# Extracting information from a set of observations

# Last lecture

We talked about how to interpret an observation (data point)

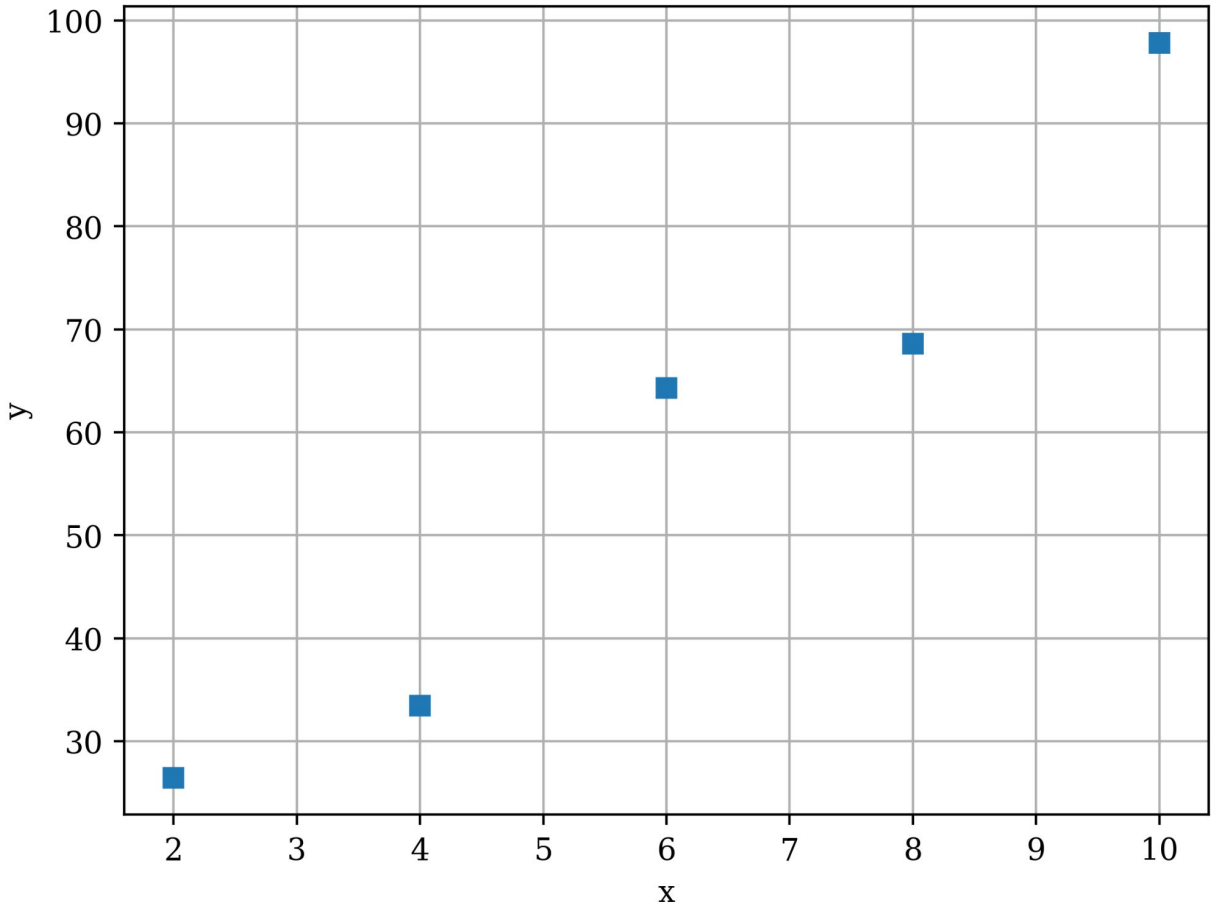
Today -

We will talk about how to extract information from a set of observations

Linear-ish looking data points

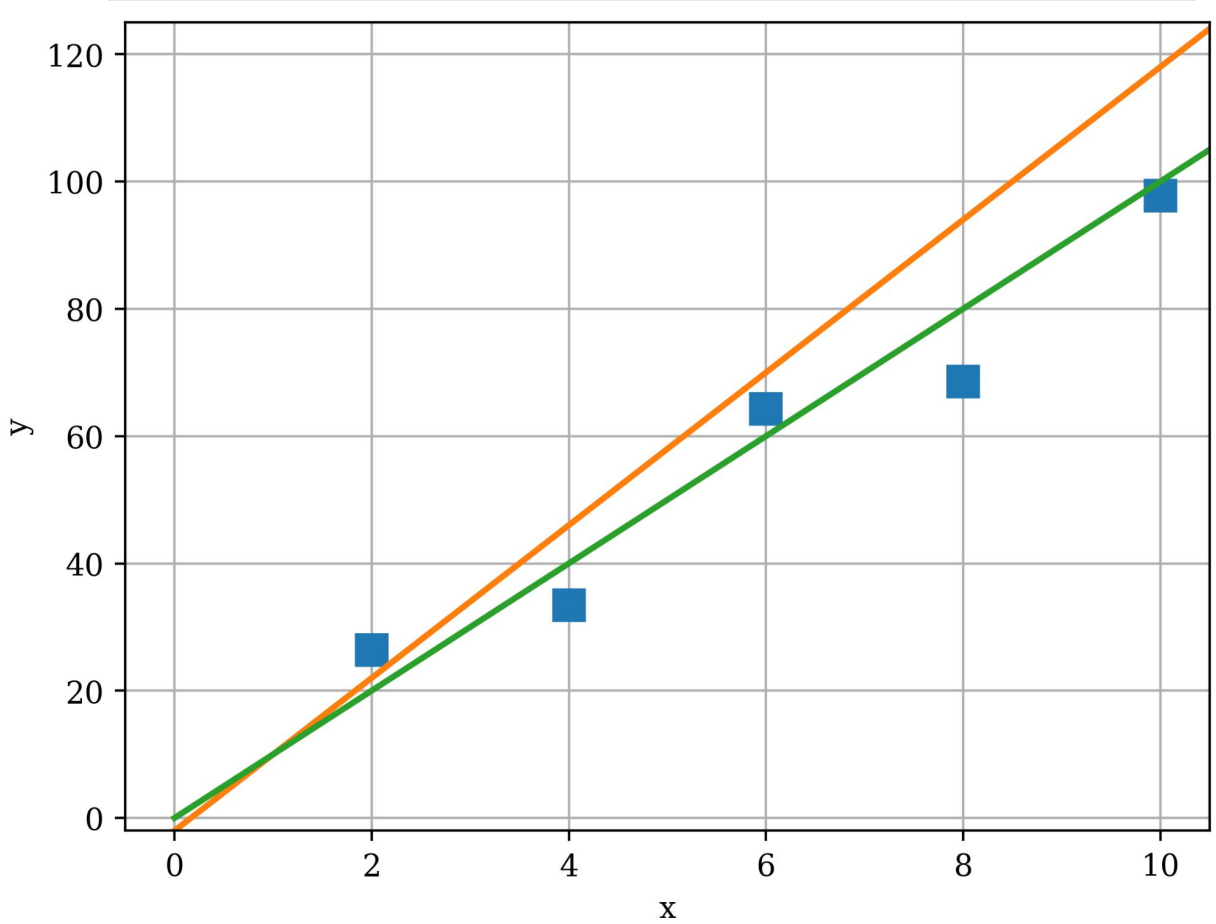
Want to find a linear trend line that describes it well

How can we do this?



How can we choose between 2 lines

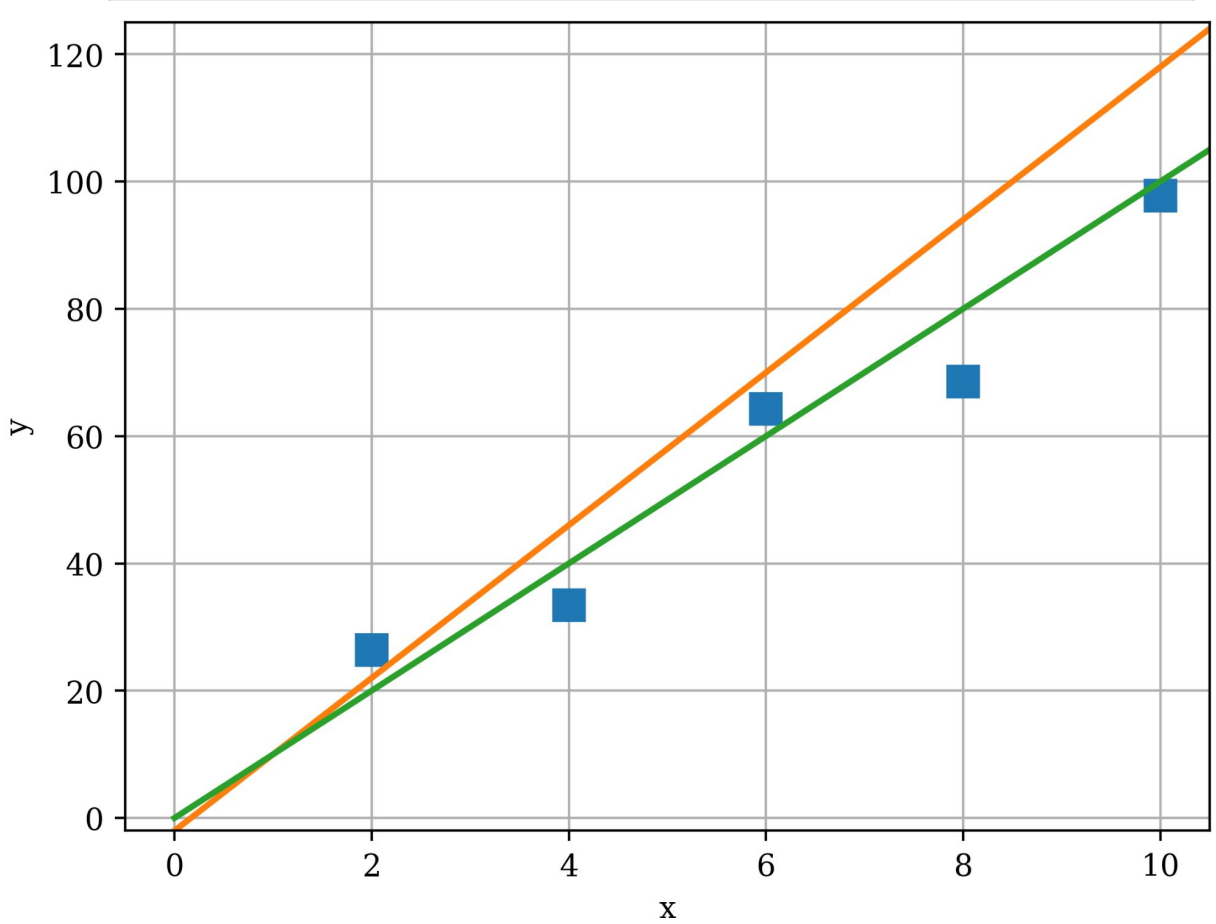
Which describes the data better?

