Implement a Snakemake backend for RECAST workflows

- Fellow: Andrii Povsten
- Project Mentors: Matthew Feickert, Lukas Heinrich

Project Description

Developing optimal cuts within a phase space that is sensitive to the desired model and evaluating the background of the standard model, along with systematics, can be a time-consuming task for analysis teams comprising multiple people. Moreover, as time goes by, new theories may emerge, leading to alternative models that manifest in the same phase space. In such cases, it would be more efficient for scientists to make adjustments to the original analysis and re-run it with the new models. However, the original analysts may have moved on to other projects. To ensure the reusability of analyses and minimize reliance on the original team, the analysis needs to be archived in a manner that allows for easy re-execution. This is where RECAST[1] comes into play.

RECAST is a framework that extends the impact of existing analyses conducted in LHC experiments. It is part of a broader initiative known as REANA[4], which aims to enhance the reproducibility of particle physics data analysis. By automating the process of passing new signal models through an analysis during its development phase, RECAST enables the analysis to be effortlessly reused in the future for interpreting new signal models within the same phase space.

Initially, when RECAST was implemented for ATLAS, there was a lack of a suitable workflow language with robust Linux container support. To address this, a tool called yadage[3] was employed, which allowed users to define each step of the analysis, specify the required analysis environment (i.e., container) for each step, and integrate them into the overall analysis workflow. However, REANA has now begun collaborating with the Snakemake[4] workflow management system, which has gained popularity in the wider scientific community and provides mature Linux container support. Snakemake is a workflow engine defined using a Snakefile, employing a domain-specific language with a syntax resembling YAML and Python. Workflows in Snakemake are described in terms of rules, where each rule specifies the inputs, Python commands to execute, and conditions such as the Docker container to use.

Snakemake creates a Directed Acyclic Graph that represents the data analysis workflow, resulting in a more readable interface.

A goal for making RECAST a more sustainable project towards the future is to implement a new backend for RECAST, utilizing the Snakemake workflow management system. This integration aims to enable the systematic interpretation of LHC searches by harnessing the features and benefits provided by Snakemake. A first step towards this goal would be to reimplement the logic of a yadage based RECAST workflow, like the ATLAS RECAST public examples[5], in Snakemake and verify that these Snakemake workflows are compatible with the REANA Snakemake workflow engine. Additional extension of the recast-atlas library's CLI API to support submission of Snakemake based workflows to REANA would provide a full example of the benefits, which would then motivate future development work of the recast-atlas library which may extend beyond the reach of this Fellow project.

The implementation of the Snakemake backend should prioritize performance optimization and scalability. This includes exploring strategies for efficient parallelization of workflow execution, utilization of distributed computing resources, and minimizing resource usage while maintaining analysis result accuracy. By focusing on these key objectives, the project aims to enhance the capabilities of RECAST by implementing a modern and efficient backend utilizing the Snakemake workflow management system.

Deliverables

Deliverable for this project would include:

- A GitHub repository that with example workflows from the ATLAS RECAST examples GitLab group translated into a Snakemake.
- Examples in the GitHub repository of these SnakeMake workflows being run with the REANA Snakemake workflow engine.
- An outline of how this work could be used to extend the recast-atlas CLI API for submission of Snakemake workflows to REANA.
- Documentation associated with the development process.
- Presentations on the associated work.

Timeline

- **Weeks 1-2**: Familiarize myself with the Snakemake workflow management system and acquire CERN account credentials for running RECAST and REANA

workflows. Understand the design of yadage and run the example RECAST ATLAS workflows of https://gitlab.cern.ch/recast-atlas/examples.

- **Weeks 3-4**: Create workflows in Snakemake that provide the same functionality as the workflows of the examples.
- **Weeks 5-7**: Submit Snakemake workflows to the REANA Snakemake workflow engine and validate the results against yadage workflows. Create and execute new test cases that highlight any differences in strengths between yadage and Snakemake. As a stretch goal, make a pull request to the recast-atlas project to extend the CLI API for submission of workflows to REANA using the Snakemake workflow engine.
- **Weeks 8-10**: Provide user support materials (guidelines, examples, tutorials).
- **Weeks 11-12**: Summarize results as a short report.

References:

[1] RECAST: https://iris-hep.org/projects/recast.html
[2] REANA: https://www.reana.io
[3] Yadage: https://github.com/yadage/yadage
[4] Snakemake: https://snakemake.readthedocs.io/en/stable/
[5] RECAST ATLAS public examples: https://gitlab.cern.ch/recast-atlas/examples