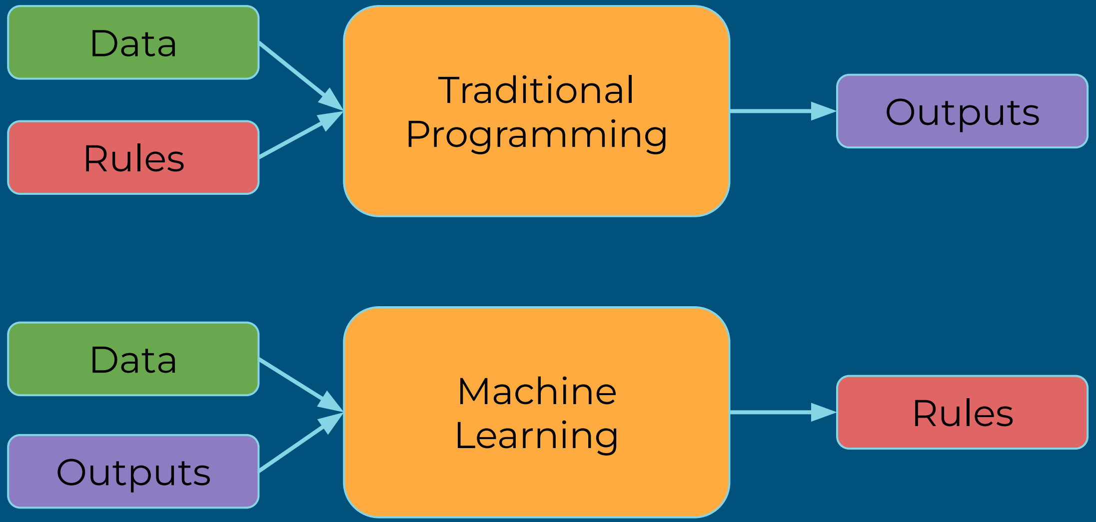
Linear Regression

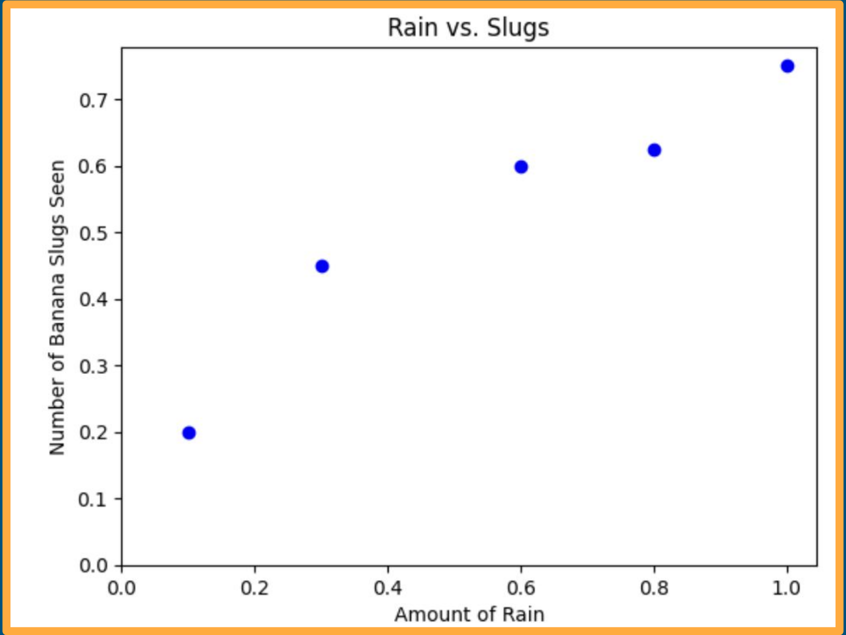
How does a computer draw a line that best follows the data?

Machine Learning vs. Traditional Programming



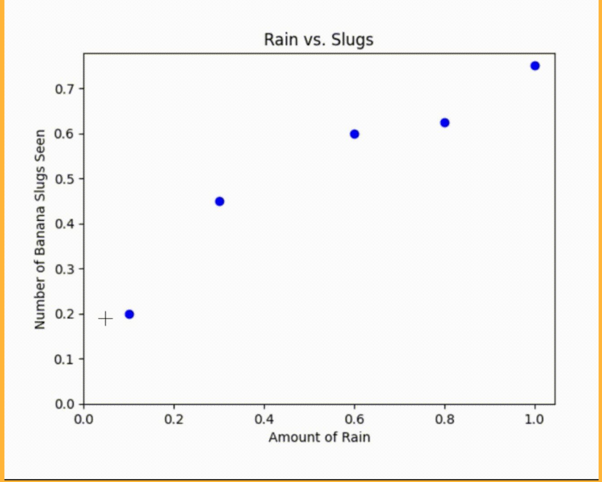
In ML the model learns and infers rules from the data, we do not program specific rules ourselves!

How would you plot a straight line that follows the trend in the data?

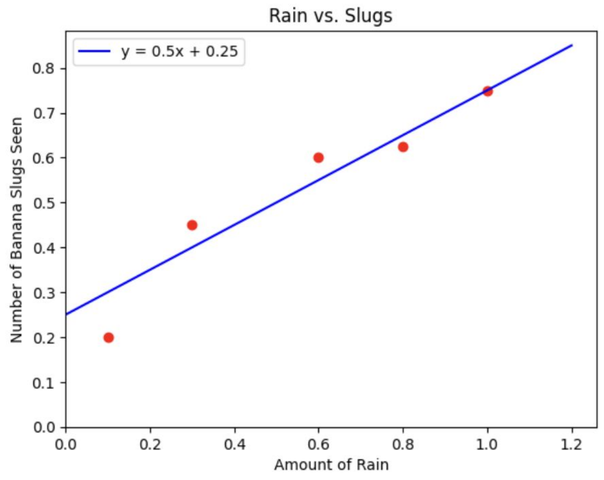


Fitting a linear line to a graph

By Hand:
(eyeball it XD)



Traditional Programming:
(come up with line equation on your own)

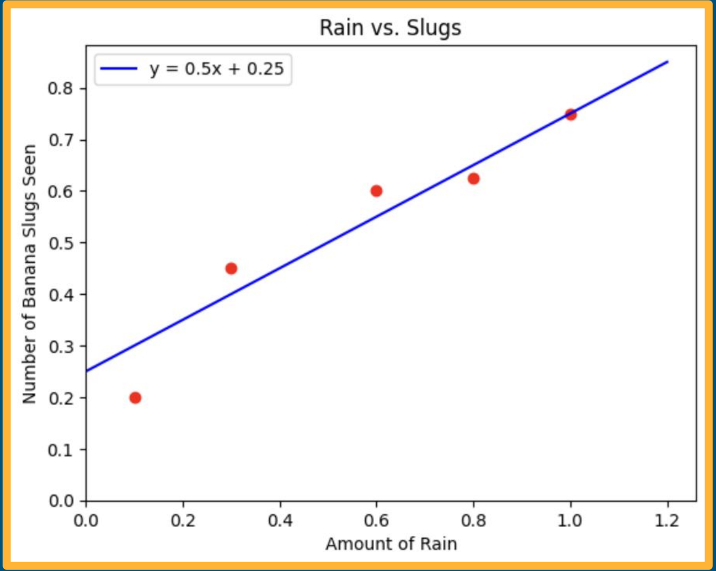


That Is Not ML!

What is Linear Regression?

We want the machine to find the line for us! This would be *Linear Regression*:

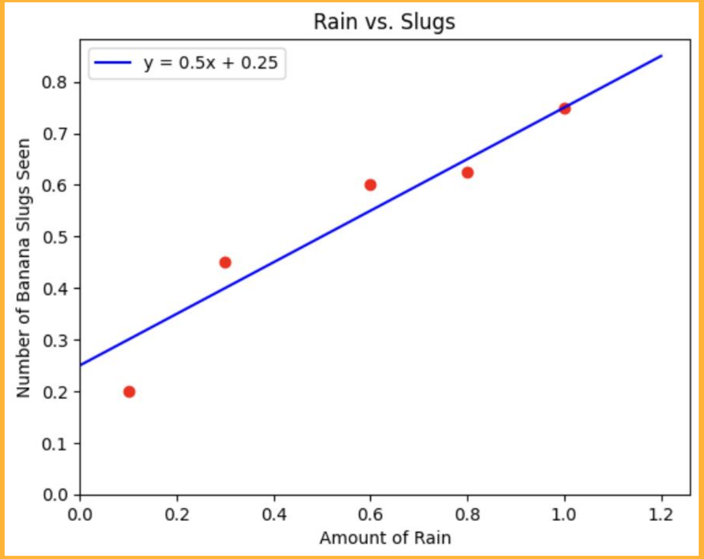
- *Linear* as in:



What is Linear Regression?

We want the machine to find the line for us! This would be *Linear Regression*:

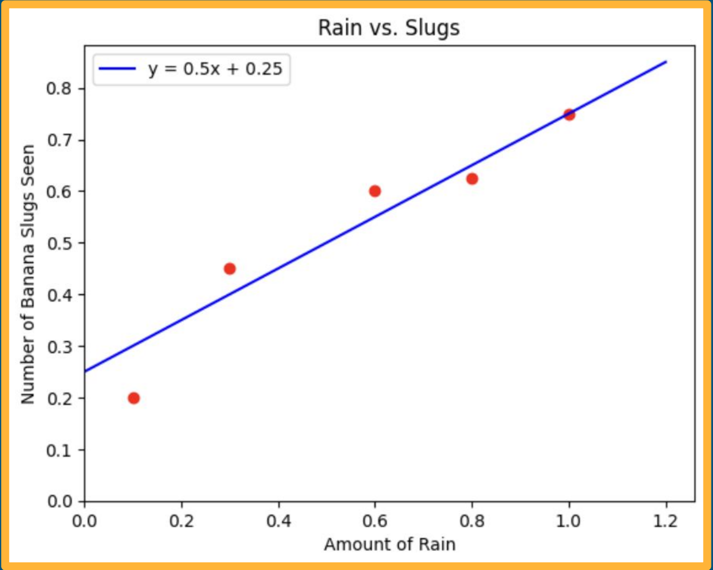
- *Linear* as in:
  - Not bendy 😅



What is Linear Regression?

