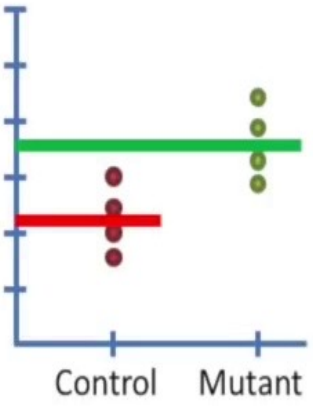
StatQuest: General Linear Models
Part 3:
Design Matrices

1	0
1	0
1	0
1	0
0	1
0	1
0	1
0	1

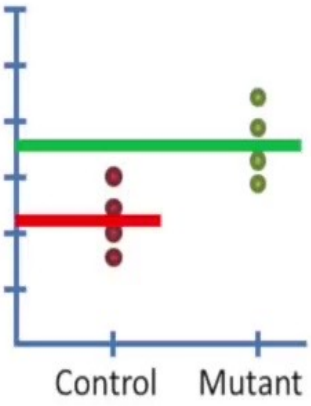
In GLMs Part 2, we ended by saying that this was not the standard design matrix for a t-test.

$y = \text{mean}_{\text{control}} + \text{mean}_{\text{mutant}}$



1	0
1	0
1	0
1	0
0	1
0	1
0	1
0	1

$y = \text{mean}_{\text{control}} + \text{mean}_{\text{mutant}}$

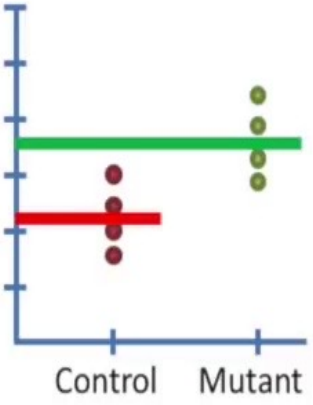


1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

This *is* the standard design matrix for a t-test.

1	0
1	0
1	0
1	0
0	1
0	1
0	1
0	1

$y = \text{mean}_{\text{control}} + \text{mean}_{\text{mutant}}$

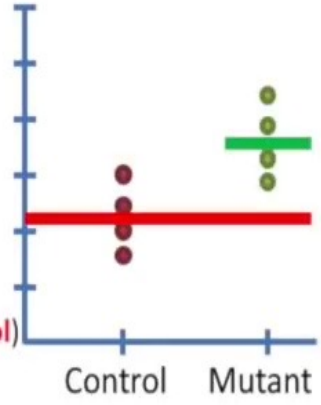


1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

This *is* the standard design matrix for a t-test.

It corresponds to a slightly different equation…

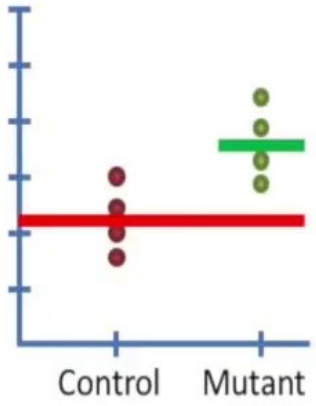
$y = \text{mean}_{\text{control}} + \text{difference}_{(\text{mutant - control})}$



1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

Let's focus on what this new design matrix and equation are all about!

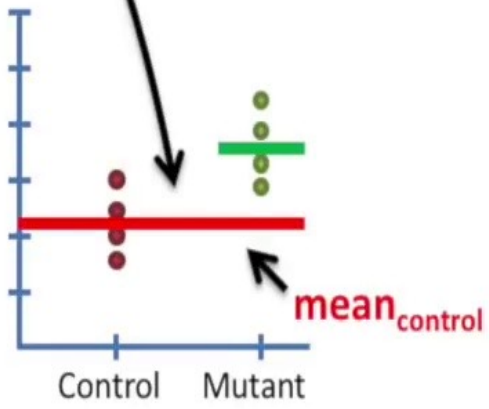
$y = \text{mean}_{\text{control}} + \text{difference}_{(\text{mutant - control})}$



In this version, all measurements, **control** and **mutant**, turn on **mean**$_{control}$

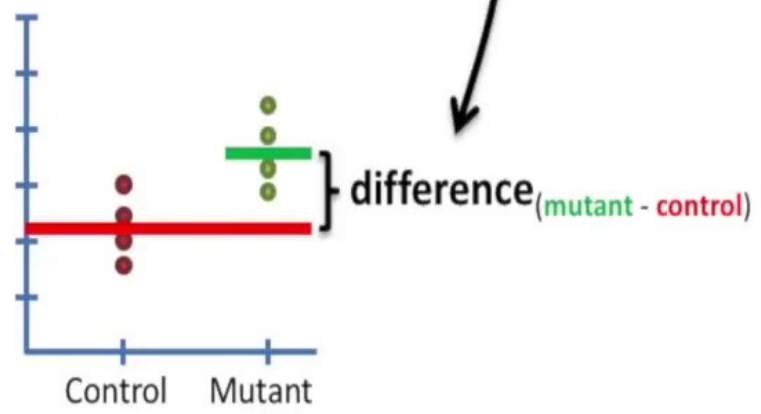
1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

y = **mean**$_{control}$ + **difference**$_{(mutant - control)}$

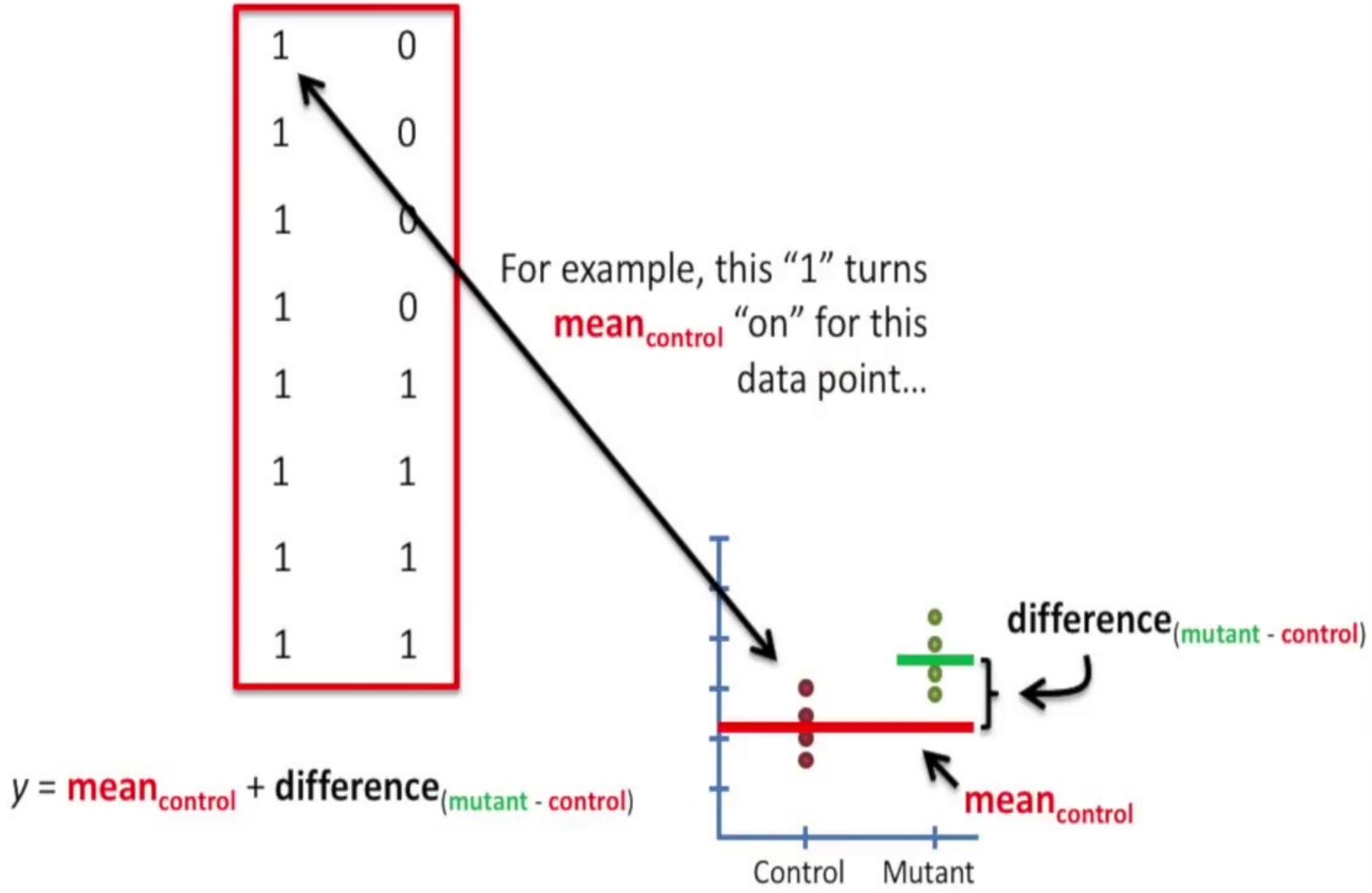


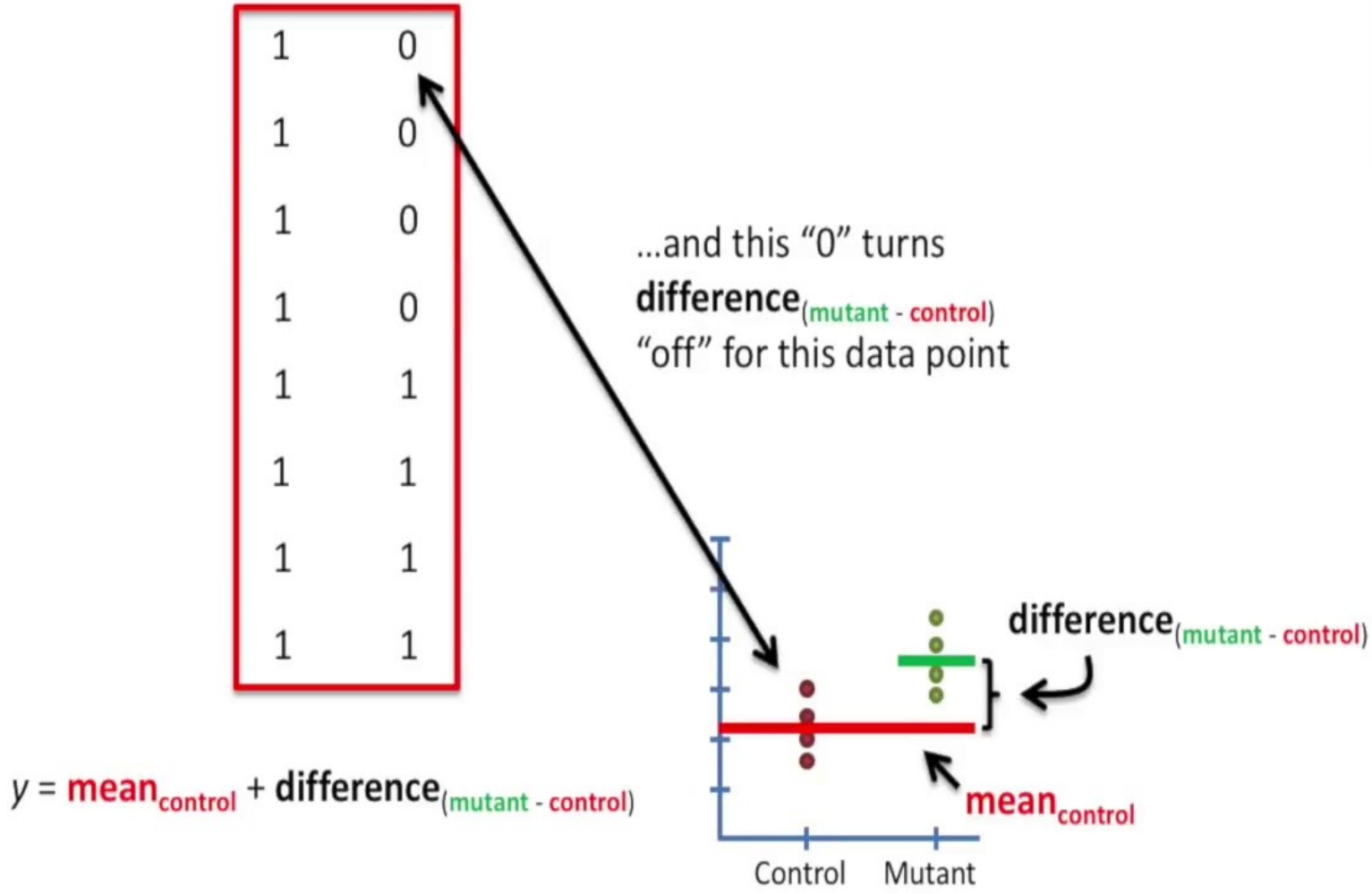
1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

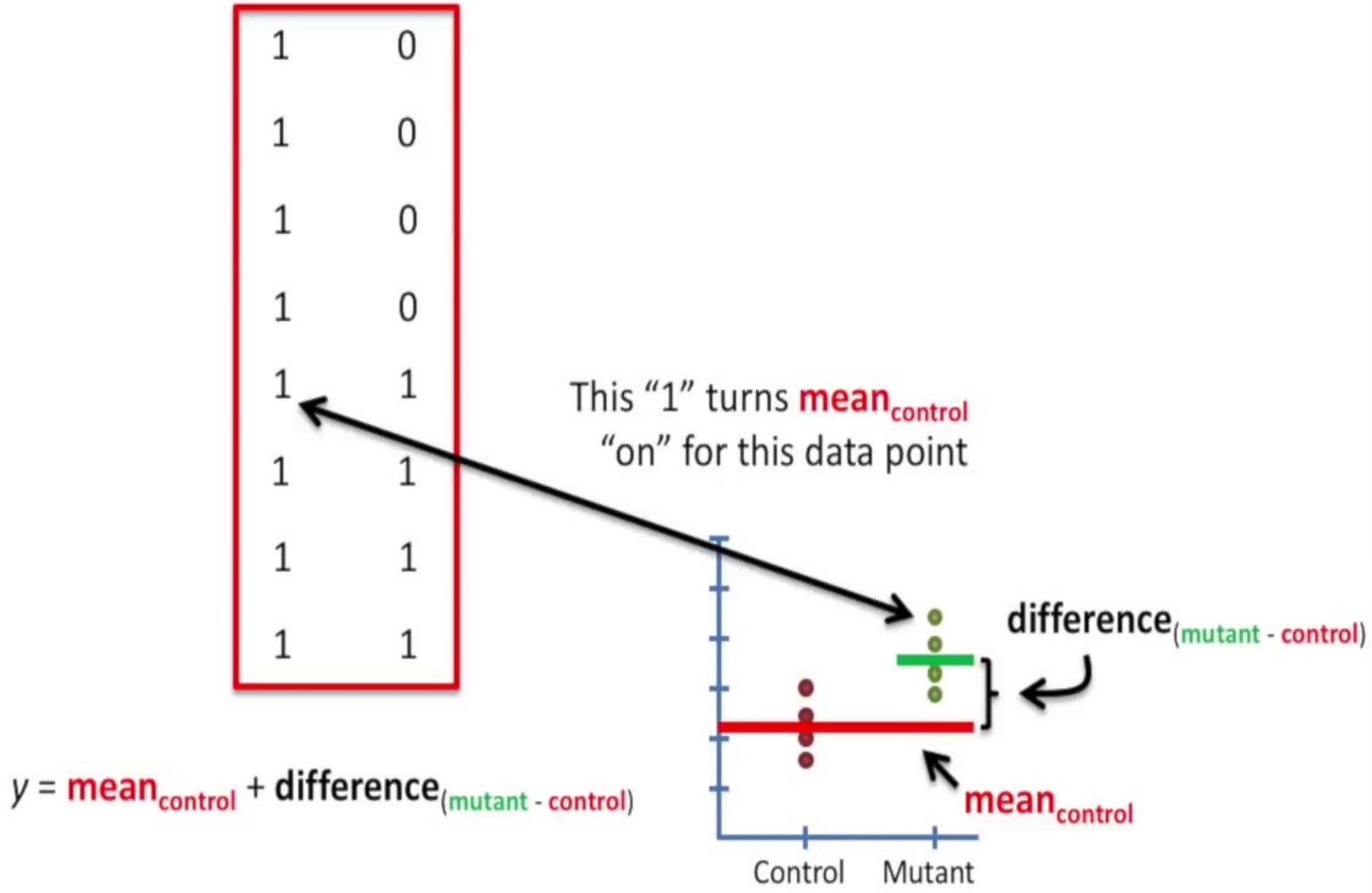
But only the mutant measurements turn on **difference**$_{(\text{mutant - control})}$

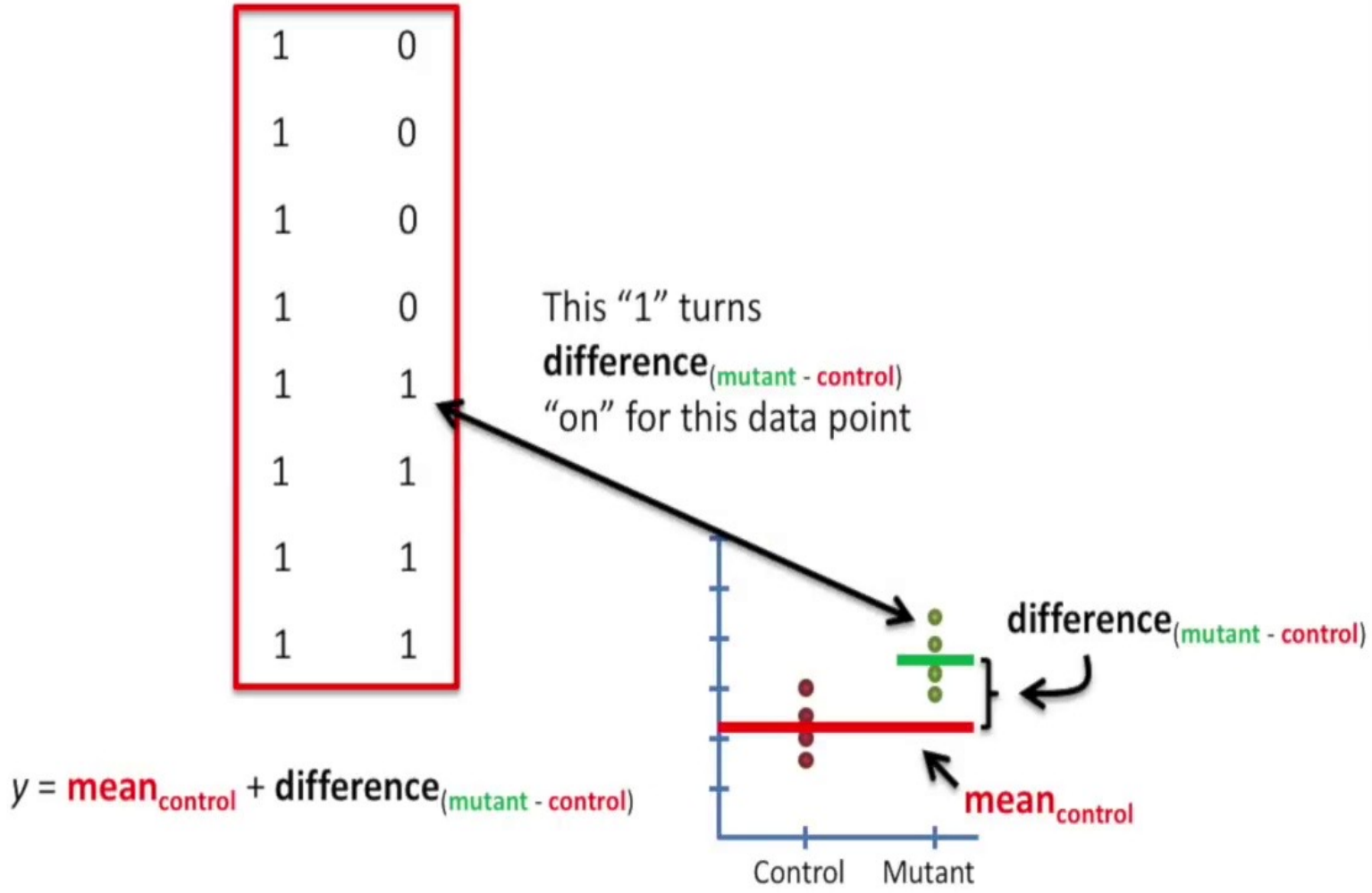


$y = \text{mean}_{\text{control}} + \text{difference}_{(\text{mutant - control})}$

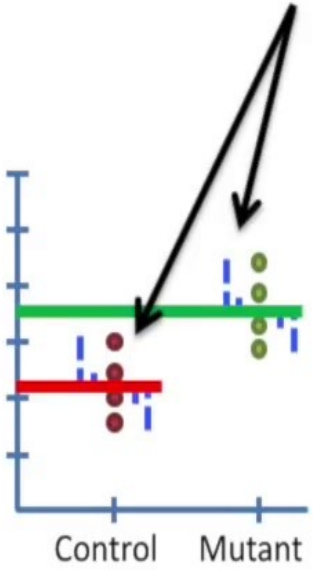

1 0
1 0
1 0
1 0
1 1
1 1
1 1
1 1
For example, this "1" turns mean control "on" for this data point...
difference (mutant - control)
mean control
Control
Mutant
y = mean control + difference (mutant - control)

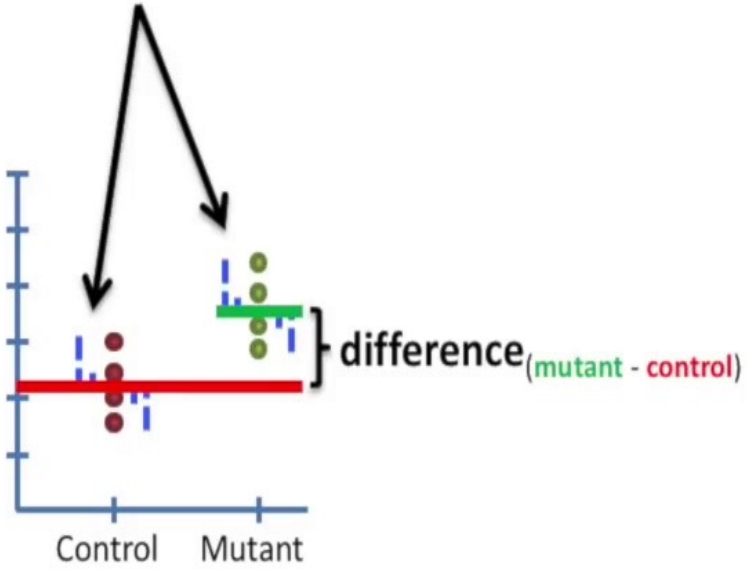

1 0
1 0
1 0
1 0
1 1
1 1
1 1
1 1
...and this "0" turns
difference(mutant - control)
"off" for this data point
difference(mutant - control)
meancontrol
Control
Mutant
y = meancontrol + difference(mutant - control)


1 0
1 0
1 0
1 0
1 1
1 1
1 1
1 1
This "1" turns mean control "on" for this data point
difference (mutant - control)
mean control
Control
Mutant
y = mean control + difference (mutant - control)


1 0
1 0
1 0
1 0
1 1
1 1
1 1
1 1
This “1” turns
difference(mutant - control)
“on” for this data point
difference(mutant - control)
mean control
Control
Mutant
y = mean control + difference(mutant - control)

The residuals are the same for both equations (and design matrices).



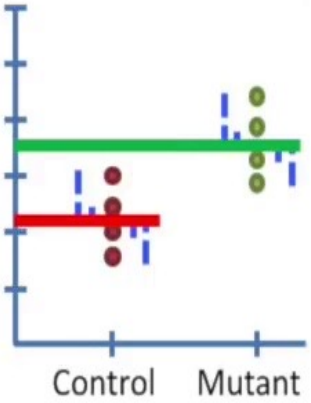


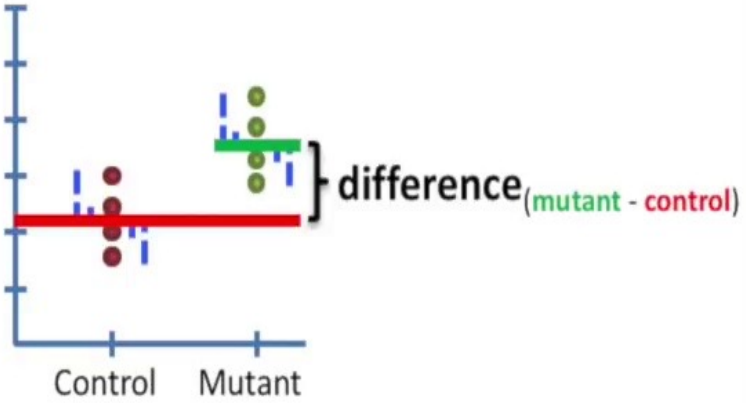
$y = \text{mean}_{\text{control}} + \text{mean}_{\text{mutant}}$

$y = \text{mean}_{\text{control}} + \text{difference}_{(\text{mutant - control})}$

The residuals are the same for both equations (and design matrices).

The equations also have the same number of parameters, 2, so p_{fit} is the same.



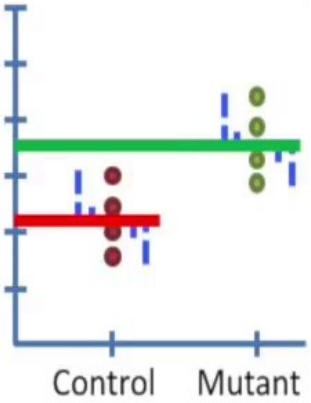


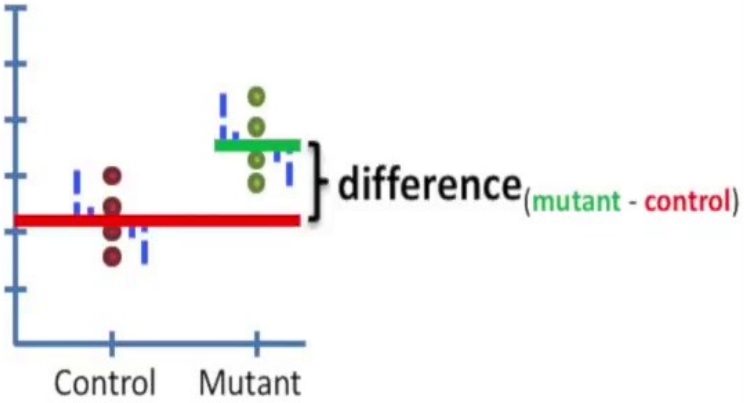
$y = \text{mean}_{control} + \text{mean}_{mutant}$

$y = \text{mean}_{control} + \text{difference}_{(mutant - control)}$

The residuals are the same for both equations (and design matrices).

The equations also have the same number of parameters, 2, so p_{fit} is the same.





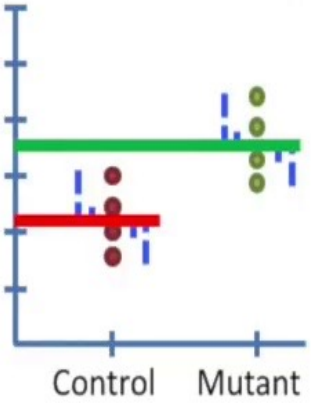
$y = \text{mean}_{control} + \text{mean}_{mutant}$

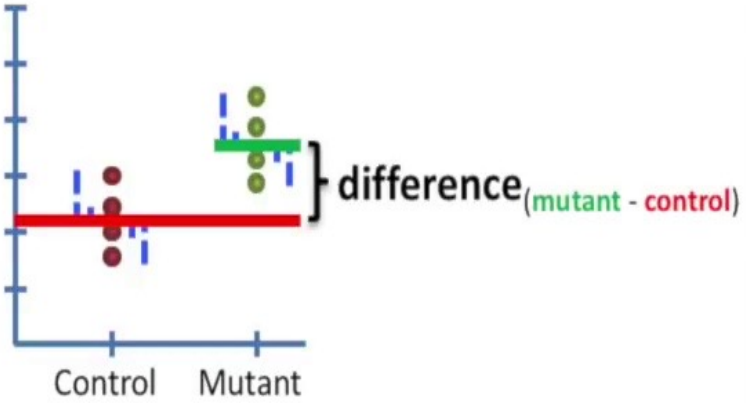
2 parameters

$y = \text{mean}_{control} + \text{difference}_{(mutant - control)}$

The residuals are the same for both equations (and design matrices).

The equations also have the same number of parameters, 2, so p_{fit} is the same.





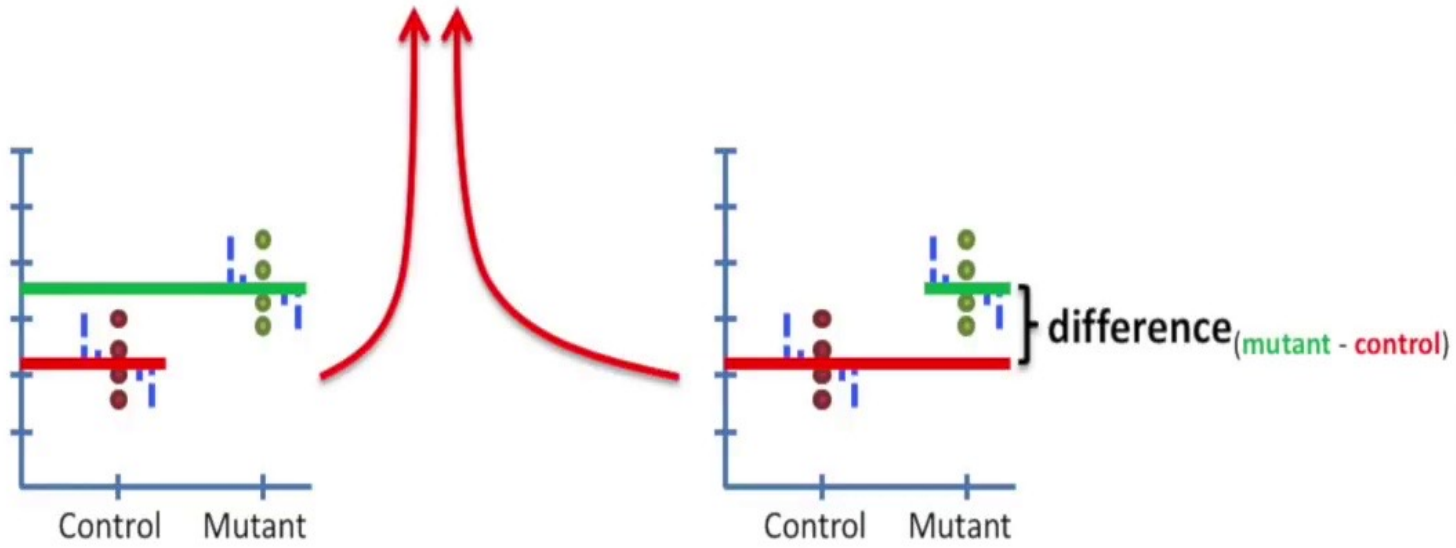
$y = \text{mean}_{control} + \text{mean}_{mutant}$

2 parameters

$y = \text{mean}_{control} + \text{difference}_{(mutant - control)}$

2 parameters

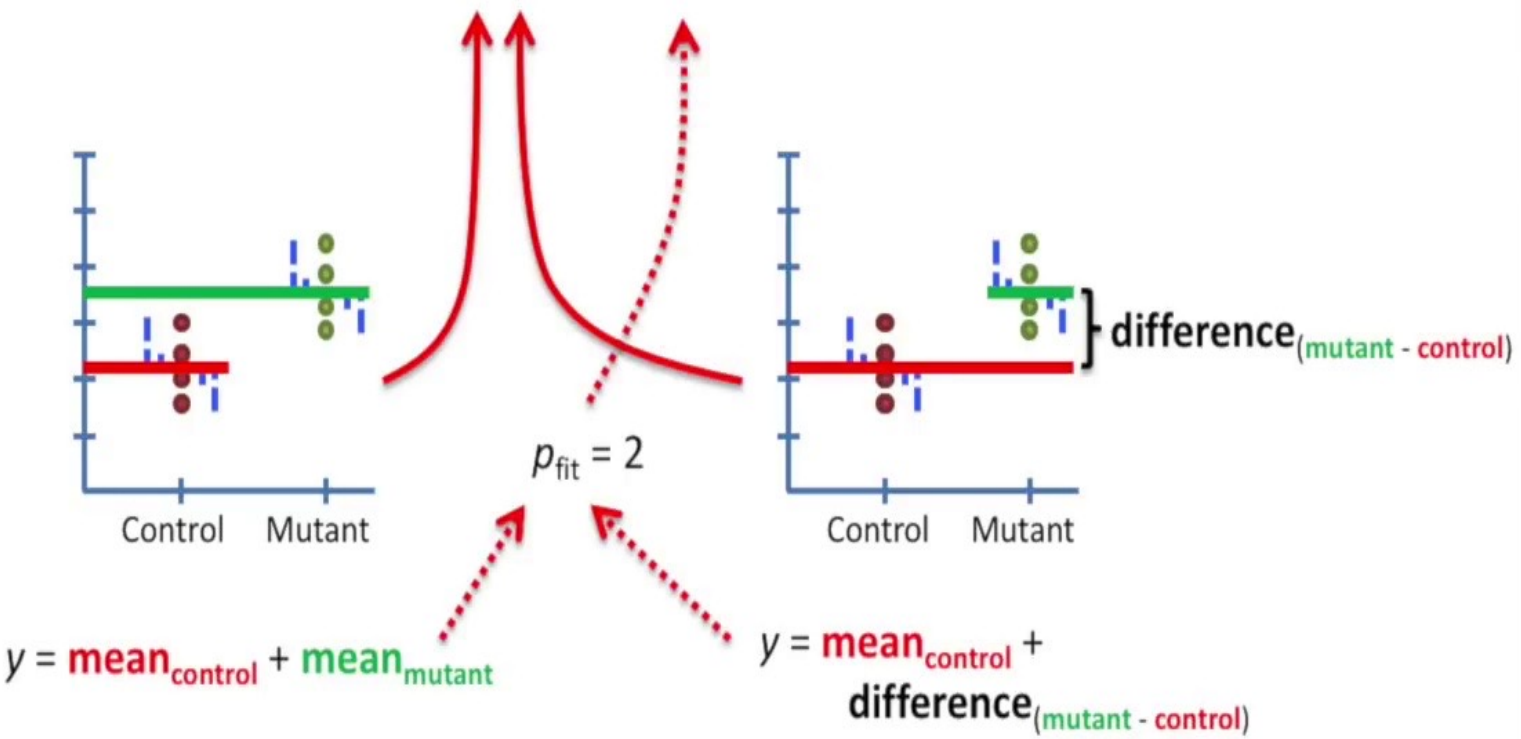
$$F = \frac{SS(mean) - SS(fit) / (p_{fit} - p_{mean})}{SS(fit) / (n - p_{fit})}$$



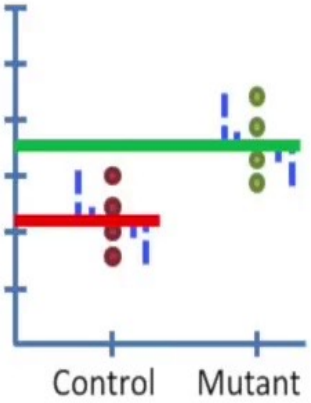
$y = \text{mean}_{control} + \text{mean}_{mutant}$

$y = \text{mean}_{control} + \text{difference}_{(mutant - control)}$

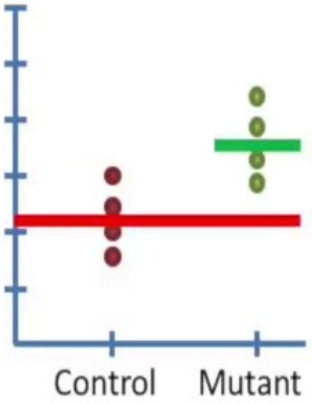
$$F = \frac{SS(mean) - SS(fit) / (p_{fit} - p_{mean})}{SS(fit) / (n - p_{fit})}$$



If they both do the same thing and result in the same p-value,
why is the one on the right more common?



