Angular Measures

Degrees, arcminutes, arcseconds: $1° = 60' = 3600''$

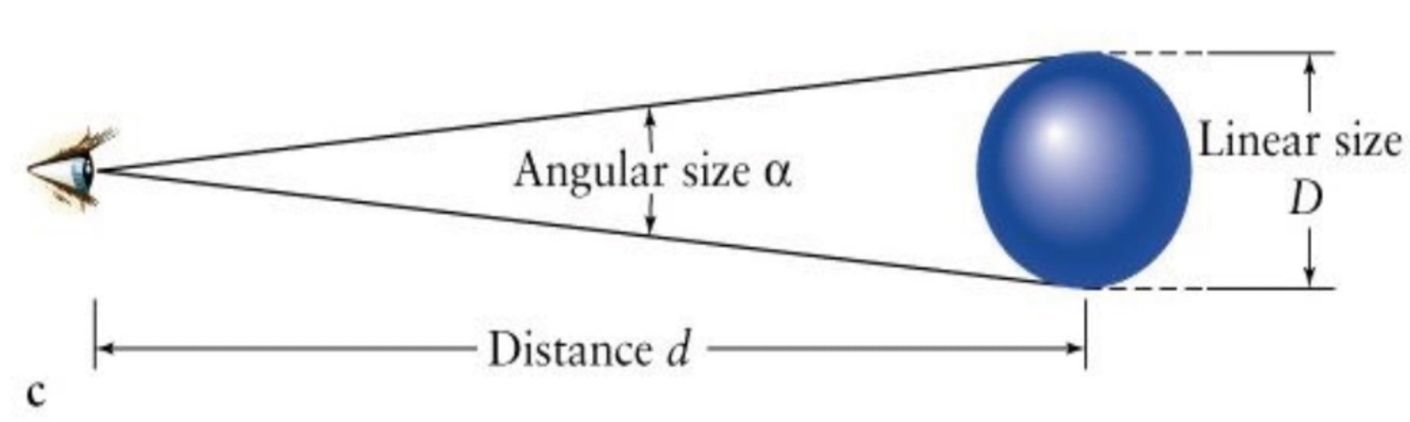
(Area in square degrees, square arcmin, square arcsec)

Radians: 2π radians $= 360°$, so 1 radian $\approx 57.3° = 206265''$

(Area: steradians, 4π steradians = whole sky)

Small Angle Approximation:

$$D = d \tan\alpha \approx d\alpha_{radians} \approx d(\alpha_{arcsec}/206265)$$



Object	Size
Sun and Moon	$\approx 0.5°$
Naked eye resolution	$\approx 1'$
Jupiter (max)	50”
Ground-based resolution	$\approx 1''$
Hubble resolution	$\approx 0.1''$
M101 (nearby spiral)	$\approx 20'$
Distant Galaxies	$< 1'$
Really Distant Galaxies	$< 2''$
Virgo Cluster	$\approx 8°$
Coma Cluster	$\approx 1°$

Coding tip: make sure to get units correct when using trig functions! They usually assume radians. If an angle is in degrees, do this:

```
np.sin(np.radians(theta))
```

Apparent Magnitude *(a measure of observed flux)*

Apparent magnitude (*m*) is the apparent brightness (flux) of an object as seen in the sky.

$$m = -2.5\log_{10}(f) + const$$

$$m_1 - m_2 = -2.5\log_{10}(f_1/f_2)$$

So:

- $\Delta m = 1$ mag → factor of ≈ 2.512 in flux
- $\Delta m = 5$ mag → factor of exactly 100 in flux
- $\Delta m = 10$ mag → factor of 100^2 = 10,000 in flux

Using differential calculus, ***if the uncertainties*** (σ) ***are small***, you can show that

$$\sigma_m = -1.086\left(\frac{\sigma_f}{f}\right) \approx \left(\frac{\sigma_f}{f}\right)$$

In other words, the uncertainty in magnitudes is approximately equal to the fractional uncertainty in flux. So, for example, a 0.1 mag uncertainty is a 10% flux uncertainty.

Object	m_V
Sun	≈ -27
Moon	≈ -13
Jupiter (max)	-2.9
Vega	0.03
Aldebaran (RGB)	0.9
Naked Eye Limit	≈ 6
Bright galaxies	$\approx 8 - 10$
SDSS faint limit	≈ 23
Aldebaran in LMC	≈ 18
Aldebaran in Virgo	≈ 30
Hubble UDF limit	≈ 31

Coding tip: remember, mags use log10.

```
🙁 np.log()   : natural log
🙂 np.log10(): base 10 log
```

Absolute Magnitudes *(and distances)*

Absolute magnitude (M) is the apparent magnitude an object would have **if** it were at a distance of 10 pc.

$$m - M = 5\log_{10}(d) - 5$$

- Distance (d) ***must*** be measured in parsecs.
- $m - M$ is known as the **distance modulus**

Again, using differential calculus, ***if the uncertainties*** (σ) ***are small***, you can show that

$$\frac{\sigma_d}{d} \approx 0.5\sigma_{(m-M)}$$

In other words, the fractional uncertainty in distance is approximately equal to the uncertainty in distance modulus. So, for example, a 0.1 mag uncertainty in distance modulus is a 5% distance uncertainty.

