Chapter 8: Models and functions

Submitted on: 04/05/2018
Word count: 5033 (including figure captions) +10 figures

Learning outcomes:

- Understand the concept of a model as it is used in science and statistics.
- Understand the notation of functions.
- Understand the relationship between mathematical functions and graphs, and how to translate between the two.
- Introduce the basic process of model based statistical reasoning.

Introduction

Models are central to statistical and scientific reasoning. In science, a **model** is formal representation of a system. Models are ubiquitous in scientific theory and abound in psychology. The more precise a model is, the more useful it is, and so it is very common to express models in the language of mathematics (or in the case of computational models, computer programs). In statistics we rely primarily on these **mathematical models—**characterisations of systems in the language of mathematics. One of the most common ways of expressing a model is as a function (or set of functions). In the next section, we introduce the concept of mathematical functions, which will feature heavily in this textbook. We then relate functions to graphs and show how the two can be represented in terms of the other. Finally, we return to the concept of the mathematical model, and give an example of its use in statistical inference.

Primer on functions

A **function** is a very important mathematical object that is central to understanding statistics and many topics in any scientific discipline. Functions are actually very straightforward to understand. This review is meant to provide a basic introduction to functions along with some exercises that should help you understand functions more completely and evaluate your own understanding. If you are already familiar with functions from previous maths courses you may want to skip this section. However, if you aren't comfortable with concepts like graphing a function, or writing a function for a curve, you might want to read on.

What are functions?

Formally, functions are a mapping of some set of values to some other set of values. In other words, a function takes some set of input values, and assigns them to some set of output values. That sounds awfully broad, doesn't it? Well, it's actually the breadth of the concept that gives it much of its power. Functions can be applied to a huge range of topics, and, consequently, can be used to describe and understand those topics.

Simple functions

Let's get a little more specific and think of a really simple function. Imagine we were to play a game where I give you a value, and you give me a value back. I'll give you a couple of examples, and then you can complete the rest.

For 1 the answer is 1
For 2 the answer is 2
For 3 the answer is 3

Ready?

For 4 the answer is?
For 5 the answer is?
For 147 the answer is?
For flower the answer is?
For sleepy panda the answer is?

Answers: [4, 5, 147, flower, sleepy panda].

I doubt you had too much trouble with this game. The answers probably came to you reasonably quickly. So, how'd you do it? I'm going to guess that you looked at the examples, noticed that each item went with itself, and made up a rule like, "for any value, the answer is that value". What you did was to formulate a function that gave the rule for playing the game. Basically, for any input, the output is that input. We could write this as a fancy function:

$$\forall x\, f(x) = x \tag{8.1}$$

which is read: "for all x, *f* of x equals x". In other words, *f* is a function that takes in a value (the specific value that a function takes is often called its *argument*), and outputs that same value. Notice that *f* applies to anything. We could input a number, a letter, a word, a sentence, a sequence of drum beats, or whatever as an argument, and *f* would give us an output. To represent this property, we use *x* as a variable and the upside-down A (∀; which is read, "for all"), and get a function that basically says, for any possible thing (which we represent with the variable *x*)[1] that is an argument to this function, produce that same thing as the output.

In all honesty, that's really all there is to functions. Everything else is just basic extensions on the above. Let's look at a few extensions, though, just to get our confidence up.

---

[1] Remember that a variable is a symbol that can take a value. We could be very specific and specify a precise value for a variable (e.g, *x*==1; or the value of *x* is one), or a range of values (e.g., 5<x<10; or the value of x is something more than 5 and less than 10), or we could use the variable to communicate that we want to talk about something without specifying a precise value for that thing. For example, let's say I want to say that all the participants in my experiment produced a score, *x*, then I am using *x* to specify participants' scores in my experimental task without talking specifically about any specific participant's score.

Here's another function:

$$\forall x\ f(x) = x + 2 \tag{8.2a}$$

Can you read this function? You should be able to. If you can't review the paragraph directly above and come back and try again. Don't worry, I'll wait…

All done? Cool. Hopefully, you read the last function as something like, "for all *x*,*f* of *x* equals *x* plus 2". Hopefully, you also understand what this function means: for any *x* value that is input to the function, the output is *x* with 2 added. Let's try this one:

$$\forall x\ g(x) = x + 2 \tag{8.2b}$$

Hold on, why did I use g instead of *f*? Well, think about it a second: x is just a variable, right? In a way, so is *f* or g. We're just saying that there is a function and we're calling it something, like *f* or g. If we wanted to we could call our function anything we wanted (like *h*, or *happiness,* or *mathIsAwesome*). What matters is the format that I'm writing in. Specifically, it matters that I'm writing in the form:

functionName(*argument*) = stuff_done_to_*argument*,

where *functionName* is the name of the function I'm defining (what we've been using f (or g) for until this point), *argument* is the variable that stands in for whatever we're taking as input to the function, and stuff_done_to_*argument* stands for some operation or set of operations that we're performing on the argument (e.g., we could add 2 to the argument, or divide the argument by 7, or do nothing to the argument, like we did in the identity function), written in the form of an equation. What's important is that maths has defined a way to specify a function, and that you can read things in that format, and interpret them correctly.

