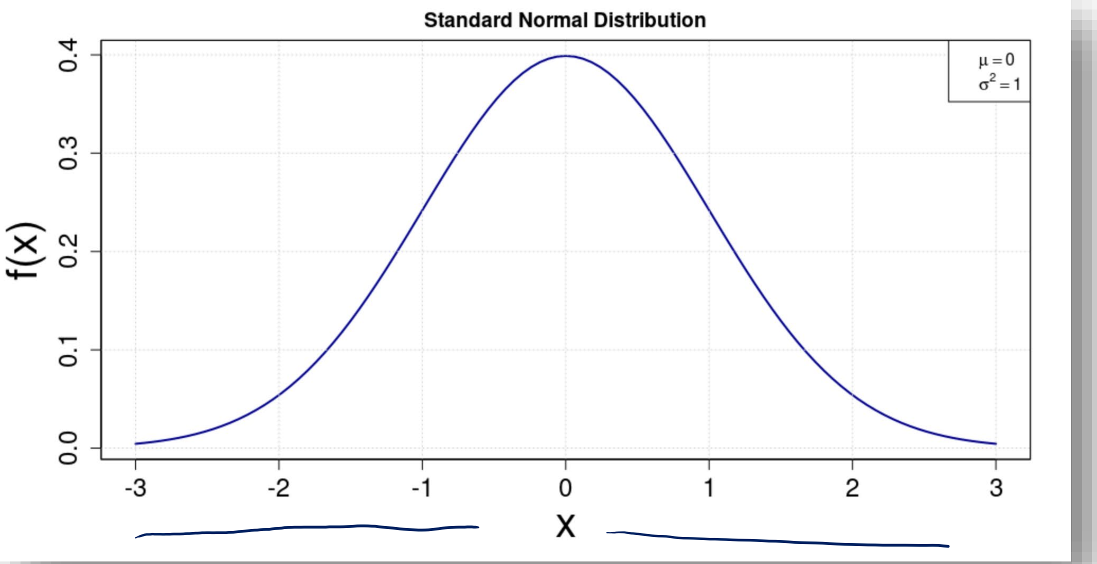
Some analytic probability distributions (plotted by me)



Probability Distributions

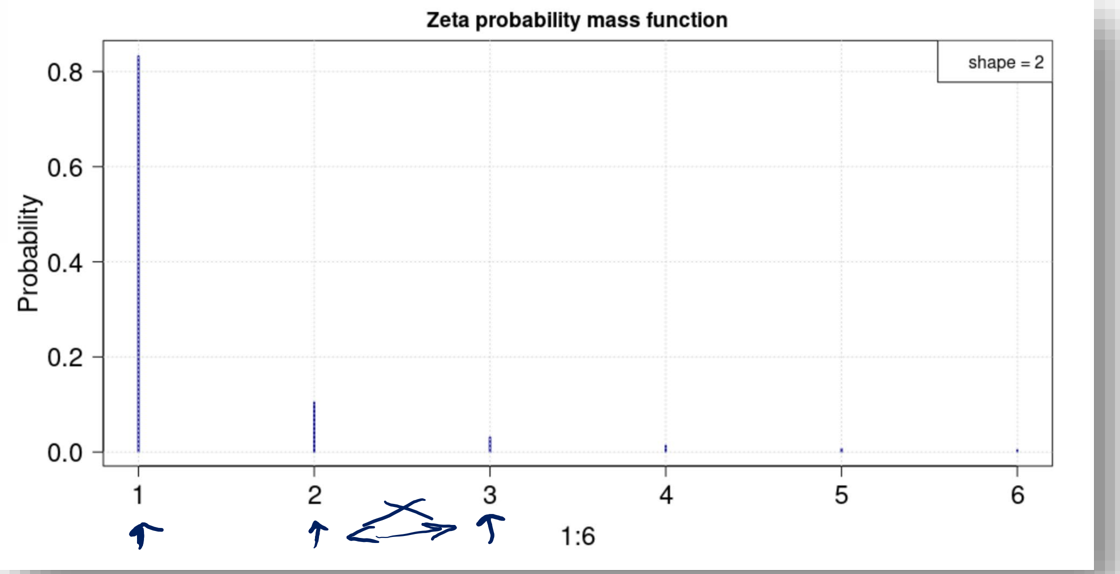
Continuous quantities

probability density function (pdf)

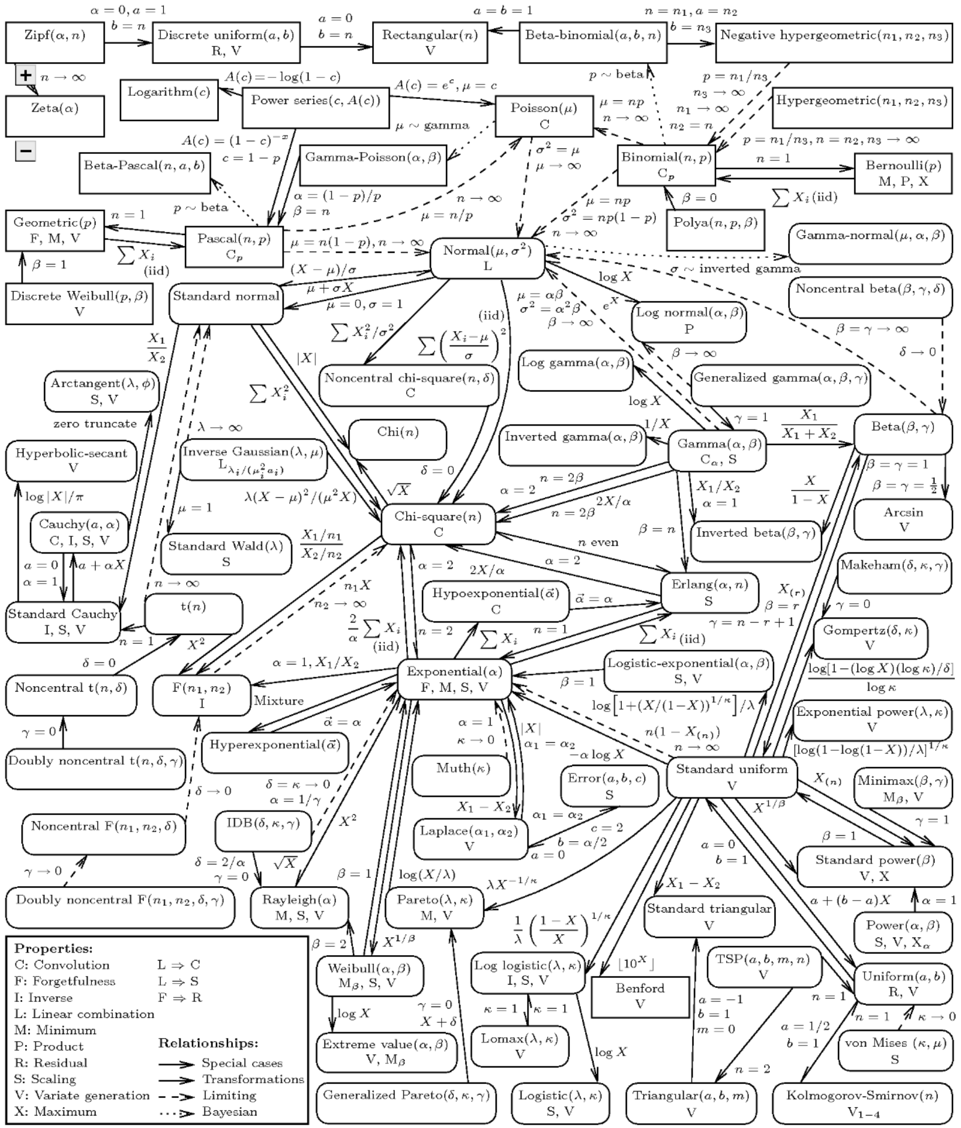


Discrete quantities

Probability mass function (pmf)

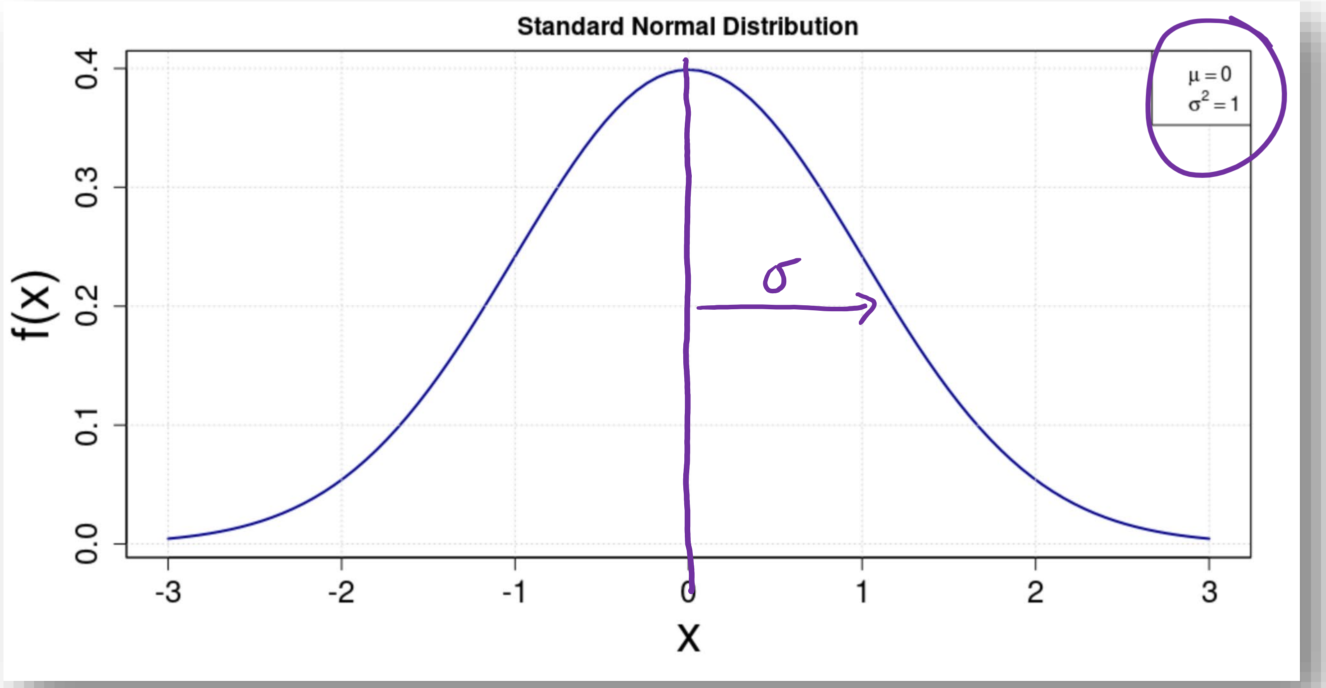


There are many univariate distributions!