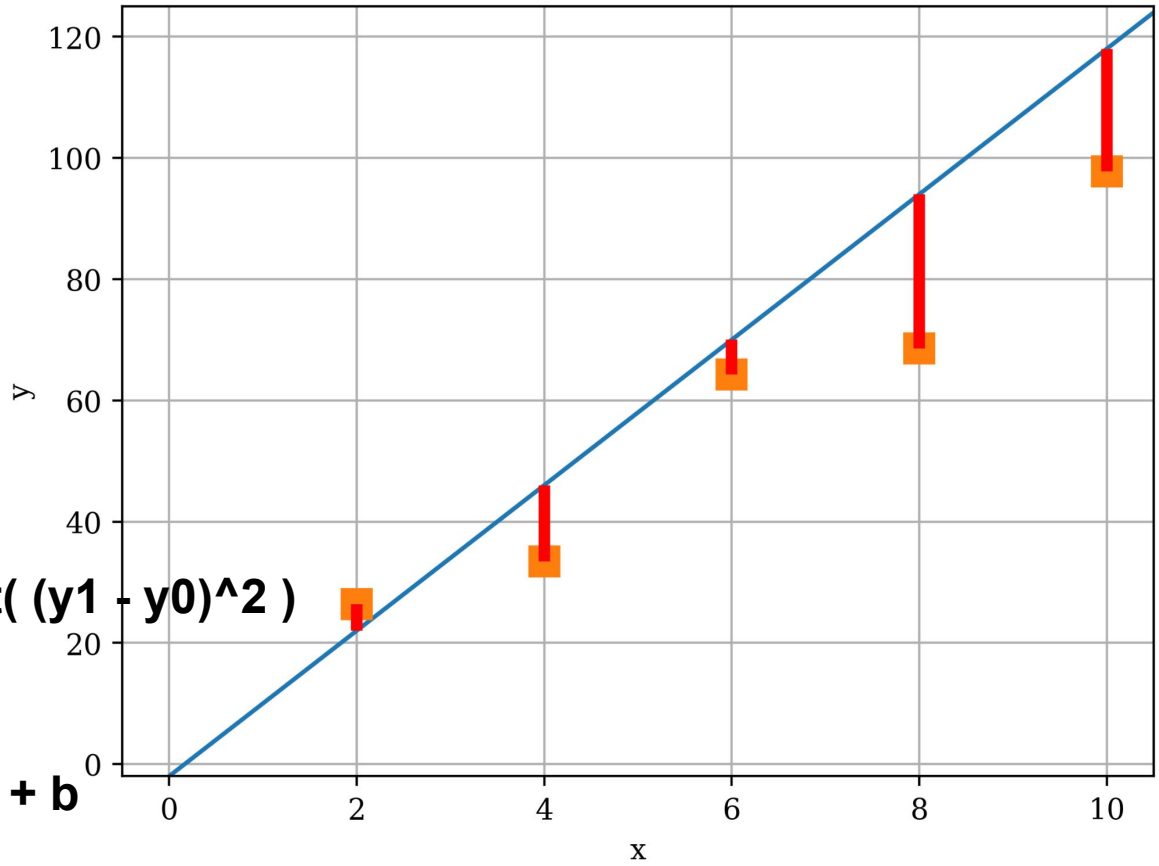


How can we choose between 2 lines

Which describes the data better?

How about the one closest to the data points?

