SBD '25-'26

Very gentle intro to **Linear Regression**

Dr. Niccolò Salvini

Section 1

Fitting a Line to data

Fitting a line to data

1 **minimize SSR**

2 **take derivative to find Least Squares**

Principles of linear regression

3 **R2**

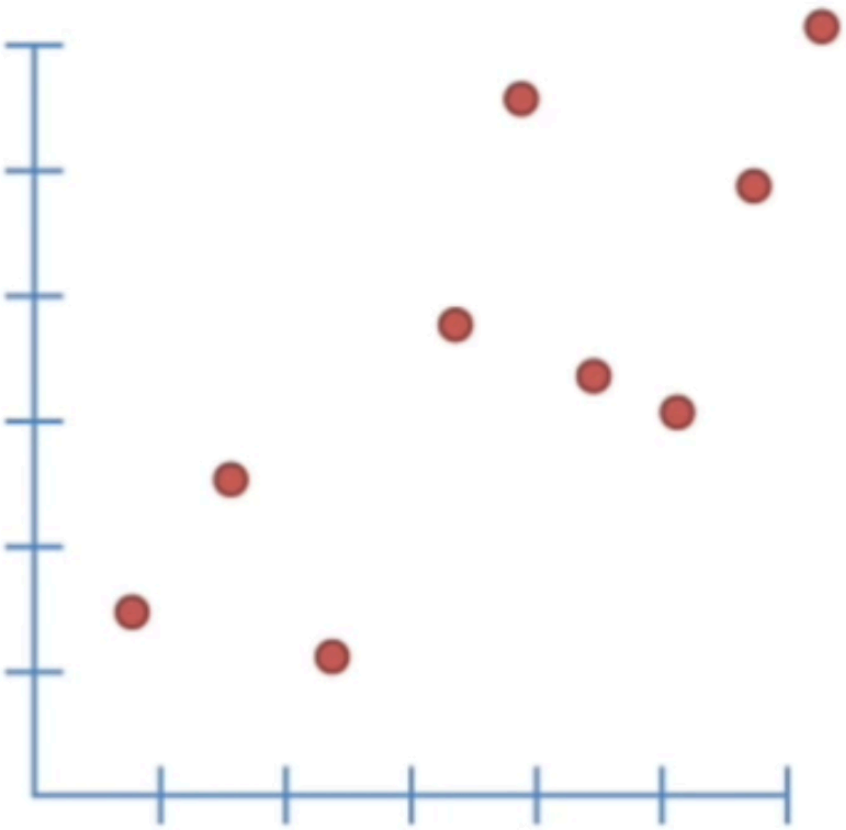
4 **actually fitting a line**

Linear regression in R

5 **lets fit a lm()**

live code session!

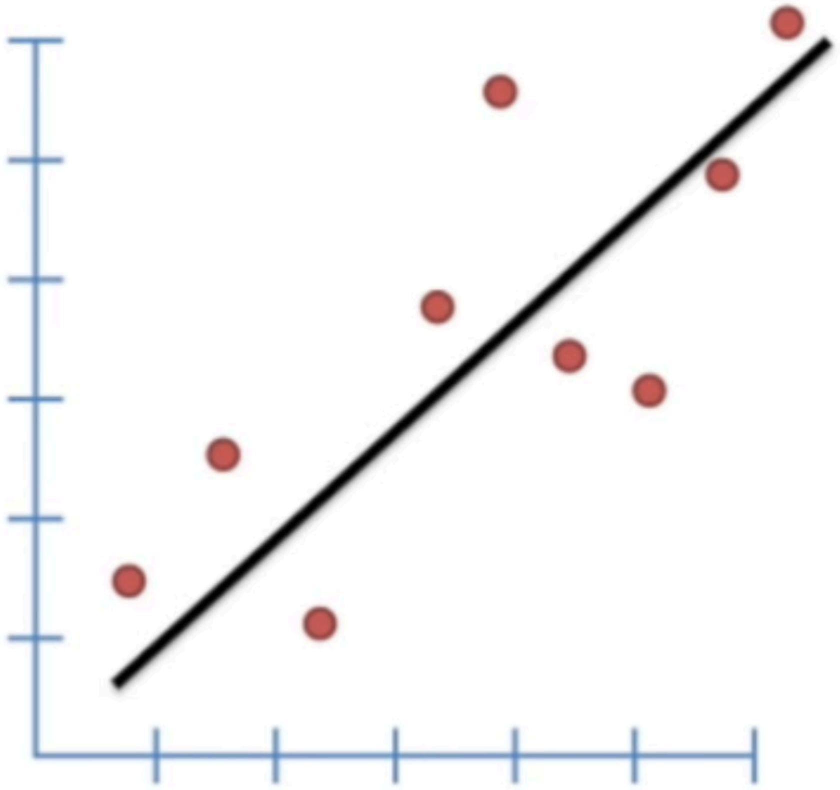
Try to fit some lines to the data



some data

we did a survey and now we got some data plotted on y-x axis

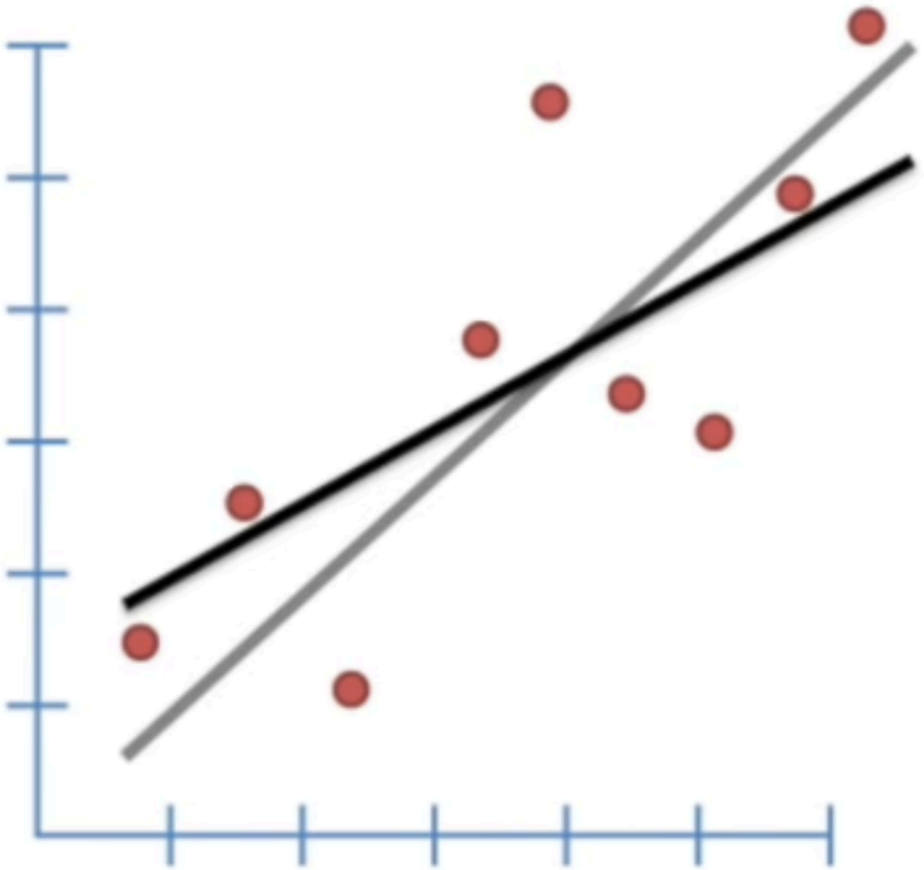
Try to fit some lines to the data



plot a line

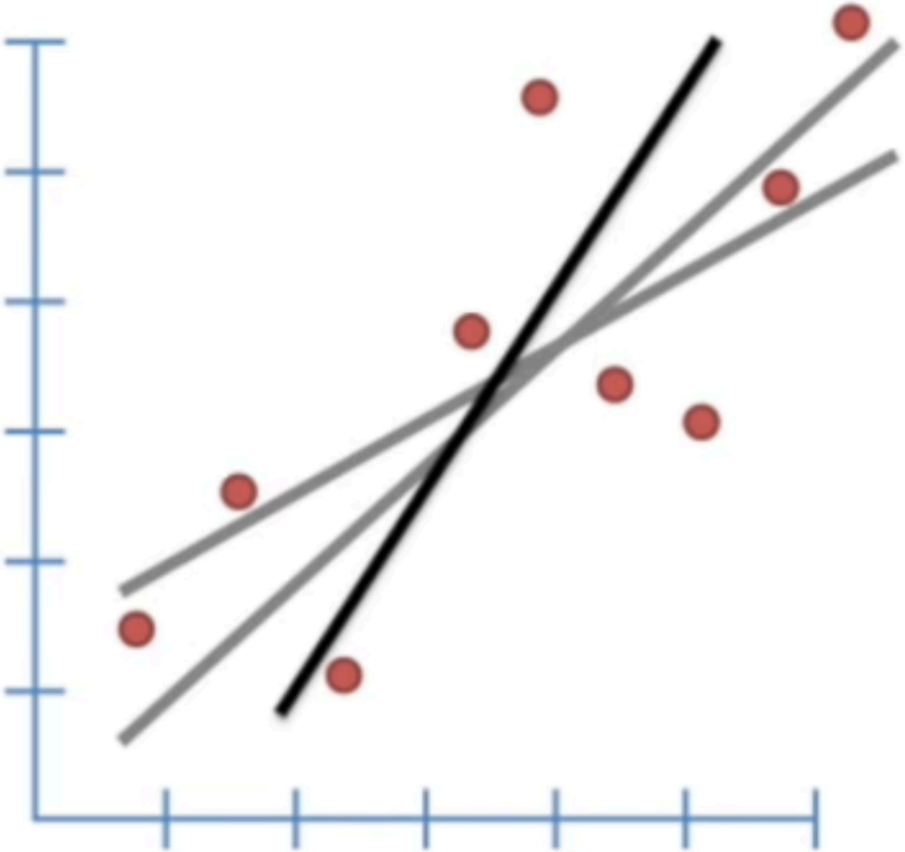
we usually want to draw a line on top of data to display trend- How about drawing a line which better interpolates data.

Try to fit some lines to the data



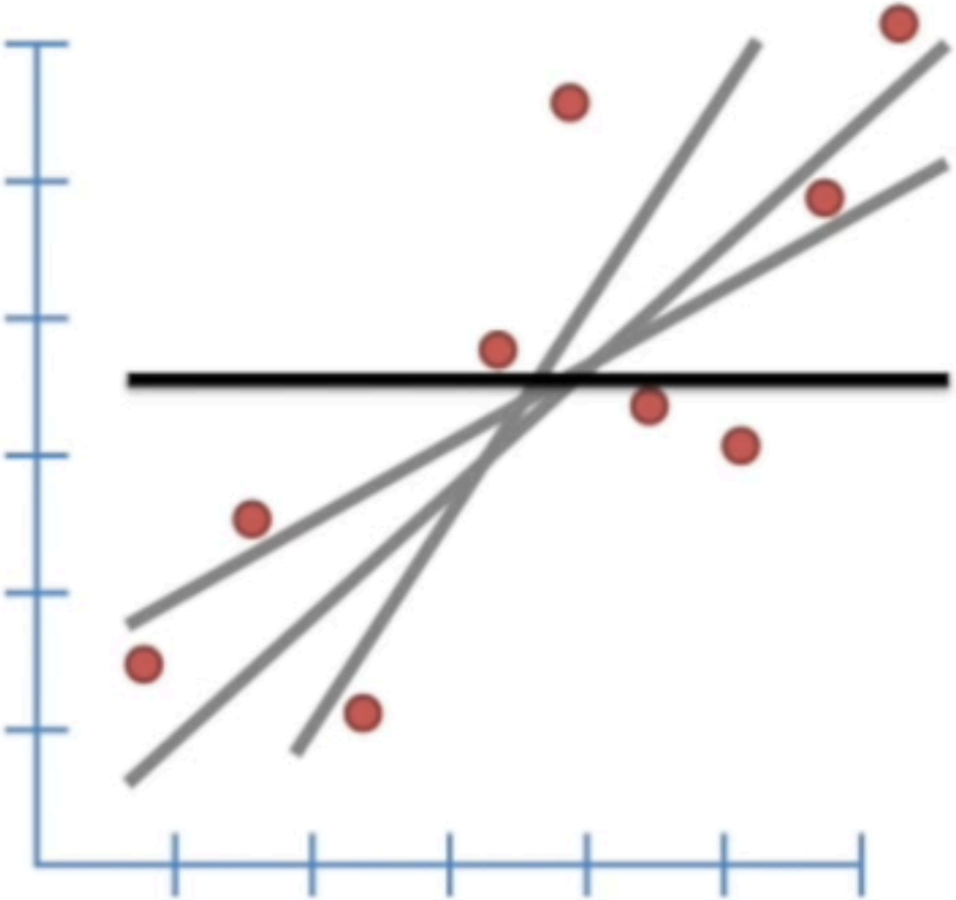
What about this line?

Try to fit some lines to the data



And This?

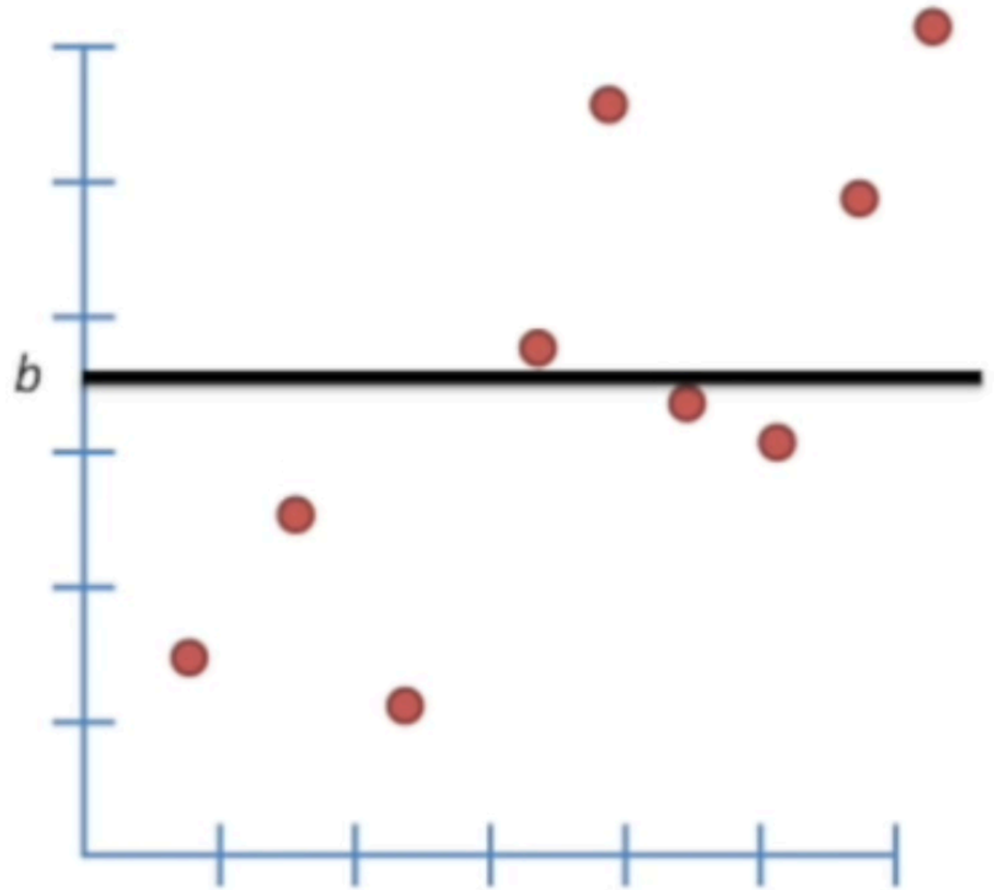
Try to fit some lines to the data



This one...

This horizontal line (y intercept) maybe is the worst one but it gives us a convenient starting point to discuss on how to find the optimal line to fit data.

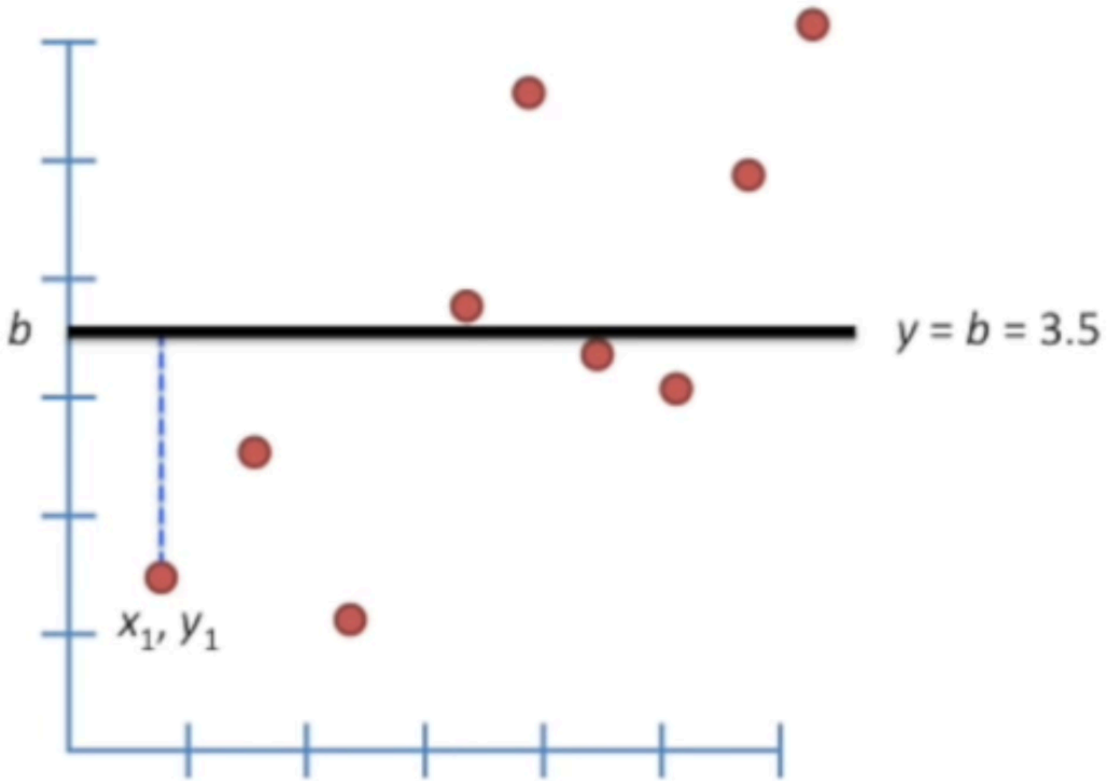
Try to calculate **residuals** against straight line



the “horriblezontal” line

this line cuts the cartesian in two and it intercepts the y axis on point b = 3.5 (count ticks number). That is 3.5 = b = y

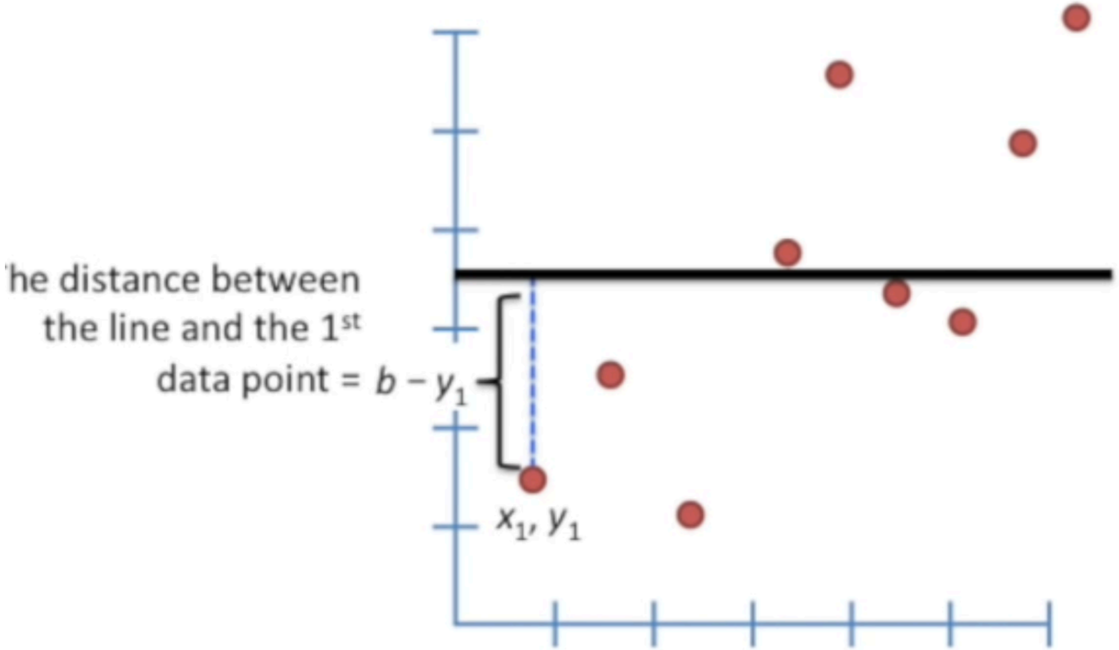
Try to calculate **residuals** against straight line



How bad?