So, for instance, what would g from Eq. 8.2b produce for the following inputs?

1?
2?
147?
1948?

Answers: [3, 4, 149, 1950].

The values should have been pretty straightforward to compute. For the input '1', the function computes '1+2', or 3; for the input '2', the function computes '2+2', or 4; and so on.

Great. What if I input 'Panda' to *f* in Eq. 8.2? That is, what is the value for *f*(Panda)? Well, depending on how far you want to take the example, it's either 'Panda+2' (just add 2 to 'Panda'), or "that makes no sense". If your answer was 'Panda+2', that is great. If your answer was "that makes no sense", and you gave this answer because 'Panda+2' can't be computed (i.e., there is no number that I can get by adding 2 to 'Panda'), that's great too. What you've just implicitly discovered is the idea of the domain and the range of a function. The **domain** is all values that can be input into the function, and the **range** is all the values produced by the elements in the domain. For example, if you take the values [1,2,3,4,5] as the domain of the function f in Eq. 8.2., then the range is [3,4,5,6,7], or all the values produced by putting the values of the domain into f (notice that the values in the range correspond to each of the values in the domain with 2 added to it.) If you said that the output of the function in Eq. 8.2 to the input 'Panda' was 'Panda+2', then you defined the domain of the function as very broad (give me anything, and I'll give it back to you as that thing '+2'). If you said "that doesn't make any sense", then you defined the domain of the function as more restricted—perhaps as natural numbers (or items that give a specific quantity when 2 is added to them).

Which is right? Either one might be. I wasn't specific about the domain of the function, f, so you're free to interpret the domain however you please. Being very precise about your functions and what values they can take is a key idea in statistics, writing computer programs that work, and understanding human thinking. As long as you've got the idea that I can define a function, and I can specify the domain that I want that function to work over, you've got the main point. (If you're interested in this idea of functions and their domains, feel free to come chat with me.)

Figure 8.1 gives a summary of the basic ideas of functions, and more and less formal and specific ways to think about them. At the top of Figure 8.1 you have the least formal way of expressing functions: You input something, something happens, and you get an output. This basic idea underlies the more formal notion of a function, which is depicted in the second row of Figure 8.1: You have an argument that you pass to a function, which performs some operation on that argument, and produces an output. On the third row of Figure 8.1 you have an example of a specific function (add 5 to input, can you write this function out in $f(x)$ form?[2]). Finally, on the bottom row of Figure 8.1 you have a specific example of the function in third row. A list of numbers is passed to the function, and an output (that list of number with 5 added to each number) is returned as output.

[2] The function add 5 to an input to produce an output can be written as: $f(x) = x + 5$.

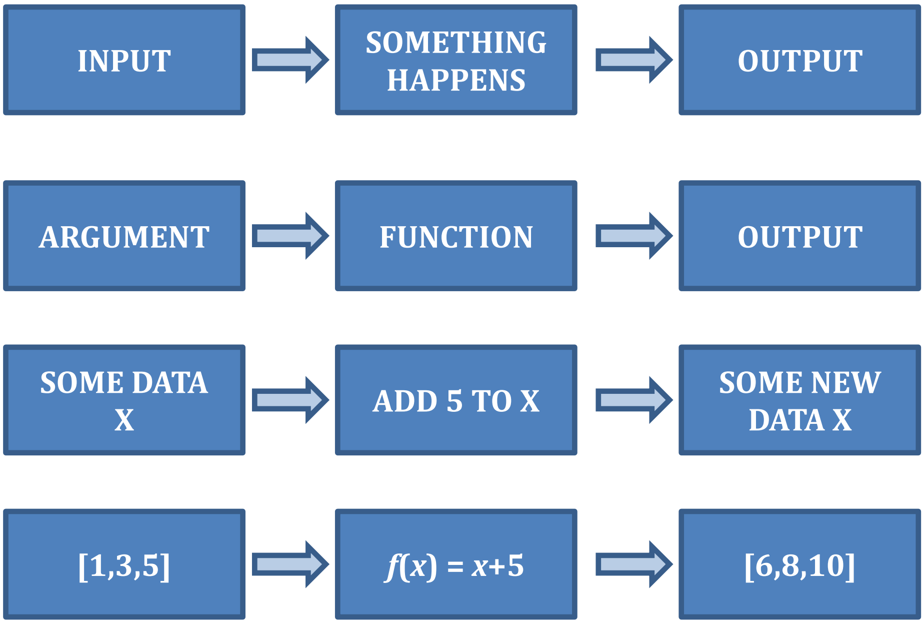

Figure 8.1. Some graphical depictions of functions.

