O'REILLY®

Essential Math for Data Science

Take Control of Your Data with Fundamental Calculus, Linear Algebra, Probability & Statistics

Early Release

RAW & UNEDITED

Hadrien Jean

1. 1. Elementary Algebra
    a. 1.1 Variables
        i. 1.1.1 From Computer Programming to Calculus
        ii. 1.1.2 Unknowns
        iii. 1.1.3 Dependent And Independent Variables
    b. 1.2 Equations And Inequalities
        i. 1.2.1 Equations
        ii. 1.2.2 Inequalities
        iii. 1.2.3 Hands-On Project: Standardization and Paris Apartments
    c. 1.3 Functions
        i. 1.3.1 From Equations To Functions
        ii. 1.3.2 Computer Programming And Mathematical Functions
        iii. 1.3.3 Nonlinear Functions
        iv. 1.3.4 Inverse Function
        v. 1.3.5 Hands-On Project: Activation Function
2. 2. Math On The Cartesian Plane
    a. 2.1 Coordinates And Vectors
        i. 2.1.1 Intuition

ii. 2.1.2 Coordinates
iii. 2.1.3 Geometric Vectors: Magnitude And Direction
iv. 2.1.4 Hands-On Project: Images As Model Inputs

b. 2.2 Distance formula
i. 2.2.1 Definitions
ii. 2.2.2 Hands-On Project: k-Nearest Neighbors

c. 2.3 Graphical Representation of Equations And Inequalities
i. 2.3.1 Intuition
ii. 2.3.2 How To Plot Equations
iii. 2.3.3 Solving Equations Graphically
iv. 2.3.4 Inequalities

d. 2.4 Slope And Intercept
i. 2.4.1 Slope
ii. 2.4.2 Intercept

e. 2.5 Nonlinear functions
i. 2.5.1 Definition
ii. 2.5.2 Function Shape

f. 2.6 Hands-On Project: MSE Cost Function With One Parameter
i. 2.6.1 Cost function

            ii. 2.6.2 Mathematical Definition of the Cost Function
            iii. 2.6.3 Implementation
3. Index

Essential Math for Data Science

1ST EDITION

Take Control of Your Data with Fundamental Calculus, Linear Algebra, Probability, and Statistics

Hadrien Jean

Essential Math for Data Science

by Hadrien Jean

Published by O’Reilly Media, Inc. , 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editors: Jessica Haberman and Sarah Grey

Production Editor: FILL IN PRODUCTION EDITOR

Copyeditor: FILL IN COPYEDITOR

Proofreader: FILL IN PROOFREADER

Indexer: FILL IN INDEXER

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Rebecca Demarest

August 2020: First Edition

Revision History for the First Edition

- 2019-10-10: First Release

See http://oreilly.com/catalog/errata.csp?isbn=9781098115562 for release details.

Chapter 1. Elementary Algebra

In this first Chapter, you will learn about basic mathematical building blocks. We will start with variables and then use them to construct equations. Finally, we will see that equations can express functional relations.

1.1 Variables

In algebra, variables are letters (like a, b, x…) representing numerical quantities. We generally use a single letter with an italic typeface as variable names (such as x).

This allows us to formulate general rules that work for any values. It is also useful to express relationships (through equations for instance) or to deal with unknowns (and eventually find their values).

Symbols representing unchanging numerical quantities (like π for instance) are called *mathematical constants* or just *constants*.

1.1.1 From Computer Programming to Calculus

In computer programming, a variable is a reference to a piece of stored information. Usually, it is like a pointer to a specific location where we can access the information. Its name is used to reference the stored value.

In mathematics, the concept is similar. We want to be able to write general rules that cover various situations. To do that we need a way of representing objects that can take various values.

Let’s start with an example. We’ll work on a dataset giving the global temperatures averaged from thousands of meteorological stations around the world. Here is a plot representing temperature anomalies (anomalies are deviations from a reference):

```
# See https://github.com/jjrennie/GHCNpy
data = pd.read_csv("data/ghcn-m-v1.csv")

data1 = data.replace(-9999, np.NaN)

data_month = data1.groupby(['year',
'month']).mean().reset_index()
data_month_mean = data_month.iloc[:, 2:].mean(axis=1)

data_year = data1.groupby(['year']).mean().reset_index()
data_year_mean = data_year.iloc[:, 2:].mean(axis=1)
```

```
plt.figure()
plt.plot(np.arange(data_year_mean.shape[0]) + 1880,
data_year_mean / 100)
plt.xlabel('Date (Year)')
plt.ylabel('Temperature anomalies (degree Celsius)')
```

```
Text(0, 0.5, 'Temperature anomalies (degree Celsius)')
```

image::images/output_12_1.png

We have two variables here, the date and the temperature. Let’s call the temperature variable T. We can see that, for example, in 1953

$T = -0.32$ while in 2003 $T = 0.19$. This shows that the value associated with the variable T is changing.