We could argue that the more is the distance between the line and the point the wort the fit. So we measure the Cartesian distance between point 1 (**x1, y1**) and **b.**

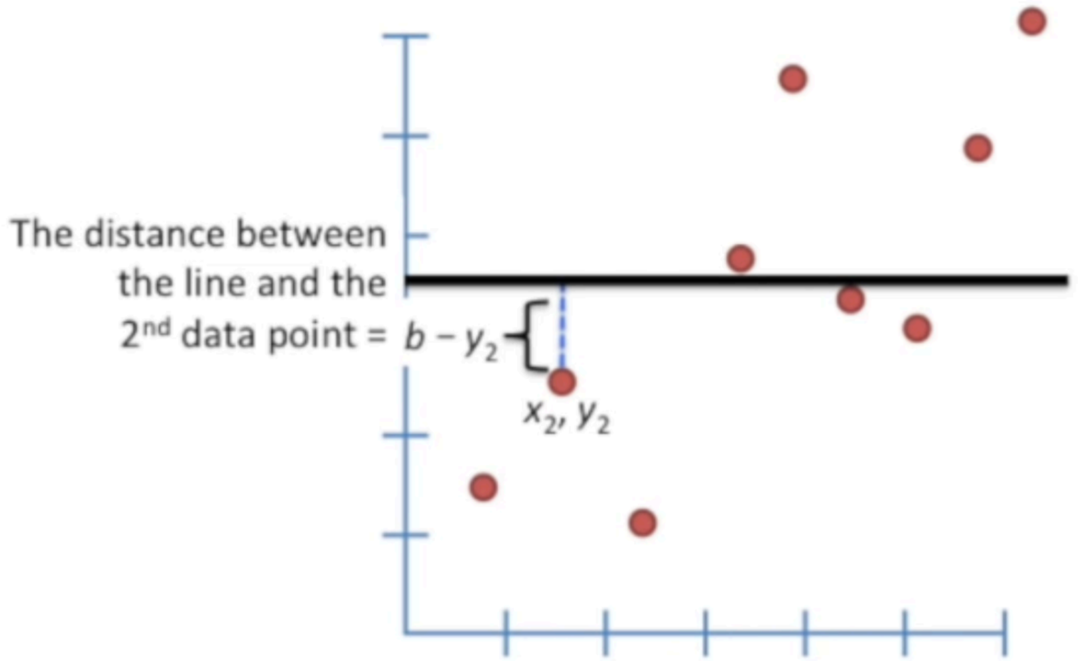
Try to calculate **residuals** against straight line



distance from point1 and b

In this case since y is horizontal then the distance from point 1 to y is equal to b - y1. These are called residuals.

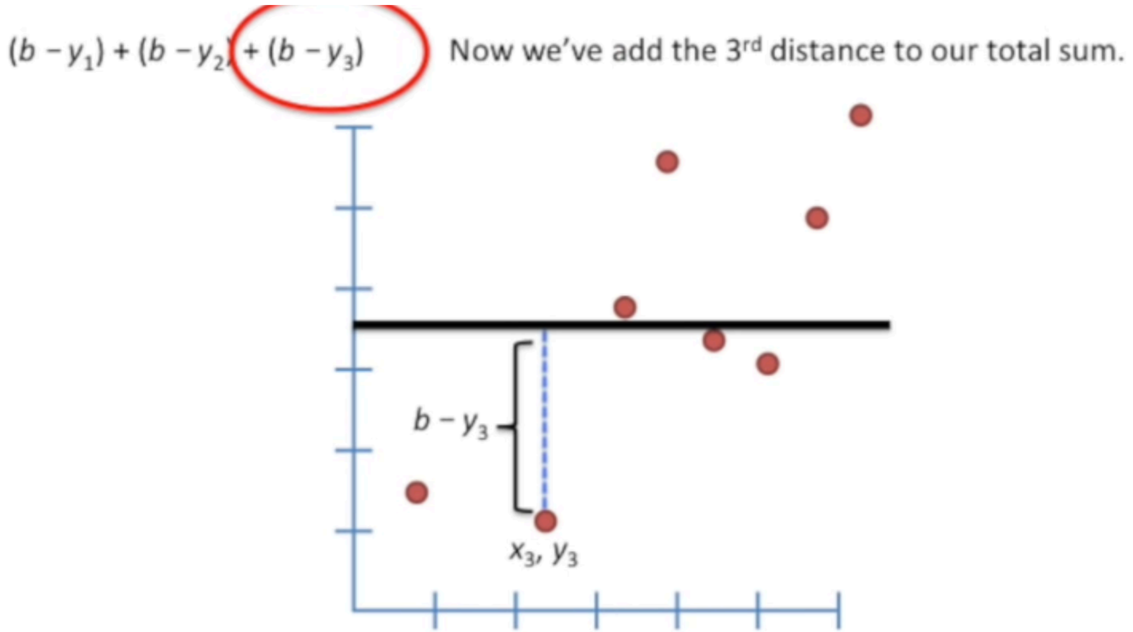
Try to calculate **residuals** against straight line



Now we go on for each point

This the distance from **point2** and **b.**
For now we are summing each distance from b for each point!

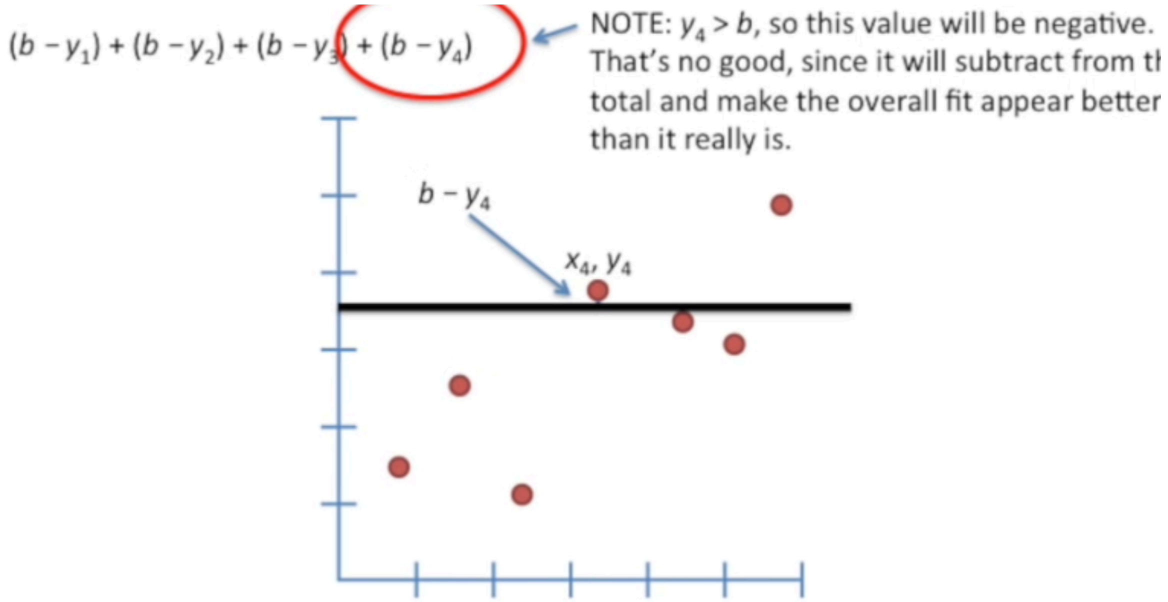
Try to calculate **residuals** against straight line



3rd distance

check in the upper part how we keep track and sum each distance

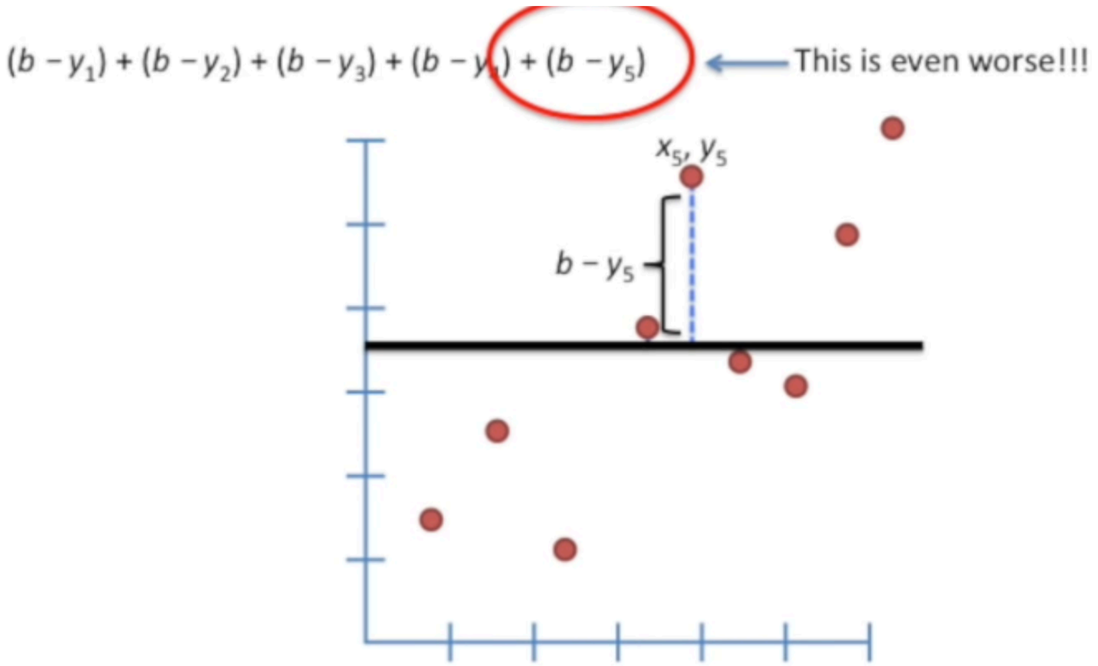
Try to calculate **residuals** against straight line



4th

Notice that if the point it is above **b** then distance in negative since b -y4 is actually a negative number.
This results actually in a better fit since then sum of distance now is decreasing.

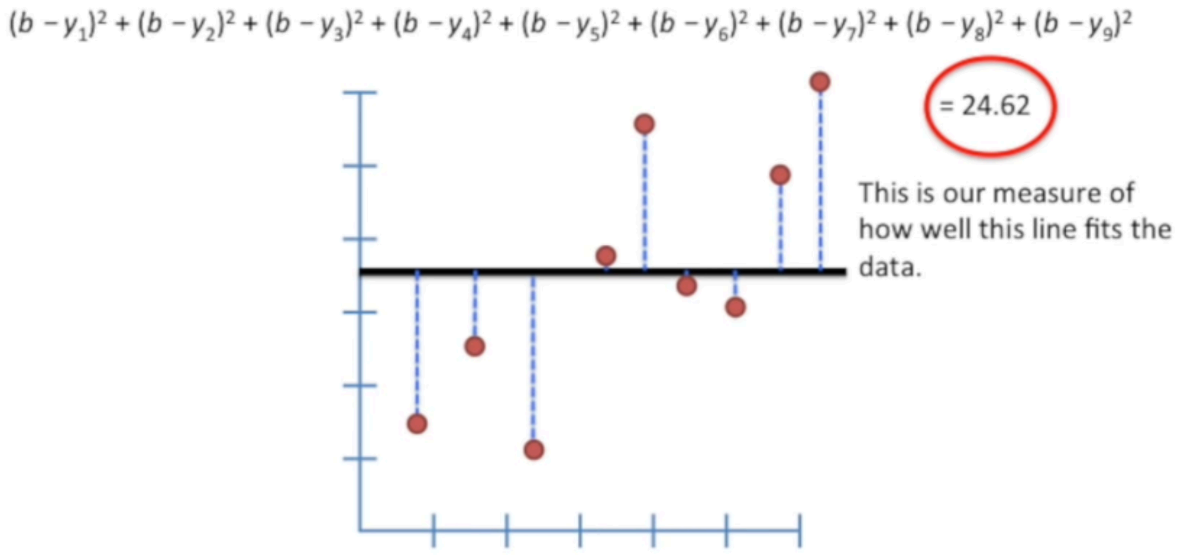
Try to calculate **residuals** against straight line



5th

This is also higherthen the 4th which it reflect in an even worst fit (since now sum of positive is quite similar to sum of negatives). We need to change that.. people used to take absolute value of each distance.

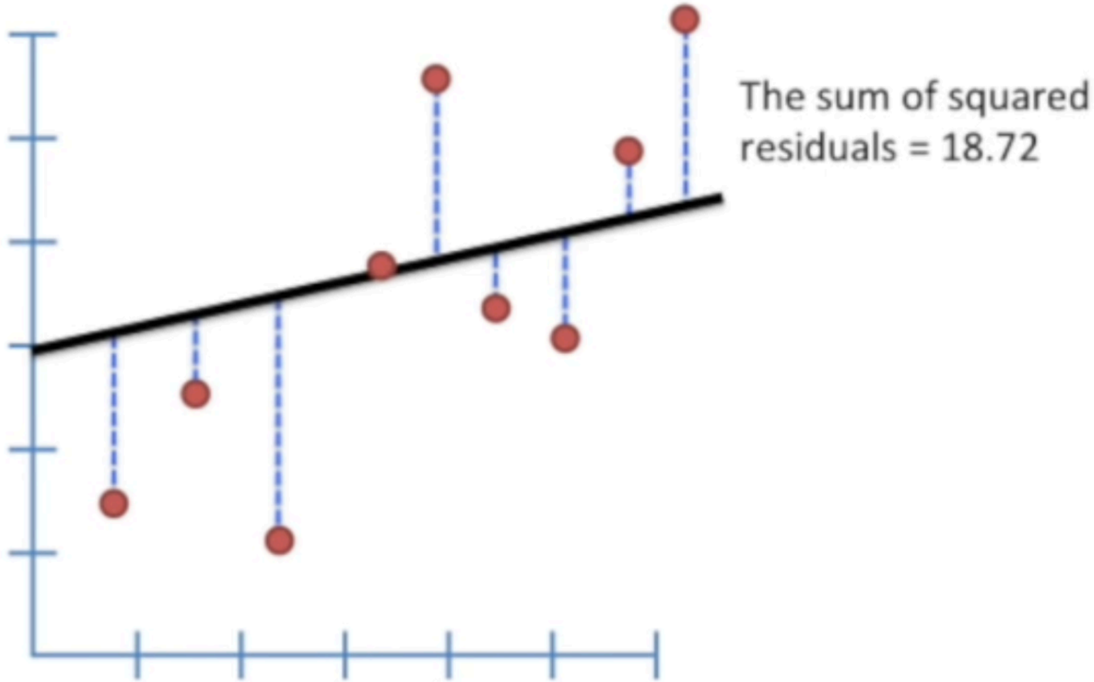
Try to calculate **residuals** against straight line



Sum of Squared distances

But now people squares each distance and then sum it together. In the “horriblezontal” case the sum is **24.2**

Least Squares

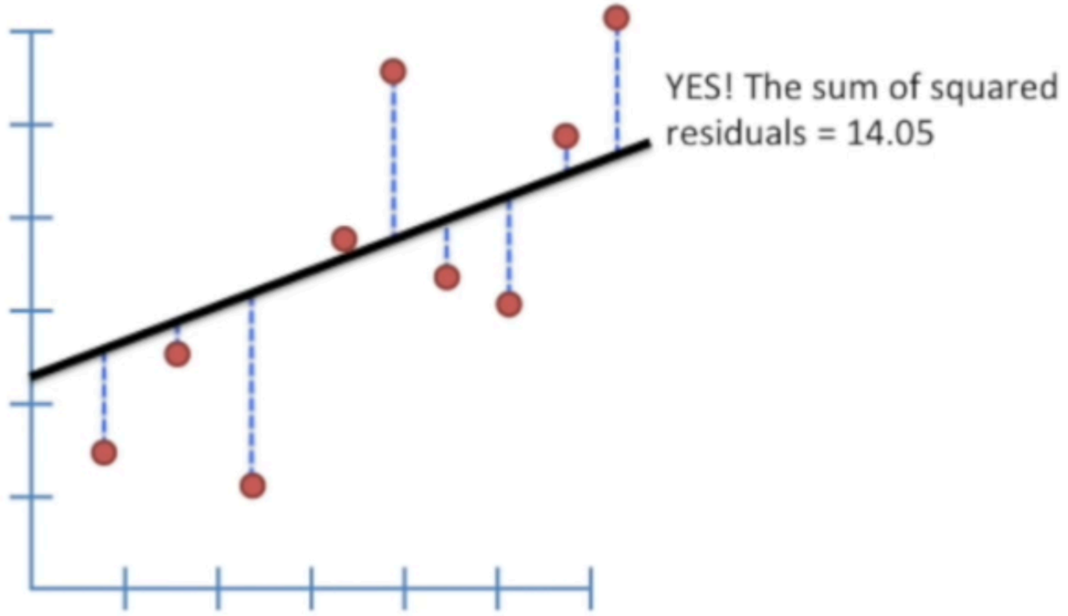


distances are actually what we call **residuals**.

so once we take the residuals from fitted line and a data points and we sum them together we obtain something called **Sum of Sqared Residuals** i.e. SSR**.**
In this case the SSR is **18.72,** which is kinda better than before!

