Code Review of Ferguson’s Model

6 May 2020. Updated 10 May 2020.

by Sue Denim (not the author’s real name)

Imperial College finally released a derivative of Ferguson’s code. I figured I’d do a review of it and send you some of the things I noticed. I don’t know your background so apologies if some of this is pitched at the wrong level.

My background. I have been writing software for 30 years. I worked at Google between 2006 and 2014, where I was a senior software engineer working on Maps, Gmail and account security. I spent the last five years at a US/UK firm where I designed the company’s database product, amongst other jobs and projects. I was also an independent consultant for a couple of years. Obviously I’m giving only my own professional opinion and not speaking for my current employer.

The code. It isn’t the code Ferguson ran to produce his famous Report 9. What’s been released on GitHub is a heavily modified derivative of it, after having been upgraded for over a month by a team from Microsoft and others. This revised codebase is split into multiple files for legibility and written in C++, whereas the original program was “a single 15,000 line file that had been worked on for a decade” (this is considered extremely poor practice). A request for the original code was made 8 days ago but ignored, and it will probably take some kind of legal compulsion to make them release it. Clearly, Imperial are too embarrassed by the state of it ever to release it of their own free will, which is unacceptable given that it was paid for by the taxpayer and belongs to them.

The model. What it’s doing is best described as “SimCity without the graphics”. It attempts to simulate households, schools, offices, people and their movements, etc. I won’t go further into the underlying assumptions, since that’s well explored elsewhere.

Non-deterministic outputs. Due to bugs, the code can produce very different results given identical inputs. They routinely act as if this is unimportant.

This problem makes the code unusable for scientific purposes, given that a key part of the scientific method is the ability to replicate results. Without replication, the findings might not be real at all – as the field of psychology has been finding out to its cost. Even if their original code was released, it’s apparent that the same numbers as in Report 9 might not come out of it.

Non-deterministic outputs may take some explanation, as it’s not something anyone previously floated as a possibility.

The documentation says:

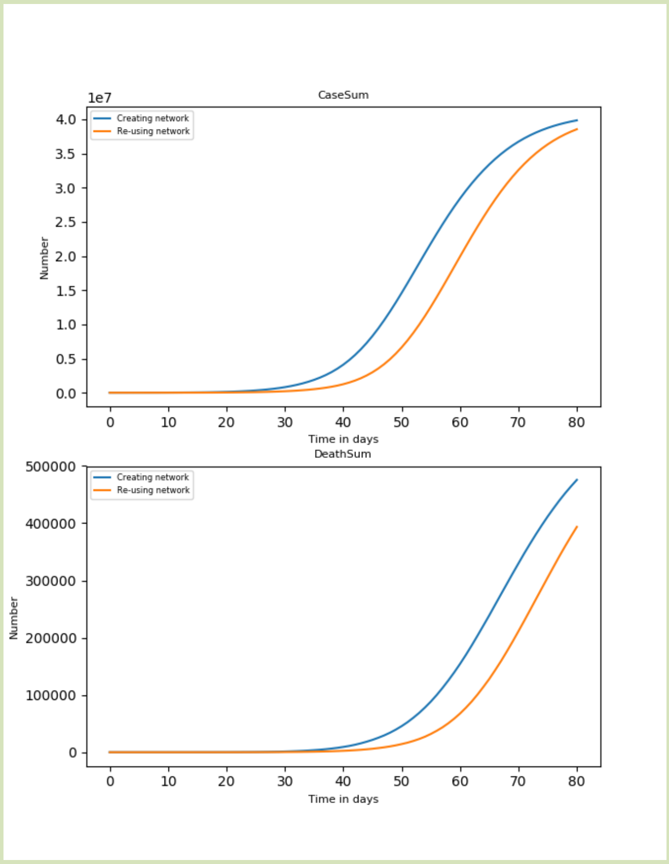
The model is stochastic. Multiple runs with different seeds should be undertaken to see average behaviour.

“Stochastic” is just a scientific-sounding word for “random”. That’s not a problem if the randomness is intentional pseudo-randomness, i.e. the randomness is derived from a starting “seed” which is iterated to produce the random numbers. Such randomness is often used in Monte Carlo techniques. It’s safe because the seed can be recorded and the same (pseudo-)random numbers produced from it in future. Any kid who’s played Minecraft is familiar with pseudo-randomness because Minecraft gives you the seeds it uses to generate the random worlds, so by sharing seeds you can share worlds.

Clearly, the documentation wants us to think that, given a starting seed, the model will always produce the same results.

Investigation reveals the truth: the code produces critically different results, even for identical starting seeds and parameters.

