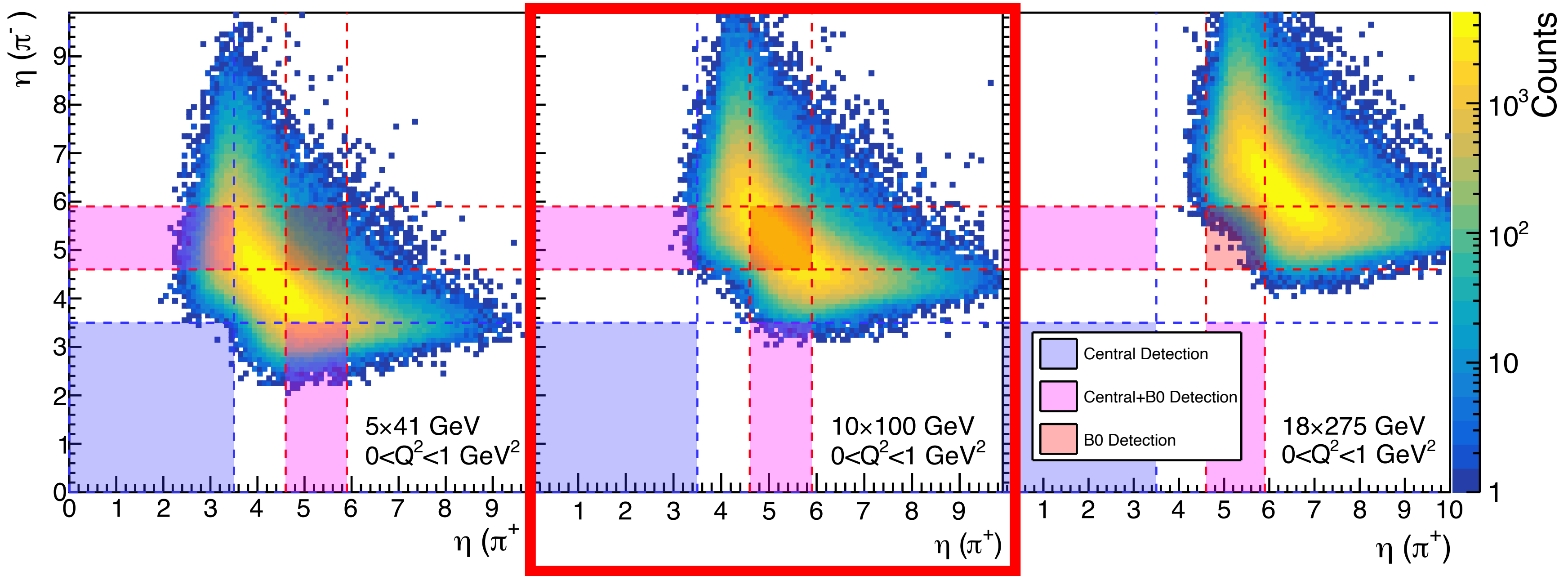
Update on u-channel $\rho \rightarrow \pi^+\pi^-$ Benchmark