We want the machine to find the line for us! This would be *Linear Regression*:

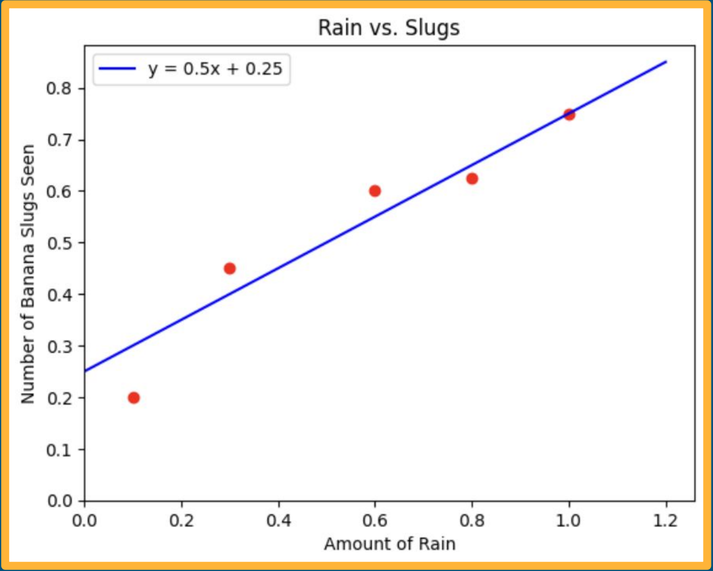
- *Linear* as in:
  - Not bendy 😅
- *Regression* as in:



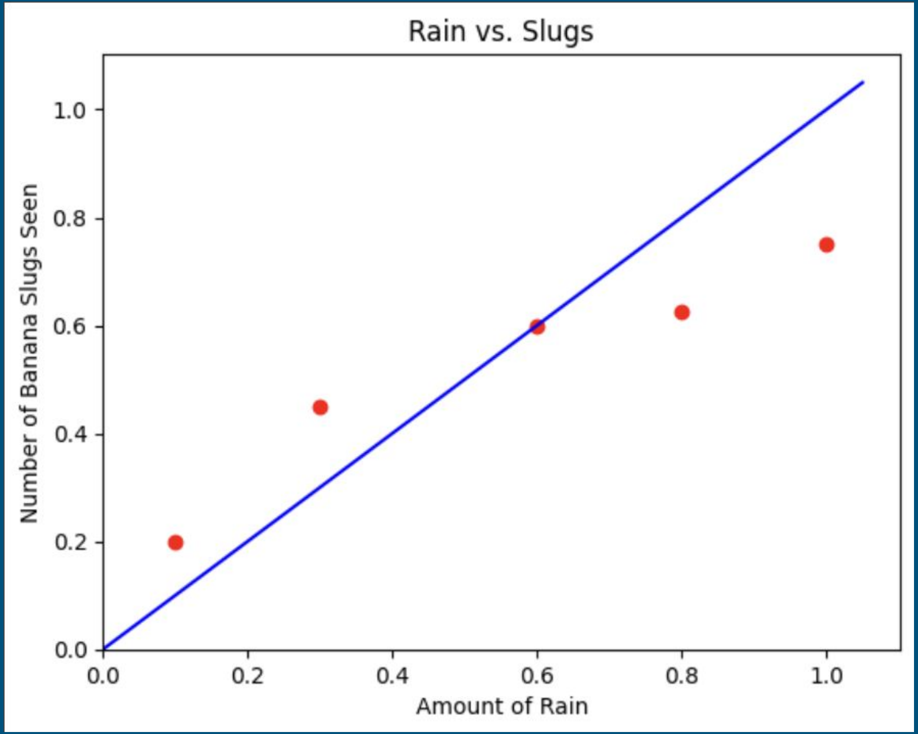
What is Linear Regression?

We want the machine to find the line for us! This would be *Linear Regression*:

- *Linear* as in:
  - Not bendy 😅
- *Regression* as in:
  - The model learns to predict a continuous value

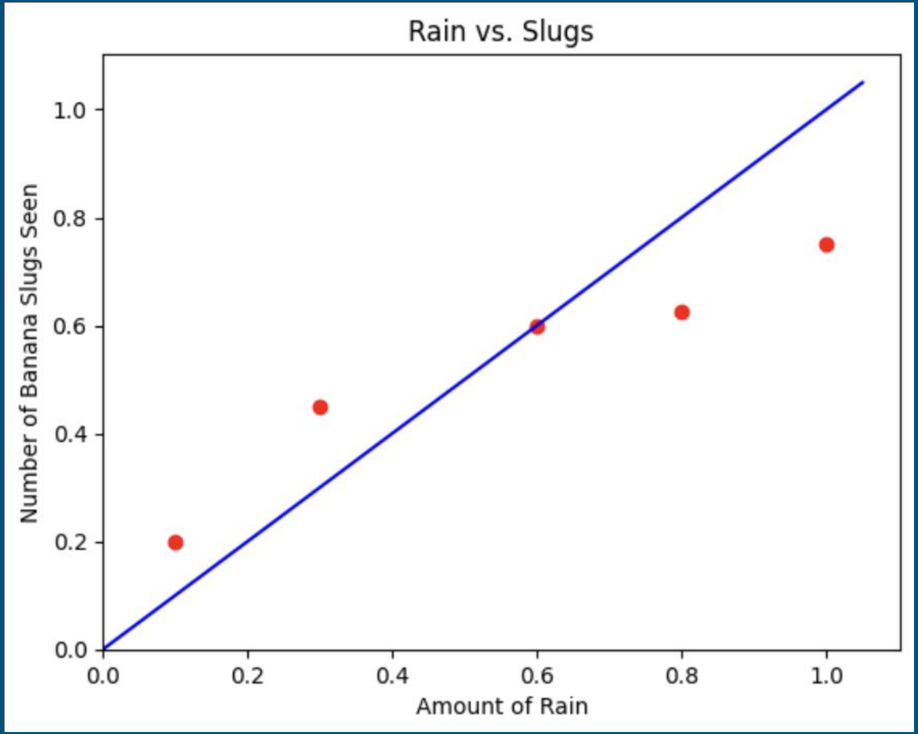


Given a line, what would the computer need to fit the line to the data?



“Fitting” a line to the data means finding the line that best follows the trend in the data.

Given a line, what would the computer need to fit the line to the data?



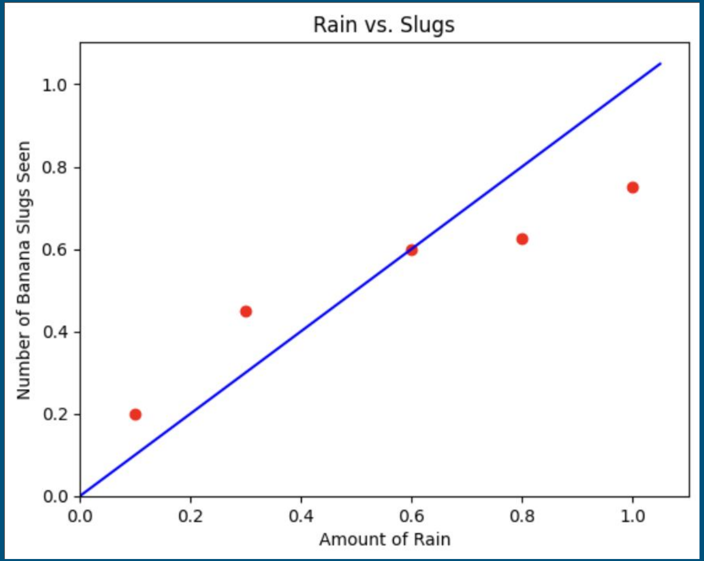
"Fitting" a line to the data means finding the line that best follows the trend in the data.

Remember:

- The equation for a line is
  - $y = mx + b$
  - Where:
    - *y* is the output
    - *x* is the input
    - *m* is the slope
    - *b* is the y-intercept

Given a line, what would we need to optimize it?

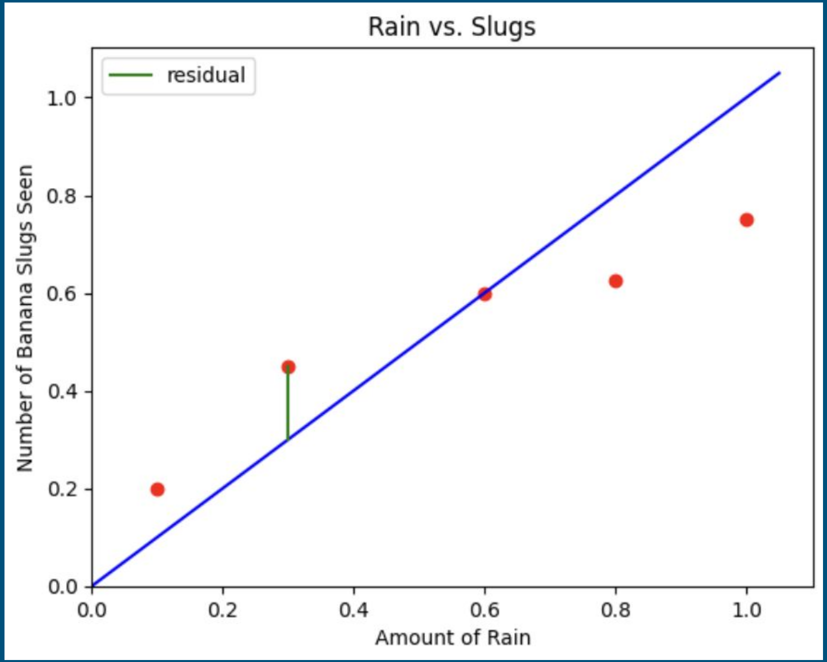
1. A measurement to determine how wrong the line is!
2. A method to minimize how wrong the line is!



Residual

$$Residual = Observed - Predicted$$

$$Residual = y - \hat{y}$$

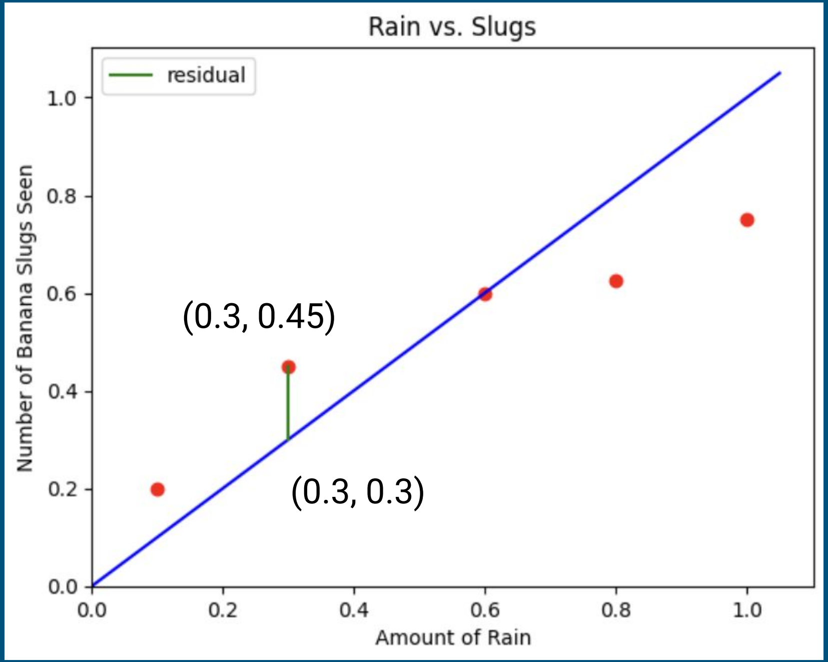


Residual

$$Residual = Observed - Predicted$$

$$Residual = y - \hat{y}$$

What is the residual for this point?