Least Squares

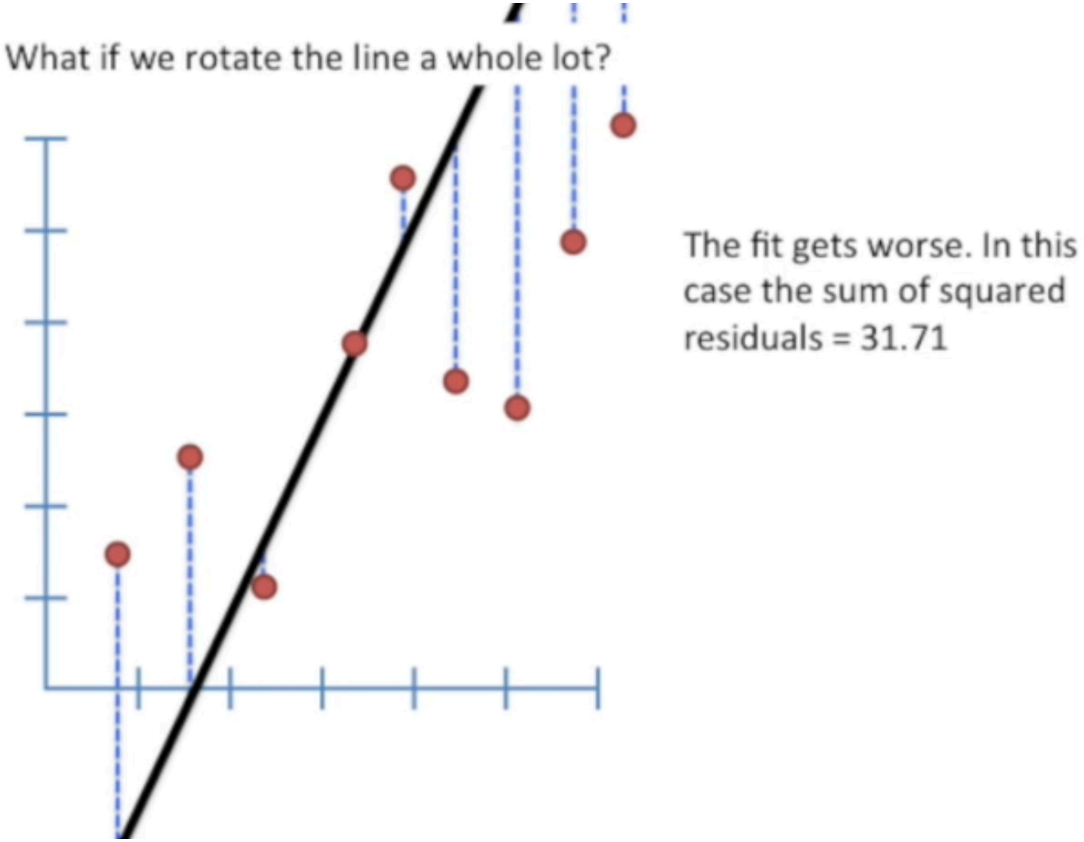
Does this fit improve if we rotate a little more?



Now lets try to slightly rotate the line

Now SSR is **14.05.** This keeps going down!, we are doing good. we are moving towards a better fit!

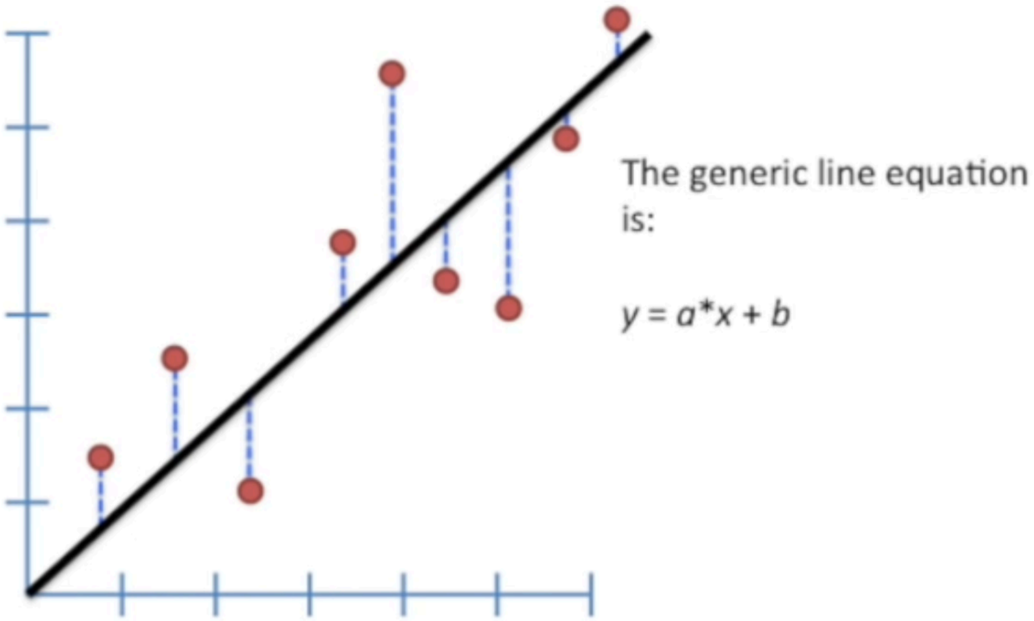
Least Squares



What if we rotate A LOT!

at this point we just want to try this out and see how the **SSR** increases, This case is **31.71** that's even worst the the horizontal case,
“there should be a sweet spot in between the horizontal and the vertical that minimizes **SSR.**

Least Squares



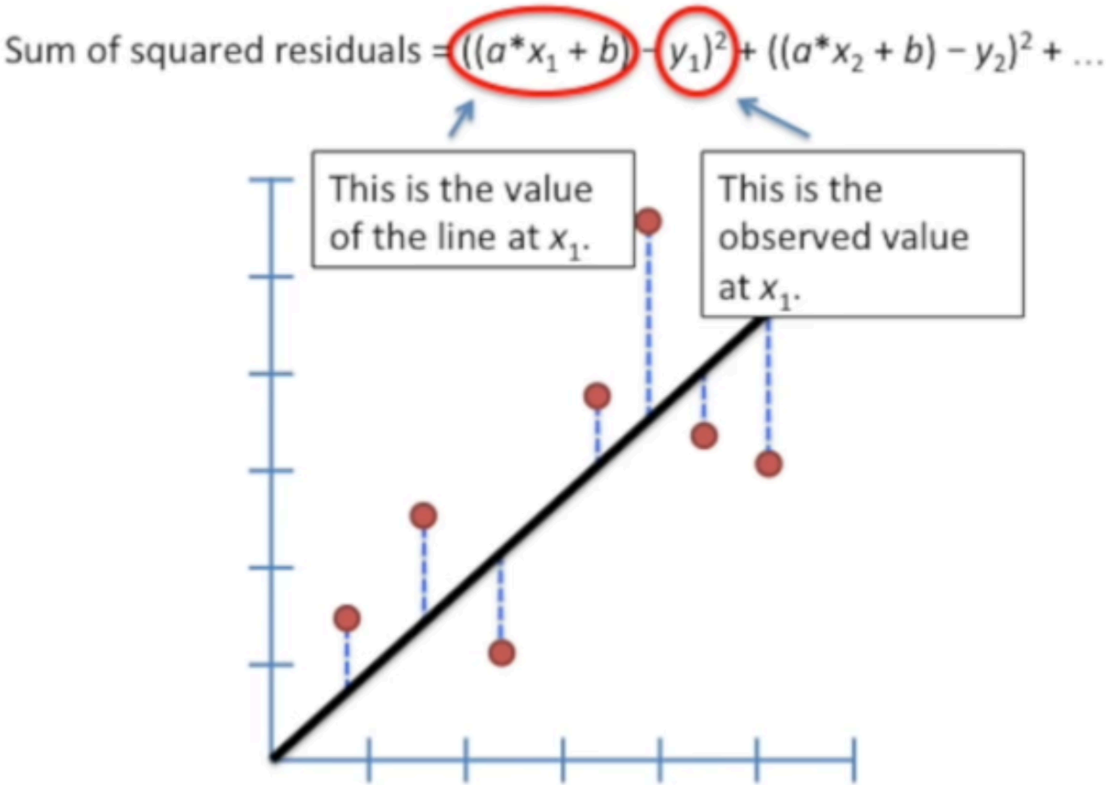
Lets rewrite it in a math notation

Do you remeber from from bachelor/high school linear algebra what is the math expression for a line with a slope?

y = a*x + b

a = slope
b = intercept

Least Squares

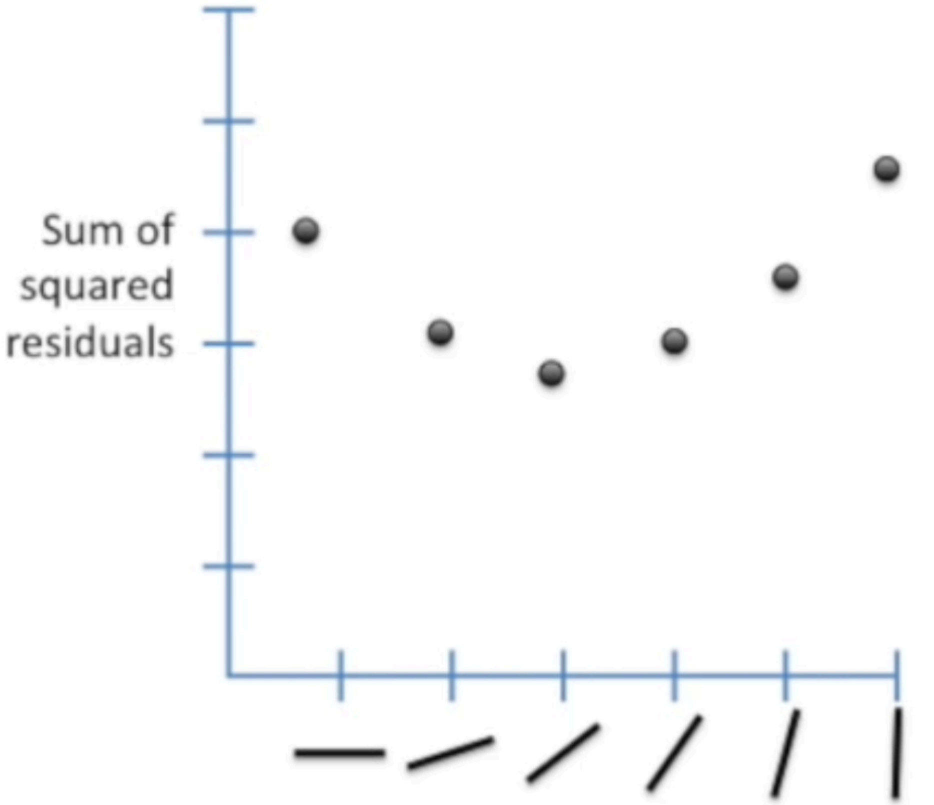


So the math problem to solve is…

"find the optimal values for a and b so that **SSR** is min".
In more general math term the SSR is the expression above the chart.
the left red oval is the value of line at x1 and the right one is the observed value of x1,
What we are really doing is calculating the distances as we did before!

This is something called **least squares, meaning "take the min along all of the SSR"**

How SSR evolves with line rotation

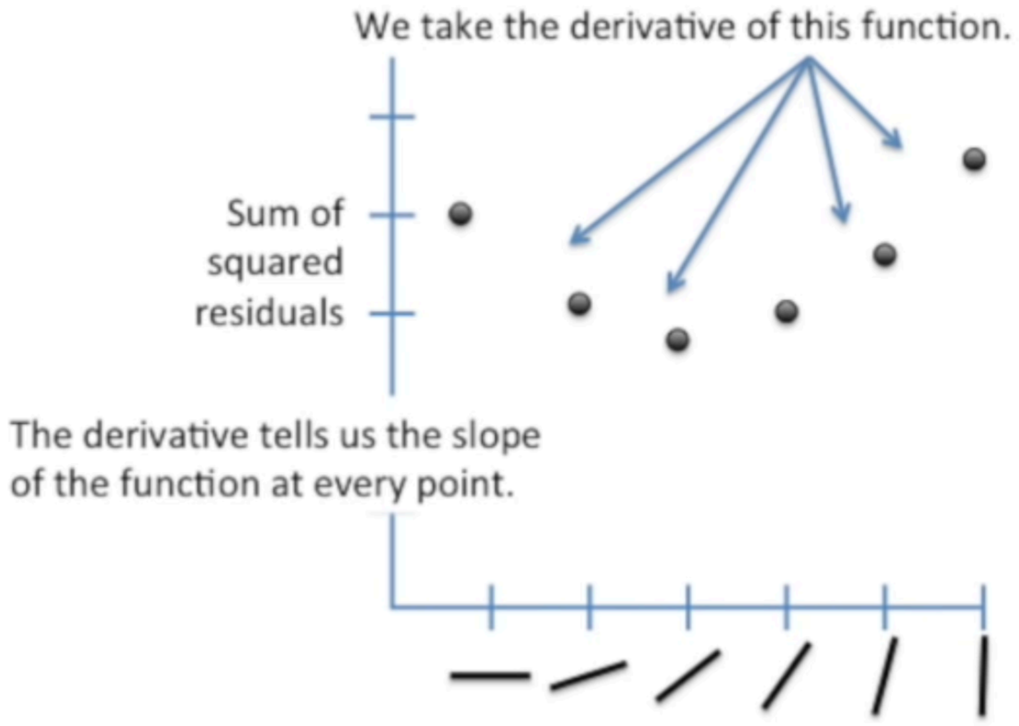


Lets plot it

on the y axis there's **SSR. on** x axis the line **rotations** we tried before.

we observe that SSR for horriblezontal is initially quite high then it keeps decreasing up to “flex” (for italian speakers “punto di flesso”), Then it suddenfly istarts to increase again the more we move toward vertical rotation

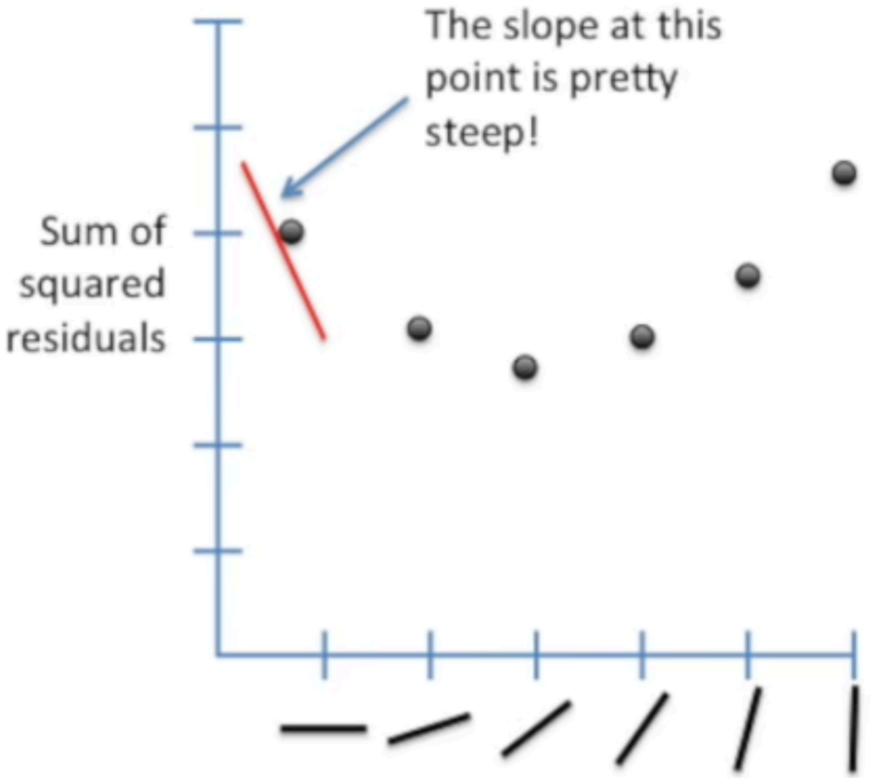
How SSR evolves with line rotation



how to find optim?

we take the derivative (i.e. the tangent' slope) of the function describing how SSR moves wrt to rotation and we look for the **0**!

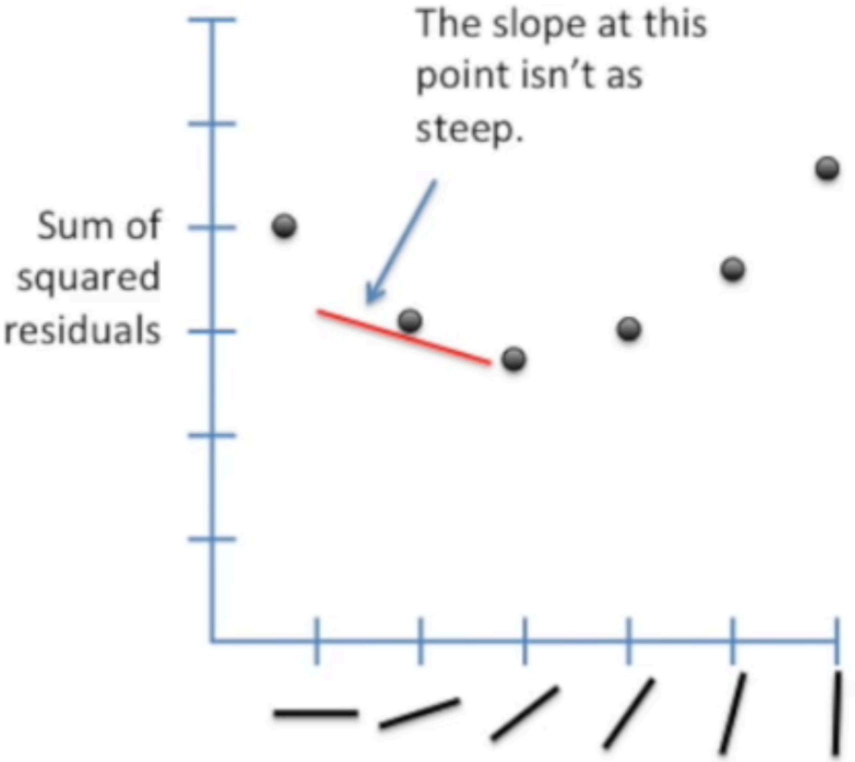
How SSR evolves with line rotation



1st rotation

We take the derivative of the SSR expression for the horriblezontal line, That's steep and it is negative, that's not what we are looking for,

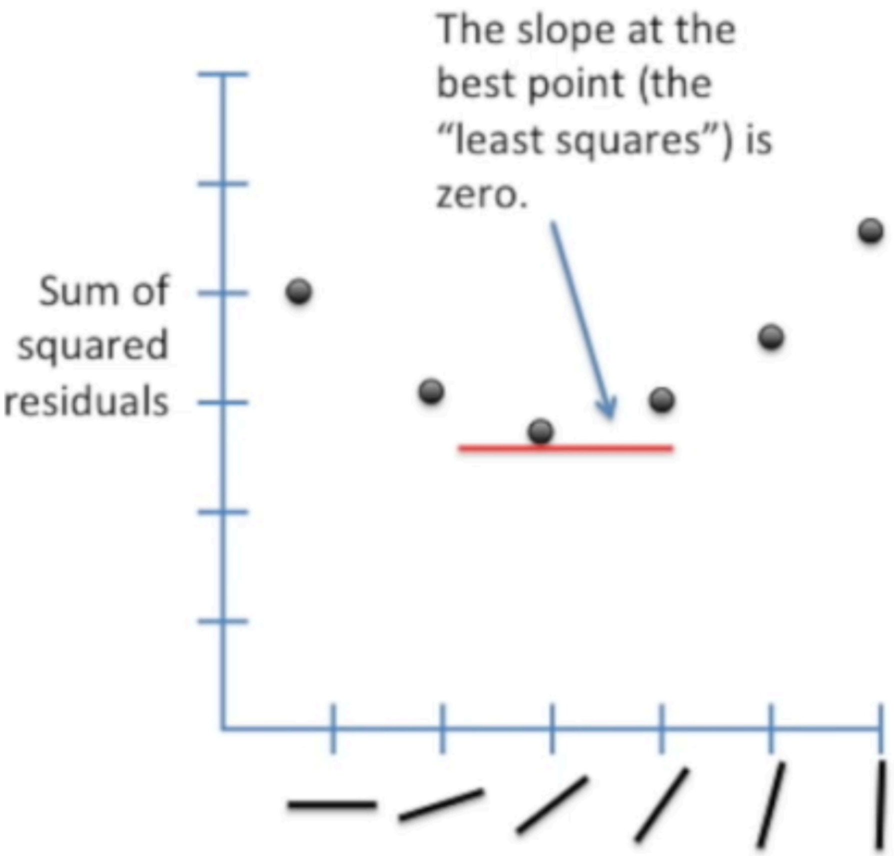
How SSR evolves with line rotation



2nd rotation

Well this is still slightly negative but that's way better. One further interesting point is that we are moving from negative to still negative, that means *monotonous*.