Object	Distance	Modulus
α Centauri	1.3 pc	-4.4
star @ 10pc	10 pc	0.0
Orion Nebula	415 pc	8.1
Galactic Center	8.2 kpc	14.6
Large Magellanic Cloud	50 kpc	18.5
Andromeda Galaxy	750 kpc	24.4
Virgo Cluster	16.5 Mpc	31.1
Coma Cluster	100 Mpc	35.0

Absolute Magnitudes *(and luminosity)*

Since absolute magnitude is the apparent magnitude at a fixed distance (10pc), it is a measure of luminosity.

$$M_1 - M_2 = -2.5\log_{10}(L_1/L_2)$$

If we take object #2 to be the Sun, we have

$$M - M_\odot = -2.5\log_{10}(L/L_\odot)$$

or

$$L = 10^{-0.4(M-M_\odot)}L_\odot$$

Object	M_V	$L_V/L_{V,\odot}$
Sun	+4.83	1.000
Vega	+0.58	80
Betelgeuse	−5.8	17,000
Large Magellanic Cloud	≈ −18.0	1.5×10^9
Andromeda Galaxy	≈ −21.7	4.0×10^{10}
M87 (giant E)	≈ −22.5	8.0×10^{10}

Remember, magnitudes are generally defined in a filter bandpass, so the luminosity refers to the luminosity ***in that bandpass****.*

Total luminosity summed over all wavelengths is called the ***bolometric luminosity****, and is almost never what we work with.*

Surface Brightness *(flux per area)*

Galaxies are extended objects; their light is spread out over a region of the sky.

To measure flux, we had the magnitude equation:

$$m = -2.5 \log_{10}(f) + const$$

In a similar way, we can define surface brightness μ as flux f per unit angular area ω:

$$\begin{aligned} \mu &= -2.5 \log_{10}(f/\omega) + const \\ &= -2.5 \log_{10}(f) + 2.5 \log_{10}(\omega) + const \\ &= m + 2.5 \log_{10}(\omega) \end{aligned}$$

VERY VERY IMPORTANT: the units are described as mag/arcsec^2, but this is not mathematically correct. Magnitudes are a logarithmic unit and you cannot multiply or divide magnitudes!

$$\mu \neq m/\omega$$

Object	μ_V [mag/arcsec^2]
Sun	−9.5
Moon	+4.5
M87 (Elliptical galaxy) center	≈ 17
UGC927 (Spiral galaxy) center	≈ 21
Dark night sky	≈ 21.8
Low surface brightness galaxy	≈ 23-26
Ultradiffuse galaxy	≈ 26-28

Surface brightness does not change with distance – it is an* ***intrinsic*** *property of a galaxy.*

** at least not as long as you are working at low redshifts, $z <$ a few tenths or so.*

Filters and Colors

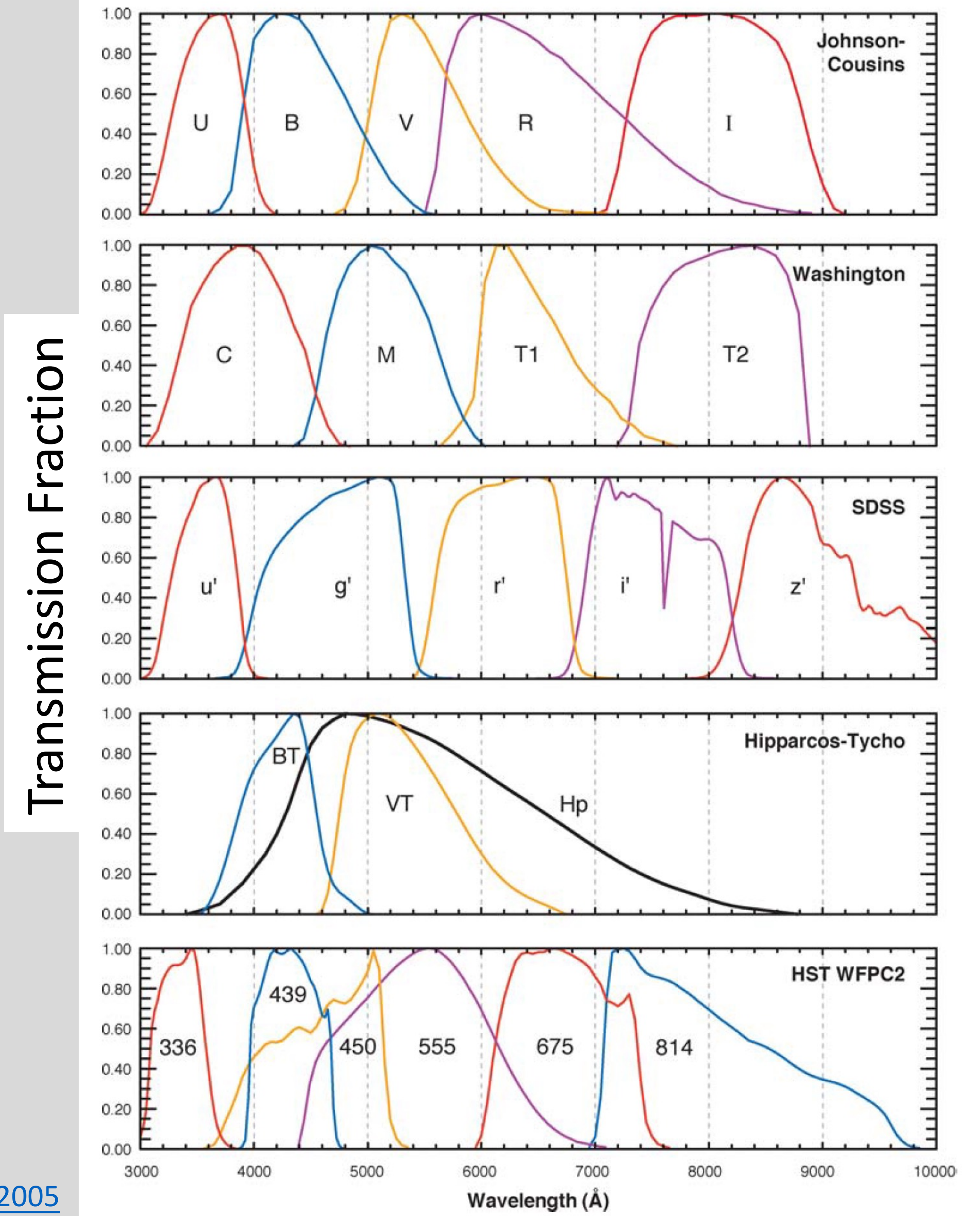
We measure fluxes/magnitudes through different filter bandpasses and define colors as the difference in magnitudes.

$$B - V = m_B - m_V = M_B - M_V$$

Convention: always list the bluer filter first, then smaller or more negative colors mean bluer objects.