We can also consider the date as a variable. It changes as well.

Variables are very powerful because they can express ideas using objects that are not fixed. In our example, we can take the physical concept of temperature, assign it a variable with a short name (T) for convenience, and characterize it mathematically. For instance, we can say that T increases over time. If the exact relation between time and temperature was known we could have written an equation synthetizing this relation (see next chapter on equations and inequalities).

1.1.2 Unknowns

Sometimes, you might encounter the name *unknown* to talk about variables. Like variables, unknowns are letters standing for a value that we don't know. It is useful to create equations using values that we don't know. One goal can be to solve the equation and find the unknown value.

1.1.3 Dependent And Independent Variables

You can also encounter the terms dependent and independent variables. Dependent variables are the variable of interest, which depends on the other variables called independent variables. For instance, in the dataset expressing the temperatures as a function of time, we are interested in the evolution of the temperatures. This is the dependent variable. Independent variables are all the other variables that can affect the dependent variable. When we look at the temperatures as a function of time, we want to see the effect of time on the temperature and not the opposite. The date is, in this case, an independent variable. The dependent variable is usually plotted on the y-axis in a cartesian plane (See ???).

1.2 Equations And Inequalities

We can use variables to construct expressions like equations or inequalities. Equations can be used to find value of an unknown and to express relations between entities.

1.2.1 Equations

Equations express relations between elements through equality. For instance, we can have:

$$2 + 2 = 4$$

Or using variables:

$$y = ax + b$$

which is the equation of a line defined by the parameters a and b.

UNIVARIATE EQUATIONS

Equations can use single variable, like in:

$$2x = 4$$

In this case, *solving the equation* means asking the question “what value of x gives 4 when this is multiply by 2?”.

It is possible to write real-world problems as equations. The synthetic and expressive notation helps to solve problems.

Example

Sonja found a great recipe book, but it uses the metric system. She needs to convert the quantities from grams to cups. She looked on the Internet and saw that 1 cup of flour corresponds to 120 grams of flour. From the recipe, she needs 200 grams of flour, so how many cups should she use? Let's formulate this problem under mathematical notation. She wants to know how many cups (the unknown, let's say x) multiplied by the weight of flour of each cup (120 grams) gives 200 grams.

$$120x = 200$$

This is an equation. It means: "what number gives 200 when we multiply it by 120?". 120 is the weight in grams of 1 cup of flour. 200 is the weight in grams that we want to get. But what is x? x is the *unknown*: the numerical value that we don't know in our problem. *Solving* the equation means finding the value of x that satisfy the expression. This is the number of cups Sonja needs to get 200 grams of flour.

So, $x \times 120$ corresponds to the total weight of flour Sonja will use. Since one cup of flour corresponds to 120 grams she multiplies the number of cups by the weight of one cup to get the total weight. For instance, if $x = 1$, she uses 1 cup of flour, thus 120 grams of flour but that's not enough. If $x = 2$, she uses 2 cups of flour: $120 \times 2 = 240$. That's too much. It looks like x is between 1 and 2.

But how can we solve this equation? The idea is to change the elements, keeping in mind that there is an equality (so you have to apply the same operations to each side of the equation or you will break the equality), in order to have only the unknown on one side ($x = something$). In our case, we can divide each side by 120 to end up with x alone on one side:

$$\begin{aligned} 120x &= 200 \\ \frac{120x}{120} &= \frac{200}{120} \\ x &= \frac{200}{120} \\ &\approx 1.67 \end{aligned}$$

So we need 1.67 cups to get 200g of flour. We used equations to determinate the value of the unknown x.

EXAMPLES VS PROOFS

Examples are not mathematical proofs. You can still find cases where the rule can be broken. However, the goal of this book is to help you get more intuition and practical knowledge of mathematics. That's why this book will emphasize examples over proofs.

MULTIVARIATE EQUATIONS

How does it work if equations have more that one variable? For instance:

$$y = 2x - 3$$

This equation expresses the relation between x and y. We can find y when we know x and we can find y when we know x. For instance, if $x = 5$:

$$\begin{aligned} y &= 2x - 3 \\ y &= (2 \times 5) - 3 = 7 \end{aligned}$$

Or if $x = -3.75$:

$$\begin{aligned} y &= 2x - 3 \\ y &= 2 \times -3.75 - 3 = -6.75 \end{aligned}$$

Or if $y = 8$:

$$\begin{aligned}
y &= 2x - 3 \\
8 &= 2x - 3 \\
8 + 3 &= 2x - 3 + 3 \\
8 + 3 &= 2x \\
\frac{8 + 3}{2} &= \frac{2x}{2} \\
\frac{11}{2} &= x \\
x &= 5.5
\end{aligned}$$

RELATION BETWEEN MULTIPLE VARIABLES

In addition to helping to find values of unknowns, equations express a relation between entities. We can reframe the example of cups and weight of flour using two variables. If x is the number of cups of flour and y the corresponding total weight of flour, we have:

$$y = 120x$$

There is a relation between the number of cups and the weight and this relation is a factor of 120.