Functions with more than one argument

So far we've only looked at a couple of functions, and these functions took a single argument. That is, the function took a single value as input and output a single value. There is no 'in principle' reason that a function needs to take only one argument, though. In fact, we can easily think of several functions that take multiple arguments. Let's look at a very simple one:

$$f(x, y) = x + y \tag{8.3}$$

which says… (try to fill it in).

Ok, you probably said something like: Eq. 8.3 can be read as, f of x and y equals $x + y$. In other words, $f(x, y)$ produces the sum of x and y. That's reasonably straightforward, right?

Ok, how about this one:

$$g(x, y) = x - y \tag{8.4}$$

which reads, g of x and y equals $x - y$. In other words, $g(x,y)$ produces the difference of x and y.

We could even go a bit crazy if we wanted:

$$f(x, y, z, a, b, c) = \frac{\sqrt[y]{x} - a^z}{bc} \tag{8.5}$$

Ugh. I know, right? Not that it really matters, but this function basically says take the yth root of x, subtract a to the zth power, and divide that total by the product of b and c. Now, you might not want to actually calculate the value associated with a set of arguments, 2,5,14,7,12,42 (in fact, if you want to do this calculation, wow!), but you should be able to put the appropriate numbers in the appropriate places so that you could work it out if you had a scientific calculator and really wanted to.

Let's practice.

Write out the following functions in words:

1) $f(x) = 3x+3$
2) $f(x) = x/2$
3) $f(x,y) = x/y$
4) $f(x, y) = x^y$
5) $f(x, y, z) = 2x + y/5z$

Using R

Before we go on it's important to understand the differences and similarities of mathematical functions and functions in a computer programming language. Functions in a programming language are a specific kind of mathematical function. A function in programming is a block of code that does something to some set of arguments, and then gives an output. Just like a mathematical function, you have an input, stuff happening to that input, then an output. Functions are useful in programming languages because it allows you to reuse code. For example, if you found yourself constantly adding up numbers and dividing by their total to get a mean, you might write yourself a block of code that took a set of numbers and output the mean, call that function *mean*, and then not have to write as much code in the future. You've already seen a number of the built I functions in R (including mean()). You could also write your own if you wanted, but we'll save that for later.

Now, let's use R to help us compute some mathematical functions more easily. Let's say we have a function like: $f(x) = 3x+3$. Let's say we want to compute that function for the range of values 1 to 5.

BEGIN CODE BOX 8.1
```
> x = seq(5)
> y = 3*x + 3
> y
[1]  6  9 12 15 18
```
END CODE BOX 8.1

Use R to give the range of values that you get from applying the following domains to each of the following functions:

1) $f(x) = 3x+3$ for x = [1,2,3,4,5,6,7,8,9,10]
2) $f(x) = x/2$ for x = [1,2,3,4,5]
3) $f(x,y) = x/y$ for x = [1,2,3,4,5] and y = [1,2,3,4,5]
4) $f(x, y) = x^y$ for x = [1,2,3,4,5] and y = [1,2,3,4,5]
5) $f(x, y, z) = 2x + y/5z$ for x = [1,2,3,4,5], y = [1,2,3,4,5], and z = [1,2,3,4,5]

Notice the problem when you used R with multiple arguments? R just gave you back the outcomes for the first element of every input passed to the function, then the second element of every input passed to the function, and so on. What if you wanted to know all the possible outcomes (e.g., you wanted to know what the value of (4) is for x = 3 and y = 5? You could do it by hand, but that's tedious. That's where writing your own functions comes in handy. For now doing that isn't so important, as R gives us most of the functions we need, but the onlne supplement covers writing simple functions.

Functions and graphs

Ok, so that's (at least the very very very beginning of) functions. Now let's look at one of the most useful ways to think about functions: as graphs! Basically, you can depict a function's behaviour in a graph (i.e., you could make a graph that shows what a function does), and you can describe a graph with a function (i.e., you could look at a graph and write a function that describes the entire thing), and doing one or the other is very often very useful in statistics (in fact, we've been doing it a lot). Let's start by thinking about using graphs to represent what functions are doing.

Representing a functions with a graph

You've already seen several examples of functions plotted graphically in your academic careers, but we're going to start with baby-steps just in case any of it is fuzzy. Consider the following function

$$f(x) = 3x + 2 \tag{8.6}$$

which reads, *f* of *x* equals 3**x* plus 2. That's pretty straight-forward; for any value of *x*, just multiply it by 3, and then add 2. Not bad. Try to work out the value of *f*(*x*) for the domain [0,1,2,3,4]. The values output by the function in Eq. 8.6 for the given domain are presented in Table 1.

Table 8.1.