Zachary Sweger
University of California, Davis

Supported in part by

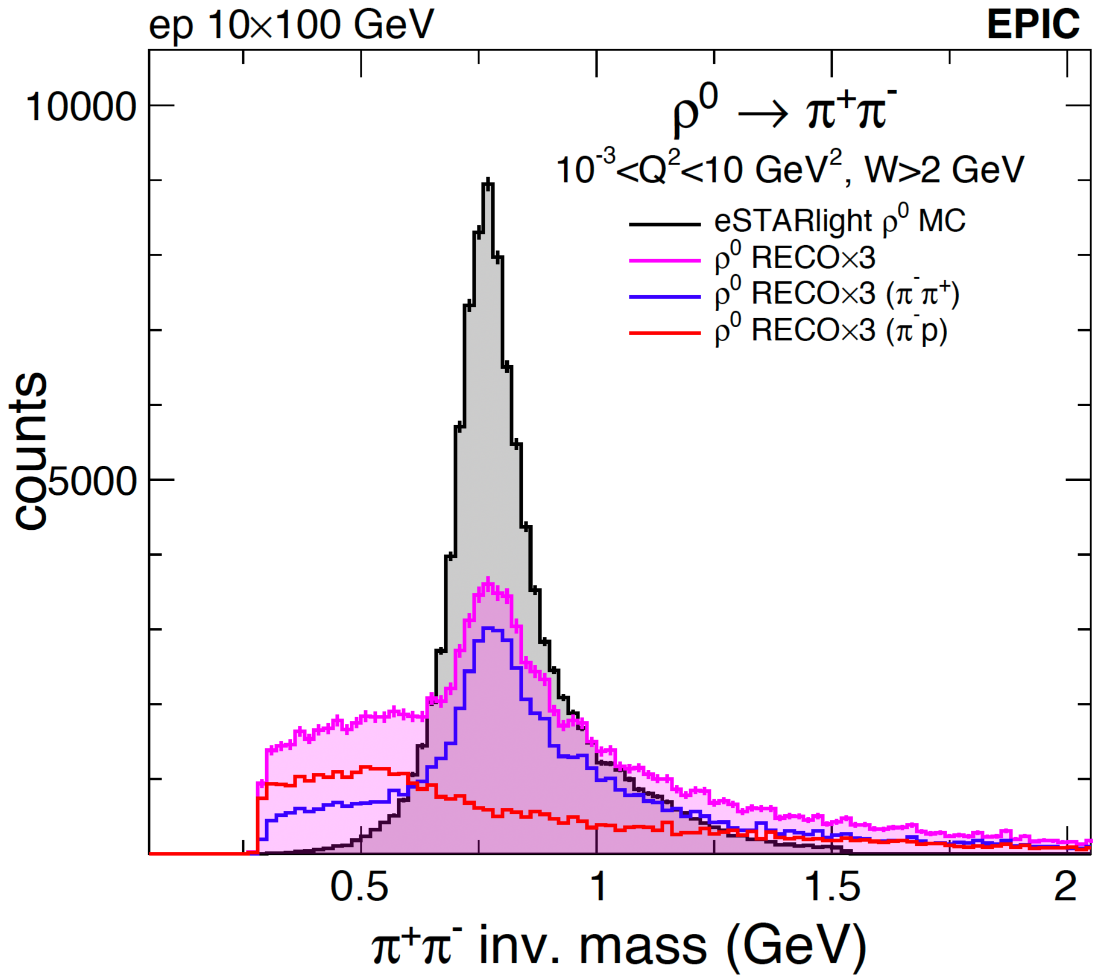
(RECAP) Far-Forward Benchmarks: Backward ρ^0

- Backward ρ production is an excellent benchmark for B0 tracking
- With $\rho \rightarrow \pi^+\pi^-$ we can test two things in the benchmark:
  1. Missing mass $e\,p \rightarrow e\,p\,\rho$
     - ❑ Tests backward (electron) detectors
     - ❑ Tests forward (hadronic) calorimeter and PID
     - ❑ Tests B0 reconstruction capability
  2. ρ mass reconstruction peak
     - ❑ If this changes, something may have changed with beam pipe, or B0 tracking resolution



Backward $\rho^0 \rightarrow \pi^+\pi^-$ Benchmark

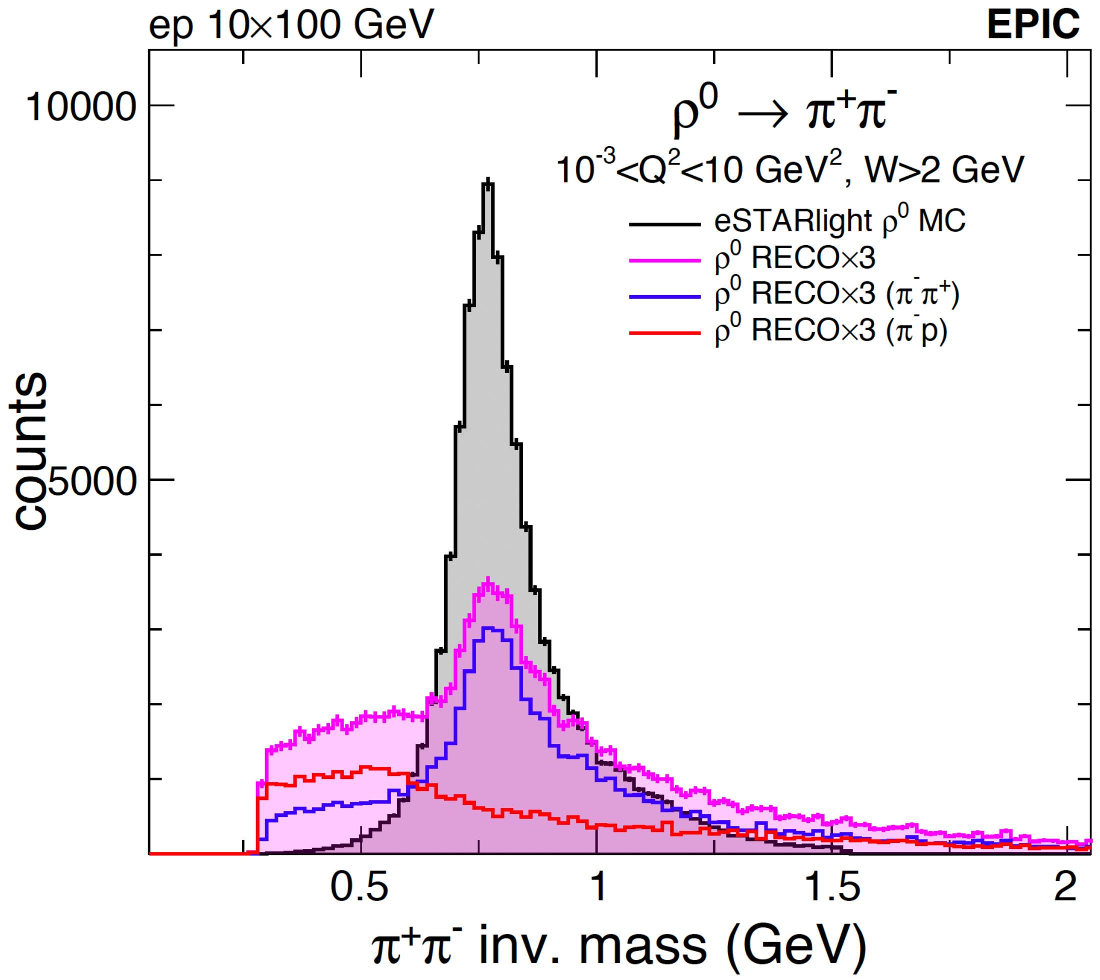
- Simulated 10×100 GeV samples included in December simulations
- With Kong's help, I was able to write some code to analyze the samples
- To construct the magenta histogram at right, I took the invariant mass of each reconstructed negative track with each reconstructed positive track
- At generator level these events only include
  $e+p \rightarrow e'p'\rho^0 \rightarrow e'p'\pi^+\pi^-$
- Background from taking invariant mass of $p+\pi^-$
- I then used the PDG ID of these tracks to identify true $\pi^+ + \pi^-$ and background $p+\pi^-$



