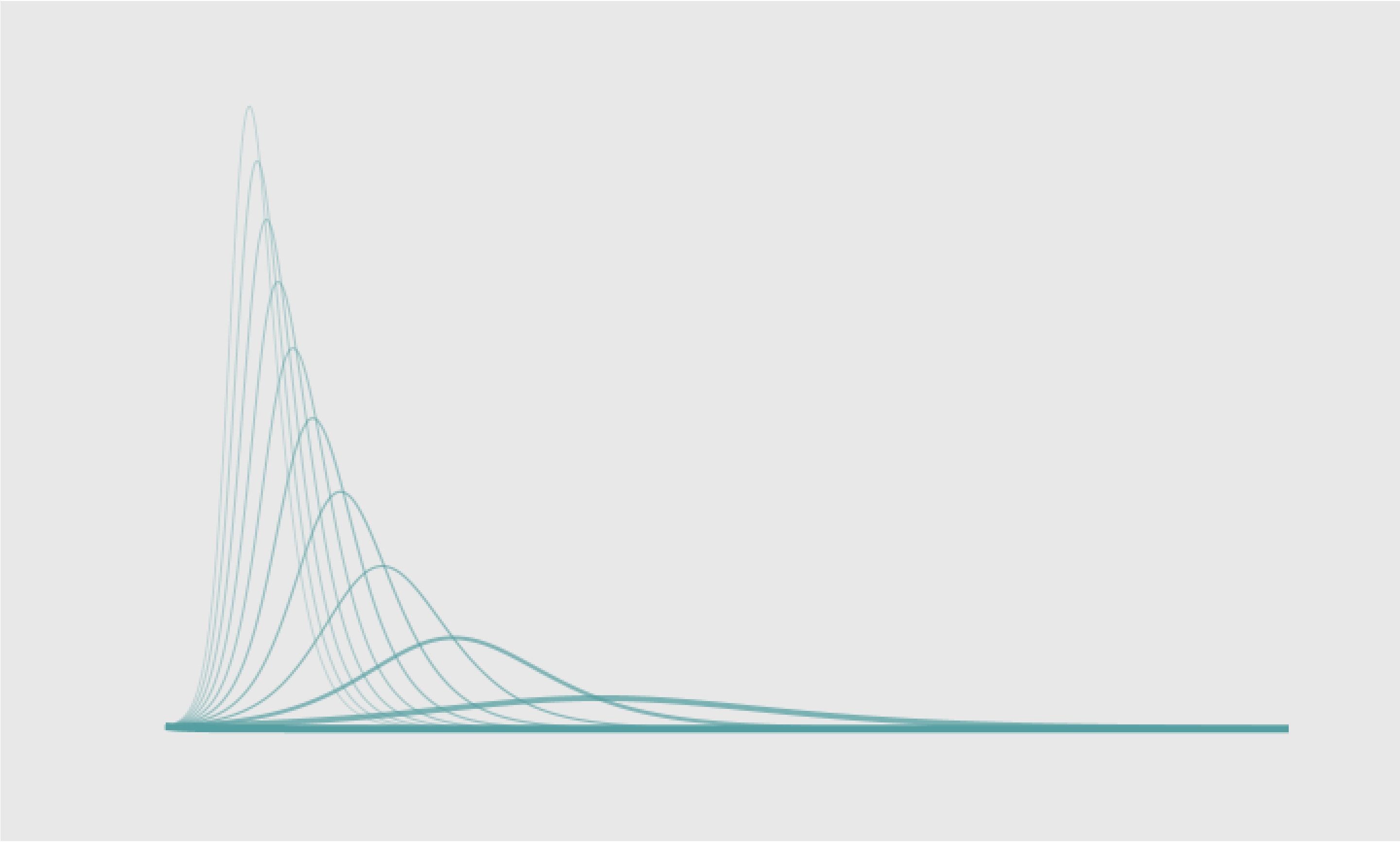
# How to Flatten the Curve, a Social Distancing Simulation and Tutorial