Let's take another example. If you want to convert meters to feet, you will need to know the relation between these two units. In this case, we know that 1 meter is approximately equal to 3.28 feet. If we call m the number of meters and f the number of feet, we can write this relation like that:

$$f = 3.28m$$

This is one way to express the relation between meters and feet. You can also consider it a *mapping* between the number of meters and the number of feet, that is to say, as a list of pairs of values.

Table 1-1. Relation between meters and feet.

meters	**feet**
1	3.28
2	6.56
3	9.84
…	…
10	32.8
…	…

We will dive into more detail about the idea that equations represent relations in section "1.3 Functions".

SOLVING EQUATIONS: BASIC TRANSFORMATIONS

Equations are quite flexible. We will see some basic transformations that we can make to solve them. You can conceive that an equation is like a tray balanced with equal weights on each side. For instance,

$$f = 3.28m$$

This equality is right. But if we look at our mapping table above, we could also write:

$$2f = 6.56m$$

And since $3.28 \times 2 = 6.56$ we can *replace in the equation*:

$$2f = 2 \times 3.28m$$

And here is the pattern:

$$\begin{aligned} 3f &= 3 \times 3.28m \\ 4f &= 4 \times 3.28m \\ \ldots \\ n \times f &= n \times 3.28m \end{aligned}$$

With our tray, we want to keep the same amount of weight on each side. We start with $1f$ on the left side and $3.28m$ on the right side. If we multiply by two what is on the left side ($2f$), we need to multiply the right side by two as well ($2 \times 3.28m$). This means that the equation is still true if you do the same operation on both sides.

You can also think of it as *changing weight of the sides* by doing these operations. For instance, let's start with

$$f = 3.28m$$

If we subtract f from both sides we'll have:

$$\begin{aligned} f - f &= 3.28m - f \\ 0 &= 3.28m - f \end{aligned}$$

This is convenient: we took f from one side and put it on the other side just changing the sign (from $+$ to $-$).

We can use these arrangements to get the value of one variable. For instance, if we want to get the value of m, we can do the following:

$$\begin{aligned} f &= 3.28m \\ \frac{f}{3.28} &= \frac{3.28m}{3.28} \\ \frac{f}{3.28} &= m \\ m &= \frac{f}{3.28} \end{aligned}$$

Feel free to take some time to practice if you are not comfortable with these transformations.

DIVISION ISSUES

Be careful! You can't do evrything.

Division by 0

You just can't divide by 0. Remember that division is the inverse of multiplication. For instance, if we try to do:

$$\frac{7}{0}$$

We ask what number, multiplied by 0, give 7. This is impossible. There is no such number.

Division by a variable

You could be tempted to divide each side by the variable to simplify an equation. However, doing so can hide answers, because when you divide by a variable, you assume that this variable is not equal to 0 (which as we just saw, is impossible). For instance,

$$\begin{aligned} x^2 &= x \\ \frac{x^2}{x} &= \frac{x}{x} \\ x &= 1 \end{aligned}$$

We just missed a solution here: 0 ($0^2 = 0$) because we implicitly assumed that $x \neq 0$ when we decided to divide by x.

1.2.2 Inequalities

Inequalities refer to a relation between numerical values that are different. For instance:

$$x < 3$$

means that the value of x is less than 3.

We can apply the same kind of operations on inequalities as on equations. For instance, if

$$x + 3 > 7$$

We can remove 3 from both sides of the inequality and get:

$$x + 3 - 3 > 7 - 3$$

thus

$$x > 4$$

Inequalities are useful to express numerical quantities relatively to each other (for instance, $x > y$ means that the value of the variable x is greater than the value of the variable y).

1.2.3 Hands-On Project: Standardization and Paris Apartments

We will study one use case example using equations: a preprocessing technic called standardization. Then we will take a real-world example of predicting Paris apartment price.

1.2.3.1 STANDARDIZATION

Equations are one of the basic tools of a data scientist. They can express general rules to transform data. Let's take an example.

Feature scaling refers to technics used as preprocessing steps (processing before data is fed to a model) to uniformize the range of our features. For instance, we want to avoid using one feature with values ranging from 0 to 0.1 and another from -10^6 to 10^6. Large differences in feature scales can be problematic in machine learning. Algorithms are trained by calculating error as a difference between a value predicted by the algorithm and the true value. Feature scales have an impact on this error and normalization can speed training of the algorithms[1].

