Recast-workflow: Fast Truth-level Interpretations

Presenter: Vladimir Ovechkin - University of Washington CS

Mentors: Shih-Chieh Hsu, Alex Schuy, Lukas Heinrich

IRIS-HEP Summer Fellowship 2020

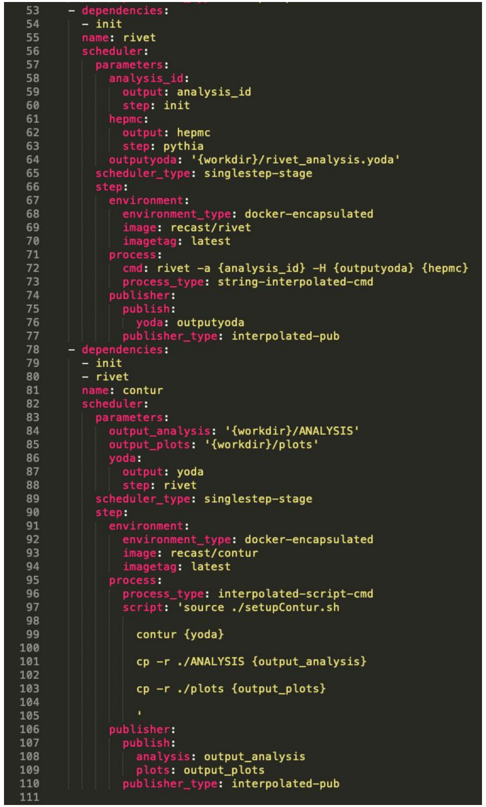
Introduction

- My project was to expand recast-workflow - a python package that builds computational workflows for quickly testing truth-level interpretations.

- To better explain my work, I will review review the motivation and core features of recast-workflow.

Motivation for Recast-wf

- Typical experiment:
  - Generate data according to a model w/ certain params
  - Select interesting data
  - Run statistical analysis
- Yadage uses yaml files to describe and execute this process
  - Much faster than running multiple times by hand
- Recast-wf quickly builds workflows to run truth-level reinterpretations before committing to full simulation



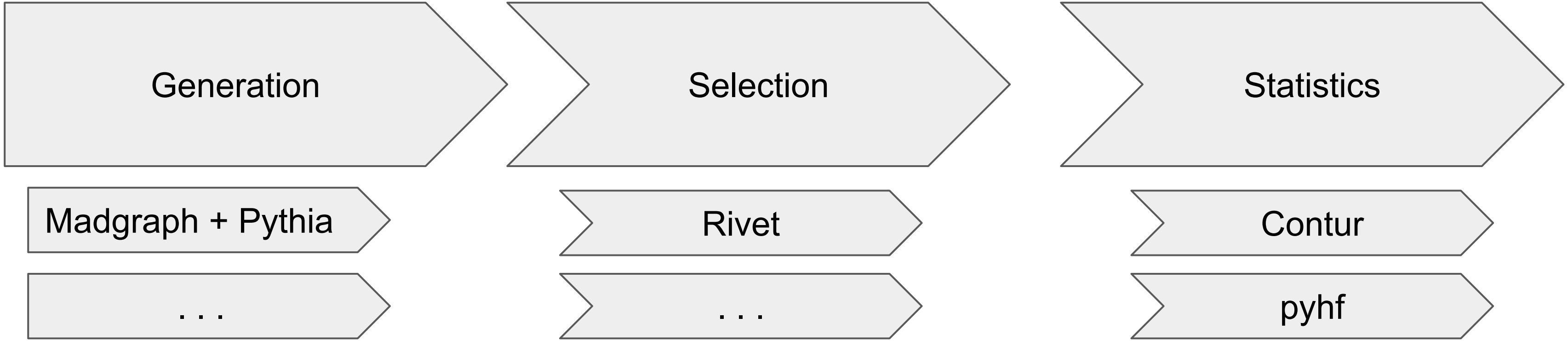
```yaml
- dependencies:
  - init
  name: rivet
  scheduler:
    parameters:
      analysis_id:
        output: analysis_id
        step: init
      hepmc:
        output: hepmc
        step: pythia
      outputyoda: '{workdir}/rivet_analysis.yoda'
    scheduler_type: singlestep-stage
    step:
      environment:
        environment_type: docker-encapsulated
        image: recast/rivet
        imagetag: latest
      process:
        cmd: rivet -a {analysis_id} -H {outputyoda} {hepmc}
        process_type: string-interpolated-cmd
      publisher:
        publish:
          yoda: outputyoda
        publisher_type: interpolated-pub
- dependencies:
  - init
  - rivet
  name: contur
  scheduler:
    parameters:
      output_analysis: '{workdir}/ANALYSIS'
      output_plots: '{workdir}/plots'
      yoda:
        output: yoda
        step: rivet
    scheduler_type: singlestep-stage
    step:
      environment:
        environment_type: docker-encapsulated
        image: recast/contur
        imagetag: latest
      process:
        process_type: interpolated-script-cmd
        script: 'source ./setupContur.sh

          contur {yoda}

          cp -r ./ANALYSIS {output_analysis}

          cp -r ./plots {output_plots}

          '
      publisher:
        publish:
          analysis: output_analysis
          plots: output_plots
        publisher_type: interpolated-pub
```

Excerpt from Example Workflow

What is Recast Workflow?