Residual

$$Residual = Observed - Predicted$$

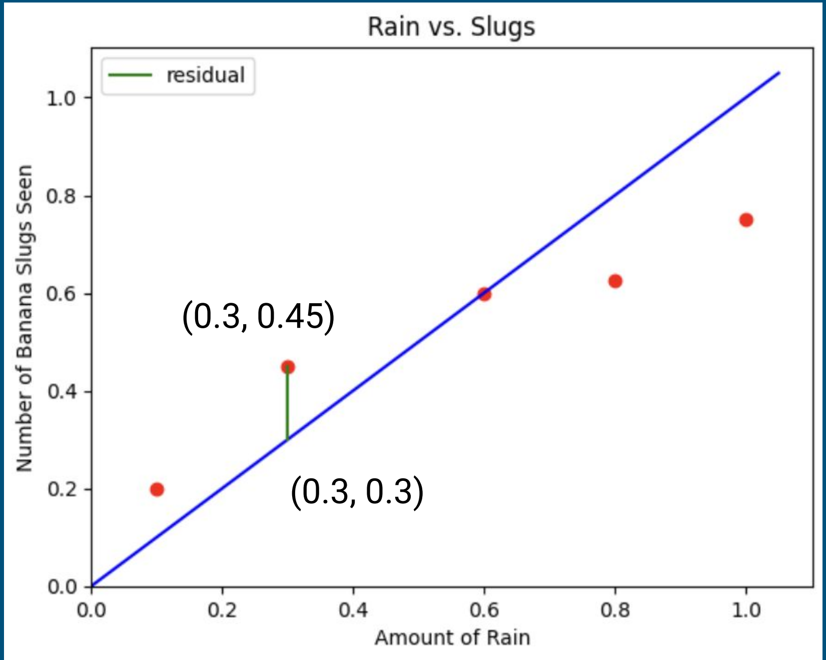
$$Residual = y - \hat{y}$$

What is the residual for this point?

```
residual = y_coords[1] - m*x_coords[1]+b
print('Observed:', y_coords[1])
print('Predicted:', m*x_coords[1]+b)
print('Residual:', residual)
```

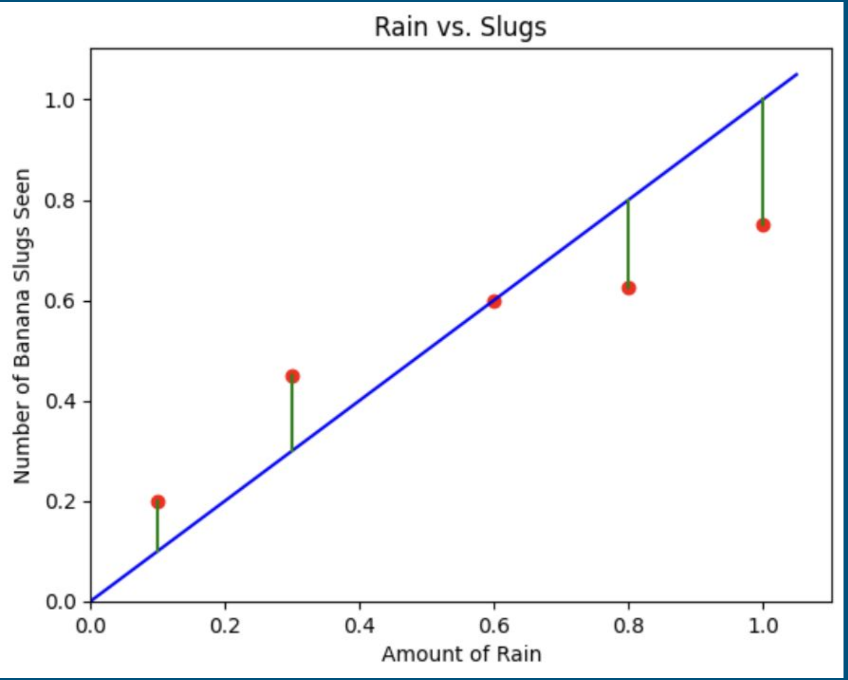
```
Observed: 0.45
Predicted: 0.3
Residual: 0.15000000000000002
```

Funny computer math lol.



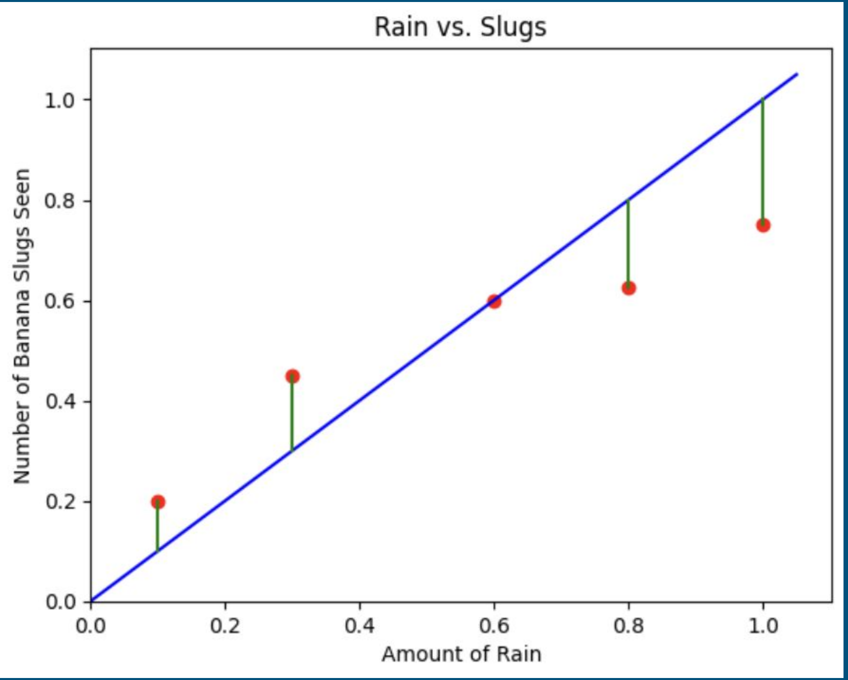
0.15 Banana Slugs Per Hour

How “wrong” is the line?



- The residual is how “wrong” the line is for a single data point.
- The smaller the residual, the better.
- How should we determine how “wrong” the entire line is?

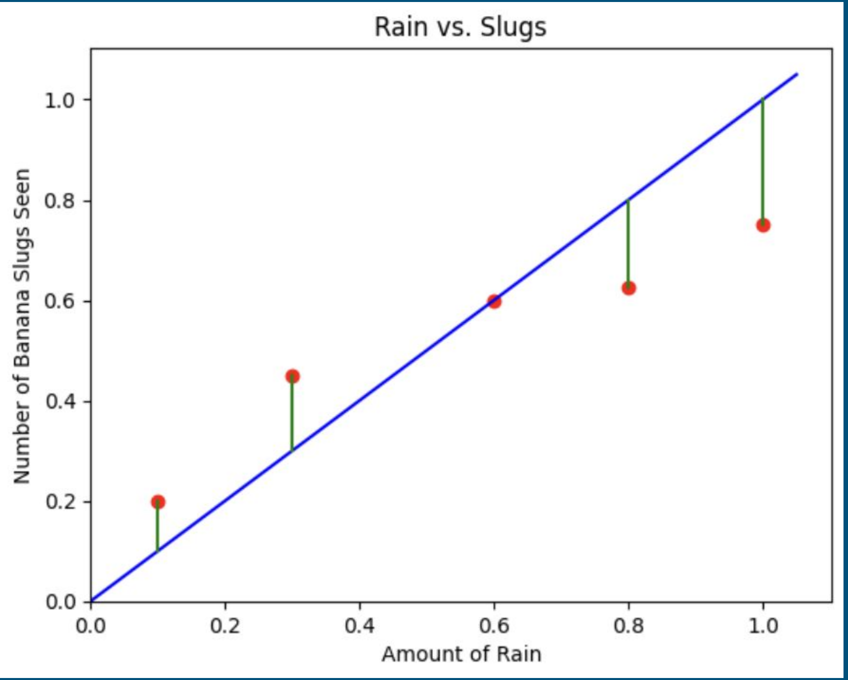
Do we just sum the residuals?



- No!
- If we sum the residuals in the current state, the residuals above the line (which are positive) will cancel out the residuals below the line (which are negative)
- So what do we do?

Squared Residuals

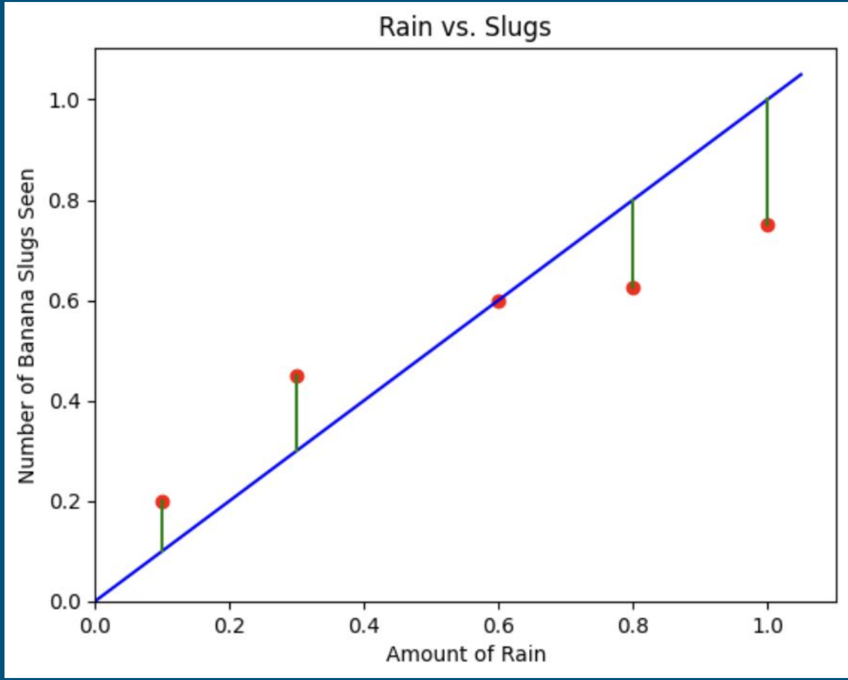
$$Squared\ Residual = (Observed - Predicted)^2$$



- To remedy this issue, we simply square the residual!
- This is the same reason we square the difference in the distance equation between points.

Sum of Squared Residuals (SSR)

$$SSR = \sum_{i=1}^{n} (Observed_i - Predicted_i)^2$$



Sum of Squared Residuals (SSR)

$$Residual = Observed_i - Predicted_i$$

$$Squared\ Residual = (Observed_i - Predicted_i)^2$$

$$Sum\ of\ Squared\ Residuals = \sum_{i=1}^{n}(Observed_i - Predicted_i)^2$$

Mean Squared Error (Side Note)

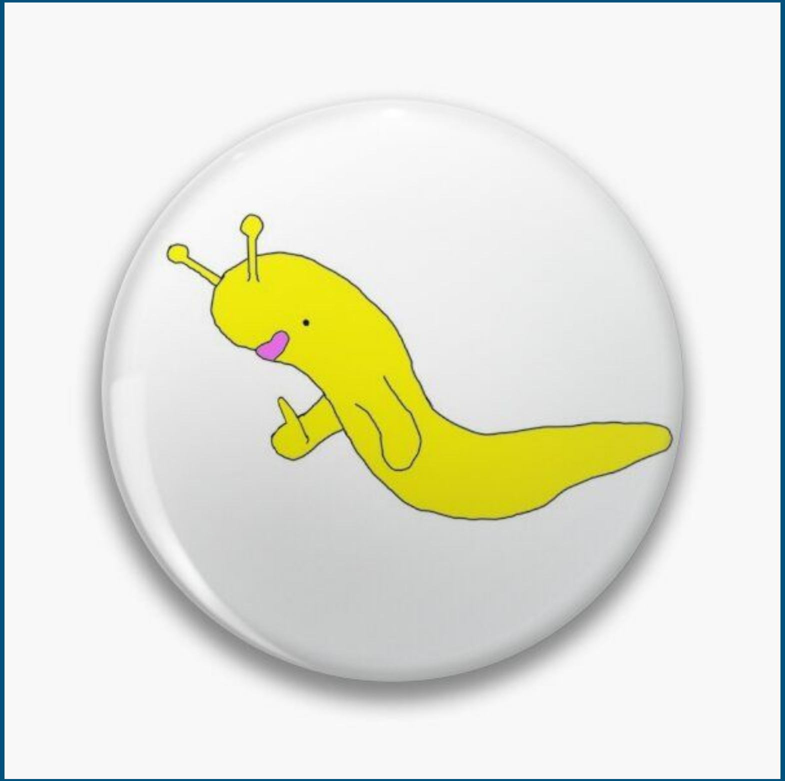
Not used in *Linear Regression*, but will be important for more complicated models.