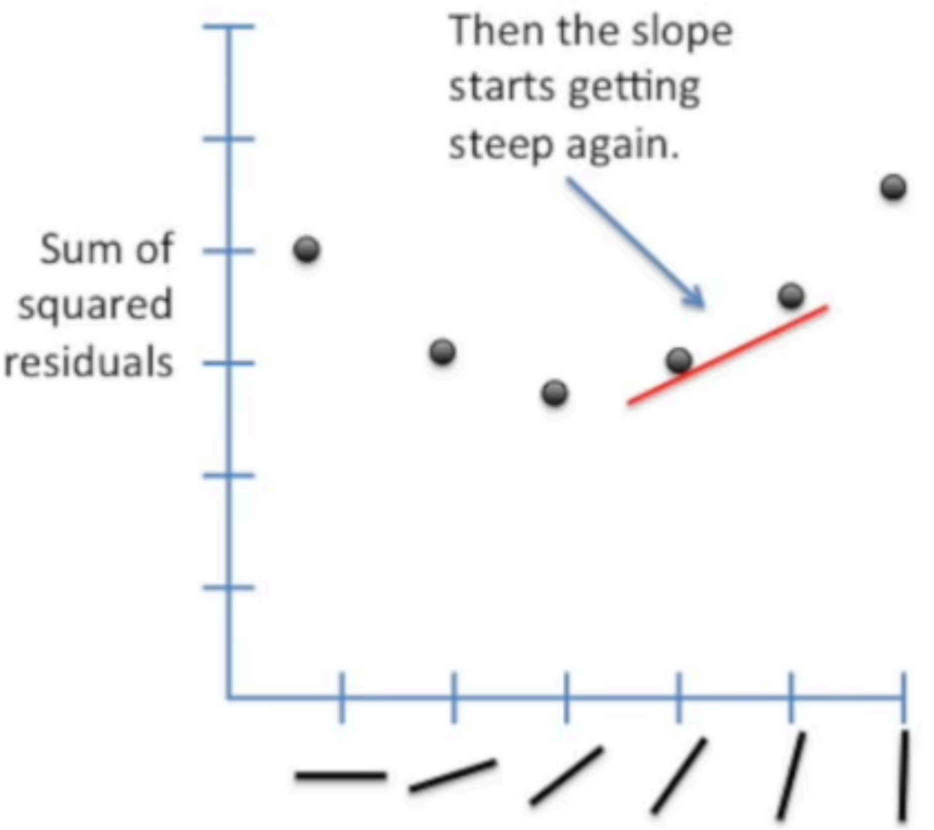
How SSR evolves with line rotation



3rd there you go!

This is **0** and we are sure it is since there are not any further min points below that. We might wanto to try for fun the other derivatives values.

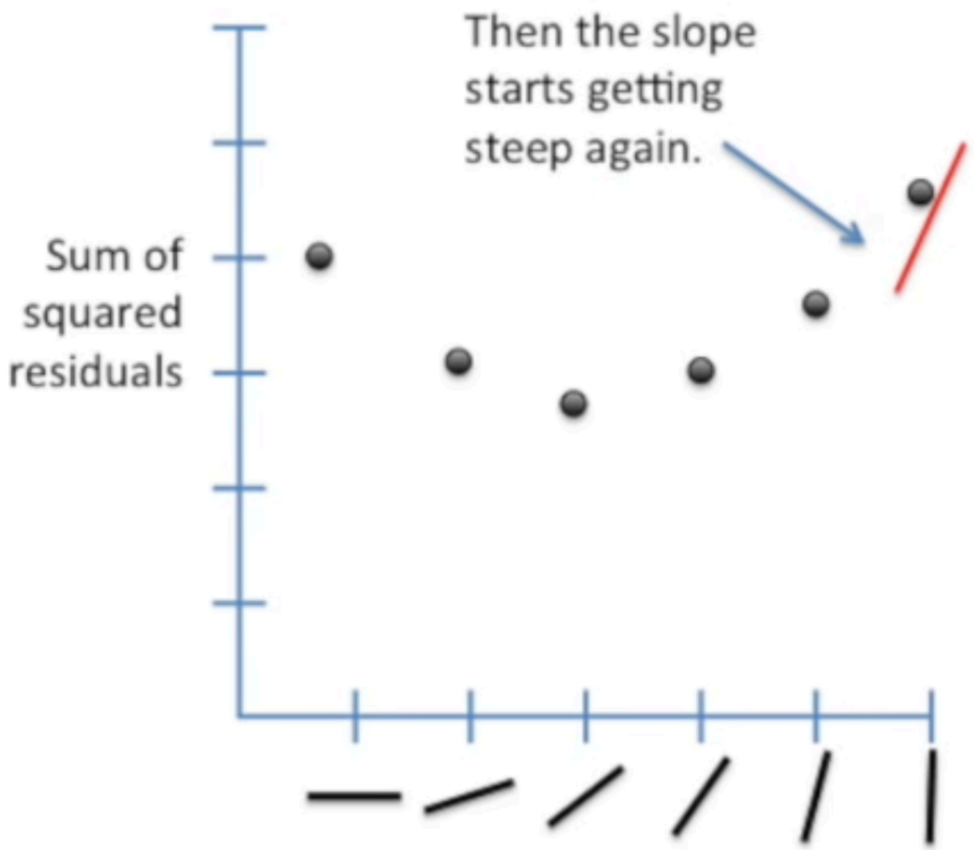
How SSR evolves with line rotation



4th now positive!

this time the derivative value is positive., This kinda confirm our hypothesis that 3rd rotation is **the best 0. since from now on we expect the curve to increase.**

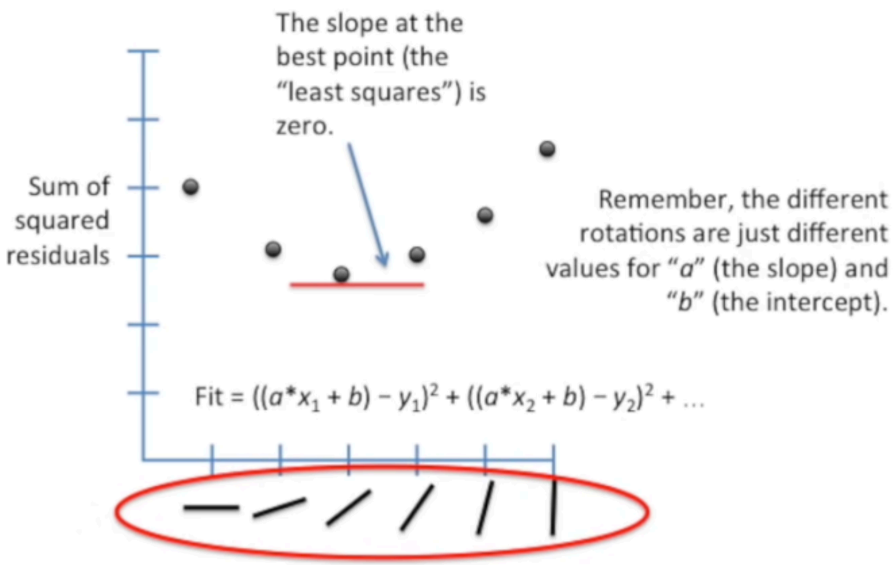
How SSR evolves with line rotation



5th that's vertical

how bad it is.

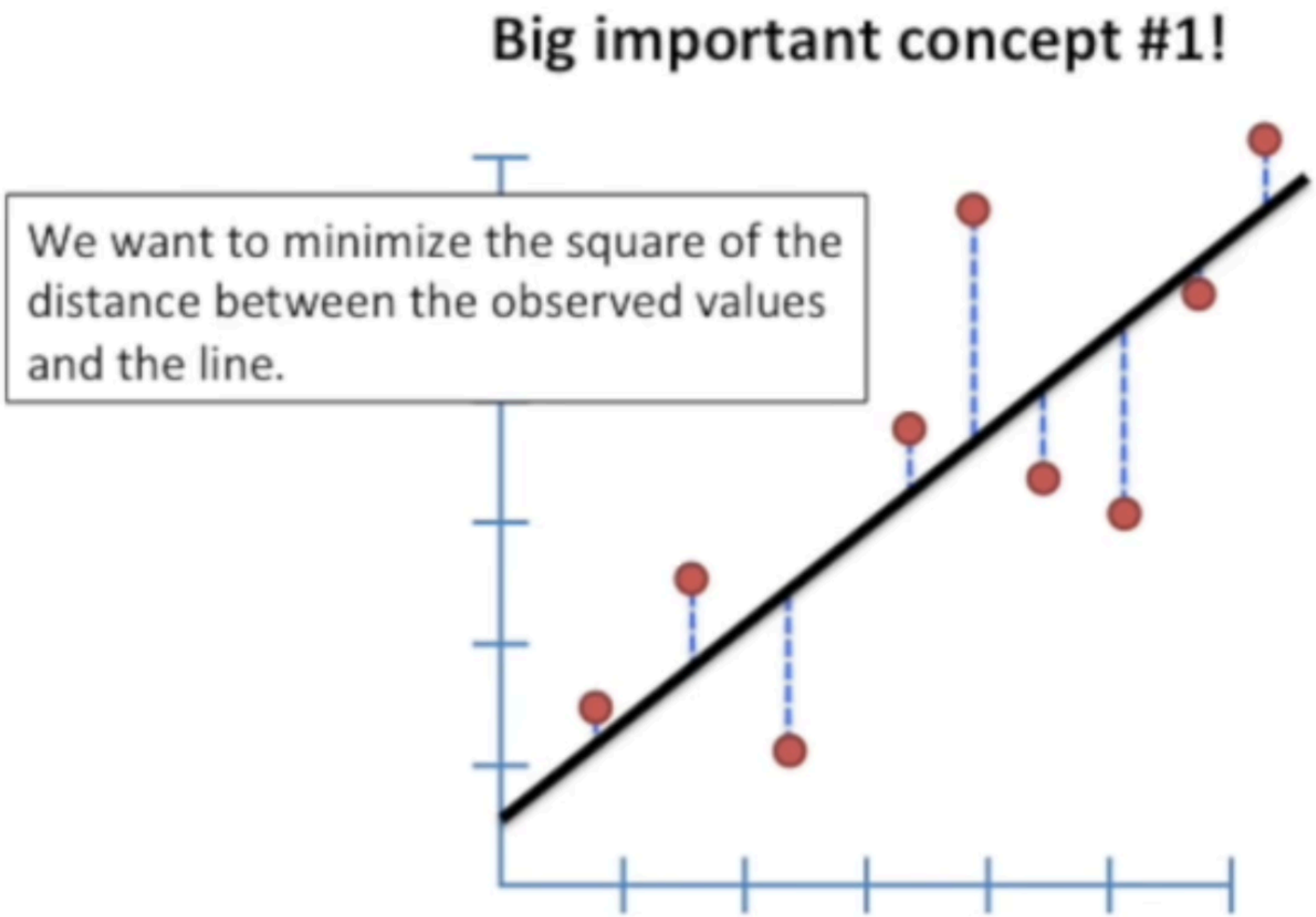
How SSR evolves with line rotation



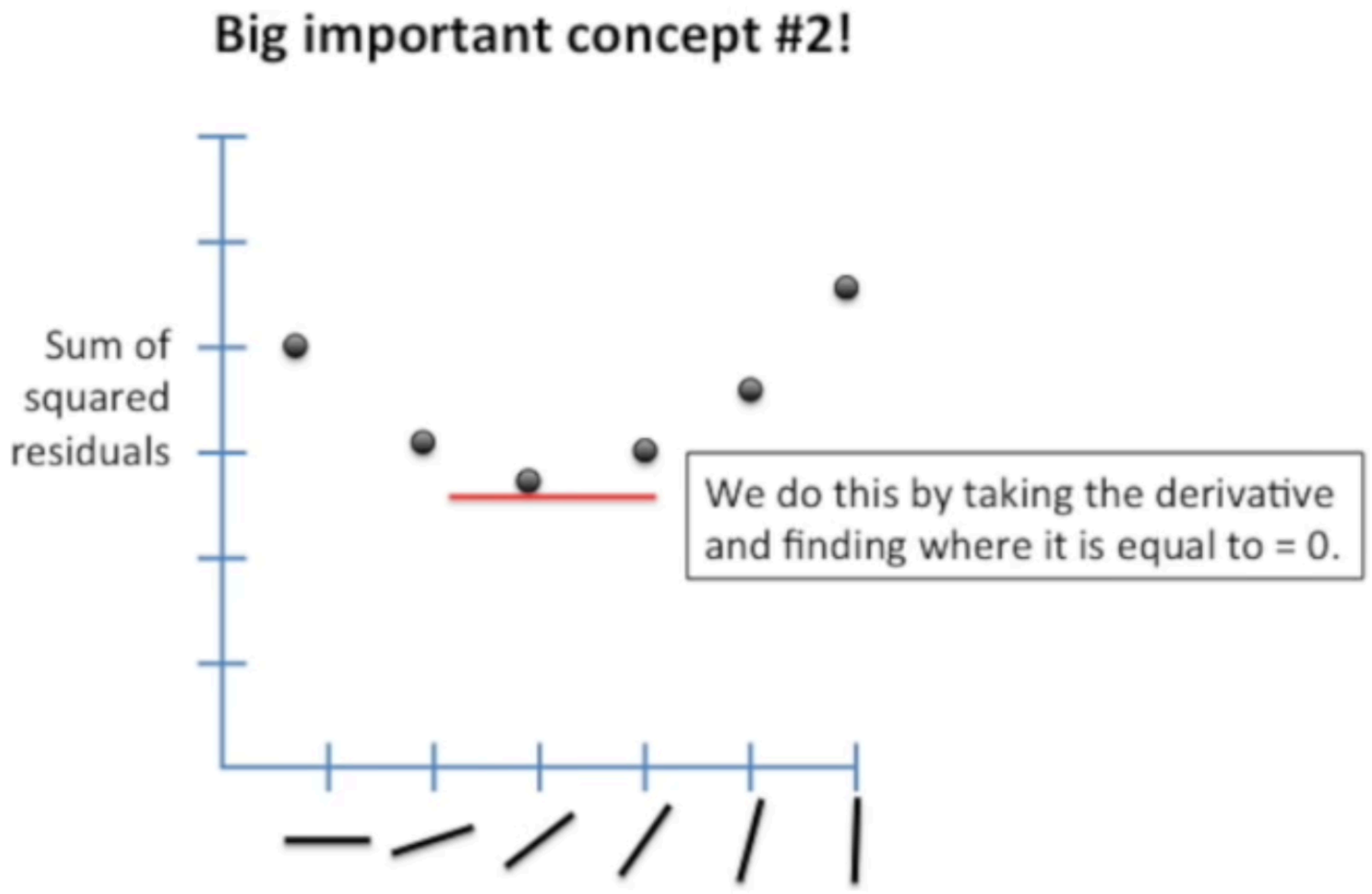
the best fit!

remeber that rotation are actually different values for **a** and **b.**

Caveat 1


Big important concept #1!
We want to minimize the square of the distance between the observed values and the line.

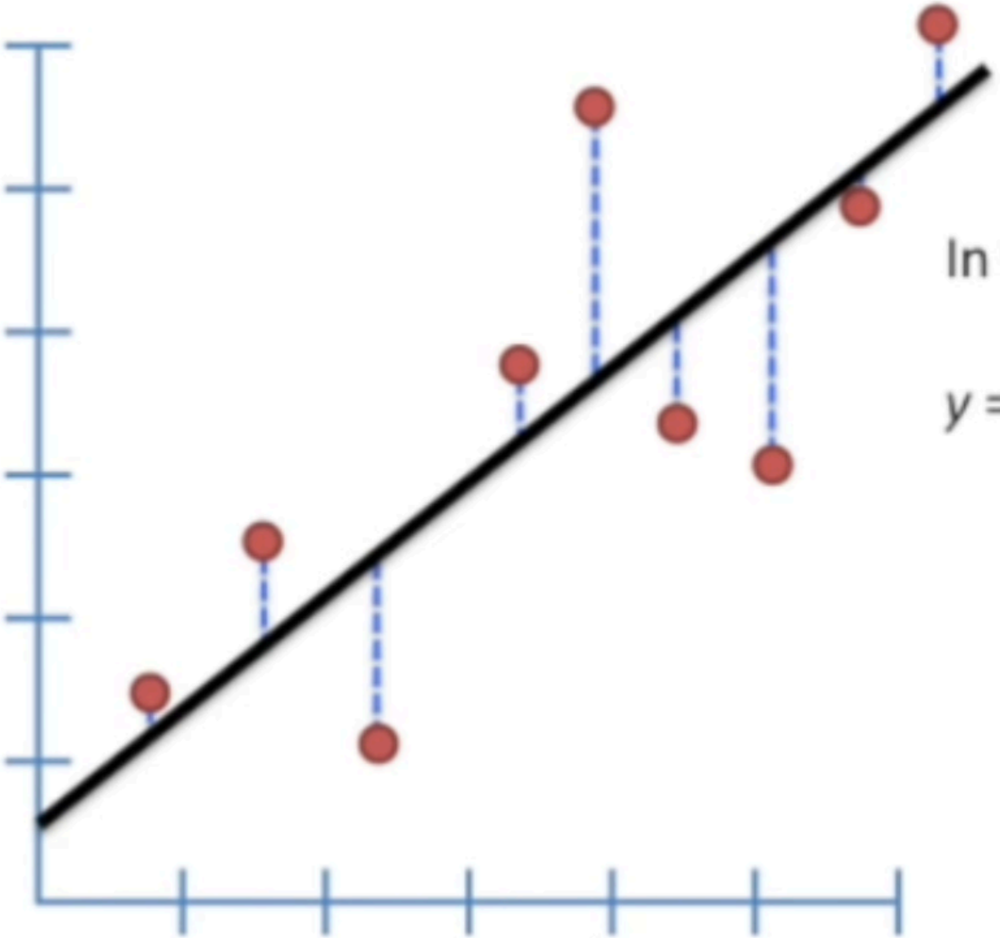
Caveat 2



Big important concept #2!
Sum of squared residuals
We do this by taking the derivative and finding where it is equal to = 0.

Final remark

The final line minimizes the sums of squares (it gives the "least squares") between it and the real data.