Input number	Function output
0	2
1	5
2	8
3	11
4	14

We know from what we've discussed above that a function assigns an output for a given input. That is, with a function, you put something in as input, and the function gives you a specific thing back as output. The simplest kind of functions we've seen is one where you have one input, and you get one output back. The function in Eq. 8.6 is this kind of function: You put in a single input number—the specific value of *x*—and you get back a specific output number—the specific output associated with that value of *x*. So, when you put in 0 as input, you get 2 as output, and when you put in 1 as input, you get 3 as output, and so on.

In Table 1, you've got a list of input numbers, and a corresponding output number for each input. Basically, the table lists the input-output relationships for a set of numbers for the function in Eq. 8.6. You could translate the information in this table (i.e., the information produced by your function) into a two-dimensional graph (a graph with 2 axes) pretty easily: You take the inputs and represent them on the x-axis and take the outputs and represent them on the y-axis. Let's do just that!

Let's start with the first row of values in Table 1: When the input=0, the output=2. We said we'd represent the input on the x-axis and output on the y-axis, so that means when x=input=0, y=output=2. Let's plot that point on a graph: See Figure 8.2.

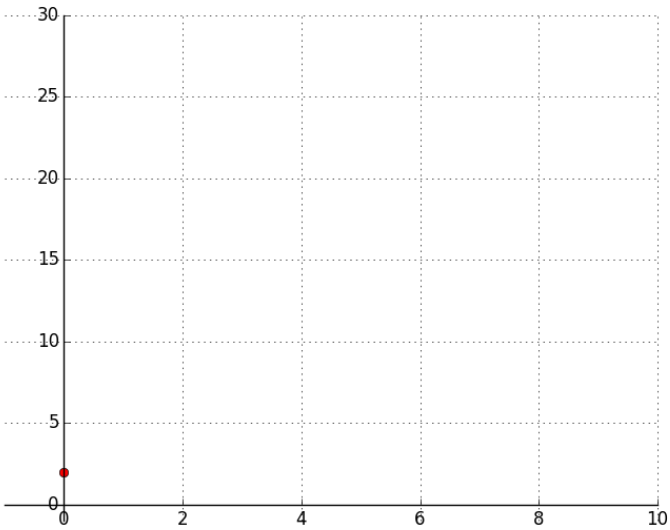

Figure 8.2. A graph with the point (0,2) (i.e., x=0, y=2) represented.

Let's continue and add a point for the second row of values in Table 2: x=input=1, and y=output=5. Let's plot that point on our graph: See Figure 8.3.

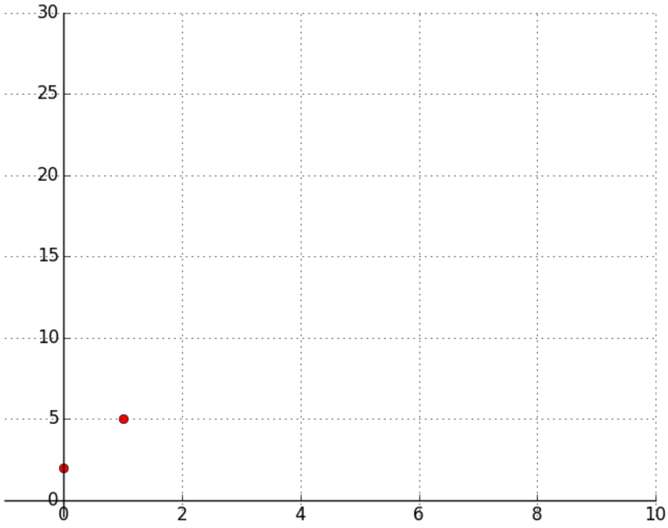

Figure 8.3. A graph with the points (0,2) and (1,5) represented.

And we could do that for the rest of the points in Table 1, as well. We can see that the inputs when graphed on the x-axis give a corresponding value for the y-axis, and we can put a point on our graph that specifies the value on y for each value on x: See Figure 8.4. Basically, the graph is capturing the input output relationship of the function from Eq. 8.6 visually.

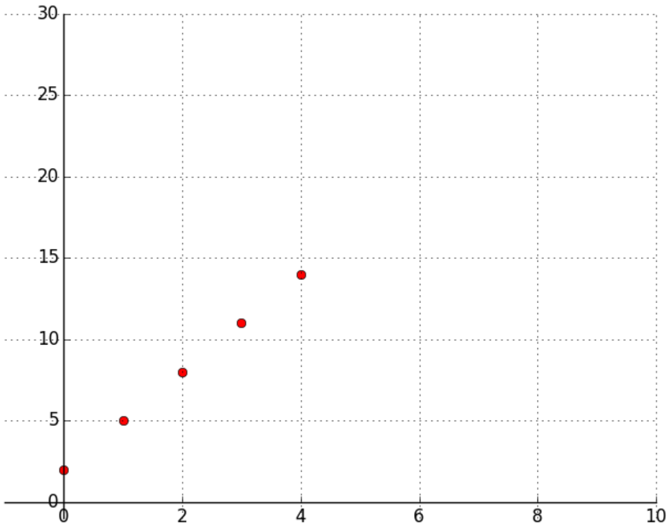

Figure 8.4. A graph with the points given in Table 1 all represented.