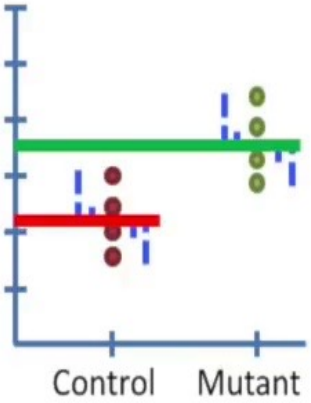
$y = \text{mean}_{\text{control}} + \text{mean}_{\text{mutant}}$

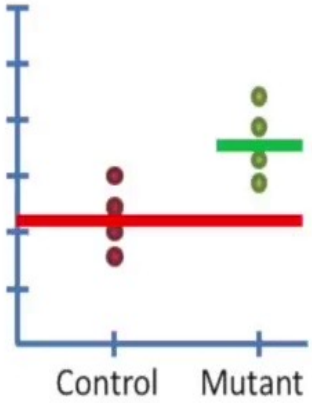


$y = \text{mean}_{\text{control}} + \text{difference}_{(\text{mutant - control})}$

If they both do the same thing and result in the same p-value, why is the one on the right more common?

I'll be honest, I don't know the answer for sure, but I think it has something to do with regression...





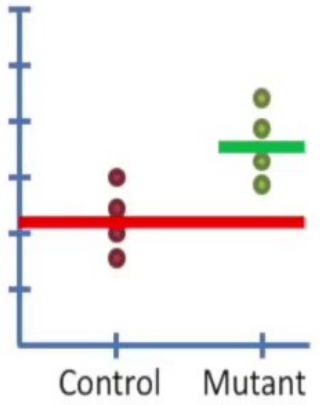
$y = \text{mean}_{\text{control}} + \text{mean}_{\text{mutant}}$

$y = \text{mean}_{\text{control}} + \text{difference}_{(\text{mutant - control})}$

$y = \mathbf{mean}_{\text{control}} + \mathbf{difference}_{(\text{mutant - control})}$

So far, we've looked at design matrices in the context of using 1's and 0's to turn parts of the equation "on" or "off"...

1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

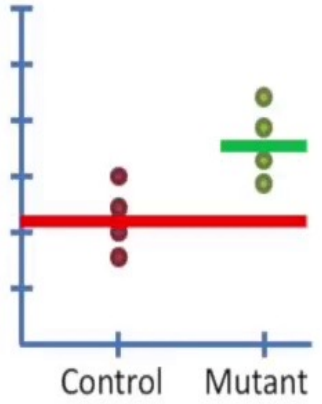


$y = \mathbf{mean}_{\text{control}} + \mathbf{difference}_{(\text{mutant - control})}$

So far, we've looked at design matrices in the context of using 1's and 0's to turn parts of the equation "on" or "off"...

...so let's take a step back and remember how it works.

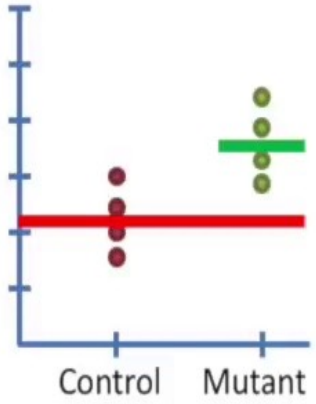
1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1



$y = 1 \times \mathbf{mean}_{\text{control}} + 0 \times \mathbf{difference}_{(\text{mutant - control})}$

Remember that the numbers in the first column are multiplied by $\mathbf{mean}_{\text{control}}$

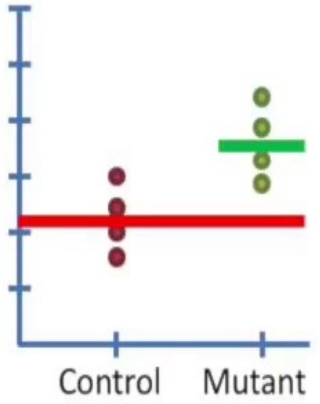
1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1



$y = 1 \times \mathbf{mean}_{\text{control}} + 0 \times \mathbf{difference}_{(\text{mutant - control})}$

...and the numbers in the second column are multiplied by **difference**$_{(\text{mutant - control})}$

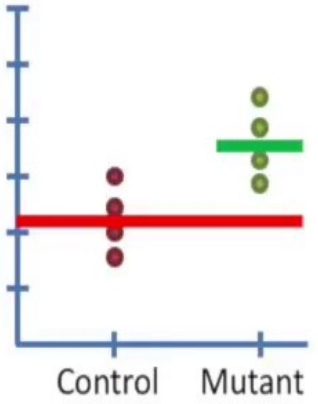
1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1



$y = 1 \times \mathbf{mean}_{\text{control}} + 0 \times \mathbf{difference}_{(\text{mutant - control})}$

Multiplying $\mathbf{mean}_{\text{control}}$ by 1 “turns it on” by just letting it be.

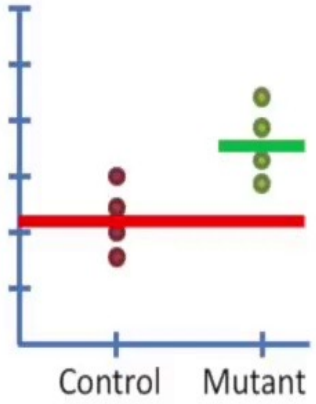
1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1



$y = 1 \times \mathbf{mean}_{\text{control}} + 0 \times \mathbf{difference}_{(\text{mutant - control})}$

Multiplying
difference$_{(\text{mutant - control})}$
by 0 makes it 0 and that
"turns it off".

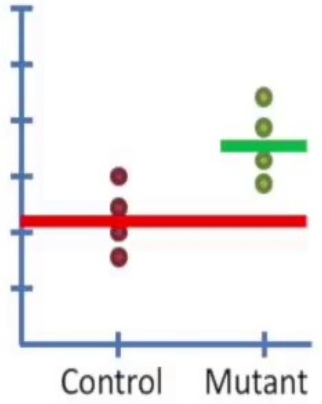
1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1



$y = 1 \times \mathbf{mean}_{\text{control}} + 0 \times \mathbf{difference}_{(\text{mutant - control})}$

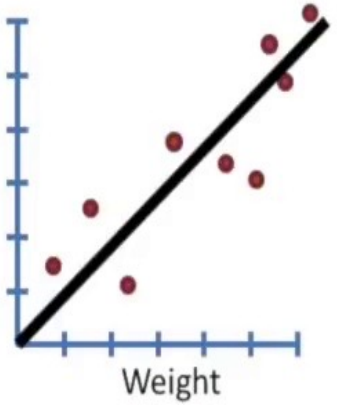
A design matrix full of 1's and 0's is perfect for doing t-tests or ANOVAs - any time we have different categories of data - but we can use other numbers.

1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1



For example, here's a design matrix for linear regression.

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

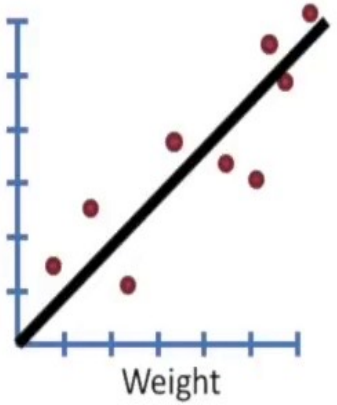


y = **y-intercept** + **slope**

For example, here's a design matrix for linear regression.

It pairs with this equation.

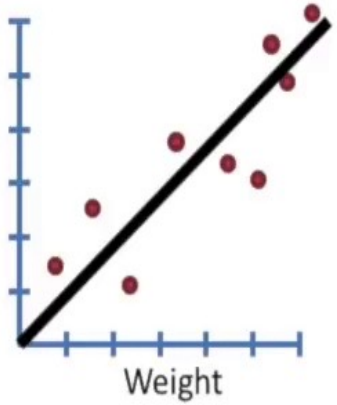
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



y = **y-intercept** + **slope**

We've got a bunch of 1s in the first column...

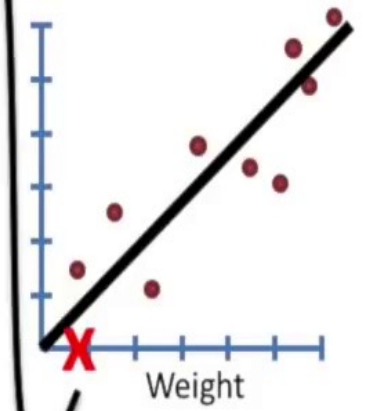
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



y = **y-intercept** + **slope**

And in the second column, we've got the x-axis position for each point...

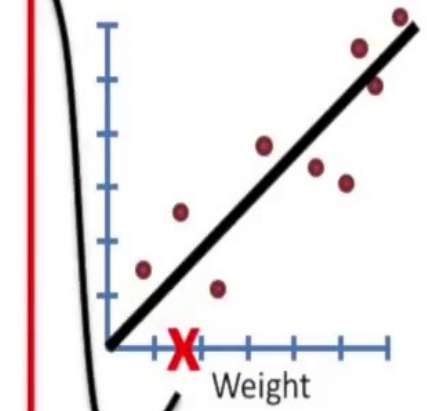
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



y = **y-intercept** + **slope**

And in the second column, we've got the x-axis position for each point...

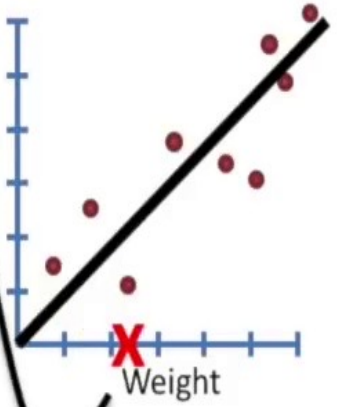
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



y = **y-intercept** + **slope**

And in the second column, we've got the x-axis position for each point...

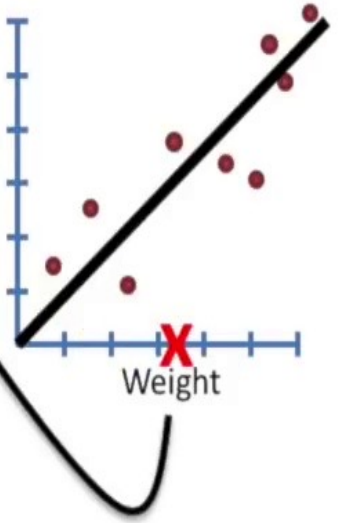
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



y = **y-intercept** + **slope**

And in the second column, we've got the x-axis position for each point...

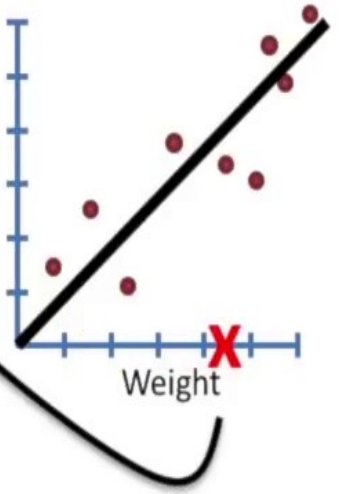
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



y = **y-intercept** + **slope**

And in the second column, we've got the x-axis position for each point...

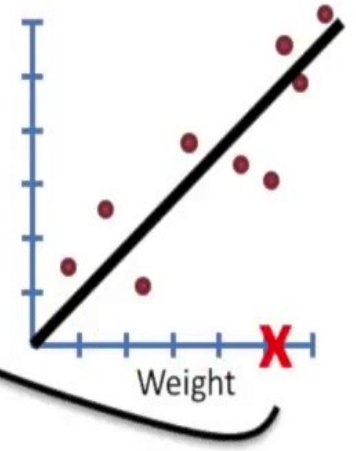
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



y = **y-intercept** + **slope**

And in the second column, we've got the x-axis position for each point...

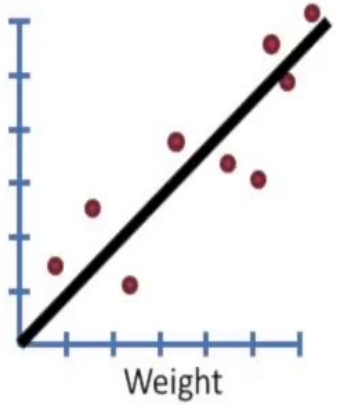
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



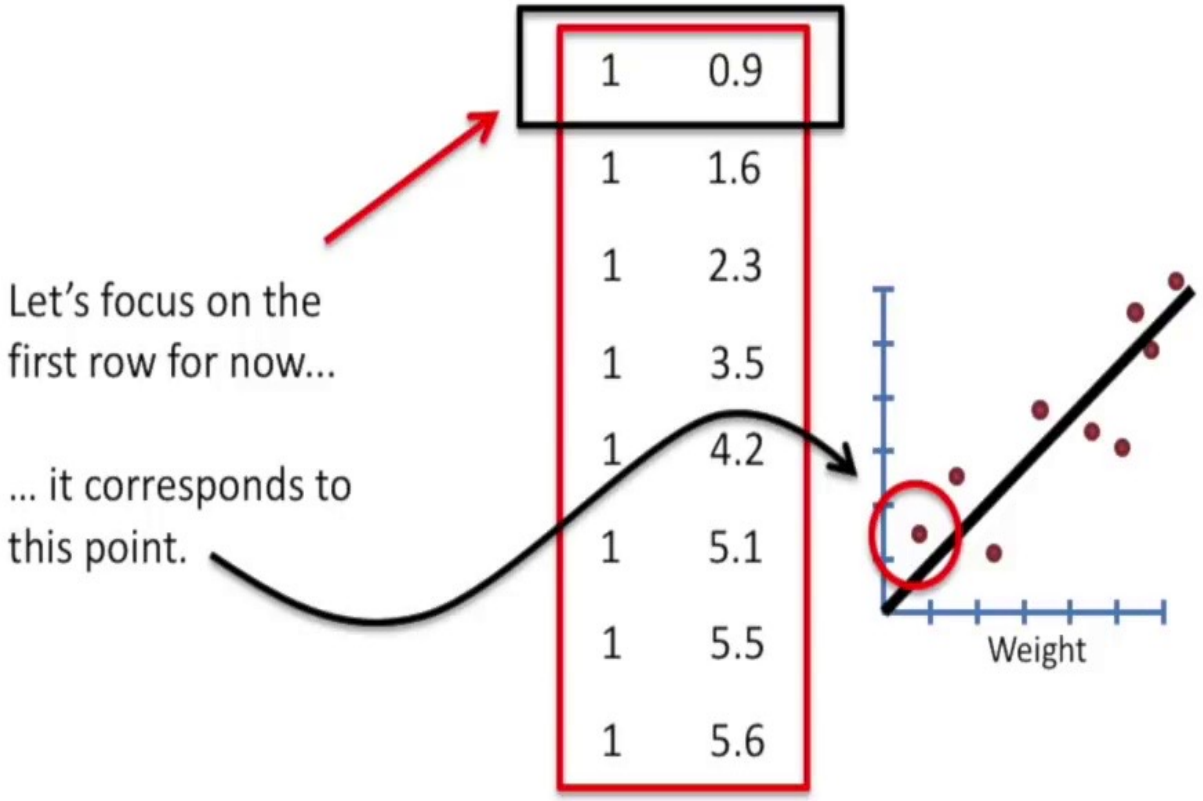
y = **y-intercept** + **slope**

Let's focus on the first row for now...

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



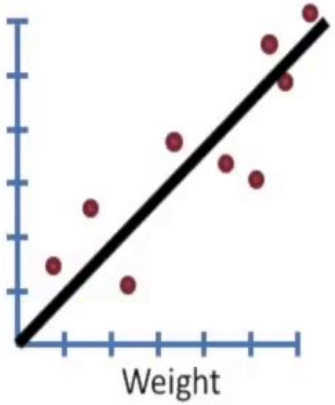
y = **y-intercept** + **slope**



$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

Just like before, the numbers in the first column multiply the first term in the formula.

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

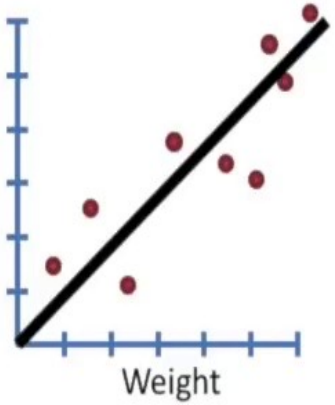


$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

Just like before, the numbers in the first column multiply the first term in the formula.

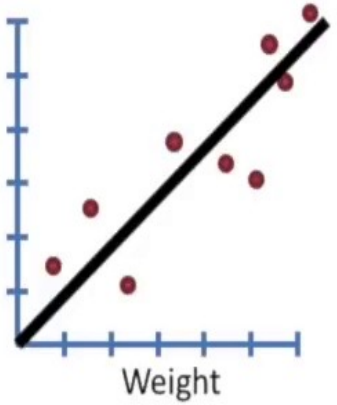
In this case, multiplying the **y-intercept** by 1 turns it "on."



$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

And just like before, the numbers in the second column multiply the second term in the formula.

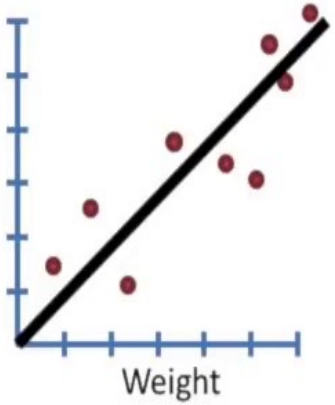


$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

And just like before, the numbers in the second column multiply the second term in the formula.

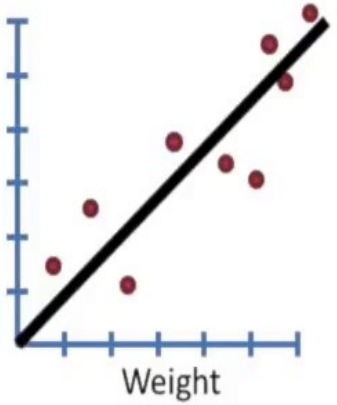
In this case, we are scaling the term for the **slope**.



$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

To make this more concrete, let's see what happens when we use real numbers for the **y-intercept** and **slope**...

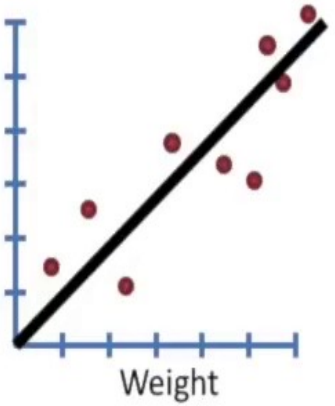
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

The **y-intercept** is super small and = 0.01

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

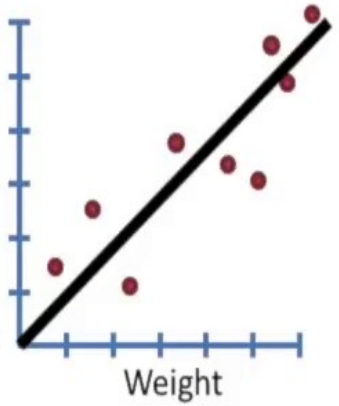


$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

The **y-intercept** is super small and = 0.01

$y = 1 \times$ **0.01** $+ 0.9 \times$ **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

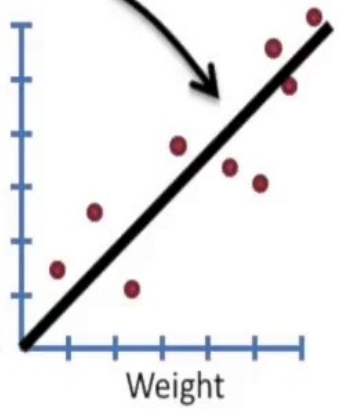


$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

The **slope** = 0.8

$y = 1 \times$ **0.01** $+ 0.9 \times$ **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

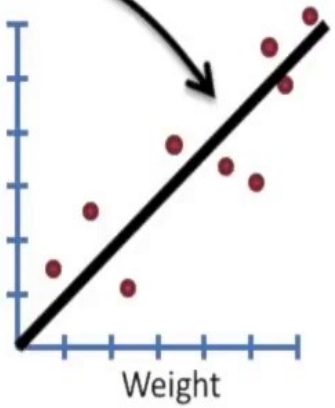


$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

The **slope** = 0.8

$y = 1 \times$ **0.01** $+ 0.9 \times$ **0.8**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

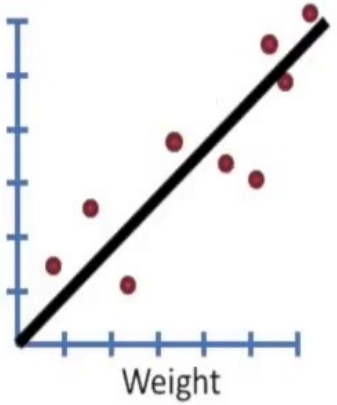


$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

...do the math...

$y = 1 \times \mathbf{0.01} + 0.9 \times \mathbf{0.8} = \mathbf{0.73}$

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

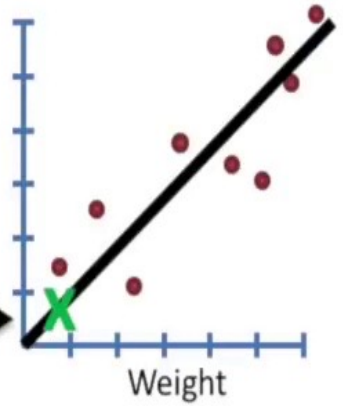


$y = 1 \times$ **y-intercept** $+ 0.9 \times$ **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

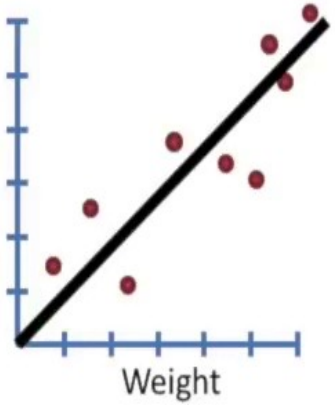
...and you get a point on the least-squares fit line that corresponds with the first data point.

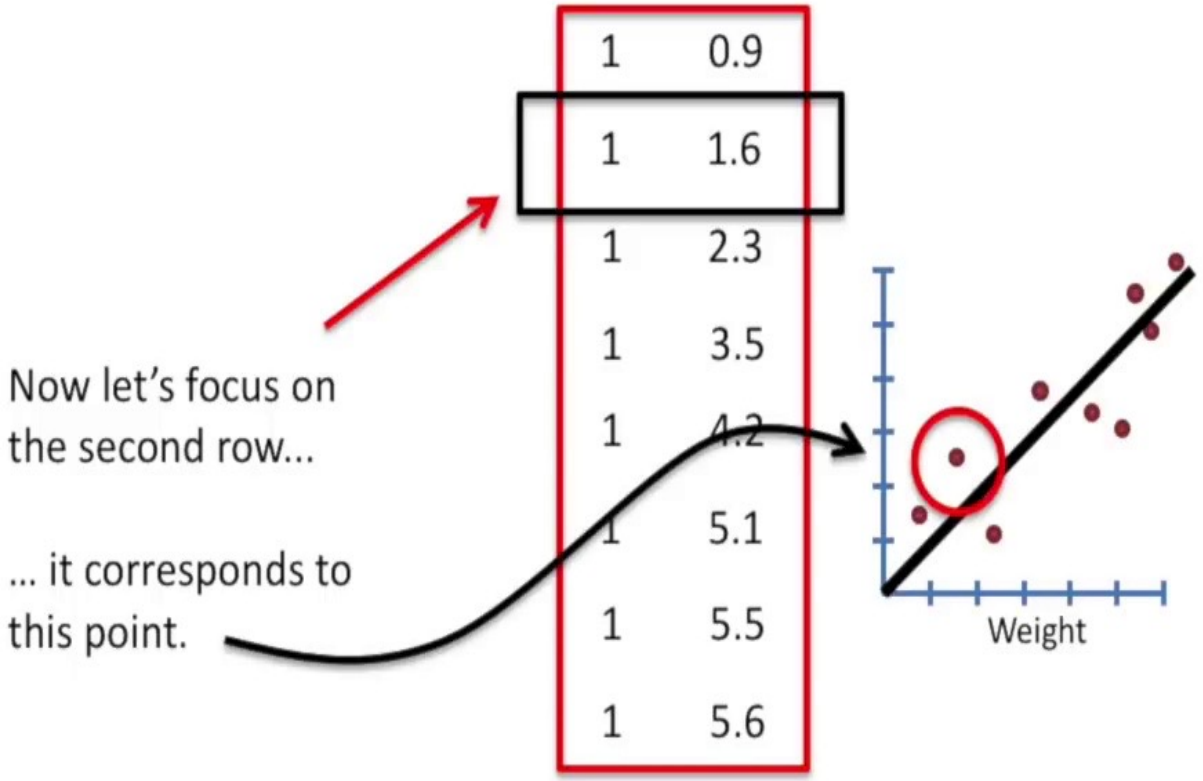
$y = 1 \times \mathbf{0.01} + 0.9 \times \mathbf{0.8} = \mathbf{0.73}$



Now let's focus on the second row...

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

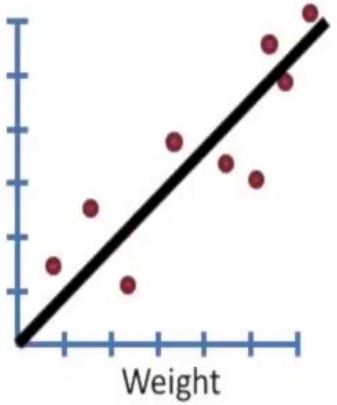



1 0.9
1 1.6
1 2.3
1 3.5
1 4.2
1 5.1
1 5.5
1 5.6
Now let's focus on the second row...
... it corresponds to this point.
Weight

$y = 1 \times$ **y-intercept** $+ 1.6 \times$ **slope**

The number in the first column multiplies the **y-intercept**...

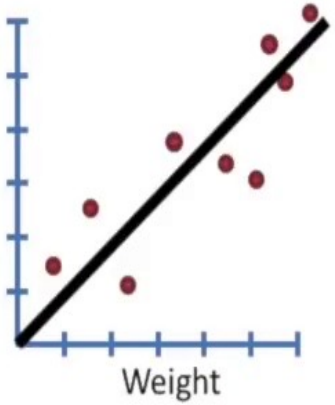
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



$y = 1 \times$ **y-intercept** $+ 1.6 \times$ **slope**

The number in the second column scales the **slope**...

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

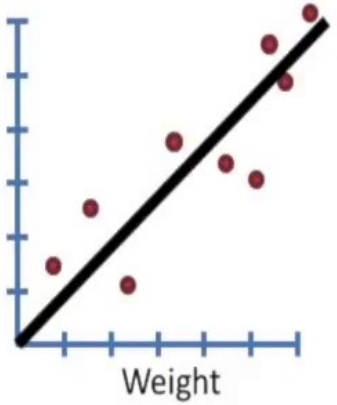


$y = 1 \times$ **y-intercept** $+ 1.6 \times$ **slope**

Plug in the **y-intercept** and the **slope** and do the math...

$y = 1 \times$ **0.01** $+ 1.6 \times$ **0.8**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

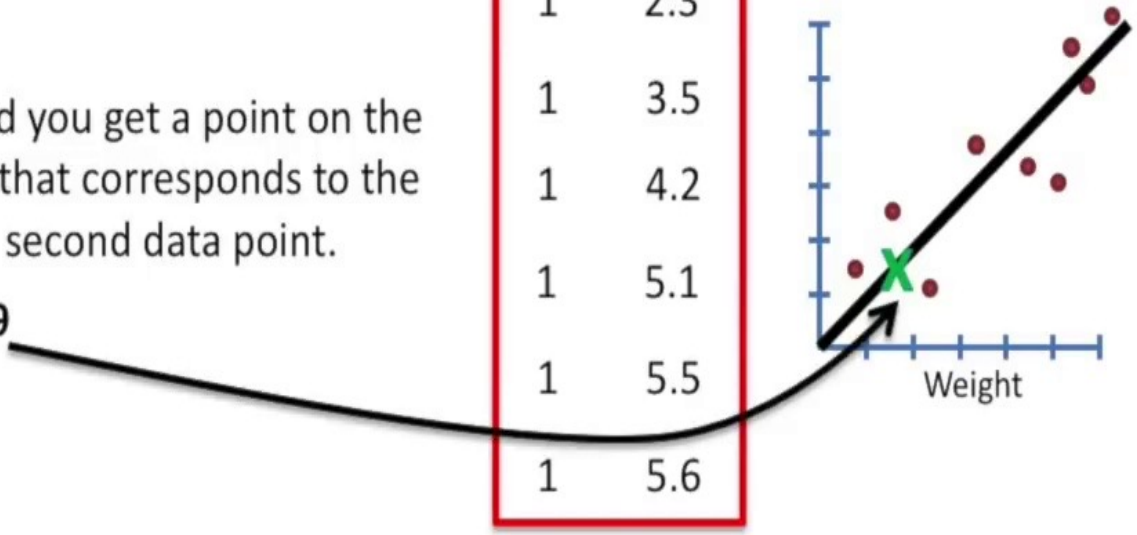


$y = 1 \times$ **y-intercept** $+ 1.6 \times$ **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

...and you get a point on the line that corresponds to the second data point.

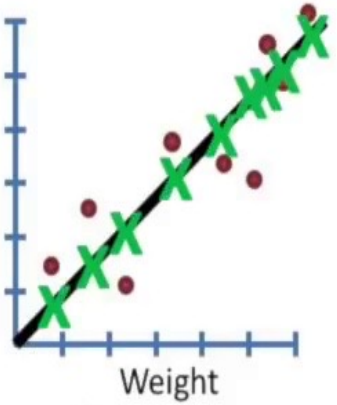
$y = 1 \times \mathbf{0.01} + 1.6 \times \mathbf{0.8} = \mathbf{1.29}$



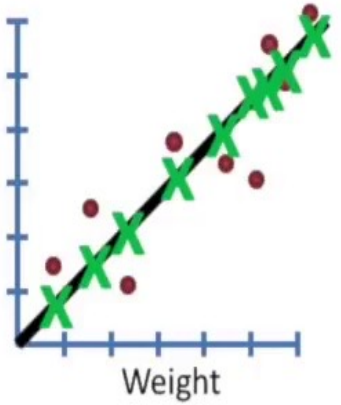
y = **y-intercept** + **slope**

1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6

Plugging each row into the equation gives us a bunch of points on the least-squares fit line.

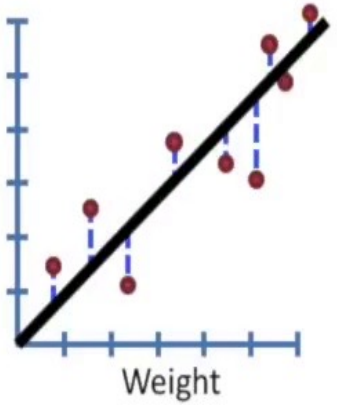


Once we have all the points on the line...