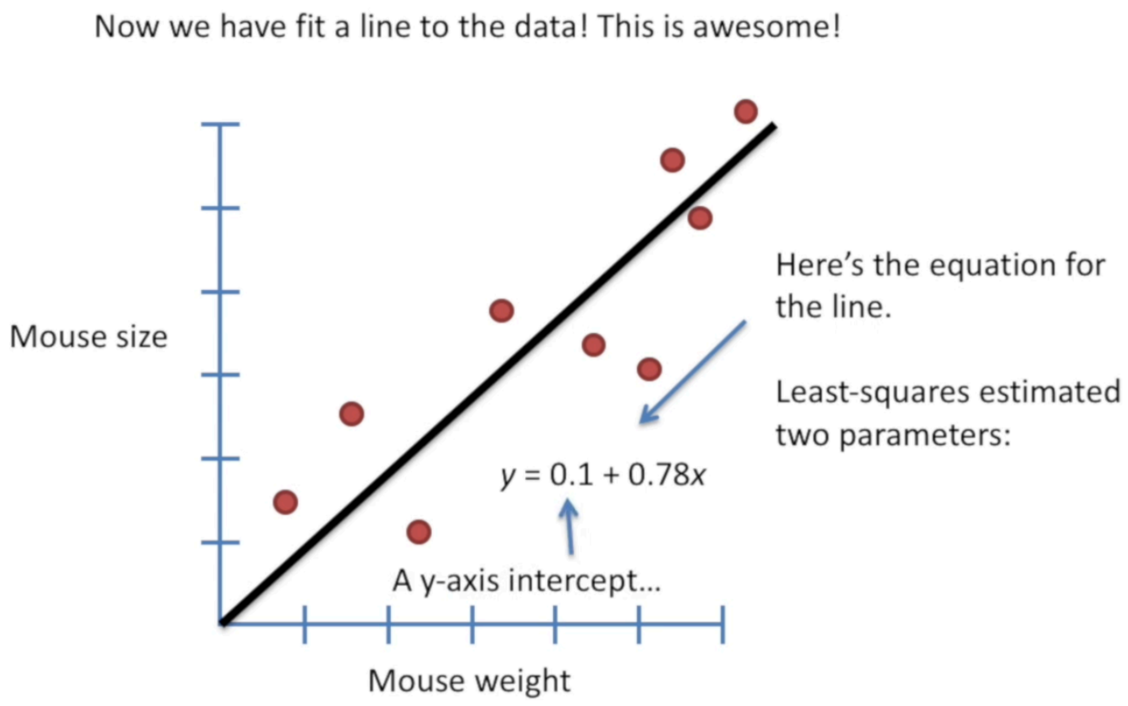
In this case, it is

$y = 0.77 * x + 0.66$

Section 2

Principles of **Linear Regression**

So now that we **know** how fit a line...

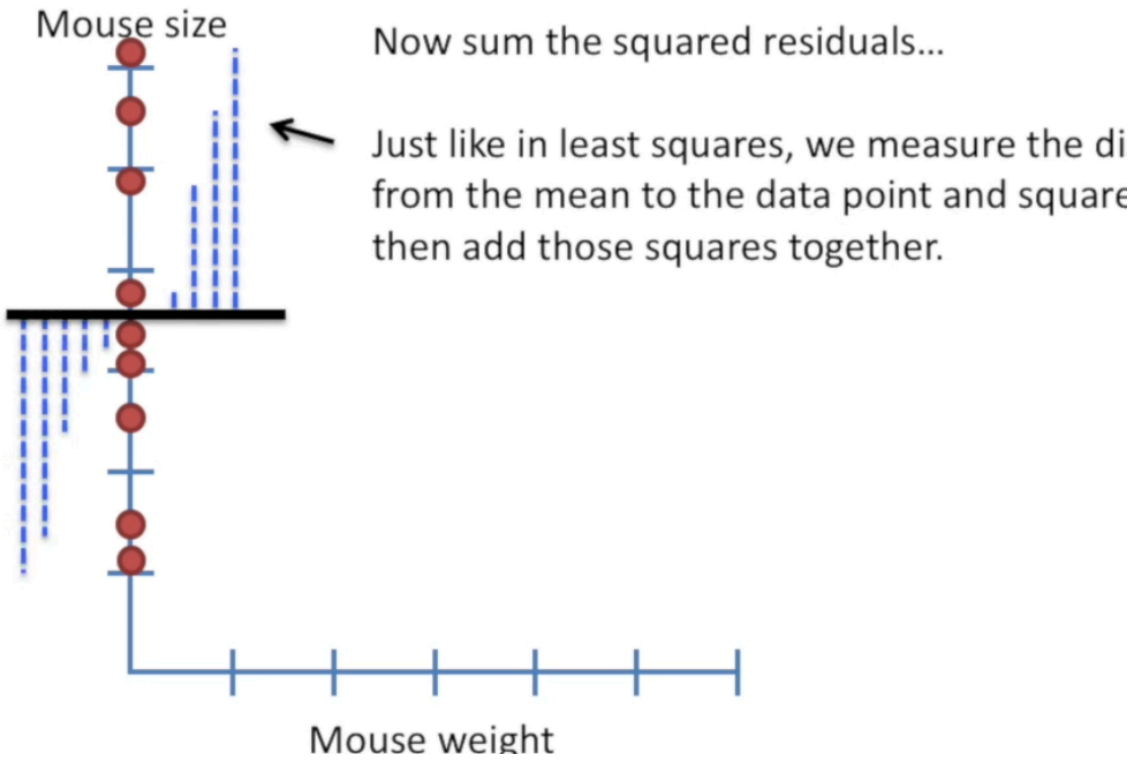


How good is the fit?

slope is not 0 means that knowing a mouse's weight will help us make a guess about that mouse's size.

but how good is that guess?

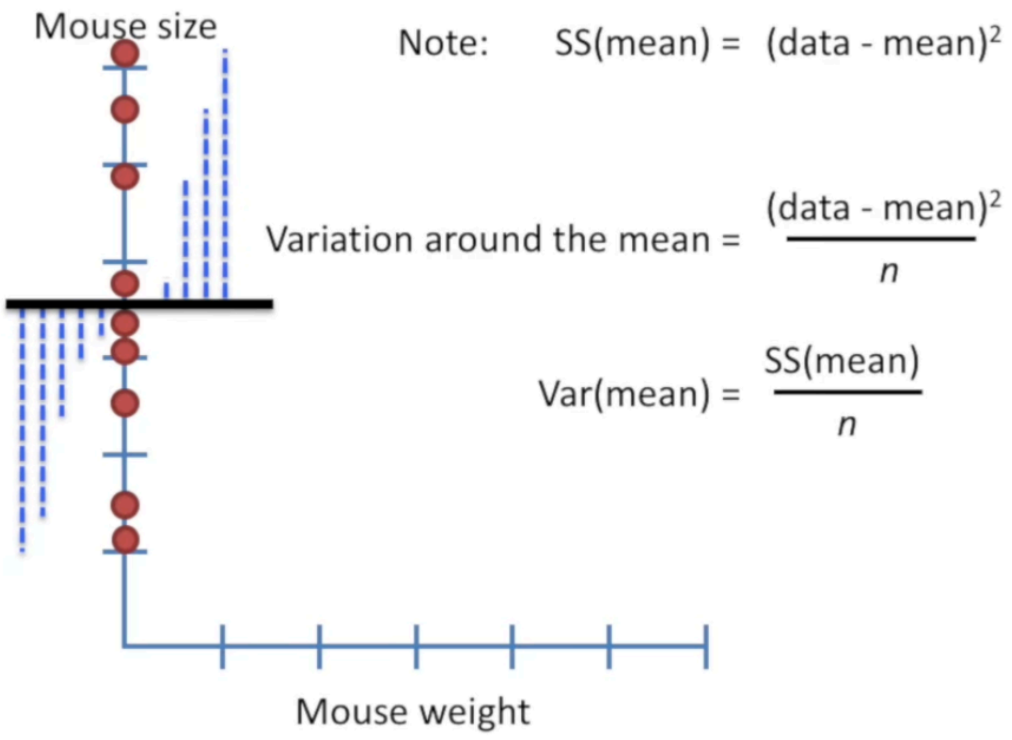
So now that we **know** how fit a line…



calculate R2

we shrink data point to the size axis showing we are interested only in mouses' size and we calculate teh distance between mouse size mean and each mouse size. this is called "**sum of squared around mean**"

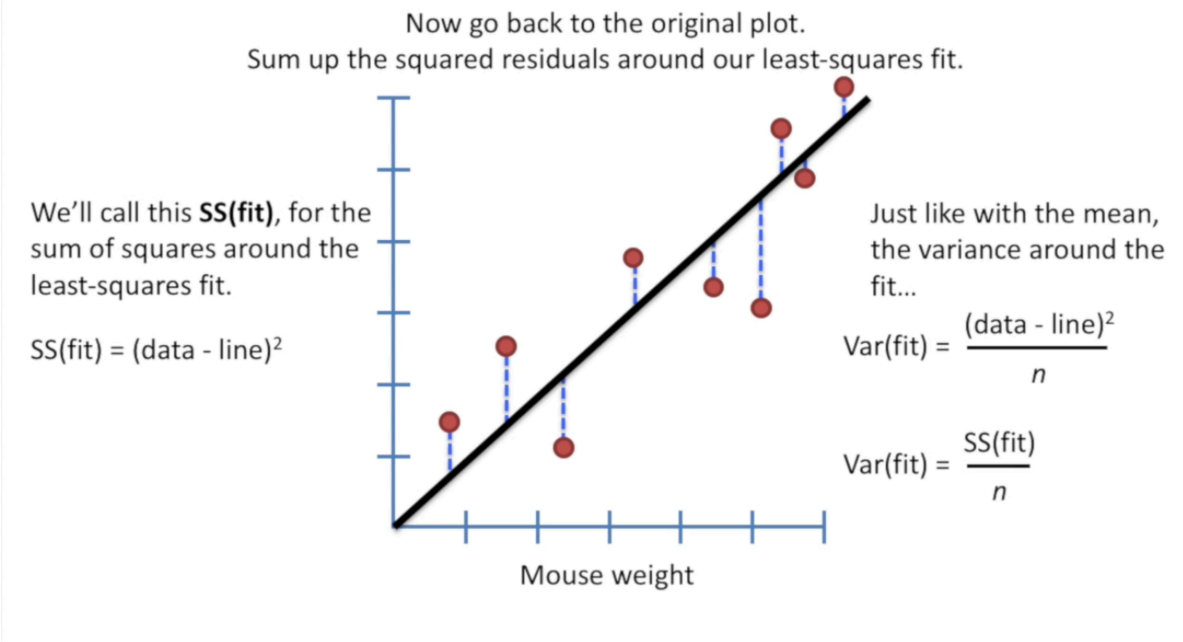
So now that we **know** how fit a line…



Variation around the mean

This is how much data is distanced from its actual mean.
Do you remember the "horriblezontal" line?

So now that we **know** how fit a line...

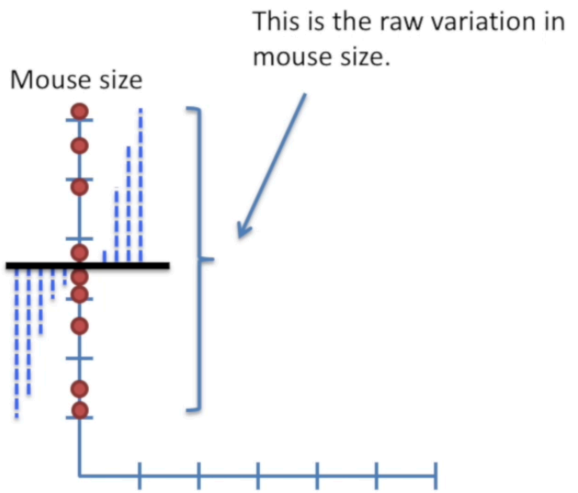


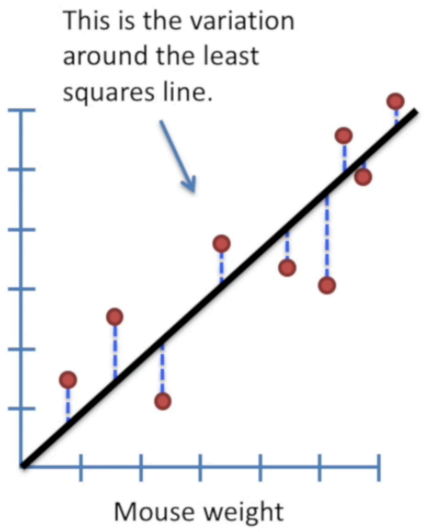
now back to the line

Now we do exactly the same, we compute residuals also from the line,

Remember the variance of "something" is always equal to the sum of squares divided by the number of those things.

So now that we **know** how fit a line...

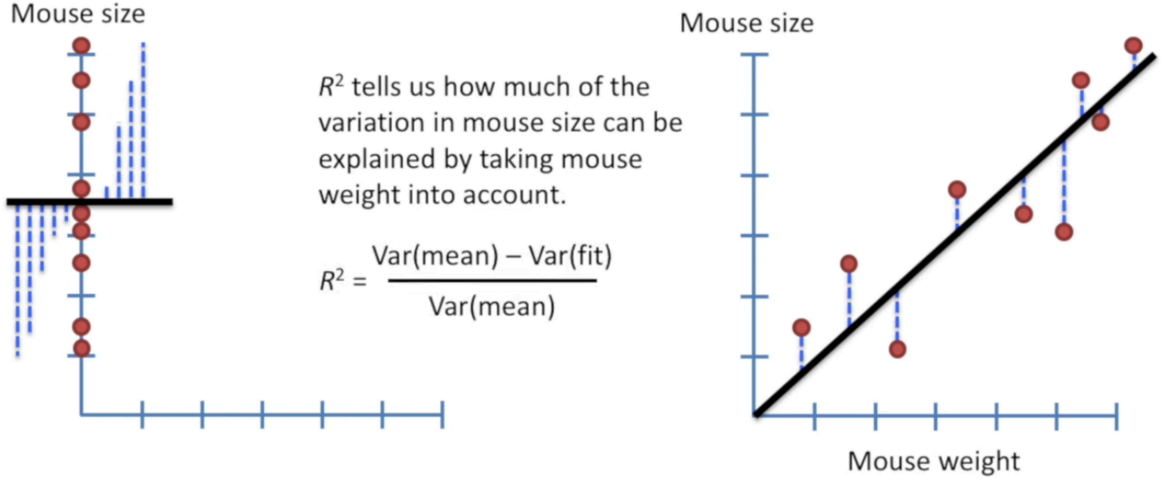




wrap it up

- **left side:** raw variation in mouse zies
- **right side:** variation around least squares line

So now that we **know** how fit a line...



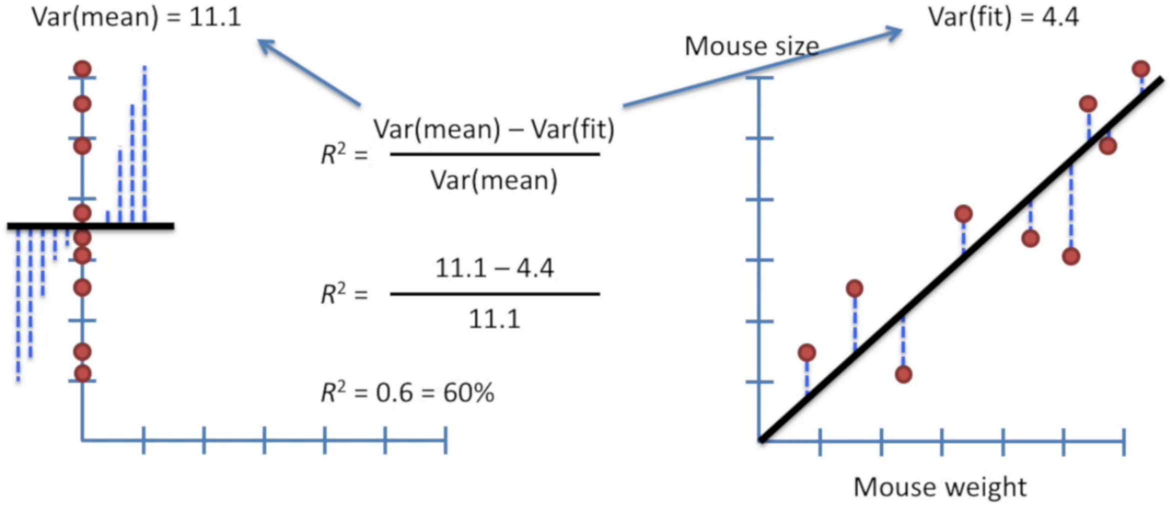
there is less

there is less variation around the fitted line than the one with the horiziontal so we sa that some of the variation may be explained by taking mouse weight into consideration.

In other words:
“heavier mice are bigger, lighter mice are smaller”

and R2 says exactly this

So now that we **know** how fit a line...



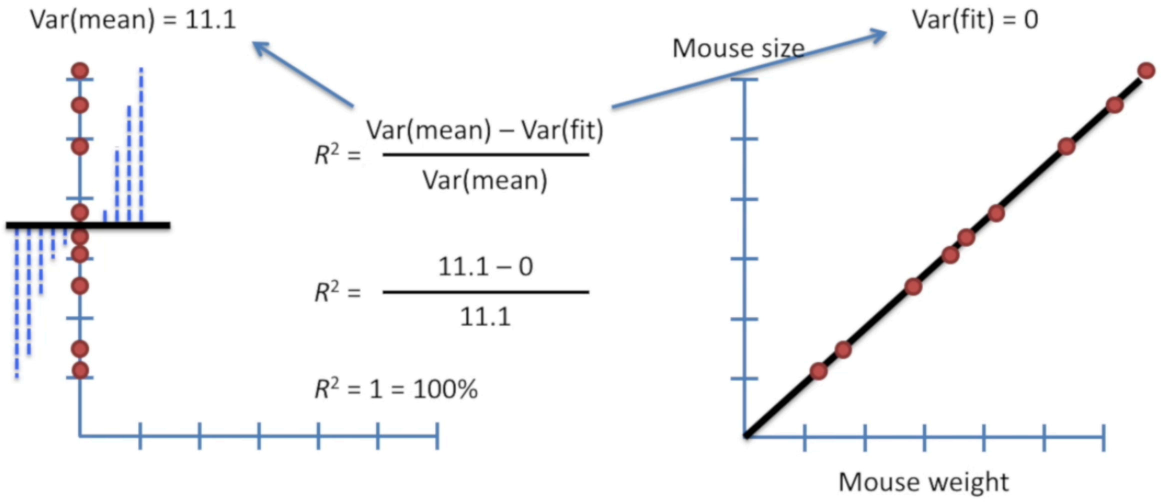
example #1

the bigger it is the better it is,

There is a 60% reduction in variance when we take the mouse weight into account.

Alternatively we can say taht mouse weight "explains" 60% of the variation in mouse size.