Looking at Figure 8.4 it's pretty clear that there's a very simple relationship between the value on the x-axis and the y-axis: Specifically, a linear (or straight-line) relationship (you might notice the similarity of the equation in our function—$f(x)$=3x+2—to the equation for a line—$y=mx+b$[3]). It should also be

[3] If you didn't know the equation for a line, don't worry. Now you know the equation to make a line is y=mx+b, where m is the slope of the line, and by is the y-intercept. If you

pretty clear at this point that the relationship depicted in the graph between values on x and values on y exactly mirrors the relationship in the function between inputs and outputs. (Not that weird given that they are exactly the same, but still a little bit cool!) We could draw a line that connects all the points in Figure 8.4 and extends beyond those points in either direction: See Figure 8.5. Using that line we can visualise the way our function behaves (as input values increase, output values increase in a linear fashion), and we can find the value that corresponds to any given input value. If, for instance, we wanted to know the output value when x=8, find the y value for the point on the line where the x-axis value is 8. Go ahead. I'll wait… hopefully you got something like 26.

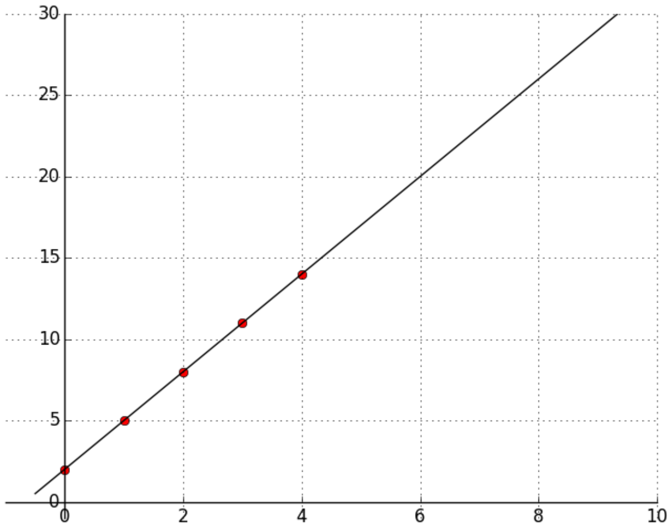

Figure 8.5. A graph depicting the line specified by the function in Eq. 8.6.

We could plot any number of functions with graphs. Here's a graph for the function, $f(x) = x^2$, or *f* of *x* equals *x* squared:

want to play around with this equation, try graphing the y value produced by the x values [1,2,3,4,5] in the equation y = 2x + 1. You should notice that each change in x produces a 2 unit change in y (e.g., when x goes from 1 to 2 (changes by one unit), y goes from 3 to 5 (changes by two units; when x goes from 2 to 3 (changes by one unit), y goes from 5 to 7 (changes by two units)), or the slope of the line is 2, and the line crosses the y axis at the value of 1, or the y-intercept is one.

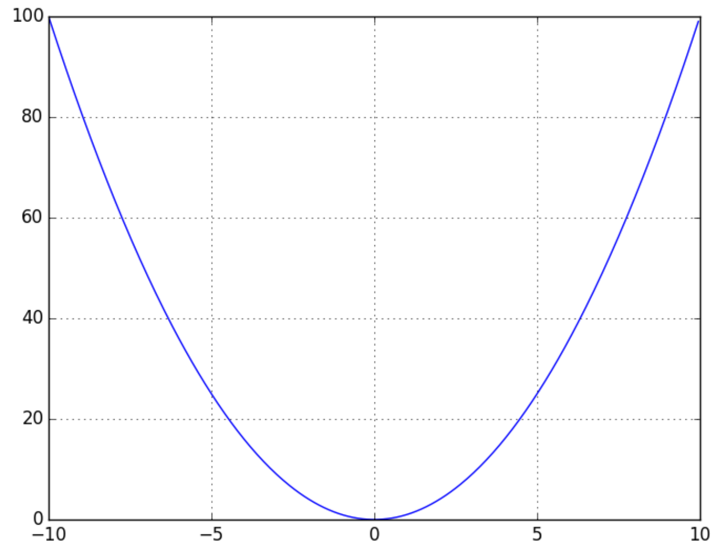

Figure 8.6. Graph of the function $f(x) = x^2$, or *f* of *x* equals *x* squared.