Backward $\rho^0 \rightarrow \pi^+\pi^-$ Benchmark

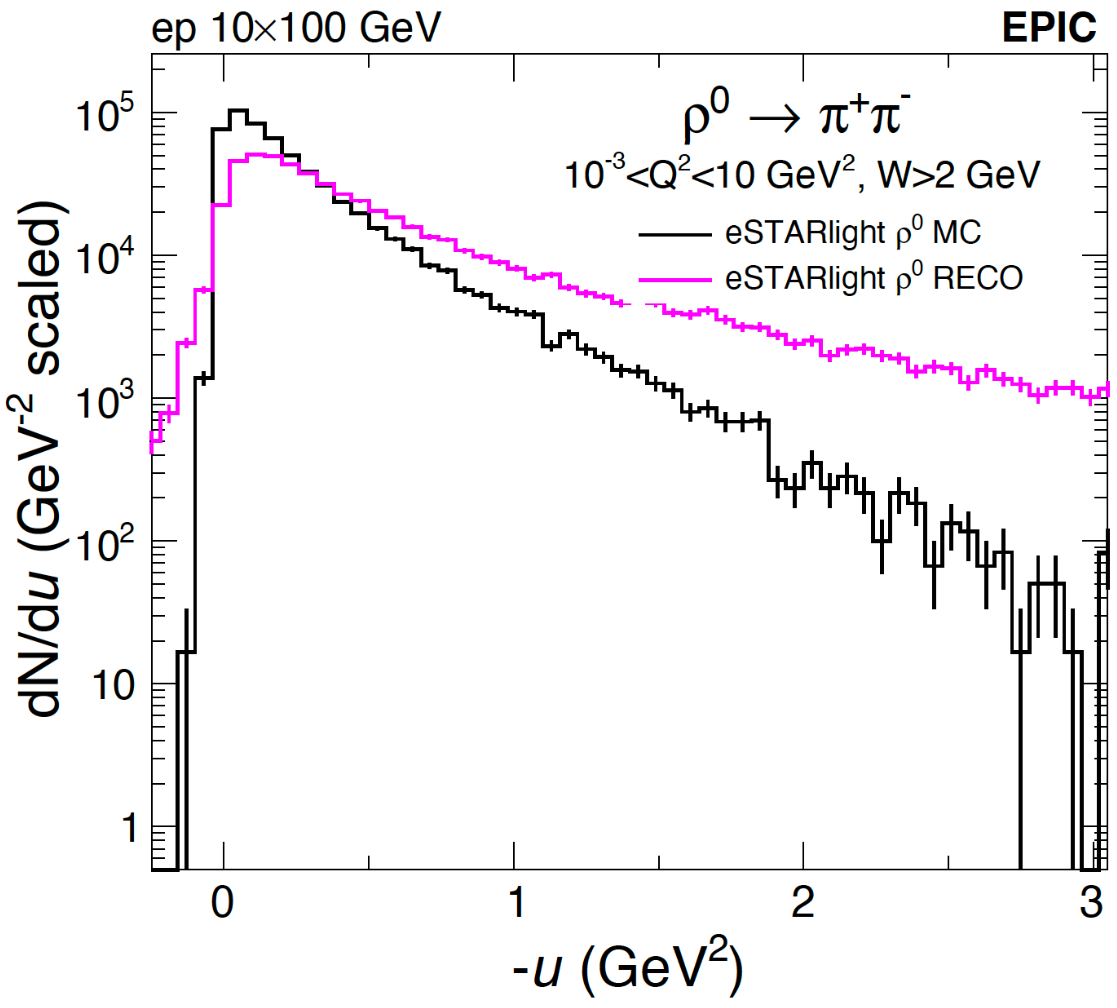
To do:

- check reco for both particles in B0
- check whether reco particles are primaries
- Investigate efficiency for both particles within acceptance
- Investigate whether exclusivity cuts help



Backward $\rho^0 \to \pi^+\pi^-$ Benchmark

- From the oppositely-charged reconstructed particle pairs, I calculate the total 4-momentum as if both particles were pions
- $u = (p_{\rho 0} - p_{p\ beam})^2$
- Again, contamination from $p+\pi^-$ pairing

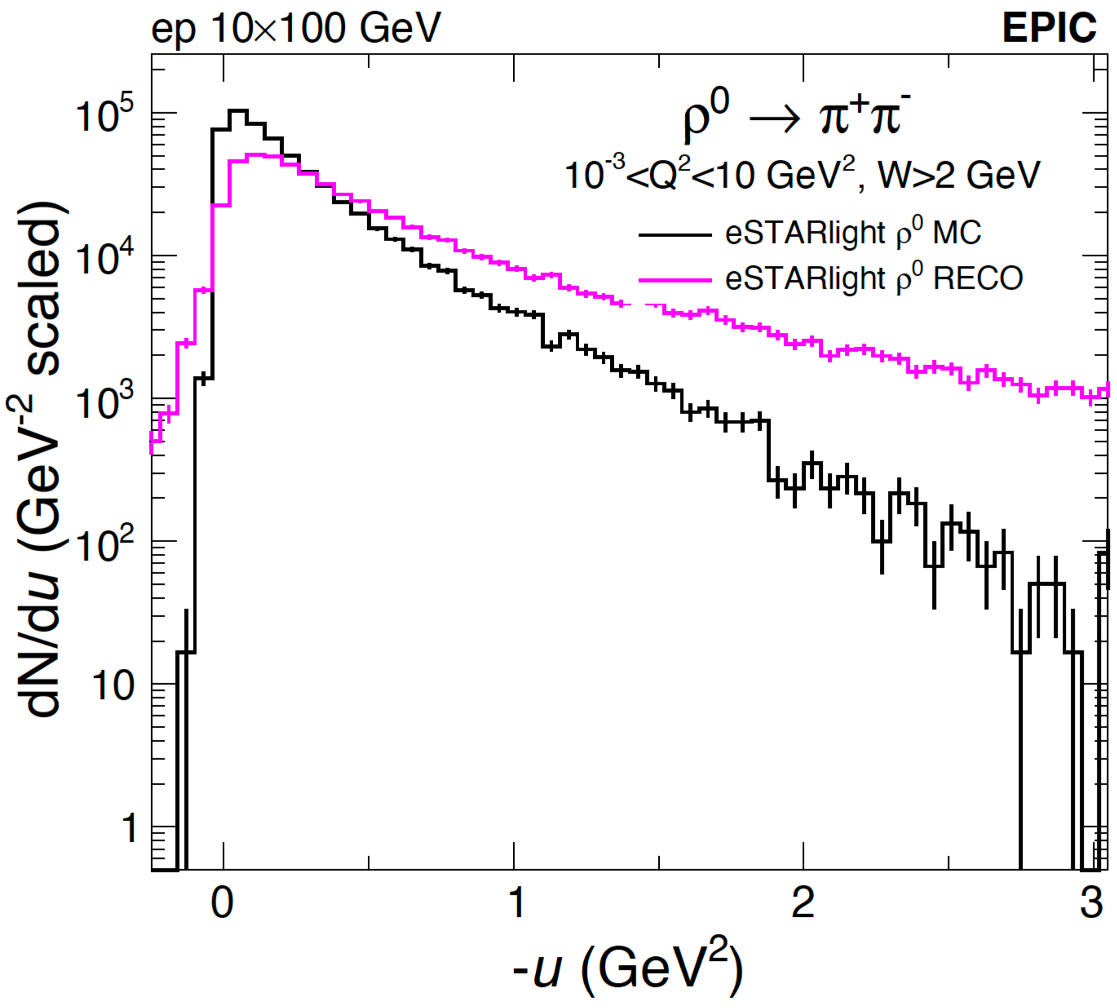


Backward $\rho^0 \rightarrow \pi^+\pi^-$ Benchmark

- From the oppositely-charged reconstructed particle pairs, I calculate the total 4-momentum as if both particles were pions
- $u = (p_{\rho 0} - p_{p\ beam})^2$
- Again, contamination from $p+\pi^-$ pairing