- The *Residual* is called the *Error*, because it measures deviations
- How do we measure the mean of a list of numbers?

$$SSR = \sum_{i=1}^{n} (Observed_i - Predicted_i)^2$$

$$MSE = \frac{SSR}{n}$$

Yay! We now have a way to determine how “wrong” our line is :D

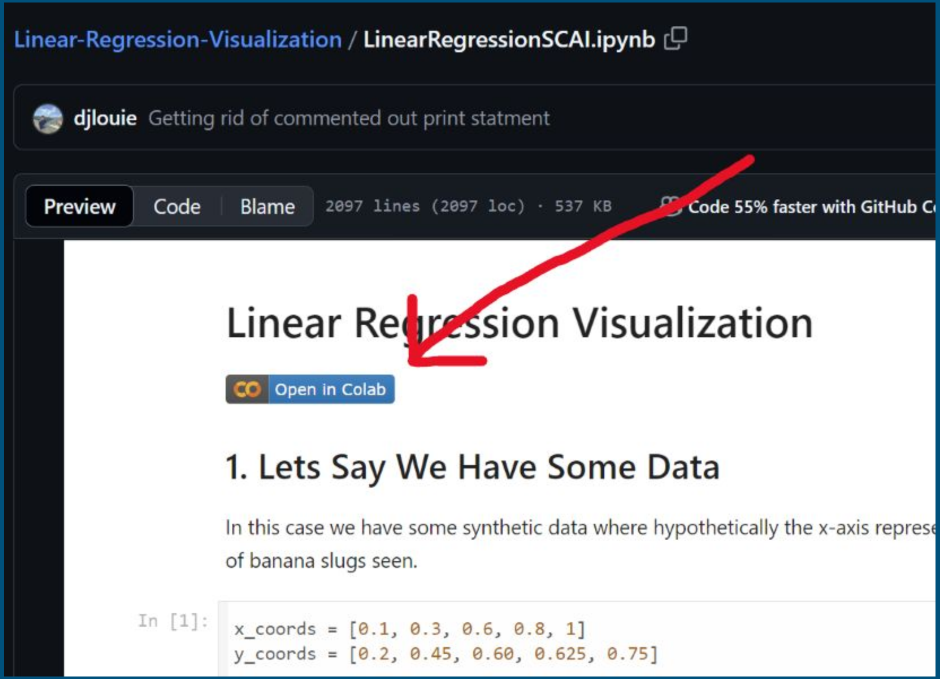


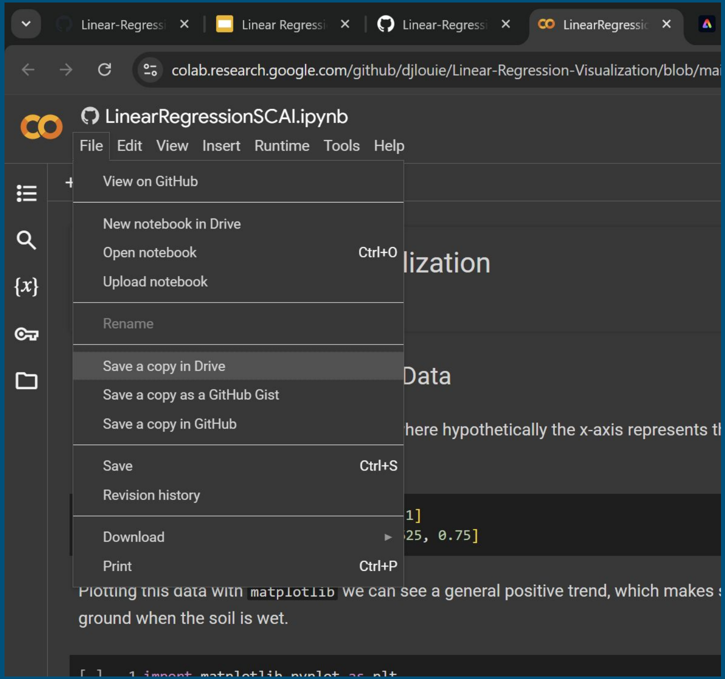
https://bit.ly/LinearRegressionSCAI

This is all possible with Python :D

Go to the code!

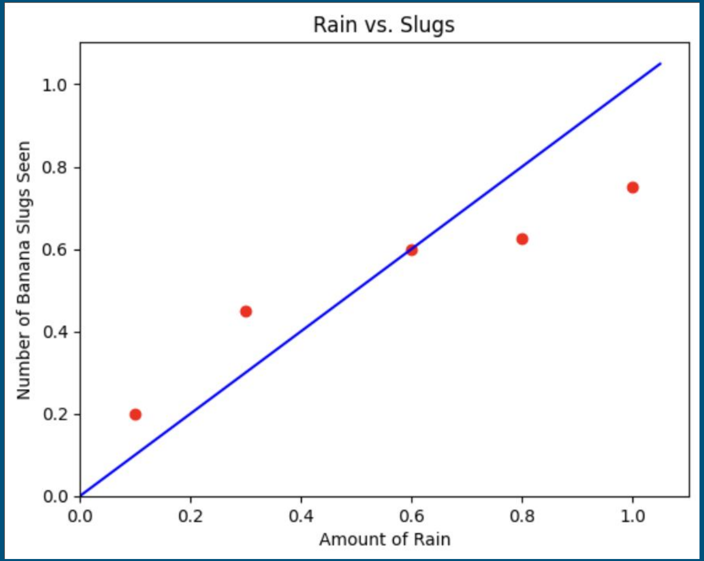
Make Sure to Save a Copy For Later!





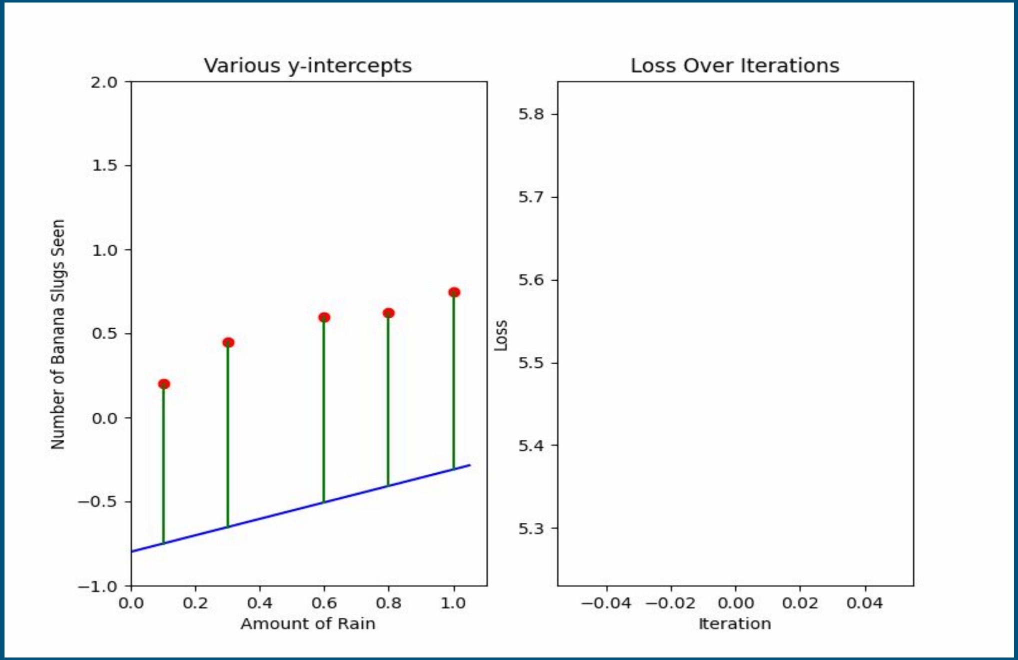
Given a line, what would we need to optimize it?

1. A measurement to determine how wrong the line is!
2. A method to minimize how wrong the line is!



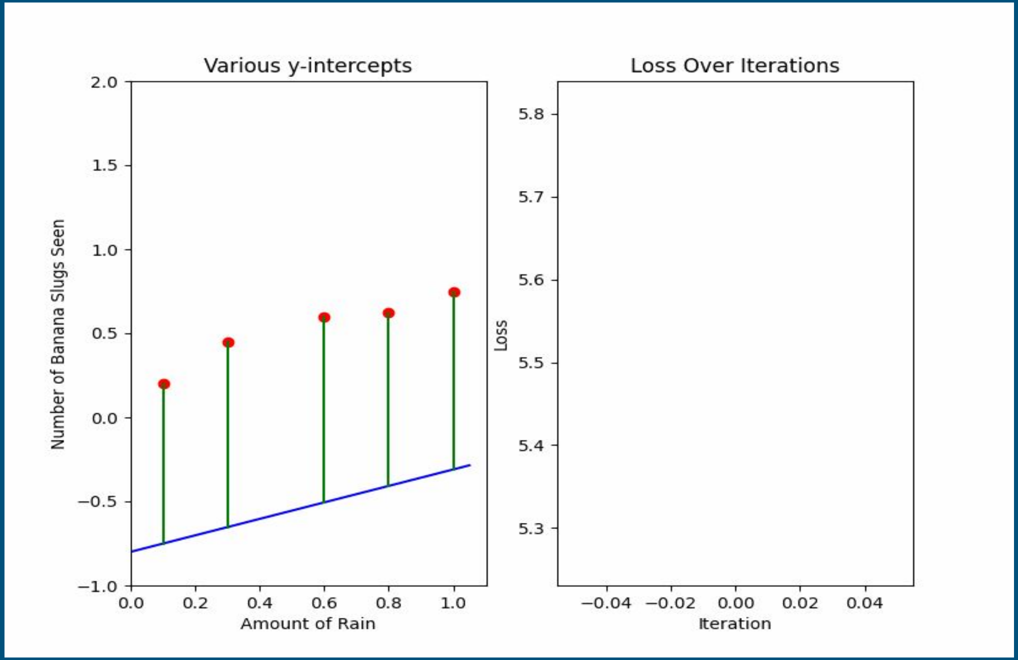
How do we minimize how “wrong” (Sum of Squared Residuals) the line is?

What does the SSR look like when we adjust one parameter?



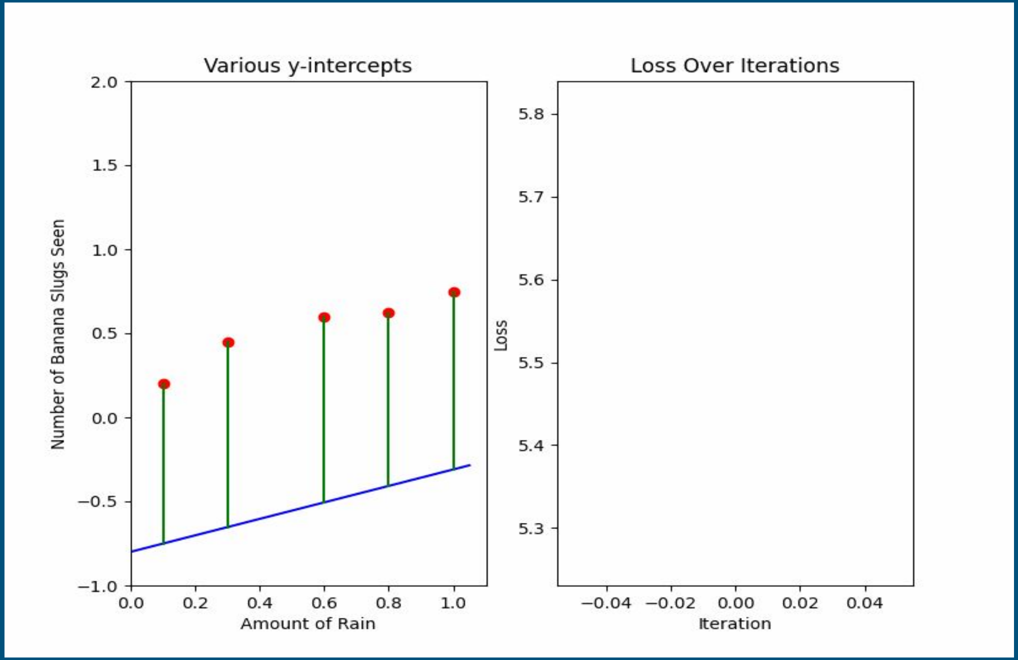
Here is an example of a line where we already fitted the slope, but we start at a a y-intercept of -0.8 and increment by +0.05 every iteration:

What does the SSR look like when we adjust one parameter?