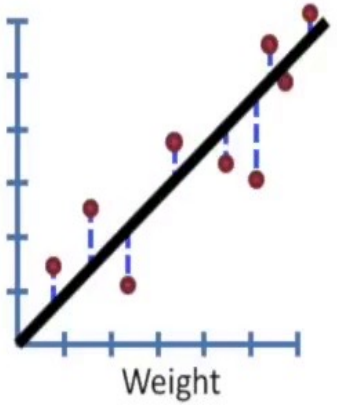
Once we have all the points on the line...

...we can calculate the residuals...



Once we have all the points on the line...

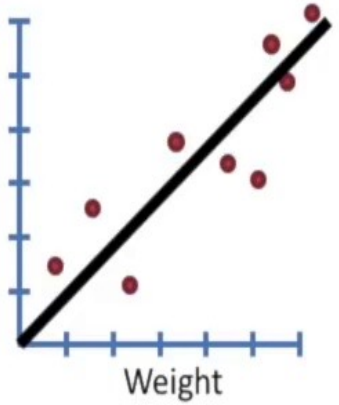
...we can calculate the residuals...



...and calculate a p-value.

This example shows that a design matrix isn't always just a bunch of 0's and 1's, but can be any set of numbers that we want to plug into an equation, one row at a time.

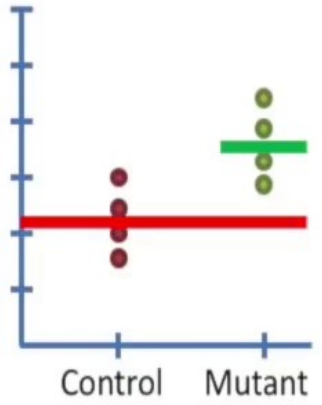
1	0.9
1	1.6
1	2.3
1	3.5
1	4.2
1	5.1
1	5.5
1	5.6



One note before we move on...

Since this style of design matrix (with 1's all down the first column) is more common, all of the examples from here on will be consistent with this format.

1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1



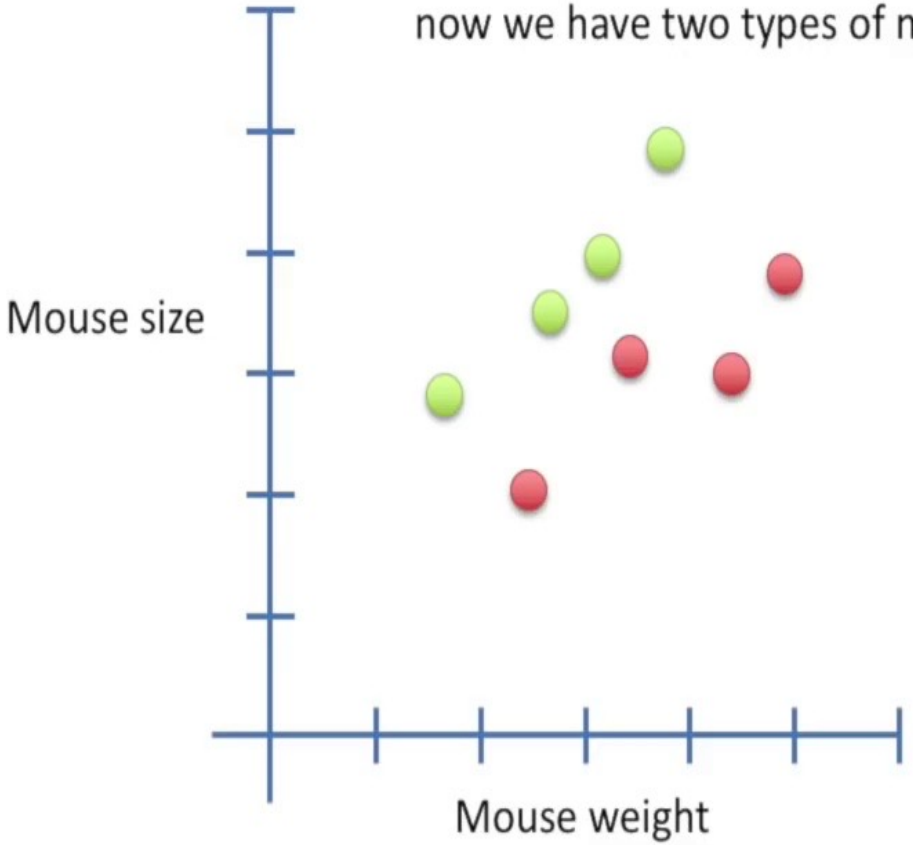
$y = \text{mean}_{\text{control}} + \text{difference}_{(\text{mutant - control})}$

Now that we know we can put any number into the design matrix, let's do something fancy...

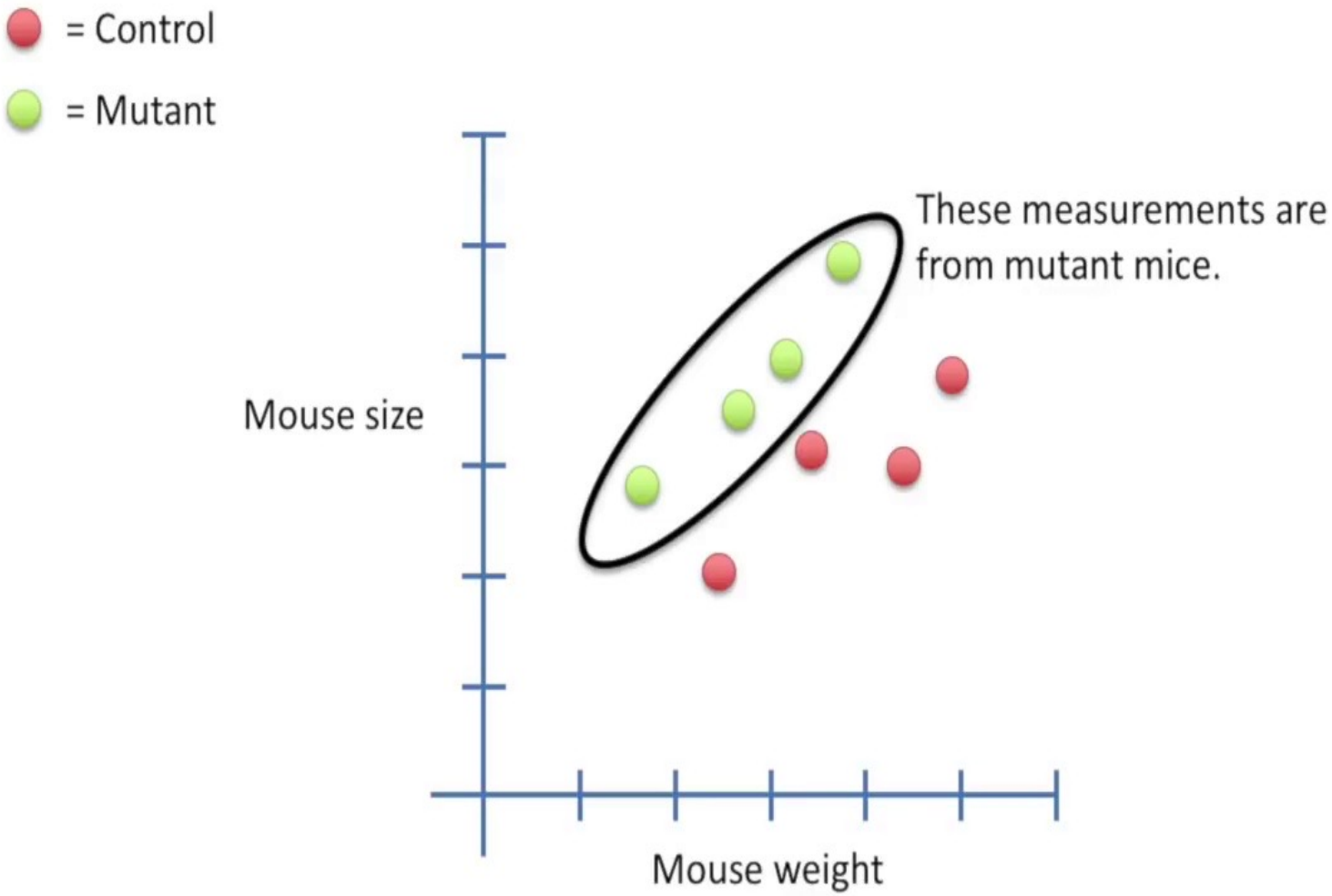
Now that we know we can put any number into the design matrix, let's do something fancy...

Let's combine a t-test and a regression!!!

We're back to the relationship between mouse weight and mouse size. However, now we have two types of mice...



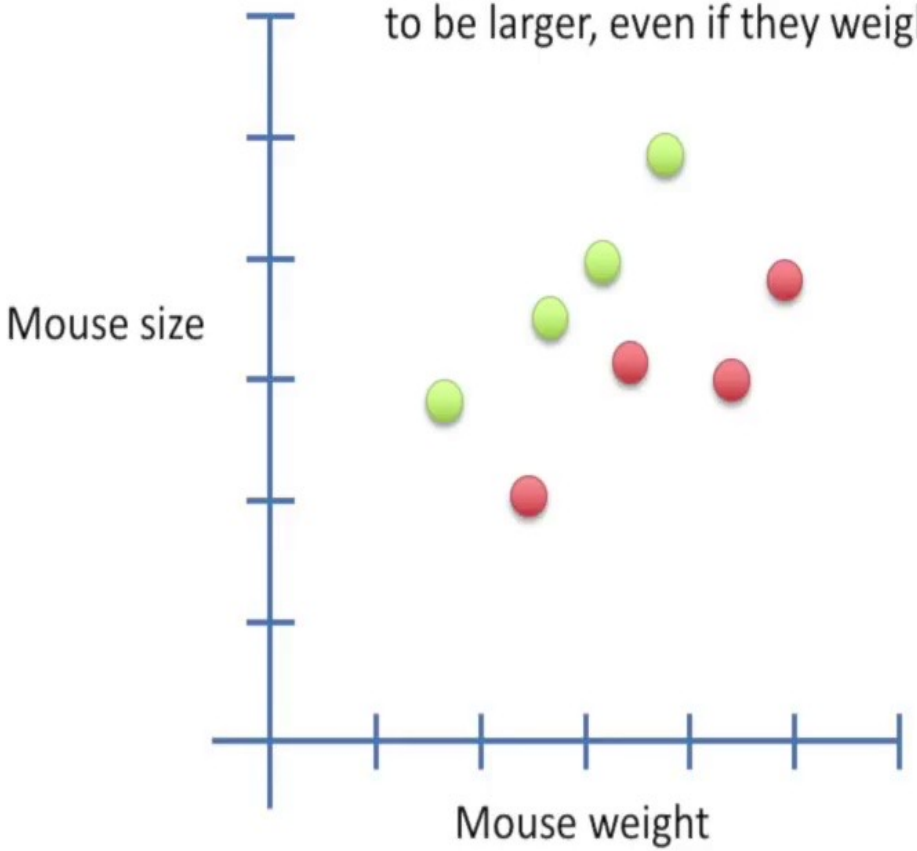

= Control
Mouse size
These measurements are
from normal, control mice.
Mouse weight

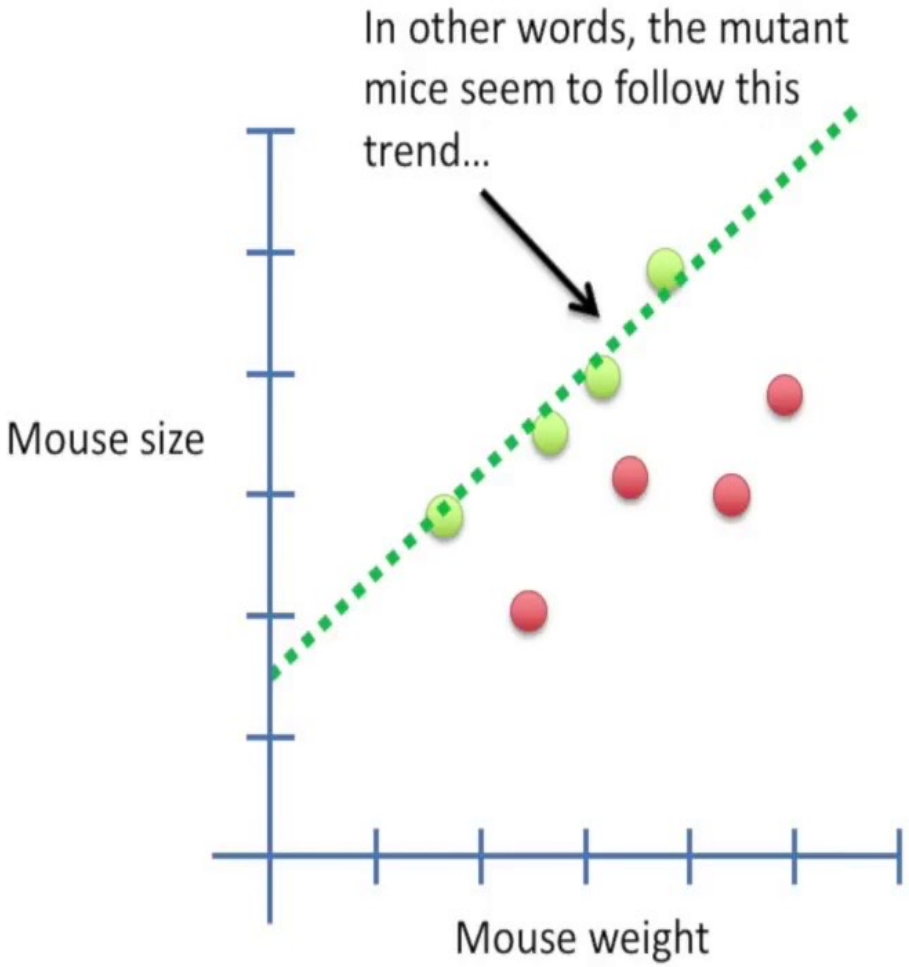

= Control
= Mutant
These measurements are from mutant mice.
Mouse size
Mouse weight

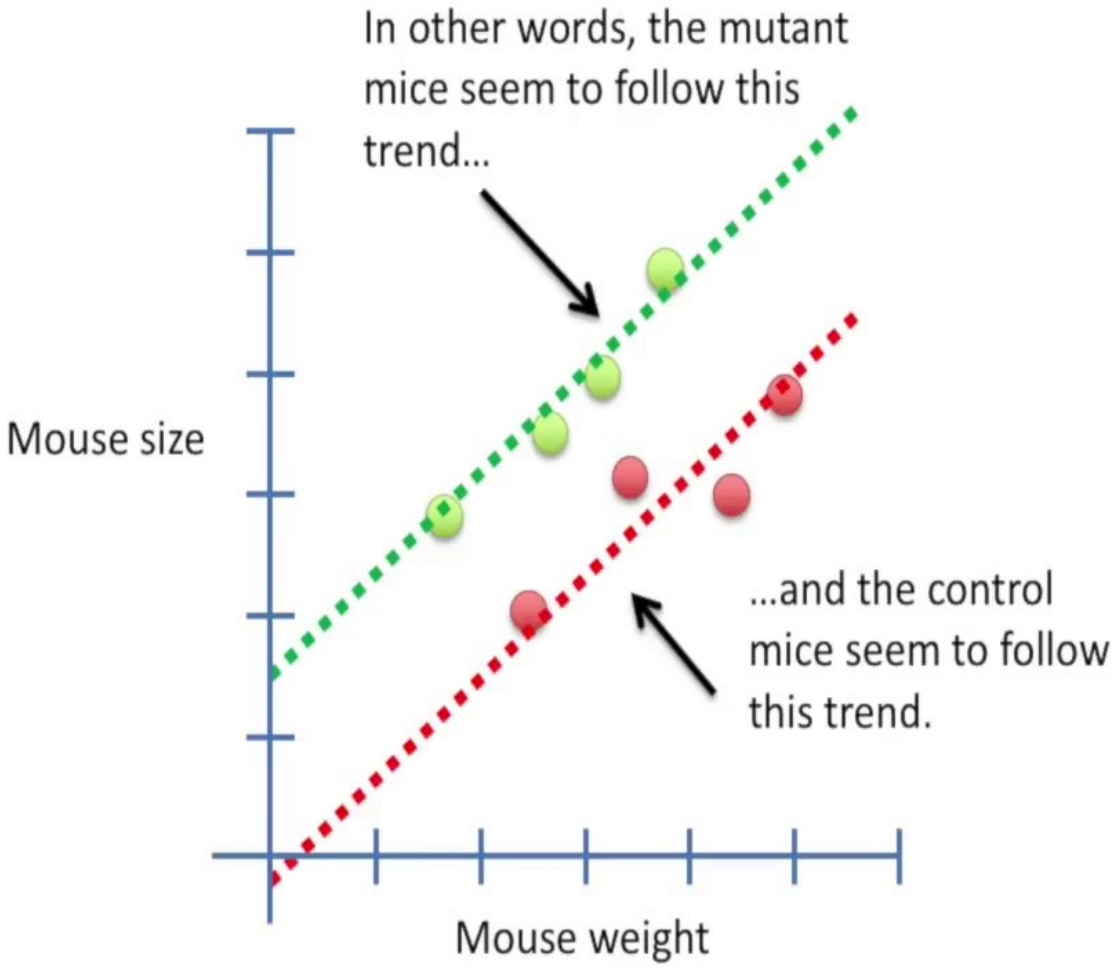
= Control

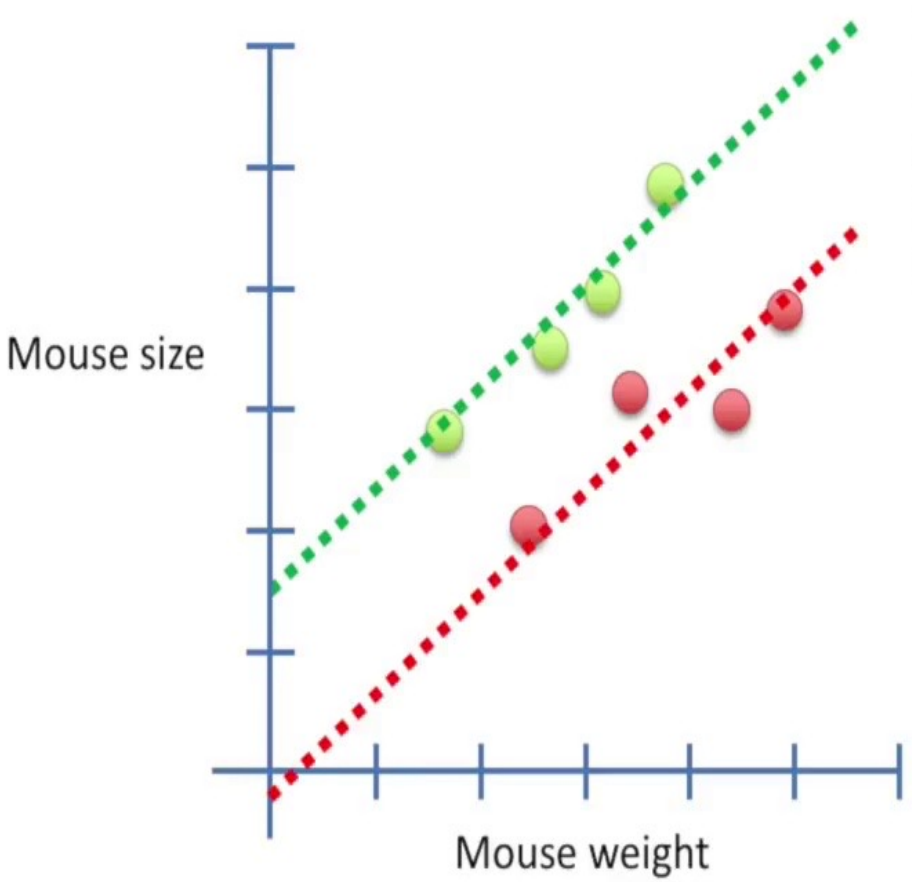
= Mutant

By eye, we can see that mutant mice tend to be larger, even if they weigh the same.



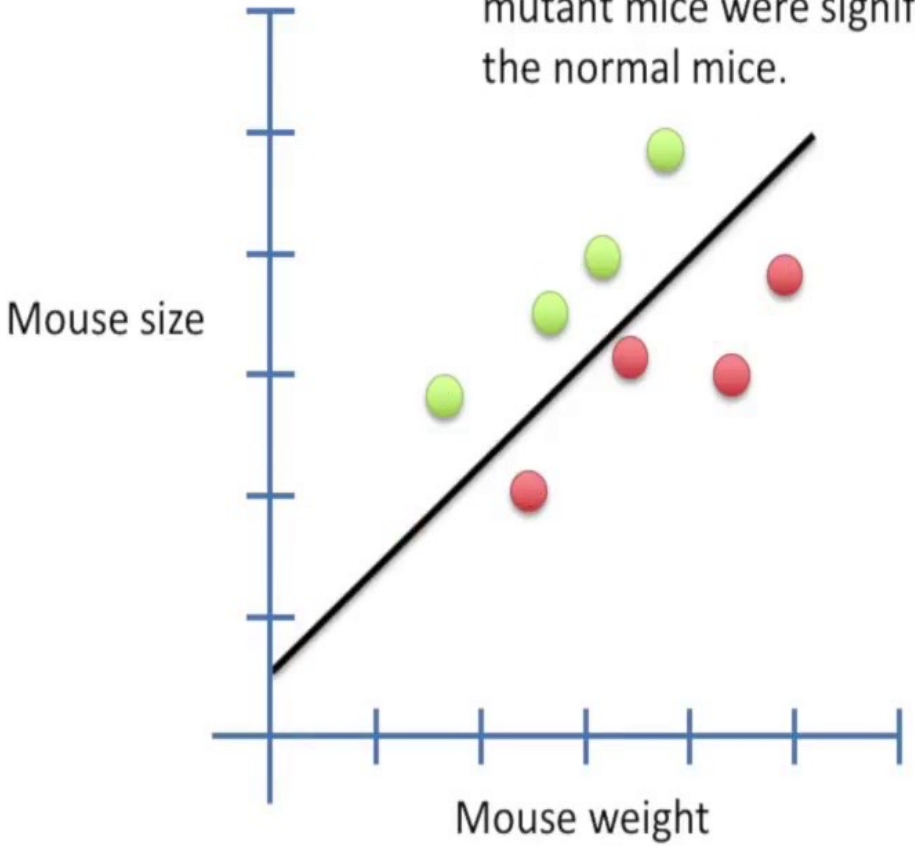

In other words, the mutant mice seem to follow this trend...
Mouse size
Mouse weight


In other words, the mutant mice seem to follow this trend...
Mouse size
...and the control mice seem to follow this trend.
Mouse weight

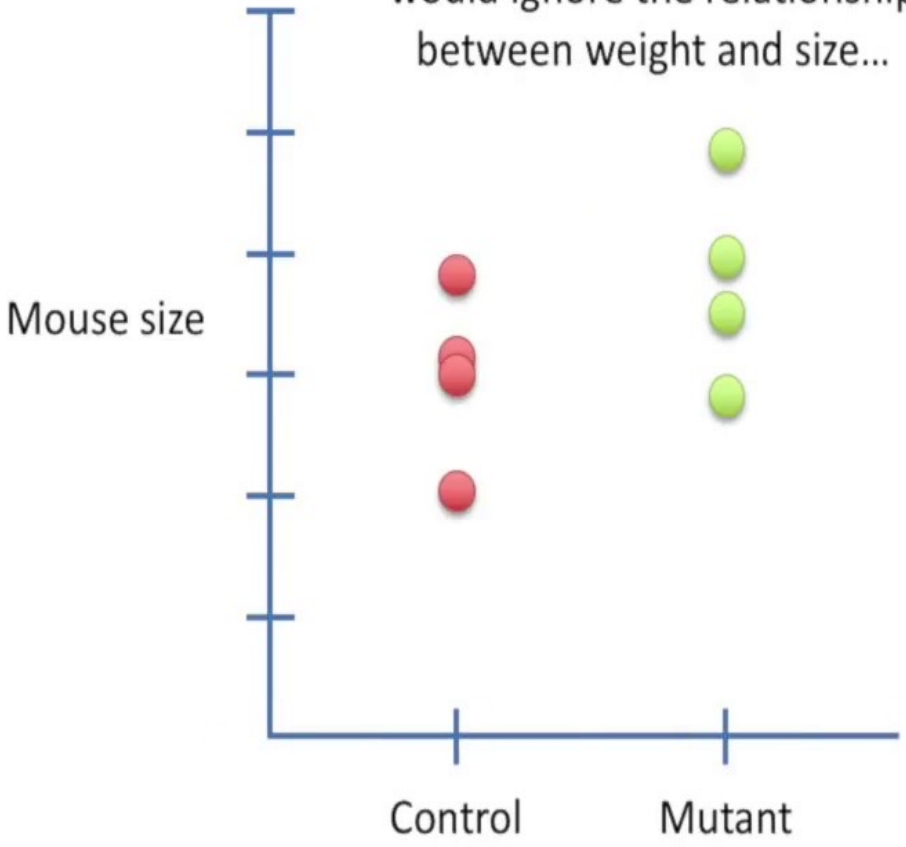


Can we use statistics to test if there is a significant difference between the two types of mice?

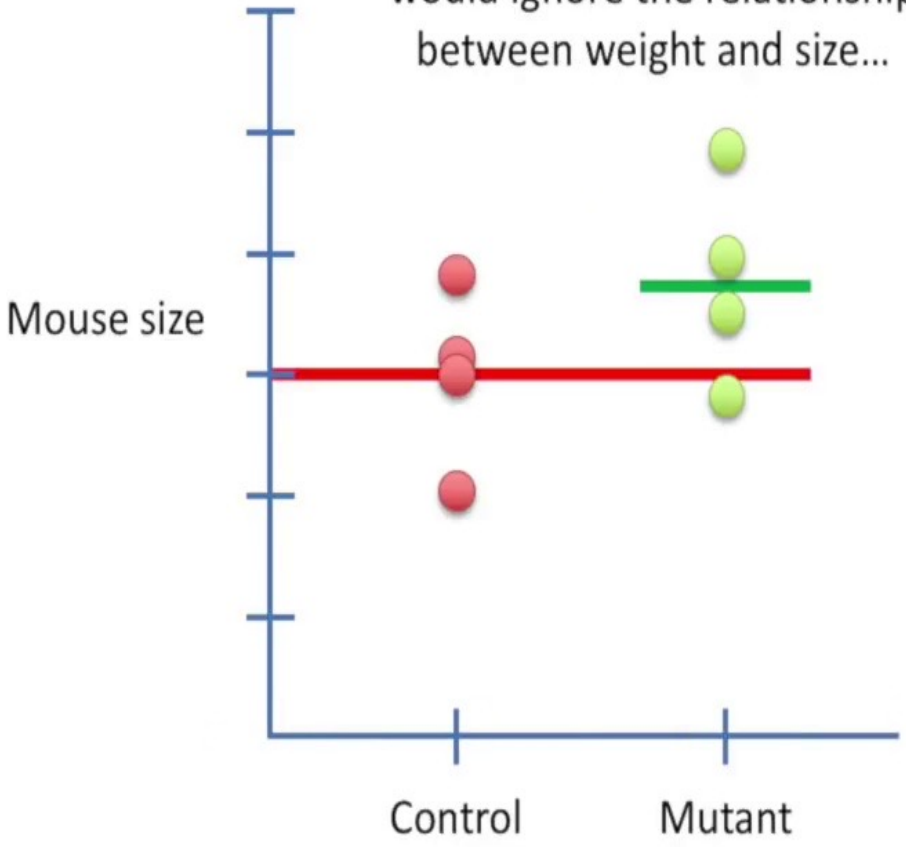
If we just did a regression, we'd get a nice looking line, but it wouldn't tell us if the mutant mice were significantly larger than the normal mice.

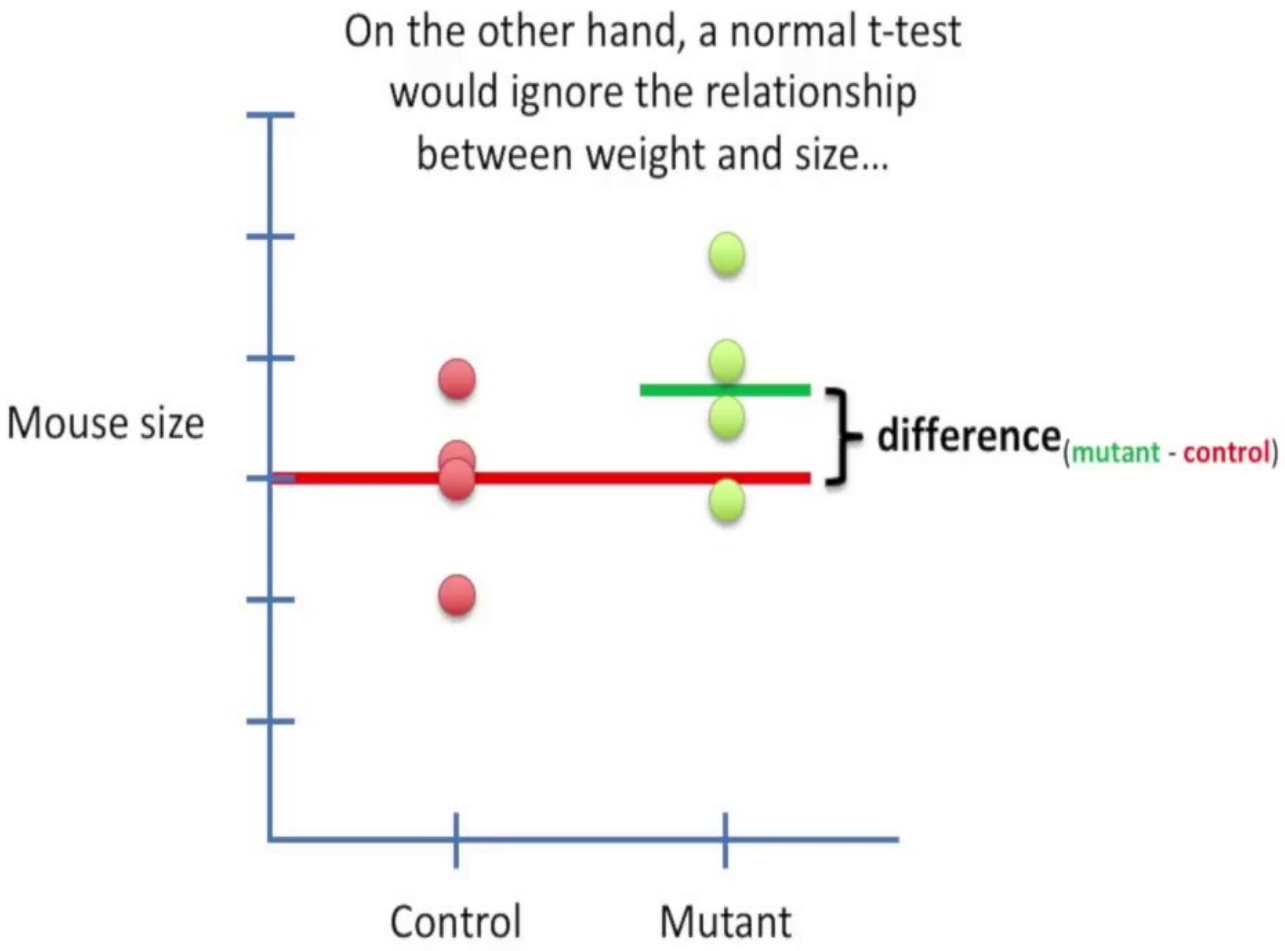


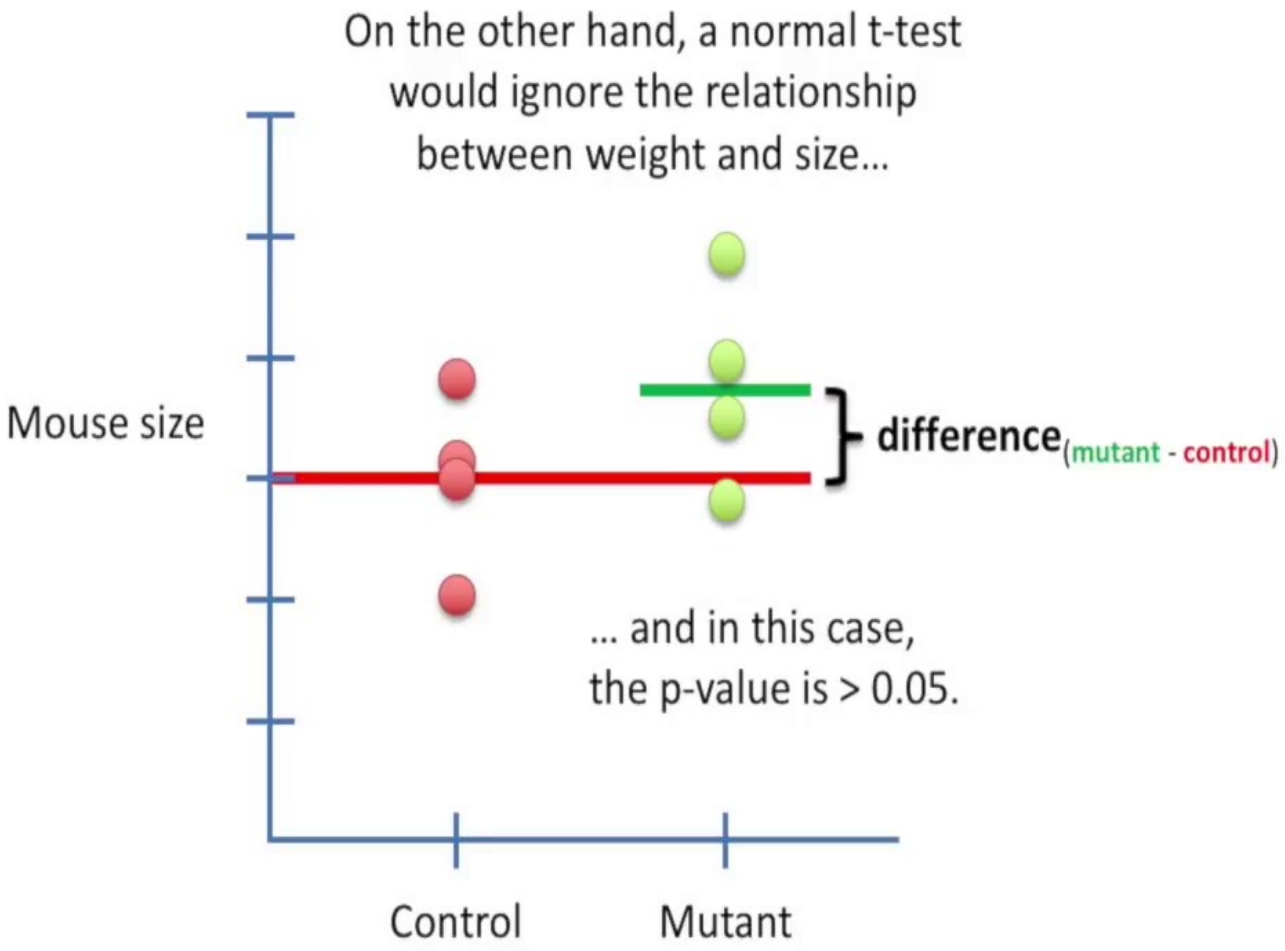
On the other hand, a normal t-test would ignore the relationship between weight and size...