http://www.math.wm.edu/~leemis/chart/UDR/UDR.html

The Normal Distribution



μ = mean

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

pdf

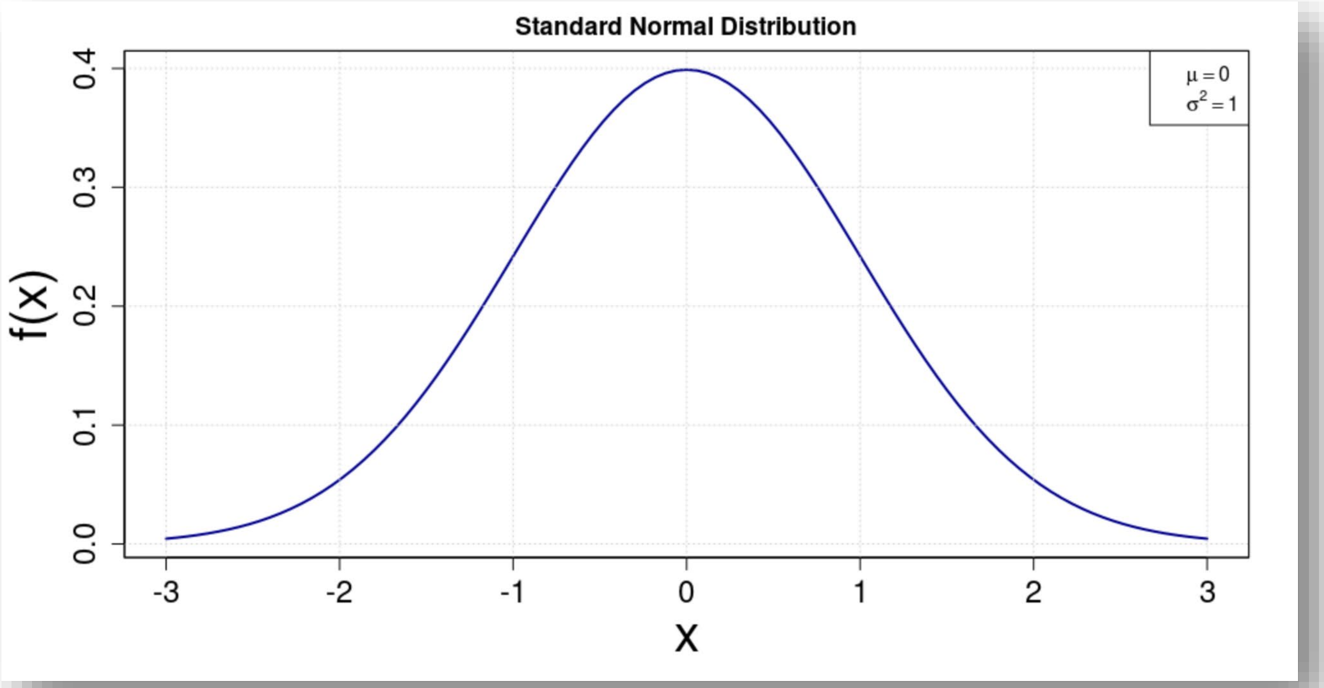
σ^2 = variance

standard deviation

= $\sqrt{\text{variance}}$

The Normal Distribution

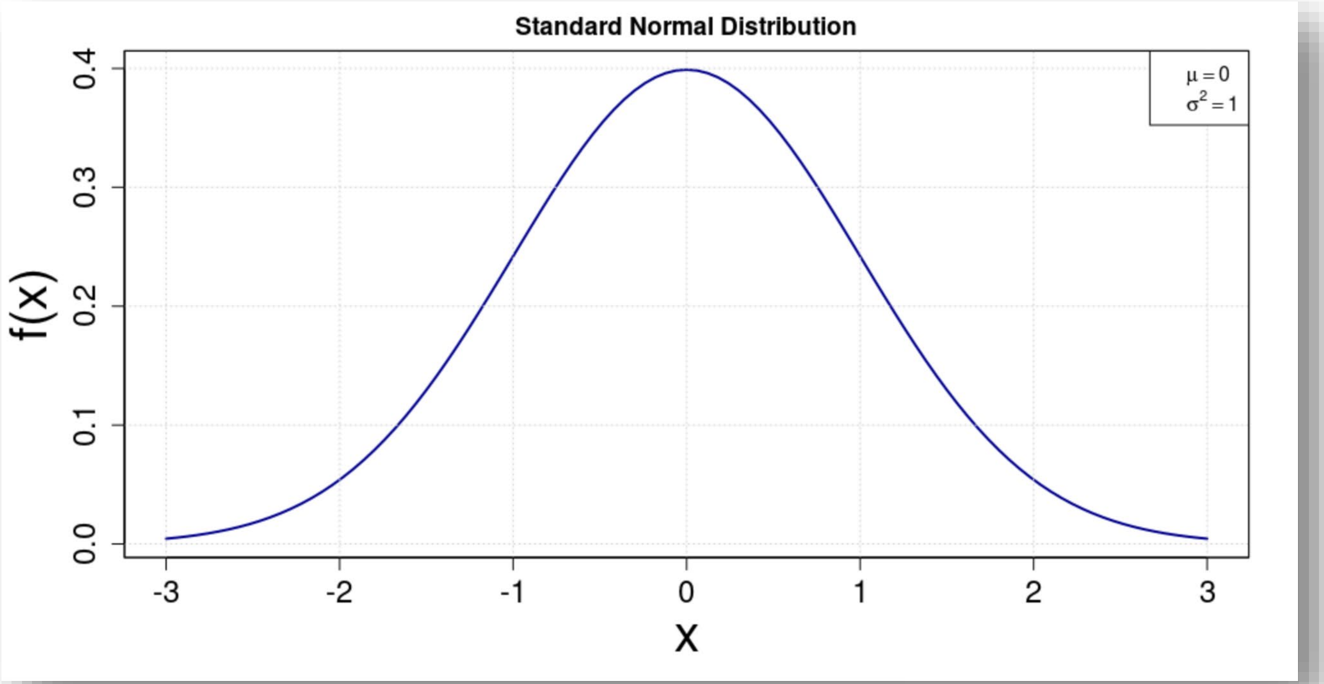
$N(\mu, \sigma^2)$

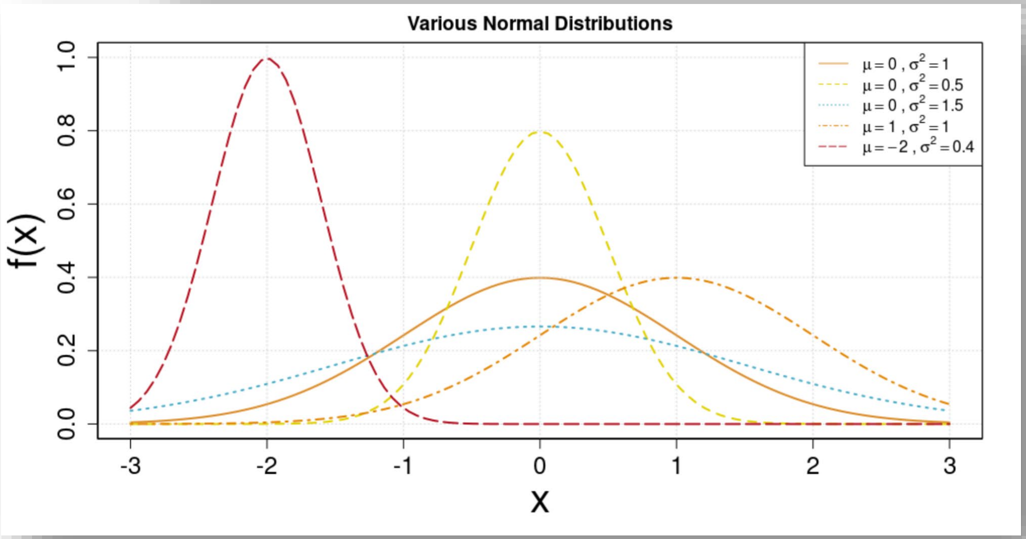


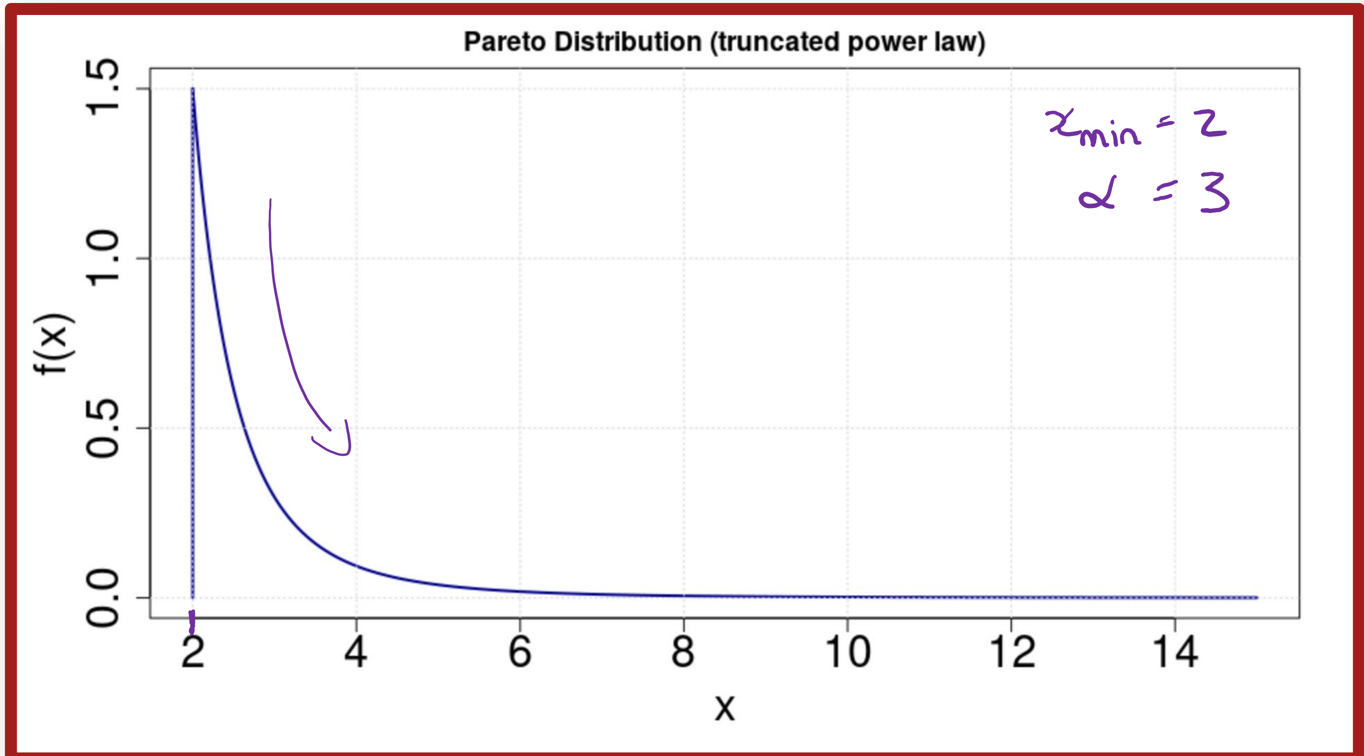
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

The Normal Distribution

The mean and variance are all you need to plot the Normal







Example:

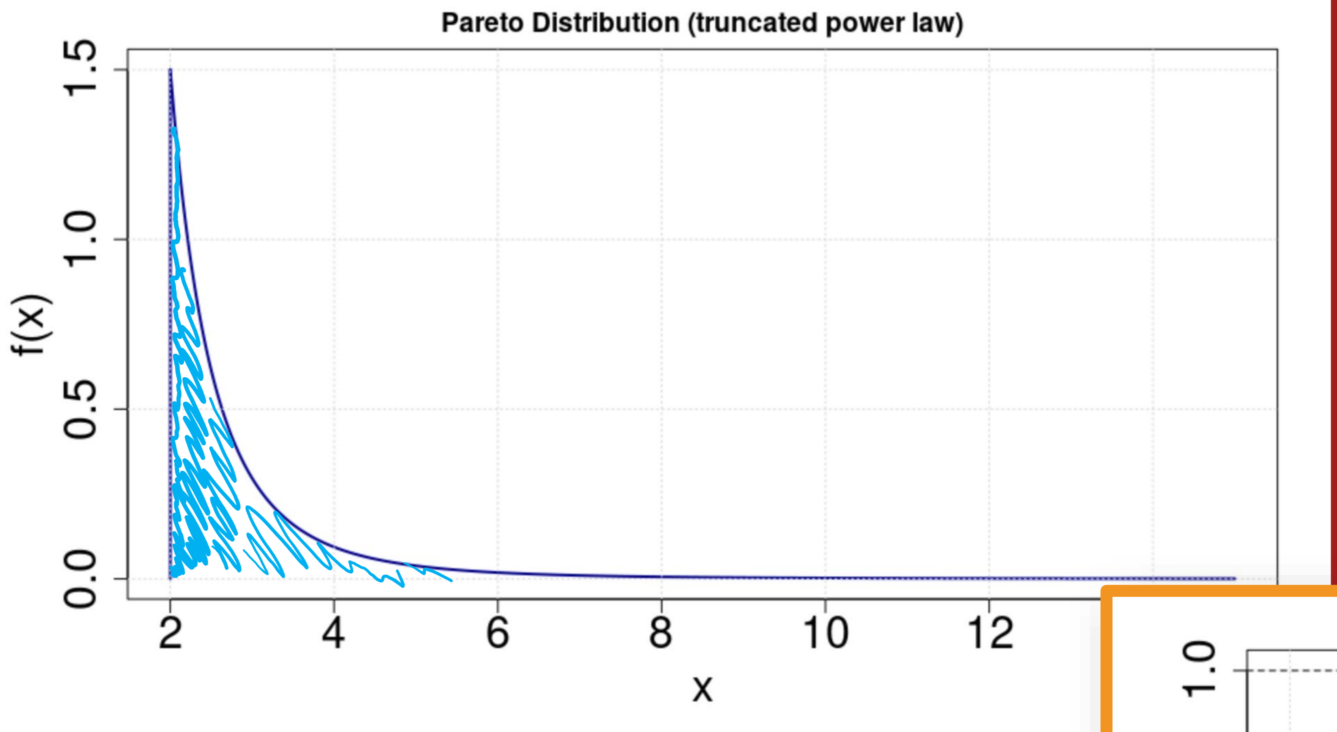
Pareto distribution (truncated power-law)

Probability distribution function (pdf)

$$\boxed{f(x) = \frac{\alpha x_{\min}^{\alpha}}{x^{\alpha+1}}}$$

$\left.\begin{matrix} x_{min} \\ \alpha \end{matrix}\right\}$ parameters

Example:
Pareto distribution (truncated power-law)



Probability distribution function (pdf)

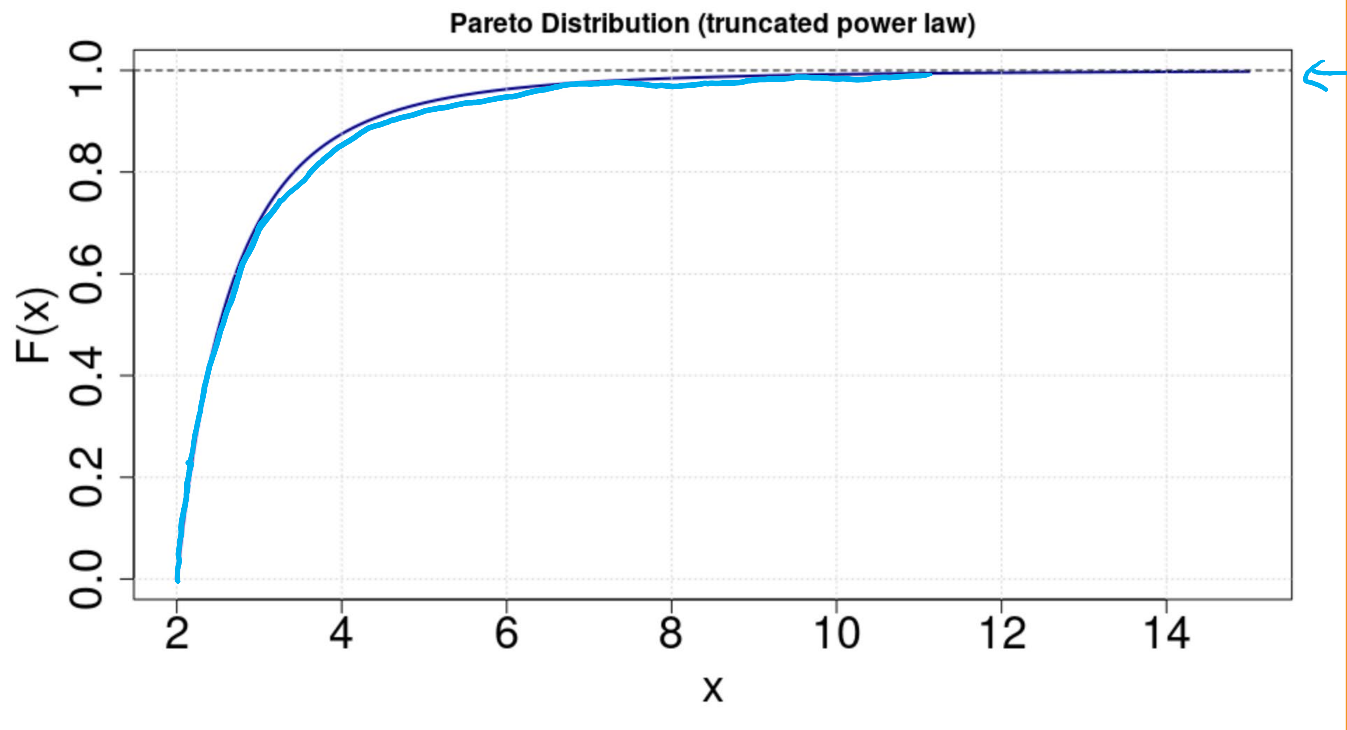
$$\underline{f(x)} = \frac{\alpha x_{\min}^{\alpha}}{x^{\alpha+1}}$$

$$\int_{-\infty}^{\infty} f(x)\,dx = 1$$

$$F(x) = \int f(x)\,dx$$

$$\underline{F(x)} = P(\underline{X \leq x}) = 1 - \left(\frac{x_{\min}}{x}\right)^{\alpha}$$