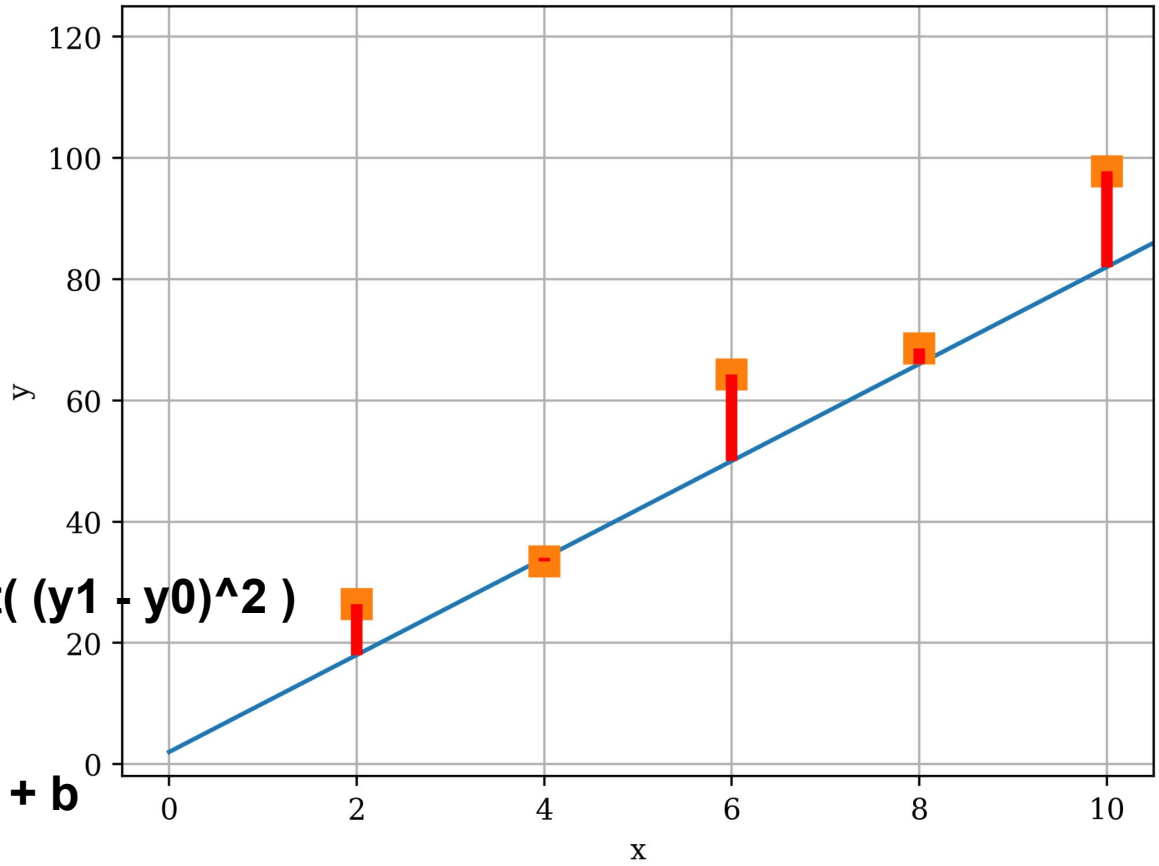





**distance = sqrt( (y1 - y0)^2 )**

**y0 = data**

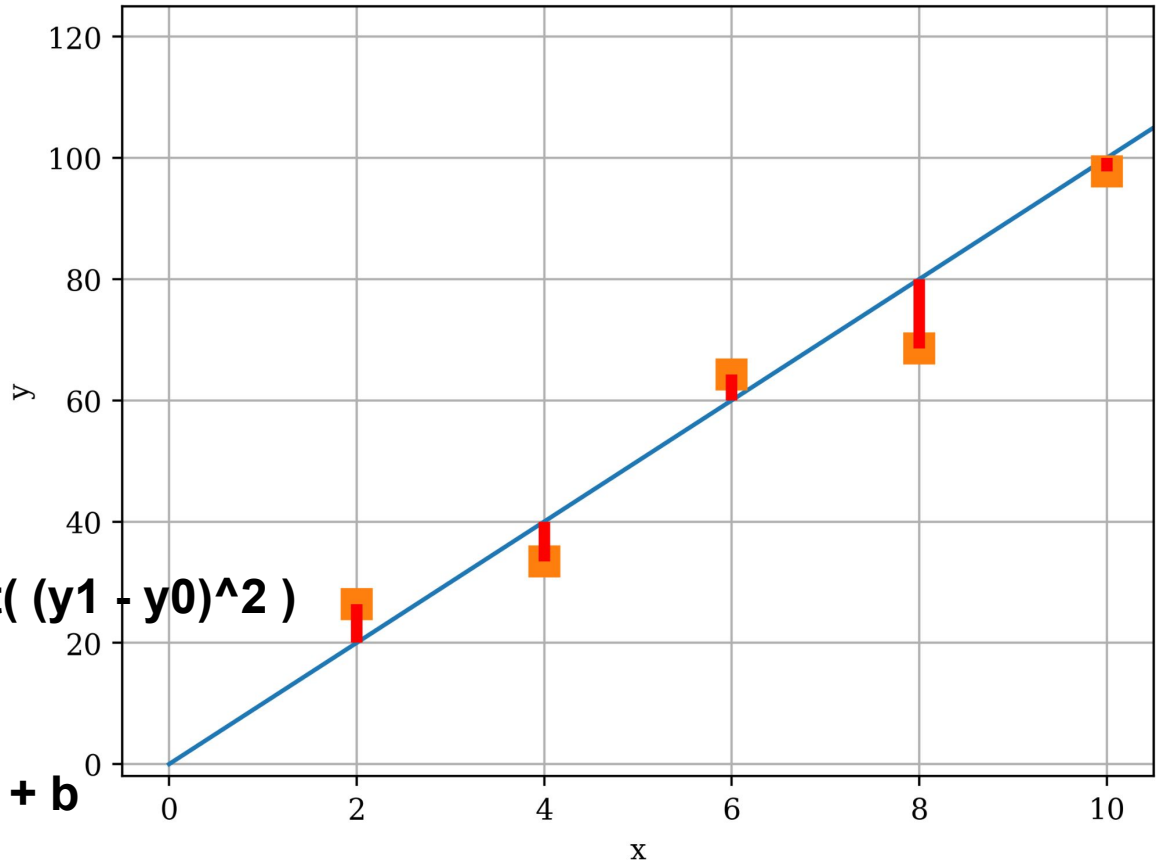
**y1 = line = m*x + b**




**distance = sqrt( (y1 - y0)^2 )**

**y0 = data**

**y1 = line = m*x + b**




**distance = sqrt( (y1 - y0)^2 )**

**y0 = data**

**y1 = line = m*x + b**

# Least Squares

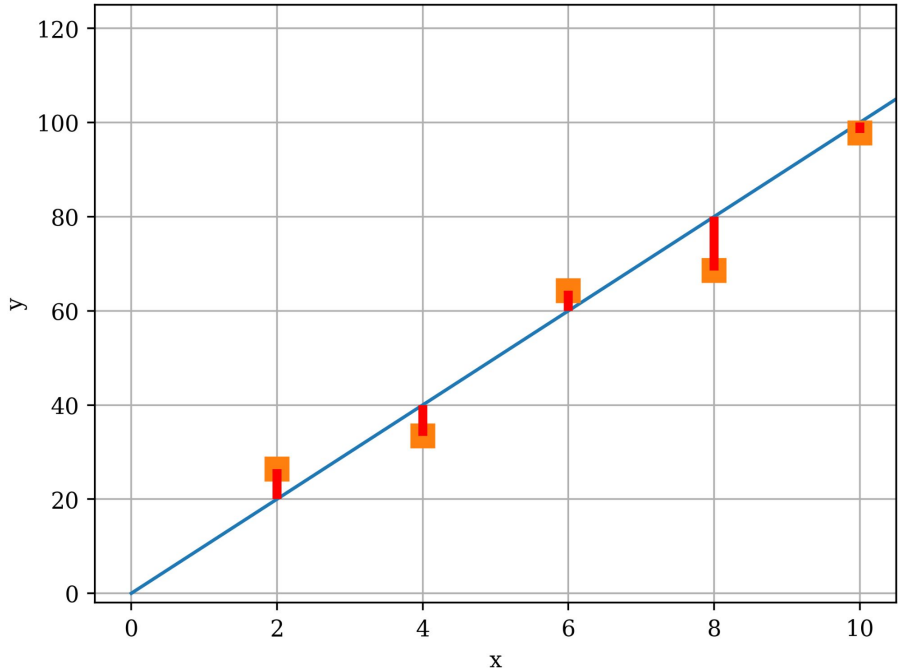
This is known as least squares fitting

a very common method

$S = \Sigma_i \, ( y(x_i) - y_i )^2$

Then find the form of y(x) that minimizes S

Here y(x) = m*x + b



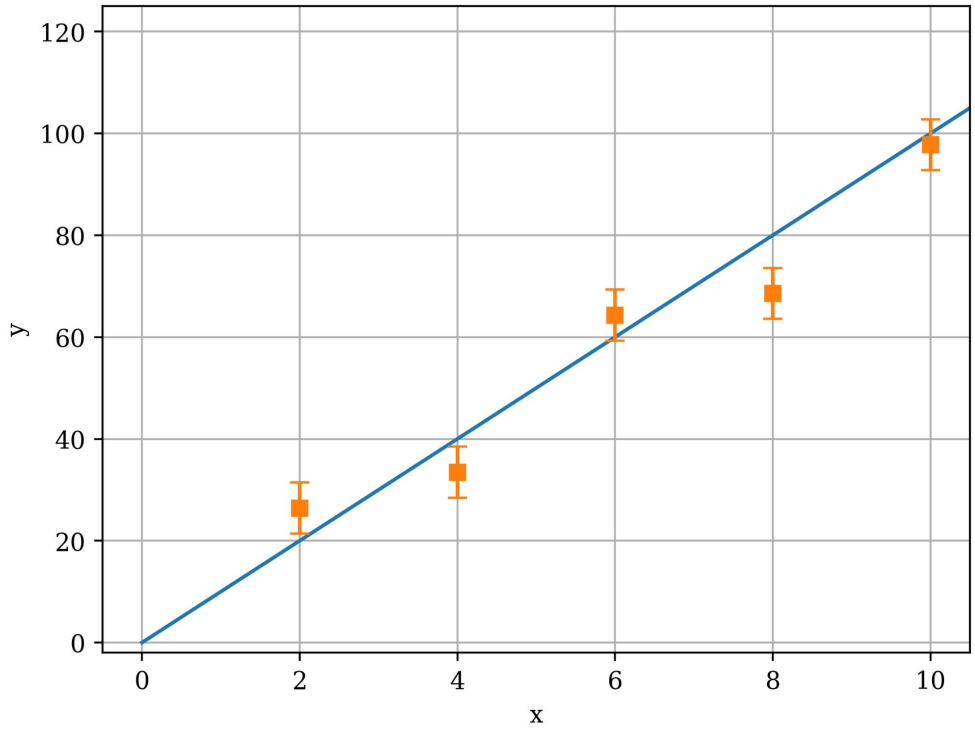
# Let’s make this realistic

These data points are observations, they should have some error to them

Let’s say they have a Gaussian error

Here’s 1 sigma error bars

How can we take into account the error doing least squares?



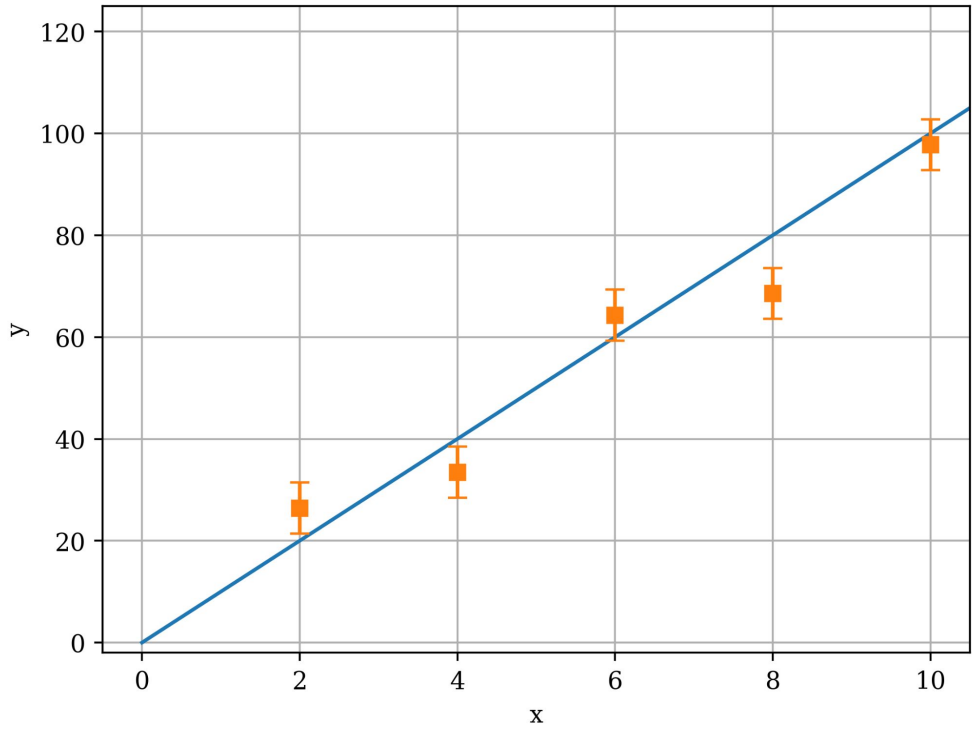
# Let’s make this realistic

How can we take into account the error doing least squares?

Let’s divide the distance by σ

$s_i = ( (y(x_i) - y_i ) / \sigma_i )^2$

It’s now how many σ’s is it from the line

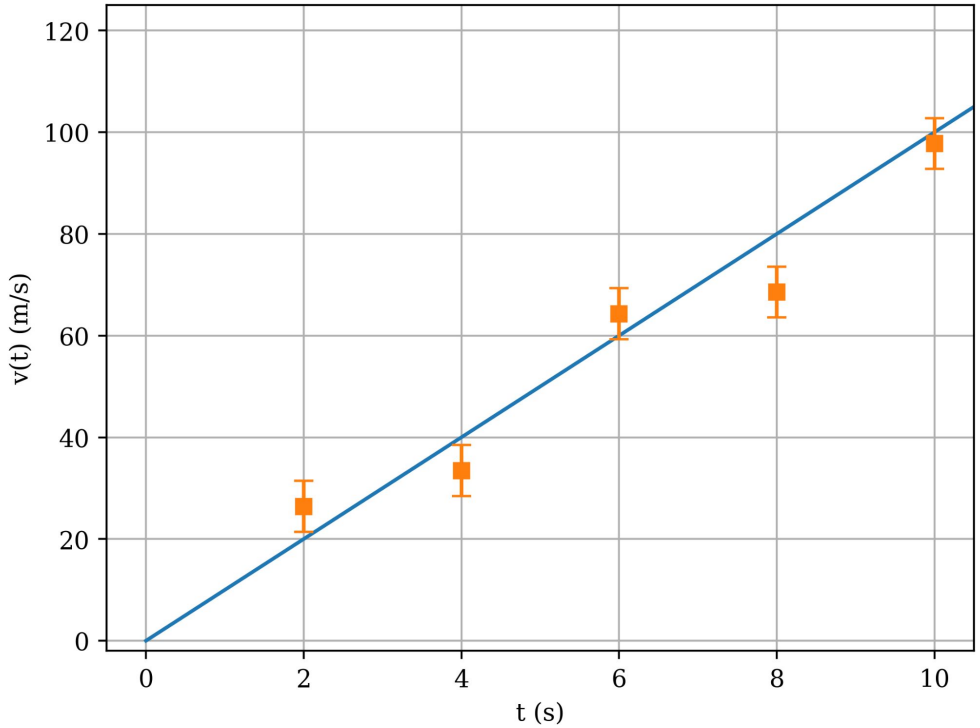


# Let’s make this realistic

More realism

Let’s say the y-axis is the speed of a car (dragster, this is pretty fast) and the x-axis is time

The speed is measured via a radar gun every 2 s, with a 1 sigma error of 5 m/s

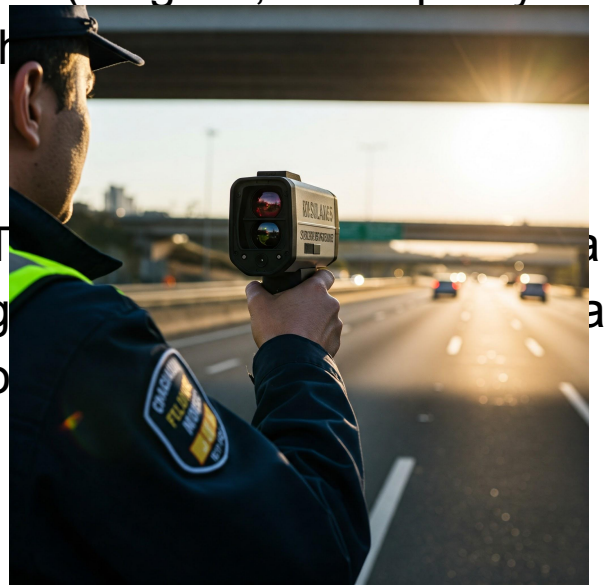


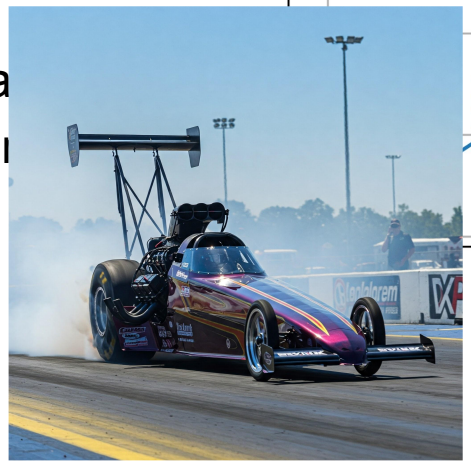
# Let’s make this realistic

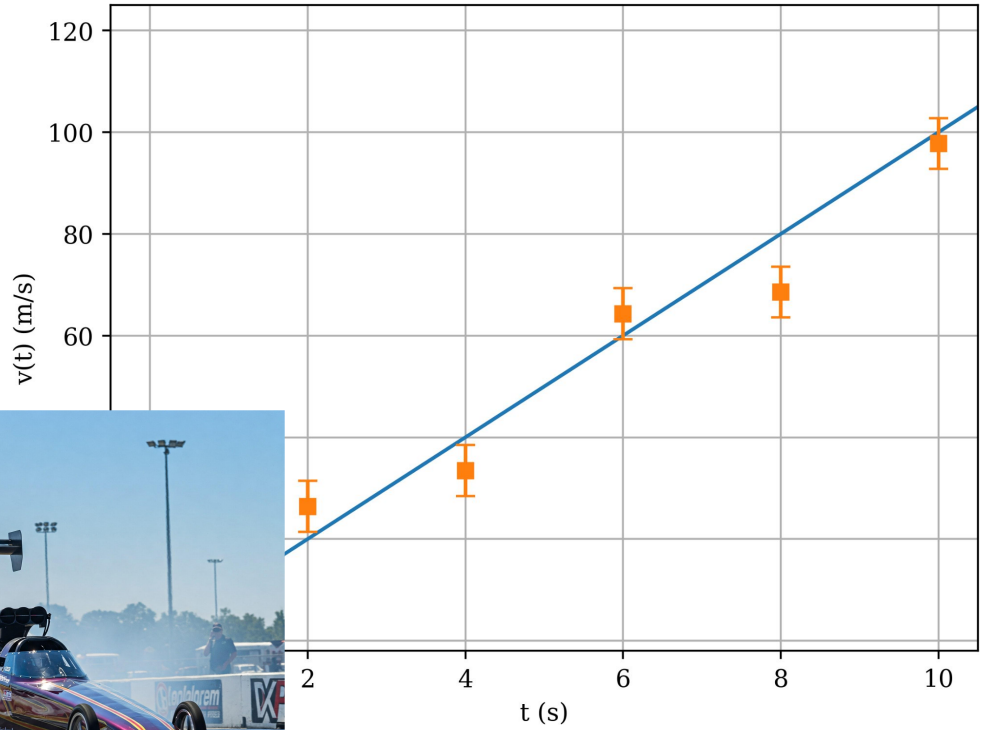
More realism

Let’s say the y-axis is the speed of a car (dragster, this is pretty fast) and th