To do:

- Evaluate contamination from proton, and non-primaries
- Evaluate resolution for both particles in B0
- Investigate exclusivity cuts
- Think about how to quantify quality of reconstruction



Integrating Benchmarks

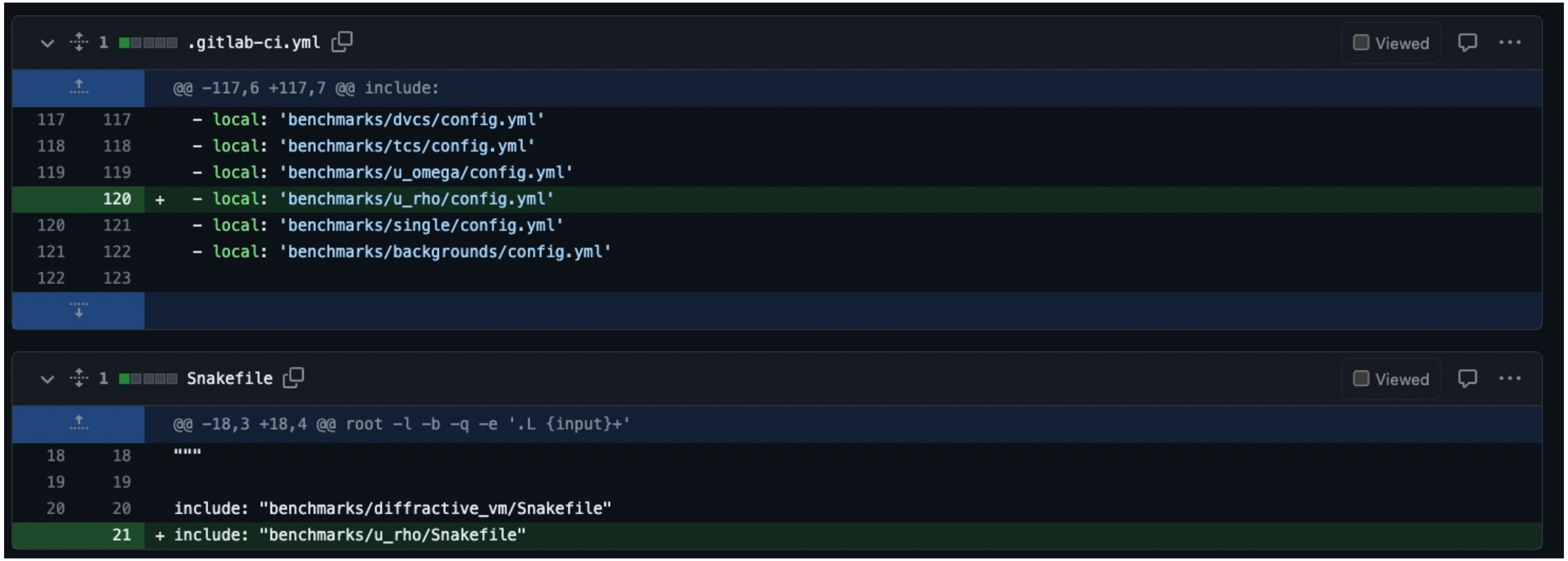
I’ll now share a few things I’ve learned about integrating a benchmark into the ePIC framework

The official page for how to do this is here:
https://eic.github.io/tutorial-developing-benchmarks/

Integrating Benchmarks: Top Level

This is my understanding of how the physics_benchmarks repository works:

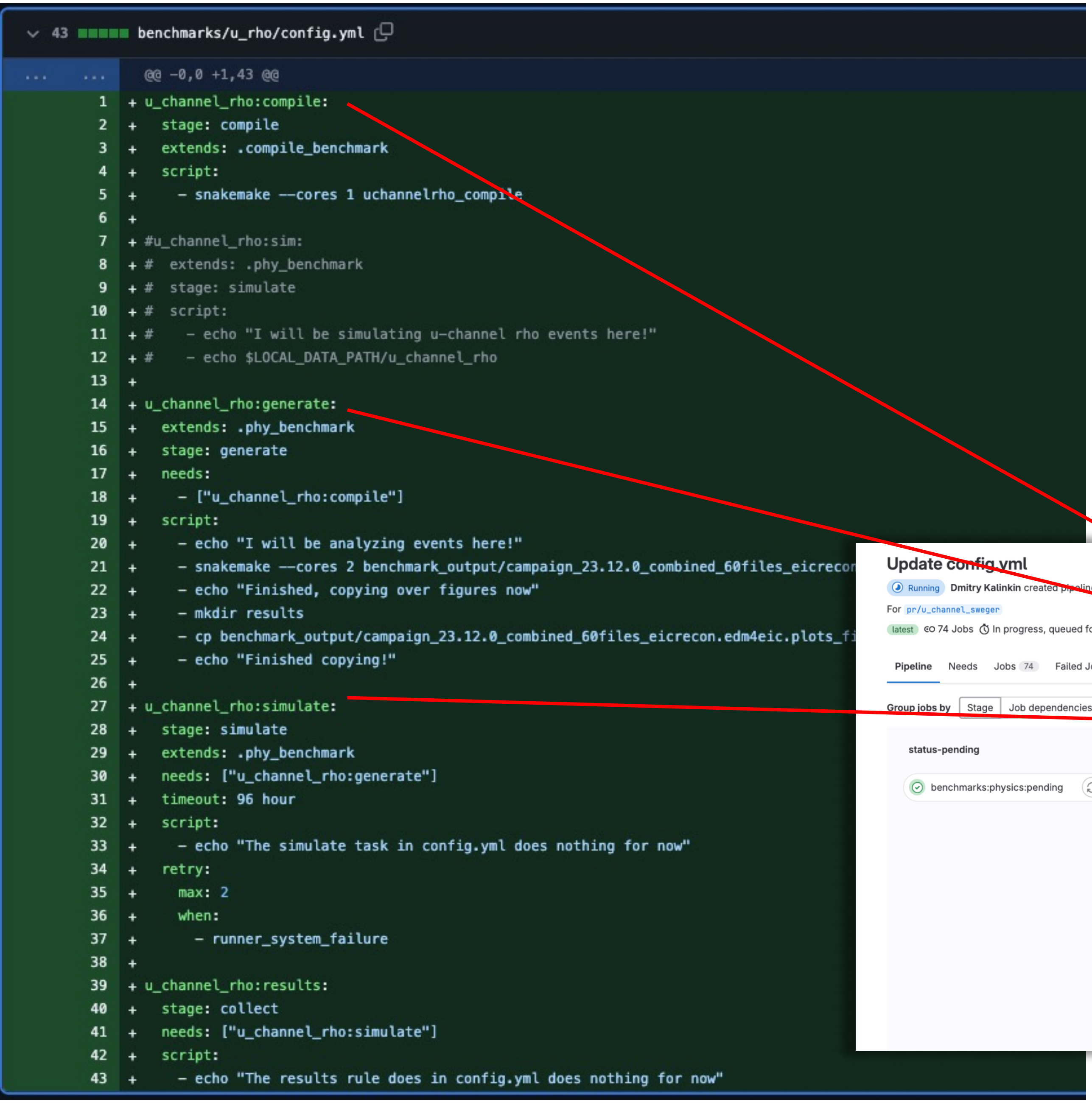
- At the top level, `.gitlab-ci.yml` and `Snakefile` instruct GitLab's pipelines which benchmarks to run.
- You need to update both of these with the paths to your benchmark yml file and snakefile!
- The tutorial only specifies updating `.gitlab-ci.yml`



```
.gitlab-ci.yml
@@ -117,6 +117,7 @@ include:
    - local: 'benchmarks/dvcs/config.yml'
    - local: 'benchmarks/tcs/config.yml'
    - local: 'benchmarks/u_omega/config.yml'
+   - local: 'benchmarks/u_rho/config.yml'
    - local: 'benchmarks/single/config.yml'
    - local: 'benchmarks/backgrounds/config.yml'