I'll illustrate with a few bugs. In issue 116 a UK "red team" at Edinburgh University reports that they tried to use a mode that stores data tables in a more efficient format for faster loading, and discovered – to their surprise – that the resulting predictions varied by around 80,000 deaths after 80 days:



That mode doesn't change anything about the world being simulated, so this was obviously a bug.

The Imperial team's response is that **it doesn't matter**: they are "aware of some small non-determinisms", but "this has historically been considered acceptable because of the general stochastic nature of the model". Note the phrasing here: Imperial know their code has such bugs, but act as if it's some inherent randomness of the universe, rather than a result of amateur coding. Apparently, in epidemiology, a difference of 80,000 deaths is "a small non-determinism".

Imperial advised Edinburgh that the problem goes away if you run the model in single-threaded mode, like they do. This means they suggest using only a single CPU core rather than the many cores that any video game would successfully use. For a simulation of a country, using only a single CPU core is obviously a dire problem – as far from supercomputing as you can get. Nonetheless, that's how Imperial use the code: they know it breaks when they try to run it faster. It's clear from reading the code that in 2014 Imperial tried to make the code use multiple CPUs to speed it up, but never made it work reliably. This sort of programming is known to be difficult and usually requires senior, experienced engineers to get good results. Results that randomly change from run to run are a common consequence of thread-safety bugs. More colloquially, these are known as "Heisenbugs".

But Edinburgh came back and reported that – even in single-threaded mode – they still see the problem. So Imperial's understanding of the issue is wrong. Finally, Imperial admit there's a bug by referencing a code change they've made that fixes it. The explanation given is "It looks like historically the second pair of seeds had been used at this point, to make the runs identical regardless of how the network was made, but that this had been changed when seed-resetting was implemented". In other words, in the process of changing the model they made it non-replicable and never noticed.

Why didn't they notice? Because their code is so deeply riddled with similar bugs and they struggled so much to fix them that they got into the habit of simply averaging the results of multiple runs to cover it up… and eventually this behaviour became normalised within the team.

In issue #30, someone reports that the model produces different outputs depending on what kind of computer it's run on (regardless of the number of CPUs). Again, the explanation is that although this new problem “will just add to the issues” … “This isn’t a problem running the model in full as it is stochastic anyway”.

Although the academic on those threads isn’t Neil Ferguson, he is well aware that the code is filled with bugs that create random results. In change #107 he authored he comments: “It includes fixes to InitModel to ensure deterministic runs with holidays enabled”. In change #158 he describes the change only as “A lot of small changes, some critical to determinacy”.

Imperial are trying to have their cake and eat it. Reports of random results are dismissed with responses like “that’s not a problem, just run it a lot of times and take the average”, but at the same time, they’re fixing such bugs when they find them. They know their code can’t withstand scrutiny, so they hid it until professionals had a chance to fix it, but the damage from over a decade of amateur hobby programming is so extensive that even Microsoft were unable to make it run right.

No tests. In the discussion of the fix for the first bug, Imperial state the code used to be deterministic in that place but they broke it without noticing when changing the code.

Regressions like that are common when working on a complex piece of software, which is why industrial software-engineering teams write automated regression tests. These are programs that run the program with varying inputs and then check the outputs are what’s expected. Every proposed change is run against every test and if any tests fail, the change may not be made.

The Imperial code doesn’t seem to have working regression tests. They tried, but the extent of the random behaviour in their code left them defeated. On 4th April they said: “However, **we haven’t had the time** to work out a scalable and maintainable way of running the regression test in a way that allows a small amount of variation, but doesn’t let the figures drift over time.”

Beyond the apparently unsalvageable nature of this specific codebase, testing model predictions faces a fundamental problem, in that the authors don’t know what the “correct” answer is until long after the fact, and by then the code has changed again anyway, thus changing the set of bugs in it. So it’s unclear what regression tests really mean for models like this – even if they had some that worked.

Undocumented equations. Much of the code consists of formulas for which no purpose is given. John Carmack (a legendary video-game programmer) surmised that some of the code might have been automatically translated from FORTRAN some years ago.

For example, on line 510 of SetupModel.cpp there is a loop over all the “places” the simulation knows about. This code appears to be trying to calculate R0 for “places”. Hotels are excluded during this pass, without explanation.

This bit of code highlights an issue Caswell Bligh has discussed in your site’s comments: R0 isn’t a real characteristic of the virus. R0 is both an input to *and an output of* these models, and is routinely adjusted for different environments and situations. Models that consume their own outputs as inputs is problem well known to the private sector – it can lead to rapid divergence and incorrect prediction. There’s a discussion of this problem in section 2.2 of the Google paper, “Machine learning: the high interest credit card of technical debt“.