- Recast-wf abstracts an experiment to 3 steps
- Then, the user can pick what tool (subworkflow) they would like to use for each step

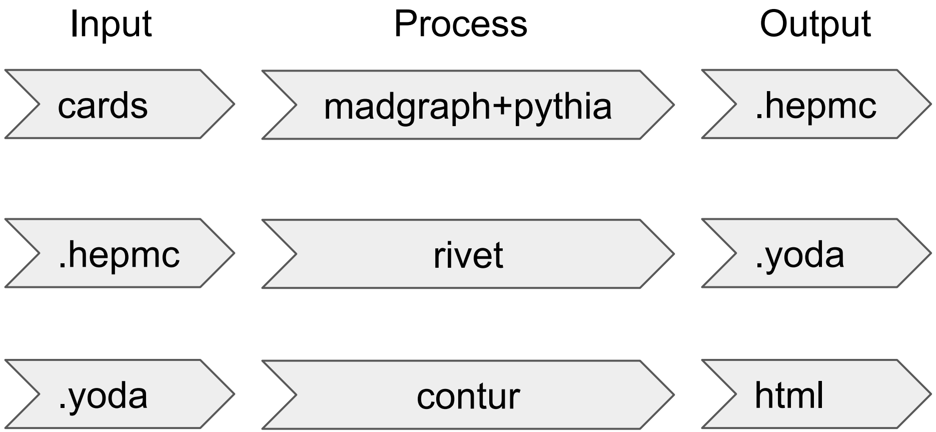
Generate simple analyses by reusing several steps:



Recast-wf Internals

- Recast-wf uses configurable file interfaces to build dependency graph of all subworkflows
- Docker images encapsulate each step
- Recast-wf generates valid step combinations

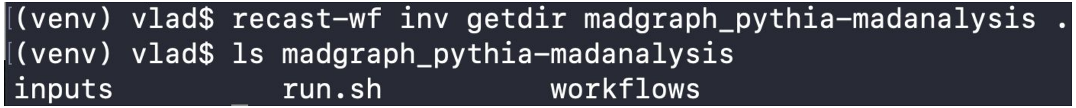
Workflow Visualized



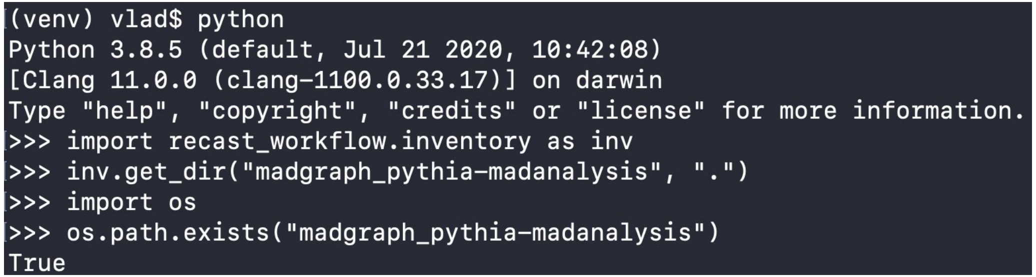
Input	Process	Output
cards	madgraph+pythia	.hepmc
.hepmc	rivet	.yoda
.yoda	contur	html

Achievements

1. Project restructure
- 73 directories, 149 files -> 45 directories, 102 files
- Converted original project to python package (rather than collection of scripts in $PYTHONPATH)
- Command line interface incorporated back into project
- Now, you can use both python and bash interchangeably!



```
[(venv) vlad$ recast-wf inv getdir madgraph_pythia-madanalysis .
[(venv) vlad$ ls madgraph_pythia-madanalysis
inputs          run.sh          workflows
```