By Nathan Yau

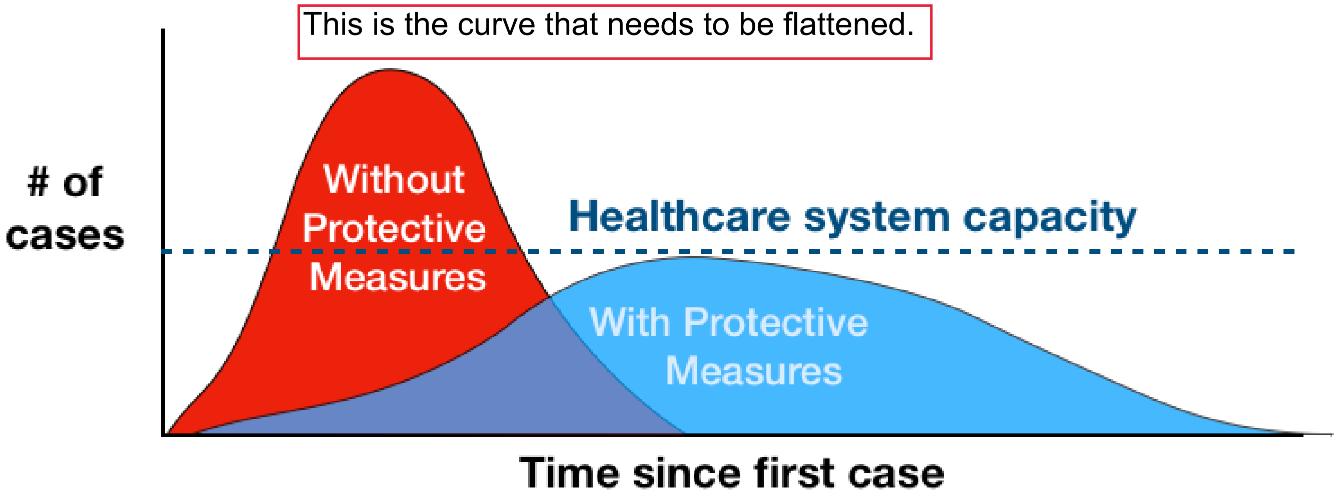
Using R, we look at how your decreased interaction with others can help slow the spread of infectious diseases.



The uncertainty around the coronavirus can make it feel like things are out of our control. We don't quite know the case fatality rate or the infection rate, because there isn't enough reliable data yet, so the numbers swing over a wide range depending on what you read.

However, we — as in the world — are mostly in agreement that we need to "flatten the curve" so that even if the coronavirus infects a lot of people, the rate is low enough so that the health care system doesn't buckle. And hopefully, in the long run, fewer people are infected.

You've probably seen a version or five of the curve by now. This one is by Drew Harris:



*Adapted from CDC / The Economist*

We want there to be enough ventilators when we reach the peak of the curve.

*So how do we flatten the curve?* It began with encouragement to wash our hands (properly) and then a push for social distancing. Where I live we've been ordered to "shelter at home" for the next three weeks. Only leave the house and interact with others for essential activities.

In a land of things we can't control, we can control social distancing almost completely, and research has shown it can be extremely effective in slowing the spread of infectious diseases.

**DISCLAIMER**
**I use a basic mathematical model in this tutorial and these are simulations of an abstract disease.**

Sheltering at home, with little ability to think about much else, I wanted to see for myself. Here's how I did it in R, how you can run simulations yourself, and how you can plot the results over time.

## Setup

I used three R packages:

- **EpiModel** — Mathematical modeling for infection disease dynamics. See more details here.
- **animation** — Used for animated GIFs, etc.
- **extrafont** — Bring in fonts that don’t come bundled with R.

Using a Susceptible-Infectious-Recovered (SIR) model, each person in the simulation can either be susceptible to the disease, infected, or recovered. You can specify the rates of each.