So now that we **know** how fit a line...



example #3

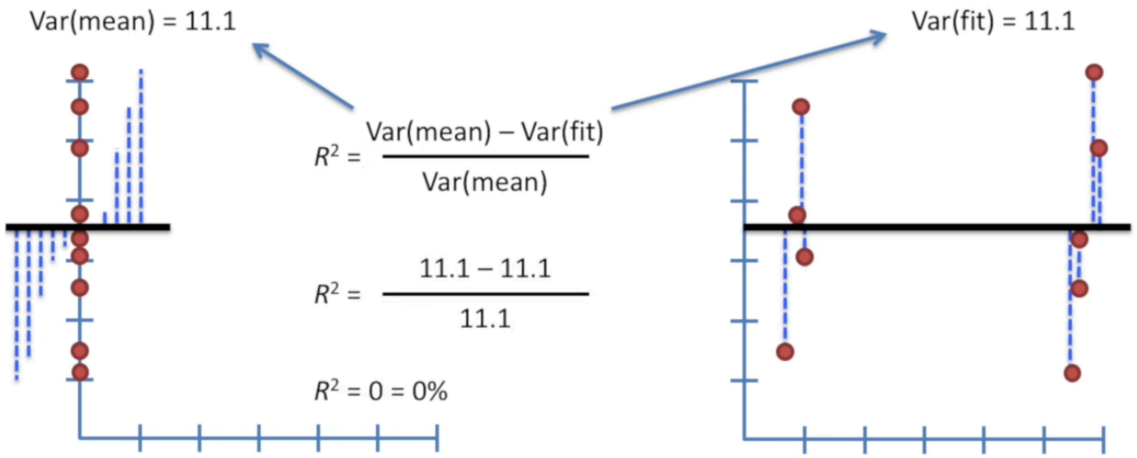
now we fit a perfect line interpolating each point.

In this case knowing mouse weight means having a perfect knowining of mouse size.

R2 is equal to 1 = 100%

mouse weight "explains" 100% of the variation in mouse size

So now that we **know** how fit a line...



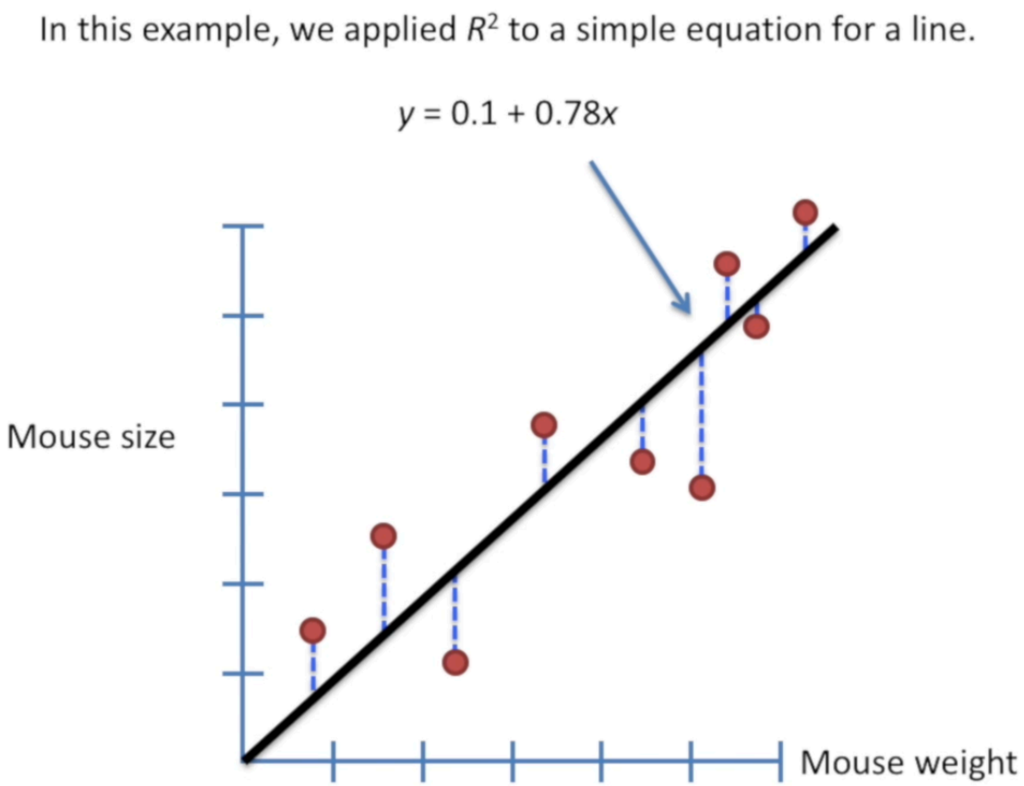
example #3

In this case knowing mouse weight does not help us predict mouse size.

As a matter of fact take a heavier mouse, this will be equally likely to be either small or big.

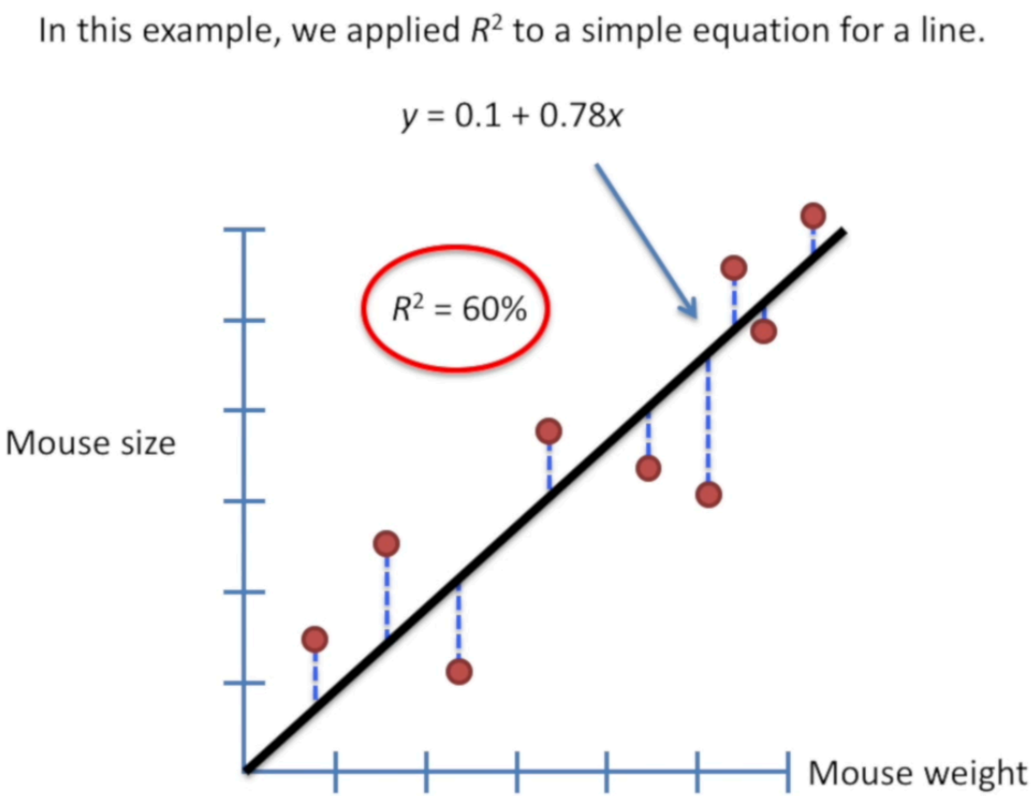
R2 is equal to 0

So now that we **know** how fit a line...



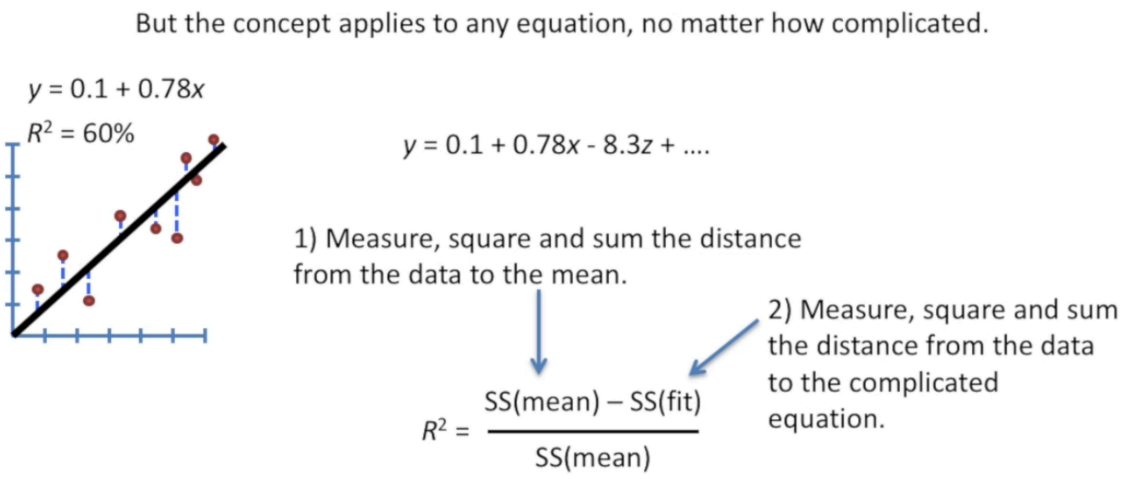
see that in eq

So now that we **know** how fit a line...



.6 (60%) R2 line fitted

So now that we **know** how fit a line…



recap: 2 passages

1. square of sum of distance to the mean
2. sum of distances from data

more than **1 regressor**

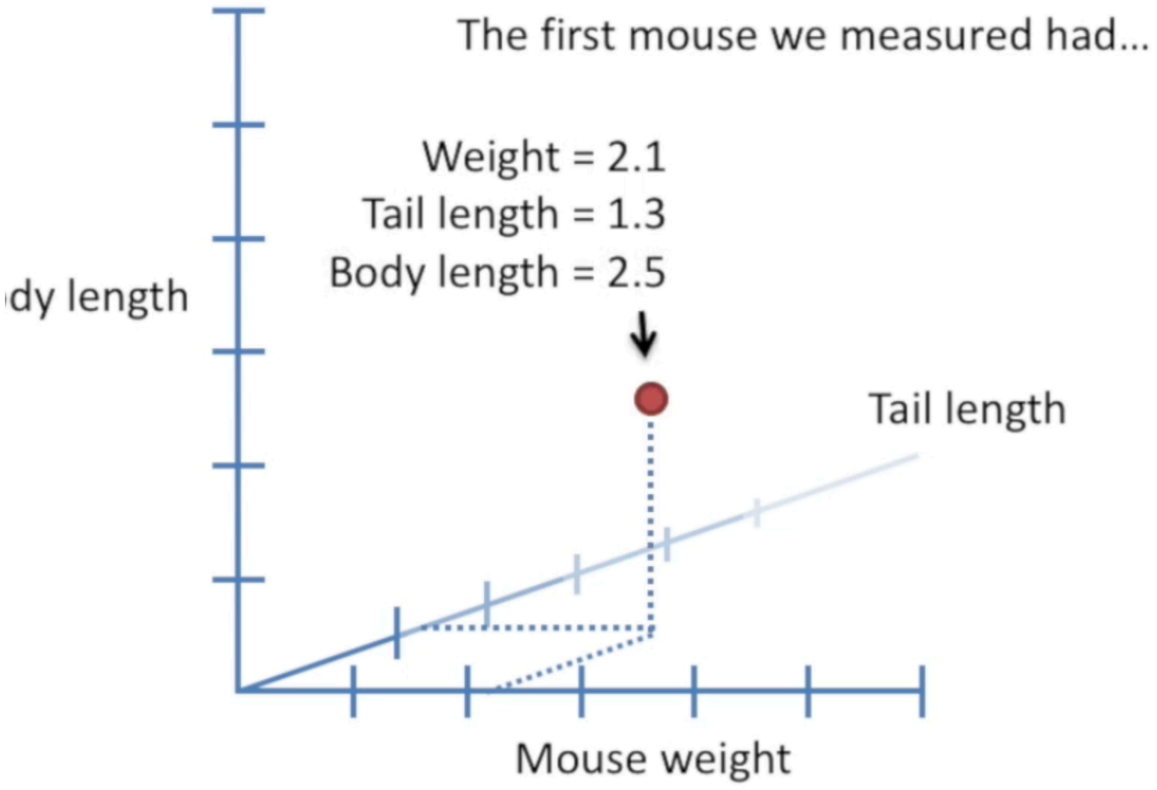
So we measured a bunch of mice...

Weight	Tail Length	Body Length
3.5	2.9	3.1
1.3	2.1	2.8
5.9	4.1	6.1
4.8	3.2	3.8
...	...	...

now we introduce “tail length”

Imagine we want to know if mouse weight adn tail length dd a good job predicting the length of the mouse body.

more than 1 regressor

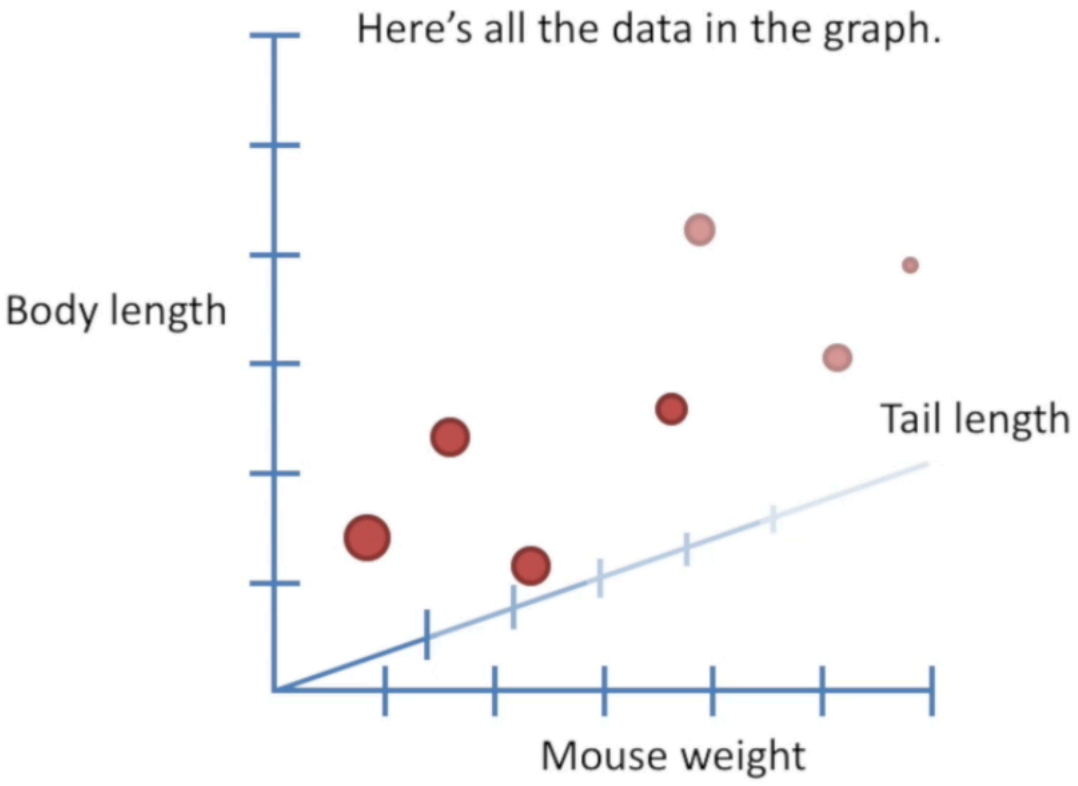


3D graph

we need a 3dimensional graph.

So we need coordinates for each of the axis.

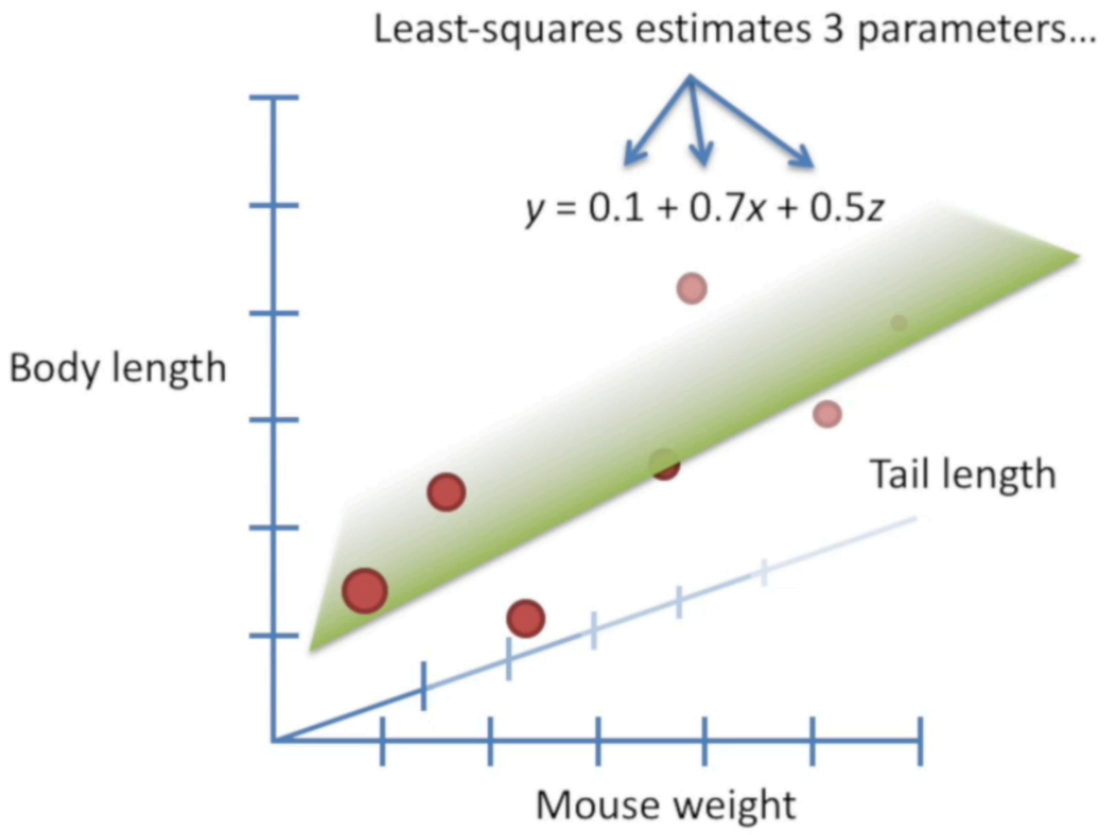
more than **1 regressor**



then data is such ...

farther points are mices with longer tails.

more than **1 regressor**

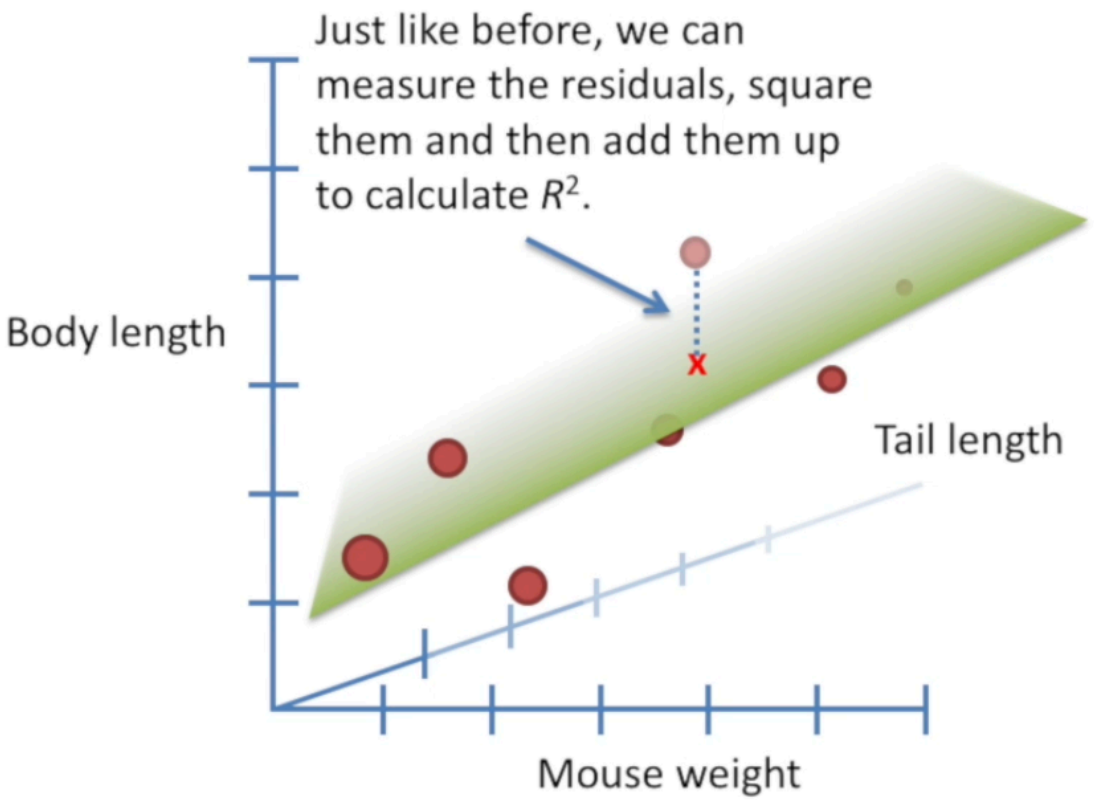


Least Square fit

since we are in a 3d settings (we are trying to predict mouse size with body length and tail length) we are no longer fitting a line, instead a **plane.**

Look at the equation.
y = “body length”
we have an intercept and 2 slopes now!

more than 1 regressor



WE DO EXACTLY AS BEFORE

We calculate the residuals a.k.a. distances from each 3d point from the plane, we square and add them together.
if the tail length is useless in predicting y (body length) then least square fit will make it equal to 0, estimating say (where z is the tail length coef.):

y = 0.1 + 0.7x + **0Z**

However what we notice is that when adding params to the equation will never make the Sum of Squaresd Fit worst (at least it is adding 0)

more than **1 regressor**

This equation…

Mouse size = 0.3 + mouse weight + flip of a coin + favorite color + astrological sign +….