The equation of standardization is the following:

$$x_{new} = \frac{x - \mu}{\sigma}$$

with the variables x_{new} being the standardized data, x the raw data, μ and σ respectively the mean and the standard deviation of x. The mean is the average of all values of x so $x - \mu$ is the differences between x and the average. The standard deviation tells how spread are the data (refer to <<>> for more details on mean and standard deviation). Dividing by the standard deviation insure that x_{new} will have a standard deviation of 1 (because $\frac{\text{standard deviation}}{\text{standard deviation}} = 1$).

The equation gives us the preprocessing rule: to get x_{new} we remove μ from x_{old} and divide by σ.

Let’s implement this equation:

```
def standardization(x):
    x_new = (x - x.mean()) / (x.std())
    return x_new

np.random.seed(439)
x = np.random.normal(10, 5, 100)
print(x.mean(), x.std())
```

```
9.441811582760224 4.964076710538008
```

```
x_new = standardization(x)
print(x_new.mean(), x_new.std())
```

```
3.5416114485542493e-16 0.9999999999999998
```

We can see that once we have standardized x, its mean is 0 and its standard deviation is 1.

1.2.3.2 PARIS APARTMENTS

Equations are deeply related to *mathematical models*. A mathematical model is a formulation of the relation between elements using mathematical terms. We will take an example of modeling Paris apartment prices. We want to find a way to predict the price of a Paris apartments from several parameters (the surface, the number of rooms, etc.). We try to find a rule to weight the two features (square meters and number of rooms) to predict the price. However, The link between the features and the price can be written as an equation:

$$a \times area + b \times room = price$$

The variables a and b are telling us the importance of each feature. A large positive value of a means that the area has a big impact on the price, with large values corresponding to large prices. A large negative value indicates an opposite relation, with large areas corresponding to small prices.

We start by looking at two apartements:

apartments.

Square Meters	Number of rooms	Price
42	2	545000
39	1	495000

We will use the Sklearn library to find the value of a and b with linear regression.

```
from sklearn.linear_model import LinearRegression

x = np.array([[42, 2], [39, 1]])
y = np.array([545000, 495000])
```

```
linearRegressor = LinearRegression()
linearRegressor.fit(x, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True,
n_jobs=None, normalize=False)
```

```
plt.scatter(x[:, 1], y)
plt.plot(x[:, 1], linearRegressor.predict(x))
```

image::images/output_36_1.png

```
print(linearRegressor.coef_)
print(linearRegressor.intercept_)
```

```
[15000.  5000.]
-94999.99999999977
```

We find from these two listings that the following values of a and b are good:

$$\begin{aligned} a \times area + b \times room &= price \\ (15000 \times area) + (5000 \times room) - 95000 &= price \end{aligned}$$

Now that we have our model, we can take any apartment, look at the features and predict its price. The equation is a model in the sense that the values are meant to model a relation between parameters and price.

Let’s check with the first apartment:

```
(15000 * 42) + (5000 * 2) - 95000
```

```
545000
```

And with the second one:

```
(15000 * 39) + (5000 * 1) - 95000
```

```
495000
```

It looks like the rule is perfect! Not so fast! Does it mean that our model is good? Let’s try with a third apartment that we didn’t use to create our model.

Square Meters	Number of rooms	Price
47	2	557000

```
(15000 * 47) + (5000 * 2) - 95000
```

```
620000
```

The predicted price is 620,000 euros while the real price is 557,000 euros. The model is not that good after all. We didn't use enough data to create it and the relation we found was specific to the two apartments we used. We call that *overfitting*.

In machine learning, the weights (a and b in our previous example) will be learned automatically.

MODEL AND TERMINOLOGY

With the prices of Parisian apartments, the equation is a model. A model is a mathematical description of a phenomenon using mathematical concepts like variables, equations, functions, and so on.

The variables $area$ and $room$ are the model features (also called independent variables). They are the input of the model: we want to predict prices according to these variables.

The variables a and b are the model parameters (or weights). They are internal to the model and learned while the model is trained. They are like configuration variables (*https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/*).

Being able to read, write and understand this kind of mathematical equations is very helpful to see how a model is working, or eventually, why it is not working. It will help you to use librairies providing implementations of these concepts, and give you light from the model selection phase to the debugging of your models.

1.3 Functions

As in computer programming, the term function describes a set of rules used to transform inputs into outputs.

image::images/function.png

1.3.1 From Equations To Functions

We saw in section "1.2 Equations And Inequalities" that equations can be seen as mapping pairs of values. In that sense, equations and functions are very similar: they can both express a relation between variables.

The difference between equations and functions is more a matter of semantics and context. As we saw, an equation is an expression that states a relation of equity between two elements (left and right sides).

FUNCTIONS MUST OUTPUT ONE SINGLE VALUE

We saw that functions and equations imply a relation between variables. For functions, this mapping has to be many-to-one. This means that functions can have multiple inputs but must output a single value per input.