Set the parameters:

```
param <- param.dcm(inf.prob = 0.2, act.rate = 1, rec.rate = 1/20,
                   a.rate = 0, ds.rate = 0, di.rate = 1/80, dr.rate = 0)
```

Here’s what each parameter represents:

- *inf.prob* — Probability of infection between a susceptible and infected person.
- *act.rate* — Average number of transmissible acts, like shaking hands.
- *rec.rate* — Average rate of recovery with immunity.
- *a.rate* — Arrival rate of new susceptible people to the population, like a birth.
- *ds.rate* — Departure rate of susceptible people, like a death not caused by the disease.
- *di.rate* — Departure rate for infected people.
- *dr.rate* — Departure rate for recovered people.

We don’t know many of the rates for sure yet, and for simplicity’s sake, we assume a fixed population where more people are not born or die of other causes (*a.rate*, *ds.rate*, and *dr.rate* are set to 0).

*inf.prob* probably goes down when we wash our hands properly and frequently, but I don’t know by how much or how well people actually wash their hands.

So for now, we focus on *act.rate*. By social distancing, we try to decrease that number. We try to bring it as close to 0 as possible, because if the virus has nowhere to go, it's a lot harder for it to spread.

Set the initial parameters of the population with 1,000 susceptible people, 1 infected, and 0 recovered:

```
init <- init.dcm(s.num = 1000, i.num = 1, r.num = 0)
```

Then set the controls for the "Deterministic Compartmental Model":

```
control <- control.dcm(type = "SIR", nsteps = 500, dt = 0.5)
```

Again we use a SIR model and run the simulation 500 times in 0.5 time unit increments. In this exercise, we're less interested in actual time units and more interested in the changes.

Run the simulation with the specified model parameters:

```
mod <- dcm(param, init, control)
```