T a ra
g a e
o





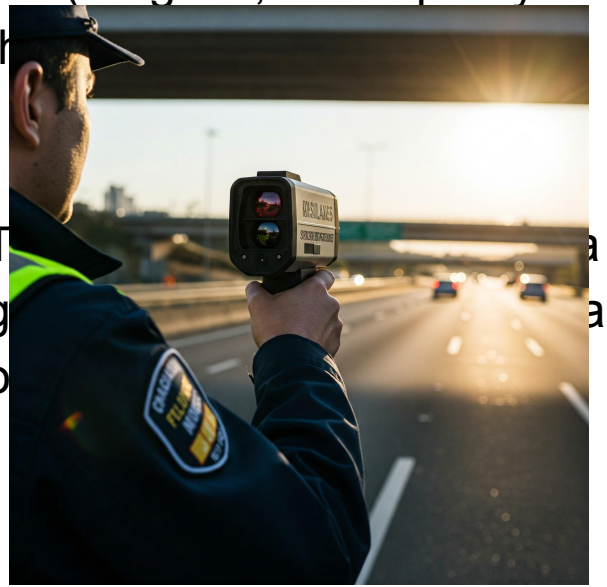


# Let’s make this realistic

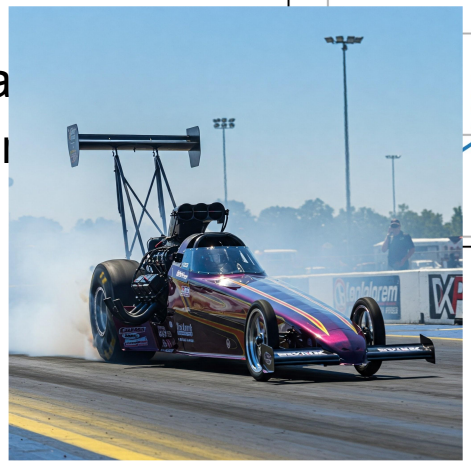
In this case, what does the slope represents?

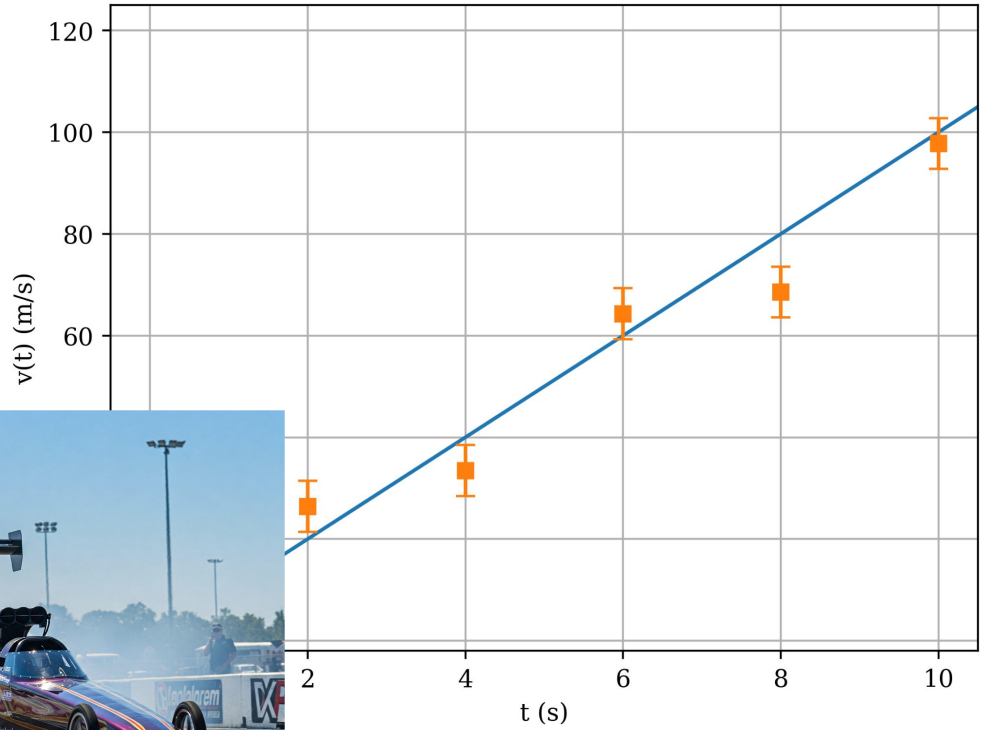
More realism

Let’s say the y-axis is the speed of a car (dragster, this is pretty fast) and th



T a ra
g a e
o





If the errors are Gaussian and

$y(x_i)$ gives us close to the “true” value

$(y(x_i) - y_i) / \sigma_i$ should follow a Standard Normal distribution (mu=0, σ=1)

$Z_i = (y(x_i) - y_i) / \sigma_i$

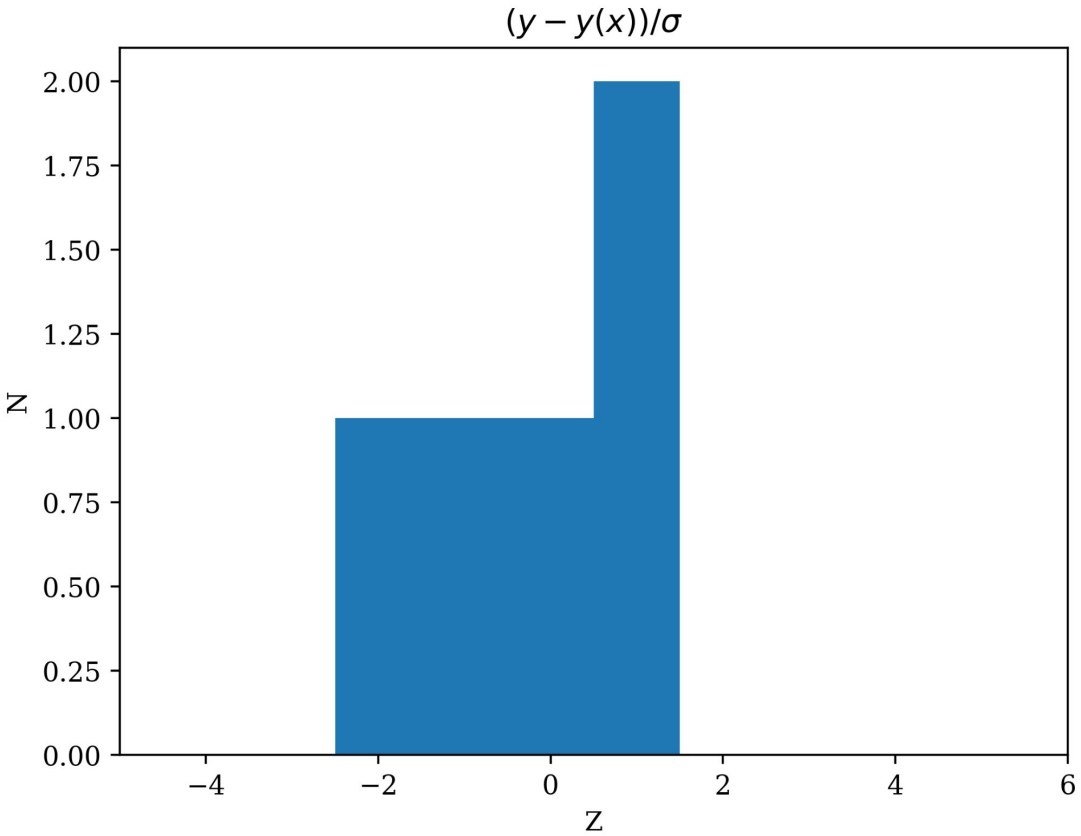
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} = \frac{1}{\sqrt{2\pi}} e^{-\frac{Z^2}{2}}$$

S = the sum of a bunch of normally distributed values squared

Let’s plot the Z values

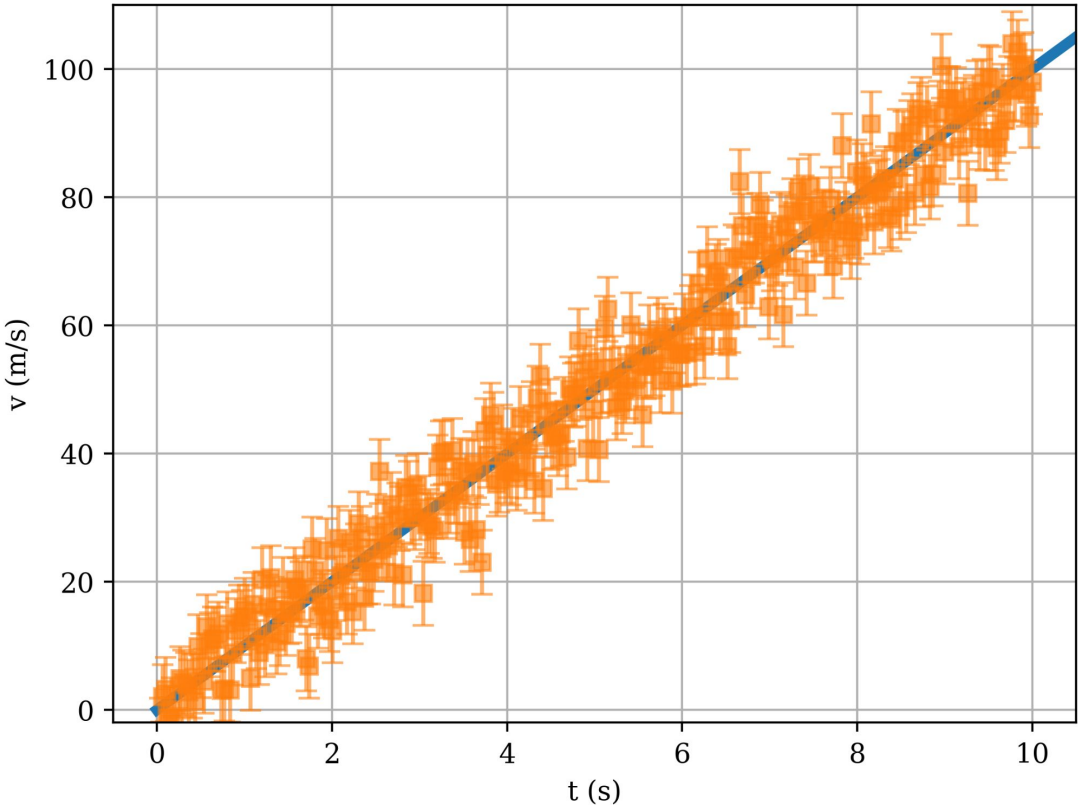
Clearly see a Gaussian,

right?