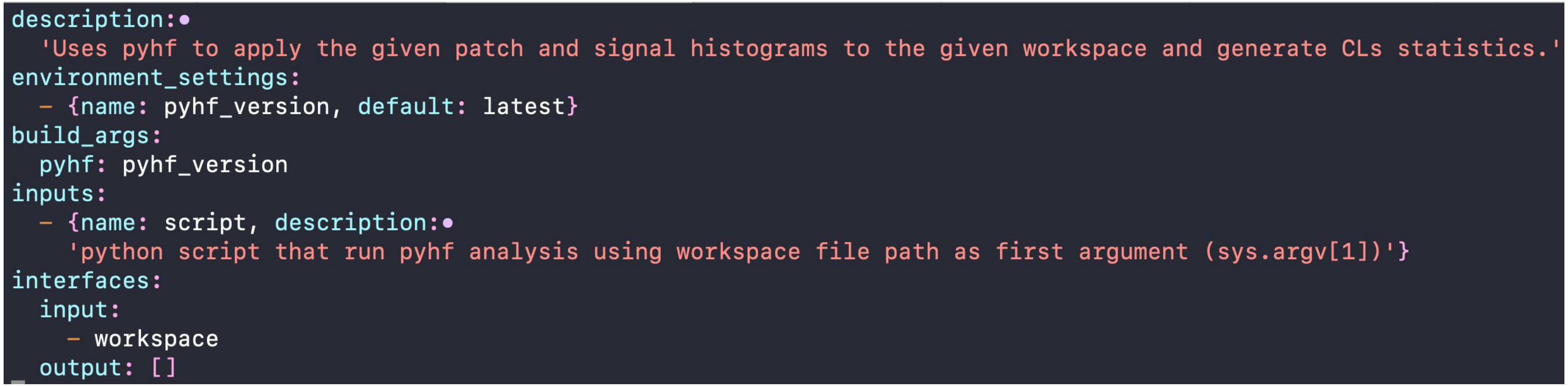
```
[(venv) vlad$ python
Python 3.8.5 (default, Jul 21 2020, 10:42:08)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> import recast_workflow.inventory as inv
[>>> inv.get_dir("madgraph_pythia-madanalysis", ".")
[>>> import os
[>>> os.path.exists("madgraph_pythia-madanalysis")
True
```

Equivalent

Achievements

2. Added pyhf Docker image, subworkflow, and pyhf workspace file interface

- Alternative subworkflow for statistics



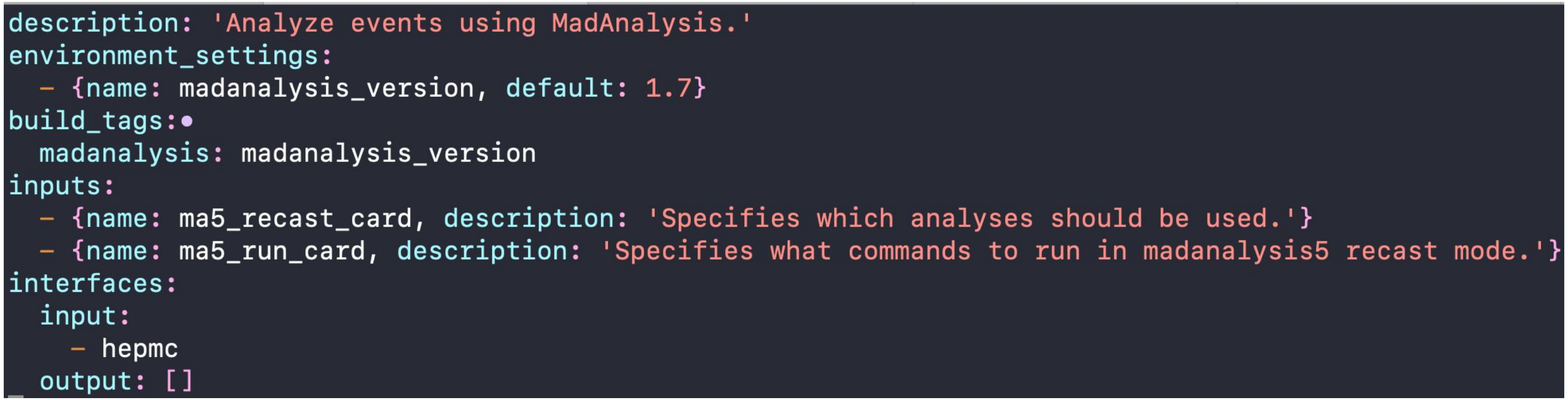
```
description:
  'Uses pyhf to apply the given patch and signal histograms to the given workspace and generate CLs statistics.'
environment_settings:
  - {name: pyhf_version, default: latest}
build_args:
  pyhf: pyhf_version
inputs:
  - {name: script, description:
    'python script that run pyhf analysis using workspace file path as first argument (sys.argv[1])'}
interfaces:
  input:
    - workspace
  output: []
```

