In this post, walk through the code that used to make a nice diagram illustrating  
the parameters in a logistic growth curve. It has been made this figure for a conference  
submission. There was a tight word limit (600 words) and a complicated  
statistical method (Bayesian nonlinear mixed effects beta regression), so   
to use a diagram to carry some of the expository load. Also, figures  
didn’t count towards the word limit, so that was a bonus .

Here will cover a few different topics:

* The pieces of the three-parameter logistic curve
* What the murky “scale” parameter does in the curve
* How to use plotmath to add mathematical copy to a plot

**Growth towards a ceiling**

Children can be hard to understand; they are learning to talk after all. You  
probably can imagine a four-year old asking politely asking for something:  
“pwetty pwease”. This understandability problem is compounded for children with  
cerebral palsy, because these kids will often have speech-motor impairments on  
top of the usual developmental patterns. My current project is a statistical  
model of how *intelligibility*—the probability that an unfamiliar listener  
understands what a child says—develops from age 3 to age 8 in children with  
cerebral palsy.

As an example, the R code plots some (simulated) data that represents a single  
child. They visited our lab 6 times, so we have intelligibility measures for  
each of those visits.

library(tidyverse)

#> -- Attaching packages ----------------------------- tidyverse 1.2.1 --

#> √ ggplot2 3.1.0 √ readr 1.3.1

#> √ tibble 2.0.1 √ purrr 0.3.0

#> √ tidyr 0.8.2 √ stringr 1.4.0

#> √ ggplot2 3.1.0 √ forcats 0.3.0

#> -- Conflicts -------------------------------- tidyverse\_conflicts() --

#> x dplyr::filter() masks stats::filter()

#> x dplyr::lag() masks stats::lag()

theme\_set(theme\_minimal())

points <- tibble(

age = c(38, 45, 52, 61, 80, 74),

prop = c(0.146, 0.241, 0.571, 0.745, 0.843, 0.738))

colors <- list(

data = "#41414550",

# data = "grey80",

fit = "#414145")

ggplot(points) +

aes(x = age, y = prop) +

geom\_point(size = 3.5, color = colors$data) +

scale\_x\_continuous(

name = "Age in months",

limits = c(0, 96),

# Because age is in months, I want breaks to land on multiples

# of 12. The `Q` in `extended\_breaks()` are "nice" numbers to use

# for axis breaks.

breaks = scales::extended\_breaks(Q = c(24, 12))) +

scale\_y\_continuous(

name = "Intelligibility",

limits = c(0, NA),

labels = scales::percent\_format(accuracy = 1))

Simulated intelligibility data.

One of the interesting features of speech development is that it finishes:  
Children stop making their usual developmental speech patterns and converge on a  
mature level of performance. They will, no doubt, continue grow and change  
through adolescence, but when it comes to making speech sounds accurately and  
reliably, most of the developmental change is done by age 8.

For the statistical models, therefore, we expected children to follow a certain  
developmental trajectory towards a ceiling: Begin at zero intelligibility,  
show a period of accelerating then decelerating growth, and finally plateau at  
some mature level of ability. This pattern of growth can be modelled using a  
logistic growth curve using three parameters: an asymptote, a midpoint when  
growth is steepest, and a scale which sets the slope of the curve. Below is the  
equation of the logistic growth curve:

But this equation doesn’t do us any good. If you are like me, you probably  
stopped paying attention when you saw exp() in the denominator. Here’s the  
logistic curve plotted for these data.

xs <- seq(0, 96, length.out = 80)

# Create the curve from the equation parameters

trend <- tibble(

age = xs,

asymptote = .8,

scale = .2,

midpoint = 48,

prop = asymptote / (1 + exp((midpoint - age) \* scale)))

ggplot(points) +

aes(x = age, y = prop) +

geom\_line(data = trend, color = colors$fit) +

geom\_point(size = 3.5, color = colors$data) +

scale\_x\_continuous(

name = "Age in months",

limits = c(0, 96),

breaks = scales::extended\_breaks(Q = c(24, 12))) +

scale\_y\_continuous(

name = "Intelligibility",

limits = c(0, NA),

labels = scales::percent\_format(accuracy = 1))

Data with logistic curve added. It asymptotes at 80%.