This means that multiple inputs can give the same output value. This is for instance what we saw with constant functions: every value of x would give the same value of y.

However, it is not possible to have a single input that can give multiple alternatives as output.

No variable

Without any variable, we can state that:

$$2 + 7 = 9$$

This expression only shows the relation between two numerical quantities, so there is no input and no output. This is not a function.

One variable

In other cases, only one variable is involved. For instance:

$$y = 2$$

This equation expresses the value of the unknown variable y (here, 2). We can say that y is the output of a function if we artificially add the input, for instance as:

$$y = 0x + 2$$

To get an idea of the mapping represented by this function, we can compute y for some values of x. At $x = 0$, $y = 0 \times 0 + 2 = 2$, at $x = 5$, $y = 5 \times 0 + 2 = 2$ and so on. We can see that y do not depends on x. The value of y will be the same for all values of x. This corresponds to an horizontal line at $y = 2$ (we will see more details on the graphical representation of function in section "2.3 Graphical Representation of Equations And Inequalities").

The functions involving only one variable (the other being equal to 0) are called constant functions and have the form:

$$f(x) = c$$

with c being a constant.

If we convert equations with only one variable to computer programming functions, the output does not depend on the input:

```
def one_variable(x):
    return 0 * x + 2
```

Or just

```
def one_variable():
    return 2
```

This emphasizes the relation between computer programming functions and mathematical notation. Here, we have just implemented a constant function.

Two variables

If the equation contains two variables, we can consider it as a function with one input and one output (and we can represent it on the Cartesian plane — see Chapter "2.3 Graphical Representation of Equations And Inequalities" — with one variable displayed on the x-axis and the other on the y-axis).

For instance,

$$y = -4x + 7$$

is an equation and also a function expressing the relation between the variable x and y. Furthermore, consider this statement:

$$f(x) = x^2$$

This is an equation, but also a function definition. There are two variables: x and $f(x)$. The input is x, f is the name of the function and x^2 is the function itself, which also corresponds to the output of the function.

MORE THAN TWO VARIABLES

Equations can have more than two variables. For instance,

$$y = x_0 + x_1 + 2$$

is an equation with 3 variables: two independent variables, x_0 and x_1 , and a dependent variable, y. This means that we would need more than two dimensions (the x-axis and y-axis) to represent it on a graph. This is similar to our previous example of apartment price as a function of several variables (area, number of rooms).

There are multiple ways to convert this equation to a computer programming function. For instance:

```
def multi_variables(x0, x1):
    y = x0 + x1 + 2
    return y
```

But maybe we want other inputs and outputs:

```
def multi_variables(x0, y):
    x1 = y - x0 - 2
    return x1
```

Either way, we need to input two of the three variables to get the third. Depending to which two we enter, the equation is the same but written in another way:

$$\begin{aligned}
y &= x_0 + x_1 + 2 \\
x_0 &= y - x_1 - 2 \\
x_1 &= y - x_0 - 2
\end{aligned}$$

These different ways to write the same equation correspond to different functions that can be created from it.

Dimensionality

In the last example, we had an equation expressing the relation between features x_0 and x_1 and the dependent variable y. Let's say that this equation corresponds to some parameters from our Paris apartment example: the area (x_0), the number of rooms (x_1) and the price (y). In some situations, this kind of equation comes from data and we want to fit parameters to data. This means that our dataset should contain the features and the corresponding prices for a lot of apartments. The *dimensions* of a dataset are the number of features or independent variables it has. For example, in

$$y = ax_0 + bx_1 + cx_2$$

y is the dependent variable and x_0, x_1, and x_2 are the independent variables. The number of dimensions is thus three.

1.3.2 Computer Programming And Mathematical Functions

Linear functions are functions that are graphically represented by lines. We have seen examples of linear functions. These functions have a rate of change (corresponding to the slope of the line) that is continuous.

The simplest cases are constant functions. For instance,

$$y = 2$$

The value of y is always the same. It does not vary according to the value of x. This means that we'll have a horizontal line for y.

Equations with non-0 x can also be linear. For instance, let's define a function that multiplies the input by two:

```
def multiply_by_two(x):
    y = x * 2
    return y

print(multiply_by_two(4))
print(multiply_by_two(34))
```

```
8
68
```

We can represent the equation graphically to show that this is a line (see section "2.3 Graphical Representation of Equations And Inequalities"). This function is a line, as shown in ???.

image::images/2x.png

To summarize, we can go from mathematical notation to code and the opposite. This is a powerful way of thinking, because mathematical notation can help you to formalize complex things while code is a practical approach that allows you to implement them and discover more insights about them.

image::images/function_example.png

FUNCTION NOTATION

When we write $f(x) = 2x$, this is nothing more than an equation. $f(x)$ corresponds to the output of the function f when we give it the input x. It is also standard to write $y = 2x$ because if we plot it we'll use x-axis and y-axis. In this case, it returns $2x$ when we give it x in input. As we have seen, x is a variable. We use this notation to say that this relation is right for any numerical value of x. For instance, if $x = 2$, we'll have $f(2) = 4$; or if $x = 4.1$, $f(4.1) = 8.2$ and so on.