- We once again observe that when the line is further away from the points, the higher the SSR or “Loss” is.

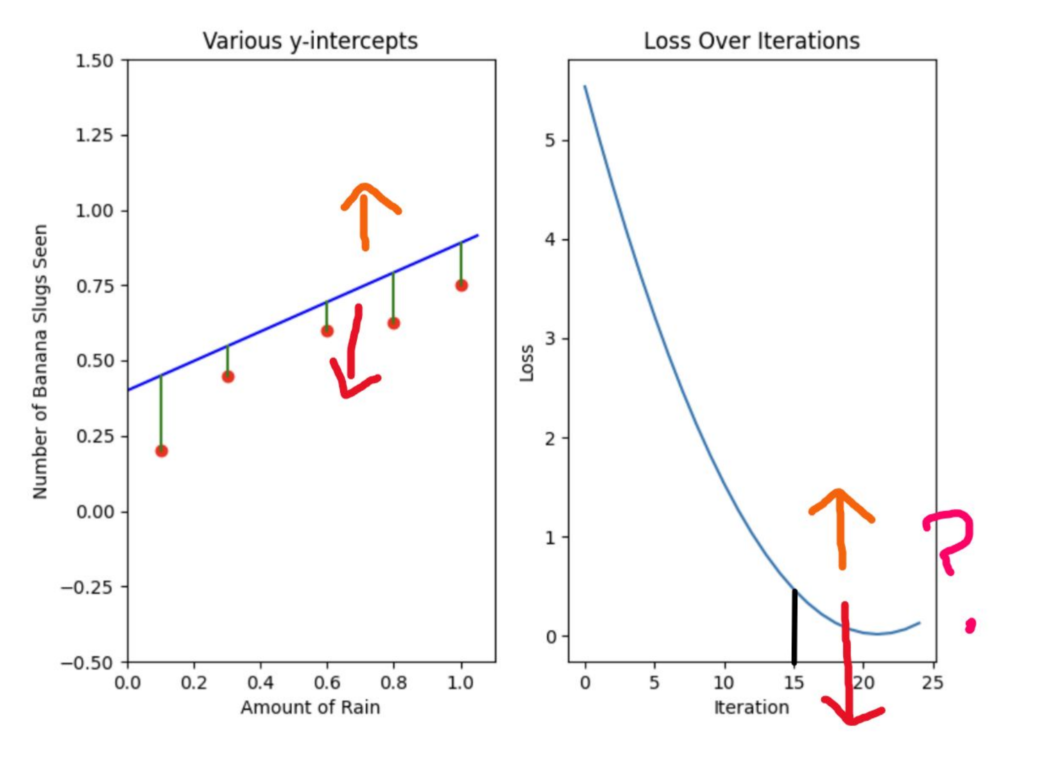
What does the SSR look like when we adjust one parameter?



- We once again observe that when the line is further away from the points, the higher the SSR or "Loss" is.
- More importantly, the the graph on the right forms a type of "loss landscape" that maps a certain b to a SSR value that is the shape of a parabola.

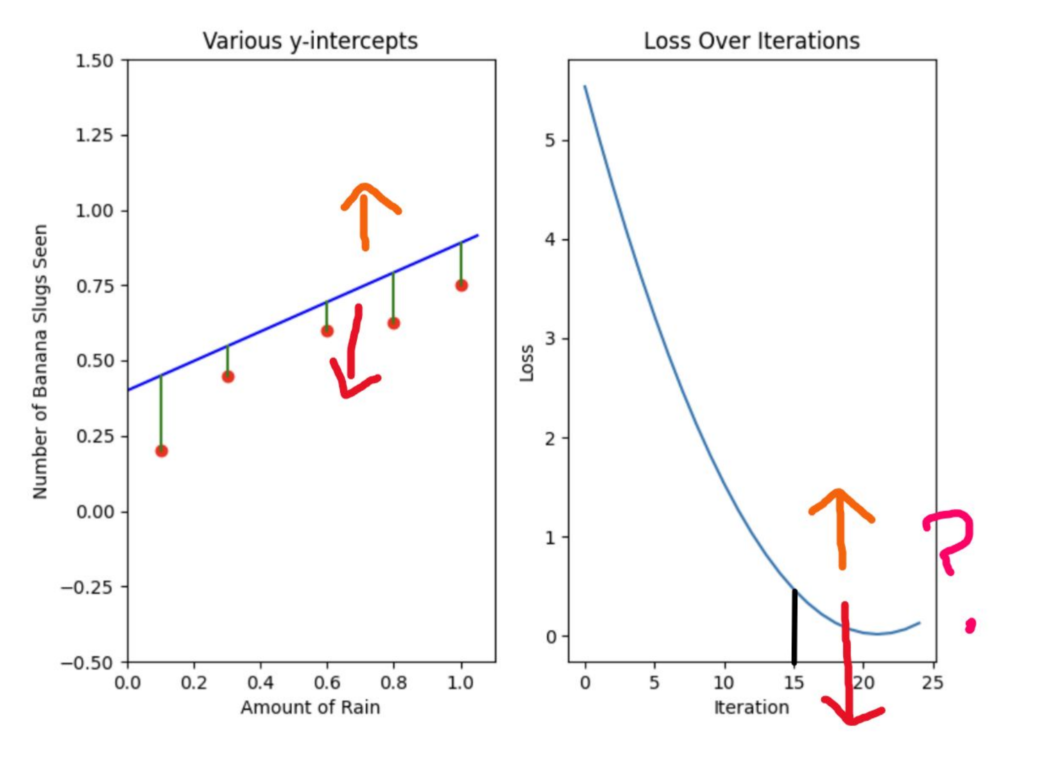
Wouldn’t it be great if we could use this information to determine which way to move the y-intercept (b)?

Do we go up or down?



Let's say we are at iteration 15. How do we know if we should increase or decrease the y-intercept?

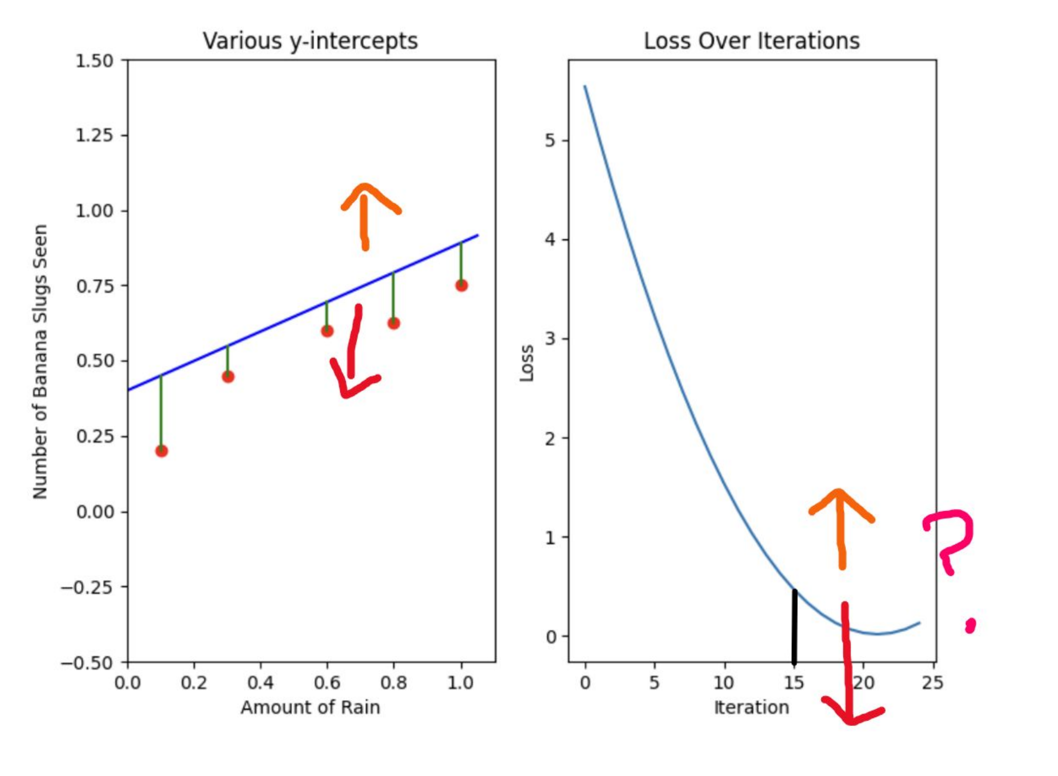
Do we go up or down?



Let's say we are at iteration 15. How do we know if we should increase or decrease the y-intercept?

- One might think we should try both directions and do which one is best, but:

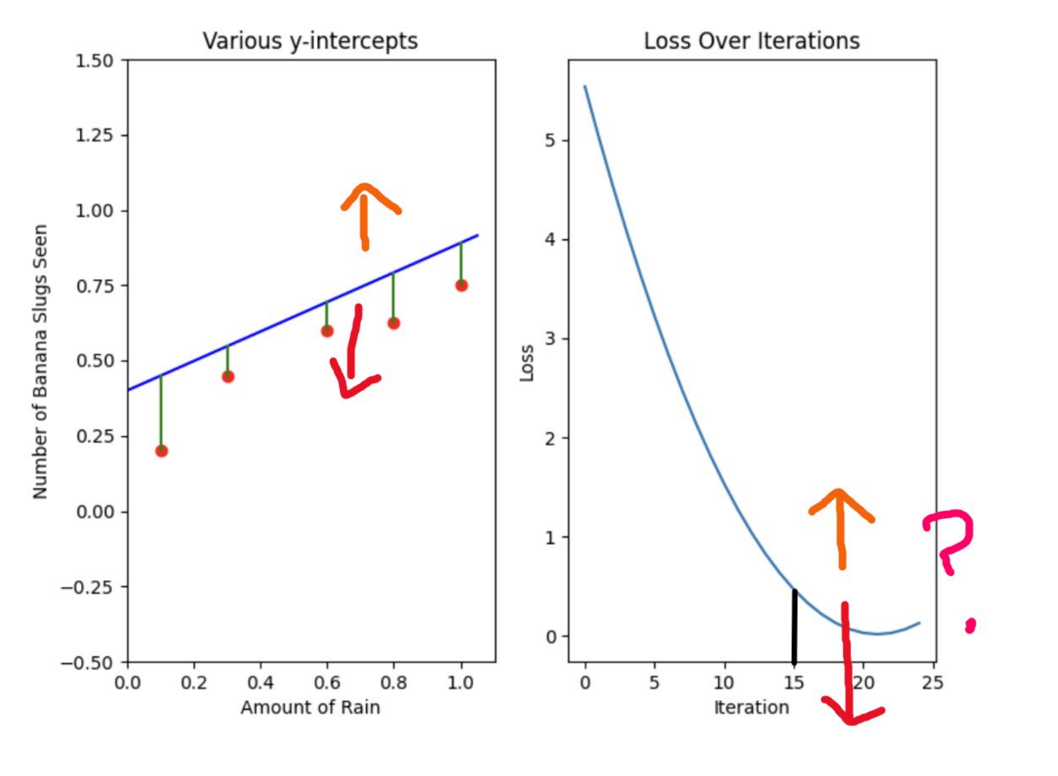
Do we go up or down?



Let's say we are at iteration 15. How do we know if we should increase or decrease the y-intercept?