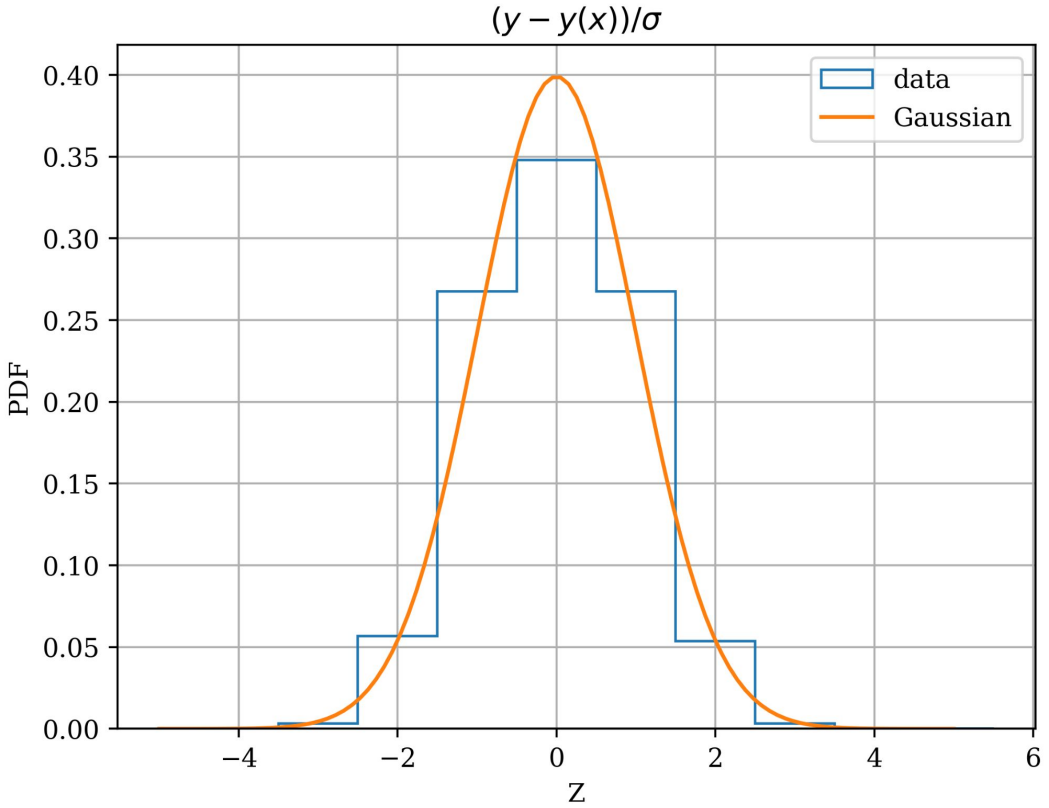
Let’s repeat this with more data points

Measure the speed 30 times a second



Let’s repeat this with more data points

Would be more exact with even more data, but actually looks like a standard normal now



So now, S or sum of squares is

S = the sum of a bunch of normally distributed values squared

What's special about that?

Well that sum happens to follow a specific probability distribution

The chi2 distribution $\chi^2$

The $\chi^2$ PDF depends on the number of values in the sum, k

The function is very complicated

The $\chi^2$ PDF depends on the number of values in the sum, k

The function is very complicated

$$\frac{1}{2^{k/2}\Gamma(k/2)}\, x^{k/2-1} e^{-x/2}$$

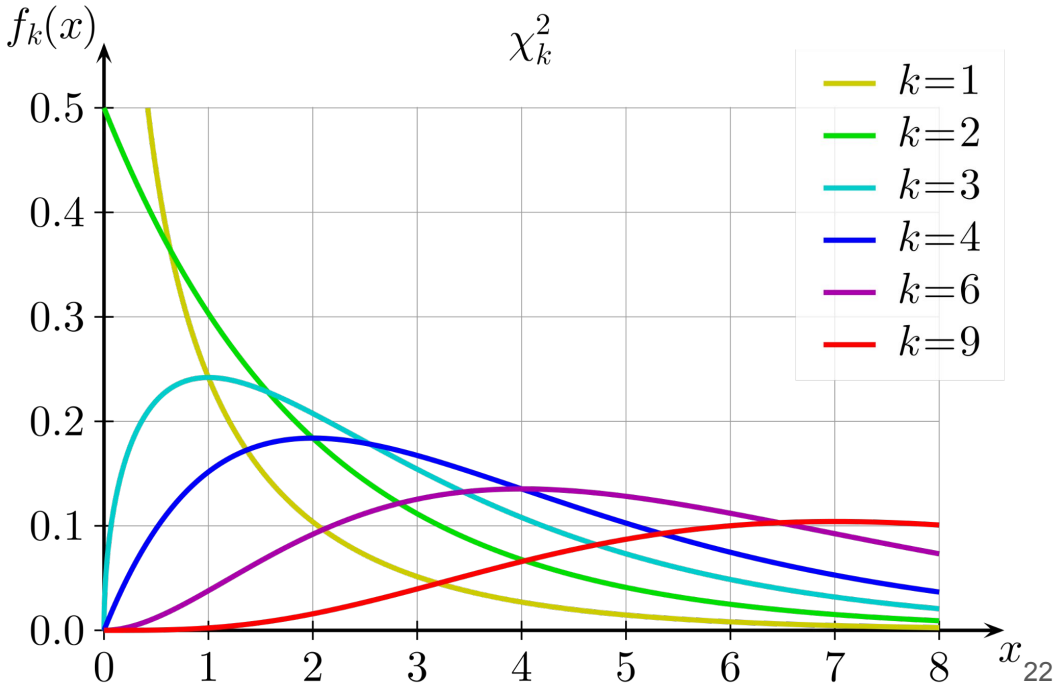
- You will all just use scipy.stats.chi2.pdf()

The $\chi^2$ PDF depends on the number of values in the sum, k

The function is very complicated

$$\frac{1}{2^{k/2}\Gamma(k/2)}\, x^{k/2-1} e^{-x/2}$$

- You will all just use scipy.stats.chi2.pdf()



Now what does that mean S should follow a certain distribution?

S itself is an observation

Every time we rerun this experiment we will get a different S

Those values of S will follow a $\chi^2$ distribution

Ok so let's have the dragster do this 10,000 times

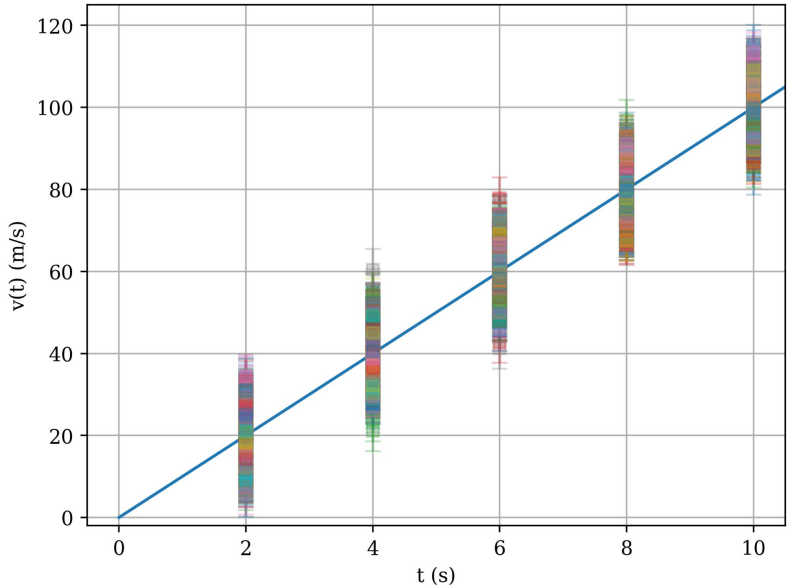
Accelerating at exactly 10 m/s2 each time

Each time we take our 5 measurements

Ok so let's have the dragster do this 10,000 times

Accelerating at exactly 10 m/s2 each time

Each time we take our 5 measurements



Ok so let's have the dragster do this 10,000 times

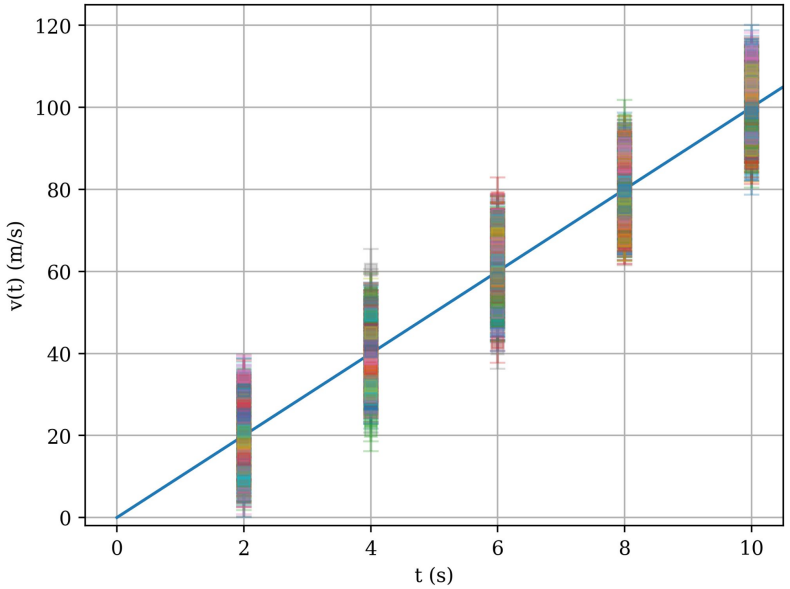
Accelerating at exactly 10 m/s2 each time

Each time we take our 5 measurements

And calculate S

$S = \Sigma_i \left( (v(t_i) - v_i ) / \sigma_i \right)^2$

$v(t_i) = a*t_i$ , where a = 10 m/s2



Ok so let's have the dragster do this 10,000 times

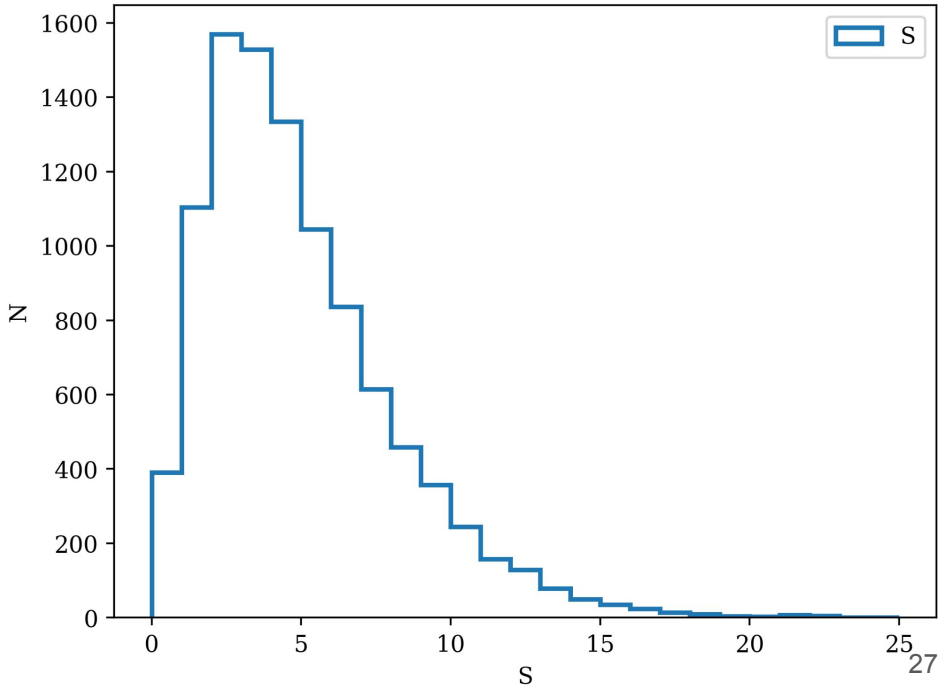
Accelerating at exactly 10 m/s2 each time