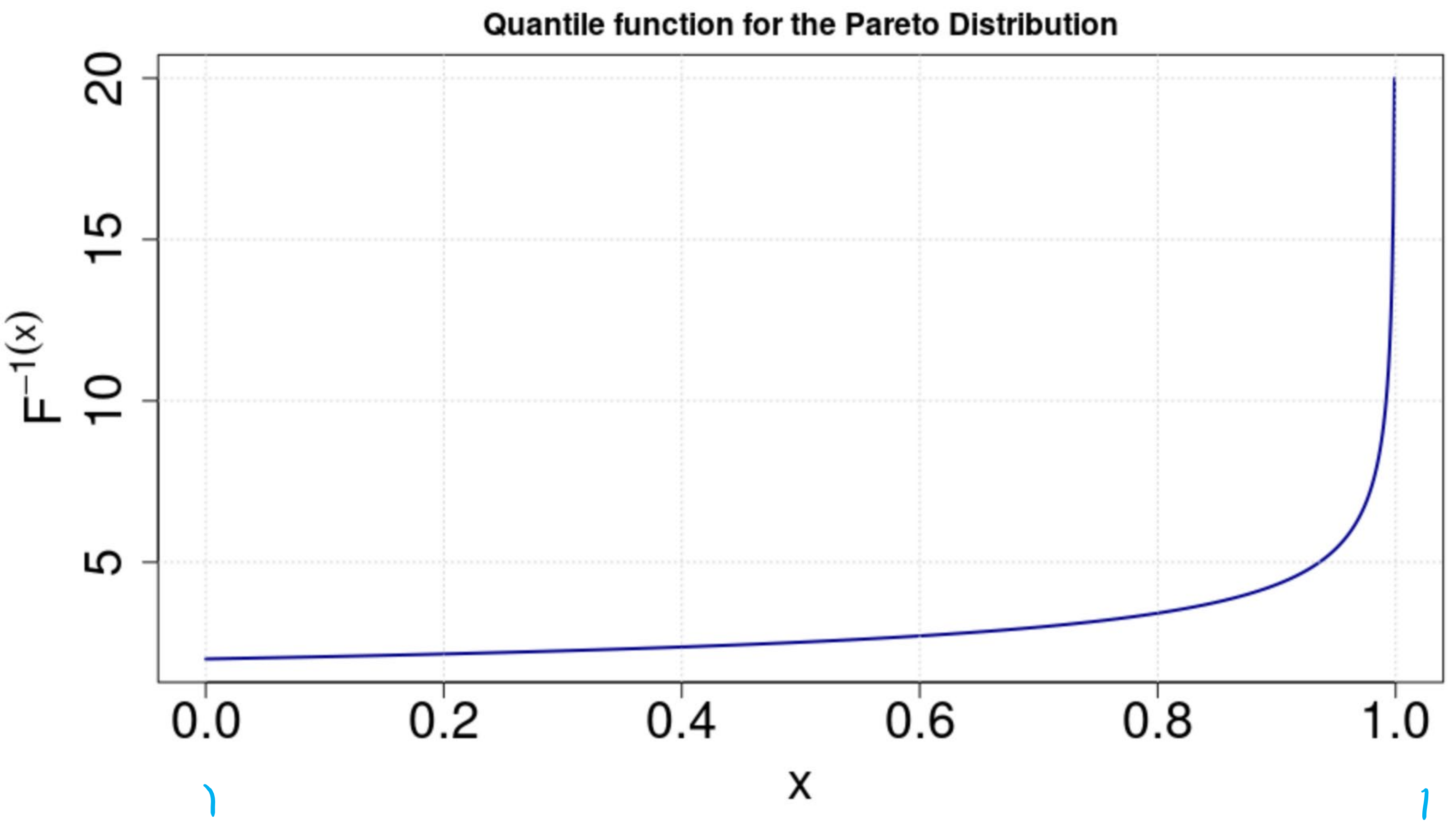
Cumulative distribution function (cdf)



Quantile Function

$\boxed{F^{-1}(x)}$

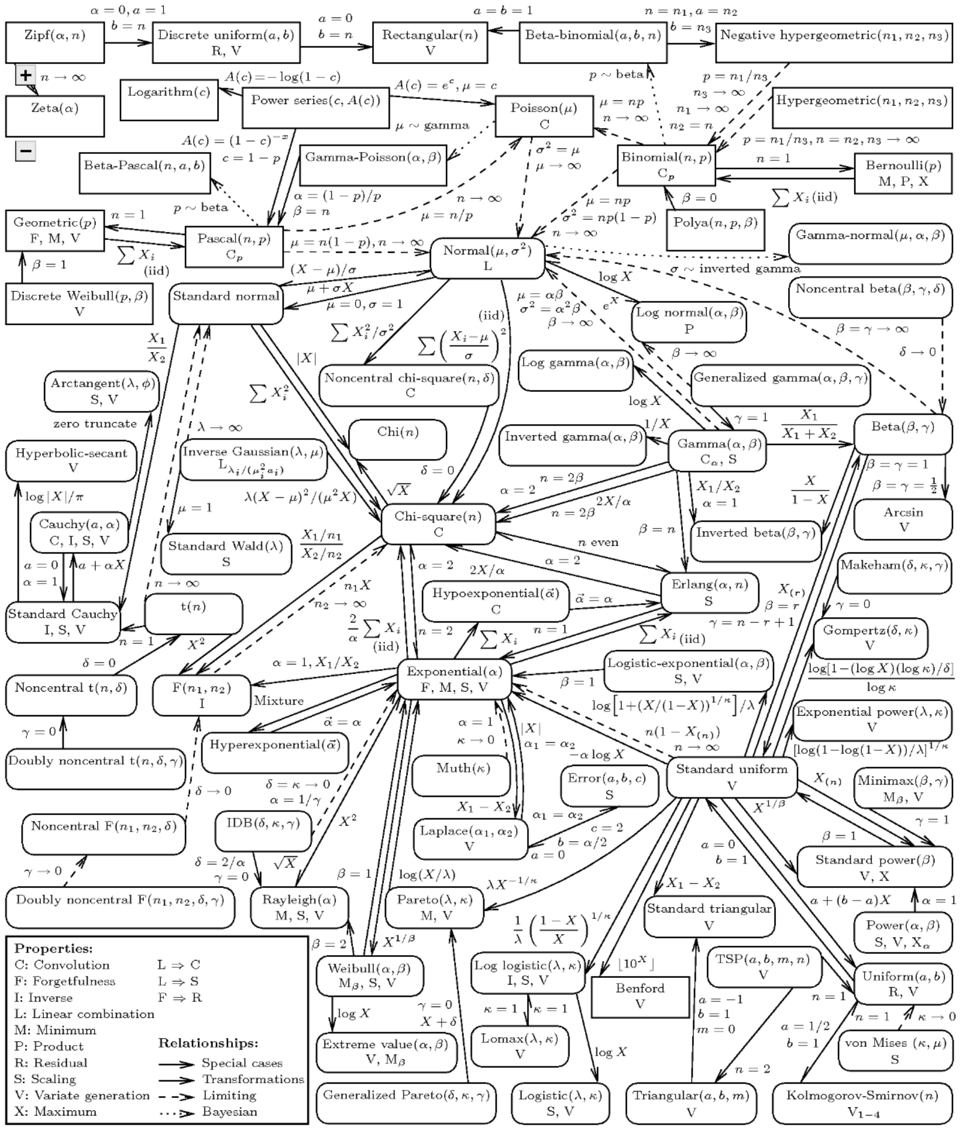
- This is the inverse of the cumulative distribution function (cdf)



HOT TIP

This chart has a pdf file for every distribution! Check out the link below

There are many univariate distributions!



http://www.math.wm.edu/~leemis/chart/UDR/UDR.html

Demo Time!
Look at the data

Mini-exercise 1

1. Plot the PDF for the χ^2 distribution, for different values of the degrees of freedom (N) of that distribution.
2. Compare this to the normal distribution.
3. What do you notice about the two distributions?
4. Now compare the normal distribution to the log normal distribution for a range of values of the mean and variance. How do the mean and variance of the log normal distribution map onto the mean, variance of the normal distribution?

COOL CATCH

Remember that R, Python have distributions coded up – don't reinvent the wheel!

Random Variables

Random Variable

- A random variable *X* is a *function* that maps an *outcome* to a *real number*
  - e.g., Let's say we decide to flip a coin repeatedly, and each time we flip it we record whether we get heads or tails with a 1 or a 0 respectively.
- In other words, *X* is a **function**. Little *x* represents the data --- *realizations* of that random variable.

$X \rightarrow$ random variable

$x \rightarrow$ data

outcomes

H 1

T 0

$\left.\begin{matrix} H \to 1 \\ T \to 0 \end{matrix}\right\} X(\text{outcome})$

outcome = RBBBRRBRBB

X as the number of blue star

$X(\text{outcome}) \longrightarrow x = 6$

A Random Variable follows a distribution

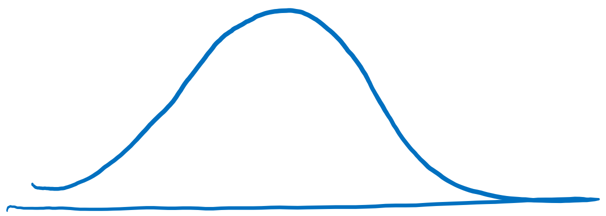
The standard statistics notation to show what distribution a random variable follows is:

distributed as

$$X \sim \underbrace{N(\mu, \sigma^2)}_{\text{normal distr.}}$$

x

$n = 10^5$



For example, we might assume that are data x (e.g. the photon counts from a star) follows a Poisson distribution

$$X \sim Pois(\lambda)$$

A Random Variable follows a distribution

The standard statistics notation to show what distribution a random variable follows is:

$$X \sim N(\mu, \sigma^2)$$

For example, we might assume that are data x (e.g. the photon counts from a star) follows a Poisson distribution

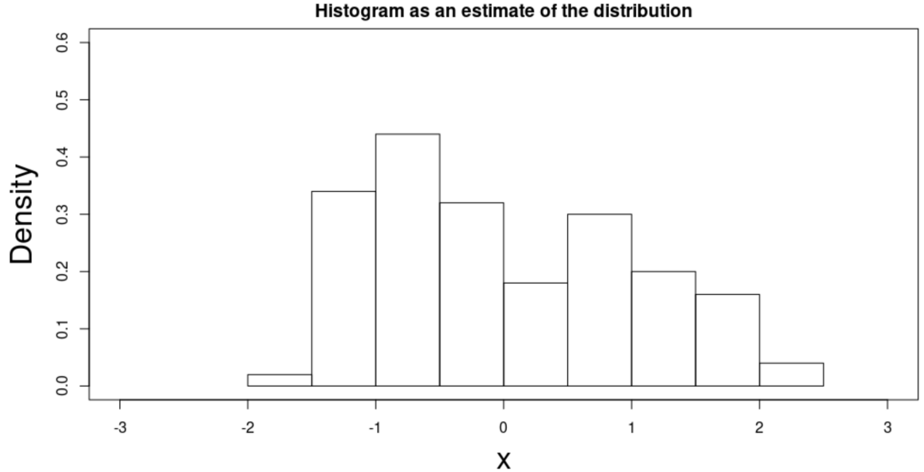
$$X \sim Pois(\lambda)$$

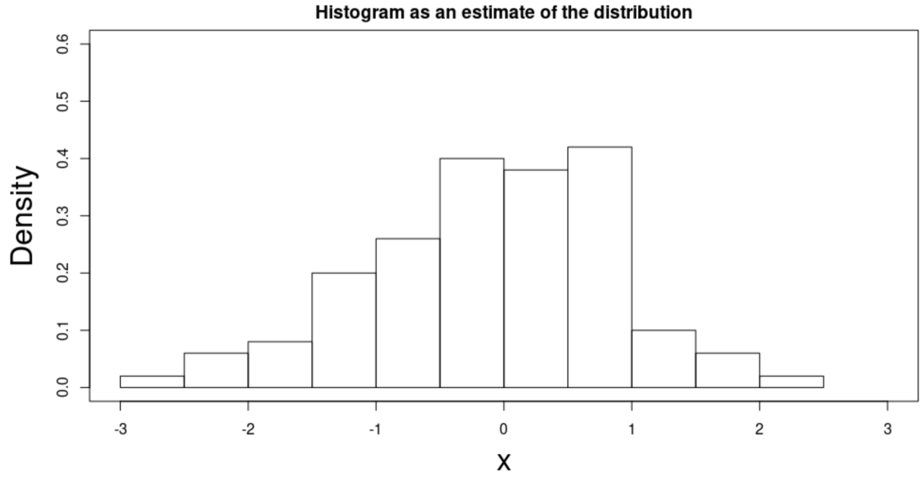
HOT TIP

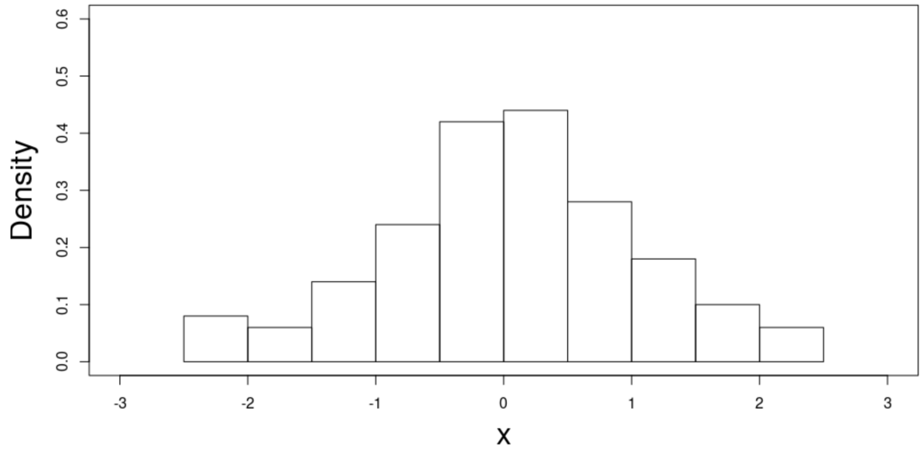
The Poisson distribution is often used to describe *counts* of events in an interval of time or space. It is a *discrete probability distribution.*