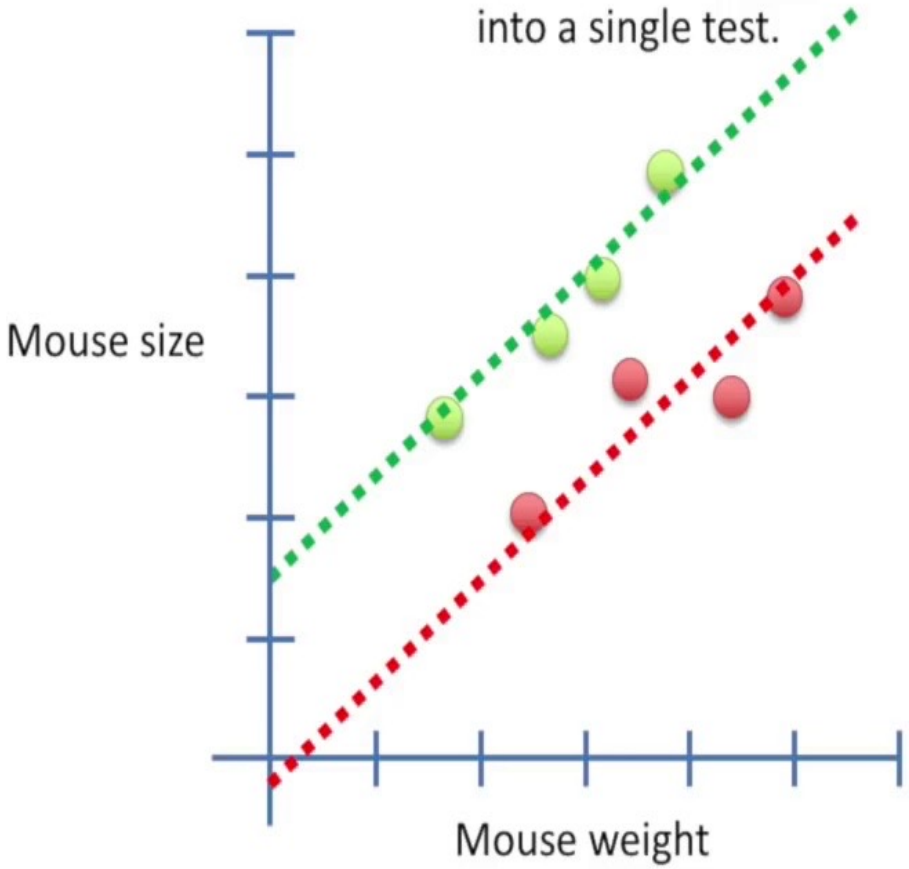
On the other hand, a normal t-test would ignore the relationship between weight and size...

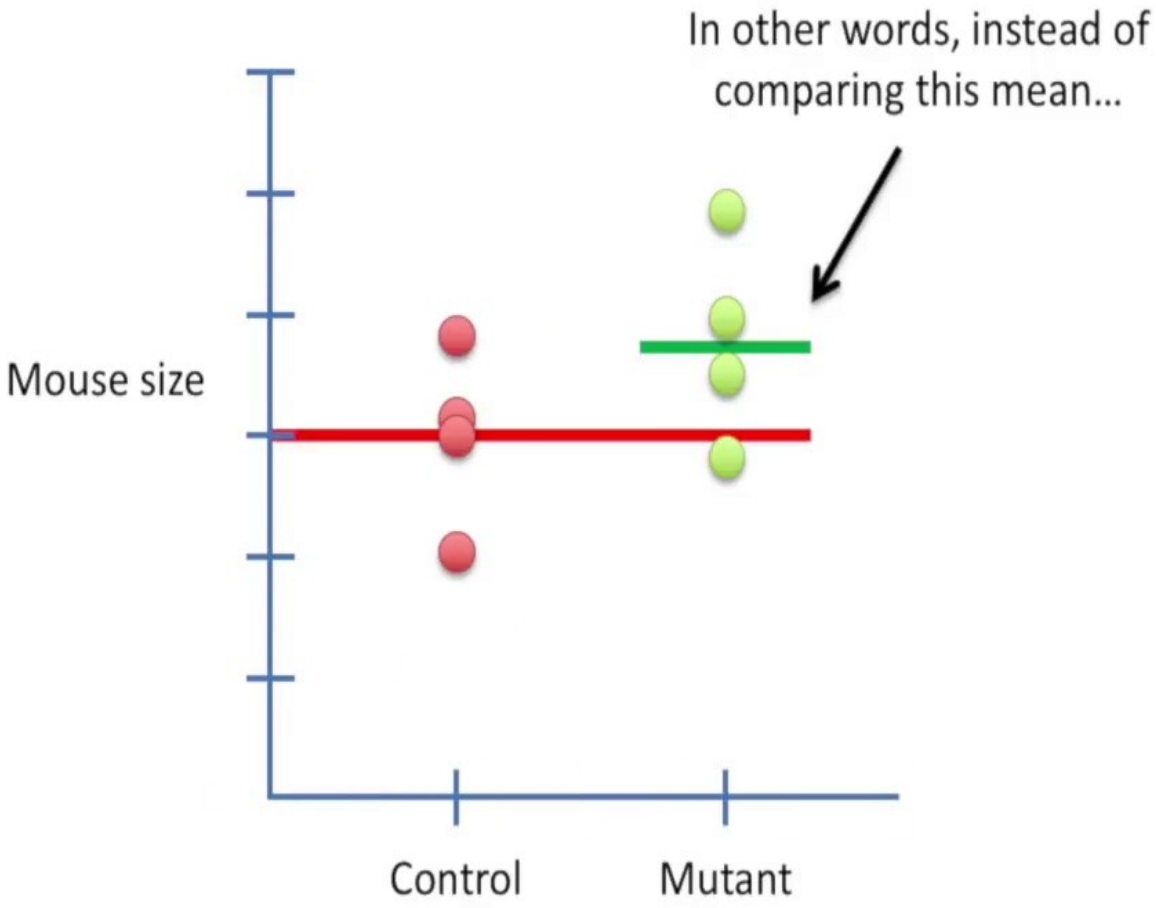


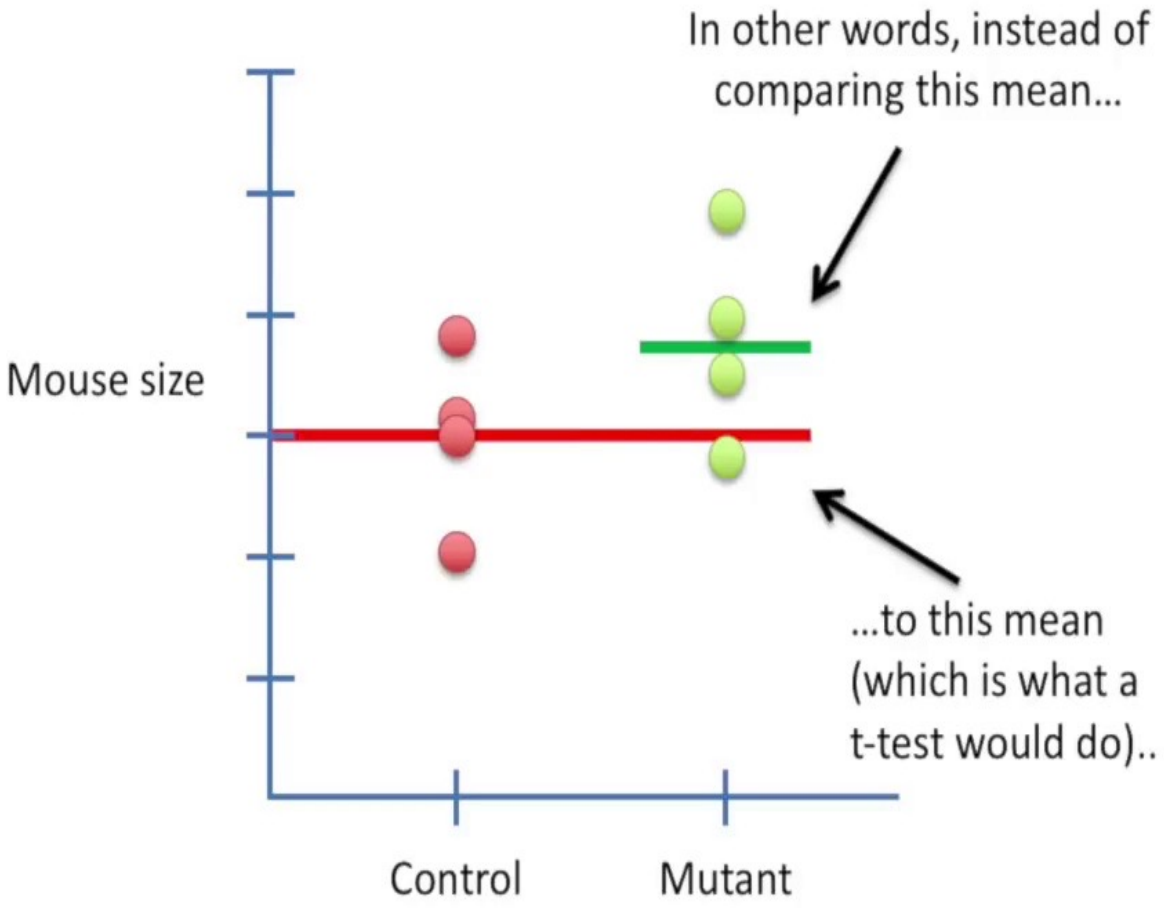

On the other hand, a normal t-test would ignore the relationship between weight and size...
Mouse size
difference(mutant - control)
Control
Mutant

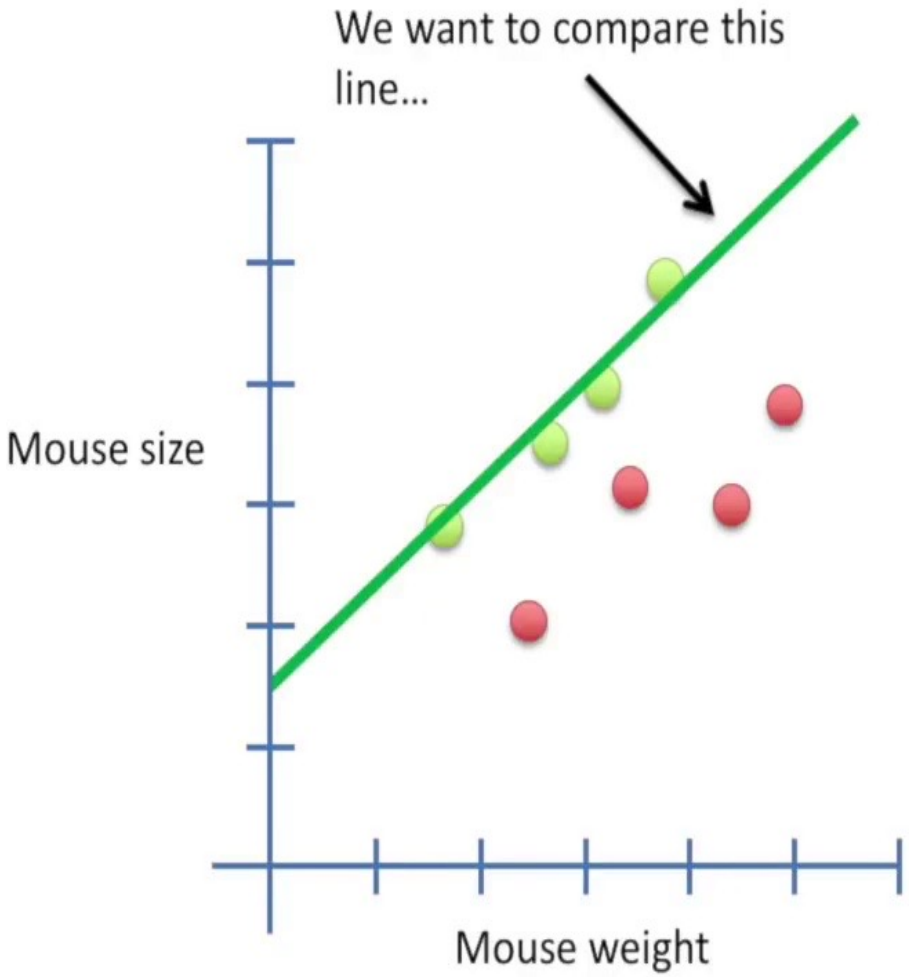

On the other hand, a normal t-test would ignore the relationship between weight and size...
Mouse size
difference(mutant - control)
... and in this case, the p-value is > 0.05.
Control
Mutant

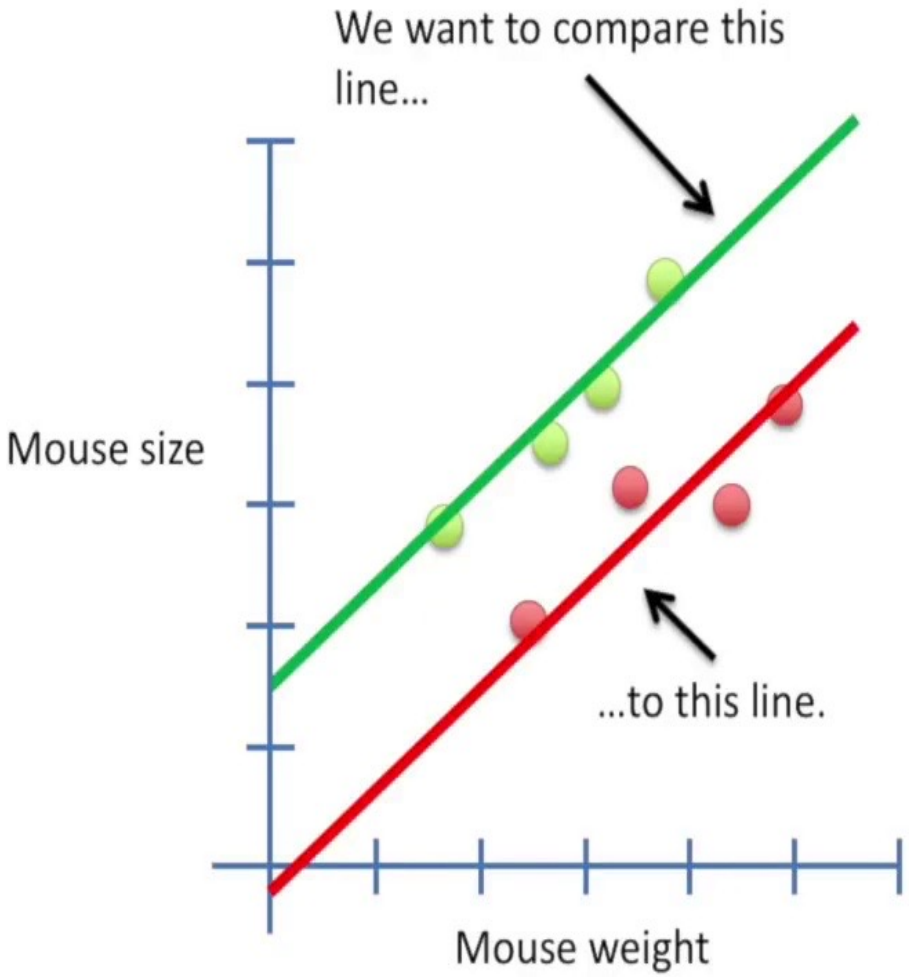
Since mouse type and the relationship between weight and size are both important, we need to combine them into a single test.



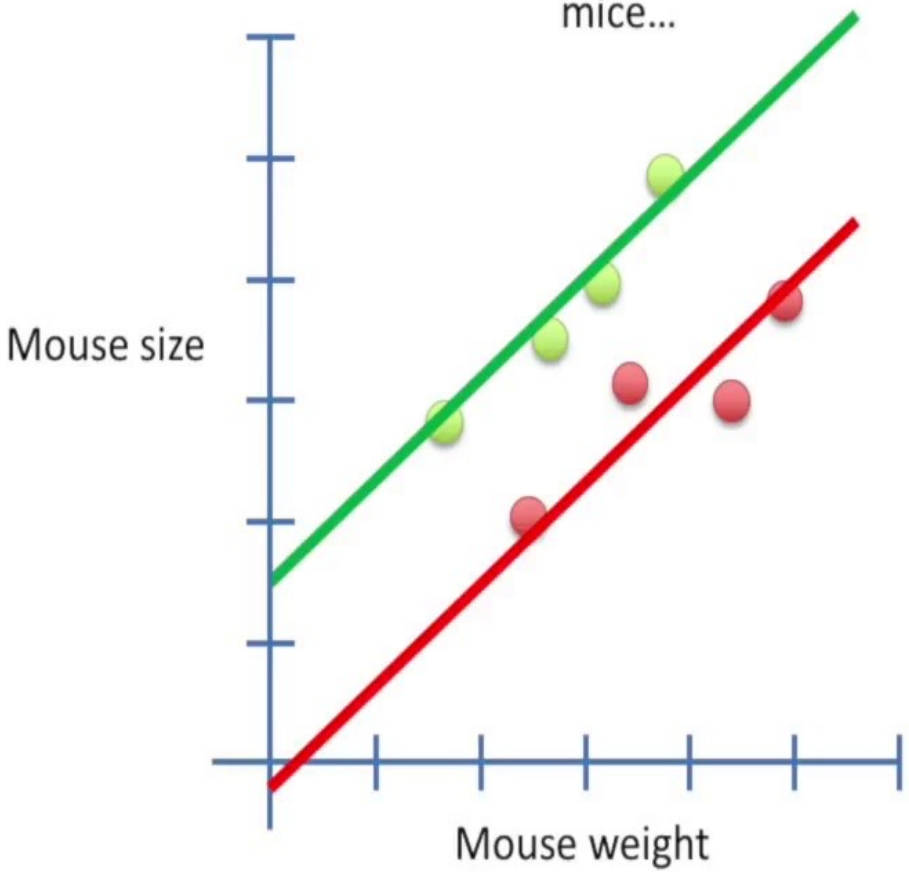

In other words, instead of comparing this mean...
Mouse size
Control
Mutant


In other words, instead of comparing this mean...
Mouse size
...to this mean (which is what a t-test would do)..
Control
Mutant


We want to compare this line...
Mouse size
Mouse weight

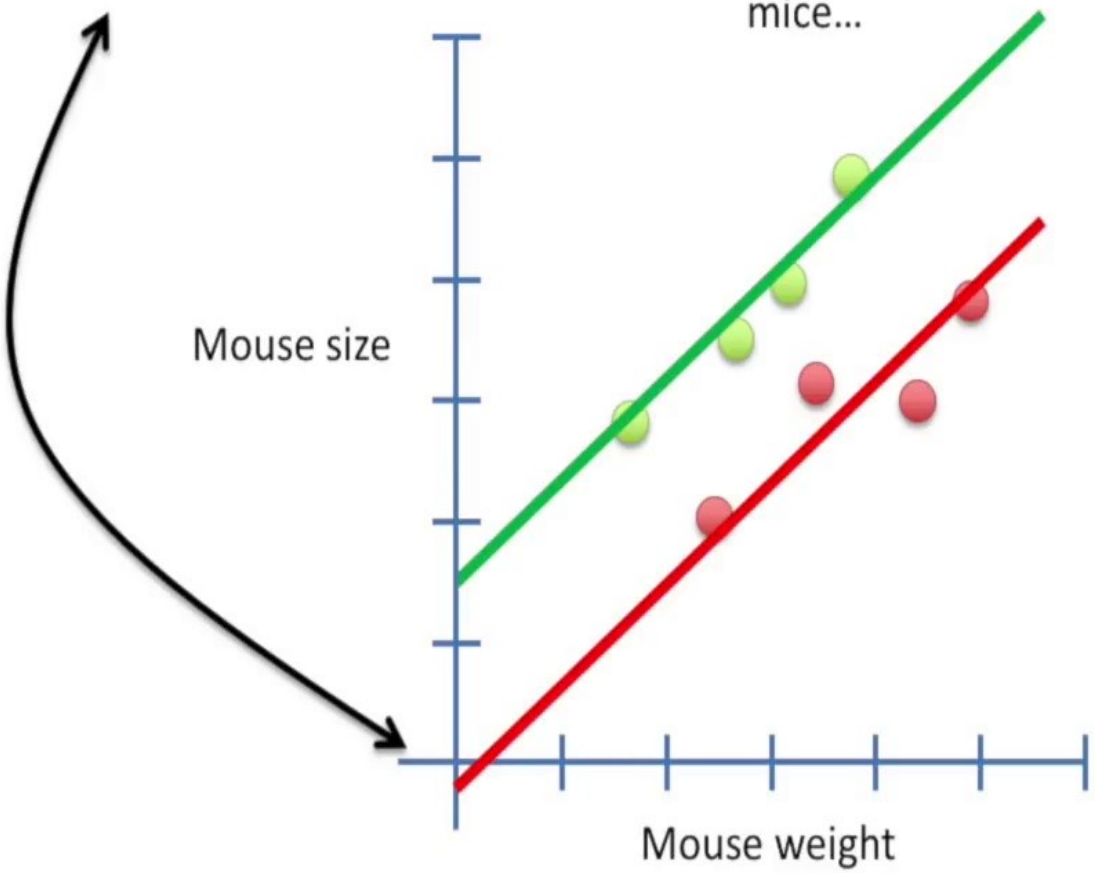

We want to compare this line...
Mouse size
...to this line.
Mouse weight

To do this, we need an equation that has a term for the y-intercept for the normal mice...



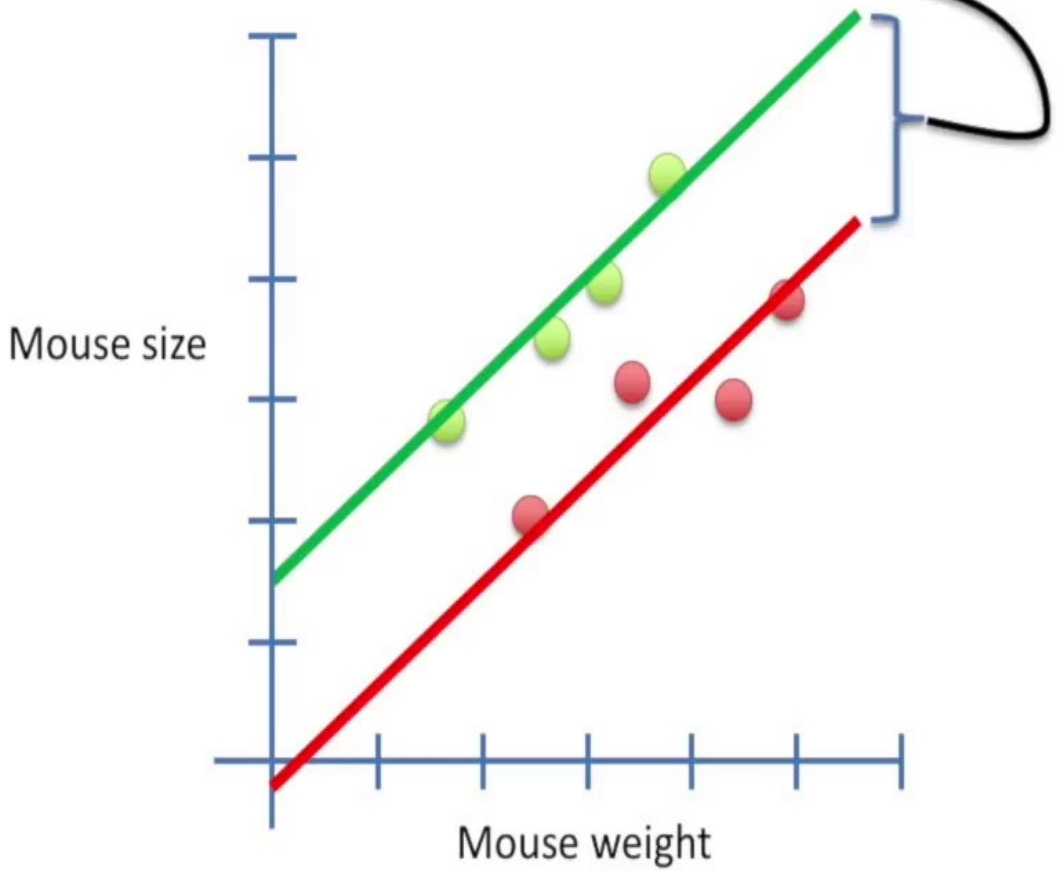
y = **control intercept**

To do this, we need an equation that has a term for the y-intercept for the normal mice...

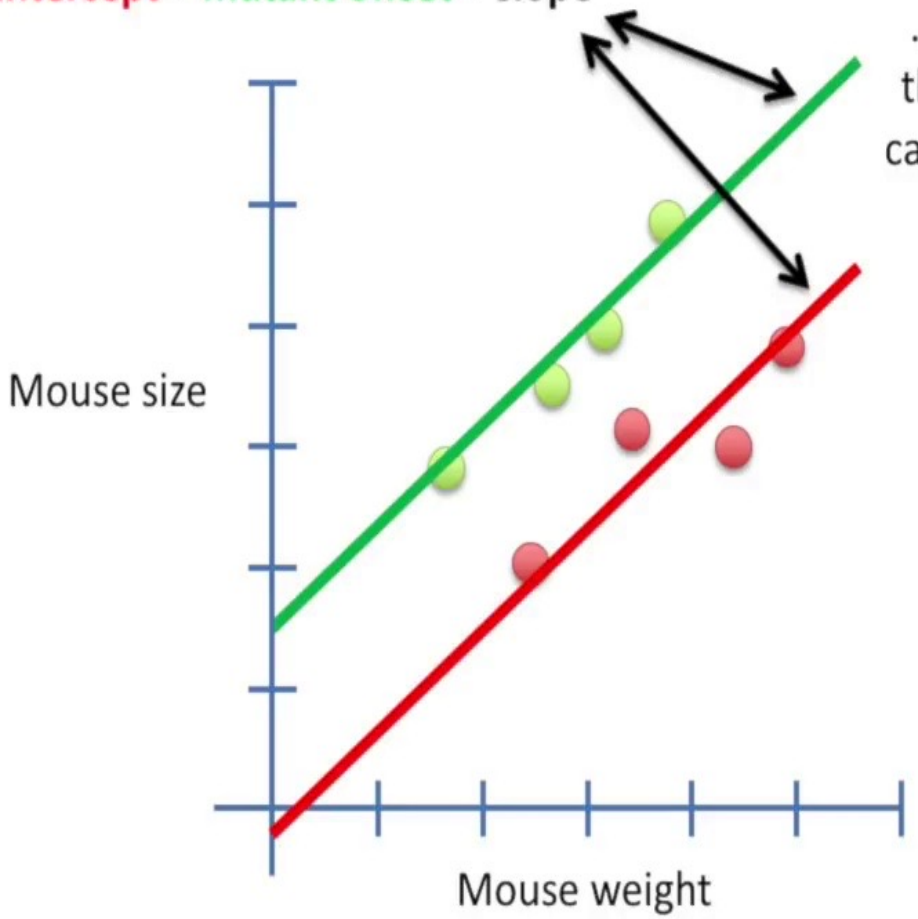


y = control intercept + mutant offset

… a term for the mutant mouse offset…

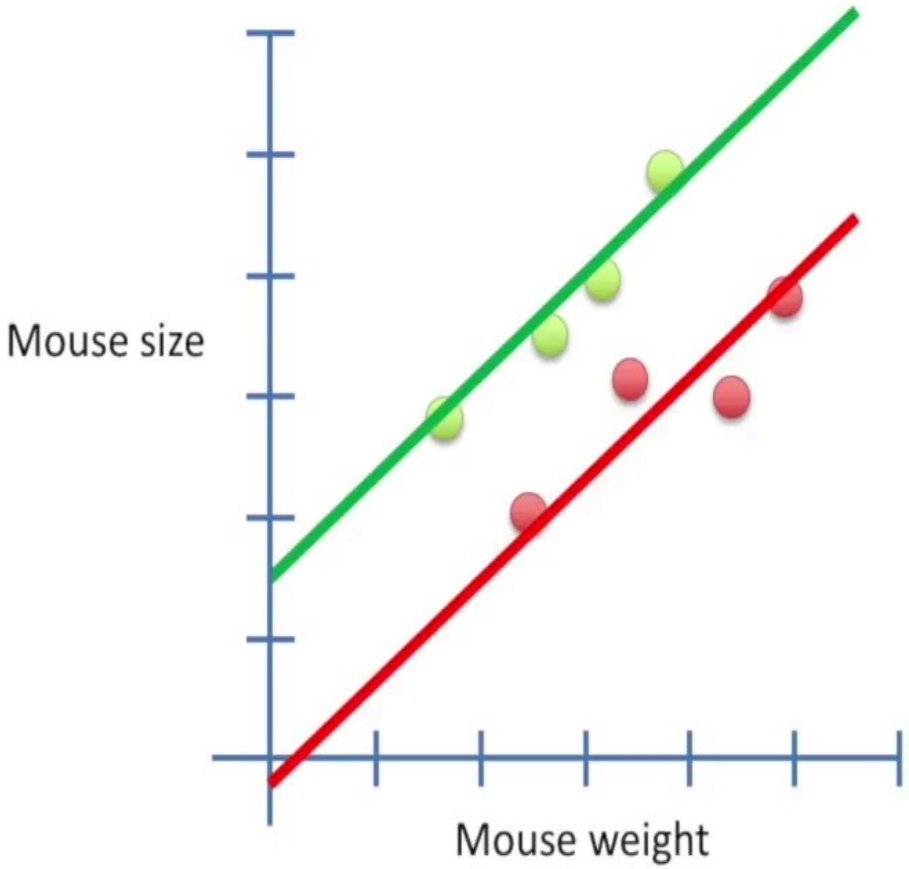


y = control intercept + mutant offset + slope



… and lastly, a term for the slope (which, in this case, is the same for both types of mice).

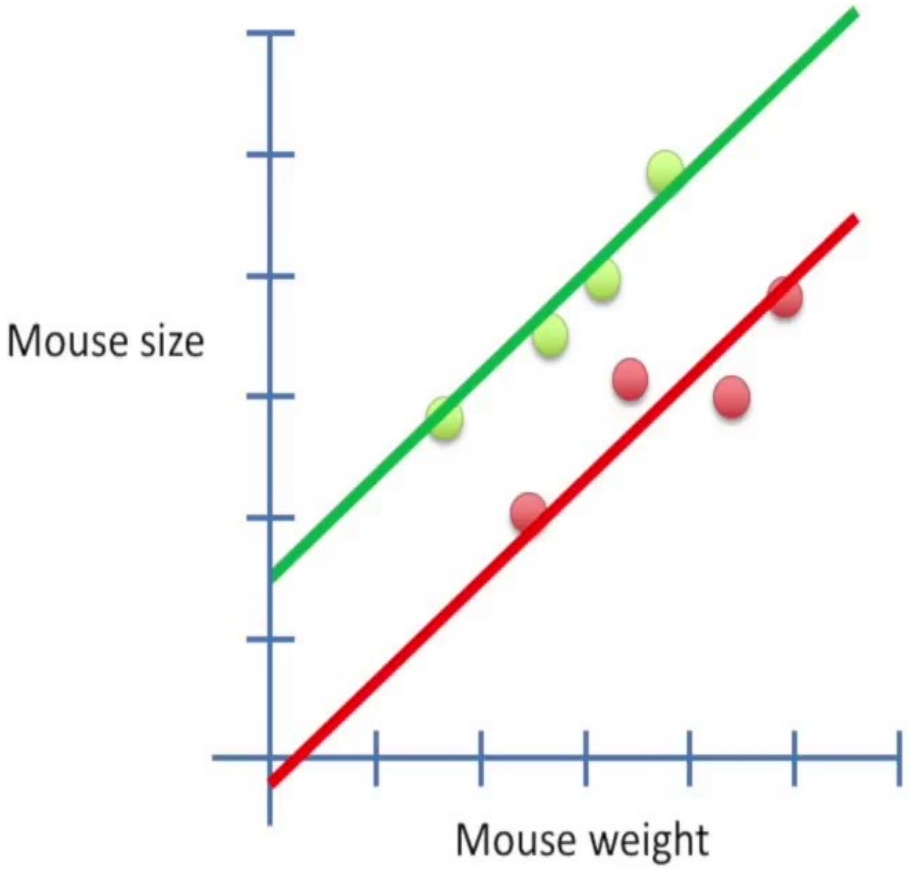
y = control intercept + mutant offset + slope



This means we need a design matrix where the first column is 1's...