Continuing development. Despite being aware of the severe problems in their code that they “haven’t had time” to fix, the Imperial team continue to add new features; for instance, the model attempts to simulate the impact of digital contact tracing apps.

Adding new features to a codebase with this many quality problems will just compound them and make them worse. If I saw this in a company I was consulting for I’d immediately advise them to halt new feature development until thorough regression testing was in place and code quality had been improved.

Conclusions. All papers based on this code should be retracted immediately. Imperial's modelling efforts should be reset with a new team that isn't under Professor Ferguson, and which has a commitment to replicable results with published code from day one.

On a personal level, I'd go further and suggest that all academic epidemiology be defunded. This sort of work is best done by the insurance sector. Insurers employ modellers and data scientists, but also employ managers whose job is to decide whether a model is accurate enough for real world usage and professional software engineers to ensure model software is properly tested, understandable and so on. Academic efforts don't have these people, and the results speak for themselves.

My identity. Sue Denim isn't a real person (read it out). I've chosen to remain anonymous partly because of the intense fighting that surrounds lockdown, but there's also a deeper reason. This situation has come about due to rampant credentialism and I'm tired of it. As the widespread dismay by programmers demonstrates, if anyone in SAGE or the Government had shown the code to a working software engineer they happened to know, alarm bells would have been rung immediately. Instead, the Government is dominated by academics who apparently felt unable to question anything done by a fellow professor. Meanwhile, average citizens like myself are told we should never question "expertise". Although I've proven my Google employment to Toby, this mentality is damaging and needs to end: please, evaluate the claims I've made for yourself, or ask a programmer you know and trust to evaluate them for you.

www.Zerohedge.com

Second Analysis of Ferguson’s Model

9 May 2020

by Sue Denim (not the author’s real name)

I’d like to provide a followup to my first analysis. Firstly because new information has come to light, and secondly to address a few points of disagreement I noticed in a minority of responses.

The hidden history. Someone realised they could unexpectedly recover parts of the deleted history from GitHub, meaning we now have an audit log of changes dating back to April 1st. This is still not exactly the original code Ferguson ran, but it’s significantly closer.

Sadly it shows that Imperial have been making some false statements.

- ICL staff claimed the released and original code are “*essentially the same functionally*”, which is why they “*do not think it would be particularly helpful to release a second codebase which is functionally the same*”.

  In fact the second change in the restored history is a fix for a critical error in the random number generator. Other changes fix data corruption bugs (another one), algorithmic errors, fixing the fact that someone on the team can’t spell household, and whilst this was taking place other Imperial academics continued to add new features related to contact tracing apps.

  The released code at the end of this process was not merely reorganised but contained fixes for severe bugs that would corrupt the internal state of the calculations. That is very different from “*essentially the same functionally*”.
- The stated justification for deleting the history was to make “*the repository rather easier to download*” because “*the history squash (erase) merged a number of changes we were making with large data files*”. “*We do not think there is much benefit in trawling through our internal commit histories*”.

  The entire repository is less than 100 megabytes. Given they recommend a computer with 20 gigabytes of memory to run the simulation for the UK, the cost of downloading the data files is immaterial. Fetching the additional history only took a few seconds on my home WiFi.

  Even if the files had been large, the tools make it easy to not download history if you don’t want it, to solve this exact problem.

I don’t quite know what to make of this. Originally I thought these claims were a result of the academics not understanding the tools they’re working with, but the Microsoft employees helping them are actually employees of a recently acquired company: GitHub. GitHub is the service they’re using to distribute the source code and files. To defend this I’d have to argue that GitHub employees don’t understand how to use GitHub, which is implausible.

I don’t think anyone involved here has any ill intent, but it seems via a chain of innocent yet compounding errors – likely trying to avoid exactly the kind of peer review they’re now getting – they have ended up making false claims in public about their work.

Effect of the bug fixes. I was curious what effect the hidden bug fixes had on the model output, especially after seeing the change to the pseudo-random number generator constants (which means the prior RNG didn’t work). I ran the latest code in single threaded mode for the baseline scenario a couple of times, to establish that it was producing the same results (*on my machine* *only*), which it did. Then I ran the version from the initial import against the latest data, to control for data changes.

The resulting output tables were radically different to the extent that they appear incomparable, e.g. the older code outputs data for negative days and a different set of columns. Comparing by row count for day 128 (7th May) gave 57,145,154 infected-but-recovered people for the initial code but only 42,436,996 for the latest code, a difference of about 34%.

I wondered if the format of the data files had changed without the program being able to detect that, so then I reran the initial import code with the initial data. This yielded 49,445,121 recoveries – yet another completely different number.

It's clear that the changes made over the past month and a half have radically altered the predictions of the model. It will probably never be possible to replicate the numbers in Report 9.