The more parameters we add to the equation, the more opportunities we have for random events to reduce **SS(fit)** and result in a better R^2

in other words…

for the model defined above lets assume that for each of the regressor we assign 0 coef.

This fit will not in any case be worst than the original y (aka mouse size) = x (mouse weight)

moreover there is also a. very small probability that fillpping a coin may impact the fit, thus having a better R2.

There more silly arguments we add to the equation the more we are likely to capture some randomness.

more than **1 regressor**

This equation...

Mouse size = 0.3 + mouse weight + flip of a coin + favorite color + astrological sign +...

The more parameters we add to the equation, the more opportunities we have for random events to reduce **SS(fit)** and result in a better R^2

Thus, people report an "adjusted R^2" value that, in essence, scales R^2 by the number of parameters.

Adjusted R2

That is the reason why people have **Adjusted R2.**

This keeps into consideration the **number of parameter in the equation**.

more than **1 regressor**

$$R^2 = \frac{\text{The variation in mouse size explained by weight}}{\text{Variation in mouse size without taking weight into account}}$$

$$F = \frac{\text{The variation in mouse size explained by weight}}{\text{The variation in mouse size not explained by weight}}$$

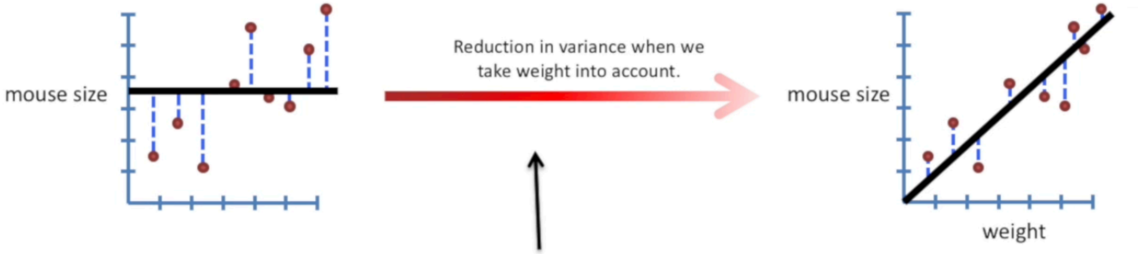
The *p*-value for R^2 comes from something called "*F*"

pvalue for R2

remember that when we add mouse weight into the equation we saw it explaining the 60% of the variation.

however let's calculate a pvalue for that we are needing this thing **F**

more than **1 regressor**

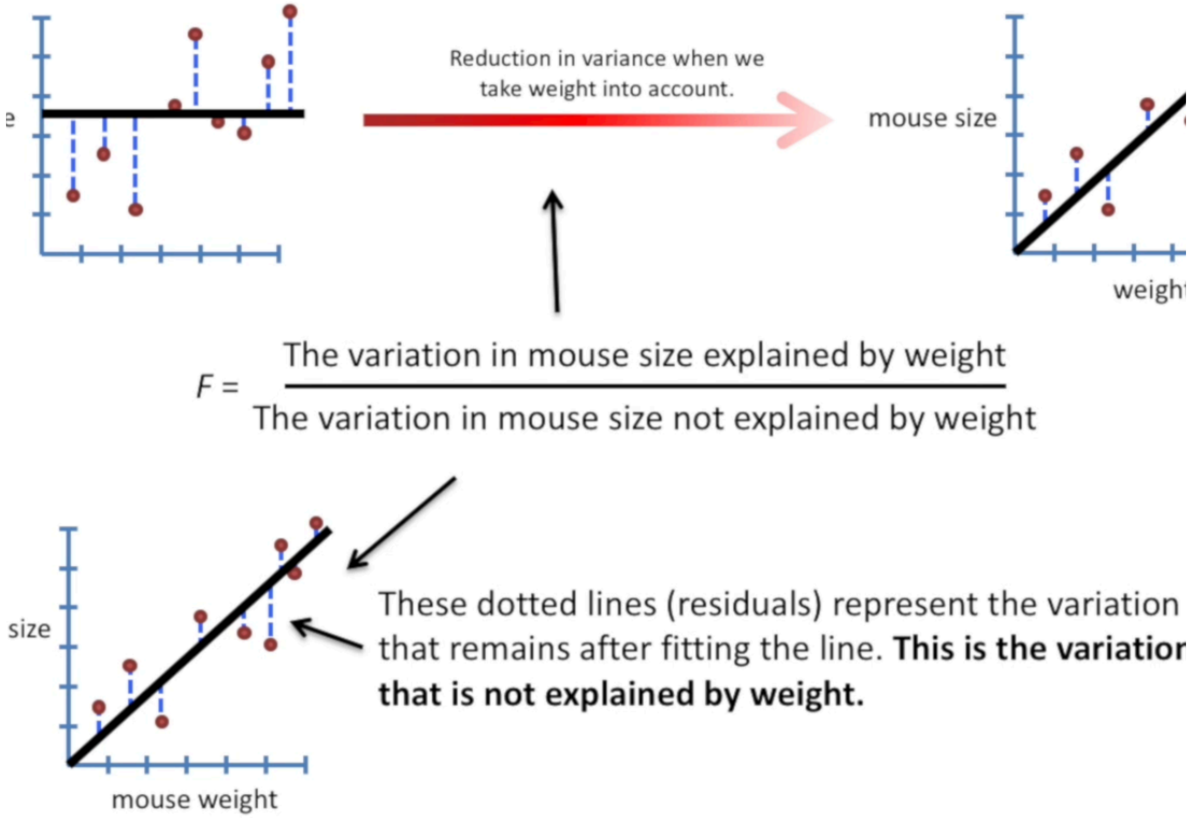


$$F = \frac{\text{The variation in mouse size explained by weight}}{\text{The variation in mouse size not explained by weight}}$$

F numerator

actually the num for **F** and **R2** is the same. taht is to say the redutcion in variance we we take mouse weight into account.

more than 1 regressor



F denominator

this is the variation not explained by weight.

more than **1 regressor**

$$R^2 = \frac{SS(mean) - SS(fit)}{SS(mean)} \qquad F = \frac{\left(SS(mean) - SS(fit)\right) / (p_{fit} - p_{mean})}{SS(fit) / (n - p_{fit})}$$

The “meat” of these equations are very similar and rely on the same “sums of squares”

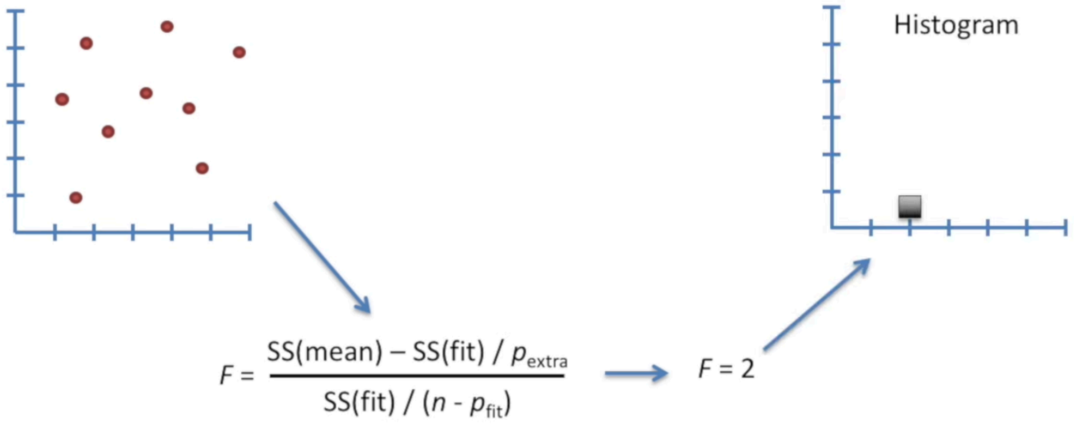
now tha math

numerators are the same, but the other terms are turning SS into variances.

pfit are the number of params in the equation
pmean is 1

just a number...

more than **1 regressor**

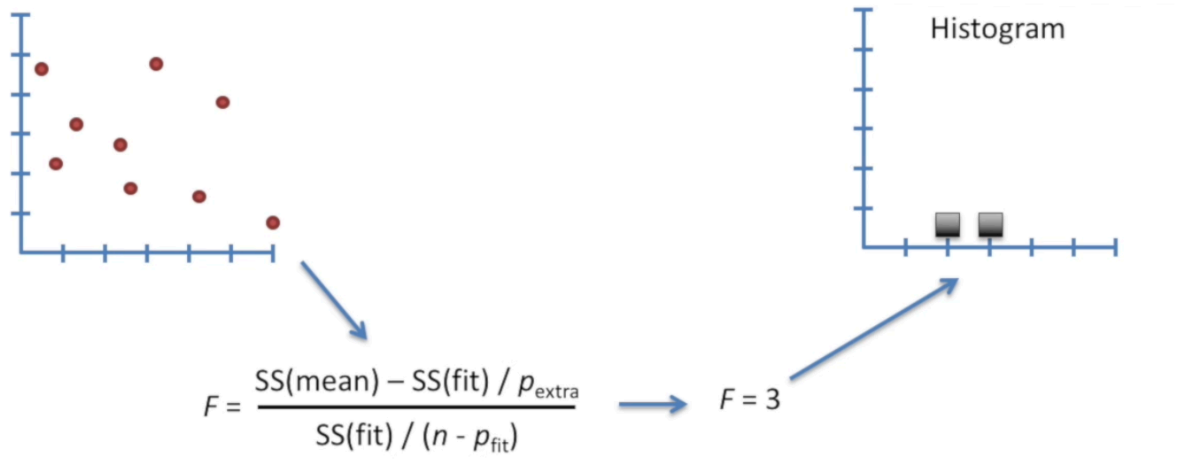


now we iterate fitting a line

we squish dots to the left and compute sum squared (SS) distances from mean. Then we compute SS from fitted line. then we **have F = 2**

Now plot F value for that iteration on an hist.

more than **1 regressor**

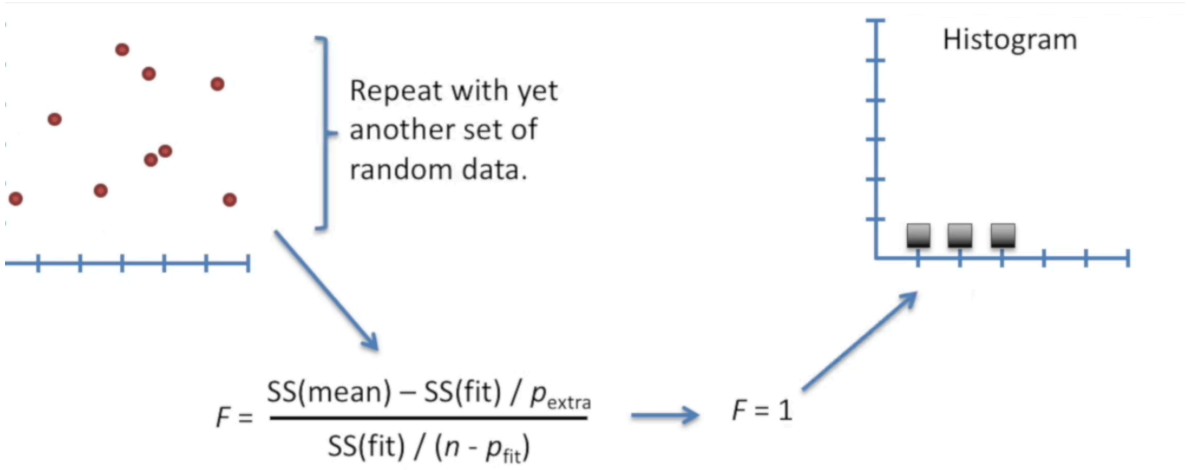


2nd iteration

we squish dots to the left and compute sum squared (SS) distances from mean. Then we compute SS from fitted line. then we **have F = 3**

then plot F om hist.

more than **1 regressor**

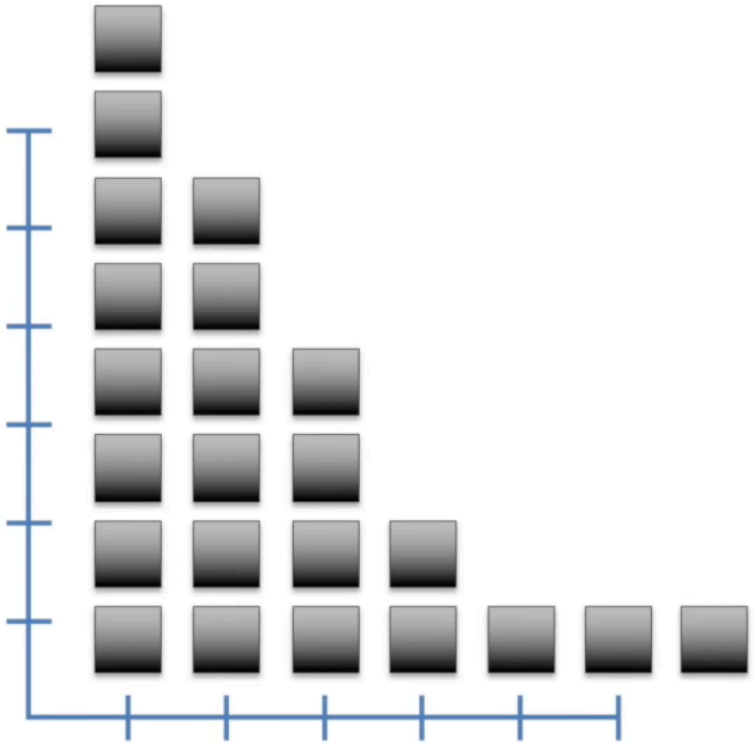


3rd iteration

we squish dots to the left and compute sum squared (SS) distances from mean. Then we compute SS from fitted line. then we **have F = 1**

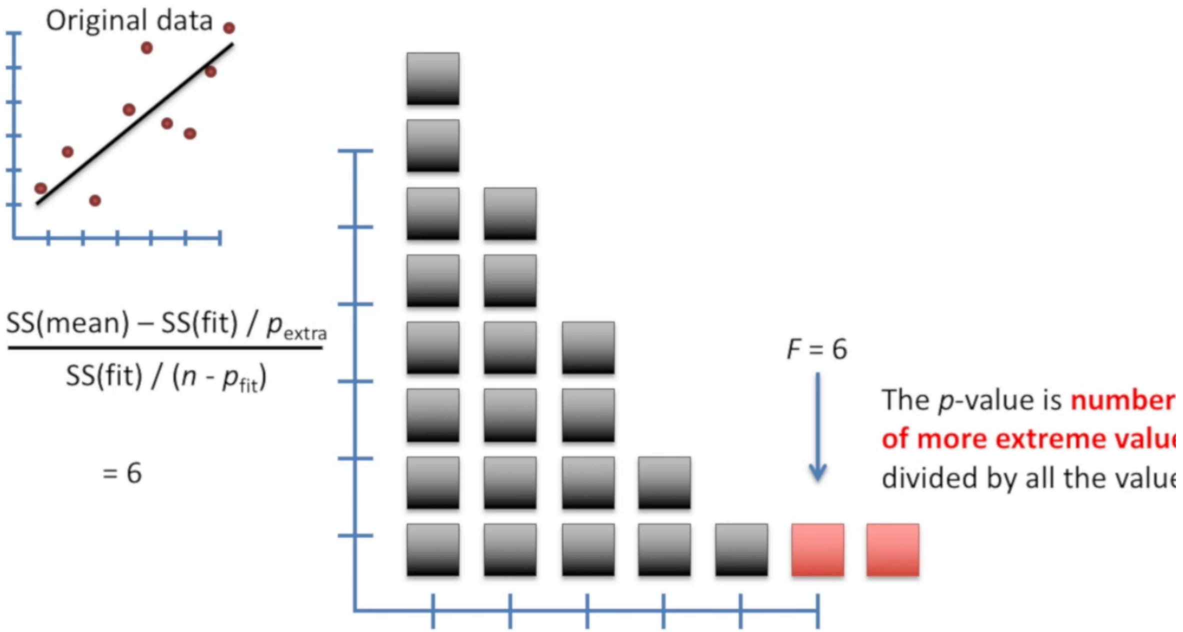
and we keep on doing that.

more than **1 regressor**



n iterations...