And here's a graph for the function, $f(x) = \log_2(x)$, or *f* of *x* equals the log-base-2 of *x*:[4]

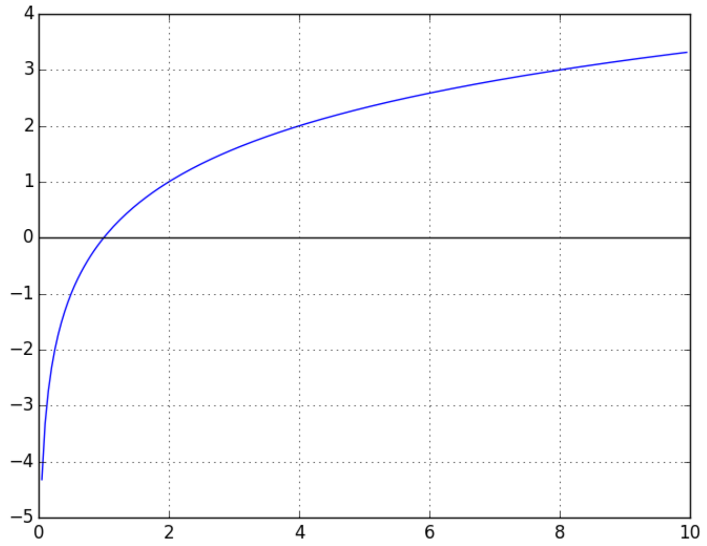

Figure 8.7. Graph of the function $f(x) = \log(x)$, or *f* of *x* equals log of *x*.

Now, here are a few problems were you can work on graphing functions for yourself.

Graphing functions with 1 argument.

[4] A log, or logarithm, is the inverse of exponentiation in mathematics. That is, the log of *x* is the number that some base number must be raised to get *x*. For example, $\log_2(x)$ is the power to which you must raise 2, to get *x*, so specifically, the $\log_2(8)$ is the number power to which you must raise 2 to get 8 ($10^3 = 8$, so $\log_{10}(8) = 3$).

Practice drawing graphs for the following functions:

1) $f(x) = 3x+3$
2) $f(x) = x/2$
3) $f(x) = x*2$
4) $f(x) = 2^x$

Graphs of functions with more than 1 argument

You've likely noticed that we've only been using functions that take a single argument and graphing those. So you ask: "What about functions with more than 1 argument?" That question is certainly fair. The answer is: We could graph the output associated with any number of inputs, it's just that it gets hard to represent those graphs on a 2-dimensional screen very quickly. For example, let's say you had a function like:

$$f(x,z) = x * z \tag{7}$$

which reads, *f* of *x* and *z* equals *x* multiplied by *z*. We could work out what the output is for a domain of values, like, x=[0,1,2] and z=[0,1,2], and we could put them in a table like Table 2.

Table 8.2.

Input_x	Input_z	Output
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	2
2	0	0
2	1	2
2	2	4

Graphing the results in Table 8.2 is as simple as adding another axis to our graph. Now instead of having a 2-dimensional graph (i.e., a graph with 2 axes), we have a 3-dimensional graph (i.e., a graph with 3 axes). In our previous examples we put our input on the x-axis, and our output on the y-axis. Now we put our first input (input_x) on the x-axis, our second input (input_z) on the z-axis, and our output on the y-axis. So, if we wanted to plot, for instance, the 6th row of Table 2, we have input_x=x=1, input_z=z=2, and output=y=2, and so we put a point on the graph were x=1, z=2, and y=2. Figure 8.8 shows a 3-d graph of the plane given by the function from Eq. 8.7.

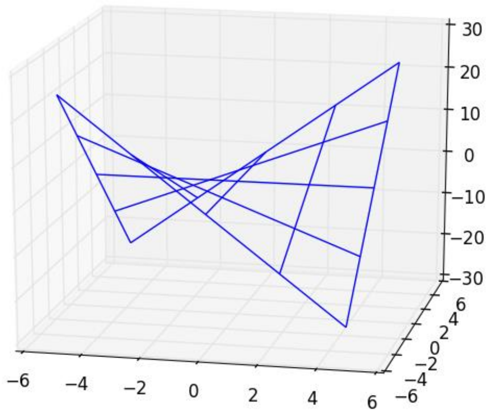

Figure 8.8. A 3D representation of the plane produced by the function in Eq. 8.7. (Notice that 3D plots are a bit hard to depict on a 2D medium (like a page or a screen), so the figure might look a bit confusing if you're not used to looking at 3D plots. Don't worry if the figure above isn't the most clear. The purpose of the figure is just to introduce the idea that you can extend the concept of graphing a function into more than just a single argument.

Sum up, review, and extensions

Alright, we've seen that we can graph the input-output relationship of a function, and that graph can help us understand that function. There should be a couple of things that are reasonably clear at this point:

1. Functions map inputs to outputs.
2. Functions can have a single input, or many inputs (and work the same way regardless of the number of inputs).
3. Graphs can represent relationships.
4. Graphs can represent the input-output relationships specified by a function.
5. So, a function and a graph can represent the exact same information, and:
    a. Any function can be represented with some graph.
    b. Any graph can be represented with some function.