Now, let’s add some labels to mark some key parts of the equation. One  
unfamiliar bit of ggplot technology here might be annotate(). Geometry  
functions like geom\_point() or geom\_text() are used to draw data that lives  
in a dataframe using aesthetic mappings defined in aes(), and these function  
draws some geometry (like a point or a label) for each row of the data. But we  
don’t have rows and rows of data to draw. annotate() is meant to handle these  
one-off annotations, and we set the aesthetics manually instead of pulling them  
from some data. The first argument of annotate() says what kind of geom to use  
for the annotation: for example, "text" calls on geom\_text() and "segment"  
calls on geom\_segment().

colors$asym <- "#E7552C"

colors$mid <- "#3B7B9E"

colors$scale <- "#1FA35C"

p <- ggplot(points) +

aes(x = age, y = prop) +

annotate(

"segment",

color = colors$mid,

x = 48, xend = 48,

y = 0, yend = .4,

linetype = "dashed") +

annotate(

"segment",

color = colors$asym,

x = 20, xend = Inf,

y = .8, yend = .8,

linetype = "dashed") +

geom\_line(data = trend, size = 1, color = colors$fit) +

geom\_point(size = 3.5, color = colors$data) +

annotate(

"text",

label = "growth plateaus at asymptote",

x = 20, y = .84,

# horizontal justification = 0 sets x position to left edge of text

hjust = 0,

color = colors$asym) +

annotate(

"text",

label = "growth steepest at midpoint",

x = 49, y = .05,

hjust = 0,

color = colors$mid) +

scale\_x\_continuous(

name = "Age in months",

limits = c(0, 96),

breaks = scales::extended\_breaks(Q = c(24, 12))) +

scale\_y\_continuous(

name = "Intelligibility",

limits = c(0, NA),

labels = scales::percent\_format(accuracy = 1))

p

The figure with the asymptote and midpoint added parameters labelled.

By the way, some other ways to describe the asymptote besides “ceiling” or  
“plateau” would be “saturation” which emphasizes how things only change a small  
amount near the asymptote or as a “limiting” factor or “capacity” which  
emphasizes how growth is no longer tenable after a certain point.

Okay, that just leaves the scale parameter.

**We need to talk about the scale parameter for a second**

In a sentence, the scale parameter controls how steep the curve is. The logistic  
curve is at its steepest at the midpoint. Growth accelerates, hits the midpoint,  
then decelerates. The rate of change on the curve is changing constantly along  
the course of the curve. Therefore, it doesn’t make sense to talk about the  
scale as the growth rate or as the slope in any particular location. It’s better  
to think of it as a growth factor, or umm, scale. I say that it “controls” the  
slope of the curve, because changing the scale will affect the overall stepness  
of the curve.

Here is the derivative of the logistic curve. (I had to ask Wolfram Alpha to do  
the math for me.) This function tells you the rate of change in the curve at any  
point.

Yeah, I don’t like it either, but I have to show you this mess to show how neat  
things are at the midpoint of the curve. When *t* is the midpoint, algebraic  
magic happens . All of the (mid − *t*) parts become 0, exp(0) is 1, so everything  
simplifies a great deal. Check it out.

In our case, with a scale of .2 and asymptote of .8, the slope at 48 months is  
(.2 / 4) \* 8 which is .04. When the curve is at its steepest, for the data  
illustrated here, intelligibility grows at a rate of 4 percentage points per  
month. That’s an upper limit on growth rate: This child never gains more than 4  
percentage points per month.

Now, we can add annotate the plot with an arrow with this slope at the midpoint.  
That seems like a good representation because this point is where the scale is  
most transparently related to the curve’s shape.

# Compute endpoints for segment with given slope in middle

slope <- (.2 / 4) \* .8

x\_step <- 2.5

y1 <- .4 + slope \* -x\_step

y2 <- .4 + slope \* x\_step

p <- p +

geom\_segment(

x = 48 - x\_step, xend = 48 + x\_step,

y = y1, yend = y2,

size = 1.2,

color = colors$scale,

arrow = arrow(ends = "both", length = unit(.1, "in"))) +

annotate(

"text",

label = "scale controls slope of curve",

x = 49, y = .38,

color = colors$scale, hjust = 0)

p

The figure with the asymptote, midpoint and scale added parameters labelled.