1
1
1
1
1
1
1
1

y = control intercept + mutant offset + slope

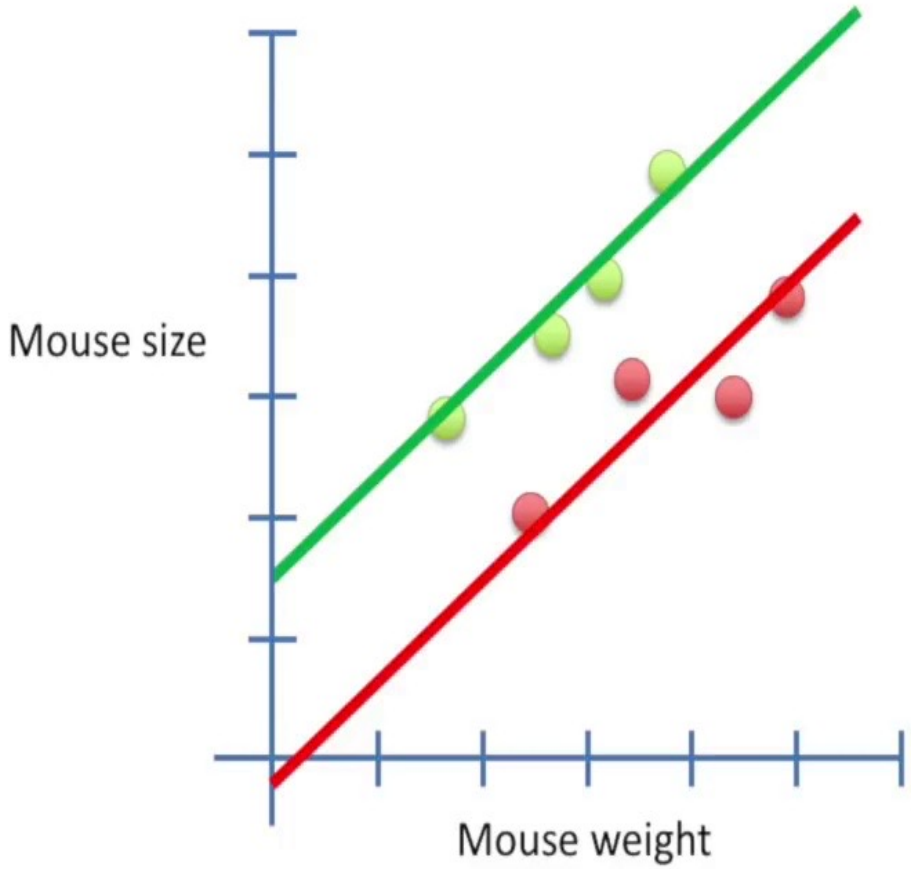


This means we need a design matrix where the first column is 1's...

1
1
1
1
1
1
1
1

This means that both lines intercept the y-axis at some point...

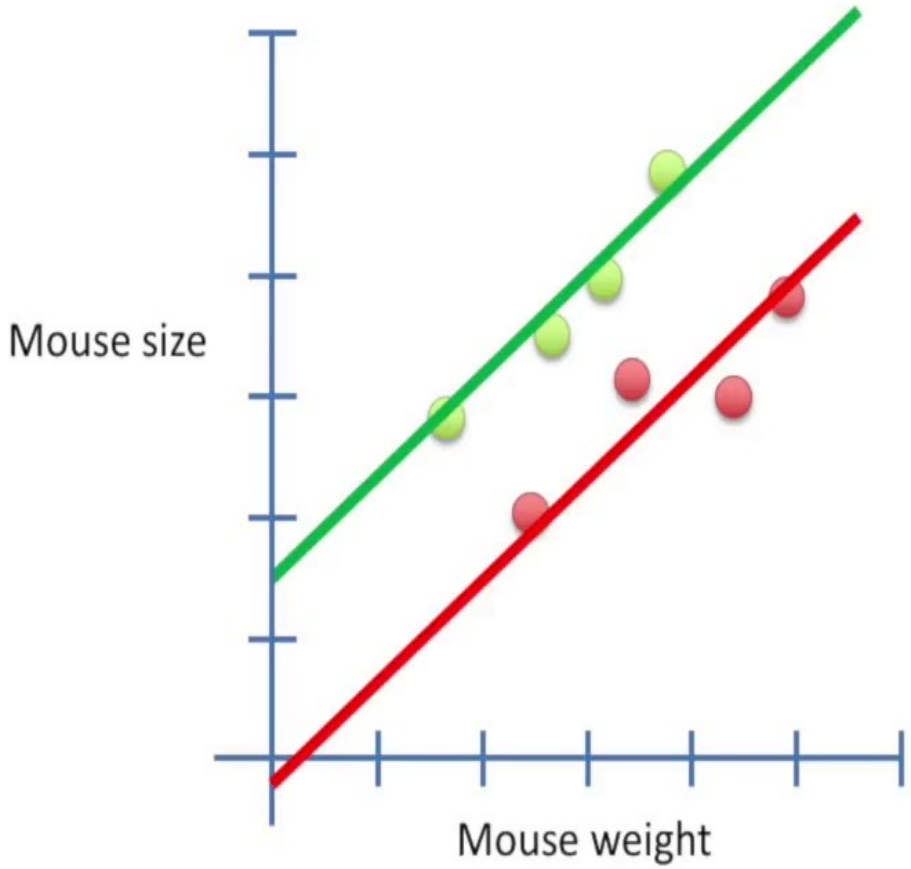
y = control intercept + mutant offset + slope



...the second column indicates whether the mutant offset is on or off...

1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

y = control intercept + mutant offset + slope

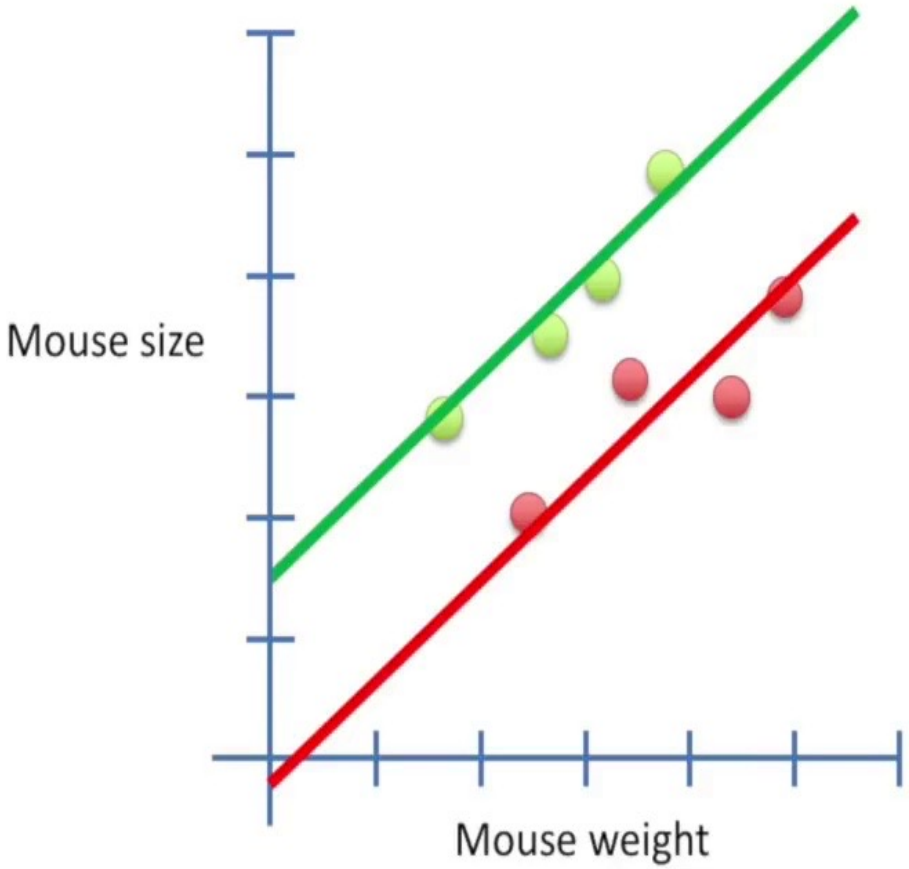


...the second column indicates whether the mutant offset is on or off...

1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

mutant offset is "off" for the control mice...

y = control intercept + mutant offset + slope

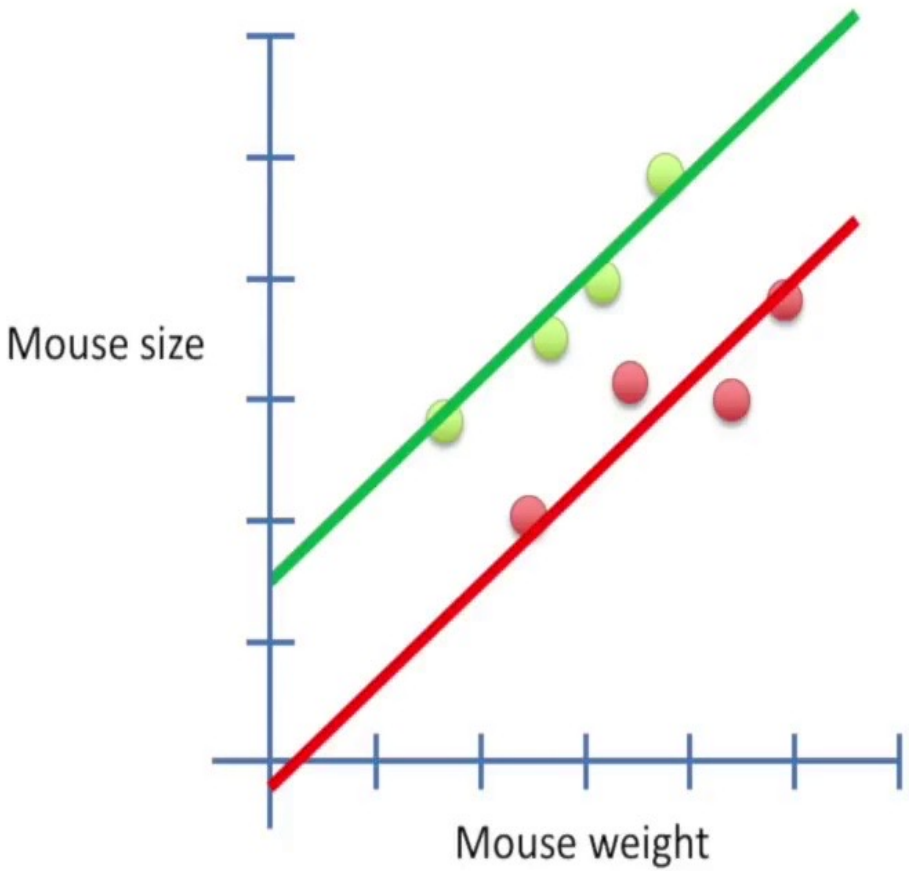


...the second column indicates whether the mutant offset is on or off...

1	0
1	0
1	0
1	0
1	1
1	1
1	1
1	1

...and "on" for the mutant mice. This allows the mutants to have their own y-intercept.

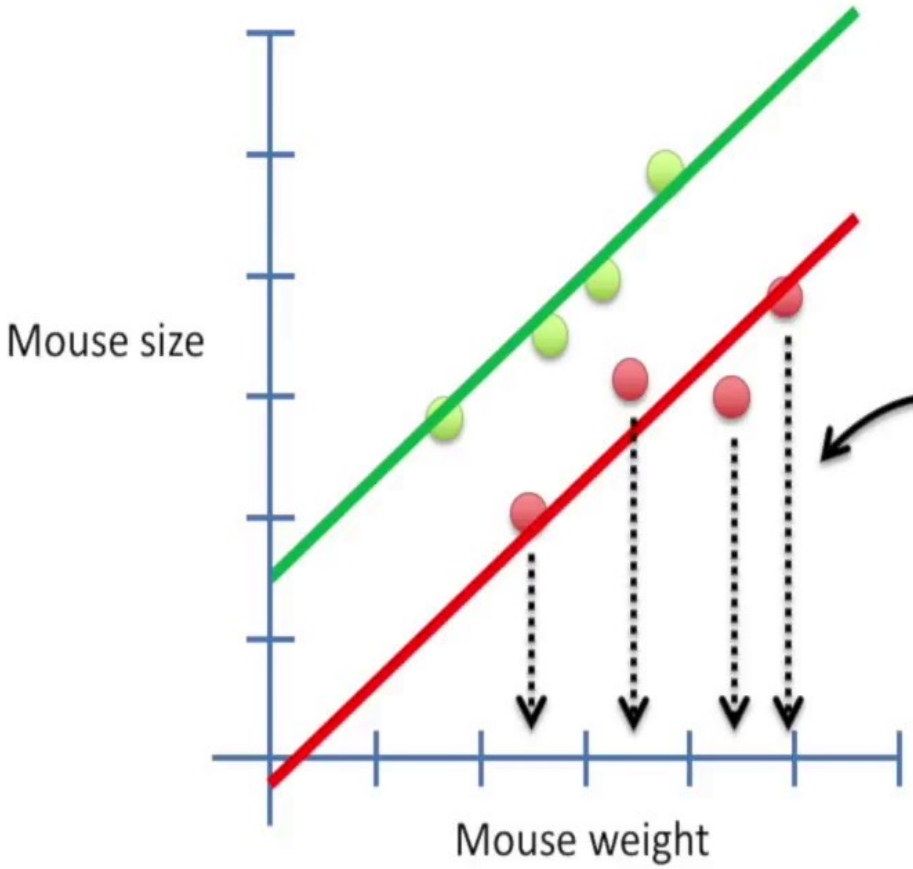
y = control intercept + mutant offset + slope



... and the last column has the weight data.

1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

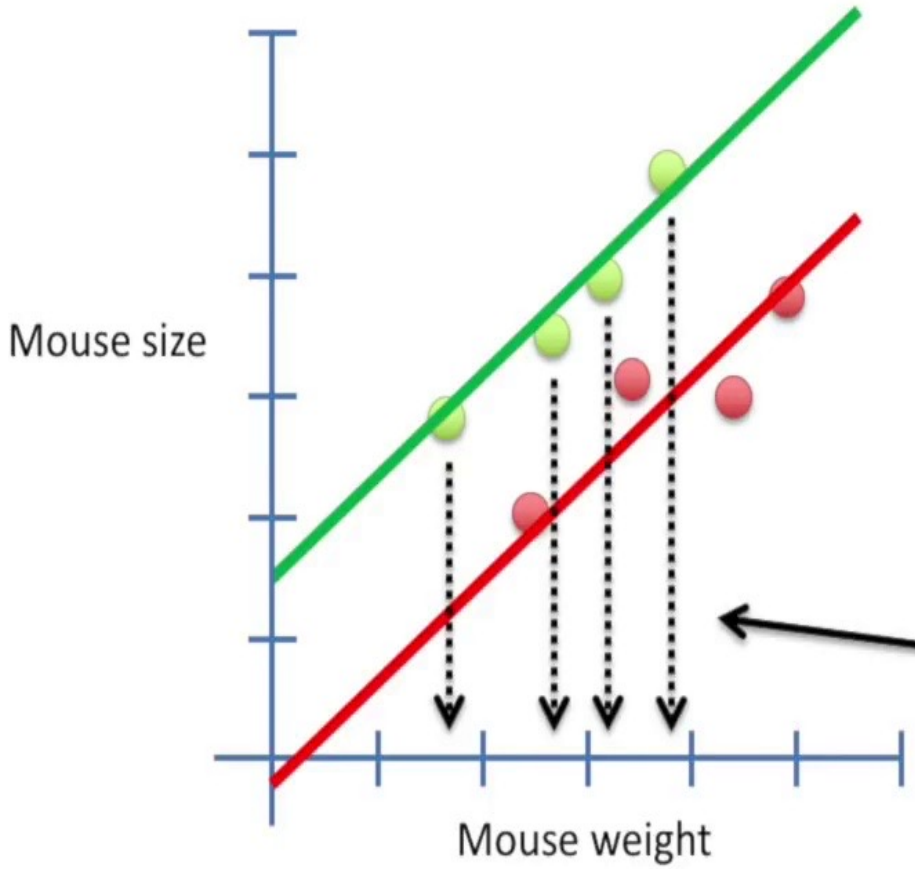
y = control intercept + mutant offset + slope



... and the last column has the weight data.

1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

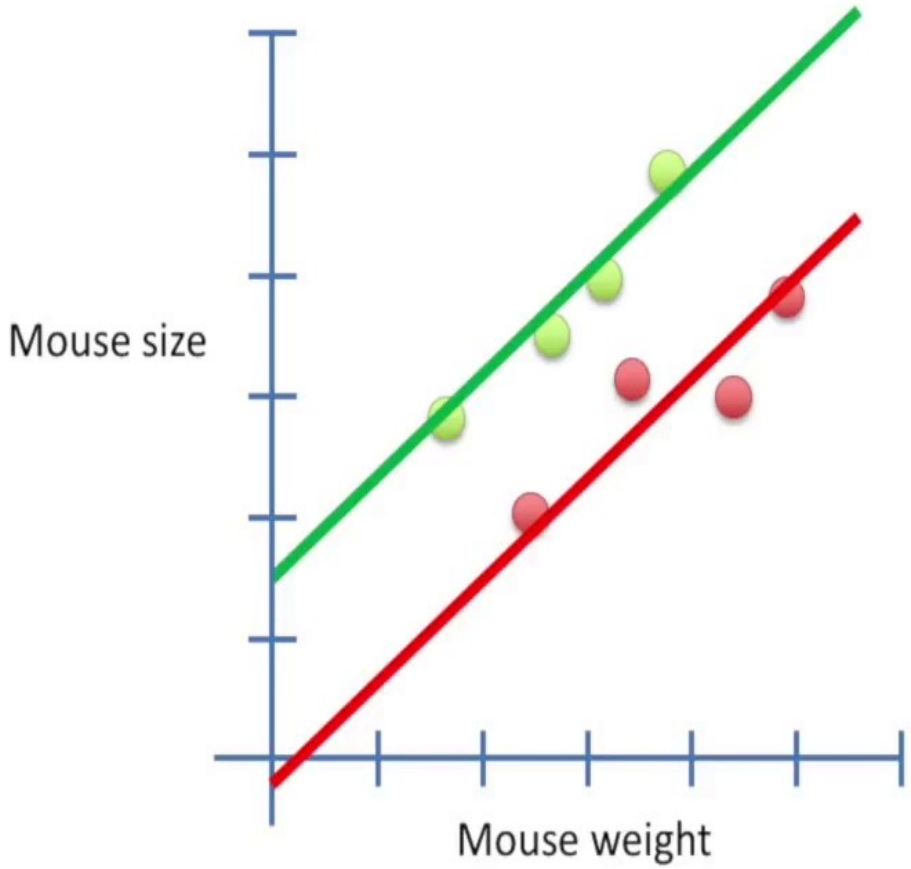
y = control intercept + mutant offset + slope



... and the last column has the weight data.

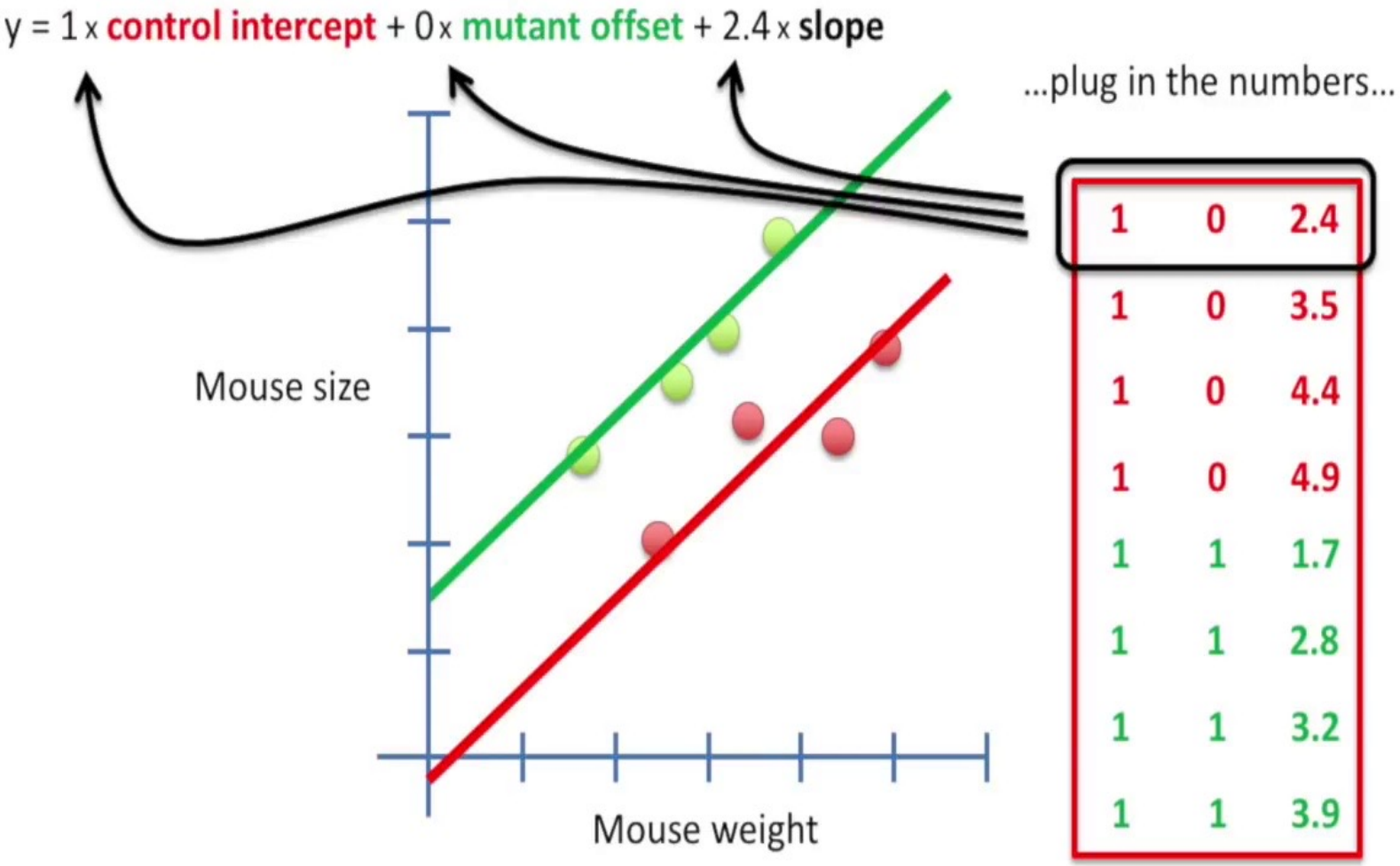
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope

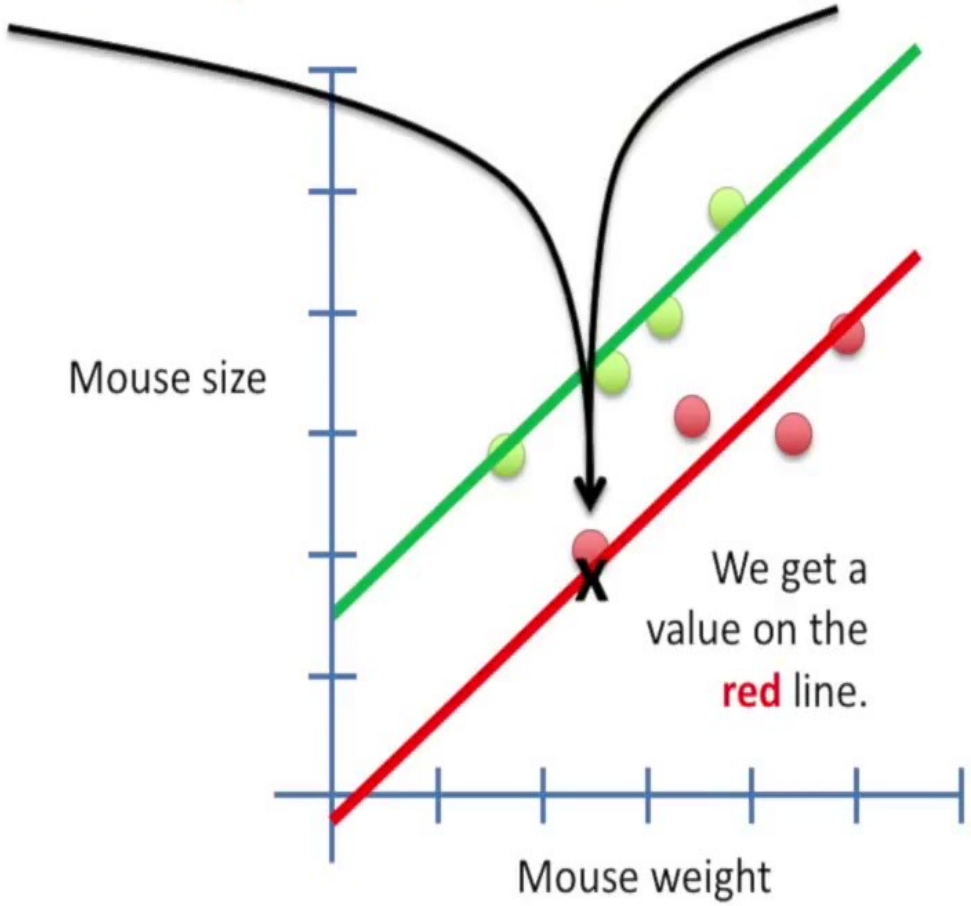


Let's focus on the first row...

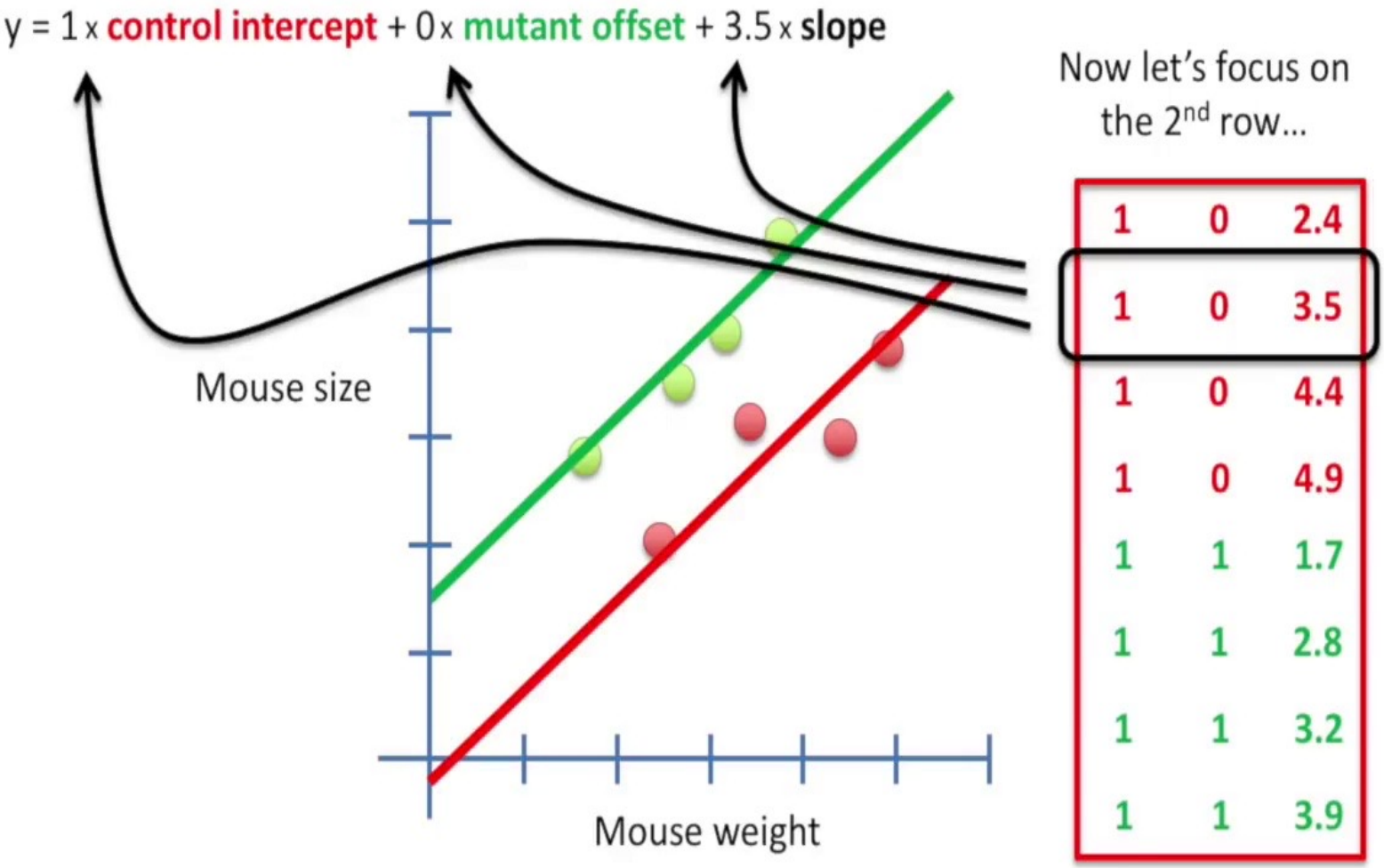
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9


y = 1 x control intercept + 0 x mutant offset + 2.4 x slope
...plug in the numbers...
Mouse size
Mouse weight
1 0 2.4
1 0 3.5
1 0 4.4
1 0 4.9
1 1 1.7
1 1 2.8
1 1 3.2
1 1 3.9

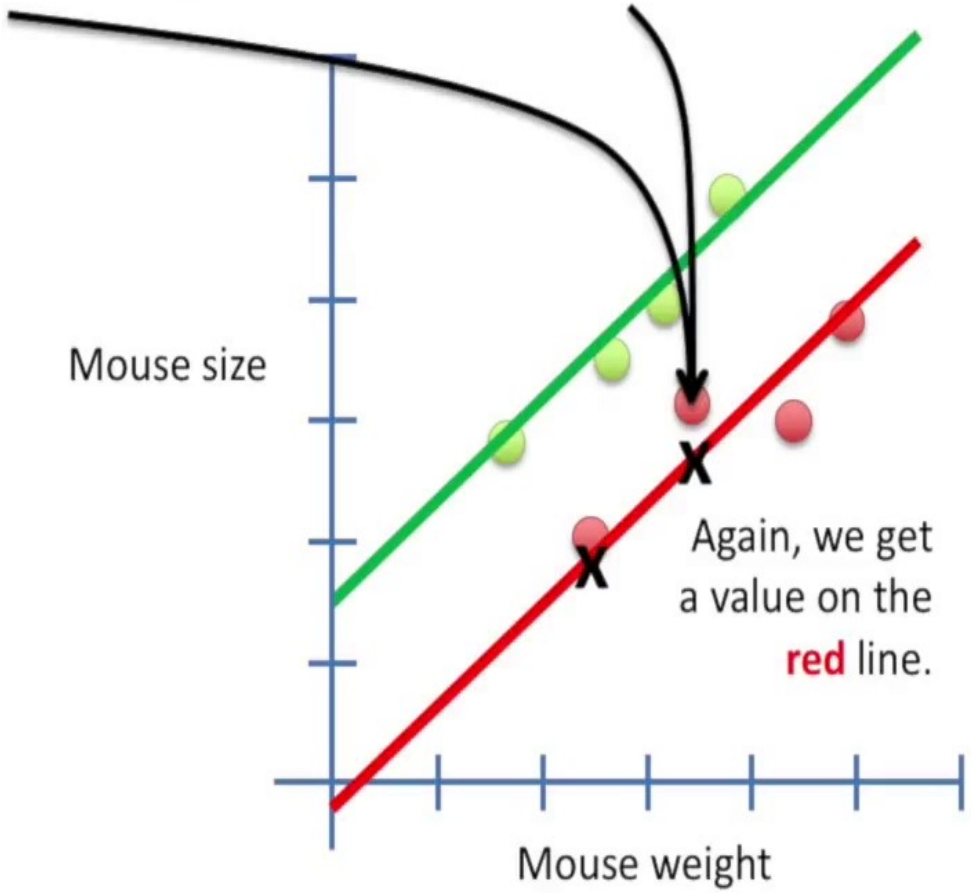
$y = 1 \times$ **control intercept** $+ 0 \times$ **mutant offset** $+ 2.4 \times$ **slope**