Dust reddens and extincts starlight.
(colors get numerically larger, magnitudes get numerically bigger)

Caution: in the (common) Vega magnitude system, $B - V = 0$ does not mean equal fluxes in B and V. It means the same color as Vega, which is a very blue star.



Bessell ARAA 2005

Color-Magnitude Diagrams (CMDs)

The key to understanding stars and stellar populations.

Stars live most of their lives on the main sequence.

Massive stars:

- bright and blue on the MS; live fast, die young (< 100 Myr)
- at end of life, evolve across the CMD to become red supergiants, then go supernova

Low mass stars:

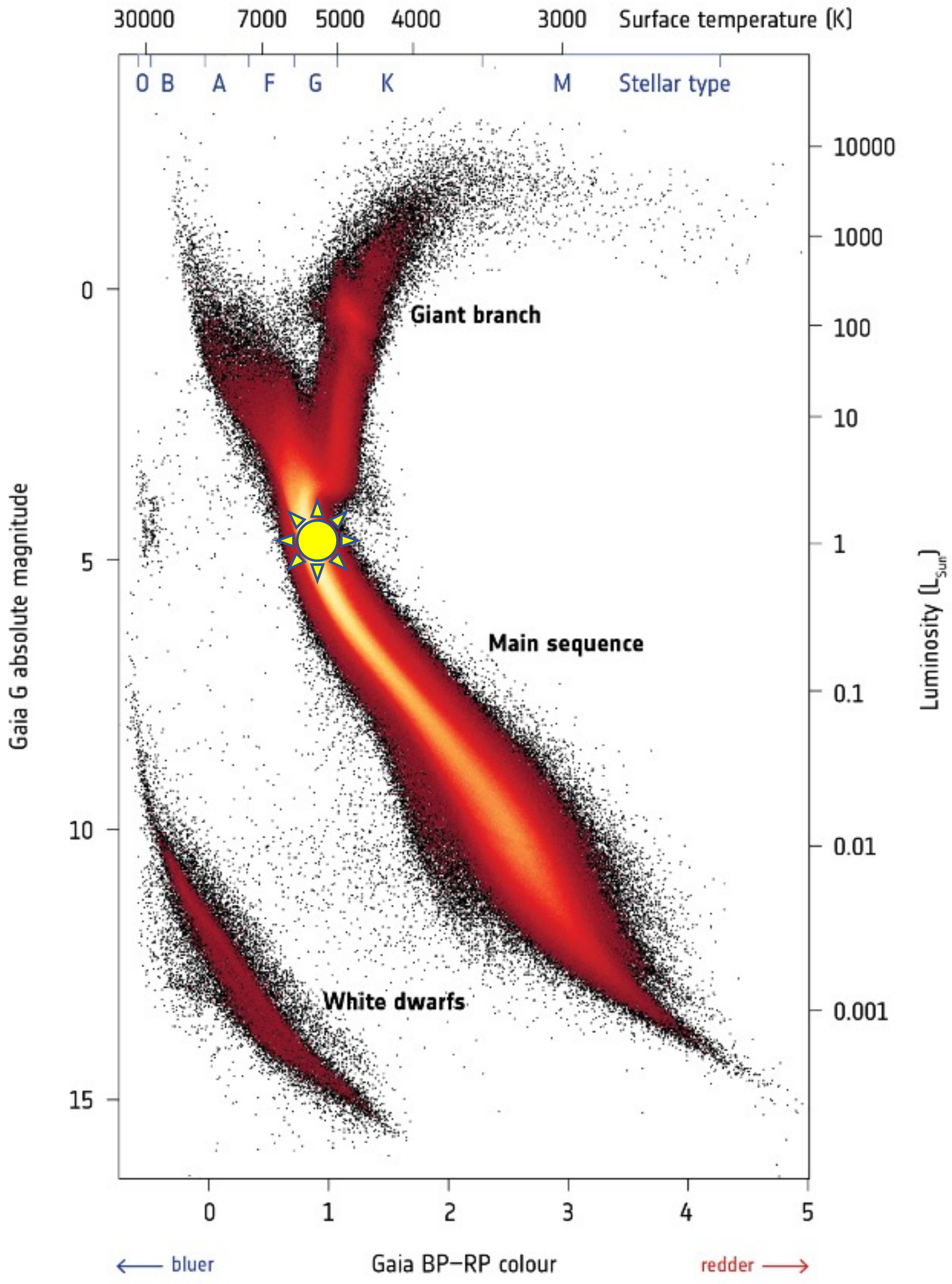
- fainter and redder on the MS; live for 10+ Gyr
- at end of life evolve up the CMD to become red giants, then eject outer layers and become a white dwarf.

Metallicity:

- Metal-poor stars are slightly bluer than metal-rich ones.

In a galaxy, the number of stars in different parts of the CMD is determined by galaxy's star formation and chemical enrichment history.

How far "down" the main sequence you can see depends on the depth of the data and the distance to the galaxy.



CMD of the solar neighborhood from Gaia

Units and Conversions

I beg you, please don't use SI units.

