Q&A

GI Spring Webinar Series: Is now the time for Capital Modelling in python?

1. Great insights. What would be the difference in the cost of using an existing capital solution versus using Python?

Dan: Python and most packages you would use to build a bespoke model are free. You would also likely be using free looks like VSCode and git for the model development. So the software costs will be very low for a bespoke build. This needs to be offset against the higher development and maintenance costs that come with owning a complex code base with multiple external dependencies.

Personally, I think the right solution lies in the middle. I think it is beyond the appetite and capacity for most capital teams to take on a completely bespoke build in Python, in the way Felix's team has done, and I think the strain on the regulators to keep up with many divergent modelling approaches will be a barrier. We won't see large scale adoption of Python for capital until we have more established example models/library solutions that people can adopt. It remains to be seen whether these will emerge out of an open source movement or developed and sold as another proprietary solution. I think the latter makes most sense, as there is clearly the market already for vendor solutions in this space, and it is difficult for an open source project to get enough traction and support to get all the boring but important stuff done like validation. Whether open source or proprietary, it is likely that any Python based solution would significantly undercut the cost of he existing vendor models, as so much advantage can be made of the excellent Python ecosystem that already exists, so there is no reinventing the wheel. Personally, I am working to develop a specialized package for capital modelling in Python called CaPytal, and some library style models that use this package. CaPytal aims to use the best of what is freely available in Python and the scientific compute packages, and wrap it up in an easy to use syntax aimed specifically for the capital modelling workflow. Please get in touch if you'd like to know more about this project.

2. Really interesting. For Felix - can you model a risk area on Python while keeping the rest of the model on a vendor solution? If yes, how do you plug these?

Felix: You certainly can. For instance cat models would typically always reside outside of Python as almost the entire market uses the main vendors. Similarly, ESG data typically comes from a third party source. But you could also even separate say reserve risk and underwriting risk if you really wanted to, model them independently and then aggregate them in Python. There are many ways of plugging. For cat models which are

typically not re-run very often, data can be stored on a database, flat file, pickle file or similar and then sourced from there in Python. There are I believe also vendors that offer some Python plug-ins potentially allowing you to control and run a vendor solution from Python code.

3. Should we build an open source capital model in Python that is available to the whole market?

Dan: It is a very interesting idea and it could well be that such a solution emerges. Open source projects typically need a core team of very good individuals to gain traction and maintain progress and quality. It isn't really happy with lots of people just nibbling away around the edges. So it will come down to whether a group of such individuals emerges with the passion to do this. The main beneficiaries would be insurance companies who get to save on their capital modelling spend, so it could maybe work if there was a coalition of carriers sponsoring the project and allowing their staff to spend time working towards it. Personally, I think it is more likely that we will see new vendor solutions emerge, that aren't completely free, but which undercut existing vendor solutions by leveraging what Python and the scientific compute ecosystem already does well.

4. How does Python compare in terms of runtime to other major software platforms?

Dan: Code written directly in Python using Python loops and objects will be slow. But Python packages like Polars and NumPy are written in other languages (Rust and C respectively) and take advantage of excellent vectorisation and parallelisation that makes them as fast as any existing platform for those calculations. The challenge will be putting all the calculations together in a way that uses these packages sensibly, doesn't create bottlenecks with native Python glue code, and manages memory usage sensibly across the model. Like with other languages or vendor capital software, well written models will run faster than poorly written models. The tools are available in Python, and I believe it is entirely possible to build Python based solutions that are as performant as any established capital software on the market, plus with a lot of other benefits such as transparency, ease of adoption, ease of maintenance, ease of debugging, better integrability with other systems, easier cloud deployment, simpler syntax, better LLM and genAI integration, lower cost.

5. how do you mange dependencies?

Martin: On dependency management, in both cases I've been involved in, rigid control of the work environment (managed in both cases by actually setting up the development environment as special purchase machines on the cloud). If you run

separate development and production environments (and you really should) you can build a Python solution into an exe including all the packages (and specific versions of packages) that are needed to run it. Change management processes to update packages, run automated tests, and smoke test to make sure that it still works. The same process as upgrading the version of one of the vesting vendor solutions really.

Felix: Agree, rigid control of work environment. We have taken the decision of upgrading once per year to the latest stable versions of all libraries we use. A requirement file specifies all versions and this file is part of the git repository and the dockerfile reads it in when the container is created. And of course run in dev first before production and test it all through before release.

6. How do you deal with regular updates / builds required in the capital model? Is it easy to do in Python or do you have a UI which supports this and makes this easier?

Dan: If you are managing any complex code base that is developing over time, I would strongly recommend adopting a git based workflow or something similar. This allows you to version control every step of your development, create branches for substantial development and merge in as part of controlled peer review, tag and maintain versions of the code used for key milestone runs, revert back when needed. The integration of git with VSCode is probably the smoothest and most intuitive user interface for git that I have worked with, and Python, being all simple text files integrates really nicely with a git workflow, as all changes can be inspected and reverted on a line by line basis.

7. What platform do you best recommend to run python models? Eg. Local / Azure / AWS etc

Felix: With Azure you get dev ops which is nice. For development being able to run locally, perhaps with a smaller scope or a lower number of simulated years certainly helps. I don't have any experience with AWS, but this is certainly viable too.