```

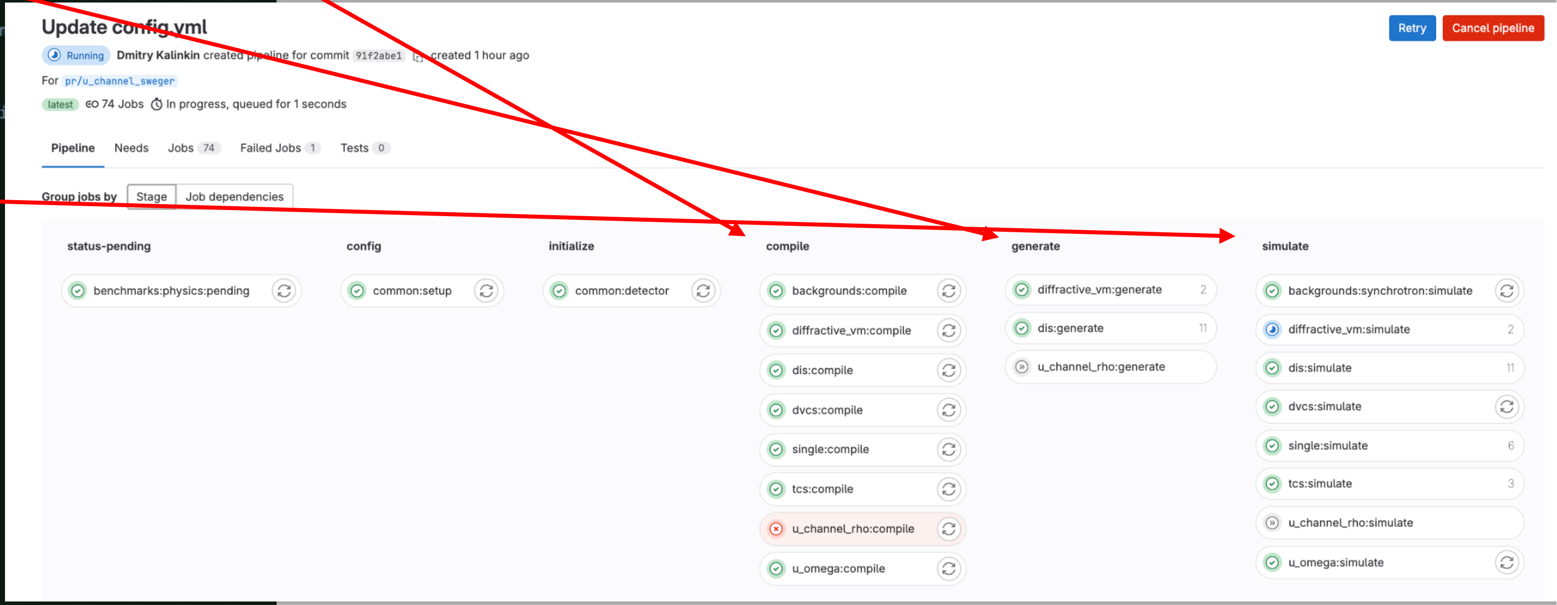
```
Snakefile
@@ -18,3 +18,4 @@ root -l -b -q -e '.L {input}+'
"""

include: "benchmarks/diffractive_vm/Snakefile"
+include: "benchmarks/u_rho/Snakefile"
```

Integrating Benchmarks: Middle Level



```
43 ■■■■■ benchmarks/u_rho/config.yml
... ...  @@ -0,0 +1,43 @@
+ u_channel_rho:compile:
+   stage: compile
+   extends: .compile_benchmark
+   script:
+     - snakemake --cores 1 uchannelrho_compile
+
+ #u_channel_rho:sim:
+ #  extends: .phy_benchmark
+ #  stage: simulate
+ #  script:
+ #    - echo "I will be simulating u-channel rho events here!"
+ #    - echo $LOCAL_DATA_PATH/u_channel_rho
+
+ u_channel_rho:generate:
+   extends: .phy_benchmark
+   stage: generate
+   needs:
+     - ["u_channel_rho:compile"]
+   script:
+     - echo "I will be analyzing events here!"
+     - snakemake --cores 2 benchmark_output/campaign_23.12.0_combined_60files_eicrecon
+     - echo "Finished, copying over figures now"
+     - mkdir results
+     - cp benchmark_output/campaign_23.12.0_combined_60files_eicrecon.edm4eic.plots_fi
+     - echo "Finished copying!"
+
+ u_channel_rho:simulate:
+   stage: simulate
+   extends: .phy_benchmark
+   needs: ["u_channel_rho:generate"]
+   timeout: 96 hour
+   script:
+     - echo "The simulate task in config.yml does nothing for now"
+   retry:
+     max: 2
+     when:
+       - runner_system_failure
+
+ u_channel_rho:results:
+   stage: collect
+   needs: ["u_channel_rho:simulate"]
+   script:
+     - echo "The results rule does in config.yml does nothing for now"
```

- The config.yml associated with your benchmark lists the instructions for GitLab's CI to follow at each step of the pipeline
- When pushing to benchmark, monitoring the pipeline tells you at which step in the config.yml instructions things are going wrong