Natural units for Galactic and extragalactic astronomy:

- distance: parsecs [pc], kiloparsecs [kpc], megaparsecs [Mpc]
- time: years [yr] or millions of years [Myr]
- mass: solar masses [$M_\odot$]
- speed: km/s

Handy "close-enough" conversions:

- 1 year ≈ $\pi \times 10^7$ seconds
- 1 km/s ≈ 1 pc/Myr ≈ 1 kpc/Gyr

Gravitational Constant, converting to stellar/planetary units (AU, $M_\odot$, yr):

$$G = 4\pi^2 \text{ AU}^3 M_\odot^{-1} \text{ yr}^{-2}$$

Gravitational Constant, converting to extragalactic units:

$$G \approx 4.3\times10^{-3} \text{ pc (km/s)}^2 \, M_\odot^{-1}$$

Coding tip: learn astropy's units functionality:

https://docs.astropy.org/en/stable/units/

Finding Data and Articles *(a sampling)*

Object or Coordinate based inquiries:

"I want data about this galaxy", or
"I want observations of objects in this part of the sky"

- NASA Extragalactic Database: http://ned.ipac.caltech.edu/
- Simbad: http://simbad.u-strasbg.fr/simbad/
- SDSS Skyserver: http://skyserver.sdss.org
- NASA Skyview: http://skyview.gsfc.nasa.gov/

Catalog based inquiries :

"I want a catalog of galaxies in the Virgo cluster", or
"I want a catalog of stars observed by Gaia"

- Vizier: http://vizier.u-strasbg.fr/
- specific data/mission websites

Literature targeted inquiries :

"I want to find papers that talk about the morphology-density relationship", or
"I want that McGaugh paper about UGC 628"

- ADS Abstract Service: https://ui.adsabs.harvard.edu/

What about Google and/or Wikipedia?

Fine as a starting point, bad as an ending point.
We want to be using scientific data sources.

Coding tip: for analysing catalog-based data, I strongly recommend installing astroquery in your Python/Astropy toolset.

Understanding a distribution of measurements

Let's say you have a repeated measurements of some value. How do we estimate the best value and uncertainty.

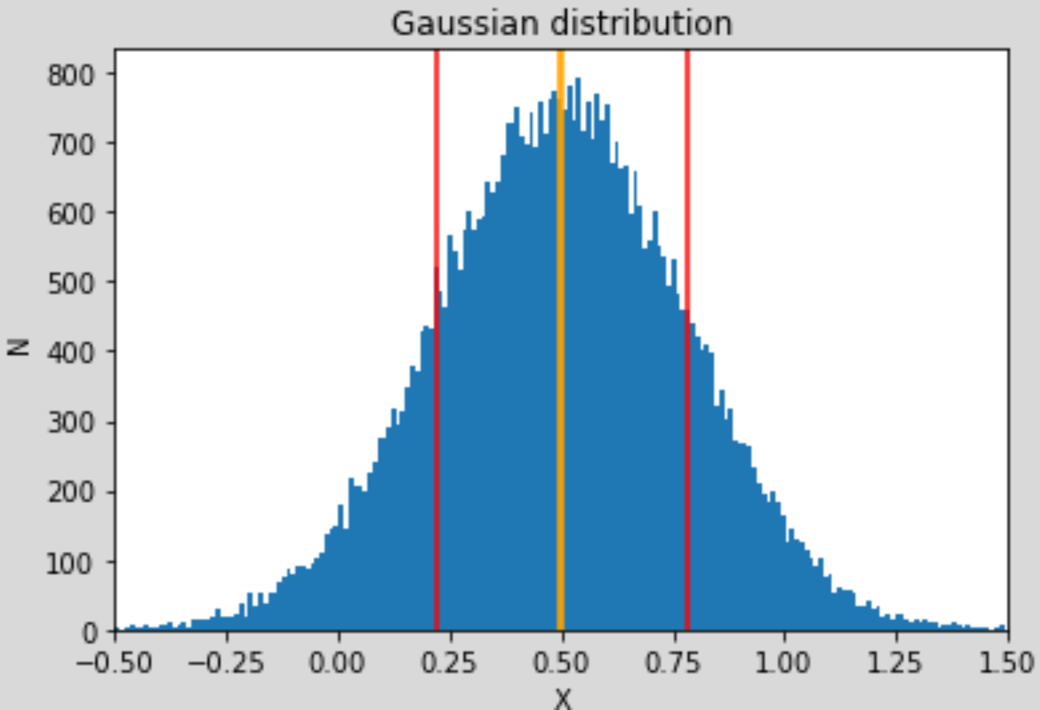
If your errors are independent and follow a Gaussian distribution:

- measure mean and standard deviation $(\bar{x}, \sigma)$
- "standard error in the mean" is given by $\sigma/\sqrt{N}$

```
mean     = np.average(data)
stdev    = np,std(data)
mean_err = stdev/np.sqrt(len(data))
```

Is this a good assumption? Take a distribution of 50,000 measurements with $\bar{x}, \sigma$ = 0.5, 0.28, look at distribution.

(yellow: mean, red: mean +/- 1σ)



Understanding a distribution of measurements

Let’s say you have a repeated measurements of some value. How do we estimate the best value and uncertainty.