1.3.3 Nonlinear Functions

Up to now, we have only looked at functions that are represented as lines. These are called *linear functions* for this reason and are defined by equations like:

$$y = ax + b$$

This means that the value of y decreases or increases in a linear way when we decrease or increase x.

With nonlinear functions, the y is not changing linearly as a function of y. For instance, with the function $sin(x)$, represented on ???, y increases and decreases repeatedly as x is increased

image::images/sine-function.png

We will see more about nonlinear functions in Chapter "2.5 Nonlinear functions".

1.3.4 Inverse Function

The *inverse* of a function is a function that allows us to go from $f(x)$ back to x. For instance, if we have:

$$f(x) = 3x + 8$$

What is the function g that takes $f(x)$ as its input and returns x? We can look at a function as a **step by step** procedure:

1. Multiply by 3. . Add 8.

```
def step_by_step(x):
    new_x = x * 3
    new_x += 8
    return new_x

step_by_step(2)
```

```
14
```

If we want to do the opposite, we have to take the steps backward and do the opposite:

1. Remove 8.
2. Divide by 3.

```
def step_by_step_inverse(y):
    new_y = y - 8
    new_y /= 3
    return new_y

step_by_step_inverse(step_by_step(2))
```

```
2.0
```

Thus the function `step_by_step_inverse()` takes the output of the function `step_by_step()` and gives back the input of the function `step_by_step()`.

Functions are important mathematical objects. As a data scientist or a machine learning practitioner, you know that equations and functions are everywhere. For example, you might need to choose between a normal equation or a gradient descent algorithm for your linear regression, try a state-of-the-art activation function for your neural network, or find the right cost function for the problem you are trying to solve.

1.3.5 Hands-On Project: Activation Function

THE RELU ACTIVATION FUNCTION

To finish this chapter, let's take the example of a famous function in machine learning: the *rectified linear unit*, also known as ReLU. This is a good example of an almost linear function. It is defined as $y = 0$ if $x < 0$, and as $y = x$ if $x >= 0$.

```
def relu(x):
    return np.maximum(x, 0)
```

The `relu()` function can take an array of values as input. For each value, if $x < 0$, the function returns 0, and if $x > 0$, the function outputs x. Using the function `maximum()` from Numpy allows us to compare each value from the array with 0 and keep the maximum. For instance,

```
np.maximum([-2, -1, 0, 1, 2, 3], 0)
```

```
array([0, 0, 0, 1, 2, 3])
```

When $x < 0$, the `maximum()` function will return 0 and when $x > 0$, it will return x. This is like two functions: a constant function for $x < 0$ and an identity function for $x > 0$. Let's plot this function:

```
# create x and y vectors
x = np.linspace(-2, 2, 100)
y = relu(x)

# choose figure size
plt.figure(figsize=(6, 6))

# Assure that ticks are displayed with a step equal to 0.5
ax = plt.gca()
ax.xaxis.set_major_locator(ticker.MultipleLocator(0.5))
ax.yaxis.set_major_locator(ticker.MultipleLocator(0.5))

# draw axes
plt.axhline(0, c='#A9A9A9')
plt.axvline(0, c='#A9A9A9')

# assure x and y axis have the same scale
plt.axis('equal')

plt.plot(x, y)
```

image::images/output_84_1.png

ReLU function is extensively used as activation function in neural networks. It is efficient to compute and neural networks train faster than with other activation function like tanh[2].

This concludes this chapter on the basics of algebra. We learned about variables, equations, and functions. We can see that all of this is related. Equations are made of variables and numerical values; some equations can be considered functions. We saw some bridge between computer science and mathematics, so feel free to experiment math with code to get a better intuition on the concepts. We can see functions as a rule converting the input into output, or as a mapping (list of pairs of values), or as a graphical object. There is more on that in Chapter ???), where we will see the importance of graphical representations as another mathematical tool.

---

1 Ioffe, Sergey; Christian Szegedy (2015). ‘Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift’.

2 Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems. 2012.