Try it out: docker pull recast/pyhf

Achievements

3. Added Madanalysis Docker image and subworkflow

- Can also do selection



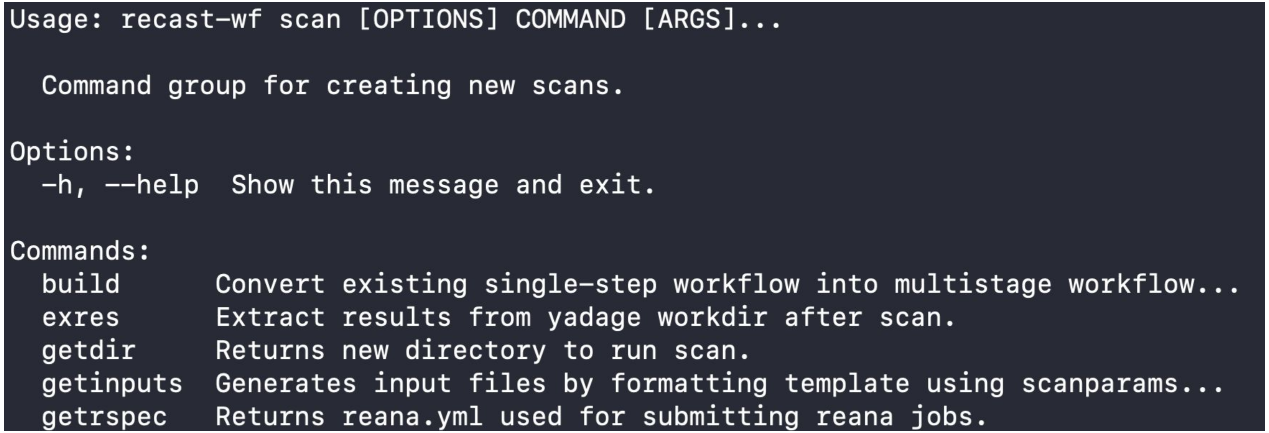
```
description: 'Analyze events using MadAnalysis.'
environment_settings:
  - {name: madanalysis_version, default: 1.7}
build_tags:
  madanalysis: madanalysis_version
inputs:
  - {name: ma5_recast_card, description: 'Specifies which analyses should be used.'}
  - {name: ma5_run_card, description: 'Specifies what commands to run in madanalysis5 recast mode.'}
interfaces:
  input:
    - hepmc
  output: []
```

Try it out: docker pull recast/madanalysis

Achievements

4. Added features to help with parameter scans

- Created features for building multistage workflows from existing single stage workflows
- Created command that builds reana.yaml for given workflow, enabling parallelization on reana cluster



```
Usage: recast-wf scan [OPTIONS] COMMAND [ARGS]...

  Command group for creating new scans.

Options:
  -h, --help  Show this message and exit.

Commands:
  build      Convert existing single-step workflow into multistage workflow...
  exres      Extract results from yadage workdir after scan.
  getdir     Returns new directory to run scan.
  getinputs  Generates input files by formatting template using scanparams...
  getrspec   Returns reana.yml used for submitting reana jobs.
```

Achievements

5. Publish recast-wf

- Two tutorials produced - single point and parameter scan
- Uploaded built distribution to pypi.org so recast-workflow is installable through pip
- Created documentation with the help of sphinx auto generation
- Documentation uploaded to readthedocs.io
- Pypi upload: https://pypi.org/project/recast-workflow/1.0.2/
- Documentation: https://pypi.org/project/recast-workflow/1.0.2/

Achievements

6. Created docker image for recast workflow (some code is non-Windows friendly)

- Figured out how to run docker containers recursively, so building workflows and running them is possible from one container.

Try it out: docker pull recast/workflow

Deliverables

- Recast-wf: https://pypi.org/project/recast-workflow/1.0.2/
- Source Code: https://github.com/vladov3000/recast_workflow
- Tutorials: https://github.com/vladov3000/recast_workflow
- Documentation: https://pypi.org/project/recast-workflow/1.0.2/
- In-depth 40-min video explanation: https://youtu.be/vlFxA82YYEY
- Docker Images (pyhf, madanalysis5, workflow): https://hub.docker.com/u/recast

Summary

- All proposal goals met:
  - Madanalysis5 subworkflow
  - Pyhf subworkflow
  - Reana integration
  - Documentation

- Ideas for future work:
  - Add .yoda to pyhf workspace converter so pyhf can be used with the rest of the subworkflows
  - Develop default python script for pyhf subworkflow
  - Optimize code for generating combinations by traversing dependency graph
  - More unit tests to increase code coverage and package reliability

Thank you!

Contact me at vladov@uw.edu