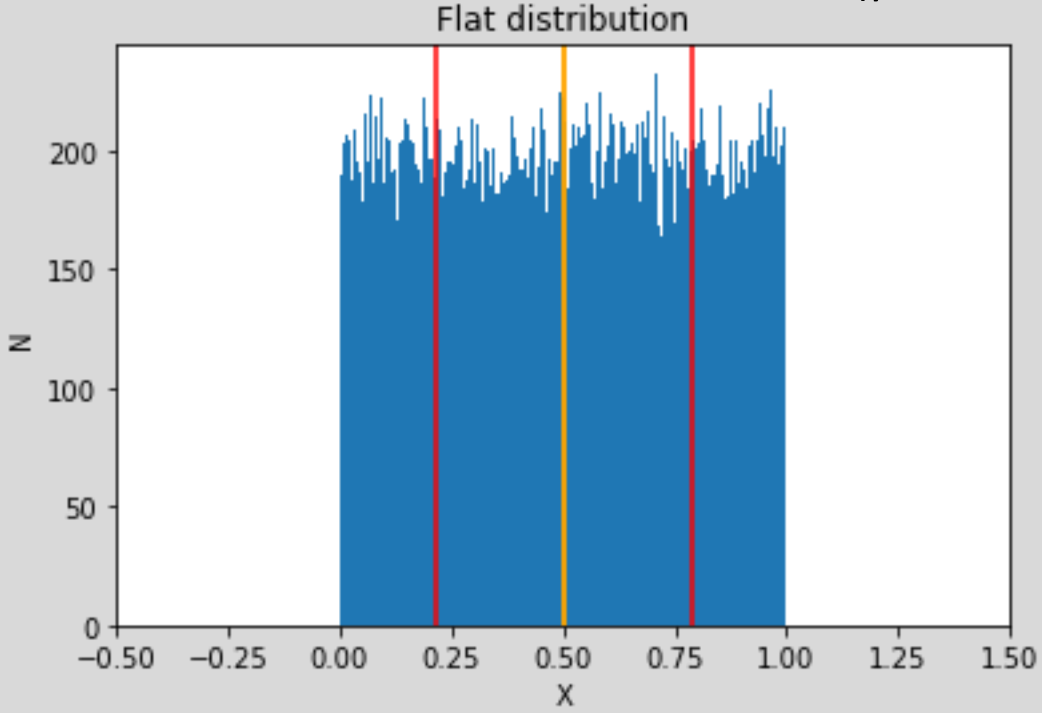
If your errors are independent and follow a Gaussian distribution:

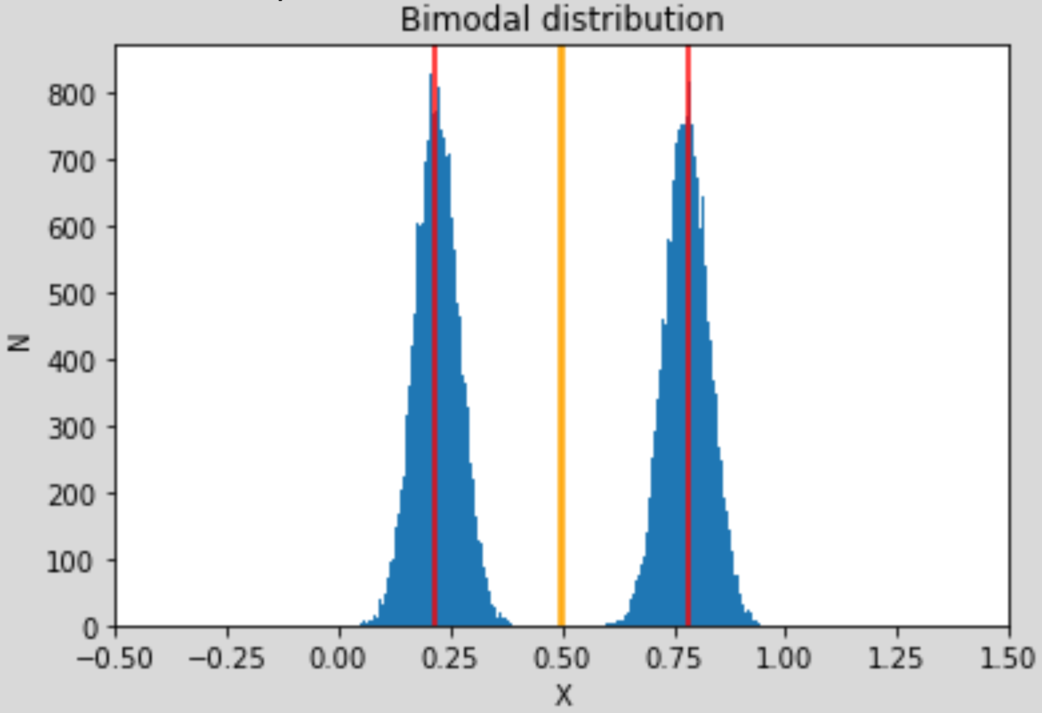
- measure mean and standard deviation $(\bar{x}, \sigma)$
- “error in the mean” is given by $\sigma/\sqrt{N}$

```
mean     = np.average(data)
stdev    = np,std(data)
mean_err = stdev/np.sqrt(len(data))
```

But other distributions can mimic the same answer, and may or may not be meaningful!

(yellow: mean, red: mean +/- 1σ)





Understanding a distribution of measurements

Let's say you have a repeated measurements of some value. How do we estimate the best value and uncertainty.

If your errors are independent and follow a Gaussian distribution:

- measure mean and standard deviation $(\bar{x}, \sigma)$
- "error in the mean" is given by $\sigma/\sqrt{N}$

```
mean     = np.average(data)
stdev    = np,std(data)
mean_err = stdev/np.sqrt(len(data))
```