- One might think we should try both directions and do which one is best, but:
  - We would have to calculate it twice and then back step

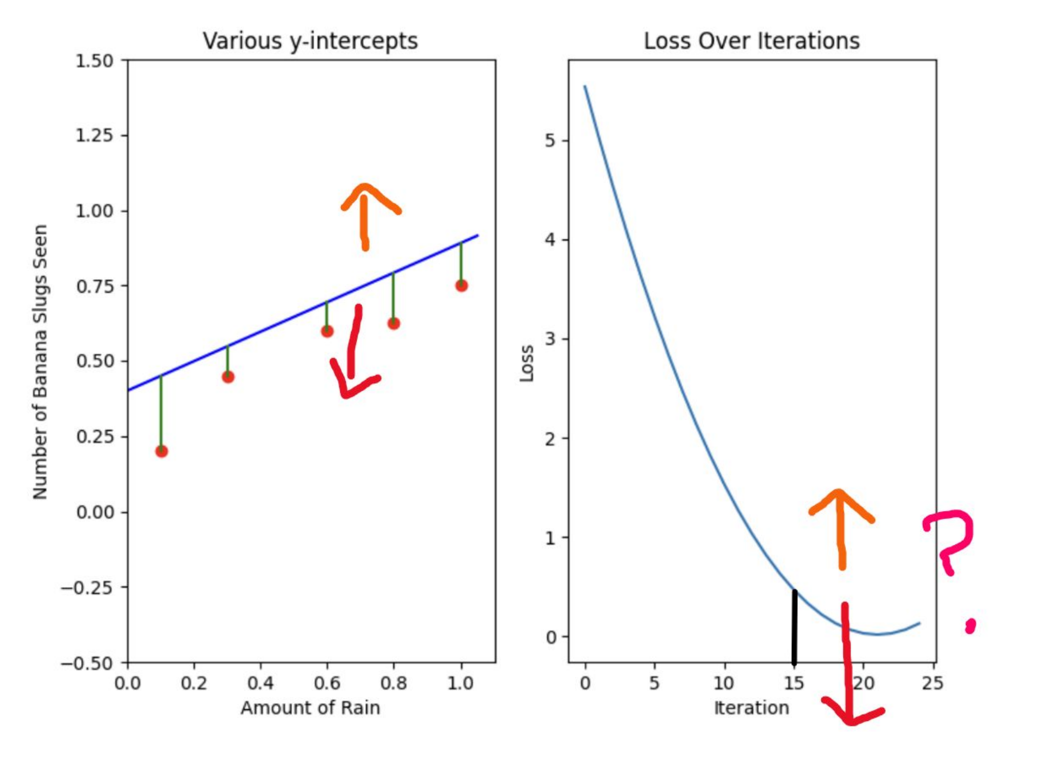
Do we go up or down?



Let’s say we are at iteration 15. How do we know if we should increase or decrease the y-intercept?

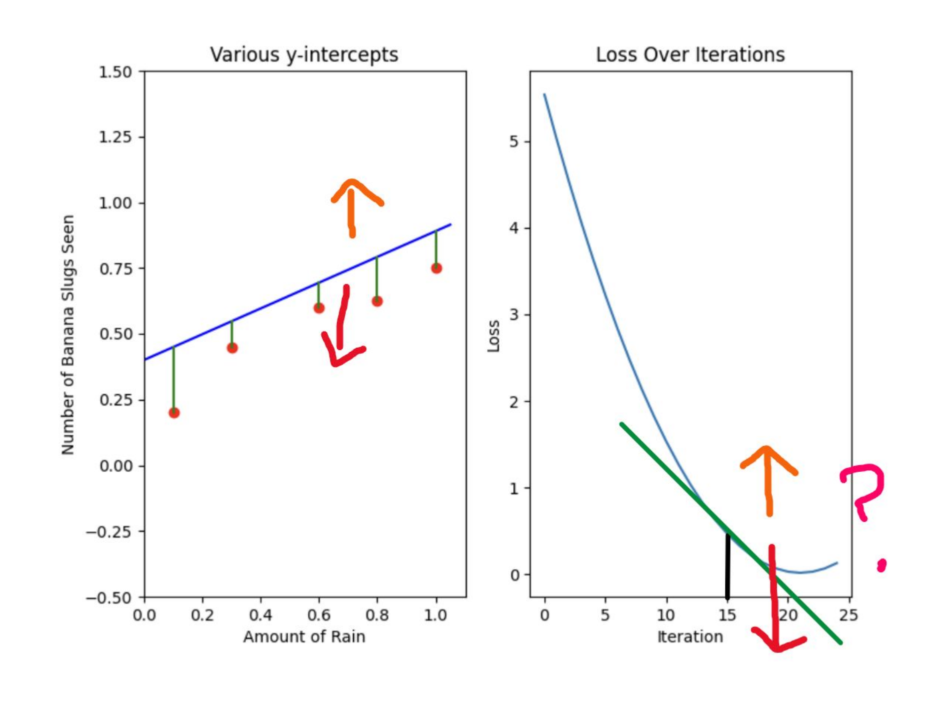
- One might think we should try both directions and do which one is best, but:
  - We would have to calculate it twice and then back step
  - We wouldn’t know how much to increase or decrease it by. What if we overshot our correction?

Do we go up or down?



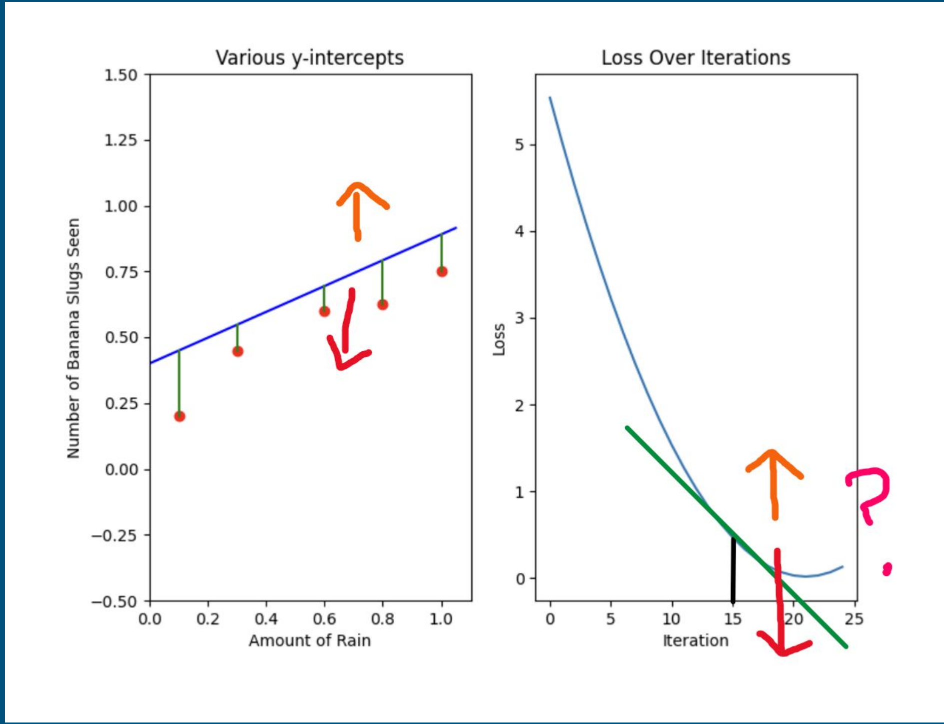
Is there a better way?

Do we go up or down?



What if we instead calculated ths slope of the loss with respect to b?

Do we go up or down?



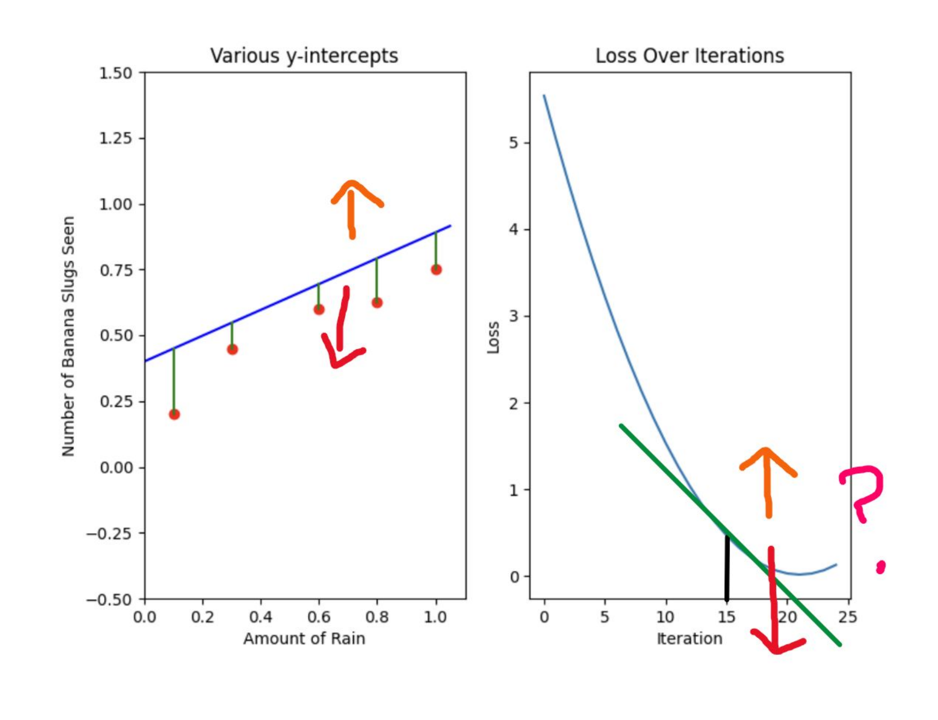
What if we instead calculated ths slope of the loss with respect to b?

- A positive slope tells us increasing b increases the loss
- A negative slope tells us decreasing b decreases the loss

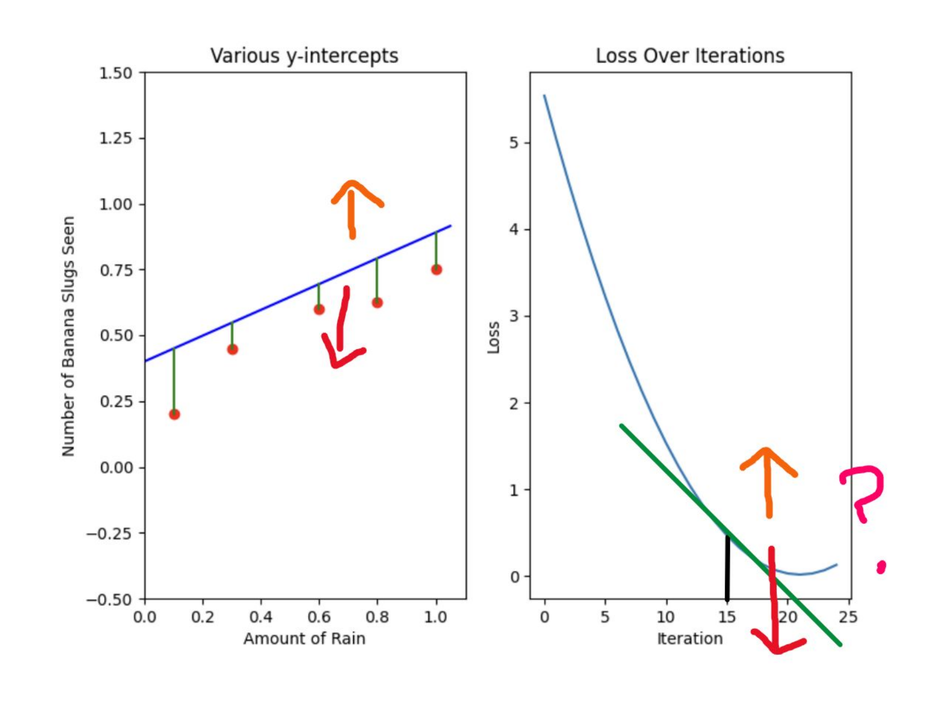
Do we go up or down?