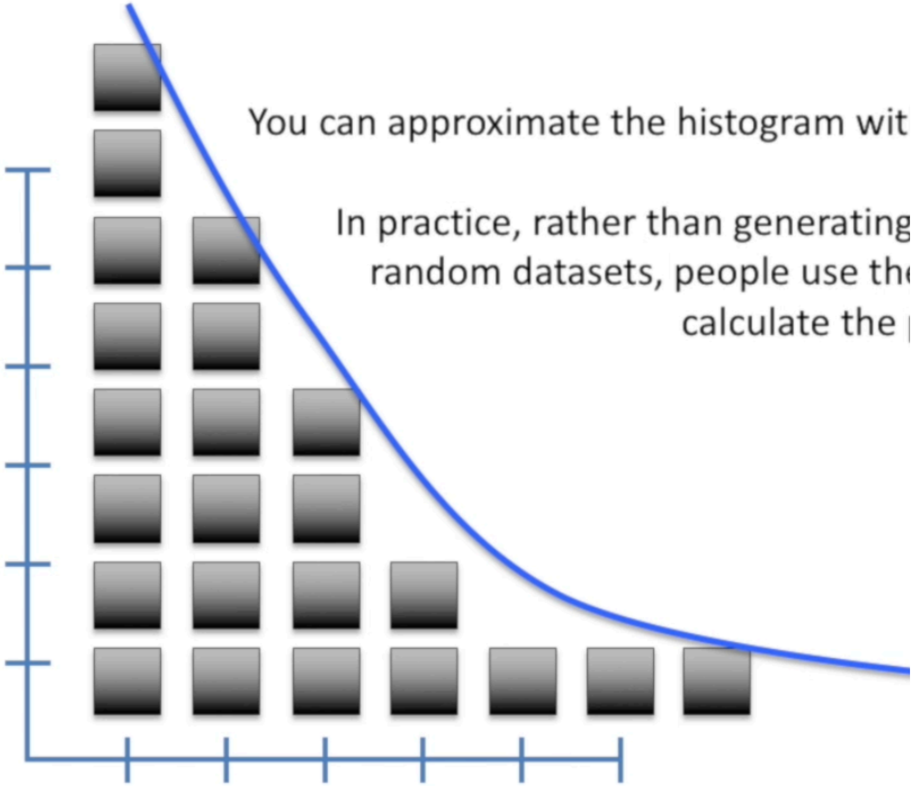
more than **1 regressor**



look at extremes

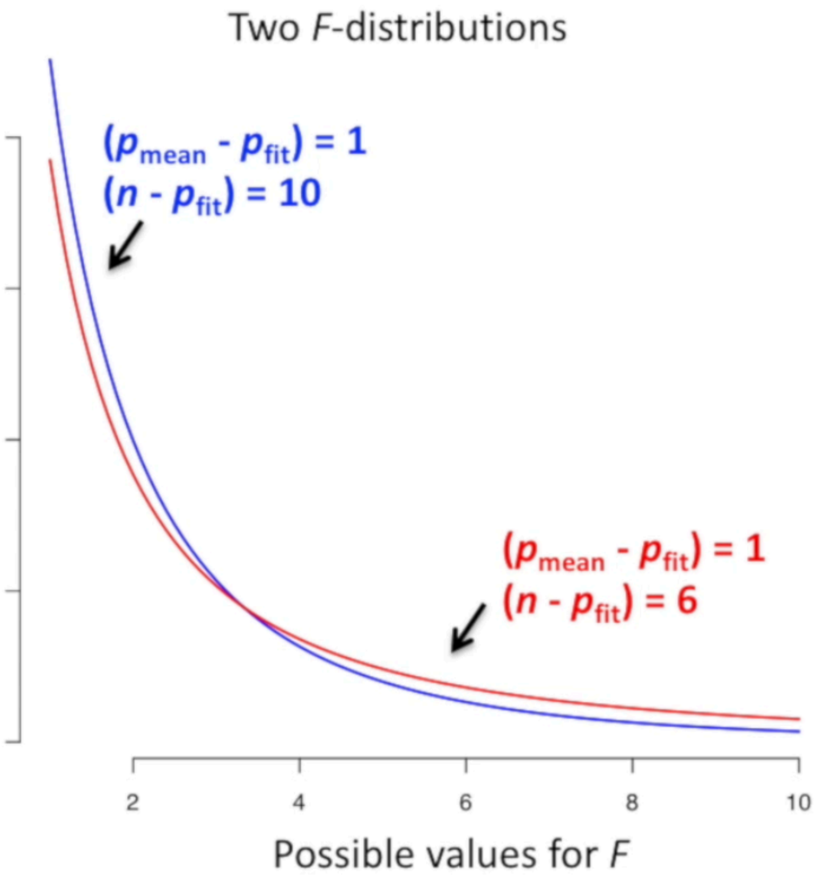
the pvalues is the number of more extremes values divided by all the values (we identify that by setting alpha confidence level).

more than 1 regressor



instead of doing n iterations you draw a line

more than **1 regressor**



pvalues for 2 samples (with same fit)

the red line represent a further F distribution and notice that it has a smaller sample size thus the distr tapers of faster.

Section 3

Linear Regression with **R**

how it is done in R

```
> mouse.data <- data.frame(
+   weight=c(0.9, 1.8, 2.4, 3.5, 3.9, 4.4, 5.1, 5.6, 6.3),
+   size=c(1.4, 2.6, 1.0, 3.7, 5.5, 3.2, 3.0, 4.9, 6.3))
> mouse.data
  weight size
1    0.9  1.4
2    1.8  2.6
3    2.4  1.0
4    3.5  3.7
5    3.9  5.5
6    4.4  3.2
7    5.1  3.0
8    5.6  4.9
9    6.3  6.3
```

generate data

At first you should import/generate data on which you need to run linear regression on. We create a dataframe with two columns (i,e. 'weight'. 'size'), this is about mouse sizes for a sample of 9 mouses.

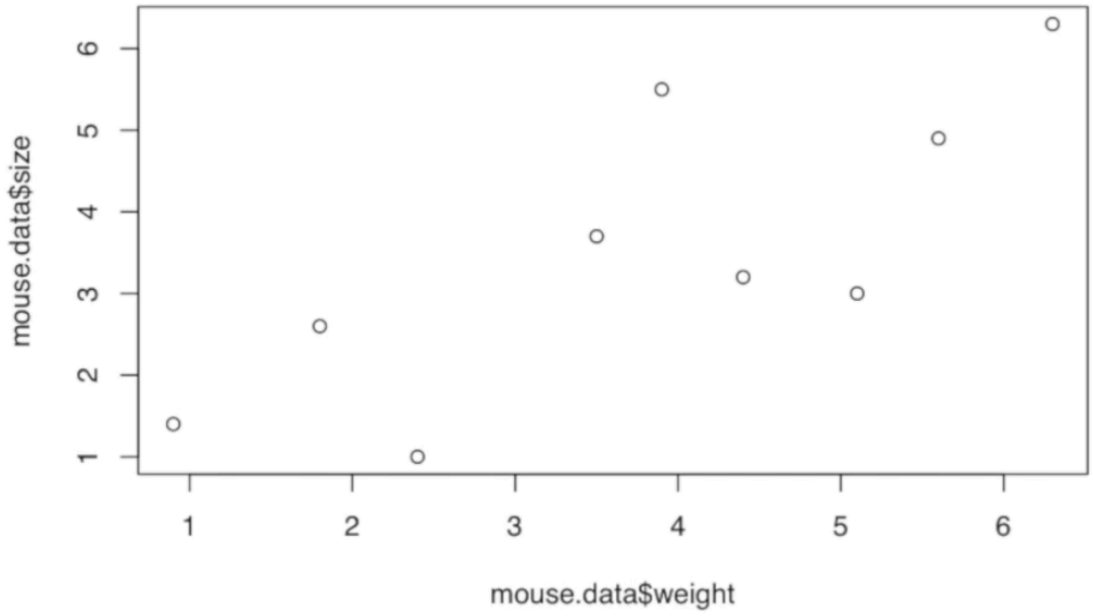
how it is done in R

```
> mouse.data <- data.frame(
+   weight=c(0.9, 1.8, 2.4, 3.5, 3.9, 4.4, 5.1, 5.6, 6.3),
+   size=c(1.4, 2.6, 1.0, 3.7, 5.5, 3.2, 3.0, 4.9, 6.3))
> mouse.data
  weight size
1    0.9  1.4
2    1.8  2.6
3    2.4  1.0
4    3.5  3.7
5    3.9  5.5
6    4.4  3.2
7    5.1  3.0
8    5.6  4.9
9    6.3  6.3
> plot(mouse.data$weight, mouse.data$size)
```

use plot()

on **x** axis we have **'size'**, on the **y** axis we have **'weight'**.

how it is done in R



plot viz

What we kinda see is that the more the size, the more the weight.

It actually makes sense isn't it?

how it is done in R

```
> mouse.data <- data.frame(
+   weight=c(0.9, 1.8, 2.4, 3.5, 3.9, 4.4, 5.1, 5.6, 6.3),
+   size=c(1.4, 2.6, 1.0, 3.7, 5.5, 3.2, 3.0, 4.9, 6.3))
> mouse.data
  weight size
1    0.9  1.4
2    1.8  2.6
3    2.4  1.0
4    3.5  3.7
5    3.9  5.5
6    4.4  3.2
7    5.1  3.0
8    5.6  4.9
9    6.3  6.3
> plot(mouse.data$weight, mouse.data$size)
> mouse.regression <- lm(size ~ weight, data=mouse.data)
```

y-values = y-intercept + slope × x-values

size = y-intercept + slope × weight

NOW we build the linear model

we call the function **lm()** which stands for "linear model", and we pass to the function the **formula** and the mouse **data**.

- *formula:* The way we specifiy the formula is that the dependent var y i.e. **'size'** stands to the left of tilde **~,** the independent vars stand to the right **'weight'** , like this y ~ x

- *data:* then you need to pass also data i.e. **'mouse.data'** within the function otherwise where is data?!

do you remember slopes and intercept?

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5482 -0.8037  0.1186  0.6186  1.8852 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   0.5813     0.9647   0.603   0.5658  
weight        0.7778     0.2334   3.332   0.0126 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.19 on 7 degrees of freedom
Multiple R-squared:  0.6133,	Adjusted R-squared:  0.558 
F-statistic:  11.1 on 1 and 7 DF,  p-value: 0.01256
```

summary of the model

this function generates all kind of outputs.

- residuals for each data point
- coefficients
- Multiple Rsquared
- etc.

Now we dig them **one** by **one**.

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)
```

The first line just prints out the original call to the lm() function.

1st section

this is the original model call, i.e. the thing you've written in the console

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5482 -0.8037  0.1186  0.6186  1.8852
```

This is a summary of the residuals (the distance from the data to the fitte line). Ideally, they should be symmetrically distributed around the line.

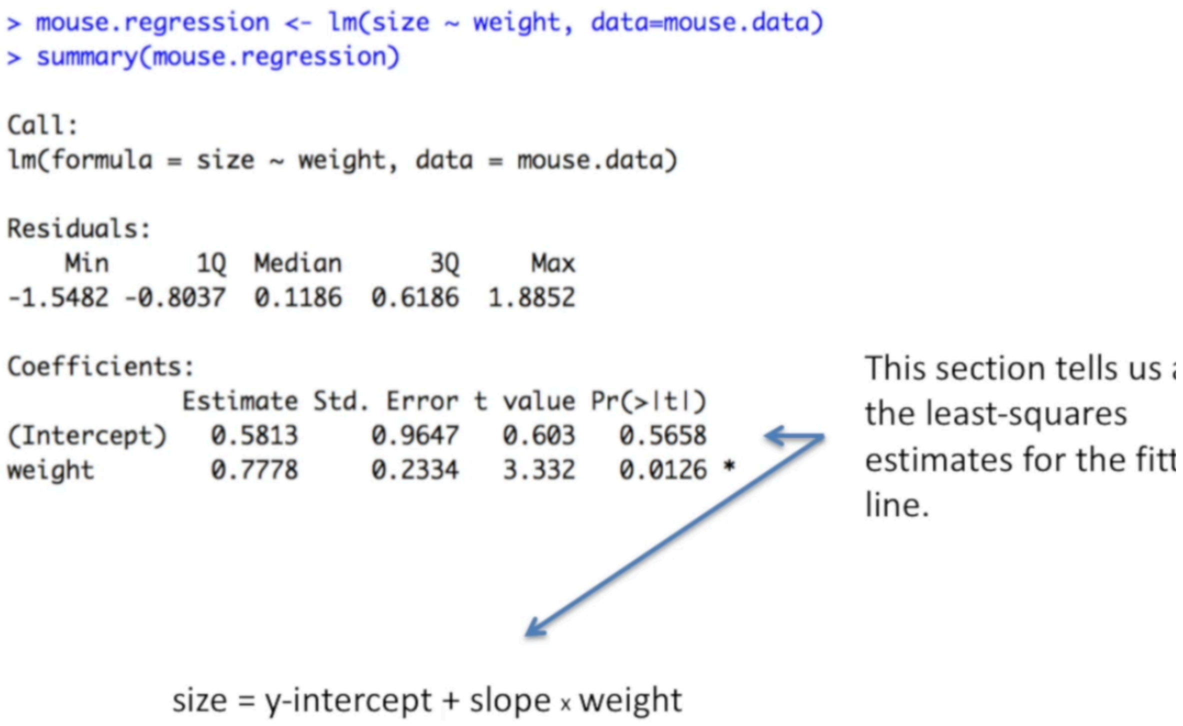
2nd section

summary of the model residuals, those are the distances from each data points to the fitted line (recall slides “fitting a line”).

Ideally those need to be symmetric, meaning the max and the min should be equally distant from 0.
Then you see 1Q (first quantile) and 3Q (third quantile)

What the heck are quantiles TA?

how it is done in R



```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5482 -0.8037  0.1186  0.6186  1.8852

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5813     0.9647   0.603   0.5658
weight        0.7778     0.2334   3.332   0.0126 *
```


3rd section

the **least square** estimate for **slope** and **intercept** i.e. a & b (recall slides from fitting a line)

Below the verbose math formulation.

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5482 -0.8037  0.1186  0.6186  1.8852 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5813     0.9647   0.603   0.5658
weight        0.7778     0.2334   3.332   0.0126 *
```

This is the value for the slope.

size = 0.5813 + 0.7778 x weight

slope & intercept

under **'Estimate Std.'** we find values for **slope** and **intercept** of the fitted linear equation.

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5482 -0.8037  0.1186  0.6186  1.8852 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   0.5813     0.9647   0.603   0.5658  
weight        0.7778     0.2334   3.332   0.0126 *
```

The standard error of the estimates and the "t value" are both provided to show you how the p-values were calculated.

std error & pvalue

those are provided to show how **pvalues** were calculated, the pvalues test wether the estimates of intercept and slope are **equal 0** against the opposite.

If they are equal to 0 they are not that useful to the model and we remove them,

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5482 -0.8037  0.1186  0.6186  1.8852

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5813     0.9647   0.603   0.5658
weight        0.7778     0.2334   3.332   0.0126 *
```

A significant p-value for weight means that it will give us a reliable guess of mouse size.

last column p-values

these are actually the p-values for the estimated intercept and slope.
Statistically speaking we are **not interested** in the intercept, so we do not really care about its p-value. Indeed we want it to have the one for **'weight'**, Moreover we want it to be less that **0.05 (5%)** significance level. That means we have a reliable coefficient estimate for the model we fitted.

right next to the pvalue we see a **star *** (you see that in next slide). By looking at the legenda we quickly understand to which level of significance the the least square estimate for slope and intercept (recall slides from fitting a line) are.

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5482 -0.8037  0.1186  0.6186  1.8852

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5813     0.9647   0.603   0.5658
weight        0.7778     0.2334   3.332   0.0126 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.19 on 7 degrees of freedom
```

This is the square root
the denominator in th
equation for *F*.

4th section res std err

this is the square root of the denominator for F.

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5482 -0.8037  0.1186  0.6186  1.8852

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5813     0.9647   0.603   0.5658
weight        0.7778     0.2334   3.332   0.0126 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.19 on 7 degrees of freedom
Multiple R-squared:  0.6133,    Adjusted R-squared:  0.558
```

5th section

this is the multiple r squared which measures how good the line fits. In other words that the weight explains the 61% of the variation inside

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5482 -0.8037  0.1186  0.6186  1.8852

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5813     0.9647   0.603   0.5658
weight        0.7778     0.2334   3.332   0.0126 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.19 on 7 degrees of freedom
Multiple R-squared:  0.6133,   Adjusted R-squared:  0.558
```

5th section *bis*

this basically what we already saw, but adjusted for the number of variables in the model.

Why need adjusting?

The more variables you insert in the model, the better the fit, up to a certain point where the model can not really assign coefficients. This penalizes that behavior.

how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5482 -0.8037  0.1186  0.6186  1.8852 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   0.5813     0.9647   0.603   0.5658  
weight        0.7778     0.2334   3.332   0.0126 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.19 on 7 degrees of freedom
Multiple R-squared:  0.6133,	Adjusted R-squared:  0.558 
F-statistic:  11.1 on 1 and 7 DF,  p-value: 0.01256
```

6th section

this line tells us if **the R squared** for the **model is significant** or not,
As we can see there are a bunch of numbers, but what we are really interested in is the **pvalue for F.**

In this case we have a reliable estimate for R, so the model.

that's it for model diagnostic...

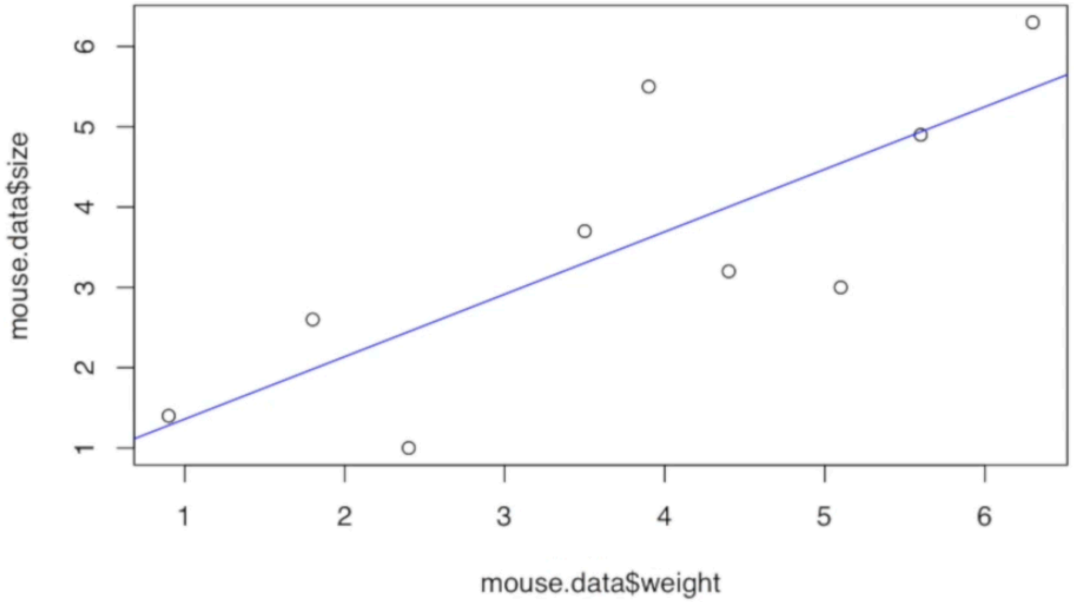
how it is done in R

```
> mouse.regression <- lm(size ~ weight, data=mouse.data)
> summary(mouse.regression)

Call:
lm(formula = size ~ weight, data = mouse.data)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5482 -0.8037  0.1186  0.6186  1.8852 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   0.5813     0.9647   0.603   0.5658  
weight        0.7778     0.2334   3.332   0.0126 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.19 on 7 degrees of freedom
Multiple R-squared:  0.6133,	Adjusted R-squared:  0.558 
F-statistic:  11.1 on 1 and 7 DF,  p-value: 0.01256

> abline(mouse.regression, col="blue")
```

Draw the line!

Now we might want to draw the line

how it is done in R



remember previous plot?

here's the line interpolating data!

Section 4

Live coding session!

PLEASE MOVE TO RSTUDIO!