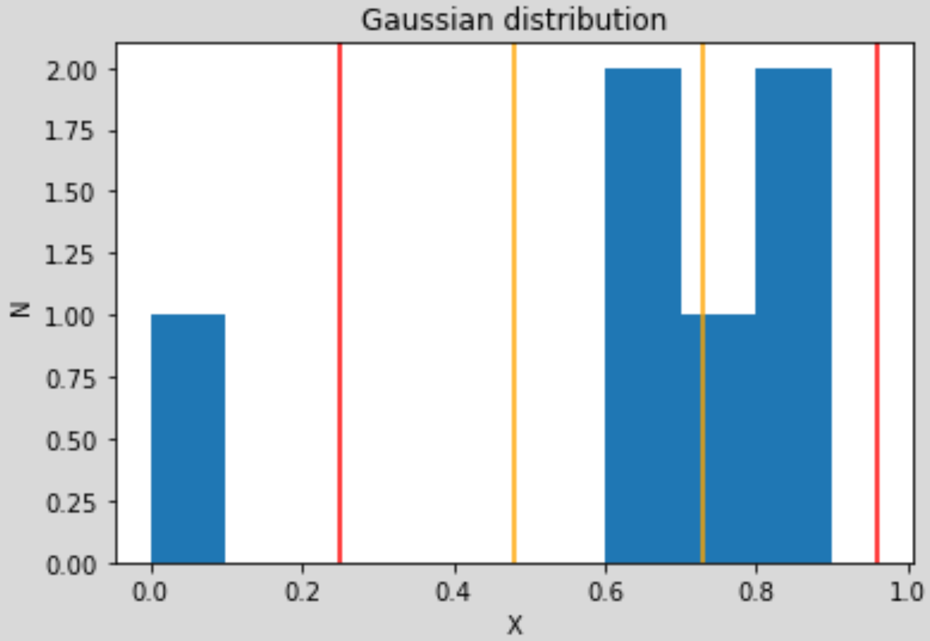
And when the amount of data is small, it can be hard to tell if these are good estimates!

(yellow: mean +/- error in mean, red: mean +/- 1σ)

Moral of the story:

Gotta look at your data!

Do a `plt.hist(data)` to be sure.



Simple Gaussian Propagation of Errors: Adding, Subtracting, Averaging

If you are adding or subtracting two things with uncertainties, the total uncertainty is the **quadrature sum** of the individual uncertainties:

$$z = x \pm y$$

$$\sigma_z^2 = \sigma_x^2 + \sigma_y^2$$

I you are averaging many data values together ($x_i \pm \sigma_{x_i}$) to get a final "best estimate" of what's being measured, the uncertainty on that estimate is given by the **standard error of the mean**:

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

$$\sigma_{\bar{x}} = \frac{1}{\sqrt{N}}\sigma_{x_i}$$

Simple Gaussian Propagation of Errors: Using a linear (or linearized) function

$$y = f(x) = ax + b$$

Propagate errors using the gradient method, adding in quadrature the error due to each of a, x, and b :

$$\sigma_y^2 = \left[\left(\frac{\partial f}{dx}\right)\sigma_x\right]^2 + \left[\left(\frac{\partial f}{da}\right)\sigma_a\right]^2 + \left[\left(\frac{\partial f}{db}\right)\sigma_b\right]^2$$

so

$$\sigma_y^2 = [a\sigma_x]^2 + [x\sigma_a]^2 + [\sigma_b]^2$$

Total uncertainty

Uncertainty due to the measurement of x

Uncertainty due to the uncertainties in a and b

Characterizing a linear (or linearized) relationship:

- Dataset of N points: $(x_i\ , y_i)$
- Fit a line to data: $y = mx + b$
- Calculate **slope**, **intercept**, and their **uncertainties**: $m \pm \sigma_m, b \pm \sigma_b$
- Calculate root-mean-square (RMS) **scatter** around the fit: $\sigma_{RMS}^2 \equiv \frac{1}{N}\Sigma\left(y_i - y_{fit}\right)^2 = \frac{1}{N}\Sigma\left(y_i - (mx_i + b)\right)^2$

The importance of scatter

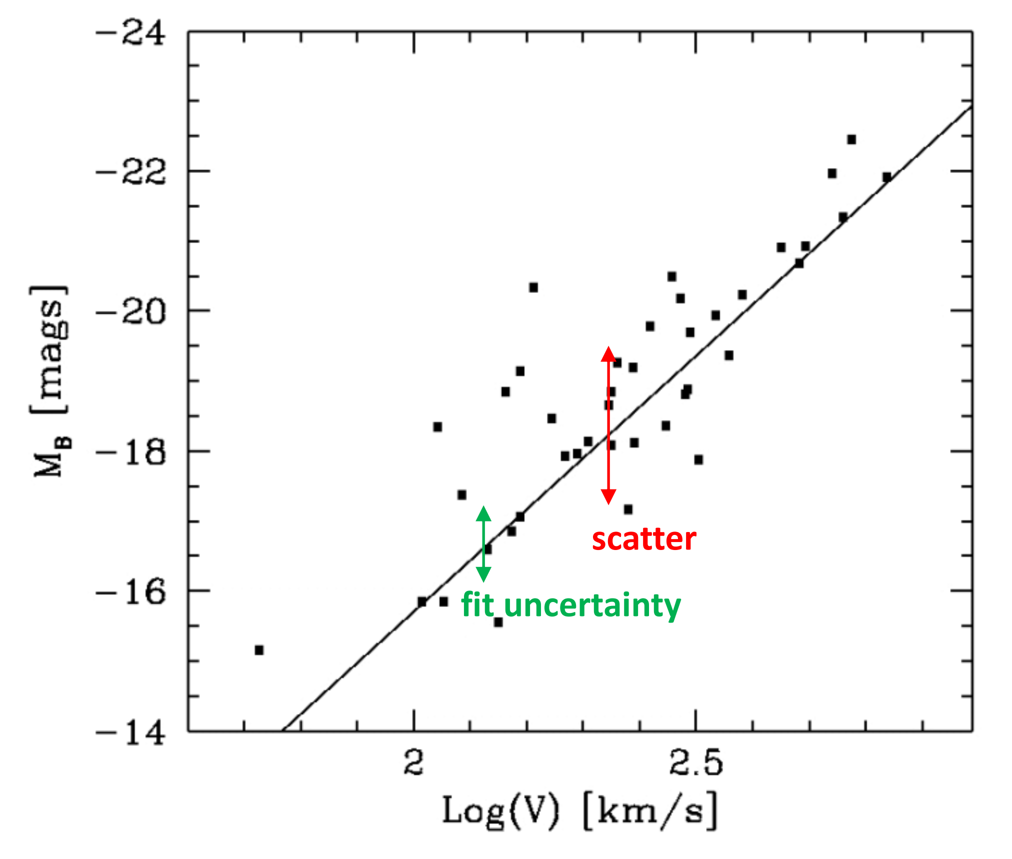
The uncertainties on the fit tell you how well-determined the fit parameters are.