It should be pretty easy to appreciate that just as we can depict a function in a graph, we can also take a graph and write the function that produced it. Take a look at the graph in Figure 8.9.

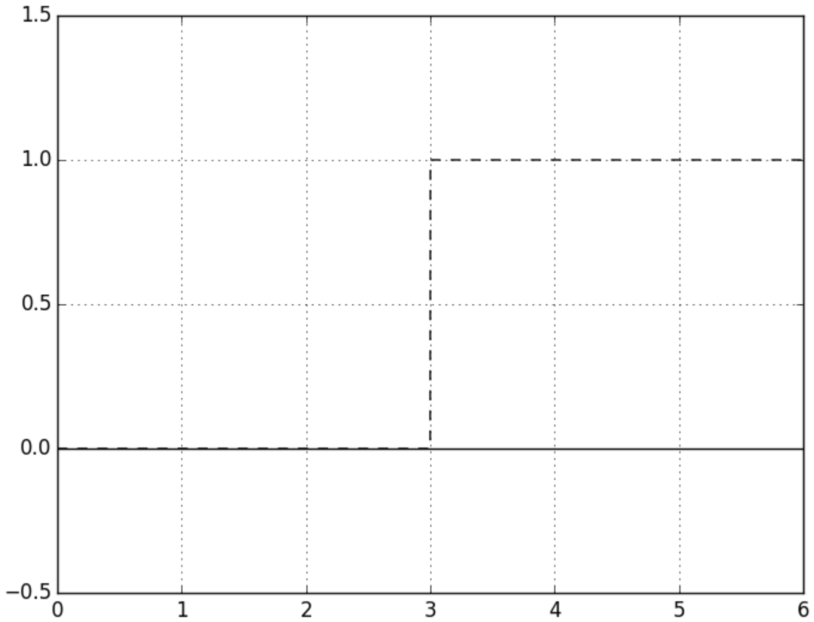

Figure 8.9. A graph representing the function: if x<3, y=0, else y=1. The function is graphed with a dashed line in order to make it distinct.

What does the graph in Figure 8.9 show? Can you describe the function that it depicts? It's pretty simple, right? For any x values less than 3, y=0, then for x values of 3 or greater, y jumps to 1. We could express the same information that is in this graph with a function equation like:

$$f(x) = \begin{cases} 1, & x \geq 3 \\ 0, & otherwise \end{cases} \tag{8.8}$$

which reads, *f* of *x* equals 1 when x is greater than or equal to 3, and 0 otherwise. So, Eq. 8.8 is saying the same thing we said in words above, and that is depicted graphically in Figure 8.9. We've got a lot of tools for expressing some powerful ideas, and it's useful to be able to switch between them as need arises, or as information is presented. Authors might choose to depict an idea with an equation, or with a graph, and you might want to note down an idea that you've been discussing with a colleague using words in a functional form so that you could graph it on your computer and see if it behaves the way you think it should. Being able to understand the same information presented in all these ways will help make you more fluent at reading scientific literature and will also give you some mental jargon you can use to express your own ideas more succinctly and give you more mental space to consider those ideas in more detail (which is part of why scientists started using jargon in the first place).

A good exercise for you now would be to look at some graphs and try to figure out, roughly, what function produced those graphs, and, when you see a function to think about what the plot of that function might look like.

Models in action

As noted above, there are numerous examples of models in psychology. For example, the Rescorla-Wagner model of associative learning (or the deltaV model; Rescorla & Wagner, 1972) provides a very accurate account of associative learning in human and non-human animals by characterising the change in the associative strength of two stimuli as a function of the salience of the two stimluli, the learnability of the relationship, and the pre-existing associations of those stimuli.

In statistics we develop mathematical models of the world, and then use these models to reason about the data we collect. We're going to come back to this series of steps frequently in this text, but it's worth going over it here just as an introduction.

At its core, statistical reasoning takes the following course: First, develop a characterisation of the world. For example, you might think something like, studying makes you do better on an exam. Second, you formalise this characterisation by building some kind of mathematical model. We're going to introduce a bunch of models and how they are used in the following chapters, but for now, let's just be a little creative. Maybe our thinking goes something along the lines of: People get better at exams the more they study, but the effect tops out somewhere (as the best that can be scored on an exam is 100%. So maybe we think something like, revising makes you better, but there's a certain amount of revision required before scores improve that much, then more revision has a very positive effect on performance, and finally, you get to a point where you really understand the material, and more studying won't really improve performance by that much. Perhaps like the line in Figure 8.10. The first few hours of studying produce only minor amounts of improvement, but performance then starts to improve rapidly with more studying, before finally topping out.