What if we instead calculated ths slope of the loss with respect to b?

- The slope magnitude (aka absolute value) tells us how sensitive the loss is to a change in b
- A larger slope magnitude also tells us we are further from a local minima



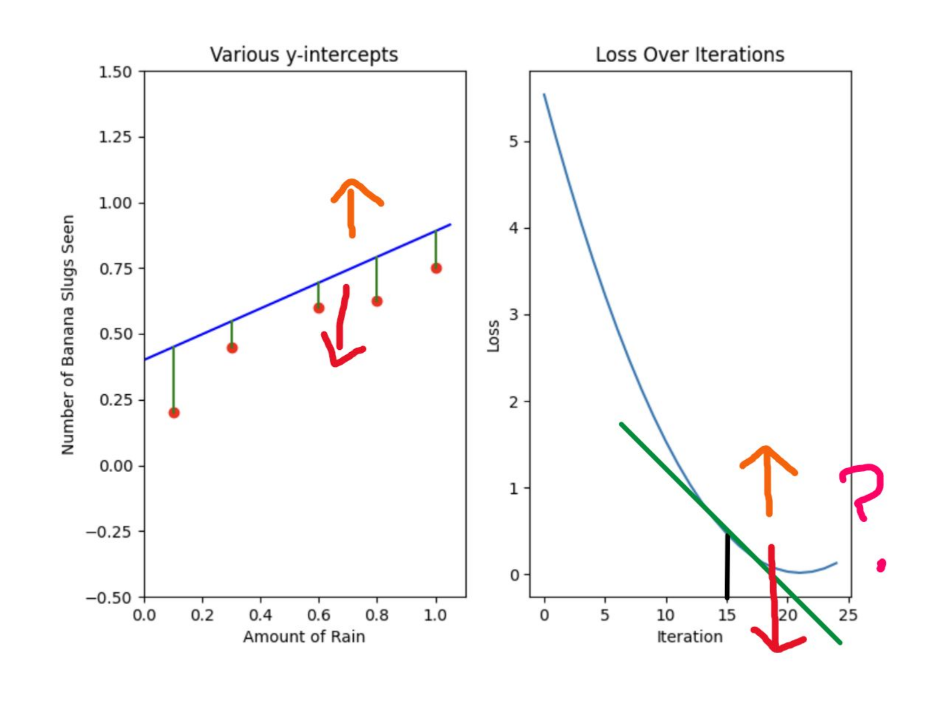
Do we go up or down?



What if we instead calculated ths slope of the loss with respect to b?

- Thus in order to adjust b, we can subtract the slope from it, and this can give us a more direct path to minimizing the loss!!!

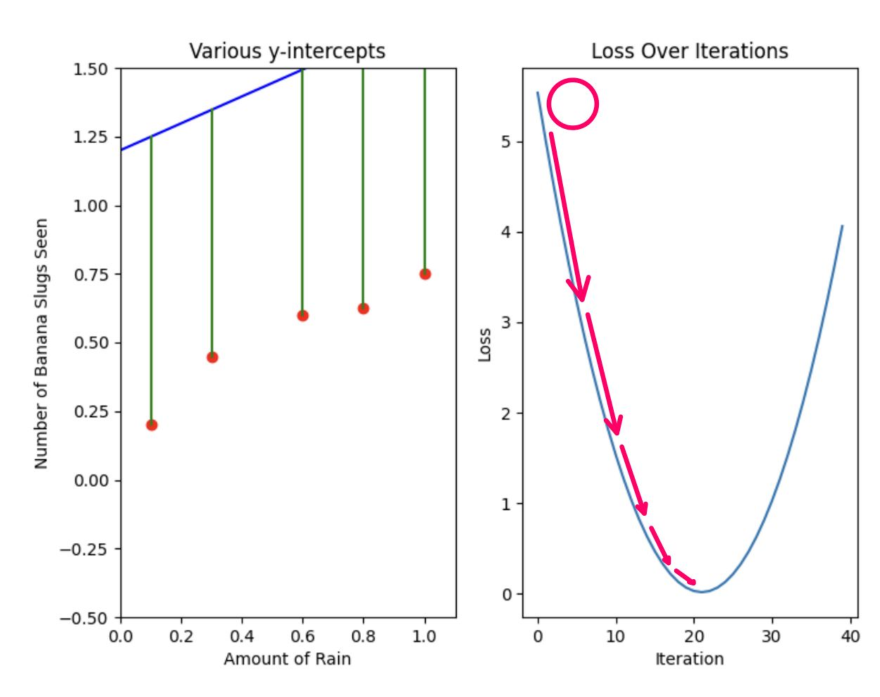
Do we go up or down?



What if we instead calculated ths slope of the loss with respect to b?

- Thus in order to adjust b, we can subtract the slope from it, and this can give us a more direct path to minimizing the loss!!!

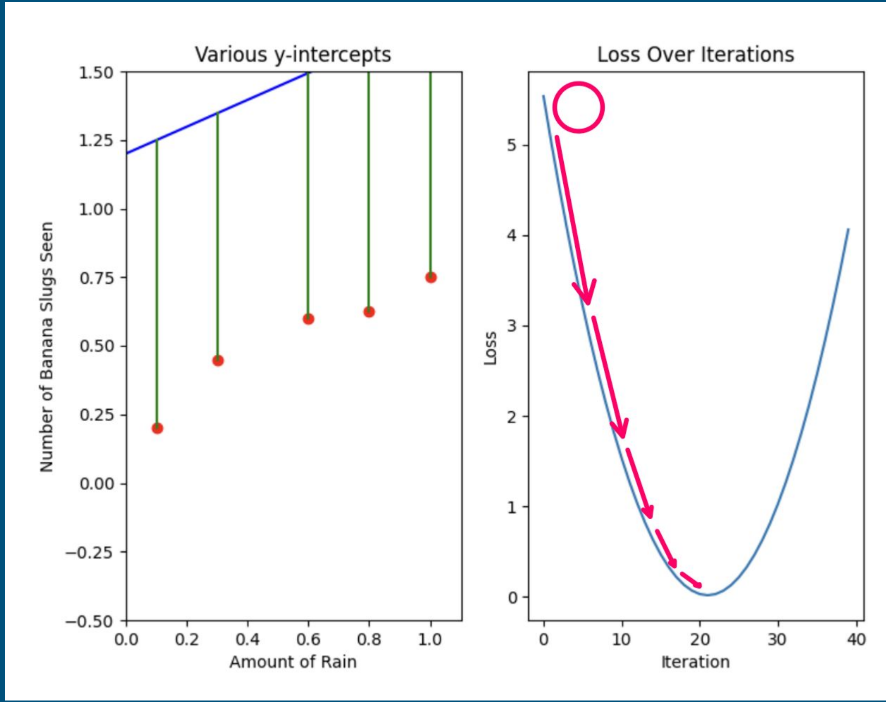
Do we go up or down?



One way to visualize this is a ball rolling down a hill or valley.

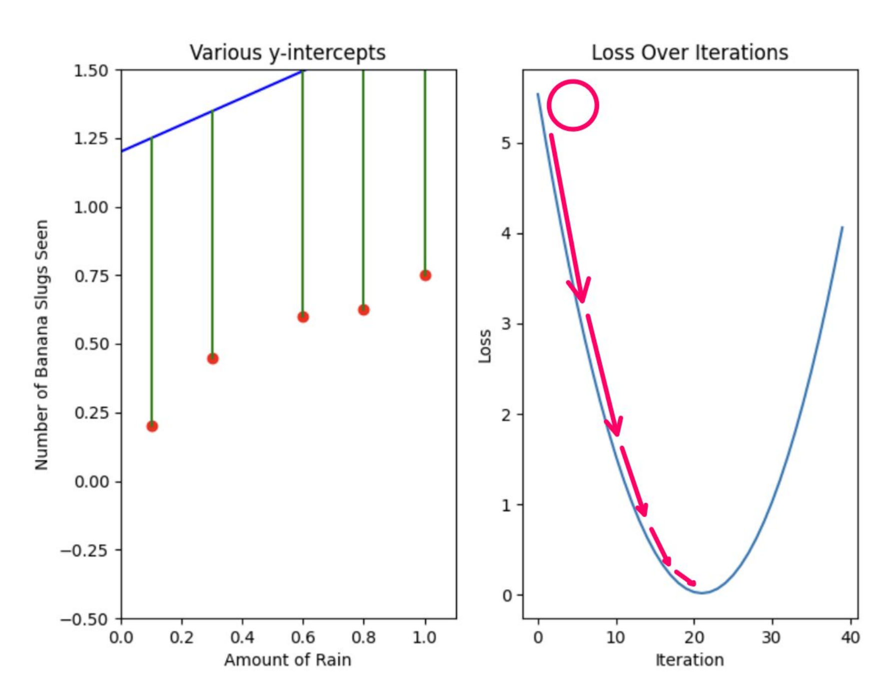
- The slope points up the hill, and the ball goes down the slope.
- Thus the "ball" or our parameters should be adjusted to follow the negative slope to find a local minima like a ball rolls to the lowest point of a hill.

Do we go up or down?



Additionally, because the magnitude of the slope is larger (think steeper) the further we are from a minima, it makes sense that when we add the negative slope we are making larger adjustments for bigger slopes and smaller adjustments for smaller slopes

Do we go up or down?



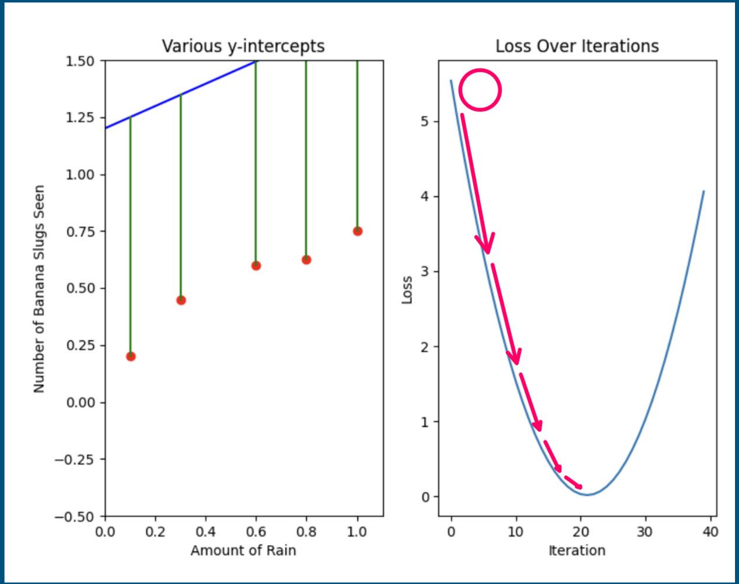
In practice, we often subtract a fraction of the derivative determined by a hyperparameter called *learning rate* between 0 and 1.

- This can also be thought of as the "step size"

Do we go up or down?

In practice, we often subtract from a fraction of the derivative determined by a hyperparameter called *learning rate* between 0 and 1.

- This can also be thought of as the “step size”
- Setting a good learning rate is important because:
    - One that is too big will make the model “step over” the minima
    - One that is too small will take too many iterations to reach a minima



This is great and all, but how do we calculate the slope?

Who here knows what a derivative is?

As you guys have learned in pre-calculus or calculus, that for lines and planes the *derivative* is the *slope* aka *rate of change*.