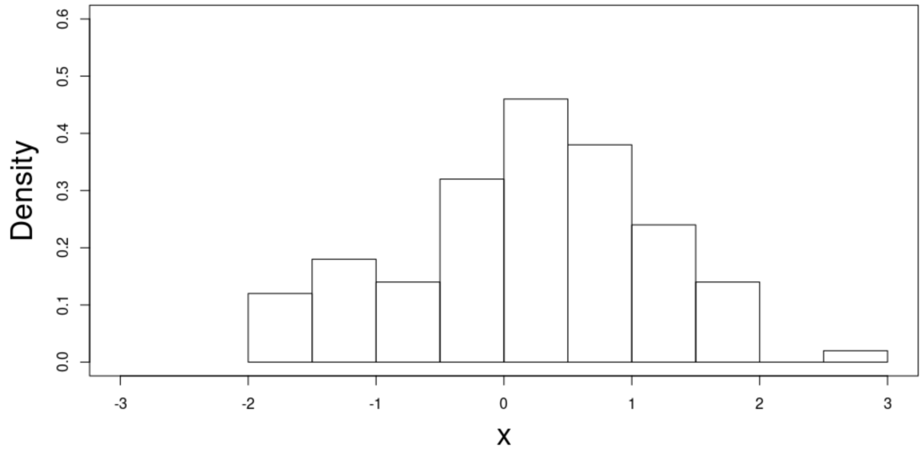
Randomness in Data

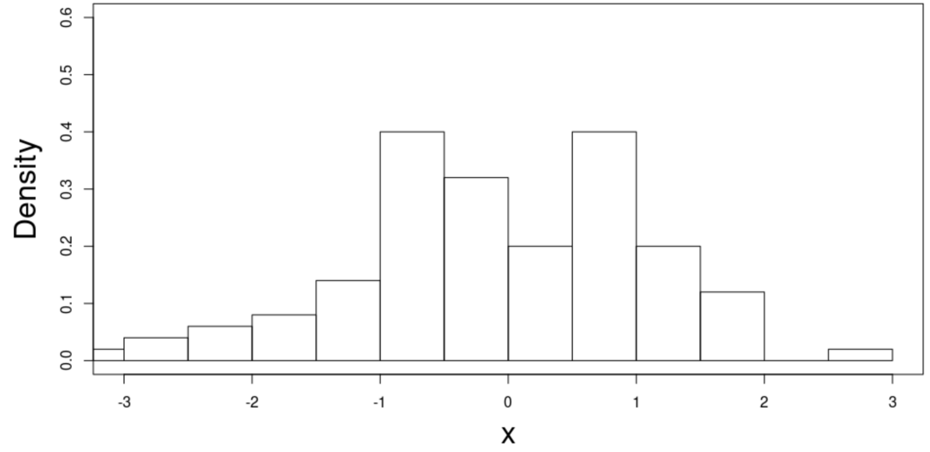
- All these histograms were generated from 100 draws from a standard normal







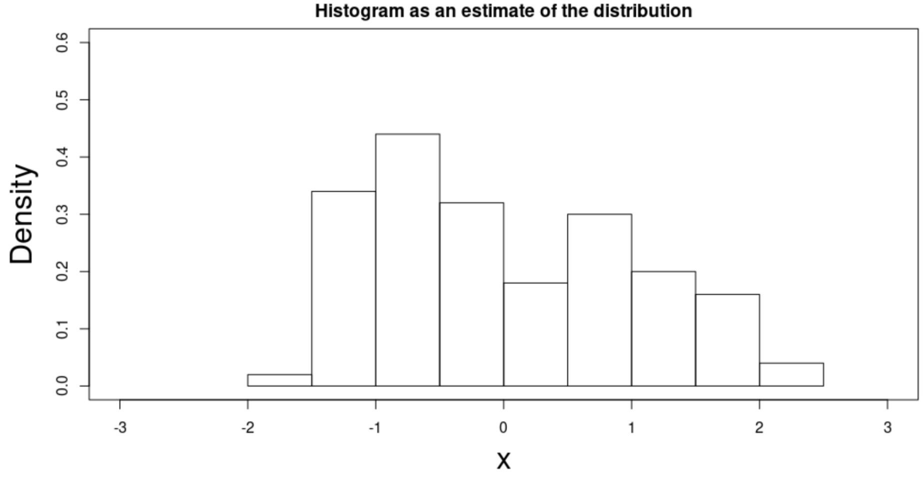


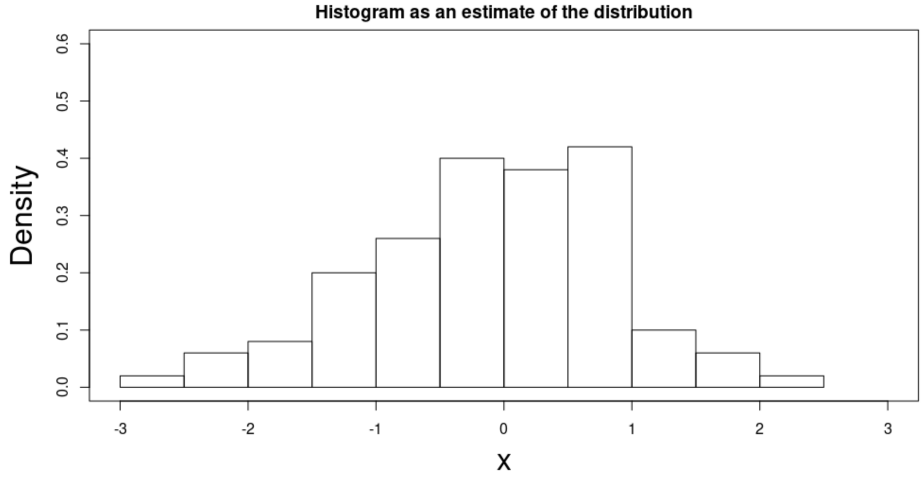


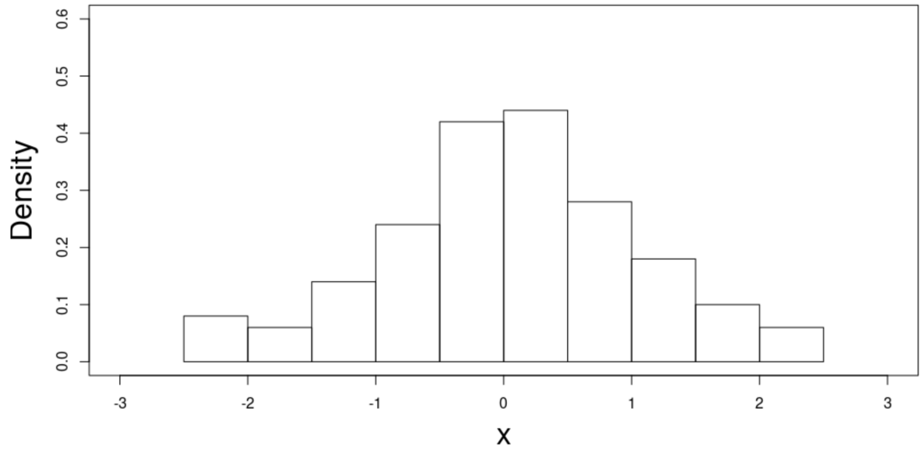
Randomness in Data

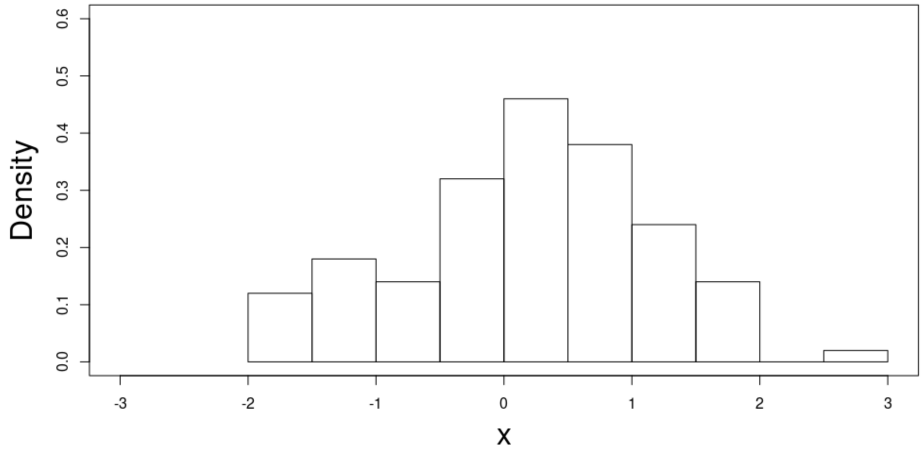
- All these histograms were generated from 100 draws from a standard normal

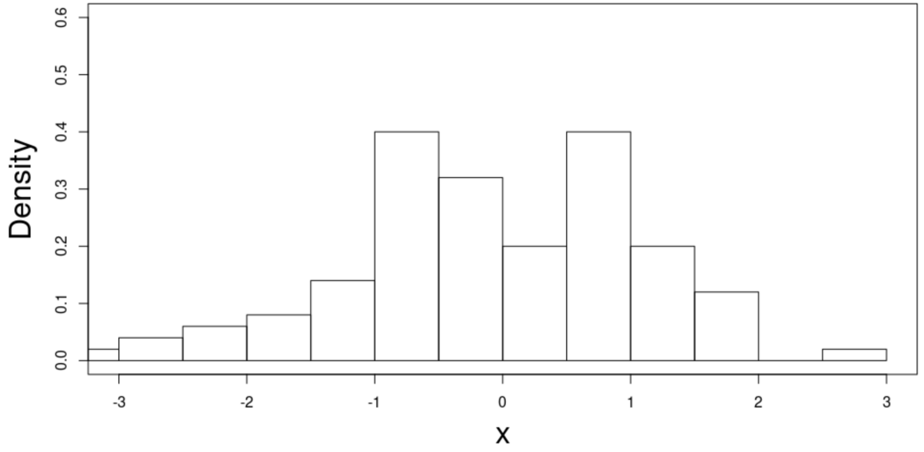
From the data, we can try to *estimate* the true distribution. We can also try to estimate the underlying parameters of the distribution. These estimates are **random variables.**









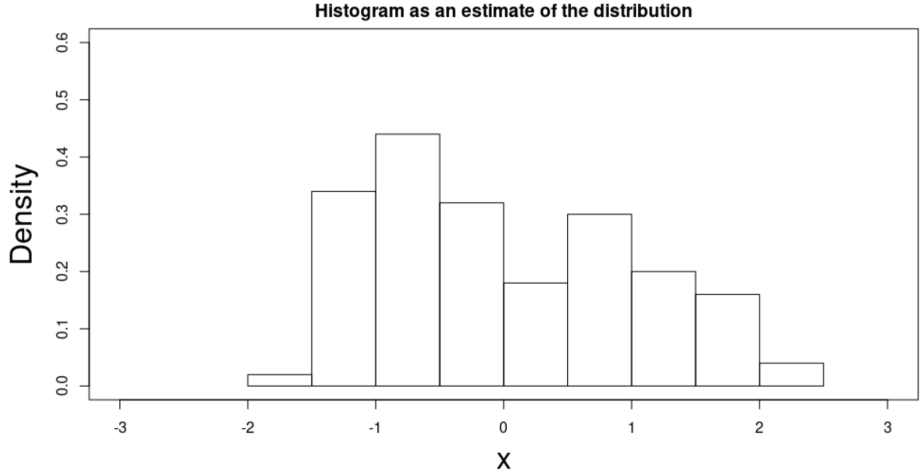


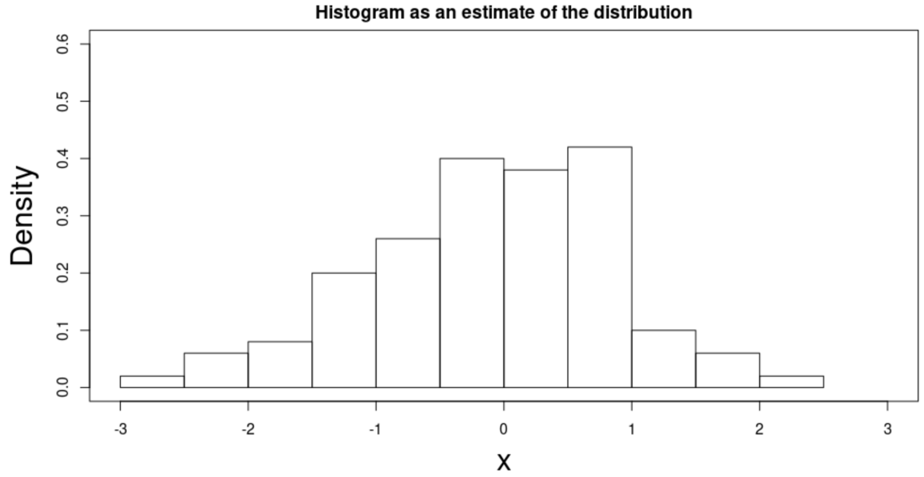
Randomness in Data

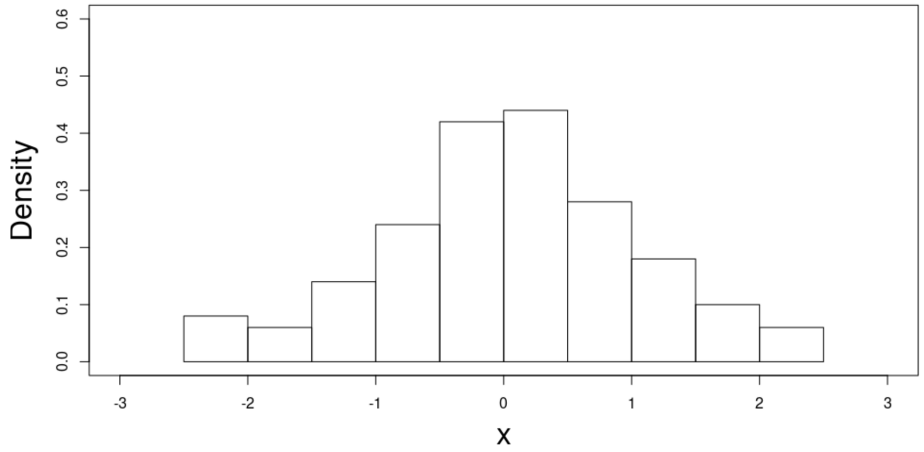
- All these histograms were generated from 100 draws from a standard normal

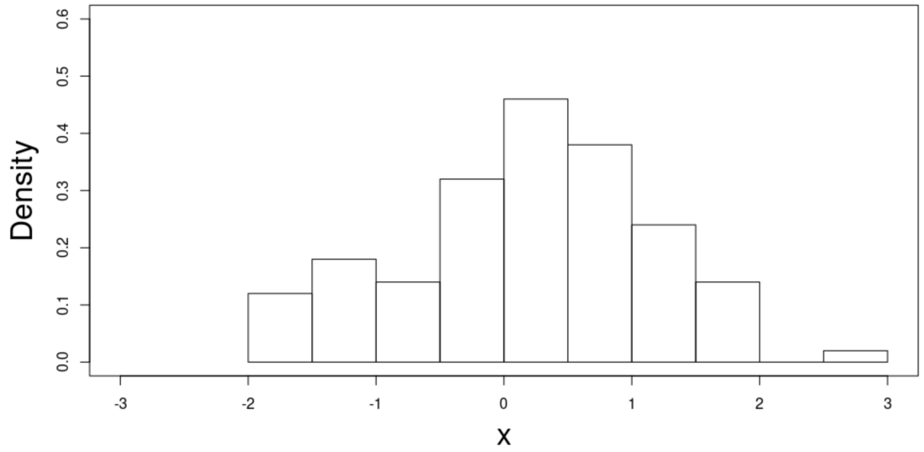
COOL CATCH

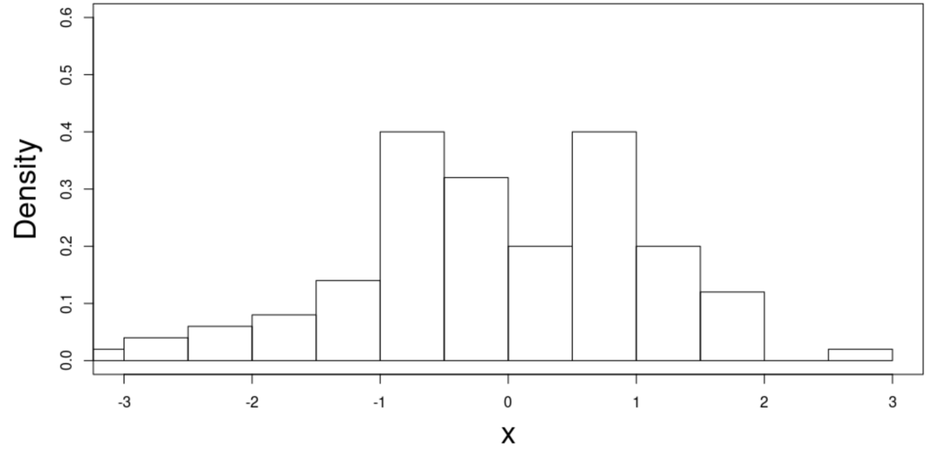
Human eyes like to look for patterns/trends. Don't mistake randomness for a signal.