Chapter 2. Math On The Cartesian Plane

We saw in chapter Chapter 1 that functions can be seen as mappings between pairs of numbers when there are two variables (such as x and y). There are a lot of ways to visualize pairs of numbers. For instance:

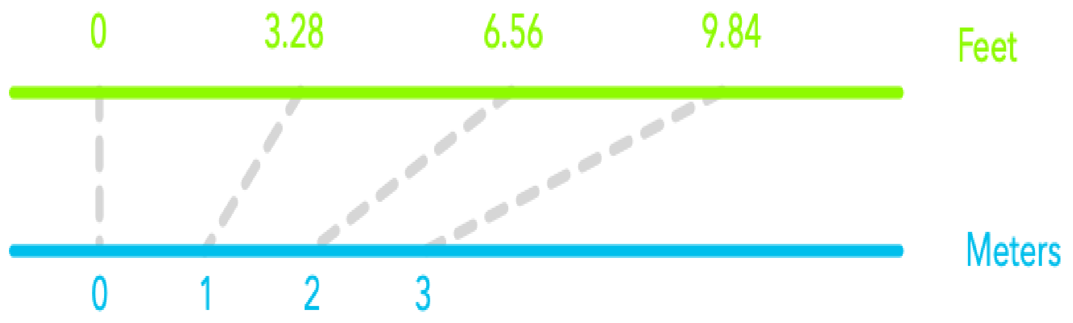

Figure 2-1. One way of representing pairs of numbers.

We used one direction (left/right) to represent both variables but it is possible to use the two directions (left/right and bottom/up).

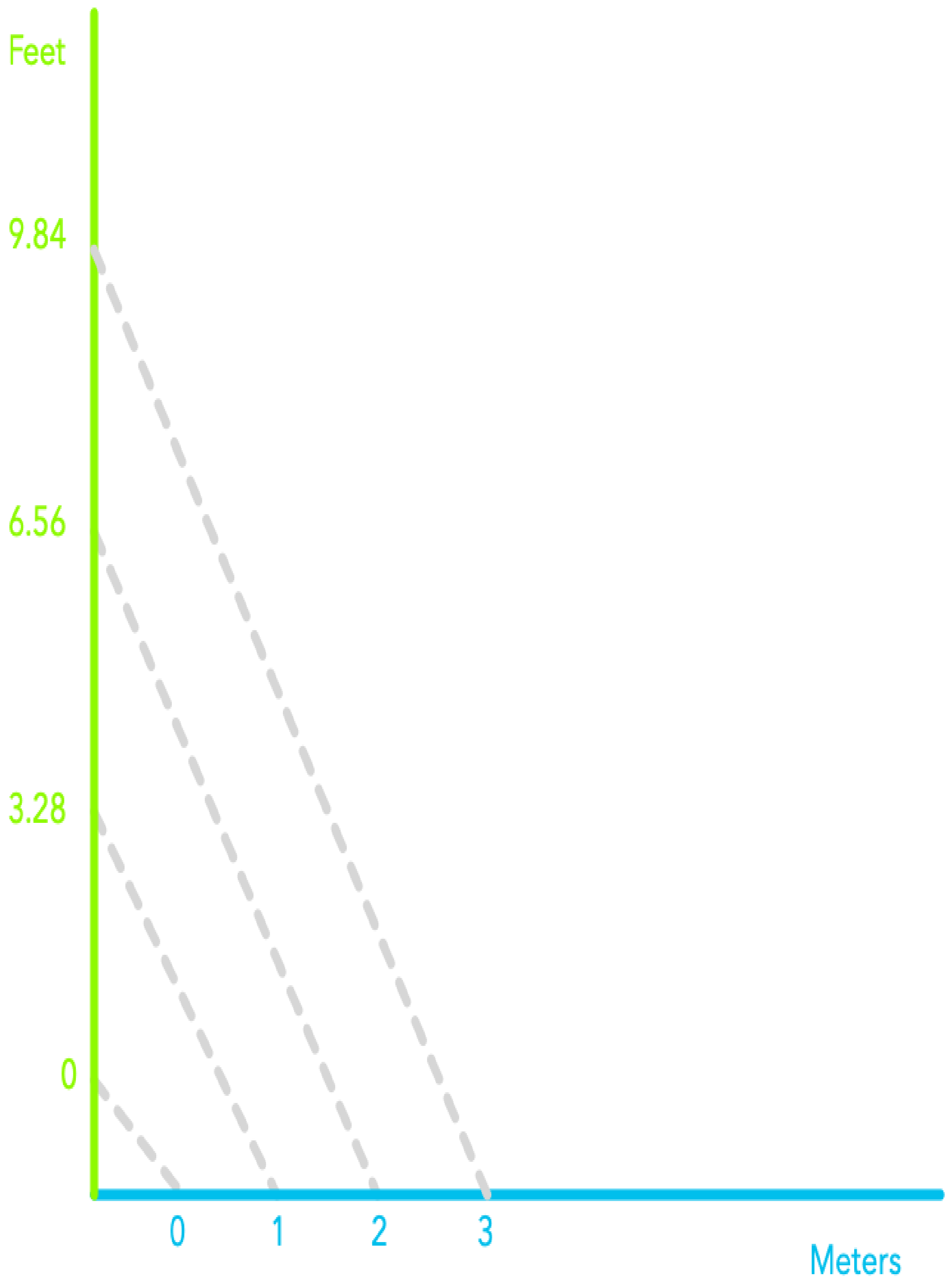

Figure 2-2. Using left-right and bottom-up axes.