Each time we take our 5 measurements

And calculate S

$S = \Sigma_i \left( (v(t_i) - v_i) / \sigma_i \right)^2$

$v(t_i) = a*t_i$ , where a = 10 m/s2



Ok so let's have the dragster do this 10,000 times

Accelerating at exactly 10 m/s2 each time

Each time we take our 5 measurements

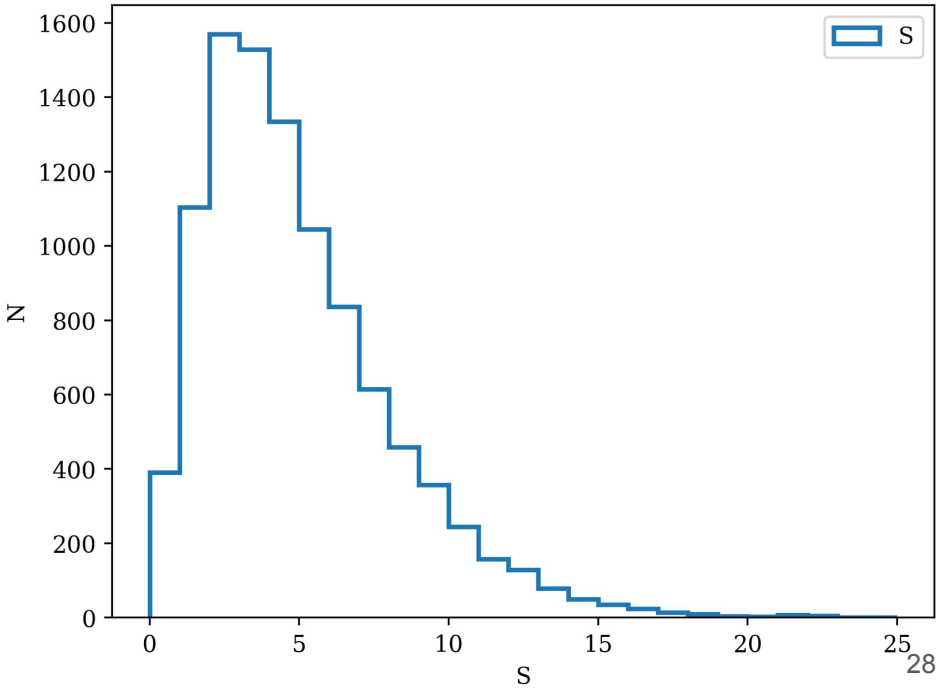
And calculate S

$S = \Sigma_i \left( (v(t_i) - v_i) / \sigma_i \right)^2$

$v(t_i) = a*t_i$ , where a = 10 m/s2

Now let's see how it compares to a $\chi^2$

What should k be?



Ok so let's have the dragster do this 10,000 times

Accelerating at exactly 10 m/s2 each time

Each time we take our 5 measurements

And calculate S

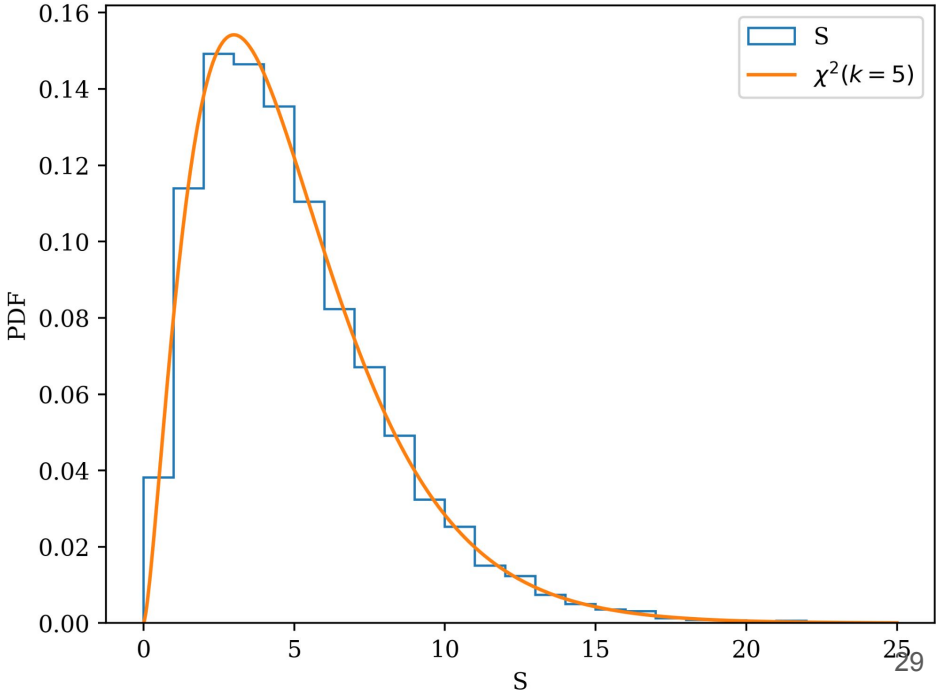
$S = \Sigma_i \left( (v(t_i) - v_i) / \sigma_i \right)^2$

$v(t_i) = a*t_i$ , where a = 10 m/s2

Now let's see how it compares to a $\chi^2$

What should k be?

5 numbers in sum, k = 5



# Goodness of fit

S (or sometimes it's just called $\chi^2$ ), not only tells us what the best parameters are to describe our data

It also tells us how well the function describes our data

# Goodness of fit

S (or sometimes it's just called $\chi^2$ ), not only tells us what the best parameters are to describe our data

It also tells us how well the function describes our data

In the previous example we used the "true" value of a (the slope)

This of course describes the data well

# Goodness of fit

S (or sometimes it's just called $\chi^2$ ), not only tells us what the best parameters are to describe our data

It also tells us how well the function describes our data

In the previous example we used the "true" value of a (the slope)

This of course describes the data well

So, all of our experiments had a $\chi^2$ value that fell within expectations of the $\chi^2$ distribution

# Goodness of fit

S (or sometimes it's just called $\chi^2$ ), not only tells us what the best parameters are to describe our data

It also tells us how well the function describes our data

In the previous example we used the "true" value of a (the slope)

This of course describes the data well

So, all of our experiments had a $\chi^2$ value that fell within expectations of the $\chi^2$ distribution

The smaller the $\chi^2$ value the closer the function is to the data, but how big is too big?

# Goodness of fit

$P(\chi^2 > 11.07, k=5) = \int_{11.07}^{\infty} p(\chi^2, k=5)\, d(\chi^2) = 0.95$

This integral is also known as the survival function stats.chi2.sf(11.07)

This means that 95% of the time $\chi^2$ should be < 11.07

So if $\chi^2$ is above that there's a chance it does not fit well

# Goodness of fit

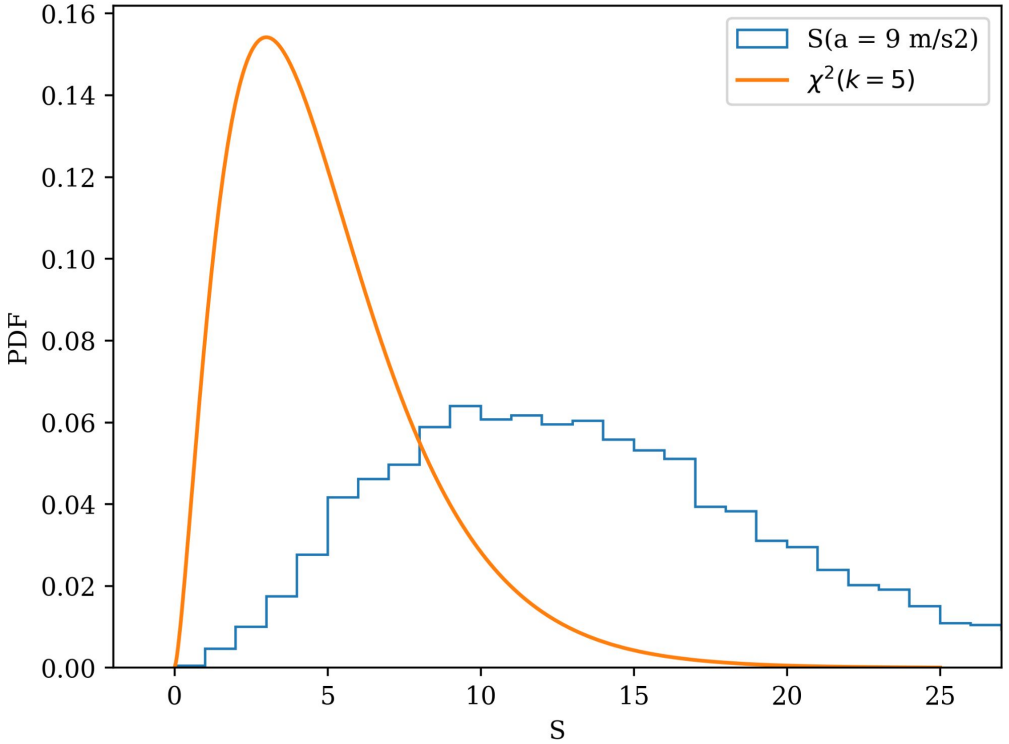
Let’s repeat the experiments and set $a = 9\ m/s^2$ when calculating $\square^2$