Dan: All are viable options. Personally I prefer a workflow on my own machine. As soon as I have to remote into virtual desktops elsewhere my user experience tends to degrade a bit. So for intensive iterative workflows I'd like to keep everything as native to my laptop/desktop as possible. So I am a big fan of having the ability to access data and run models locally, for development purposes, even if for the larger production model runs I need to remote in somewhere or send off an api call to run the model. Ideally that would be available from the same workflow tools I use for my local development work, so I can trigger large production runs remotely or local development runs for inspection and debugging from the same interface with single click convenience. This type of workflow is certainly achievable with Python.

8. If someone wants to learn Python, where can they turn to as a start?

Dan: To learn Python you first need a basic grasp on the syntax, importing modules, functions, classes etc. There are good online resources like codecademy for this.

Early on I would complement this syntax learning with a good book like Fluent Python by Luciano Ramalho.

You also need to know how to setup virtual environments and manage dependencies using looks like poetry. I would get into good habits with this early, but it does add to the initial learning curve. Similarly it is good to learn version control through git, though not essential initially until you are working on a substantial project.

You will most likely be using established packages heavily such as NumPy, pandas/polars, SciPy for modelling, or tools like dash/niceGUI/plotly for reporting. These have good documentation and there are also books and tutorials on each of these. There is too much there to try and read up on in advance, so you'll need to learn these as you come across them on specific projects.

I would recommend also getting familiar with wring code through notebooks like Jupyter Notebooks or Marimo. These are great for exploratory analysis in Python or building dynamic reports.

The most valuable way to learn is by trying to code something yourself. Starting with some scripts to clean data or automate parts of your workflow is a great start. If you are interested in really accelerating your learning you could collaborate on a large project, such as an open source project, or become a contributor to the CaPytal python project.

You can also check out Tom Durkins presentation through the same IFoA spring series which talks about how actuaries can get into coding and develop their coding mindset.

Please connect with me on Linked In if you are interested in learning more about Python for capital, as I also post material and references on their that are relevant.

9. How easy is it to expand a python capital model solution to parallel/distributed processing if necessary?

Dan: Some Python packages already offer very good vectorisation and parallelisation of compute, and you would likely be using these for most of your calculations anyway. NumPy is vectorised and very fast but works naturally on a single machine and generally single core. You can achieve further parallelisation across cores using Python native packages like joblib. Polars is naturally multi-core but still single machine. If you mean going further, and distributing a single model calculation across multiple machines, then this would probably need to be considered early on in the design of your model. I'm

not aware of perfect drop-ins that will automatically distribute the compute across agents, but some packages are emerging that are specifically for this type of out-of-memory computation, such as Dask and Ray. The implementation approach and benefit will also be affected by your actuarial model design. Are there places in the middle of the model where stats need to be calculated, or simulations reshuffled, that require a re-aggregation of all of the sims, and if so, do these bottlenecks offset the advantages? You would also need to think about how random seeds are managed across agents if each is doing its own simulation. There are costs to adopting these distributed libraries if they aren't needed, they are typically harder to debug, they can be slower for smaller problems that do fit on a single machine, they have a steeper learning curve and require more infrastructure complexity. So I would say multi-agent parallelism needs to be designed in early and probably only where the problem is known to require it.

10. how available are pre-built packages for capital modelling calculations that are designed optimally for our uses? of course we'd want in-house code dev, but don't want everyone "reinventing the wheel" for core functionality

Felix: I don't think there are very dedicated capital modelling packages, but there are plenty of mathematical / statistical packages that are very useful, such as NumPy, SciPy etc. There is rarely a need to program mathematical logic.

Dan: There are many great Python packages that do bits of what we need very well, SciPy for simulation, NumPy for compute, Pandas/Polars for filtering and joining. I don't believe any of these are a perfect one-stop-shop for all of it. I think there is space for a custom package to emerge that wraps the best of these open source packages into a single optimised package targeted for the actuarial/capital modelling workflow. This would simplify the dependency management that can be a challenge when using multiple third party packages together, as there would only be one package needed for the whole model. It would provide a super clean and intuitive syntax familiar to users of the established modelling tools. It would abstract away the need to manage your own data structures and formats (arrays vs dataframes, rows vs columns etc), and it could provide additional functionality beyond those provided by the existing packages, such as management of dependency groups for simulation reordering copula approaches. It could also go further and provide additional config layers (for controlling things like sim count and seed across a whole model), integrate with supporting data and run version management frameworks, integrate logging and profiling packages to support model inspection and development, and include UI components for managing runs and inspecting results. But it definitely wouldn't try to reinvent the wheel, and would use the best of the open source packages available to accelerate its development and keep its

production and maintenance cost to a minimum. If anyone in interested in bringing such a package about please let me know.

11. How can you incorporate ESG - economic scenario generator in python capital model?

Dan: Python is brilliant for ingesting data from various formats and apis. Pandas and Polars for example, can easily import from csv, database, parquet format using in-built methods. For large external datasets like Cat or ESG data, it is likely that these will be imported from a file into a dataframe like format and then processed for use in the model. This is really no different to what I have seen with the existing vendor model solutions where the Cat and ESG tends to get imported from a file, typically a series of CSVs.

12. "Feedback point: I appreciate it was a discussion rather than a presentation, but it might have been fun to see a simple example of what Python code for a simulation model might look like."

Dan: I'm hoping to release some materials soon illustrating what an implementation of some common capital modelling logic steps would look like in NumPy, Polars and CaPytal. Please follow me on Linked In or get in touch if you would be interested in that sort of thing.