This is a more common way to represent pairs of values. Now that we are using both directions, each point in this space has two coordinates: one describing its position on the left/right axis and another one describing its position on the bottom/up axis. So we can draw points from our mapping with these coordinates (and consider the point (0, 0) as the origin).

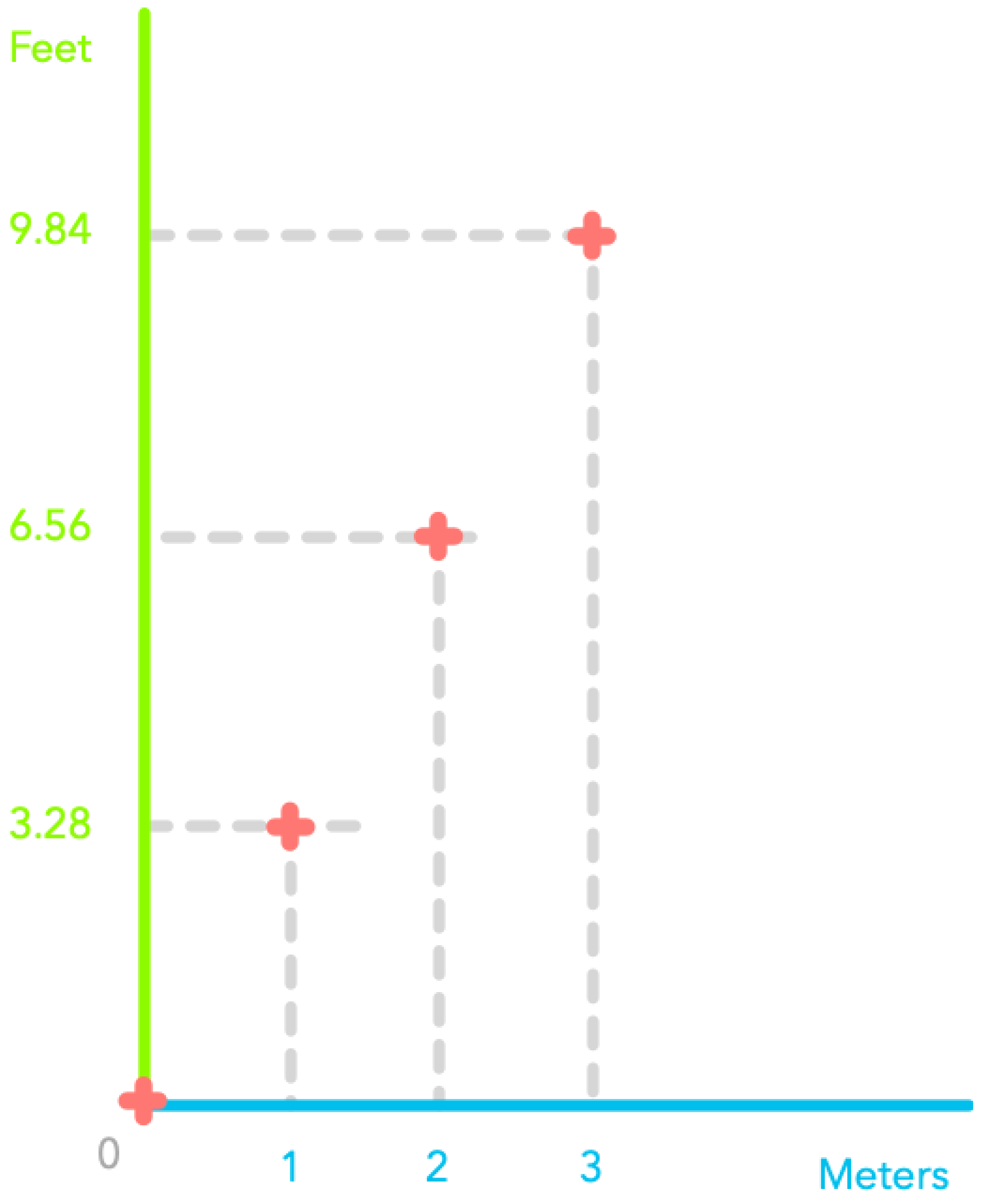

Figure 2-3. Points coordinates in these axes.

This representation of a two-dimensional space is standard and uses a coordinate system that we call a *Cartesian plane* (after the philosopher and mathematician René Descartes). The vertical and horizontal lines are the *coordinate axes* (or *coordinate basis*), usually labeled x for the horizontal ...

Index