Randomness in Data

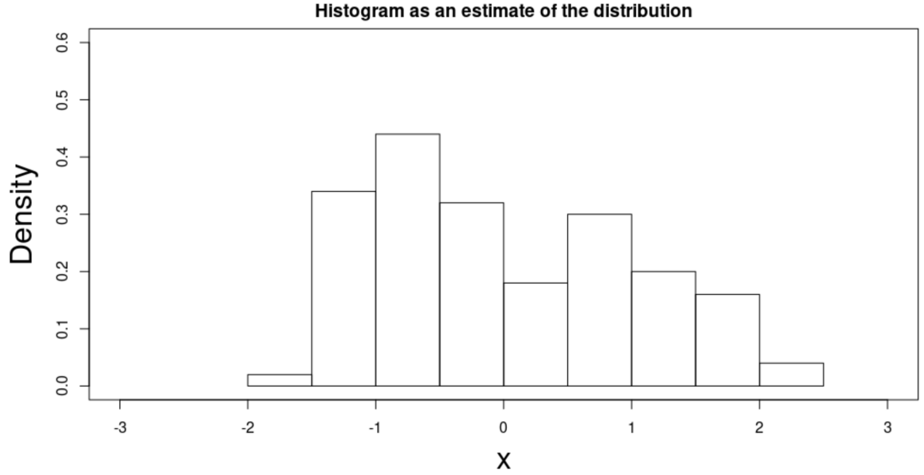
- All these histograms were generated from 100 draws from a standard normal

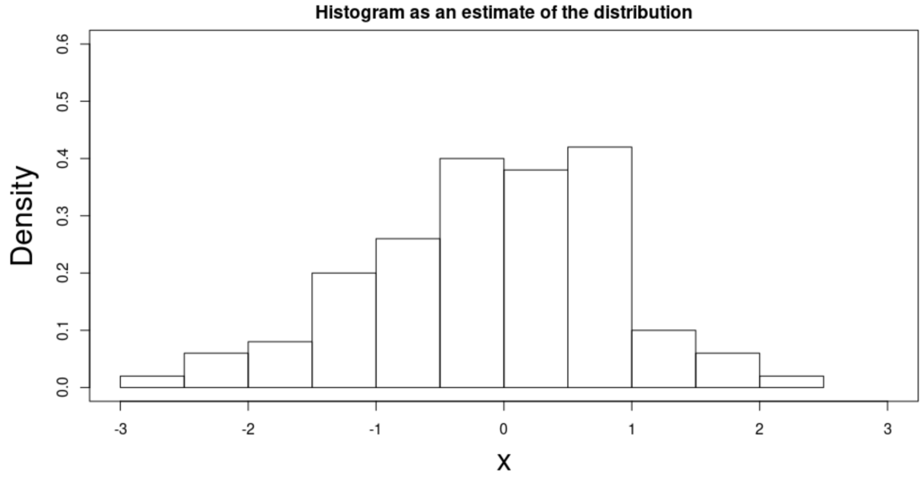
COOL CATCH

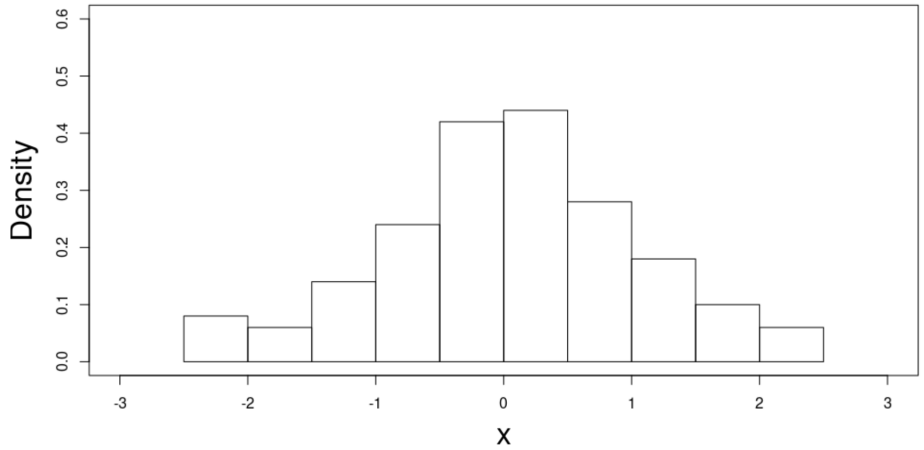
Human eyes like to look for patterns/trends. Don't mistake randomness for a signal.

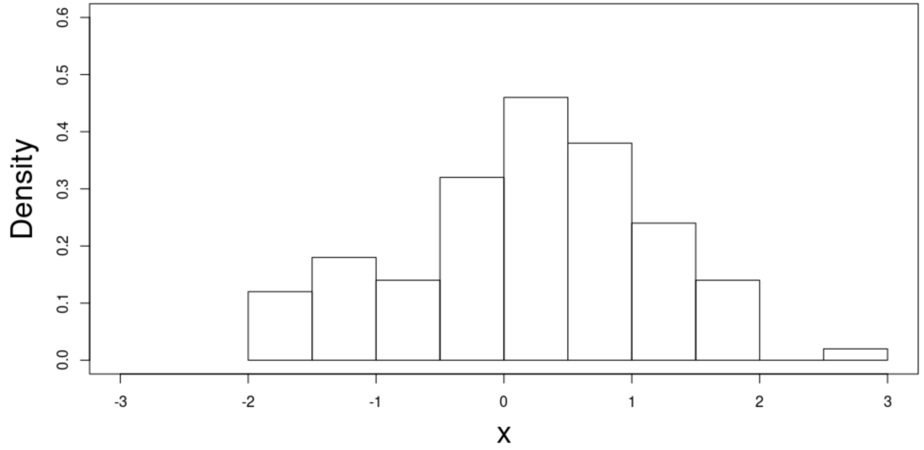
HOT TIP

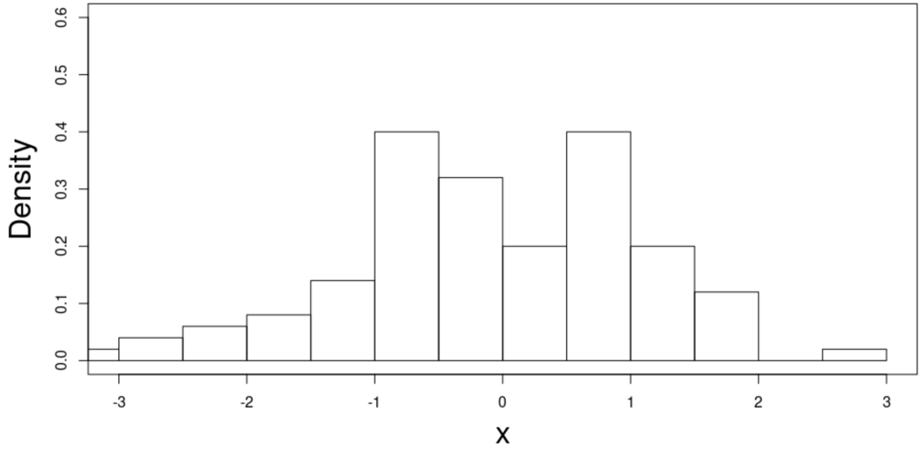
Sometimes we want to *generate* randomness to create mock data that looks "real"





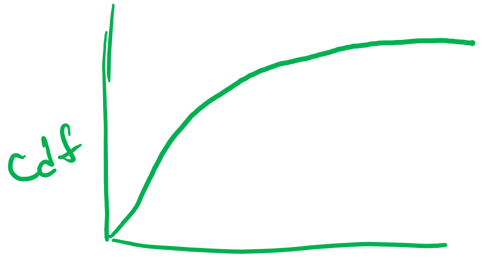






Sampling from a distribution

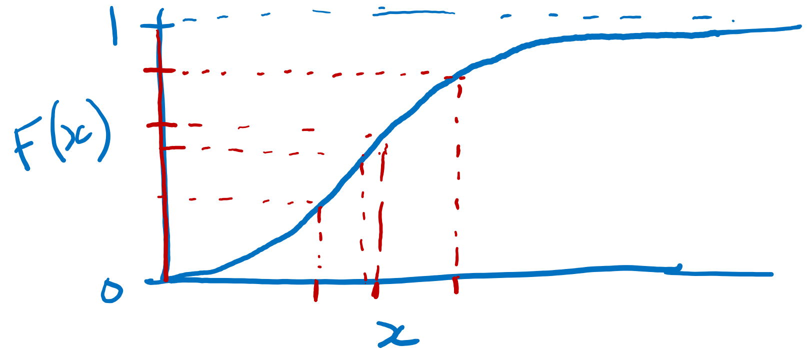
Sampling from a distribution (two basic approaches)



Inverse cdf Method

this is not always known

- First choice if the inverse cdf is tractable



$x \sim f(x)$

$F^{-1}(x)$

uniform

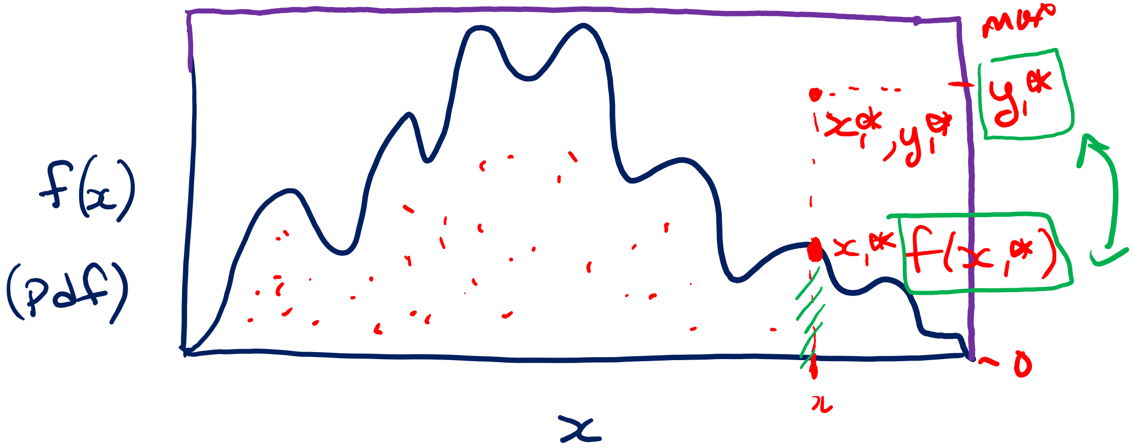
draw from $U(0,1)$

pass to $F^{-1}(x)$

$\longrightarrow x$

Accept/Reject Algorithm

- Useful when you can't write down the inverse cdf



$x_i^* \rightarrow f(x_i^*)$

compare

draw y_i^*

Mini-exercise 2

- Use the *accept-reject* approach to transform numbers generated from a uniform distribution into those following the distribution $P(x) = (1/(e-1))\exp(x)$ for $0<x<1$ and 0 elsewhere
  - Draw two random samples x^*, y^* from the U(0,1) distribution
  - If $y^* < c\,f(x^*)$, keep x^* [remember the normalization *c* here]
  - If not, draw another two random samples from the distribution
  - Continue until you have 100 samples
  - Histogram the samples and over plot the PDF

- Use *CDF sampling* to do the same thing above.
  - To do this, compute the CDF F(X) by integrating the PDF P(x) from $-\infty$ to X
  - Then find the inverse $F^{-1}(X)$of the CDF.
    [HINT: Remember an inverse function $F^{1}(x)$ is such that $F(F^{1}(x)) = x$]
  - Draw a random samples x_1 from the U(0,1) distribution
  - Then the variable $y = F^{-1}(x_1)$ will have the probability distribution you seek
  - Continue until you have 100 samples
  - Histogram the samples and over plot the PDF

Estimates of Distributions

Visualizing Empirical Distributions

- Histograms (in frequency or relative frequency)
- Boxplots
- Kernel Density Estimators
- Empirical cumulative distribution functions (ecdfs)
- Bar charts, stacked bar chart, mosaic plots, contingency tables, ...

Visualizing Empirical Distributions

- Histograms (in frequency or relative frequency)
- Boxplots
- Kernel Density Estimators
- Empirical cumulative distribution functions (ecdfs)
- Bar charts, stacked bar chart, mosaic plots, contingency tables, ...