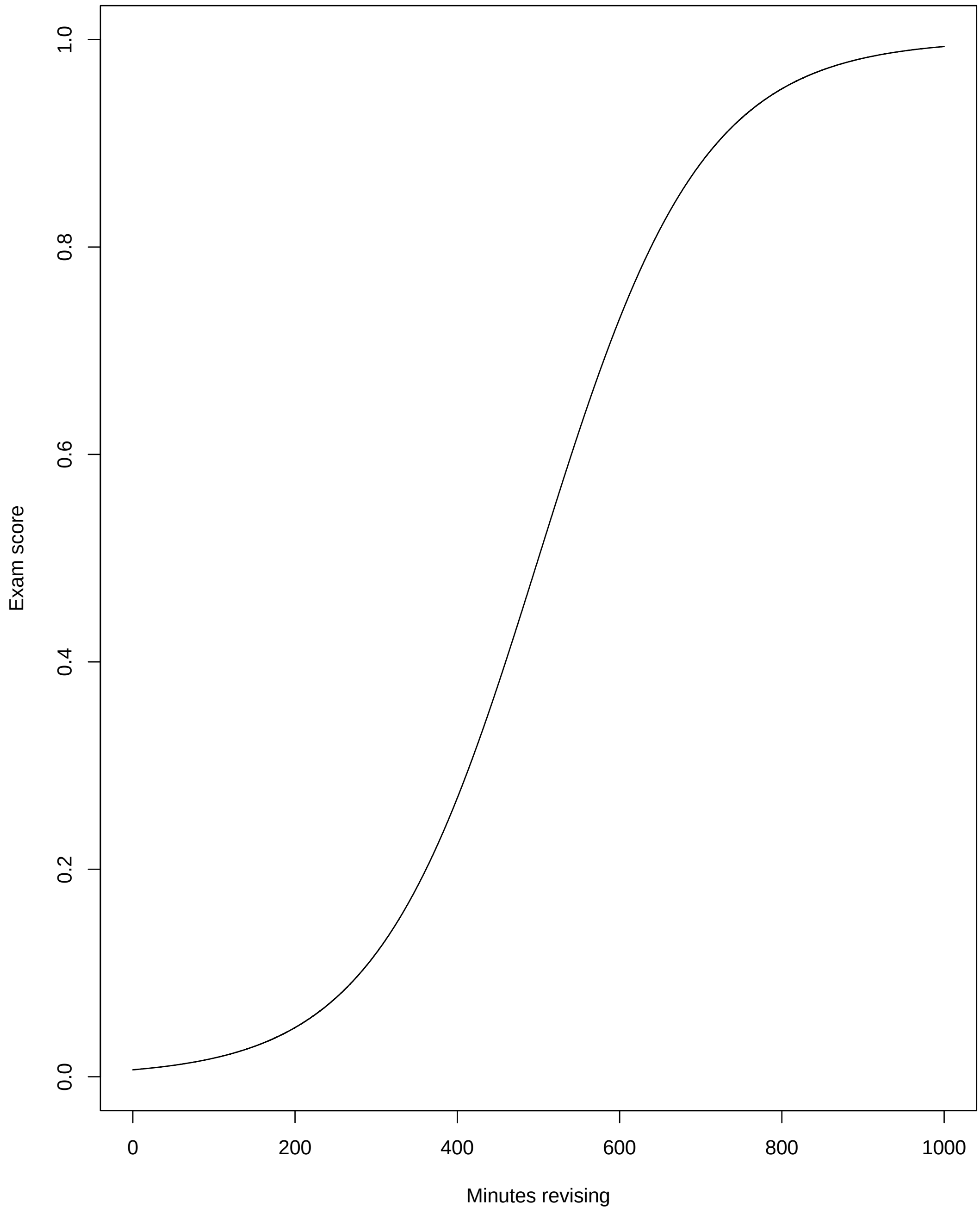

Figure 8.10. A hypothetical model of exam performance as a function of time spent revision.

Third, we would collect observations from the world. Perhaps we might run an experiment, where participants revised for different amounts of time, and their performance on an exam was measured. Fourth, we would evaluate the model in light of the data we collected. Does the data fit with our model? Because of things like measurement error it's unlikely to fit exactly, but what kinds of variation from our model are expected, and what kinds of variation would force us to rethink our specific model or even our conception of the world? In the following chapters we're going to cover methods for addressing these questions. As will find, statistics gives us a very powerful set of tools for providing answers, but the tools are probabilistic—that is, they don't give us definite answers, but rather probabilities of specific answers. As such, it's still up to use, the users of statistical methods to interpret them properly. As we noted earlier in the text, we're hoping to give you the tools understand the various common methods in statistics for psychology and the behavioural sciences. In the next few chapters we're going to cover more foundational concepts for statistics. First, we'll go over basic probability (the mathematical currency of statistics), and then we'll combine the points on functions and graphs from this chapter with the tools of probability to introduce probabilistic models called distributions.