FAKED IN LONDON:

UK destroys the World Economy with faulty CV-19 data. Zerohedge.com

Computer Model That Locked Down The World Turns Out To Be badly coded.

It was a UK Imperial College computer model that forecasted 500K deaths in the UK (and 2.5 million in the US) should policymakers pursue a "herd immunity" approach (a la Sweden), that influenced them to reverse course and go full lockdown instead. The model was produced by a team headed by Neil Ferguson, (who recently resigned his post advising the UK government when it surfaced that he was himself violating lockdown directives by breaking self-isolation for dalliances with a married woman).

The source code behind the model was to be made available to the public, and after numerous delays and excuses in doing so, has finally been posted to GitHub

A code review has been undertaken by an anonymous ex-Google software engineer here, who tells us the GitHub repository code has been heavily massaged by Microsoft engineers, and others, in an effort to whip the code into shape to safely expose it to the public. Alas, they seem to have failed and numerous flaws and bugs from the original software persist in the released version. Requests for the unedited version of the original code behind the model have gone unanswered.

The most worrisome outcome of the review is that the code produces "non-deterministic outputs"

Non-deterministic outputs. Due to bugs, the code can produce very different results given identical inputs. They routinely acted 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.

The documentation proffers the rationalization that iterations of the model should be run and then differing results averaged together to produce a resultant model. However, any decent piece of software, especially one that is creating a model, should produce the same result if it is fed the same initial data, or "seed". This code doesn't.

"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.

In one instance, a team at the Edinburgh University attempted to modify the code so that they could store the data in tables that would make it more efficient to load and run. Performance issues aside, simply moving or optimizing *where* the input data comes from should have no effect on the *output* of processing, given the same input data. What the Edinburgh team found however, was this optimization produced a variation in the output, ***"the resulting predictions varied by around 80,000 deaths after 80 days"*** which is nearly 3X the total number of UK deaths to date.

Edinburgh reported the bug to Imperial, who dismissed it as "a small non-determinism" and told them the problem goes away if you run the code on a single CPU (which the reviewer notes "is as far away from supercomputing as one can get").

Alas, the Edinburgh team found that software still produced different results if it was run on a single CPU. It shouldn't, provided it is coded properly. Whether the software is run on a single CPU or multi-threaded, the only difference should be the speed at which the output is produced. Given the same input conditions, the outputs should be the same. It isn't, and Imperial knew this.

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".

Another team even found that the output varied depending on *what type of computer* it was run on.

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".

The response illustrates the burning question: Why didn't the Imperial College team realize their software was so flawed?

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.

Most of us are familiar with the computing adage, "Garbage In/Garbage Out" and the untrained reader may think that's what being asserted in this code review. It isn't. What's being asserted is that output is garbage, *regardless of the input.*

In this case, the output we're experiencing as a result is a worldwide lockdown and shutdown of the global economy, and we don't really know if this was necessary or not because we have no actual data (aside from Sweden) and severely flawed models.

Read the entire code review here.

Tyler Durden Thu, 05/07/2020 - 22:25

Computer Model That Locked Down The World Turns Out To Be Sh*tcode