**Adding the equation**

For my conference submission, I didn’t want to include the equation in the text.  
It was just too low-level of a detail for the 600-word limit. So I added the  
equation to the plot using plotmath.

I’m not exactly sure what this feature should be called, but ?plotmath is  
what you type to open the help page, so that’s what I call it. You can add math  
to a plot by providing an expression() which is parsed into mathematical copy,  
or by passing a string and setting parse = TRUE. Here is a demo of both  
approaches.

ggplot(tibble(x = 1:3)) +

aes(x = x) +

geom\_text(

aes(y = 1),

label = expression(1 + 100 + pi)) +

geom\_text(

aes(y = .5),

label = "frac(mu, 100)",

parse = TRUE) +

xlim(0, 4) +

ylim(0, 1.1)

#> Warning in is.na(x): is.na() applied to non-(list or vector) of type

#> 'expression'

Demo of plotmath

# (I don't know what this warning is about.)

For this plot, we’re going to create a helper function that pre-sets parse to  
TRUE and pre-sets the location for the equation.

# Helper to plot an equation in a pre-set spot

annotate\_eq <- function(label, ...) {

annotate("text", x = 0, y = .6, label = label, parse = TRUE,

hjust = 0, size = 4, ...)

}

Then we just add the equation to the plot.

p + annotate\_eq(

label = "f(t)==frac(asymptote, 1 + exp((mid-t)%\*%scale))",

color = colors$fit)

Labelled plot from earlier with an equation added to it.

This is a perfectly serviceable plot, but we can get fancier. I gave the  
parameter annotations different colors for a reason .

**Phantom menaces**

Plotmath provides a function called phantom() for adding placeholders to  
an equation. phantom(x) will make space for *x* in the equation but it  
won’t draw it. Therefore, we can phantom() out all of the parameters to draw  
the non-parameter parts of the equation in black.

p1 <- p +

annotate\_eq(

label = "

f(t) == frac(

phantom(asymptote),

1 + exp((phantom(mid) - t) %\*% phantom(scale))

)",

color = colors$fit)

p1

Labelled plot from earlier with an equation added to it, except there are blanks for 'asymptote', 'mid', and 'scale'.

Then we layer on the other parts of the equation in different colors, using  
phantom() as needed so we don’t overwrite the black parts. We also use  
atop(); it does the same thing as frac() except it doesn’t draw a fraction  
line. Here’s the addition of the asymptote.

p2 <- p1 +

annotate\_eq(

label = "

phantom(f(t) == symbol('')) ~ atop(

asymptote,

phantom(1 + exp((mid-t) %\*% scale))

)",

color = colors$asym)

p2

Labelled plot from earlier with an equation added to it, except there are blanks for 'mid', and 'scale'. 'Asymptote' is in equation in color.

But the other parameters are not that simple. The plotmath help page states that  
“A mathematical expression must obey the normal rules of syntax for any R  
expression” so that means that we can’t do something like phantom(1 + ) x"  
because the 1 + is not valid R syntax. So to blank out parts of  
expressions, we create expressions using paste() to put symbols next to each  
other and symbol() to refer to symbols/operators as characters.

I have to be honest, however: it took a lot of fiddling to get this work right.  
Therefore, I have added the following disclaimer: *Don’t study this code.  
Just observe what is possible, but observe all the hacky code required.* 

p2 +

annotate\_eq(

label = "

phantom(f(t) == symbol('')) ~ atop(

phantom(asymptote),

phantom(1 + exp((mid-t) \* symbol(''))) ~ scale

)",

color = colors$scale) +

annotate\_eq(

label = "

phantom(f(t) == symbol('')) ~ atop(

phantom(asymptote),

paste(

phantom(paste(1 + exp, symbol(')'), symbol(')'))),

mid,

phantom(paste(symbol('-'), t, symbol(')') \* scale))

)

)",

color = colors$mid)

Labelled plot from earlier with an equation added to it. All three parameters appear in color in the equation.

There we have it—my wonderful, colorful diagram! Take *that* word count!

By the way, if you know a better way to plot partially colorized math equations  
or how to blank out subexpressions in an easier way, I would love to hear it.