1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

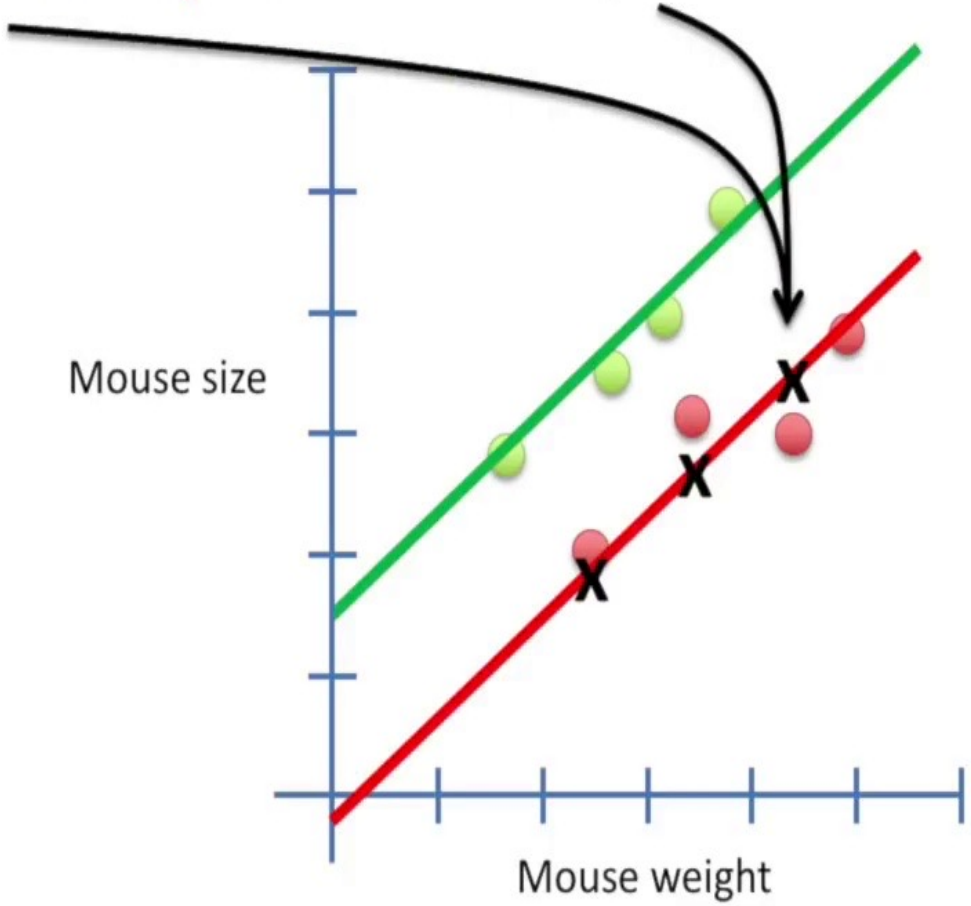

y = 1 x control intercept + 0 x mutant offset + 3.5 x slope
Mouse size
Mouse weight
Now let's focus on the 2nd row...
1 0 2.4
1 0 3.5
1 0 4.4
1 0 4.9
1 1 1.7
1 1 2.8
1 1 3.2
1 1 3.9

y = control intercept + mutant offset + slope



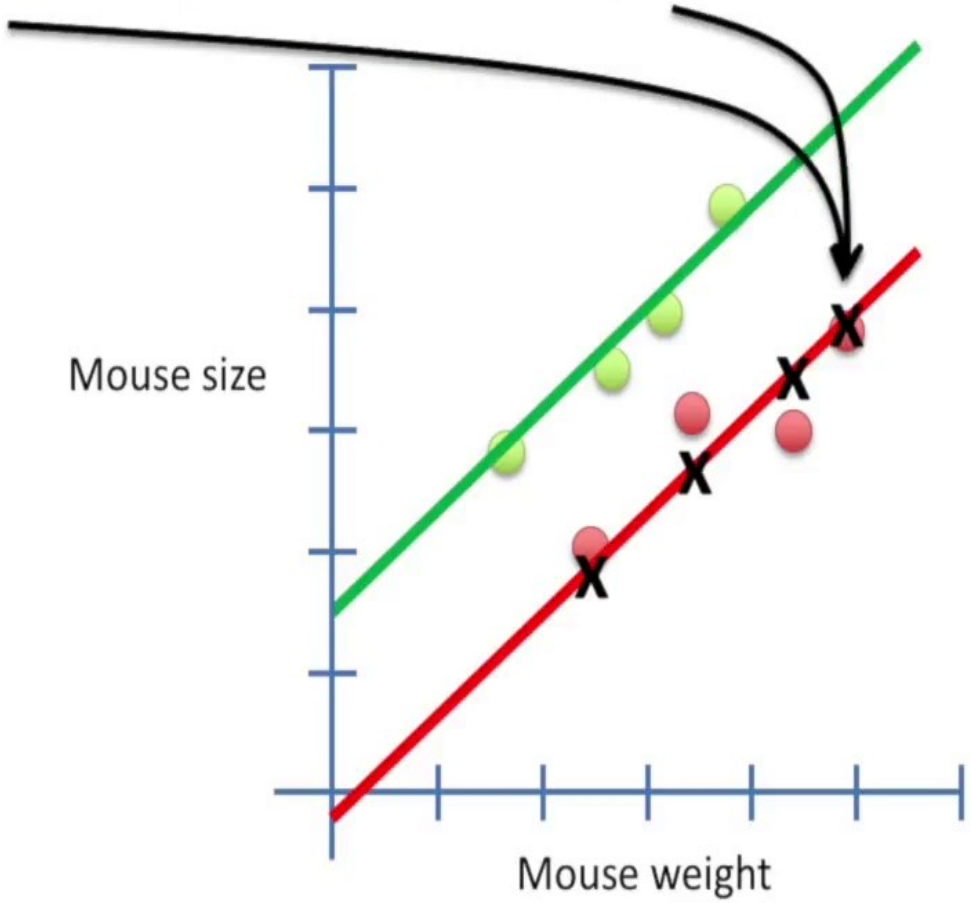
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope



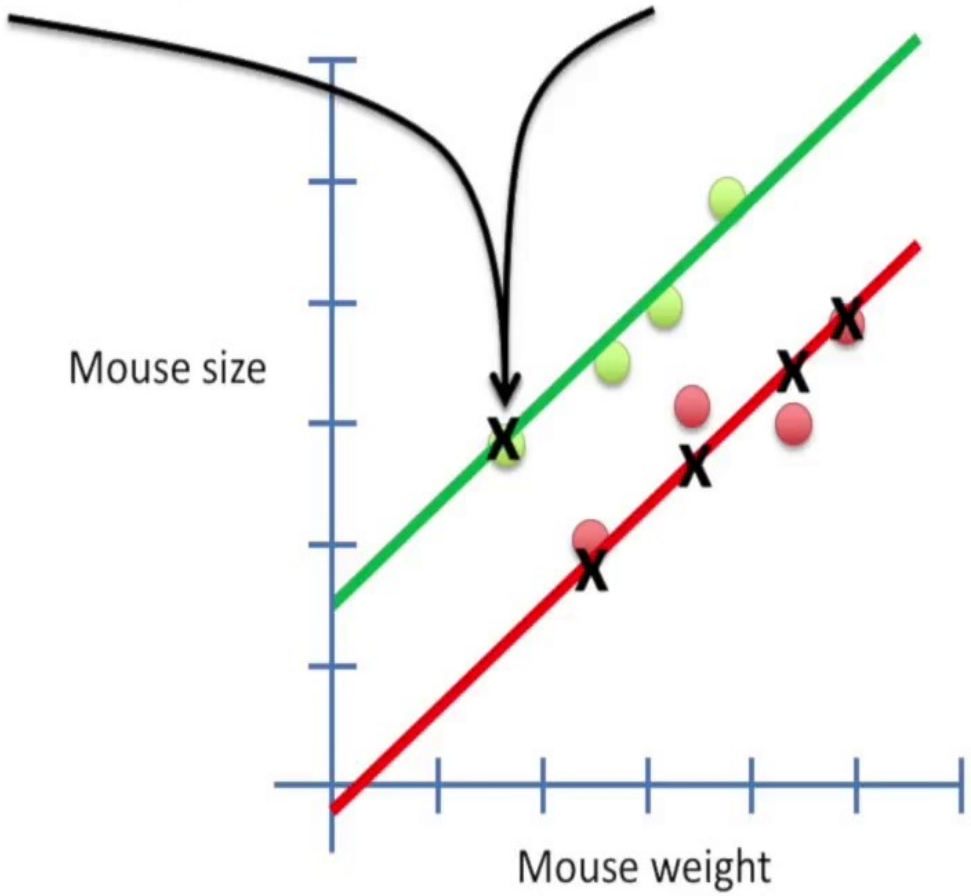
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope



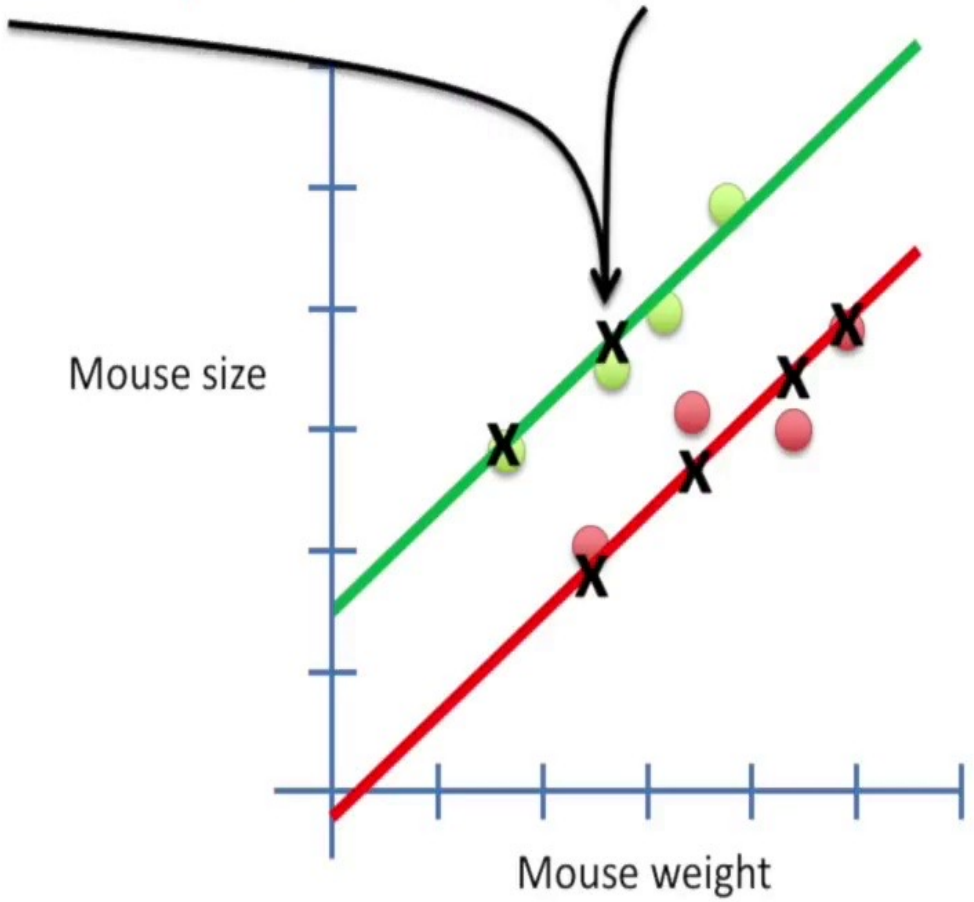
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope



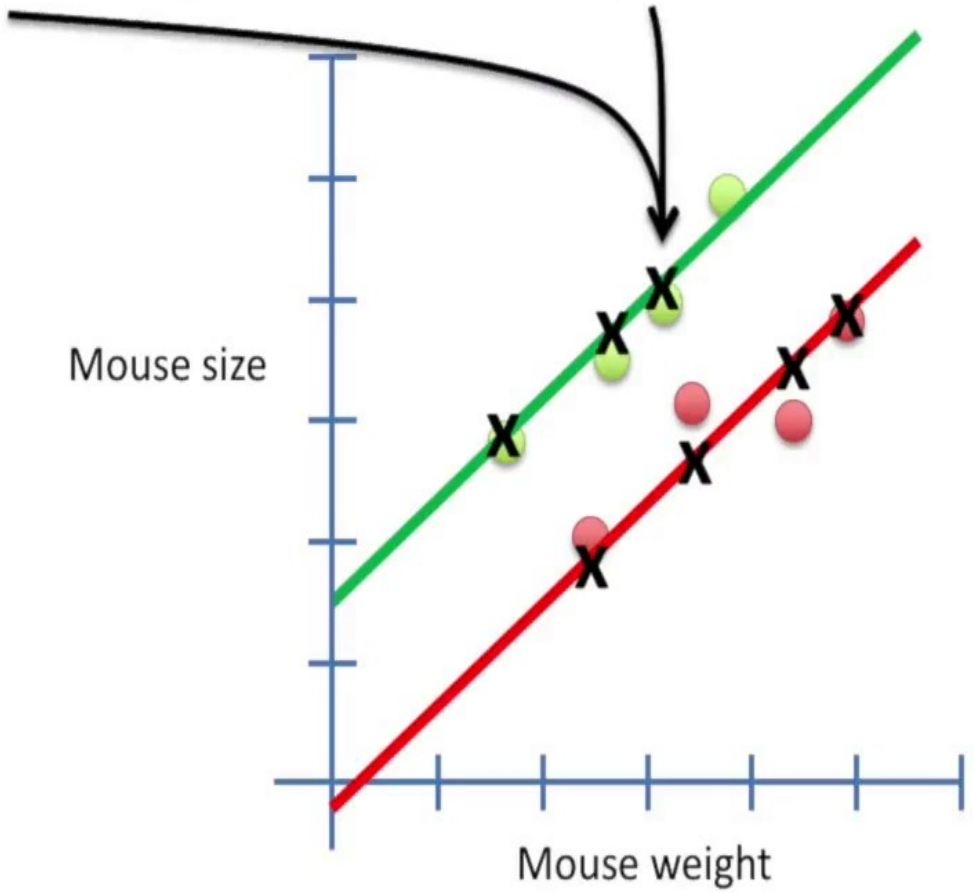
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope



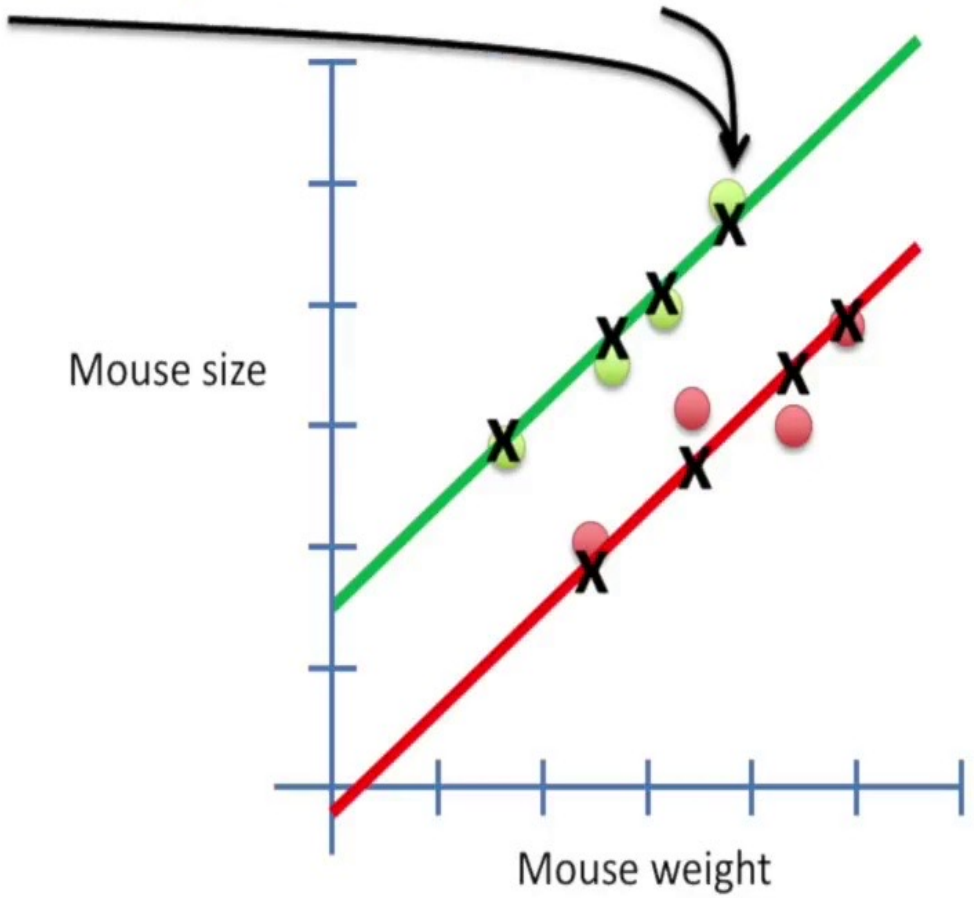
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope



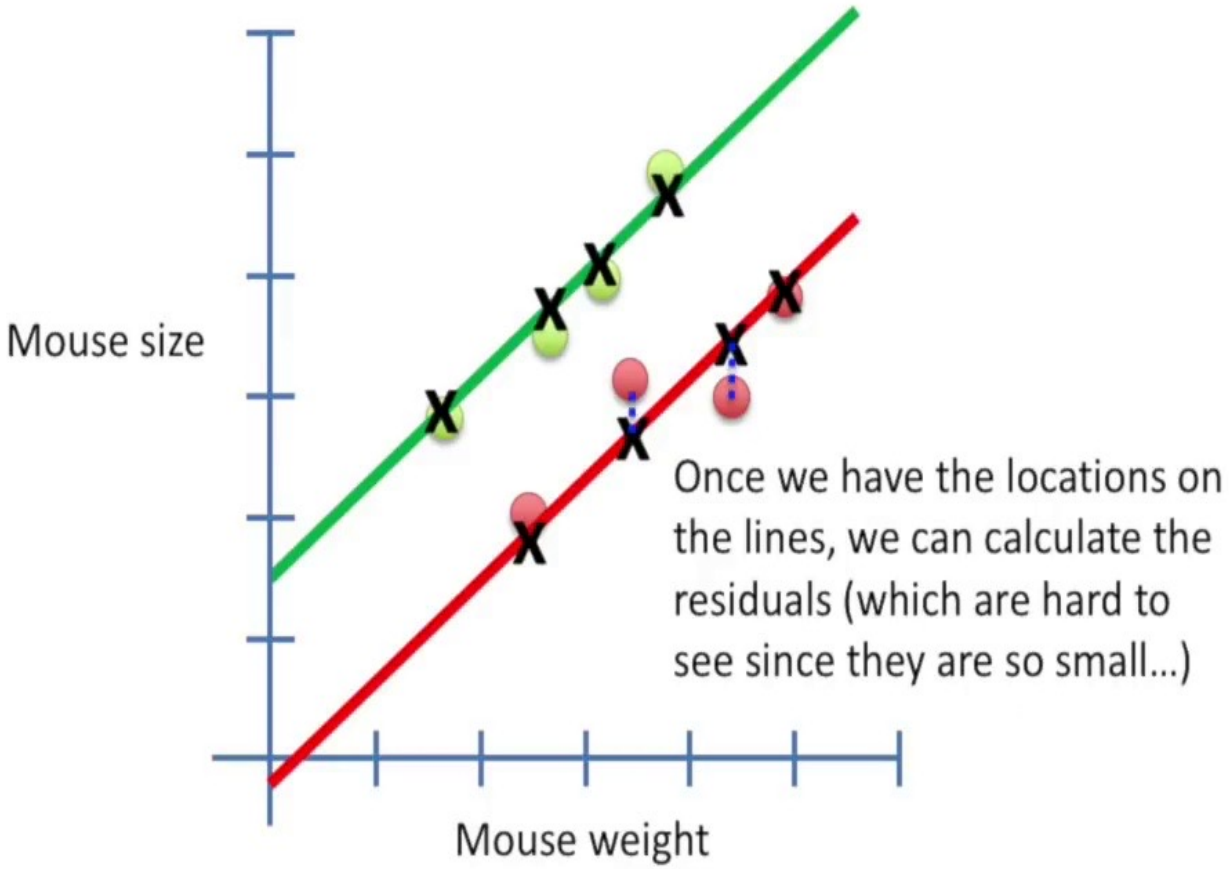
1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope

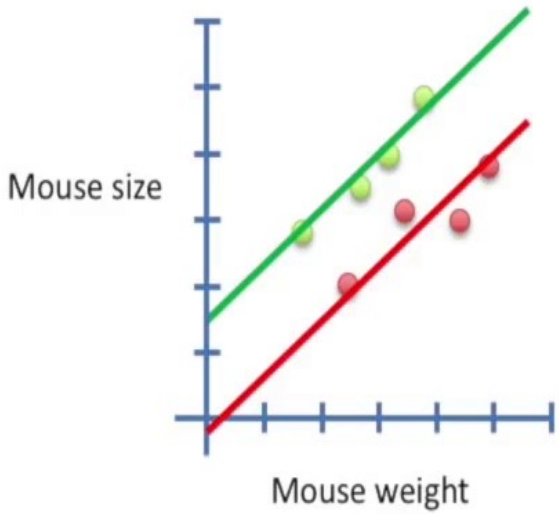


1	0	2.4
1	0	3.5
1	0	4.4
1	0	4.9
1	1	1.7
1	1	2.8
1	1	3.2
1	1	3.9

y = control intercept + mutant offset + slope

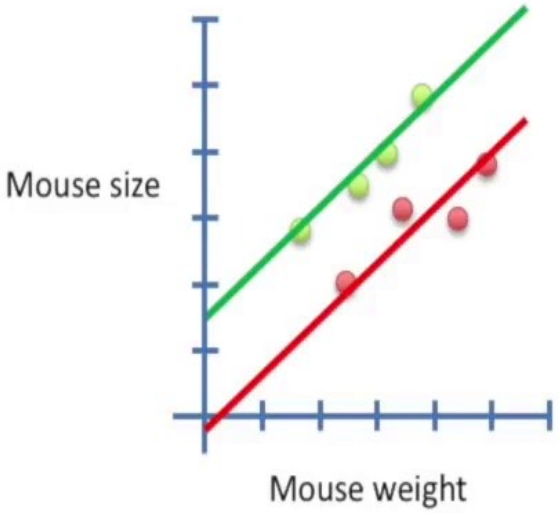


y = control intercept + mutant offset + slope



Now compare the fancy model...

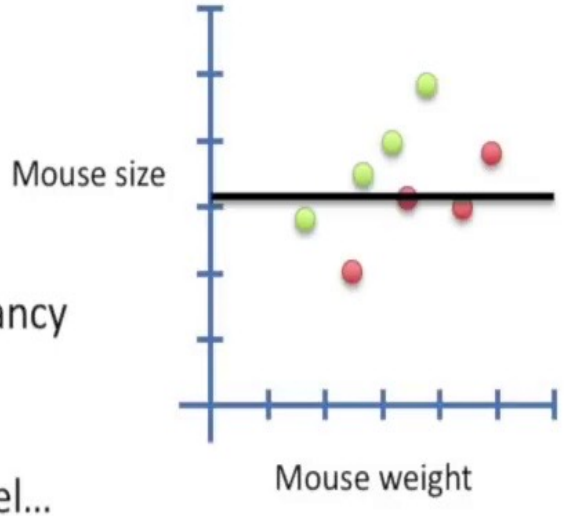
y = control intercept + mutant offset + slope



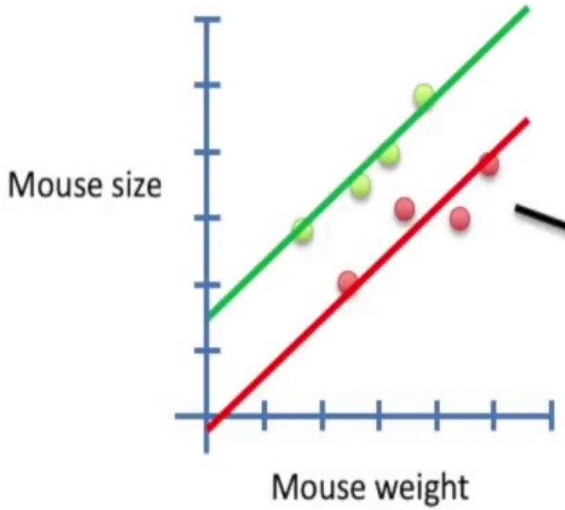
Now compare the fancy model...

...to a simpler model...

y = overall mean

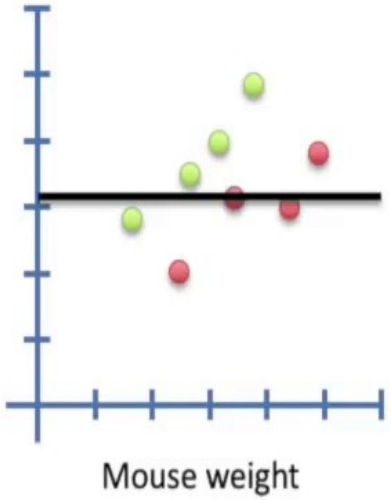


y = **control intercept** + **mutant offset** + **slope**



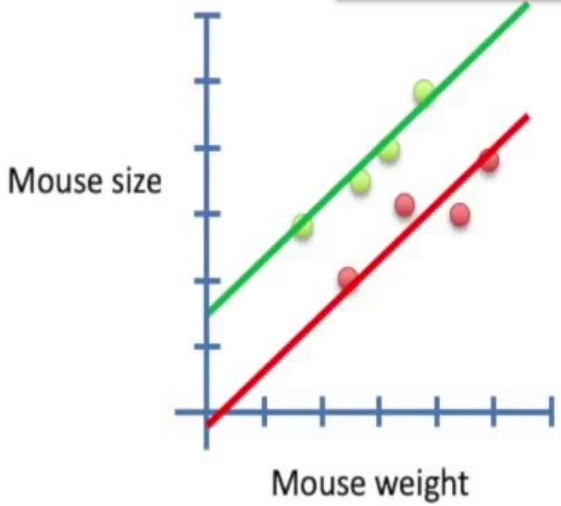
Plug in the sum of squares of the residuals for the fancy model...

y = **overall mean**



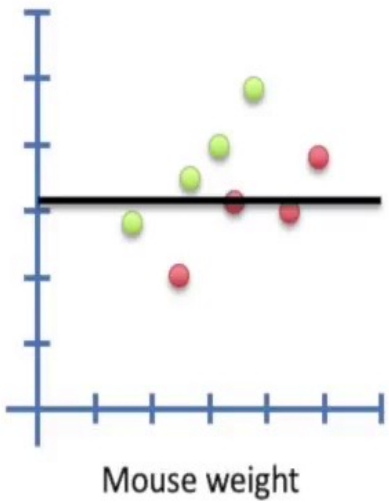
$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})}$$

y = control intercept + mutant offset + slope



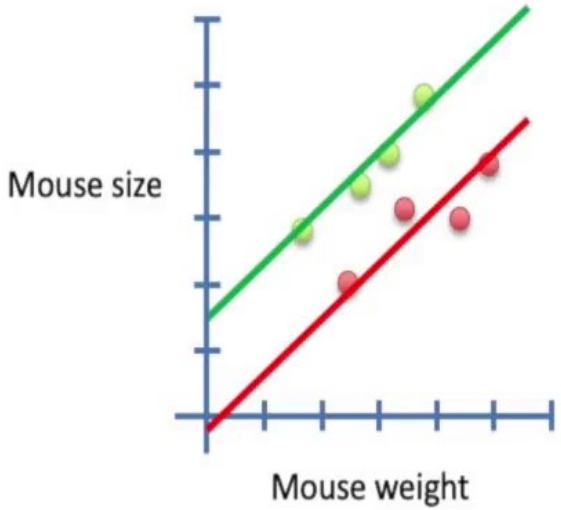
Plug in 3 for p_{fancy}, since there are 3 parameters.

y = overall mean



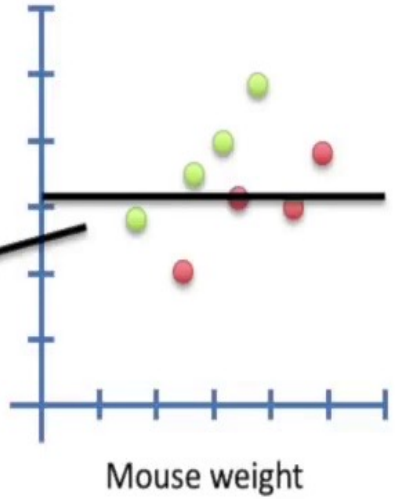
$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})}$$

y = **control intercept** + **mutant offset** + **slope**



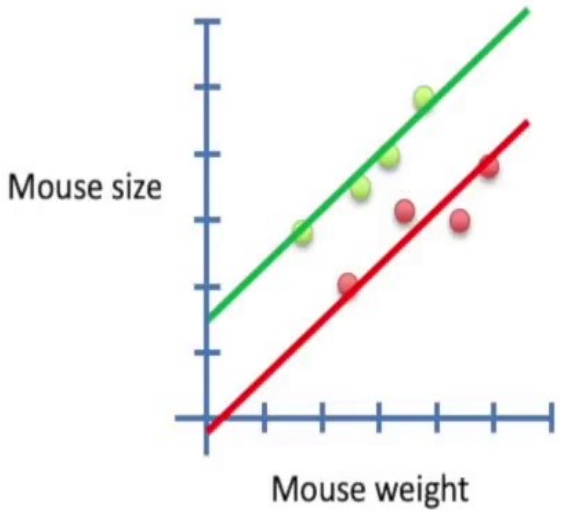
Plug in the sum of squares of the residuals for the simple model...

y = **overall mean**



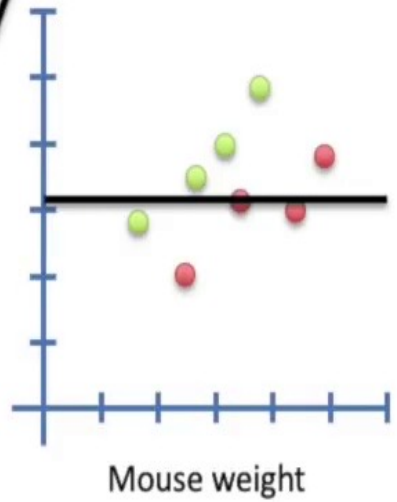
$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})}$$

y = **control intercept** + **mutant offset** + **slope**



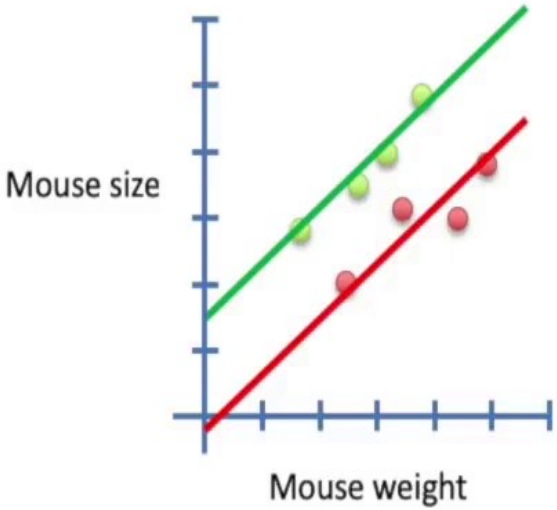
And plug in 1 for p_{simple}, since there is only one parameter in the simple equation.

y = **overall mean**

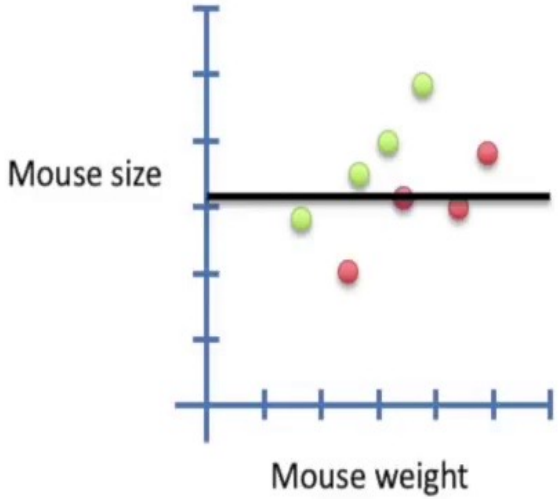


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})}$$

y = **control intercept** + **mutant offset** + **slope**

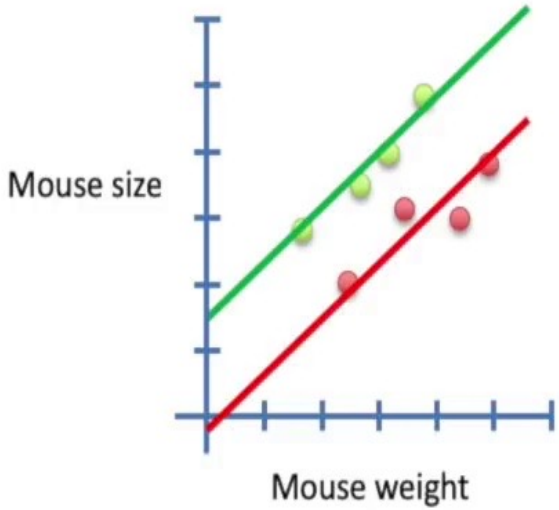


y = **overall mean**

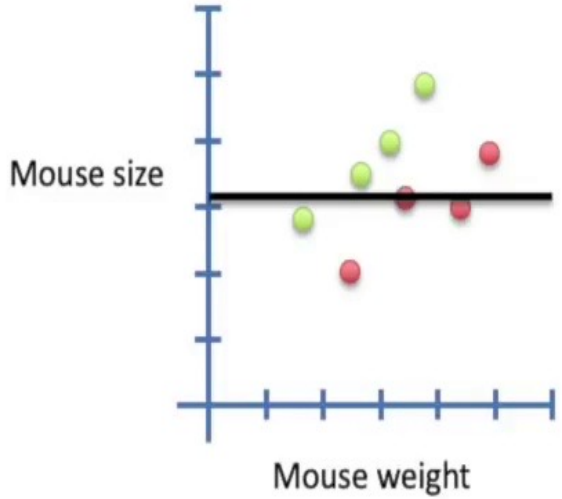


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})} = 21.88$$

y = **control intercept** + **mutant offset** + **slope**



y = **overall mean**

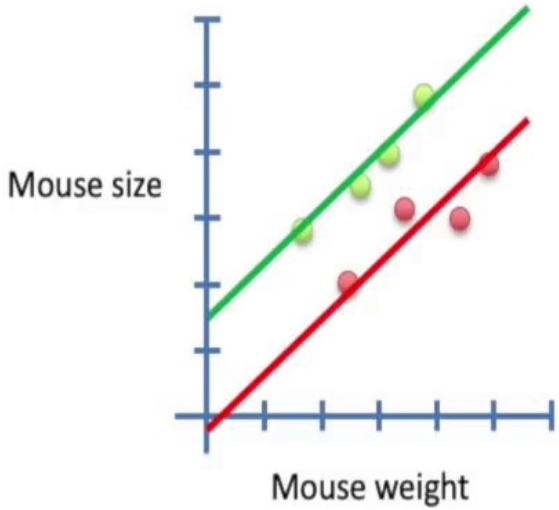


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})} = 21.88$$

BAM!!!