The scatter of the fit tells you how well, on average, individual data points obey the relationship.

Example: Tully Fisher Relationship ⇒

Lower fit uncertainties mean that the TF relationship is better-determined.

Large scatter means any one galaxy may not perfectly obey TF.



Characterizing a linear (or linearized) relationship (least squares fitting, assuming Gaussian statistics):

```
# make a linear fit, and calculate uncertainty and scatter

good = <some criterion> # dont want to include bad data

coeff, cov = np.polyfit(x[good],y[good],1,cov=True)

coeff_err = np.sqrt(np.diag(cov))

print(' slope = {:.3f} +/- {:.3f}'.format(coeff[0],coeff_err[0]))

print('intercept = {:.3f} +/- {:.3f}'.format(coeff[1],coeff_err[1]))

polynomial=np.poly1d(coeff)

xfit=np.linspace(x.min(),x.max())

plt.plot(xfit,polynomial(xfit),color='green',lw=3)

print(' scatter = {:.3f}'.format(np.std(y[good]-polynomial(x[good]))))
```

Linearization

Sometimes you will need to fit a power law, or a sinusoid, or an exponential. These are non-linear models, but can be made linear.

Power Law: $\mathrm{y} = x^{\alpha}$, fit for α

Linearize it: $\log(y) = \log(x^{\alpha}) = \alpha \log(x)$, so fit a straight line to log(y) versus log(x), then the slope is alpha

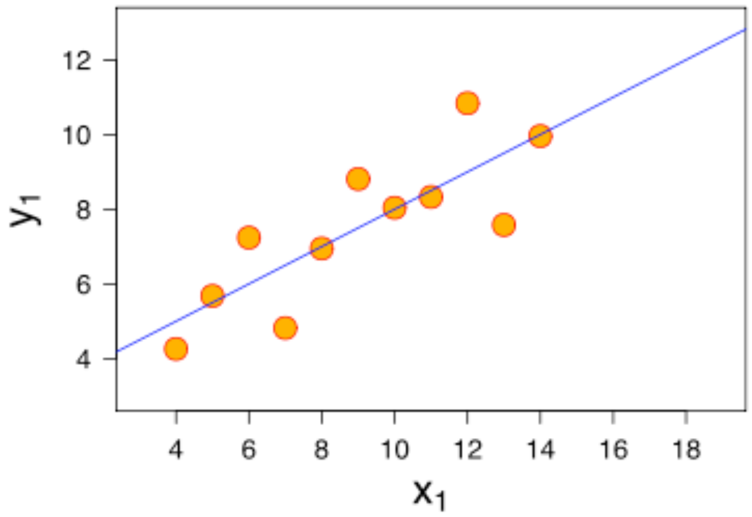
Sine function: $\mathrm{y} = A \sin x + B$, fit for A and B.

Linearize it: it is already linear if you fit a straight line to y vs sin(x) rather than y vs x.

Exponential: $\mathrm{y} = e^{-x/h}$, fit for h

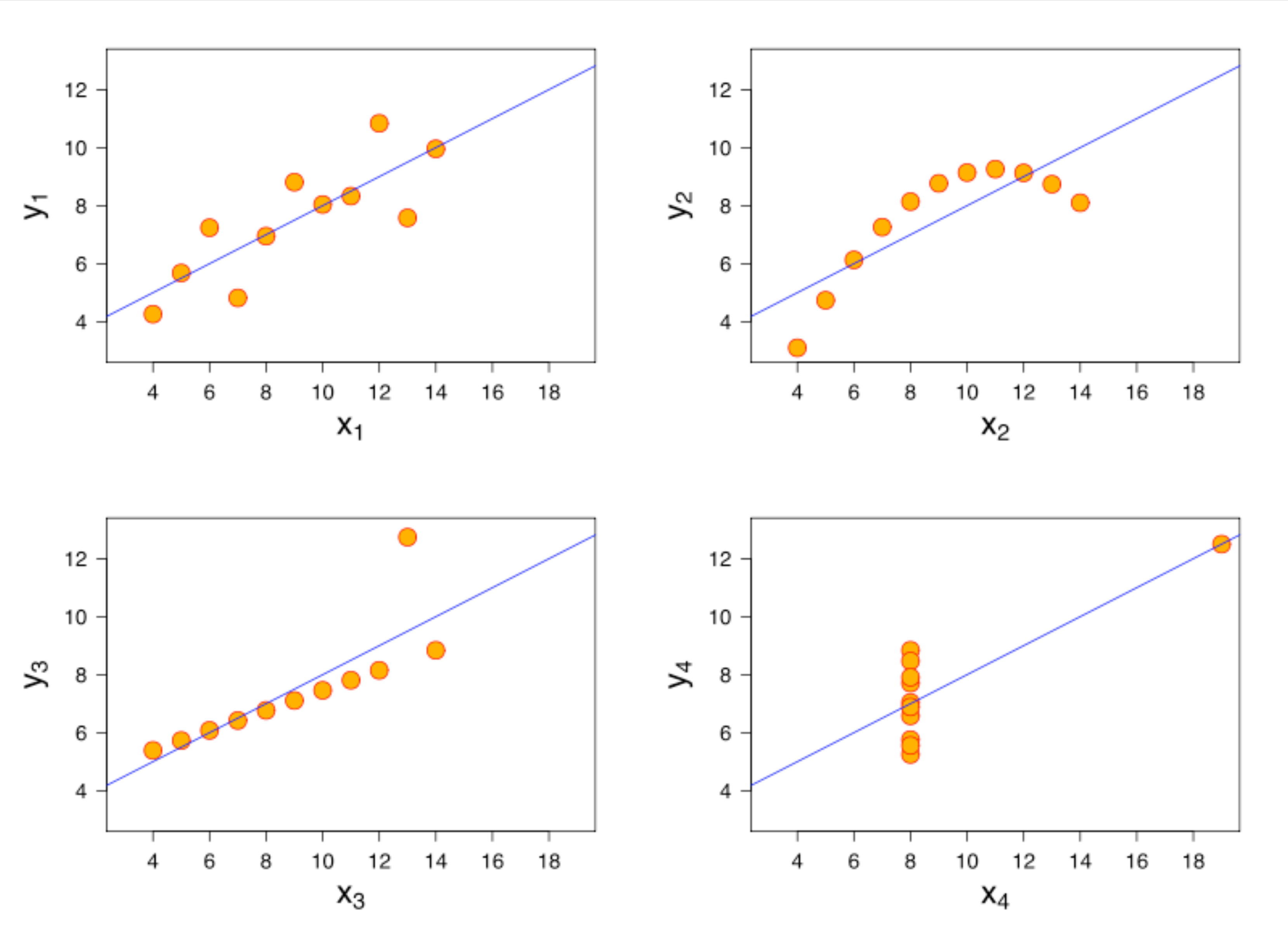
Linearize it: $\ln y = \ln\left(e^{-x/h}\right) = \frac{-1}{h} x$, so fit a straight line to ln(y) vs x and then h is -1/slope.