# Goodness of fit

Let's repeat the experiments and set a = 9 m/s$^2$ when calculating $\chi^2$

Distributions do not match any more

61.5% of the time $\chi^2$ was > 11.07



# Goodness of fit

And if we instead set a = 8 $m/s^2$

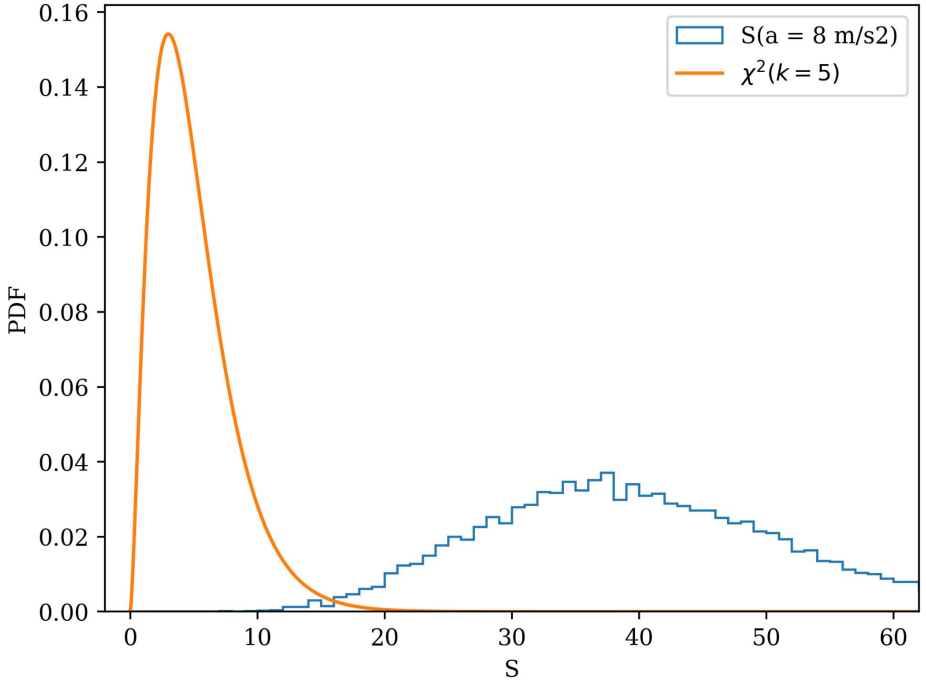
when calculating $\square^2$

# Goodness of fit

And if we instead set a = 8 m/s$^2$ when calculating $\chi^2$

Distributions do not match at all

99.9% of the time $\chi^2$ was > 11.07

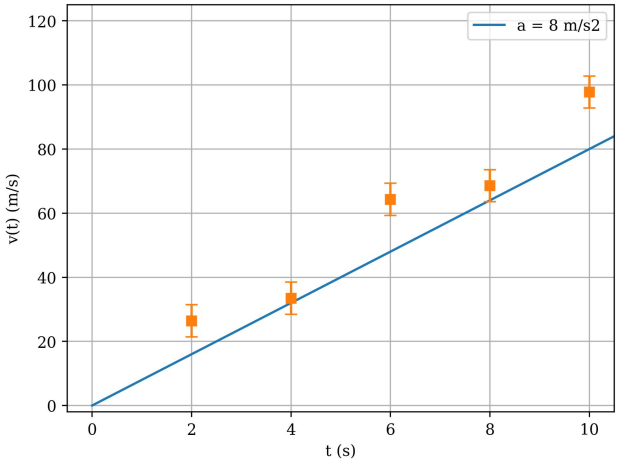


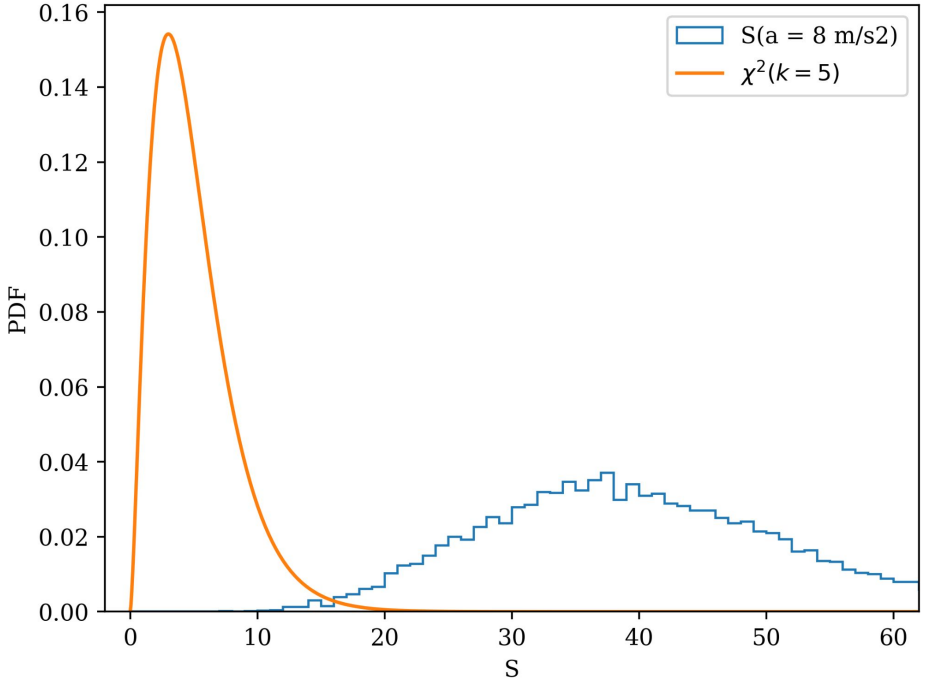
# Goodness of fit

And if we instead set a = 8 m/s$^2$ when calculating $\chi^2$

Distributions do not match at all

99.9% of the time $\chi^2$ was > 11.07



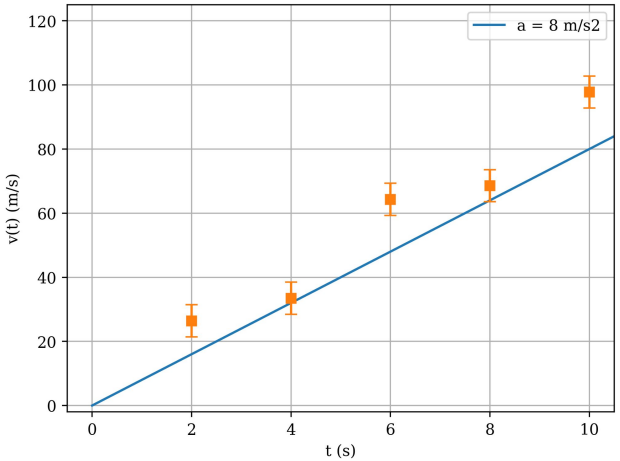



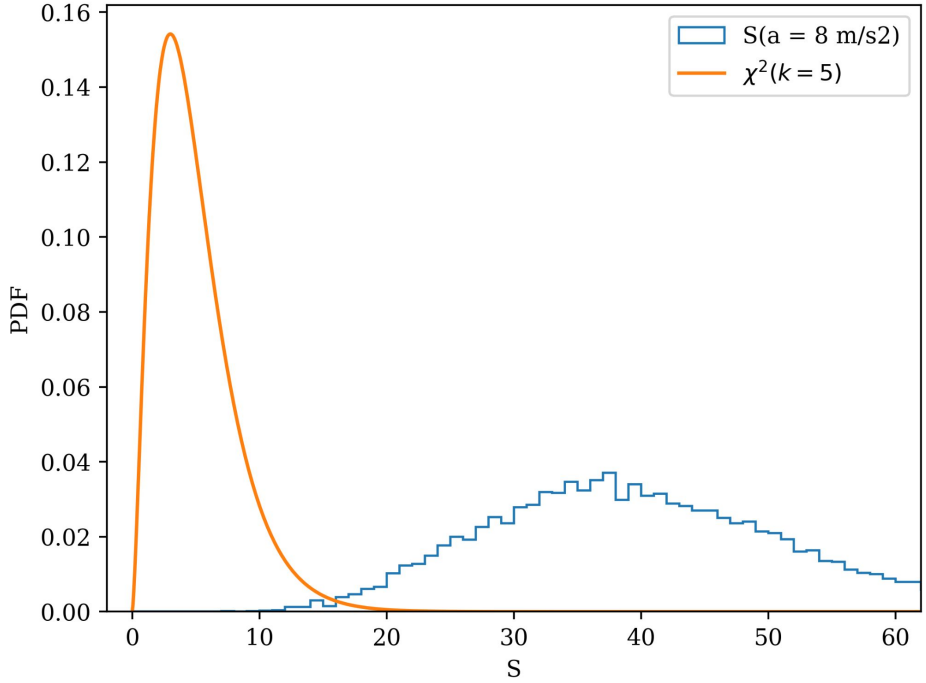
# Goodness of fit

And if we instead set a = 8 m/s$^2$ when calculating $\chi^2$

Distributions do not match at all

99.9% of the time $\chi^2$ was > 11.07




**We'll go more in depth on this next class**

# How to actually fit to the data

So far we haven't actually "fit" the data, we've only assumed some values of the slope

We started with a method called "least squares",

The optimal set of parameters will minimize S, or $\square^2$

# How to actually fit to the data

So far we haven't actually "fit" the data, we've only assumed some values of the slope

We started with a method called "least squares",

The optimal set of parameters will minimize S, or $\chi^2$

How do we minimize a function?

# How to actually fit to the data

So far we haven't actually "fit" the data, we've only assumed some values of the slope

We started with a method called "least squares",

The optimal set of parameters will minimize S, or $\chi^2$

How do we minimize a function?

- Calculus
- Brute force
- A little bit of both

# How to actually fit to the data

So far we haven't actually "fit" the data, we've only assumed some values of the slope

We started with a method called "least squares",

The optimal set of parameters will minimize S, or $\chi^2$

How do we minimize a function?

- Calculus
- Brute force
- A little bit of both - numerical minimization, we'll come back to this

# Calculus!

$$S = \sum_i \left( y_i - \left( m \cdot x_i \right) \right)^2$$

Assume b=0

$$\frac{\partial S}{\partial m} = 0 = -2 \sum_i \left( y_i - \left( m \cdot x_i \right) \right) x_i$$

$$0 = \sum_i \left( y_i x_i \right) - m \sum_i \left( x_i^2 \right)$$

$$m = \frac{\sum_i \left( y_i x_i \right)}{\sum_i \left( x_i^2 \right)}$$

An analytical solution like this really only exists for a linear function

# Calculus!

$$S = \sum_i \left( y_i - \left( m \cdot x_i \right) \right)^2$$

Assume b=0

$$\frac{\partial S}{\partial m} = 0 = -2 \sum_i \left( y_i - \left( m \cdot x_i \right) \right) x_i$$

$$0 = \sum_i \left( y_i x_i \right) - m \sum_i \left( x_i^2 \right)$$

$$m = \frac{\sum_i \left( y_i x_i \right)}{\sum_i \left( x_i^2 \right)}$$

# Calculus!

$$S = \sum_i \left( \frac{y_i}{\sigma_i} - \frac{\left( m \cdot x_i \right)}{\sigma_i} \right)^2$$

$$m = \frac{\sum_i \left( y_i x_i / \sigma_i^2 \right)}{\sum_i \left( x_i / \sigma_i \right)^2}$$

With sigma included

$$m = \sum y_i x_i / \sum x_i^2$$

$m = \sum y_i x_i / \sum x_i^2$

m = (26.43x2 + 33.47x4 + 64.31x6 + 68.59x8 + 97.80x10) / (2^2 + 4^2 + 6^2 + 8^2 + 10^2)

$m = \sum y_i x_i / \sum x_i^2$

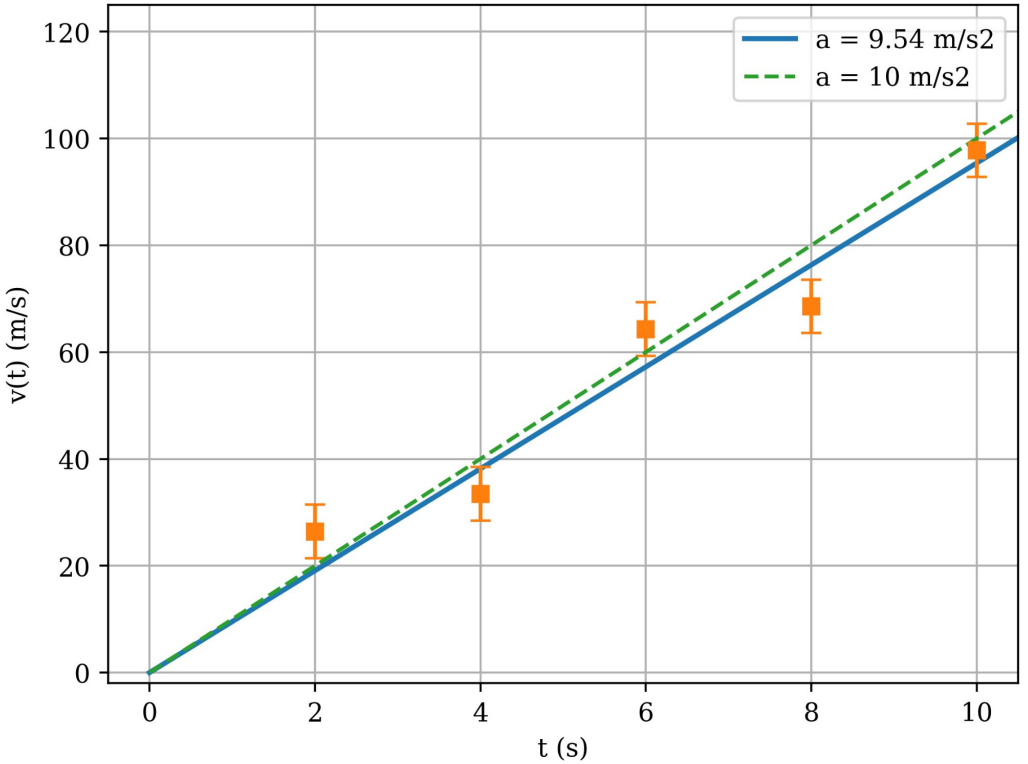
m = (26.43x2 + 33.47x4 + 64.31x6 + 68.59x8 + 97.80x10) / (2^2 + 4^2 + 6^2 + 8^2 + 10^2)