Estimating Parameters of Distributions

- Method of moments
- Maximum Likelihood Estimators
- Bayesian inference

Box Plots

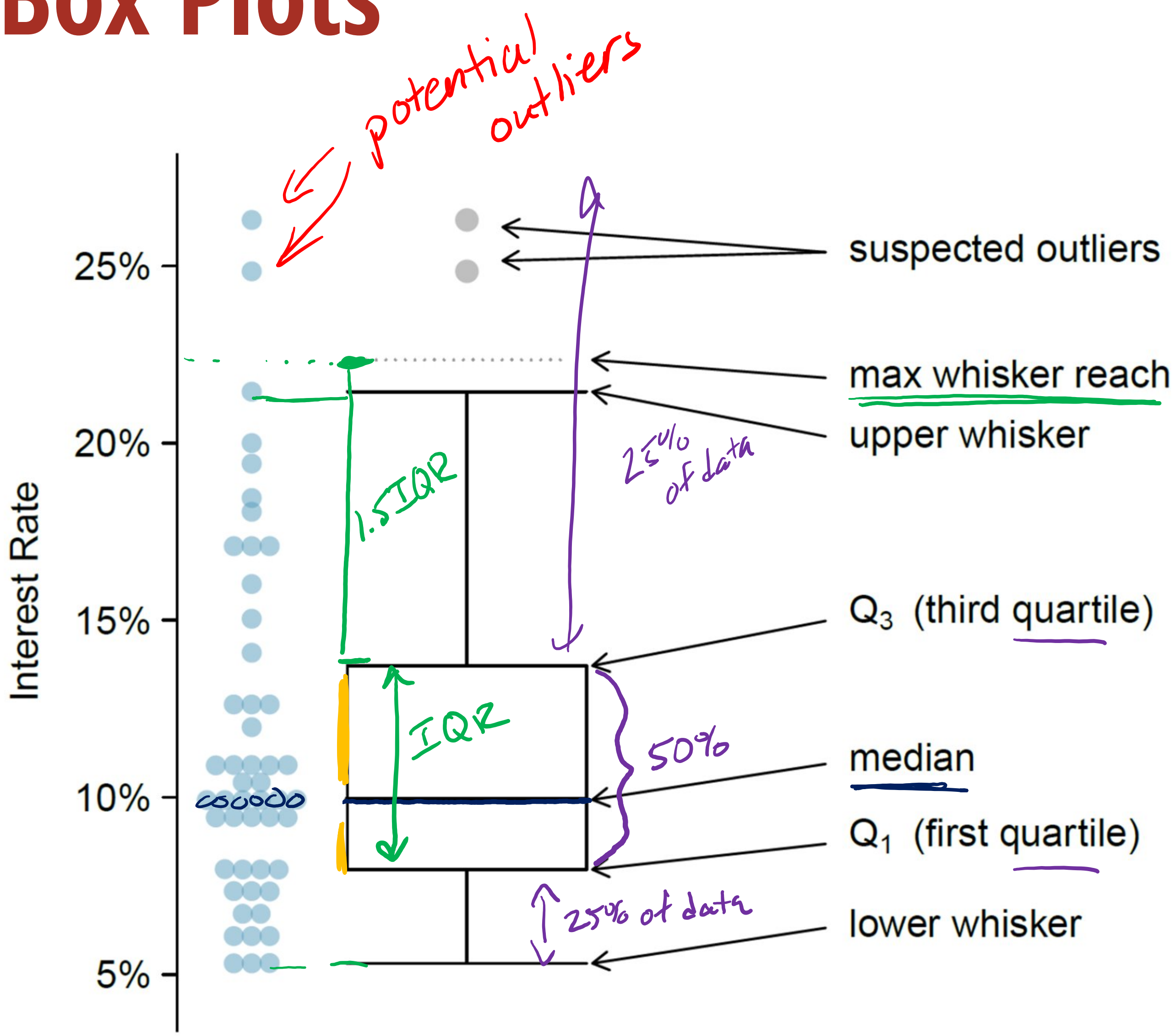
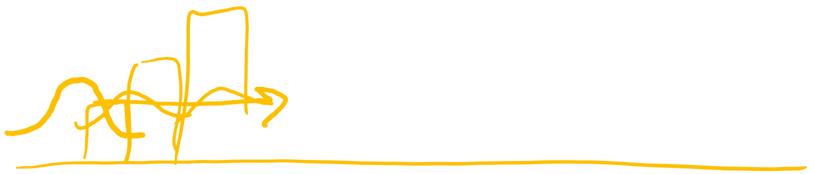


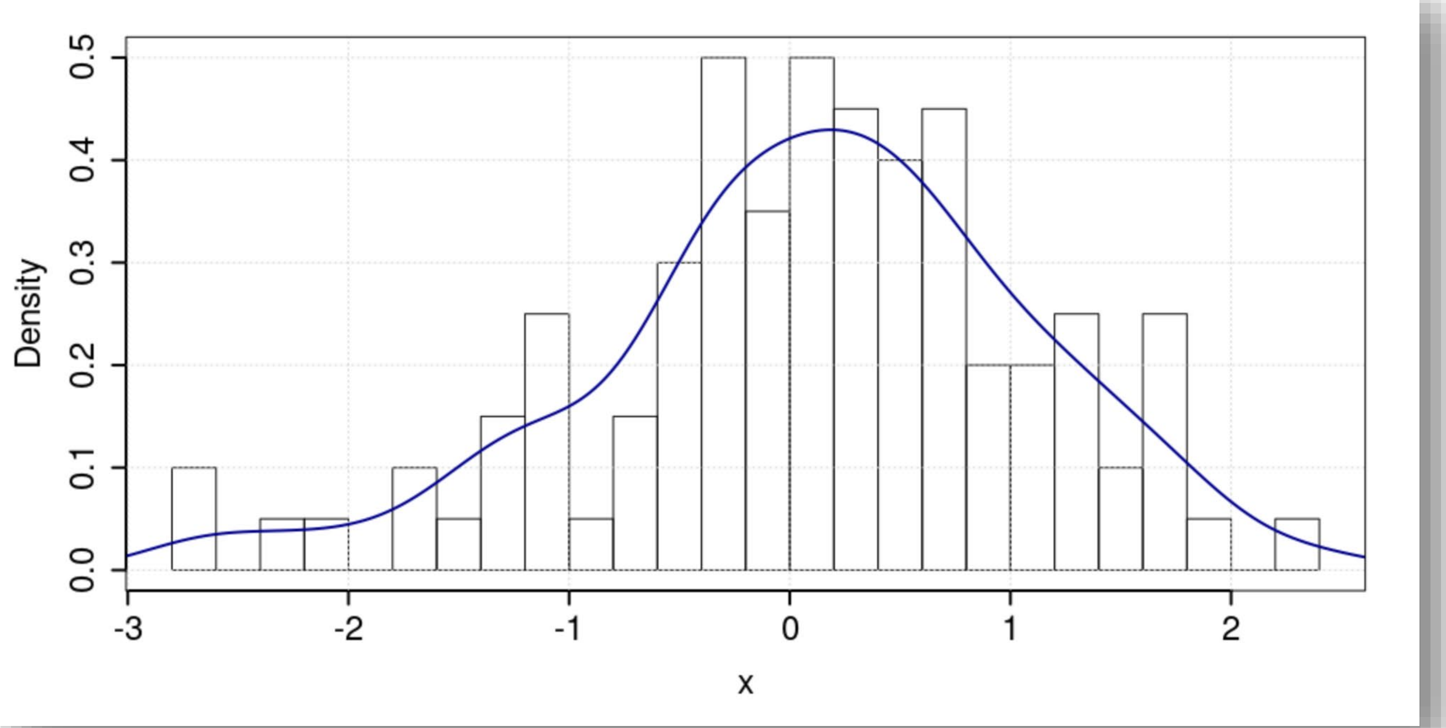
Figure 2.10, OpenIntro (4th ed.)

- Building a box plot:
  - Find the median first
  - Draw a rectangle that shows the interquartile range (IQR) $Q_3 - Q_1$
  - Extend the whiskers out to the furthest data point that is still within 1.5xIQR
  - Show the individual points that are outside the whiskers

Kernel Density Estimates

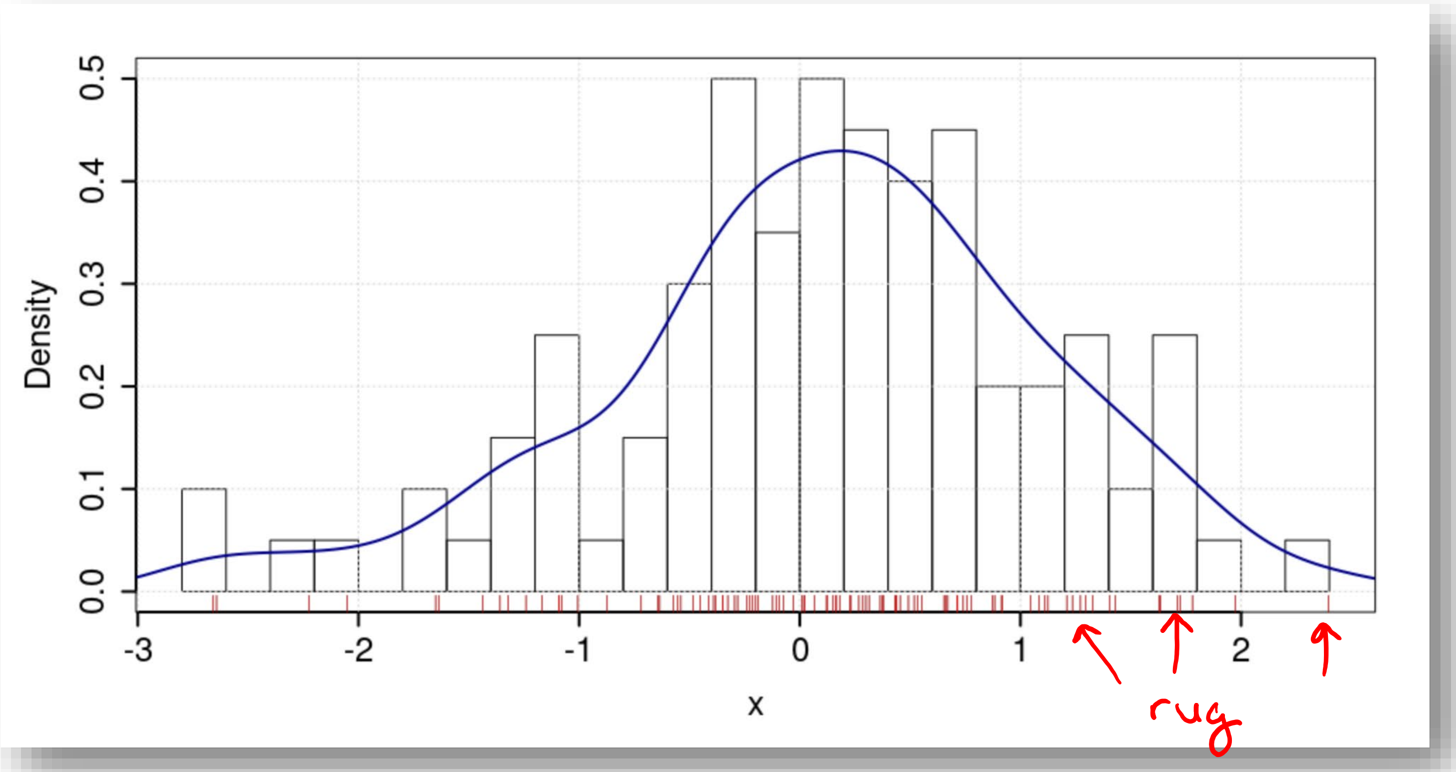


- Less sensitive to bin size, bin choice



Kernel Density Estimates

- Less sensitive to bin size, bin choice
- Helpful to add a "rug"



Kernel Density Estimates

?par → global parameters

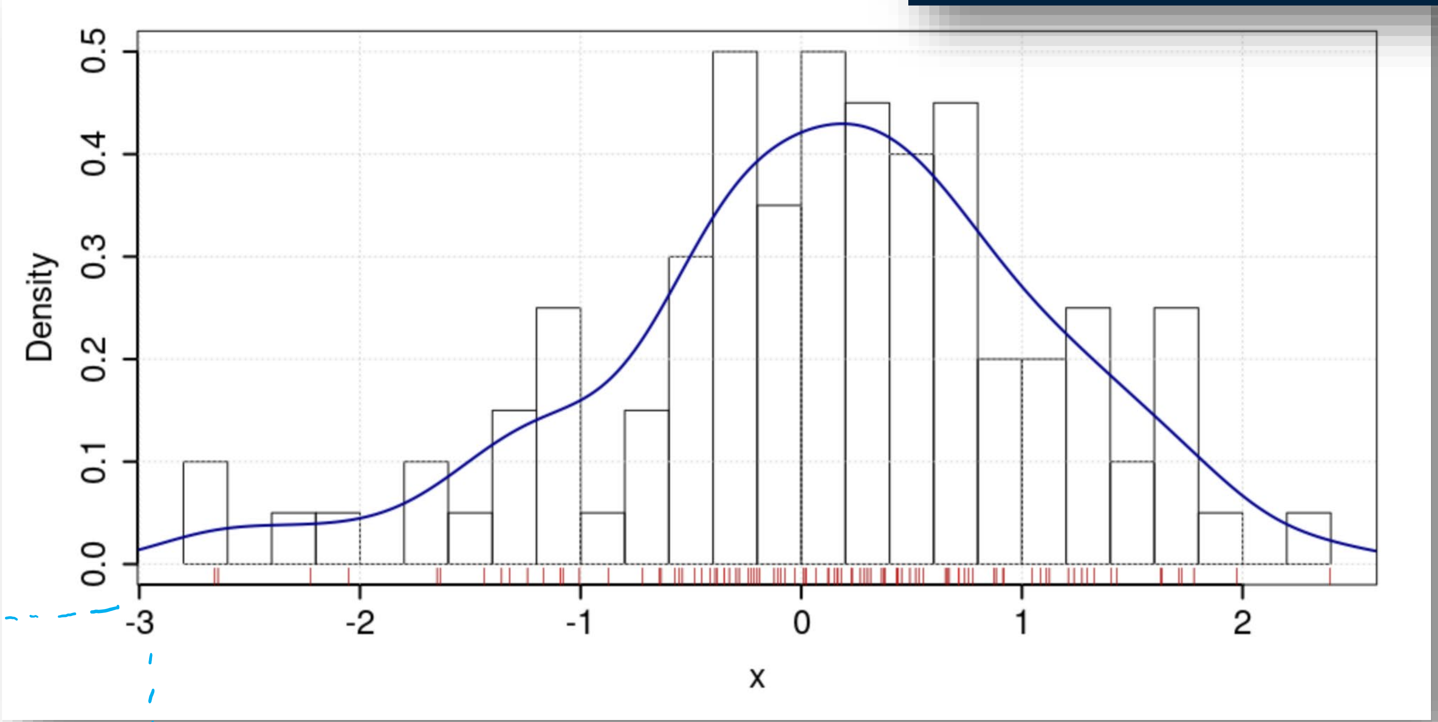
- Less sensitive to bin size, bin choice
- Helpful to add a "rug"
- (really quick to plot in R)

margins bottom left top right

```
par(mar=c(5,5,2,2))
hist(x, breaks = 20, freq = FALSE, lwd=2, main="", cex.lab=1.5, cex.axis=1.5)
grid()
lines(density(x), col="darkblue", lwd=2)
box()
rug(x, col="red")
```

← histogram

— added KDE to the plot



HOT TIP

You do not have to import any modules or packages to make these kinds of plots in R. The functions are just there!