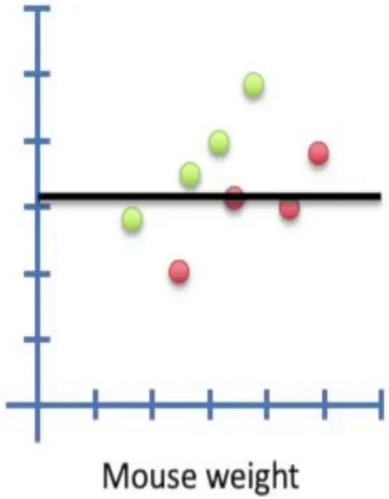
p-value = 0.003

y = **control intercept** + **mutant offset** + **slope**



The small p-value says that taking weight and mouse type into account is significantly better at predicting size than just using the average size.

y = **overall mean**

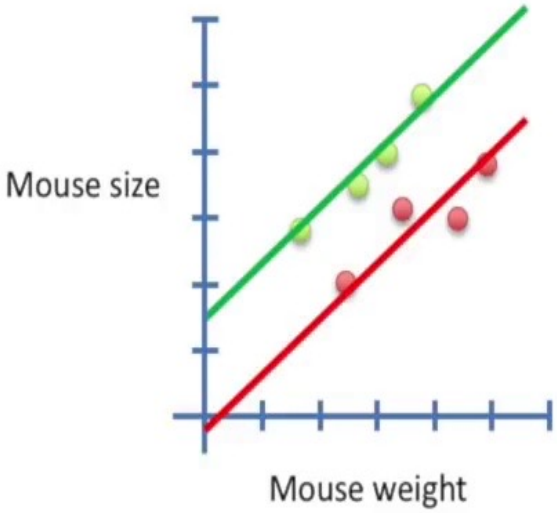


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})} = 21.88$$

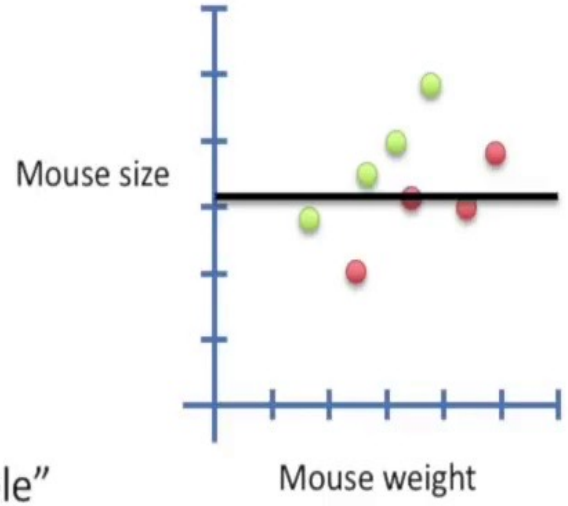
BAM!!!

p-value = 0.003

y = **control intercept** + **mutant offset** + **slope**

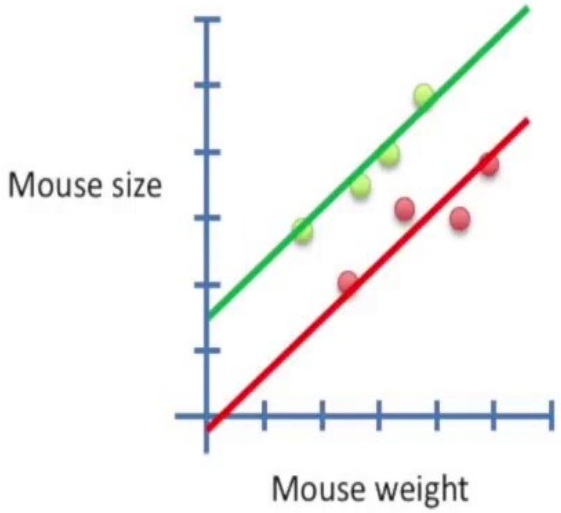


y = **overall mean**

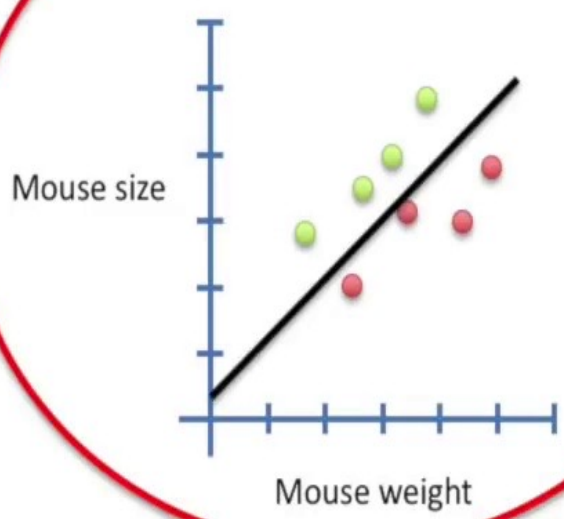


NOTE! The "simple" model can be *any* simpler model.

y = control intercept + mutant offset + slope

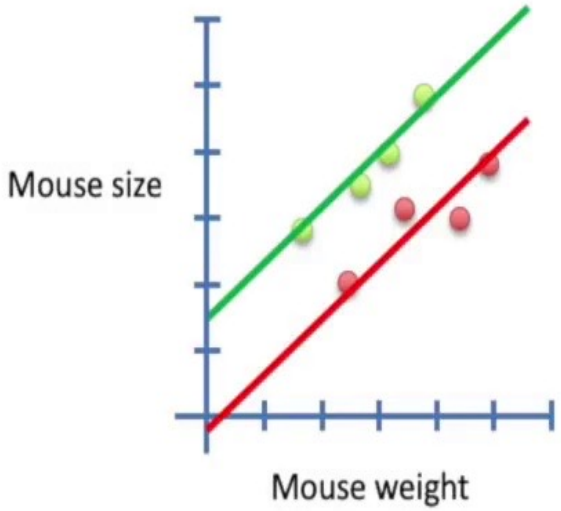


y = intercept + slope



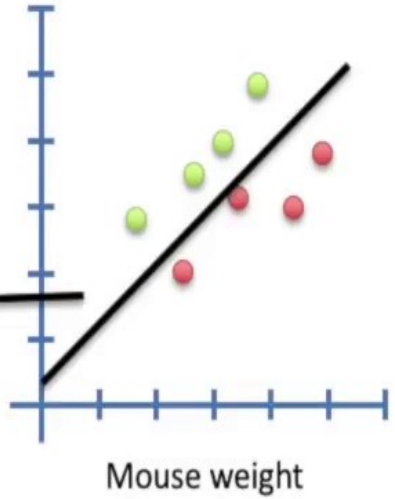
This model takes weight into account, but ignores the fact that some mice are normal and others are mutants.

y = **control intercept** + **mutant offset** + **slope**



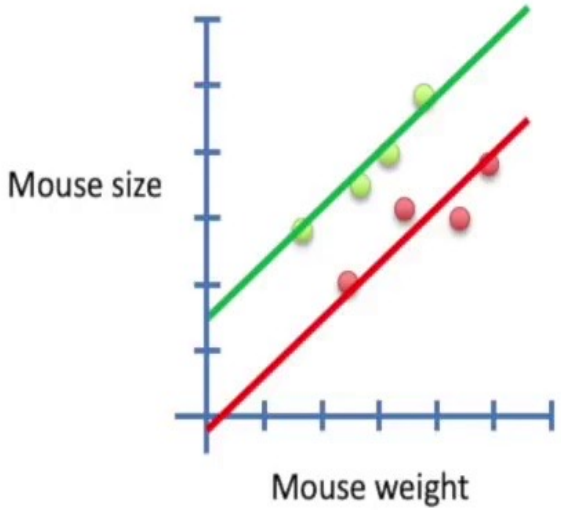
Plug in the sum of squares of the residuals, just like before.

y = **intercept** + **slope**



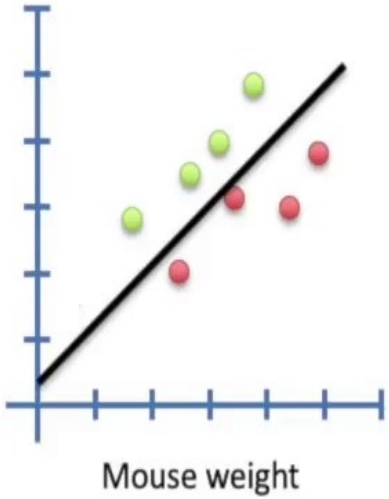
$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})}$$

y = **control intercept** + **mutant offset** + **slope**



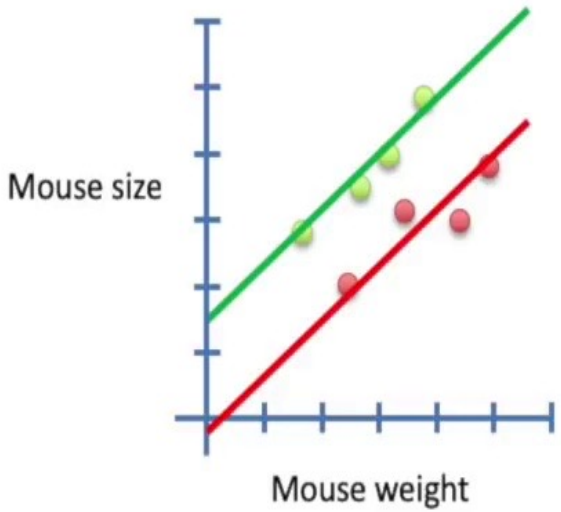
This equation has 2 parameters, so $p_{simple} = 2$

y = **intercept** + **slope**

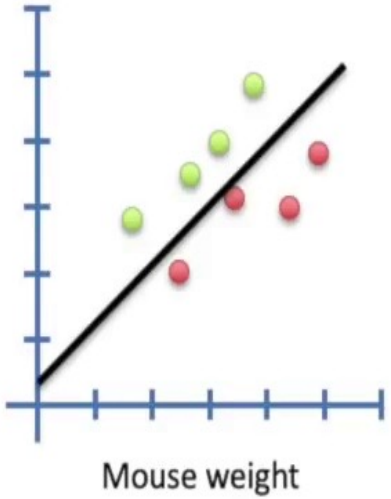


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})}$$

y = **control intercept** + **mutant offset** + **slope**



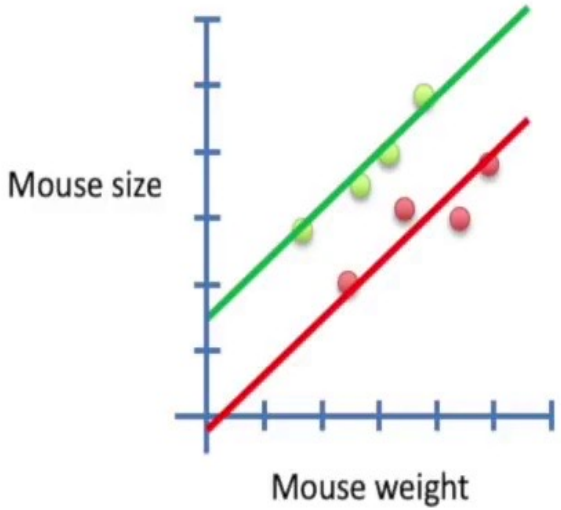
y = **intercept** + **slope**



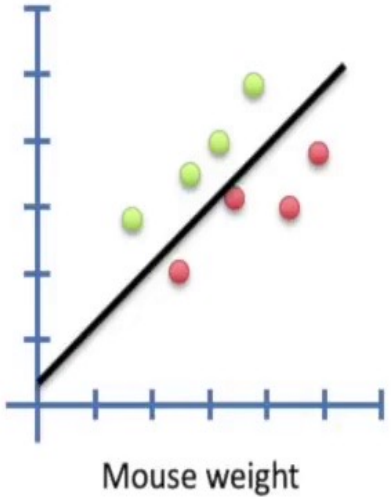
$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})} = 32.6$$

p-value = 0.0023

y = **control intercept** + **mutant offset** + **slope**



y = **intercept** + **slope**

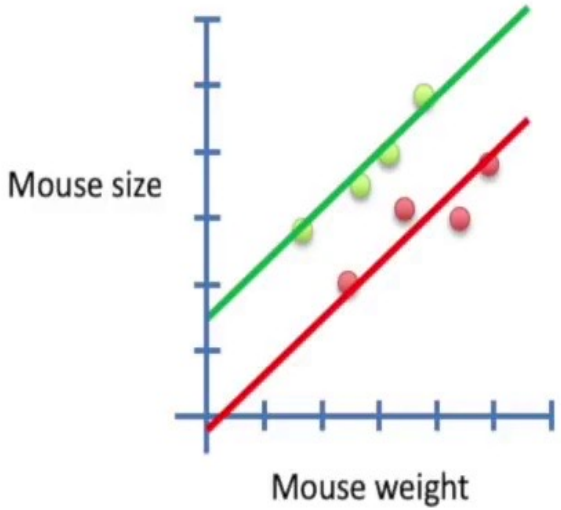


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})} = 32.6$$

Double BAM!

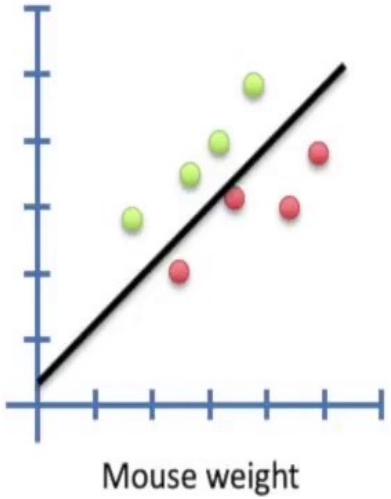
p-value = 0.0023

y = **control intercept** + **mutant offset** + **slope**



This small p-value suggests that using both weight and mouse type is better at predicting mouse size than weight alone.

y = **intercept** + **slope**

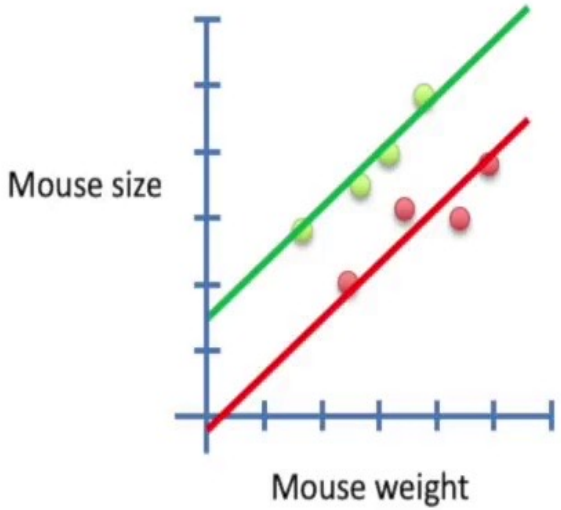


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})} = 32.6$$

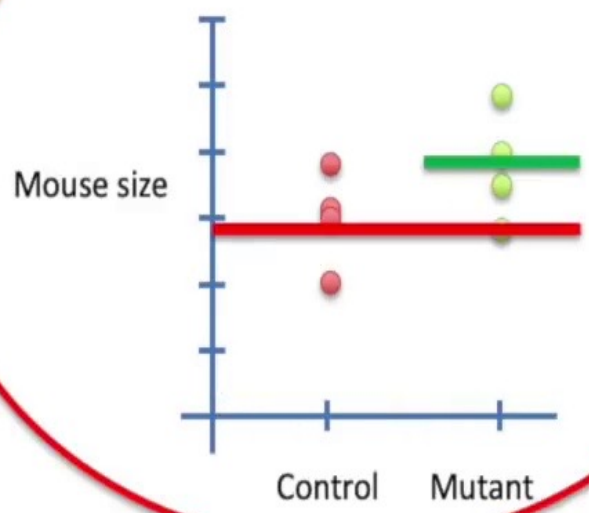
Double BAM!

p-value = 0.0023

y = control intercept + mutant offset + slope



y = $\text{mean}_{\text{control}}$ + difference

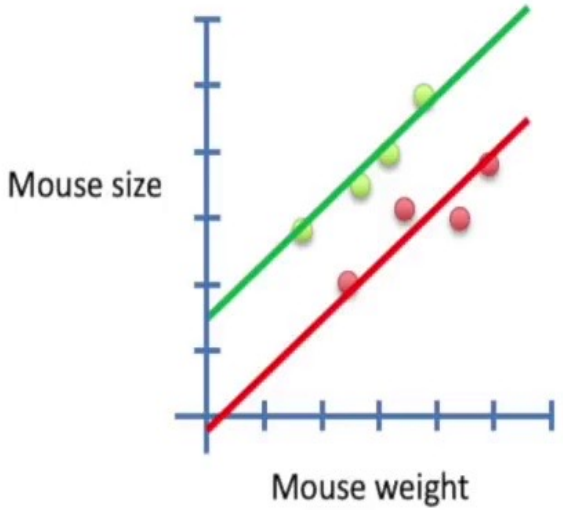


Here's another simple model.

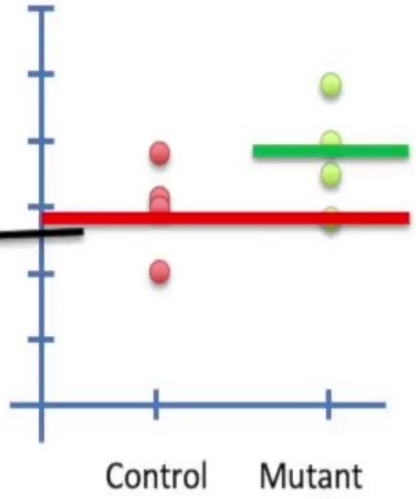
This model ignores mouse weight.

y = control intercept + mutant offset + slope

y = $\text{mean}_{\text{control}}$ + difference

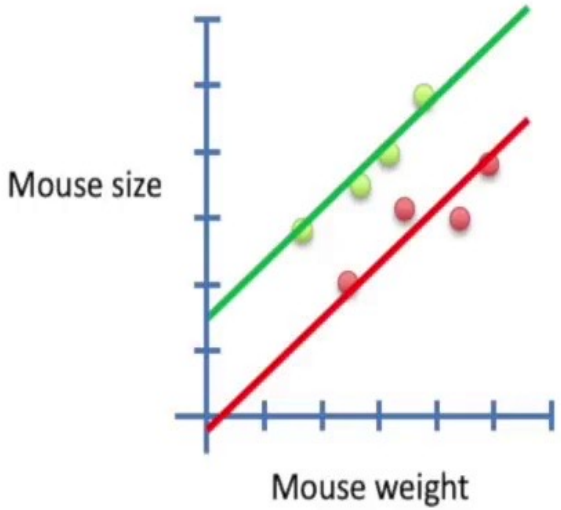


Again, plug in the sum of squares of the residuals.



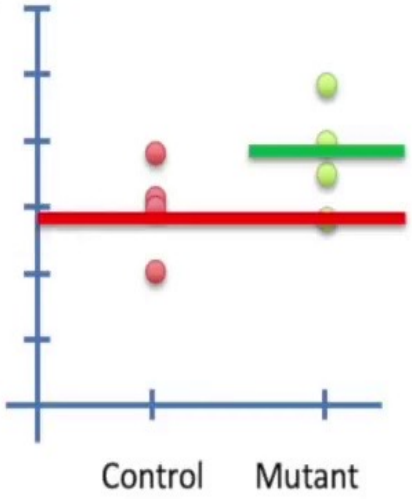
$$F = \frac{SS(\text{simple}) - SS(\text{fancy}) \,/\, (p_{\text{fancy}} - p_{\text{simple}})}{SS(\text{fancy}) \,/\, (n - p_{\text{fancy}})}$$

y = control intercept + mutant offset + slope



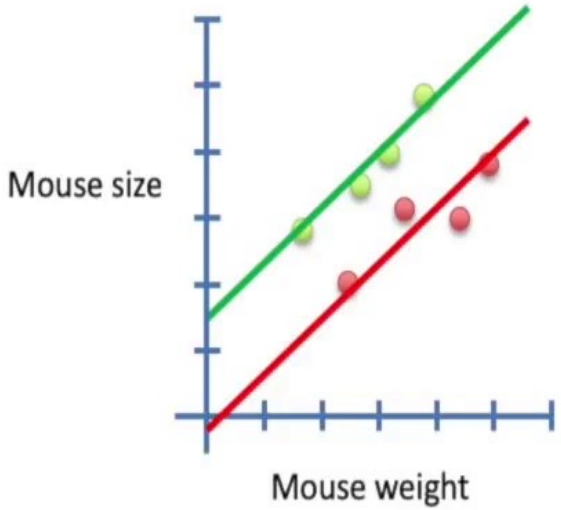
The equation has 2 parameters, so $p_{simple} = 2$

y = $mean_{control}$ + difference

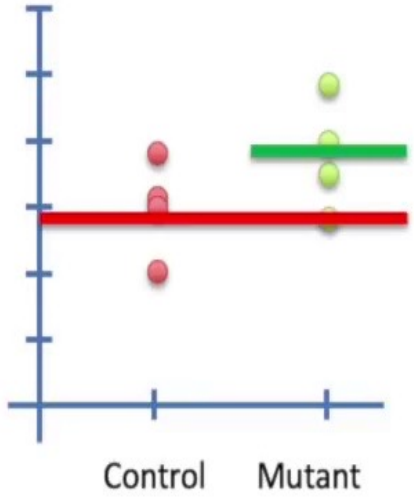


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})}$$

y = control intercept + mutant offset + slope



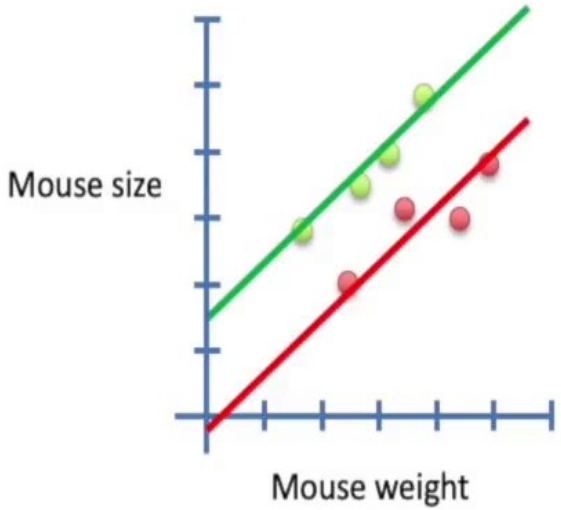
y = $\text{mean}_{\text{control}}$ + difference



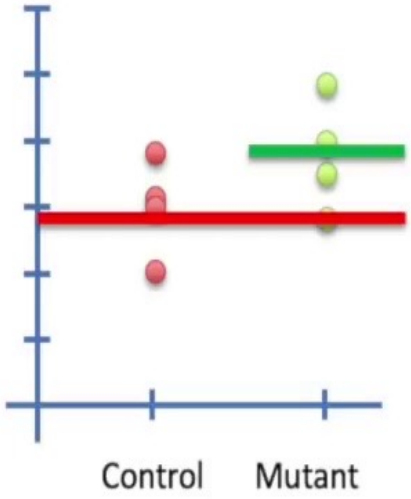
$$F = \frac{SS(\text{simple}) - SS(\text{fancy}) \,/\, (p_{\text{fancy}} - p_{\text{simple}})}{SS(\text{fancy}) \,/\, (n - p_{\text{fancy}})} = 31.06$$

p-value = 0.0025

y = **control intercept** + **mutant offset** + **slope**



y = **mean**$_{control}$ + **difference**

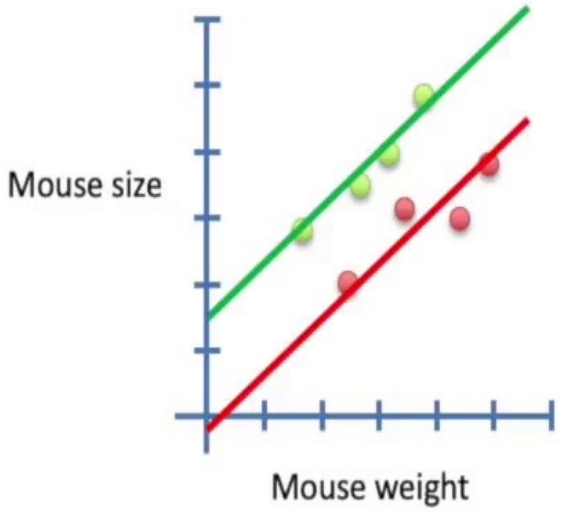


$$F = \frac{SS(simple) - SS(fancy) / (p_{fancy} - p_{simple})}{SS(fancy) / (n - p_{fancy})} = 31.06$$

TRIPLE BAM!!!

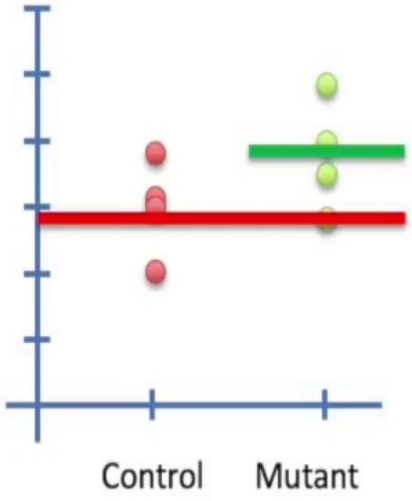
p-value = 0.0025

y = control intercept + mutant offset + slope



This small p-value suggests that using mouse weight and type is better at predicting mouse size than mouse type alone.

y = $\text{mean}_{\text{control}}$ + difference

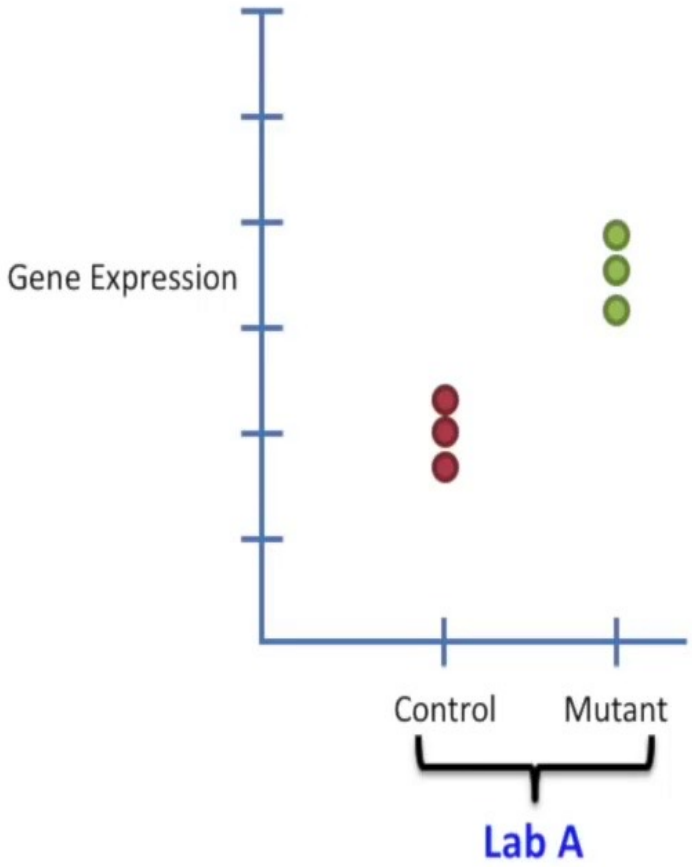


$$F = \frac{SS(\text{simple}) - SS(\text{fancy}) / (p_{\text{fancy}} - p_{\text{simple}})}{SS(\text{fancy}) / (n - p_{\text{fancy}})} = 31.06$$

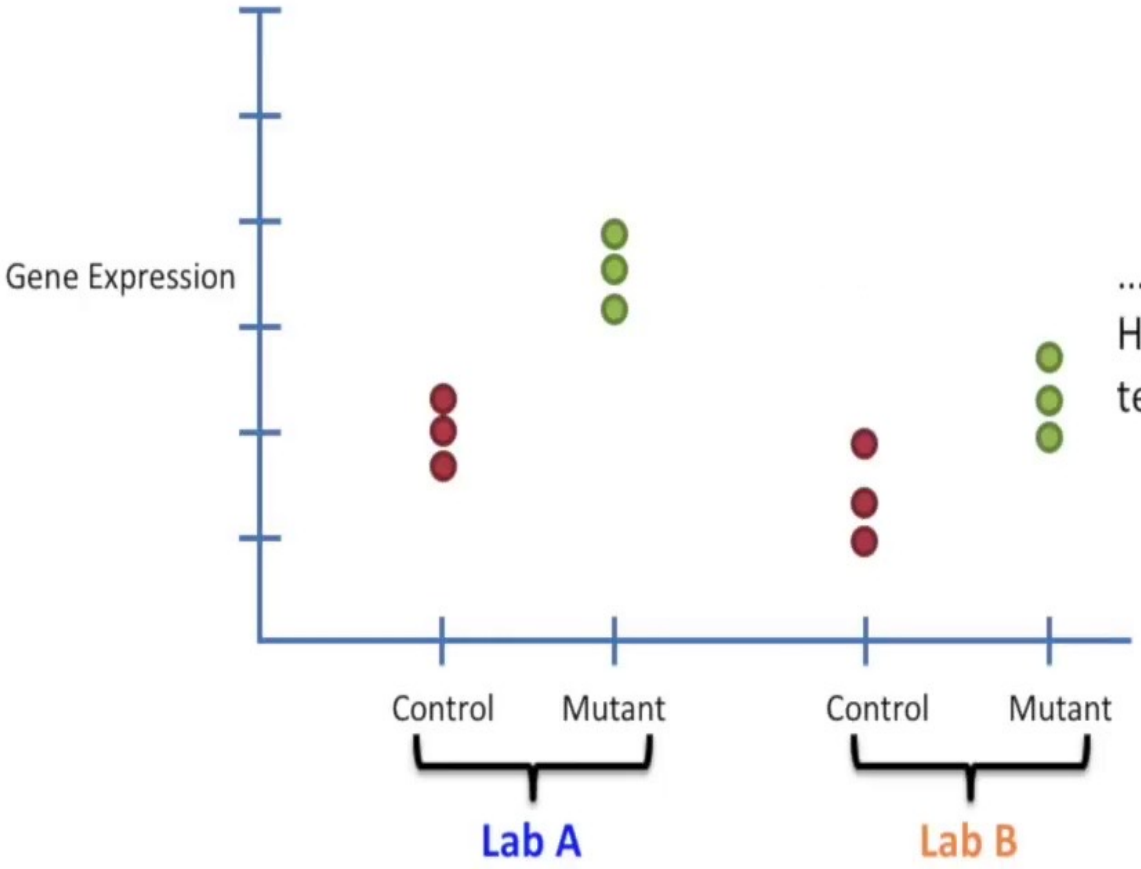
TRIPLE BAM!!!

p-value = 0.0025

One last example...