But be careful with fits...



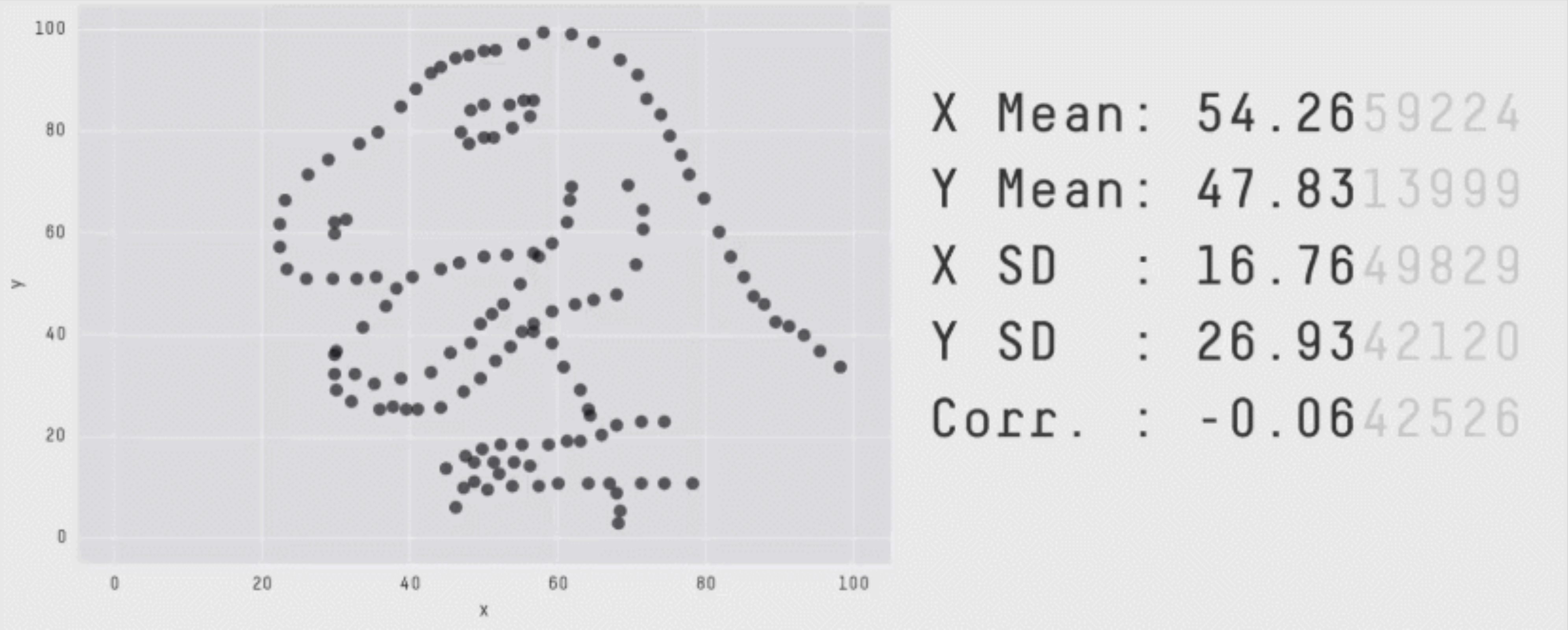
Anscombe's quartet: Fit y=mx+b and get the same r (correlation coefficient), m, b, σ_m, σ_b, σ_{RMS}

But be careful with fits...



Anscombe's quartet: Fit y=mx+b and get the same r (correlation coefficient), m, b, σ_m, σ_b, σ_{RMS}

Beware the datasaurus!



Moral of the story: ALWAYS PLOT YOUR DATA AND ALWAYS OVERPLOT YOUR FITS!