Summary Statistics

(Tukey)

Five-Number Summary

You almost get all five from a boxplot.

- Minimum
- 1st quartile
- Median
- 3rd quartile
- Maximum

boxplot

Five-Number Summary

You almost get all five from a boxplot.

- Minimum
- 1st quartile
- Median
- 3rd quartile
- Maximum

Fivenum function is in base R

```
> moons <- c(0, 0, 1, 2, 63, 61, 27, 13)
 > fivenum(moons)
[1]  0.0  0.5  7.5 44.0 63.0
```

min   1st Q.   median   3rd Q.   max

Examples above from:
https://en.wikipedia.org/wiki/Five-number_summary

Five-Number Summary

You almost get all five from a boxplot.

- Minimum
- 1st quartile
- Median
- 3rd quartile
- Maximum

Fivenum function is in base R

```
> moons <- c(0, 0, 1, 2, 63, 61, 27, 13)
 > fivenum(moons)
[1]  0.0  0.5  7.5 44.0 63.0
```

Fivenum function must be written in Python

```
import numpy as np

def fivenum(data):
    """Five-number summary."""
    return np.percentile(data, [0, 25, 50, 75, 100], interpolation='midpoint')

moons = [0, 0, 1, 2, 63, 61, 27, 13]
print(fivenum(moons))
[  0.    0.5   7.5  44.    63. ]
```

Examples above from:
https://en.wikipedia.org/wiki/Five-number_summary

Five-Number Summary

You almost get all five from a boxplot.

- Minimum
- 1st quartile
- Median
- 3rd quartile
- Maximum

Fivenum function is in base R

```
> moons <- c(0, 0, 1, 2, 63, 61, 27, 13)
 > fivenum(moons)
[1]  0.0  0.5  7.5 44.0 63.0
```

Fivenum function must be written in Python

```
import numpy

def fivenum(c
    """Five-r
    return np

moons = [0, 0
print(fivenum
[  0.    0.5
```

HOT TIP

Want a quick way to see distribution details in python? Pip install “knowyourdata”:

```
In [3]:  kyd(x)
```

Basic Statistics

Mean: -0.0007368 **Std Dev:** 0.9813

Min:	-3.161	**-99% CI:**	-2.53
1Q:	-0.6406	**-95% CI:**	-1.905
Median:	0.01018	**-68% CI:**	-0.9957
3Q:	0.6503	**+68% CI:**	0.9445
Max:	3.651	**+95% CI:**	1.938
		+99% CI:	2.493

Array Structure

Number of Dimensions:	1
Shape of Dimensions:	(2000,)
Array Data Type:	float64
Memory Size:	15.7KiB
Number of NaN:	0
Number of Inf:	0

Five-Number Summary

You almost get all five from a boxplot.

- Minimum
- 1st quartile
- Median
- 3rd quartile
- Maximum

Summary Statistics

Includes the mean too:

- Minimum
- 1st quartile
- Median
- Mean
- 3rd quartile
- Maximum

```
> fivenum(x)
[1] -2.6609228 -0.4021807  0.1650809  0.7280392  2.3974525
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.6609 -0.3964  0.1651  0.1059  0.7216  2.3975
```

Exercise

1. Download the data set `xvalues.csv` from the website
2. Generate a histogram for these values using bin widths of 2, from -8 to 4. *Before going to part b)*, what do you notice about this distribution? Would you hypothesize what distribution the data came from?
3. Generate a new histogram for these values using bin widths of 2, starting instead from -7.
4. Make a boxplot of these data and find the summary statistics
5. Make a kernel density estimate plot of the distribution. How does this compare to the other options?
6. Based on your figures, comment on the pros and cons of each estimate of the distribution (histogram, boxplot, KDE)
7. Standardize the data from question 1, and make a new histogram and boxplot. Compare these to your histogram and boxplot in question 1.
8. What are the mean and standard deviation of the standardized data?
9. Check the 68-95-99 rule using the standardized data. Is the empirical rule applicable here? Why or why not?

?boxplot

?density

Stretch Goal:

Make an empirical CDFs of the data and compare to the CDF of a normal. Or make a Q-Q plot (look up what this is)!

Confidence intervals

Confidence intervals

- An astronomer has reported that the proportion of stars in binary systems is 0.771 with a 95% confidence interval of (0.63, 0.870).
- *What does this interval mean?*
- *Is a confidence interval a random variable?*

- http://www.rossmanchance.com/applets/ConfSim.html

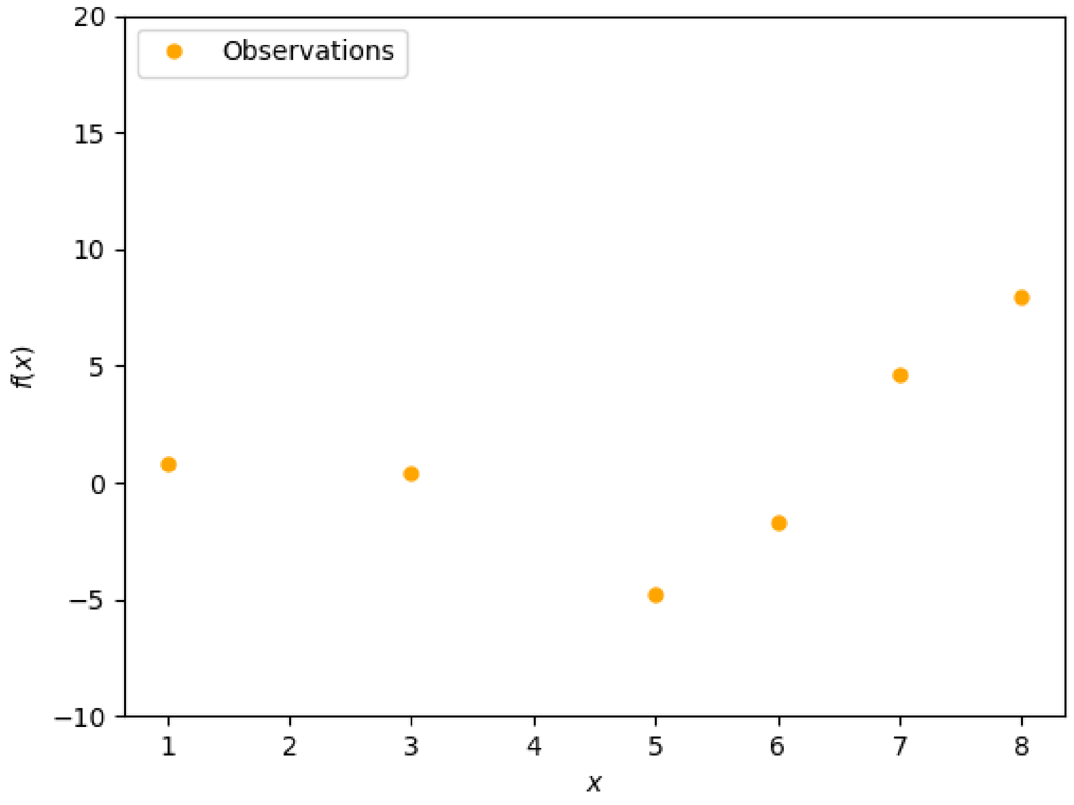
Extra slides

These slides are for extra reading once you master the earlier slides.

Smoothing and optimization

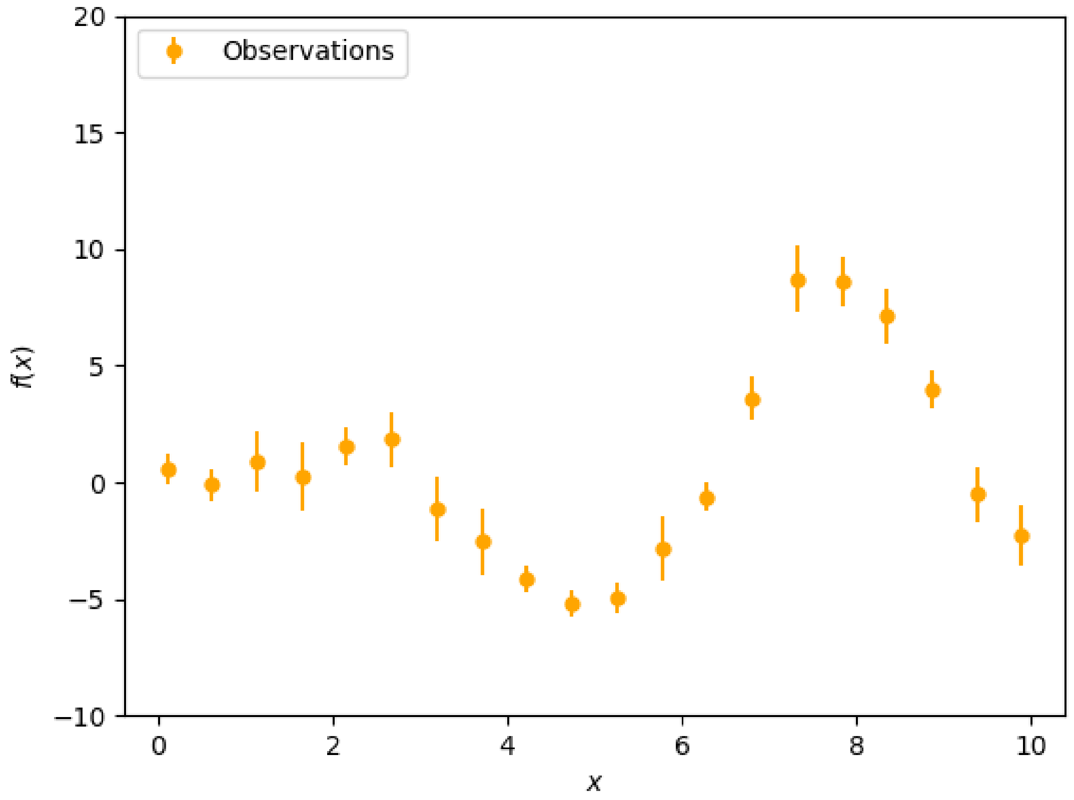
How to make data more pliable

- Sometimes you'll get data that looks like this:



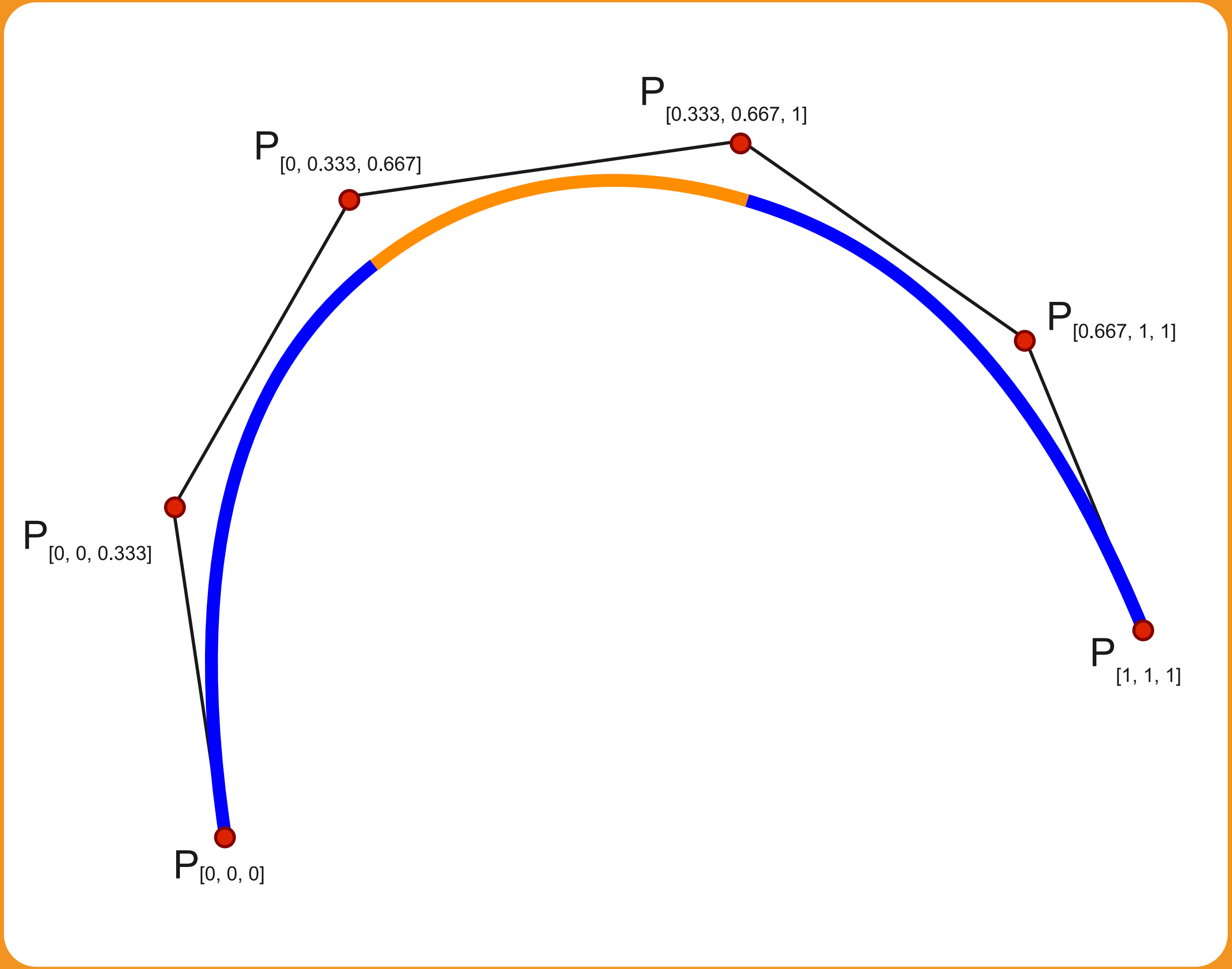
How to make data more pliable

- or this (with error bars)



Splining

- The first thing you might think to do is to spline the data.
- Spline is a polynomial fit between points that ensure that the curve you fit goes through each point you have.
- The smoothness depends on the order of the polynomial (e.g. linear, quadratic, cubic)
- Splines are very bad at *extrapolation* and can suffer if you have large gaps between points