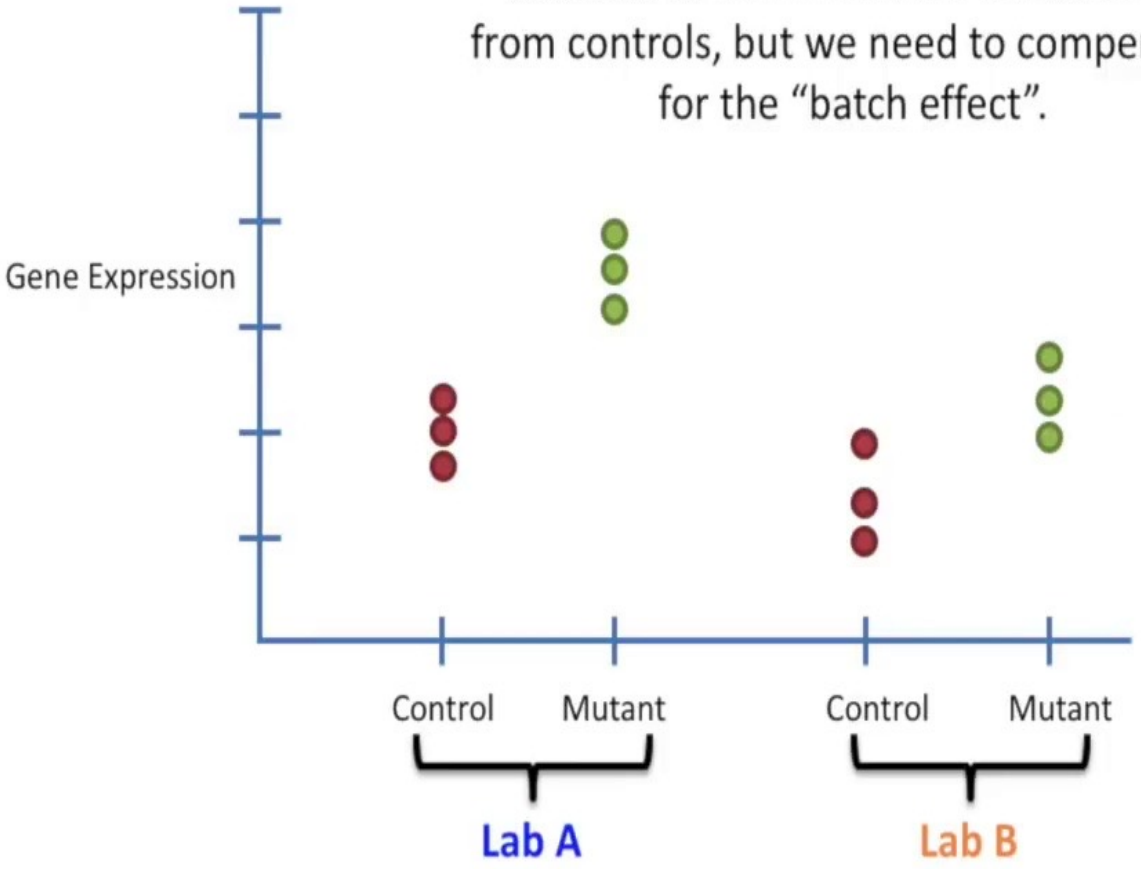


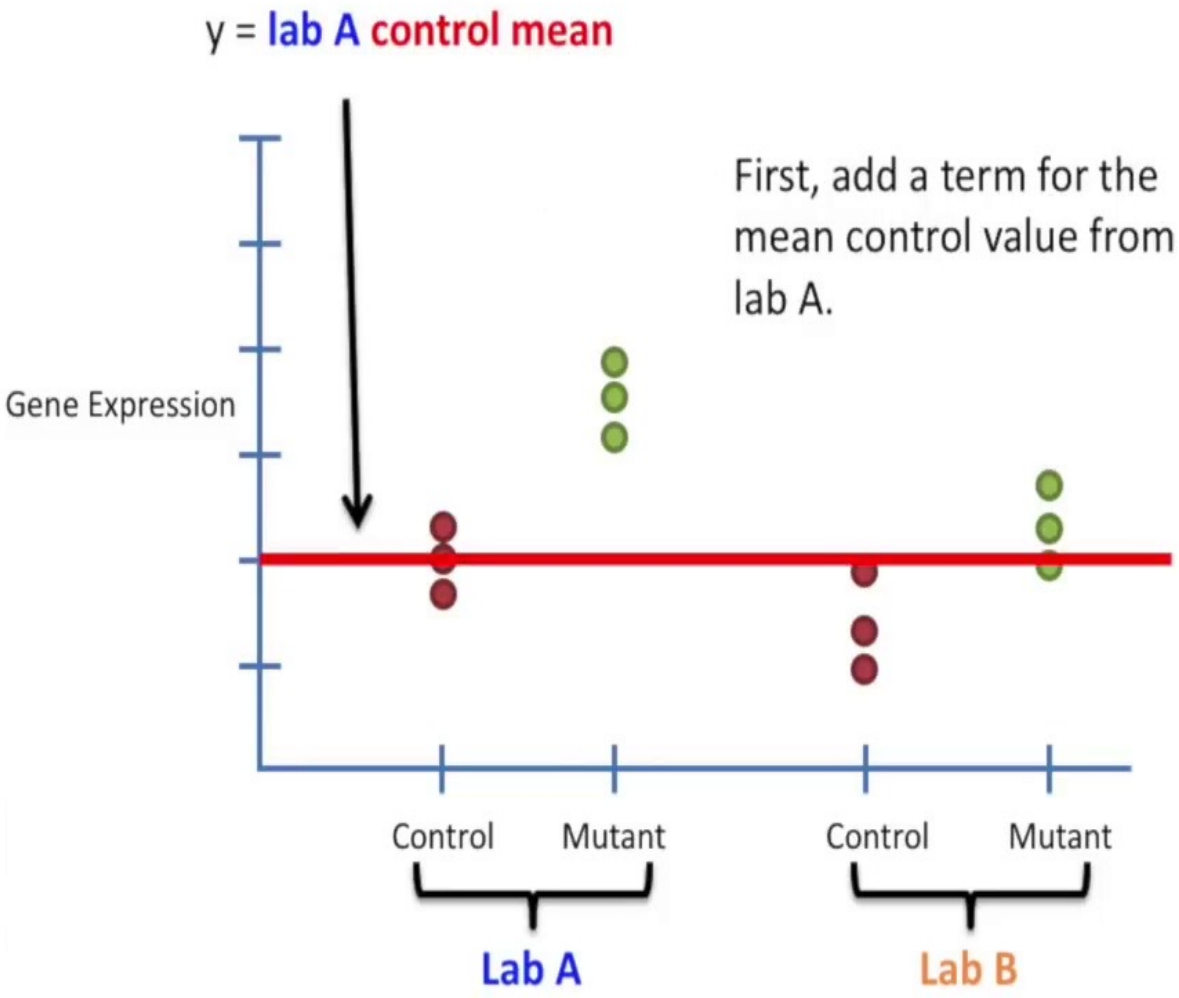
Lab A does an experiment...

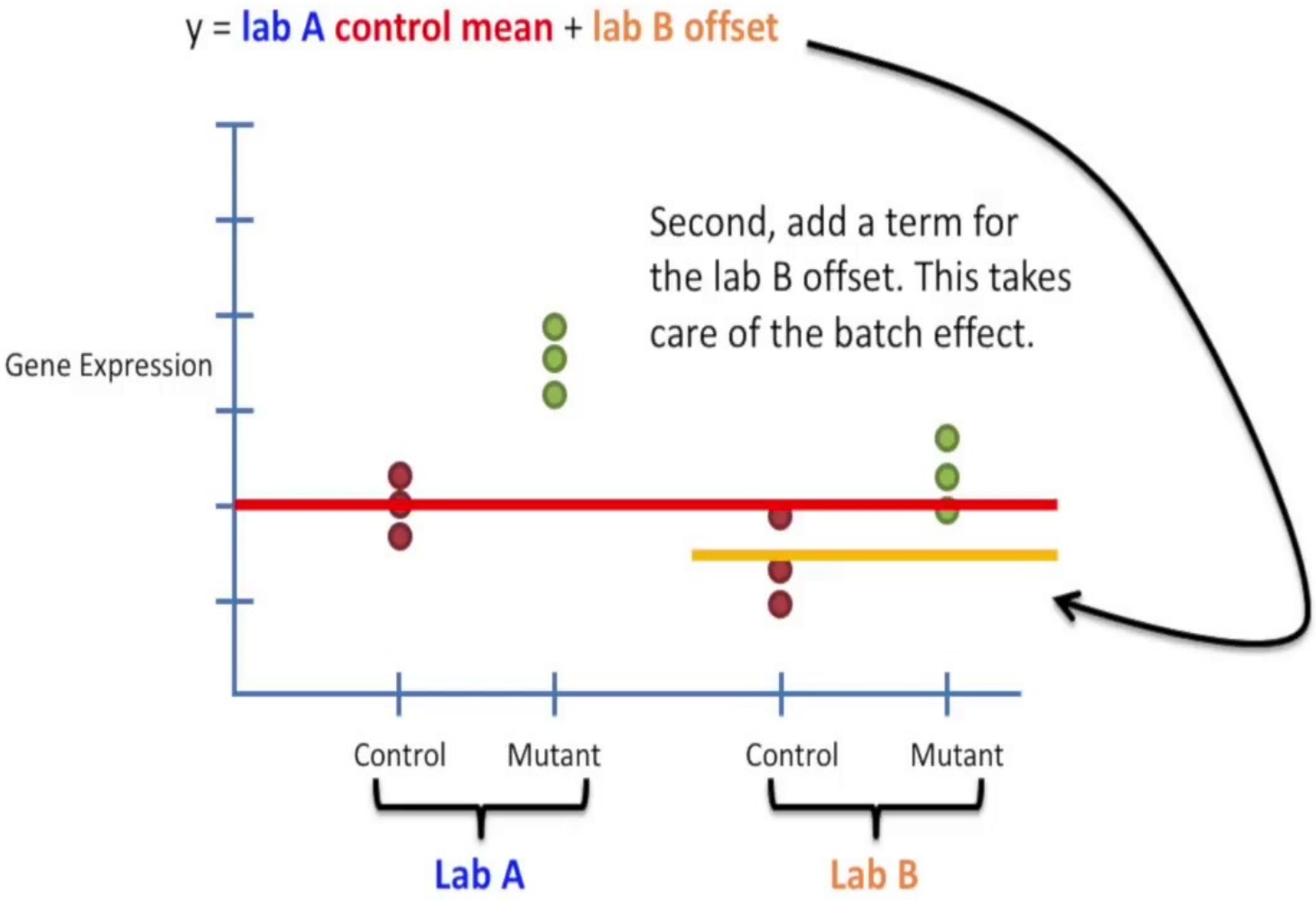


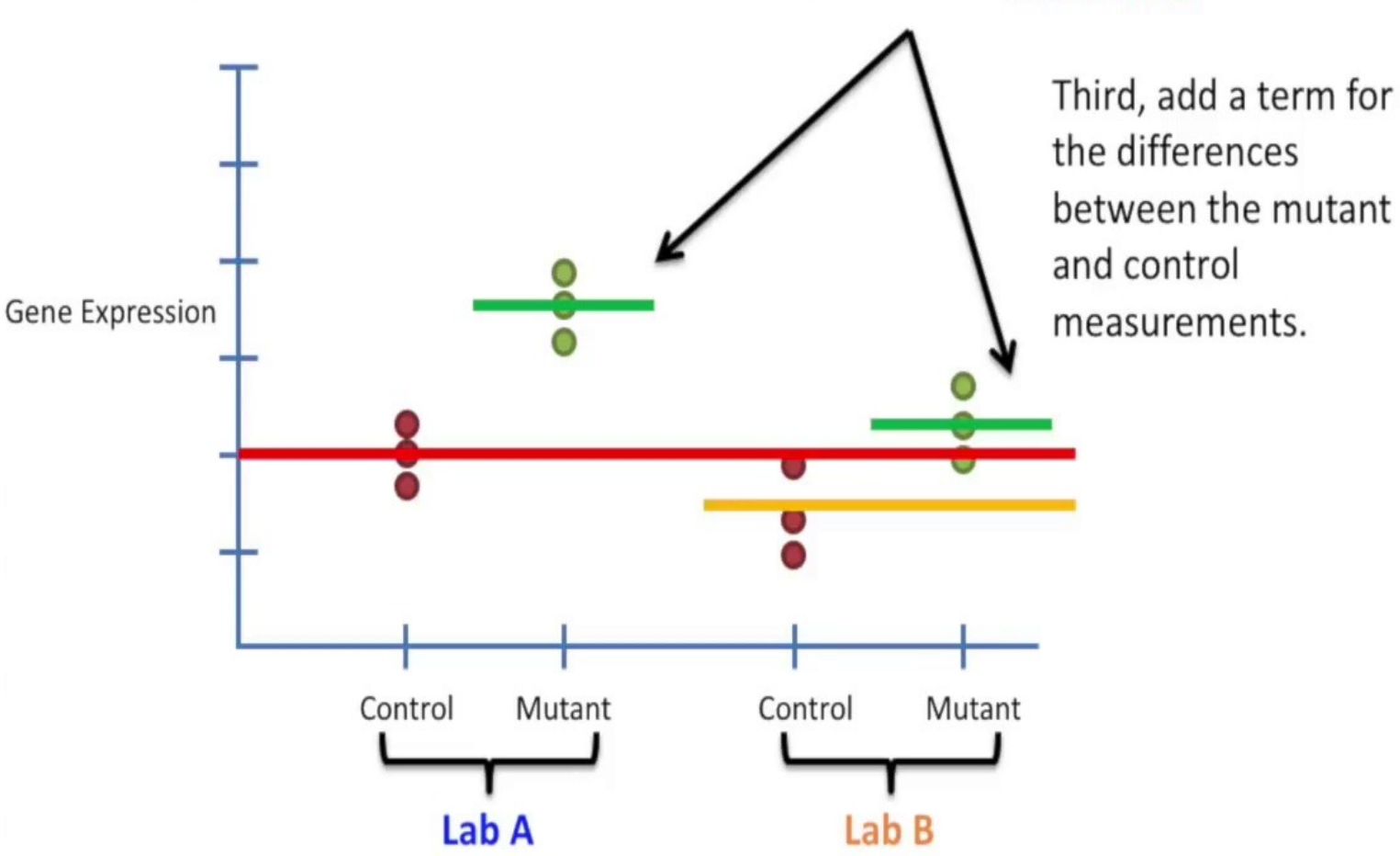
...then Lab B replicates it.
However, their measurements tended to be smaller over all...

We would like to combine these two datasets to see if mutants are different from controls, but we need to compensate for the “batch effect”.

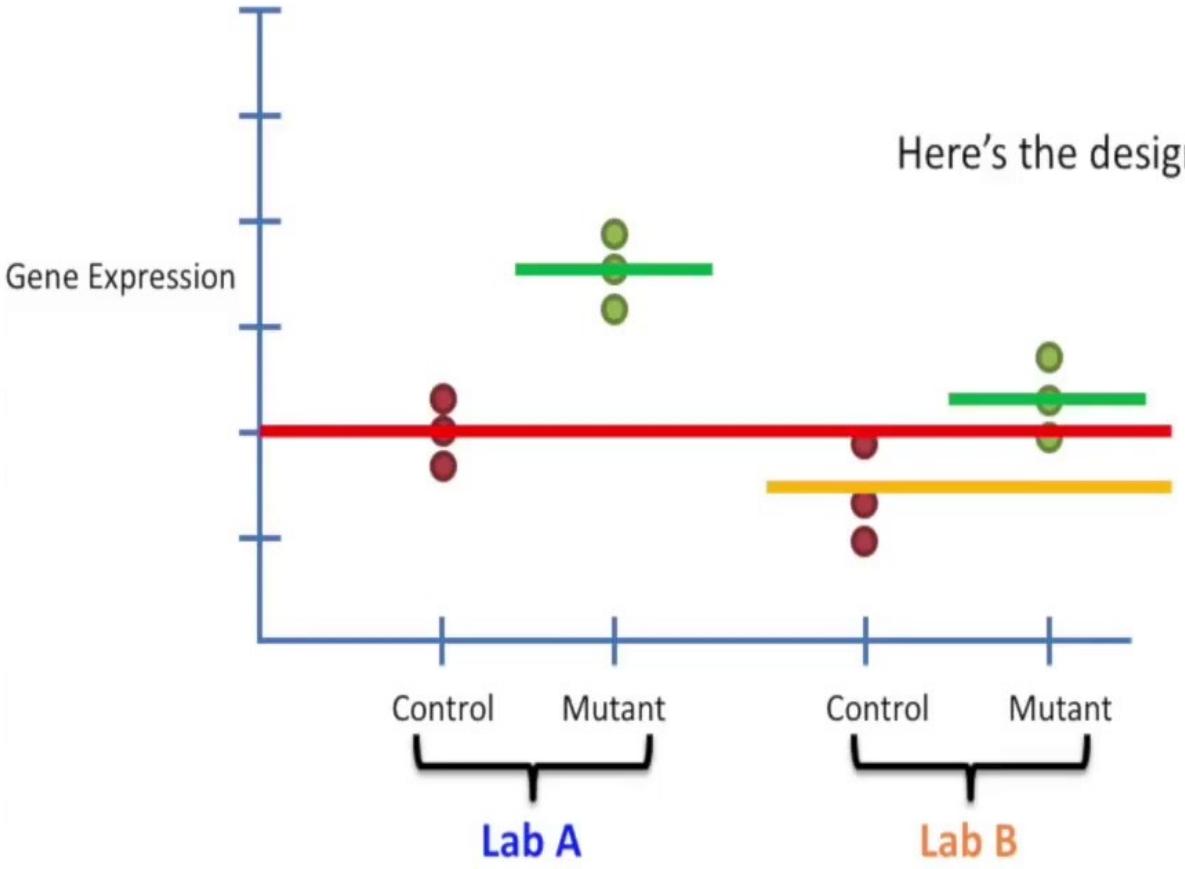



y = lab A control mean
First, add a term for the mean control value from lab A.
Gene Expression
Control
Mutant
Control
Mutant
Lab A
Lab B


y = lab A control mean + lab B offset
Second, add a term for the lab B offset. This takes care of the batch effect.
Gene Expression
Control
Mutant
Control
Mutant
Lab A
Lab B


y = lab A control mean + lab B offset + difference(mutant - control)
Third, add a term for the differences between the mutant and control measurements.
Gene Expression
Control
Mutant
Control
Mutant
Lab A
Lab B

y = lab A control mean + lab B offset + difference$_{(\text{mutant - control})}$

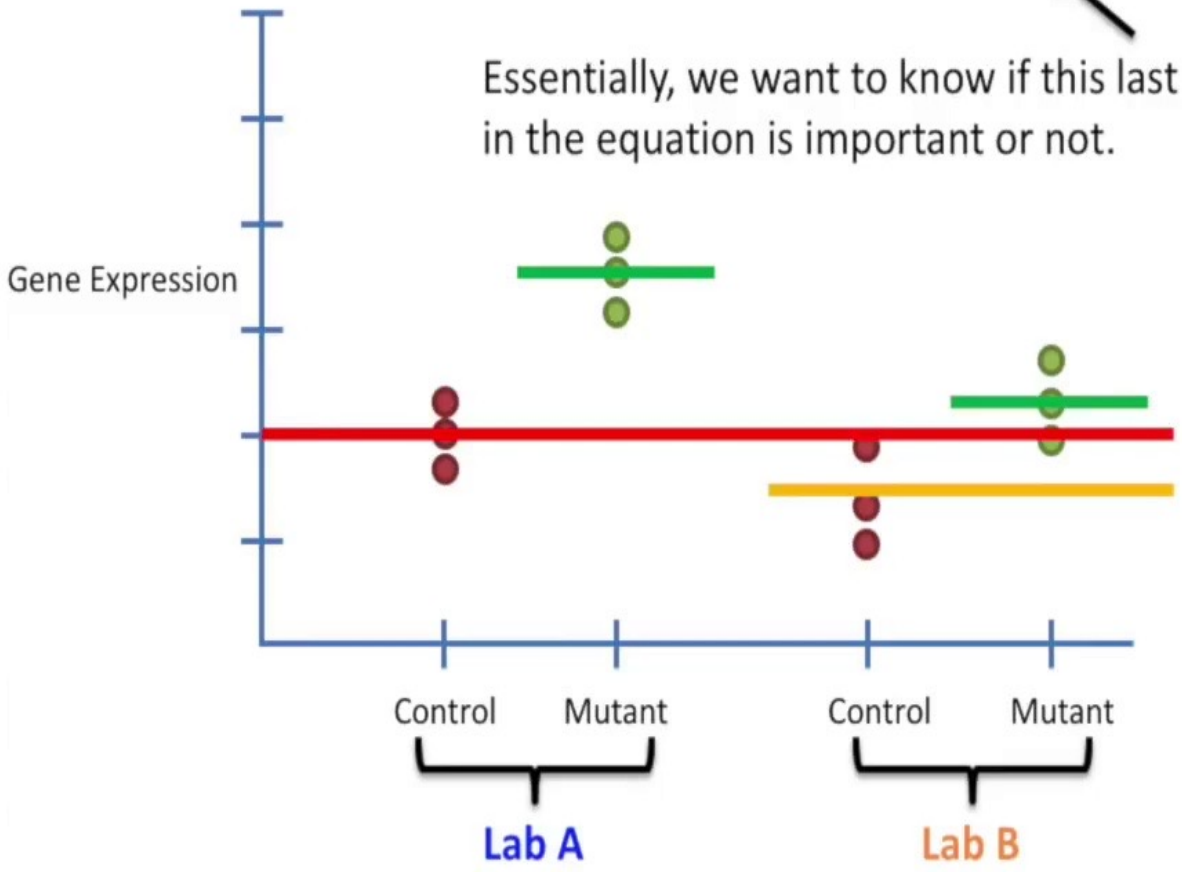


Here's the design matrix...

1	0	0
1	0	0
1	0	0
1	0	1
1	0	1
1	0	1
1	1	0
1	1	0
1	1	0
1	1	1
1	1	1
1	1	1

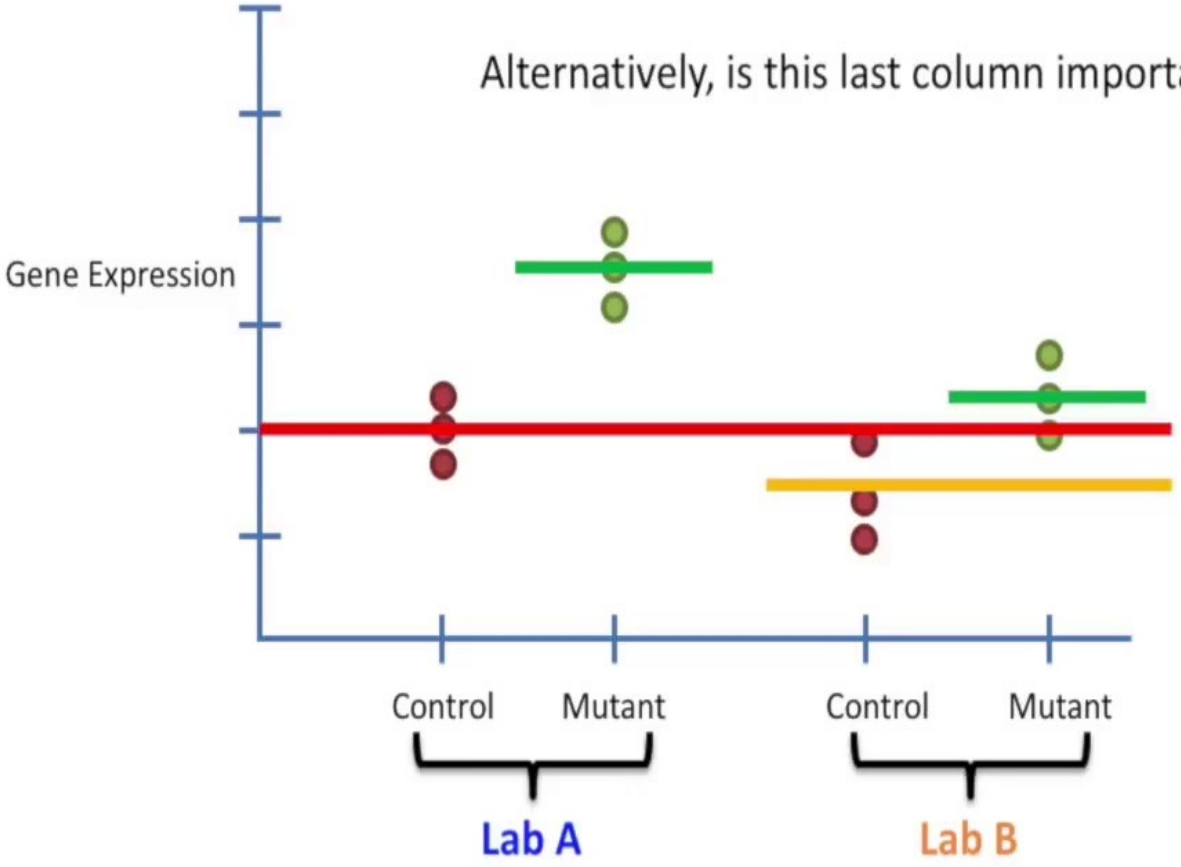
y = lab A control mean + lab B offset + difference$_{(\text{mutant - control})}$

Essentially, we want to know if this last term in the equation is important or not.



1	0	0
1	0	0
1	0	0
1	0	1
1	0	1
1	0	1
1	1	0
1	1	0
1	1	0
1	1	1
1	1	1
1	1	1

y = lab A control mean + lab B offset + difference$_{(\text{mutant - control})}$

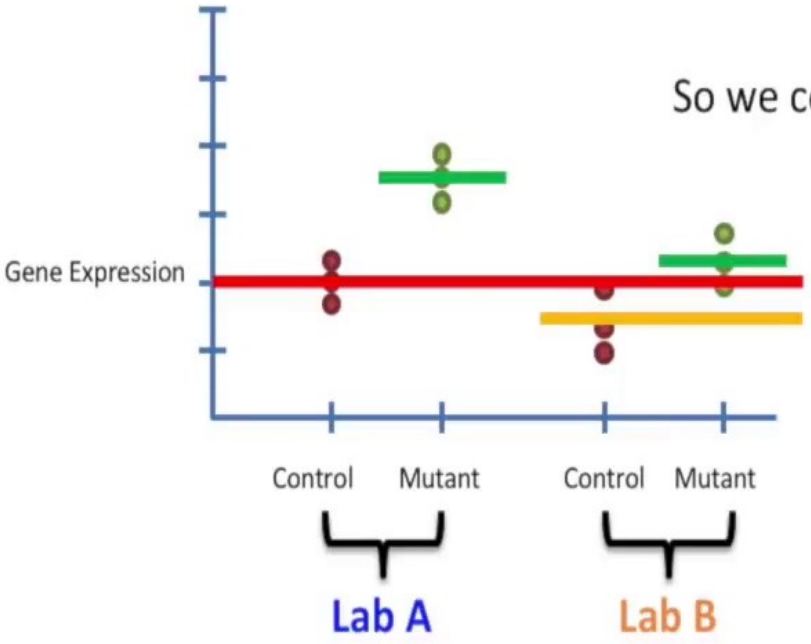


Alternatively, is this last column important?

1	0	0
1	0	0
1	0	0
1	0	1
1	0	1
1	0	1
1	1	0
1	1	0
1	1	0
1	1	1
1	1	1
1	1	1

$$y = \text{lab A control mean} + \text{lab B offset} + \text{difference}_{(\text{mutant - control})}$$

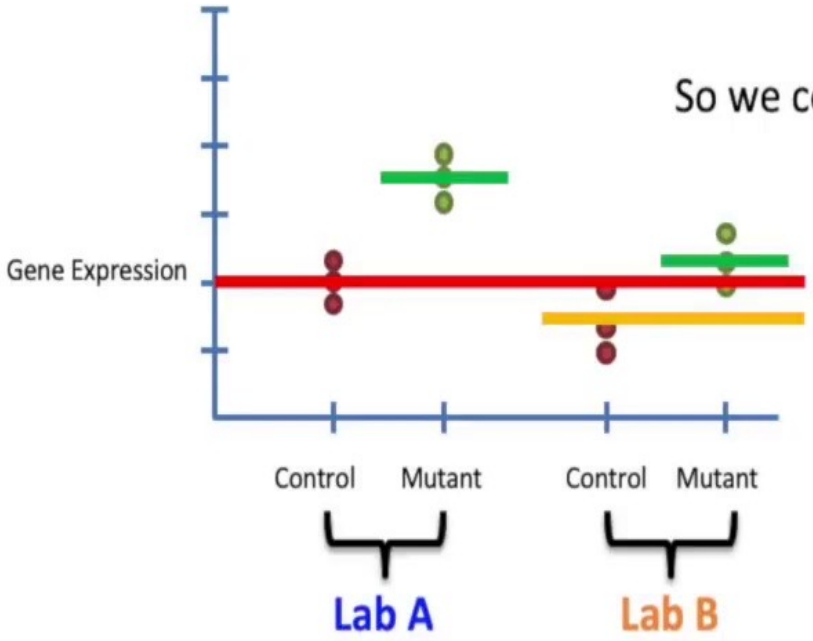
So we compare the fit of this fancy equation...



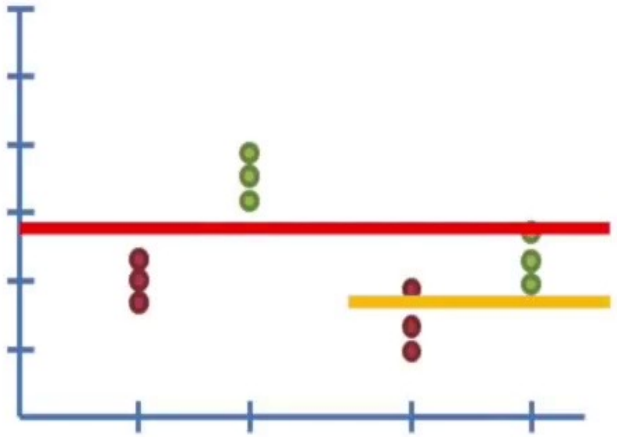
y = lab A control mean + lab B offset + difference$_{(\text{mutant - control})}$

So we compare the fit of this fancy equation...

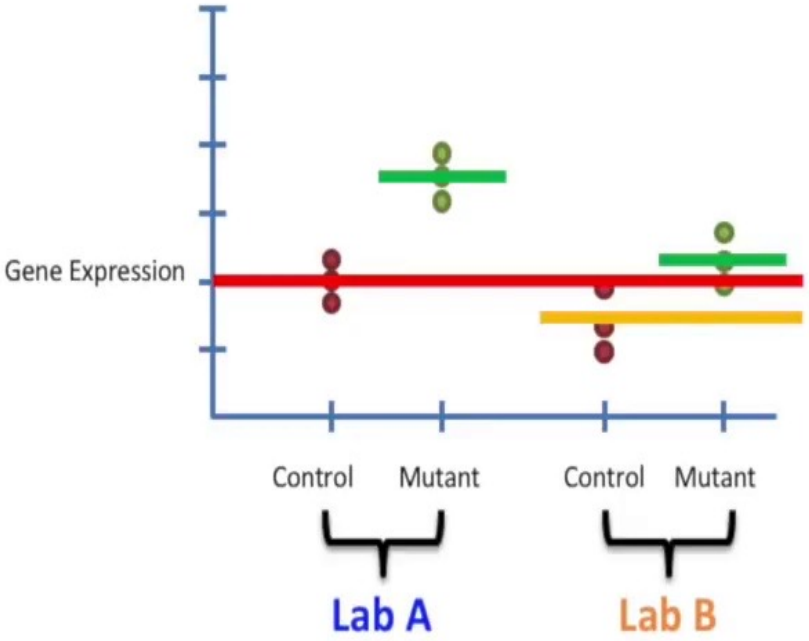
...to this simpler one (that ignores the control/mutant difference.



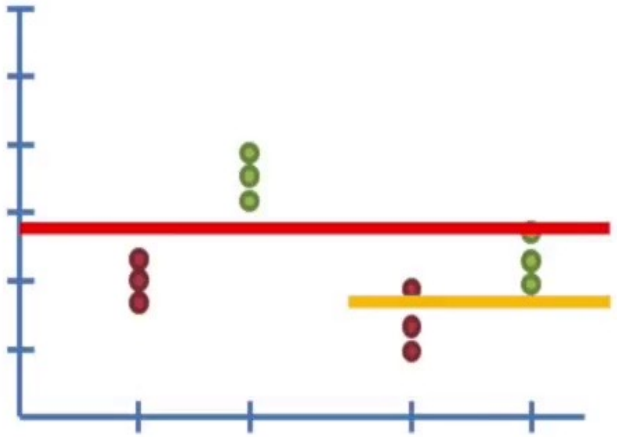
y = lab A + lab B offset



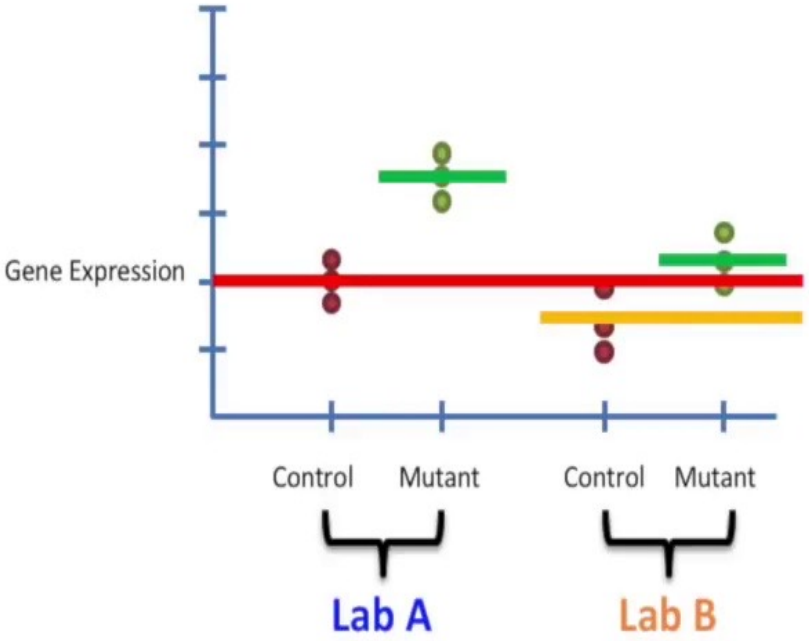
y = lab A control mean + lab B offset + difference$_{(\text{mutant - control})}$



A small p-value will tell us that the equation that keeps track of the mutant/control difference predicts the gene expression better than one that does not.

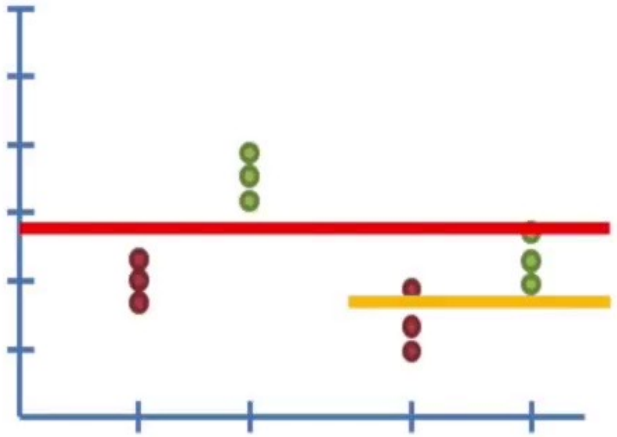


$$y = \text{lab A control mean} + \text{lab B offset} + \text{difference}_{(\text{mutant - control})}$$



A small p-value will tell us that the equation that keeps track of the mutant/control difference predicts the gene expression better than one that does not.

That will mean that the difference between controls and mutants is significant.



The End!!!!