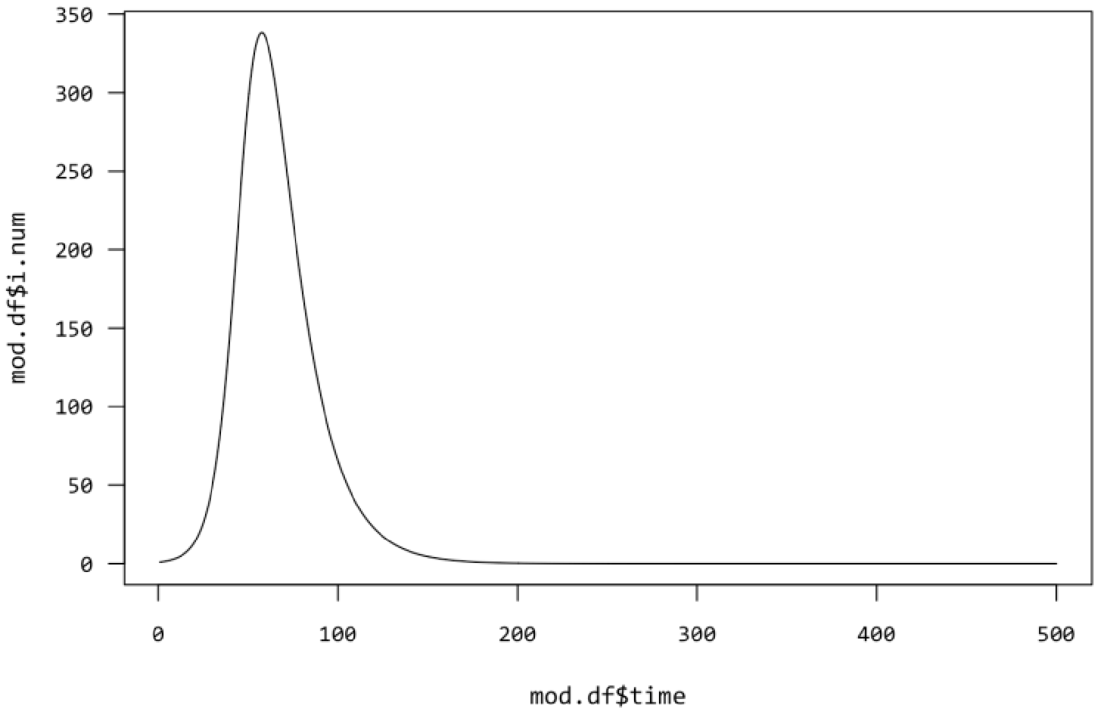
## Plot

The simulation outputs results for several variables, but for our purposes, we're interested in the number infected over time.

```
> mod.df[1:10, c("time", "i.num")]
   time    i.num
1   1.0 1.000000
2   1.5 1.072396
3   2.0 1.150021
4   2.5 1.233252
5   3.0 1.322491
6   3.5 1.418169
7   4.0 1.520750
8   4.5 1.630726
9   5.0 1.748629
10  5.5 1.875025
# Plot it.
```

```
plot(mod.df$time, mod.df$i.num, type="l")
```

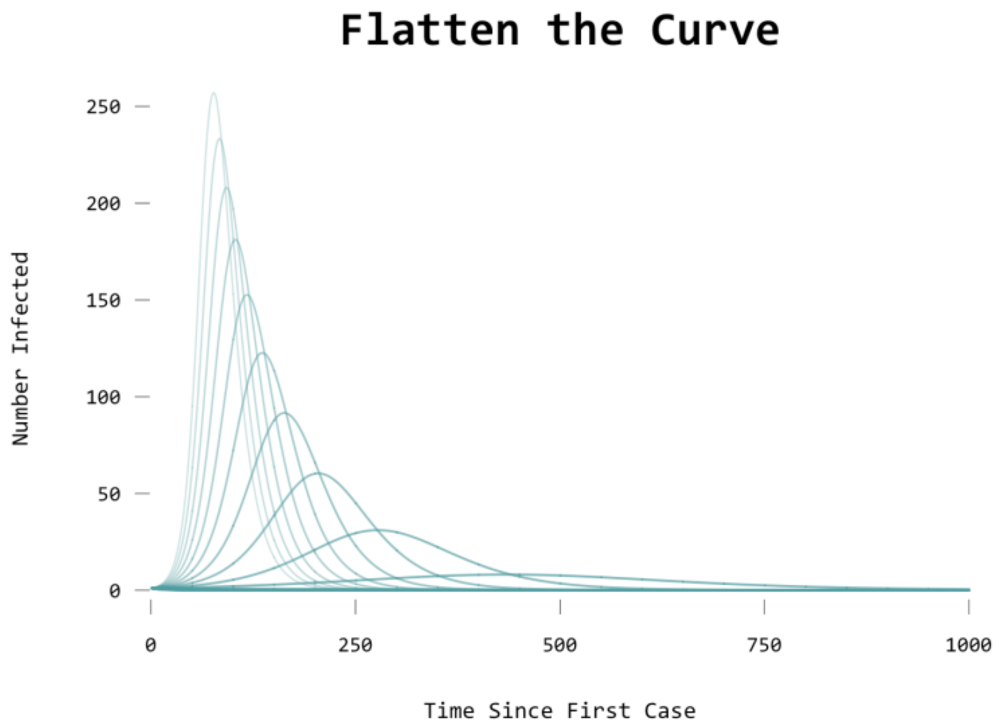
It looks like a curve:



It's time to flatten it by running a simulation for several act.rate values, ranging from 0.8 to 0.

act.rates <- seq(.8, 0, by=-.05)

The fewer acts between people, the flatter the curve, and the closer you get to zero, the fewer chances of the disease finding another host:



Again, we're working with a simplified model for an abstract infectious disease. Real life is more complex and there are many unknowns. But, we do the best we can with the information we have and we take it from there one day at a time.

To summarize:

1. Wash your hands.
2. Keep your distance.

These are the things you can control as an individual. Together, we flatten the curve.

About the Author

Nathan Yau is a statistician who works primarily with visualization. He earned his PhD in statistics from UCLA, is the author of two best-selling books — Data Points and Visualize This — and runs FlowingData. Introvert. Likes food. Likes beer.