Python's curve_fit

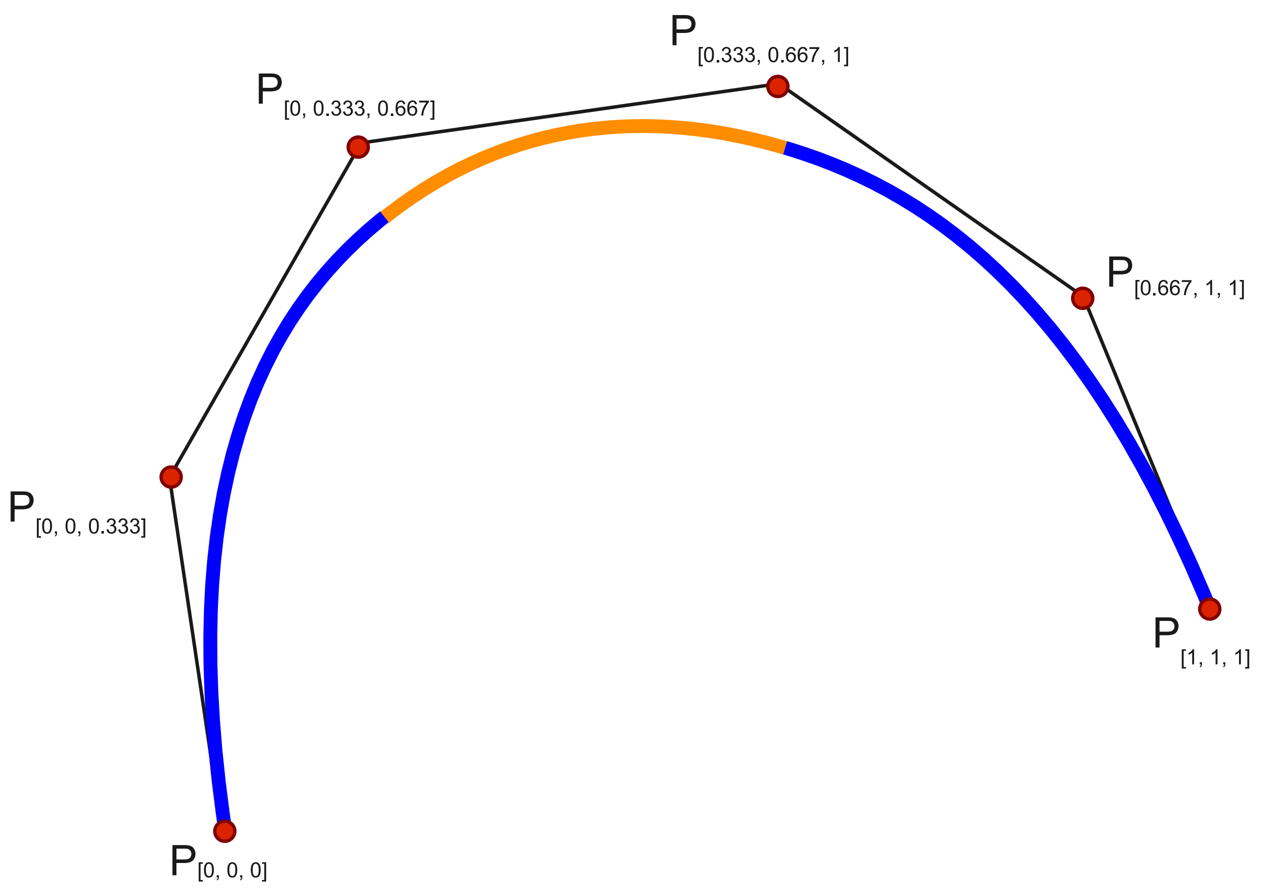
- If you think you can guess the functional form of the curve you can always fit for the parameters of that curve.
- In python that is through functions like scipy.curve_fit

"""

e_fit(ff,x,y)

HOT TIP

Curve Fitting (also known as "optimization") is an open ended problem, that leads all the way to cutting-edge deep learning of today!



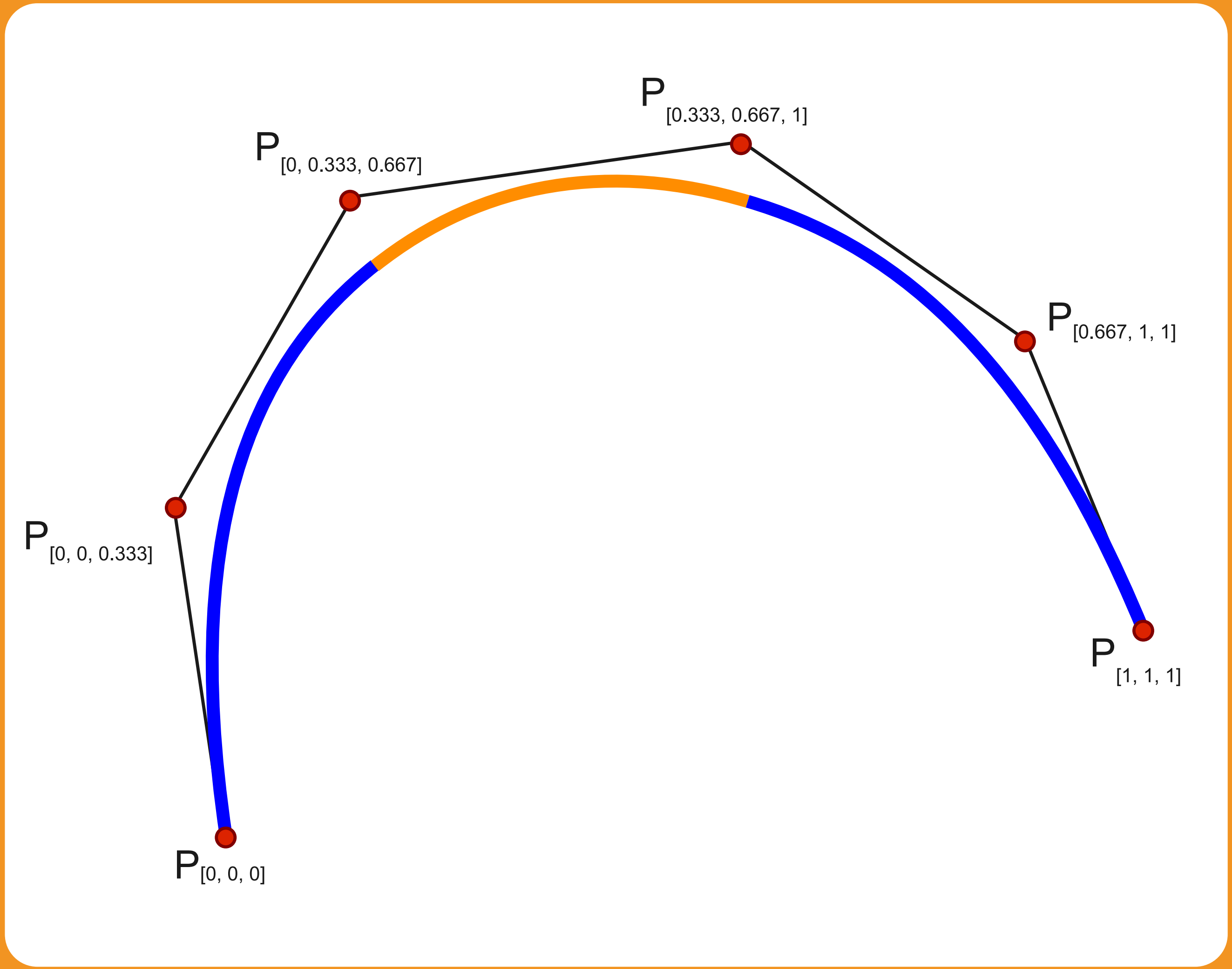
Python's curve_fit

- If you think you can guess the functional form of the curve you can always fit for the parameters of that curve.
- In python that is through functions like scipy.curve_fit

- def ff(x,a, b):

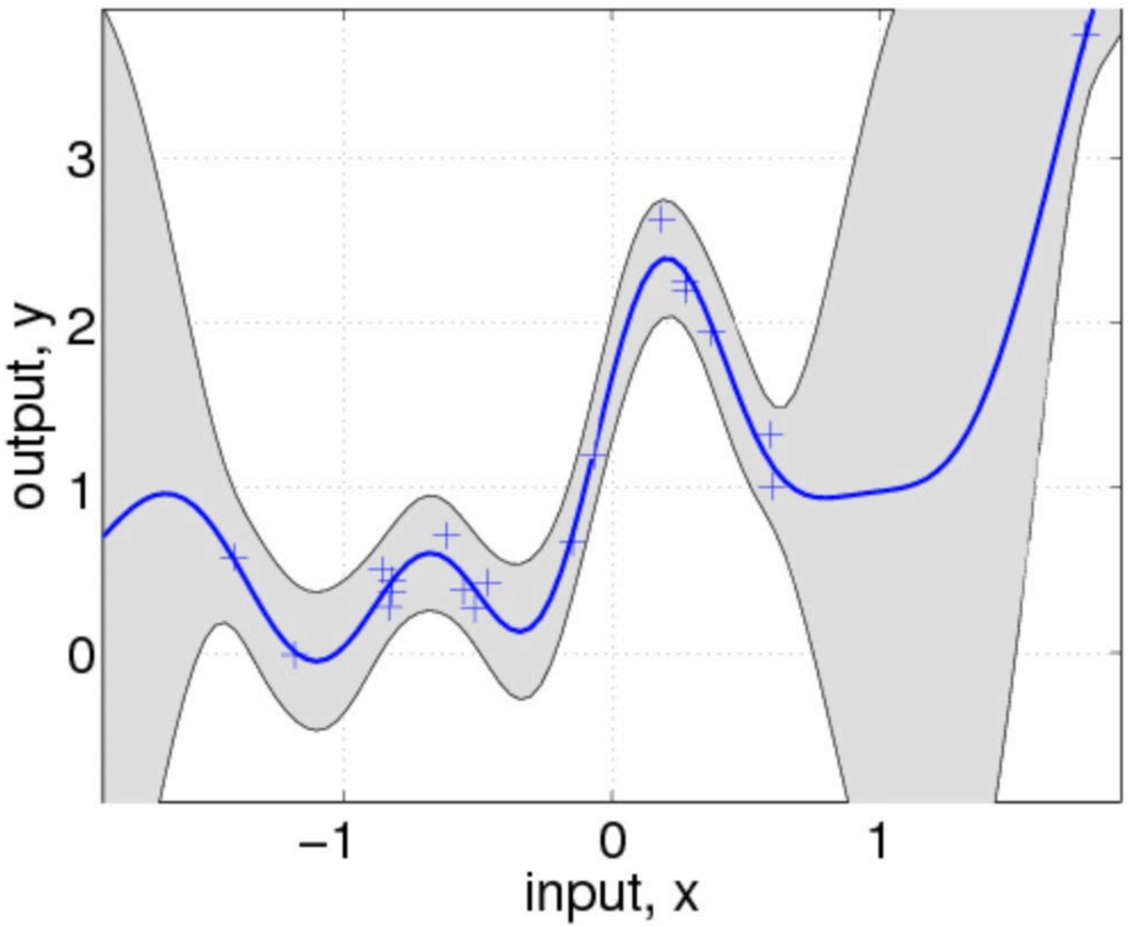
  """The function to predict."""

  return a*x * np.sin(b*x)
- fitparams, fiterror = curve_fit(ff,x,y)



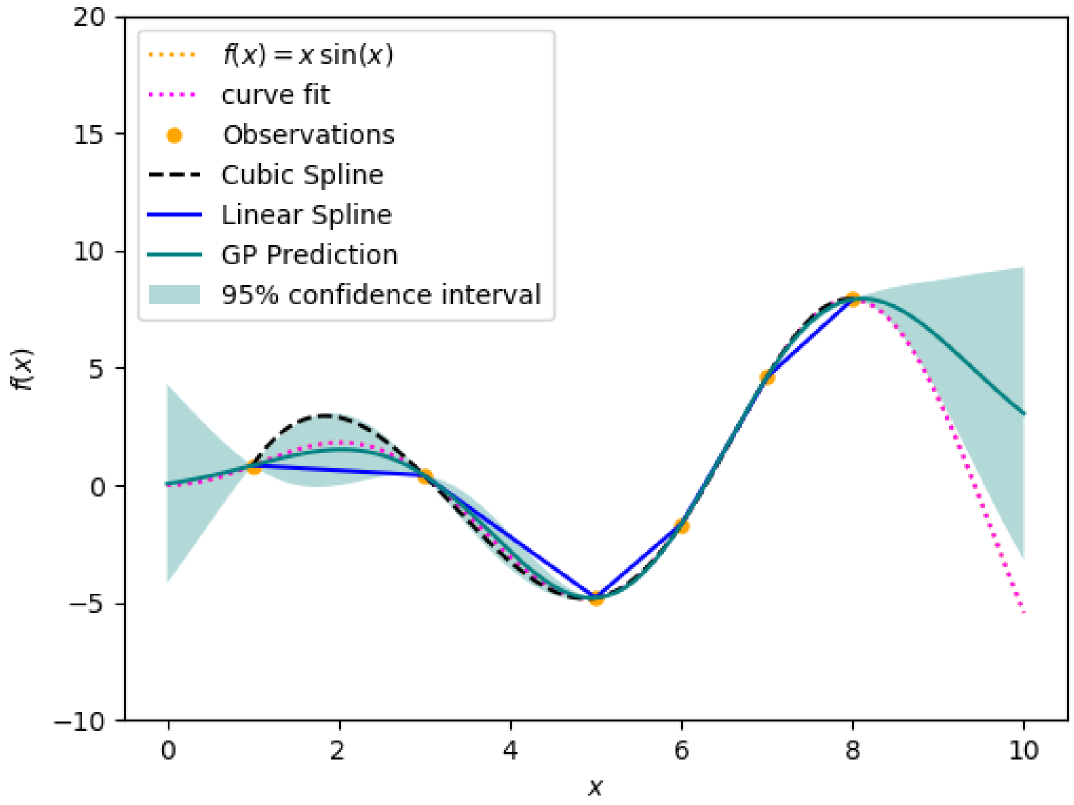
Gaussian process

- Modelling data as a Gaussian Process (GP) assumes that every point is modelled as a series of multi-variate Gaussians in a linear combination. It gives you the error on your fit -- [I like to call it the 'sausage of uncertainty'
- The key thing with GP modelling is specifying the "sigma" of the GP – or in multi-dimensional space, the *kernel*



Examples

- Simple example from Vanderplaas ++
- This code is provided in your exercise set – and shows a combination of methods



Examples

- Simple example from Vanderplaas ++
- This code is provided in your exercise set – and shows a combination of methods