m = 9.54

$m = \sum y_i x_i / \sum x_i^2$

m = (26.43x2 + 33.47x4 + 64.31x6 + 68.59x8 + 97.80x10) / (2^2 + 4^2 + 6^2 + 8^2 + 10^2)

m = 9.54



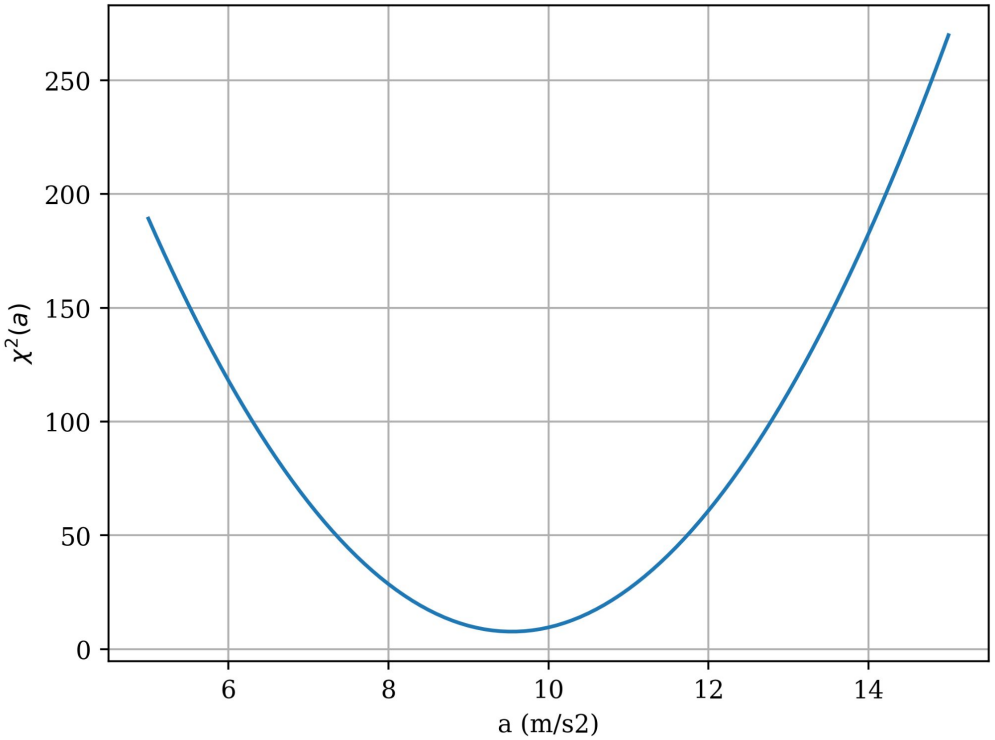
# Is this actually a minimum? What does S(m) look like?

# Is this actually a minimum? What does S(m) look like?

Yes, it is a minimum

S(m) is

-

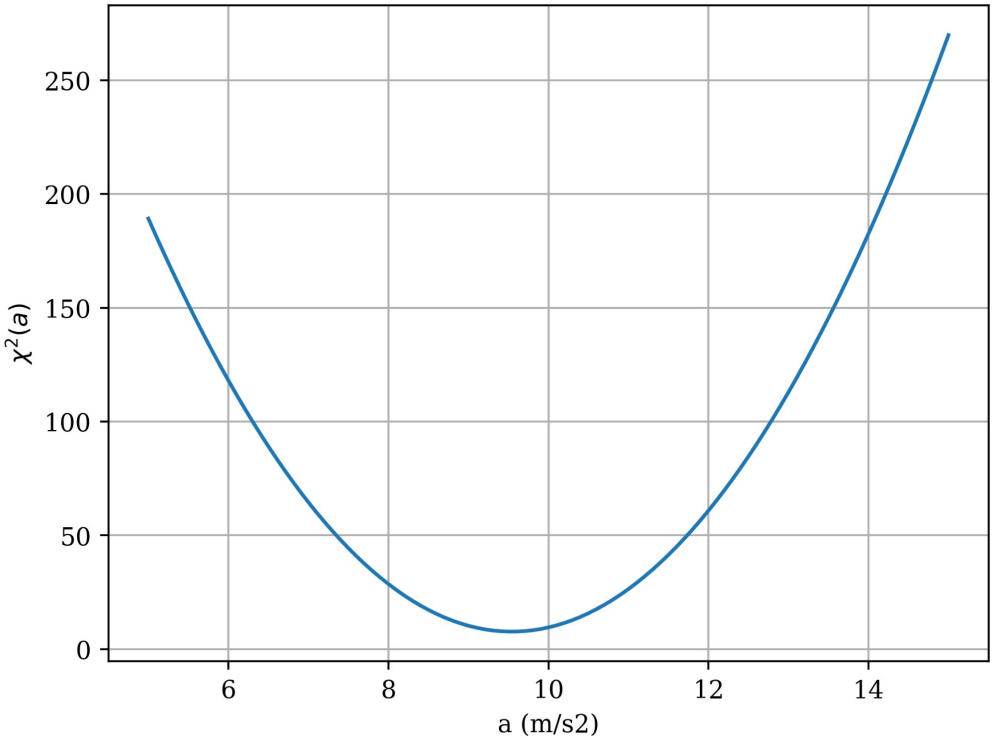


# Is this actually a minimum? What does S(m) look like?

Yes, it is a minimum

S(m) is

- parabolic
- has a clear single minimum
- continuous



# Brute Force!

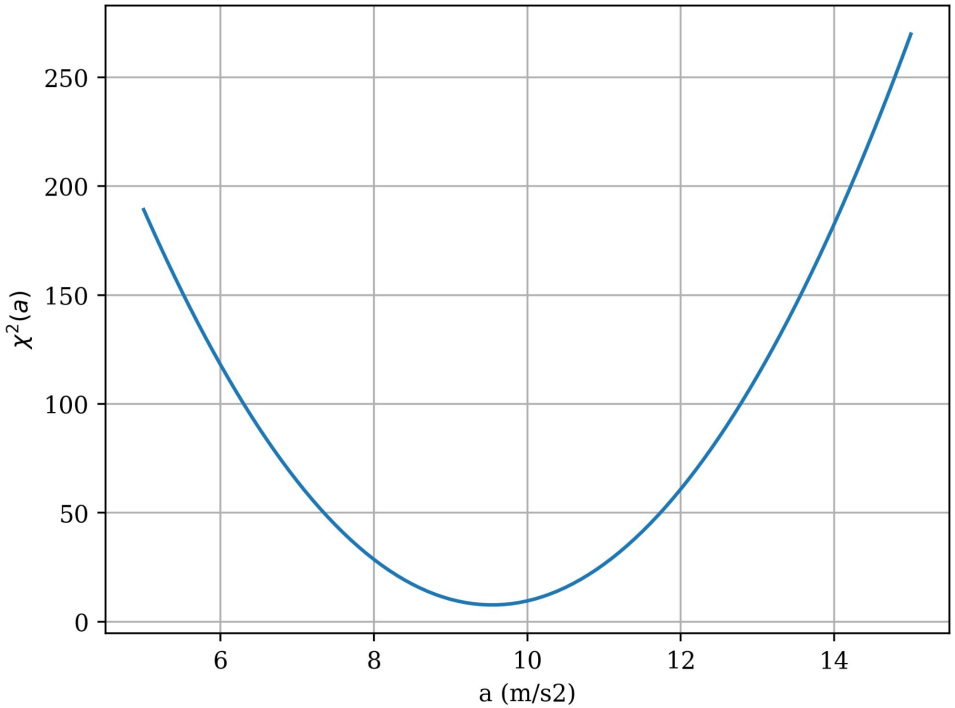
The brute force method works just like this

```
a_arr = np.linspace(0, 100, big_integer)

for i in range(big_integer):

        Chi2_arr[i] = np.sum(((y - a_arr[i]*t)/sigma)**2)

a_best = a_arr[np.argmin(Chi2_arr)]
```



# Brute Force!

Obvious downsides -

Takes a lot more compute power

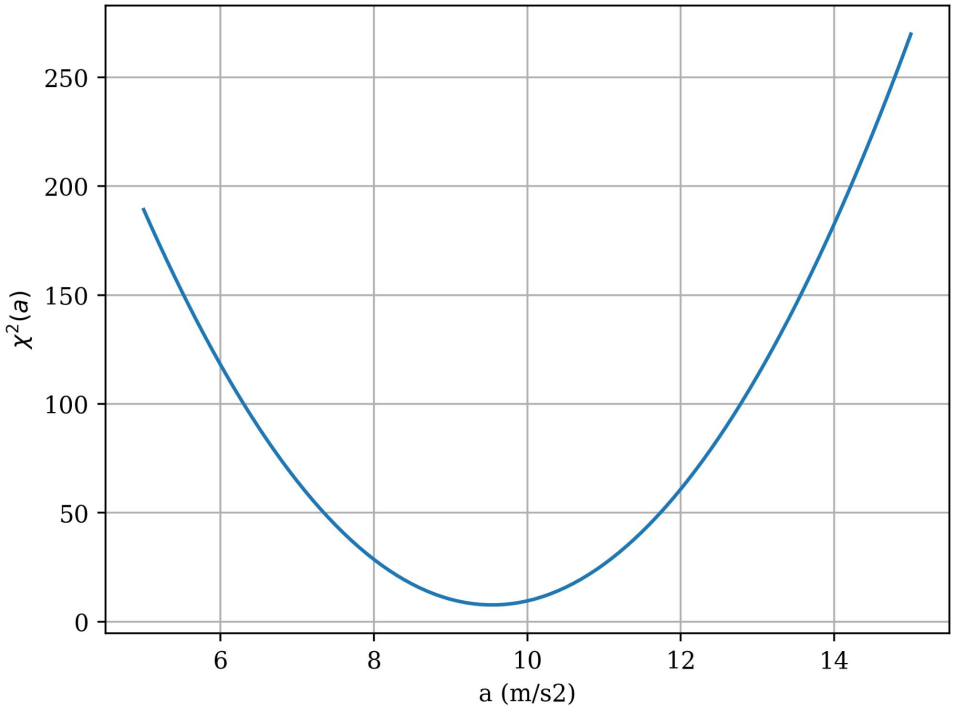
Not as accurate

- only as accurate as step size
- Need to know bounds to start and end at

Pros -

Very simple to do!

There doesn't need to be an analytical solution



# Next Time!

- Here we found the “best-fit” parameter for a model using data
- Next time we will find our error PDF for that parameter
- We will also assess whether a model, even with its “best-fit” parameters describes the data well