Political attention. I was glad to see the analysis was read by members of Parliament. In particular, via David Davis MP the work was seen by Steve Baker – one of the few British MPs who has been a working software engineer. Baker's assessment was similar to that of most programmers: "*David Davis is right. As a software engineer, I am appalled. Read this now*". Hopefully at some point the right questions will be asked in Parliament. They should focus on reforming how code is used in academia in general, as the issue is structural incentives rather than a single team. The next paragraph will demonstrate that.

Do the bugs matter? Some people don't seem to understand why these bugs are important (e.g. this computational biology student, or this cosmology lecturer at Queen Mary). A few people have claimed I don't understand models, as if Google has no experience with them.

Imagine you want to explore the effects of some policy, like compulsory mask wearing. You change the code and rerun the model with the same seed as before. The number of projected deaths goes up rather than down. Is that because:

- The simulation is telling you something important?
- You made a coding error?
- The operating system decided to check for updates at some critical moment, changing the thread scheduling, the consequent ordering of floating point additions and thus changing the results?

You have absolutely no idea what happened.

In a correctly written model this situation can't occur. A change in the outputs means something real and can be investigated. It's either intentional or a bug. Once you're satisfied you can explain the changes, you can then run the simulation more times with new seeds to estimate some uncertainty intervals.

In an uncontrollable model like ICL's you can't get repeatable results and if the expected size of the change is less than the arbitrary variations, you can't conclude anything from the model. And exactly because the variations are arbitrary, you don't actually know how large they can get, which means there's no way to conclude anything at all.

I ran the simulation three times with the code as of commit 030c350, with the default parameters, fixed seeds and configuration. A correct program would have yielded three identical outputs. For May 7th the max difference of the three runs was 46,266 deaths or around 1.5x the actual UK total so far. This level of variance may look "small" when compared to the enormous overall projections (which it seems are incorrect) but imagine trying to use these values for policymaking. The Nightingale hospitals added on the order of 10-15,000 places, so the uncontrolled differences due to bugs are larger than the NHS's entire crash expansion programme. How can any government use this to test policy?

An average of wrong is wrong. There appears to be a seriously concerning issue with how British universities are teaching programming to scientists. Some of them seem to think hardware-triggered variations don't matter if you average the outputs (they apparently call this an "ensemble model").

Averaging samples to eliminate random noise works only if the noise is actually random. The mishmash of iteratively accumulated floating point uncertainty, uninitialised reads, broken shuffles, broken random number generators and other issues in this model may yield *unexpected* output changes but they *are not truly random deviations, so* they can't just be averaged out. Taking the average of a lot of faulty measurements doesn't give a correct measurement. And though it would be convenient for the computer industry if it were true, you can't fix data corruption by averaging.

I'd recommend all scientists writing code in C/C++ read this training material from Intel. It explains how code that works with fractional numbers (floating point) can look deterministic yet end up giving non-reproducible results. It also explains how to fix it.

Processes not people. This is important: the problem here is not really the individuals working on the model. The people in the Imperial team would quickly do a lot better if placed in the context of a well run software company. The problem is the lack of institutional controls and processes. All programmers have written buggy code they aren't proud of: the difference between ICL and the software industry is the latter has processes to detect and prevent mistakes.

For standards to improve academics must lose the mentality that the rules don't apply to them. In a formal petition to ICL to retract papers based on the model you can see comments "explaining" that scientists don't need to unit test their code, that criticising them will just cause them to avoid peer review in future, and other entirely unacceptable positions. Eventually a modeller from the private sector gives them a reality check. In particular academics shouldn't have to be convinced to open their code to scrutiny; it should be a mandatory part of grant funding.

The deeper question here is whether Imperial College administrators have any institutional awareness of how out of control this department has become, and whether they care. If not, why not? Does the title "Professor at Imperial" mean anything at all, or is the respect it currently garners just groupthink?

Insurance. Someone who works in reinsurance posted an excellent comment in which they claim:

- There are private sector epidemiological models that are more accurate than ICL's.
- Despite that they're still too inaccurate, so they don't use them.
- "*We always use 2 different internal models plus for major decisions an external, independent view normally from a broker. It's unbelievable that a decision of this magnitude was based off a single model*"

They conclude by saying "*I really wonder why these major multinational model vendors who bring in hundreds of millions in license fees from the insurance industry alone were not consulted during the course of this pandemic.*"

A few people criticised the suggestion for epidemiology to be taken over by the insurance industry. They had insults ("mad", "insane", "adding 1 and 1 to get 11,000" etc) but no arguments, so they lose that debate by default. Whilst it wouldn't work in the UK where health insurance hardly matters, in most of the world insurers play a key part in evaluating relative health risks.

END