- In machine learning we like to use an even more general term called the *gradient*, which extends the concept of the derivative to multivariate calculus.

$$\nabla f = \frac{df}{dx}\hat{i} + \frac{df}{dy}\hat{j} + \frac{df}{dz}\hat{k}$$

Don’t worry if you don’t understand this.

https://bit.ly/LinearRegressionSCAI

Luckily, we can easily calculate the gradient with PyTorch!

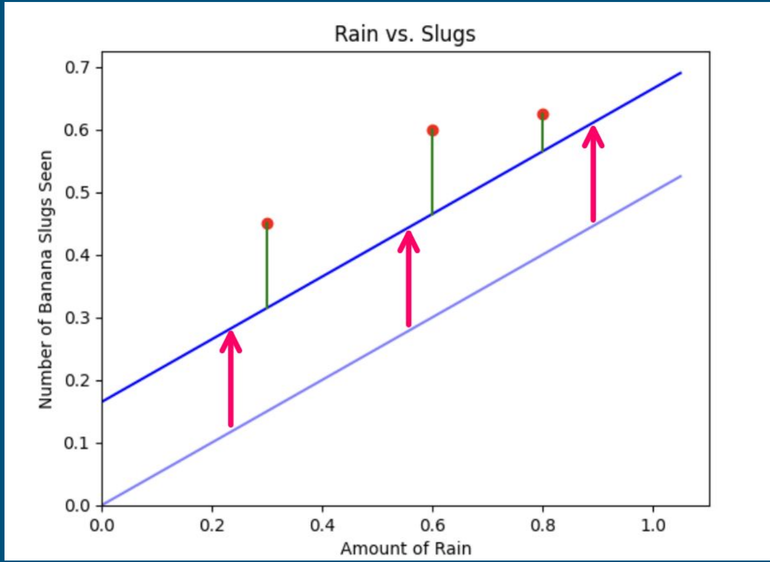
Go to the code!

Update the Y-Intercept!

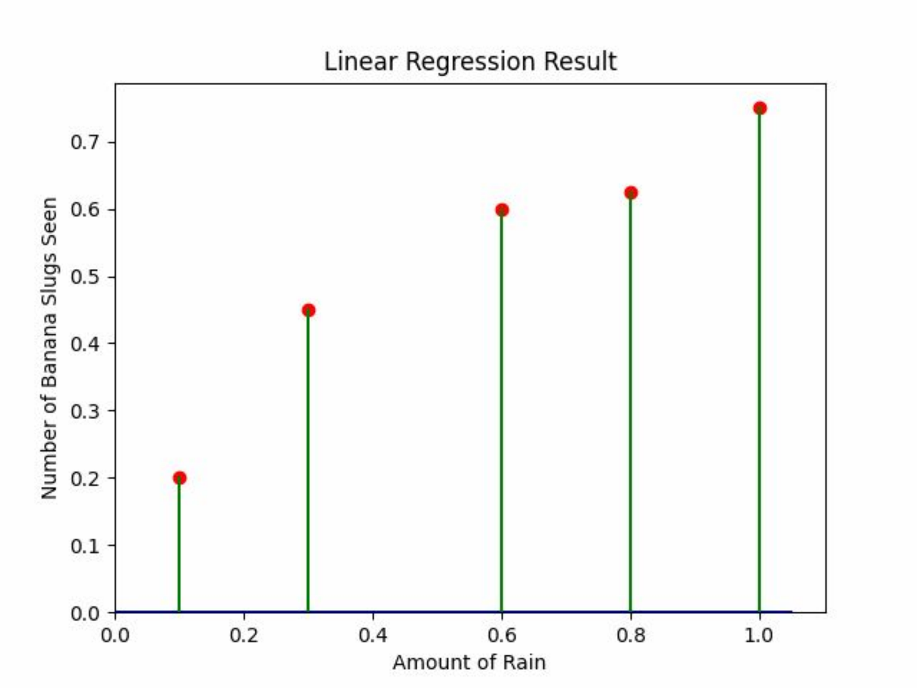
Remember that to do that we subtract it by a fraction of the gradient! In this case let's say the *learning rate* is 0.1.

$$b_{new} = b_{old} - lr * \frac{dSSR}{db}$$

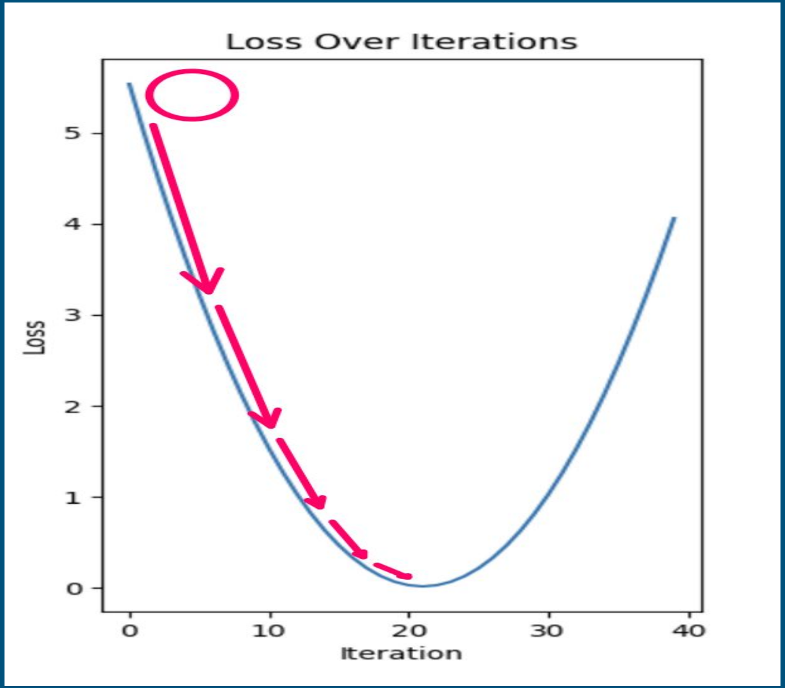
$$b_{new} = 0 - (0.1 * -1.65) = 0.165$$



Now repeat this for many iterations!



We call this entire process Gradient Descent!



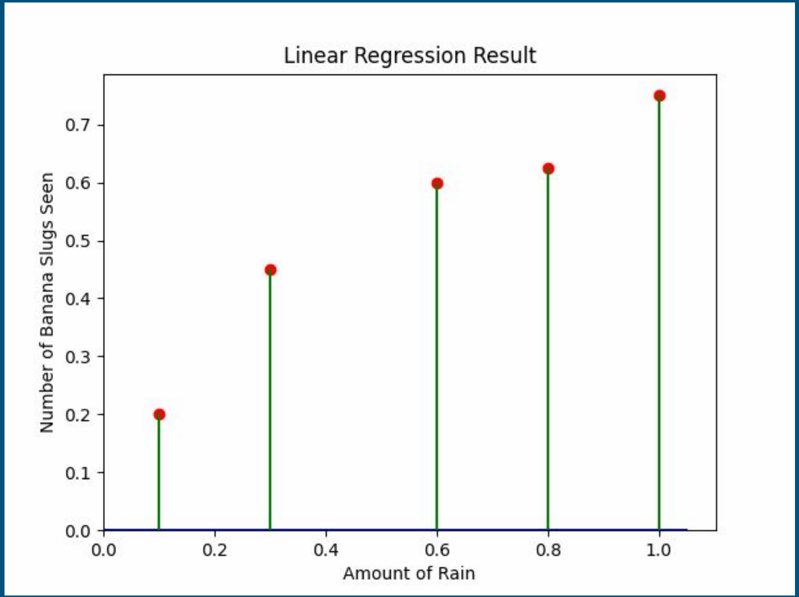
Now you understand linear regression with gradient descent!!!!!

It is very simple to go from only optimizing the y-intercept (b) to optimizing the slope (m) and y-intercept simultaneously!

- Every iteration:
  - Calculate: $\frac{dSSR}{db}$ and $\frac{dSSR}{dm}$
  - Update b and m with:

$$b_{new} = b_{old} - lr * \frac{dSSR}{db}$$

$$m_{new} = m_{old} - lr * \frac{dSSR}{dm}$$



https://bit.ly/LinearRegressionSCAI

Let's put all of that knowledge together with Python

Go to the code!

Higher dimensions?!?!

What if we want to use additional data to predict the number of banana slugs, such as the temperature? Is that possible?

Yes!

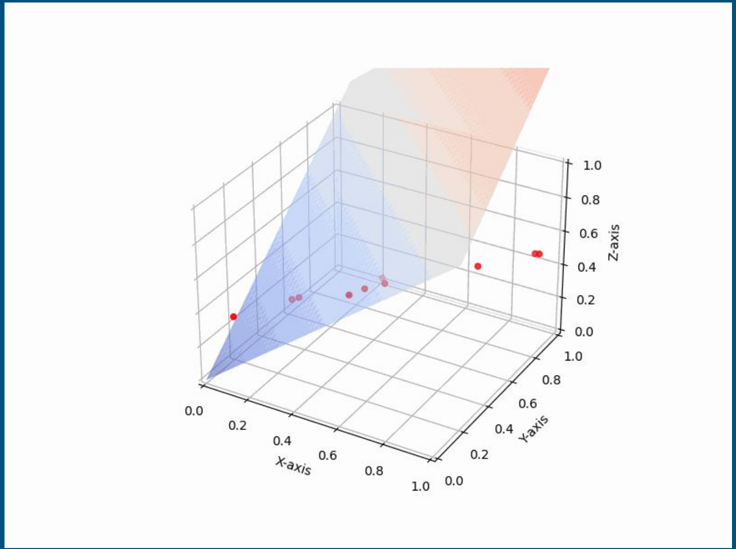
- Using 1 feature to predict 1 output is fitting/optimizing a 2D line.
  - $\hat{y} = mx + b$

Higher dimensions?!?!

What if we want to use additional data to predict the number of banana slugs, such as the temperature? Is that possible?

Yes!

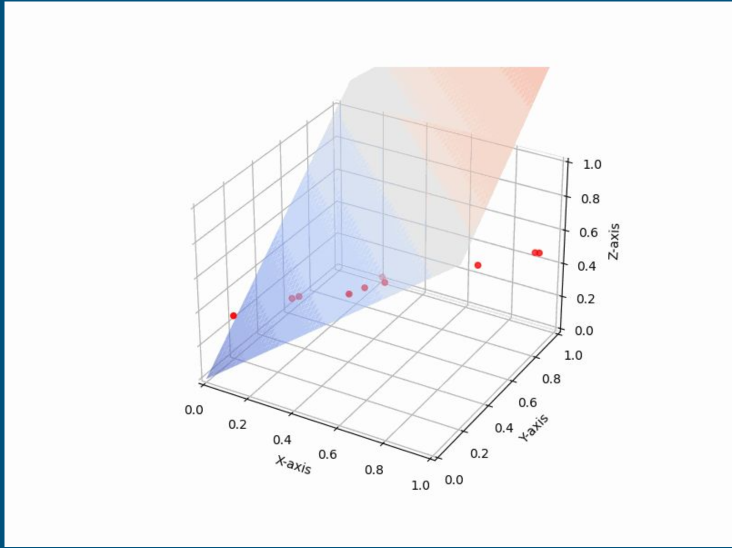
- Using 1 feature to predict 1 output is fitting/optimizing a 2D line.
    - $\hat{y} = mx + b$
- Using 2 features to predict 1 output is fitting/optimizing a 3D plane.
    - $\hat{y} = m_1 x_1 + m_2 x_2 + b$



Higher dimensions?!?!

Another way you may see this written is instead of m for *slope*, we replace it with w for *weight*. Then instead of calling b the *y-intercept* we will call it the *bias*. Then for any linear regression model with n input features the equation is:

$$\hat{y} = \sum_{i=1}^{n} w_i x_i + b$$



Thank you!

Keep in Touch :

discord.gg/santacruzai

linktr.ee/santacruzai

Feedback :