Update config.yml

Running Dmitry Kalinkin created pipeline for commit 91f2abe1 created 1 hour ago

For pr/u_channel_sweger

latest 74 Jobs In progress, queued for 1 seconds

Pipeline | Needs | Jobs 74 | Failed Jobs 1 | Tests 0

Group jobs by Stage | Job dependencies

status-pending
- benchmarks:physics:pending

config
- common:setup

initialize
- common:detector

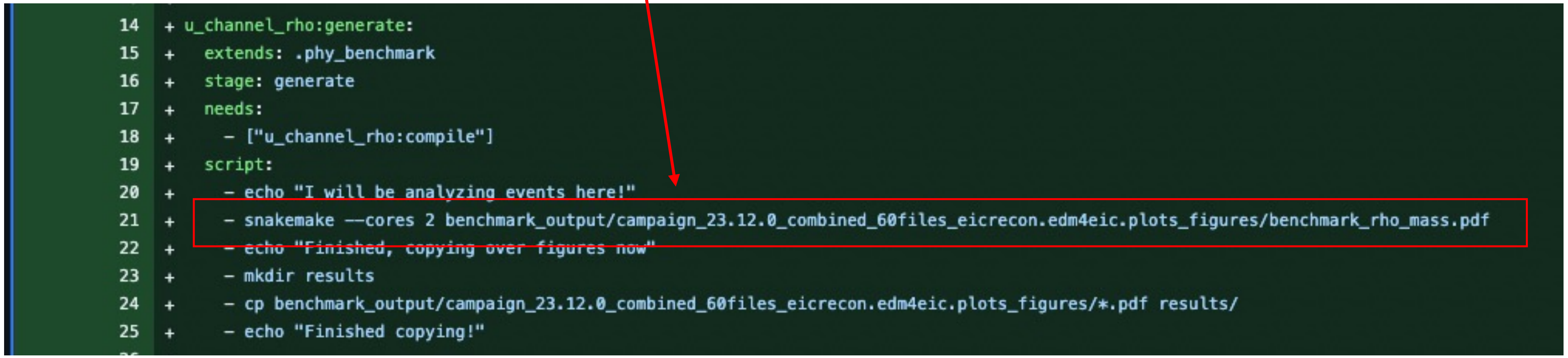
compile
- backgrounds:compile
- diffractive_vm:compile
- dis:compile
- dvcs:compile
- single:compile
- tcs:compile
- u_channel_rho:compile
- u_omega:compile

generate
- diffractive_vm:generate 2
- dis:generate 11
- u_channel_rho:generate

simulate
- backgrounds:synchrotron:simulate
- diffractive_vm:simulate 2
- dis:simulate 11
- dvcs:simulate
- single:simulate 6
- tcs:simulate 3
- u_channel_rho:simulate
- u_omega:simulate

Integrating Benchmarks: Middle Level

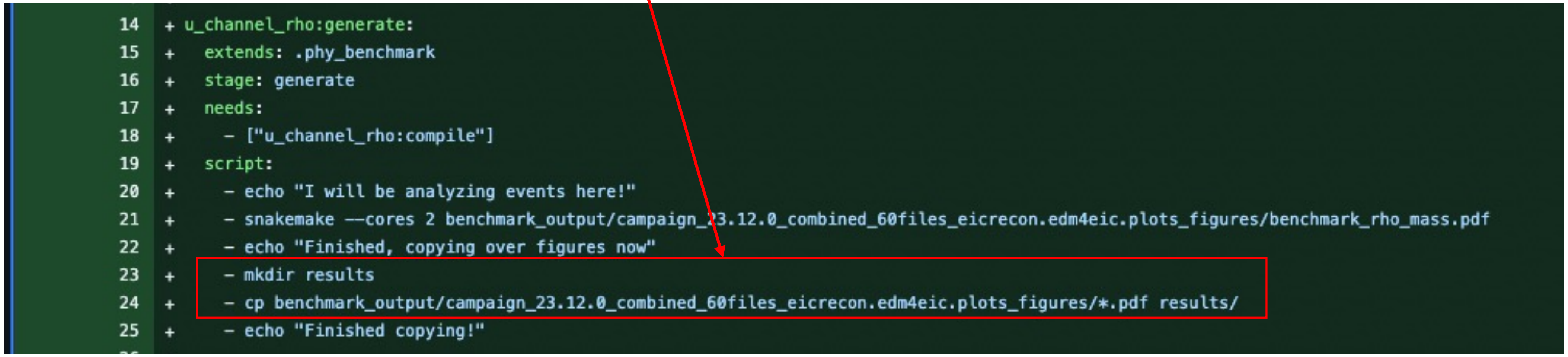
- config.yml can be used to call the Snakefile which contains instructions on how to generate the physics plots
- To create plots using the Snakefile, config.yml calls "snakemake" with the argument being the plot you want to produce



```
+ u_channel_rho:generate:
+   extends: .phy_benchmark
+   stage: generate
+   needs:
+     - ["u_channel_rho:compile"]
+   script:
+     - echo "I will be analyzing events here!"
+     - snakemake --cores 2 benchmark_output/campaign_23.12.0_combined_60files_eicrecon.edm4eic.plots_figures/benchmark_rho_mass.pdf
+     - echo "Finished, copying over figures now"
+     - mkdir results
+     - cp benchmark_output/campaign_23.12.0_combined_60files_eicrecon.edm4eic.plots_figures/*.pdf results/
+     - echo "Finished copying!"
```

Integrating Benchmarks: Middle Level

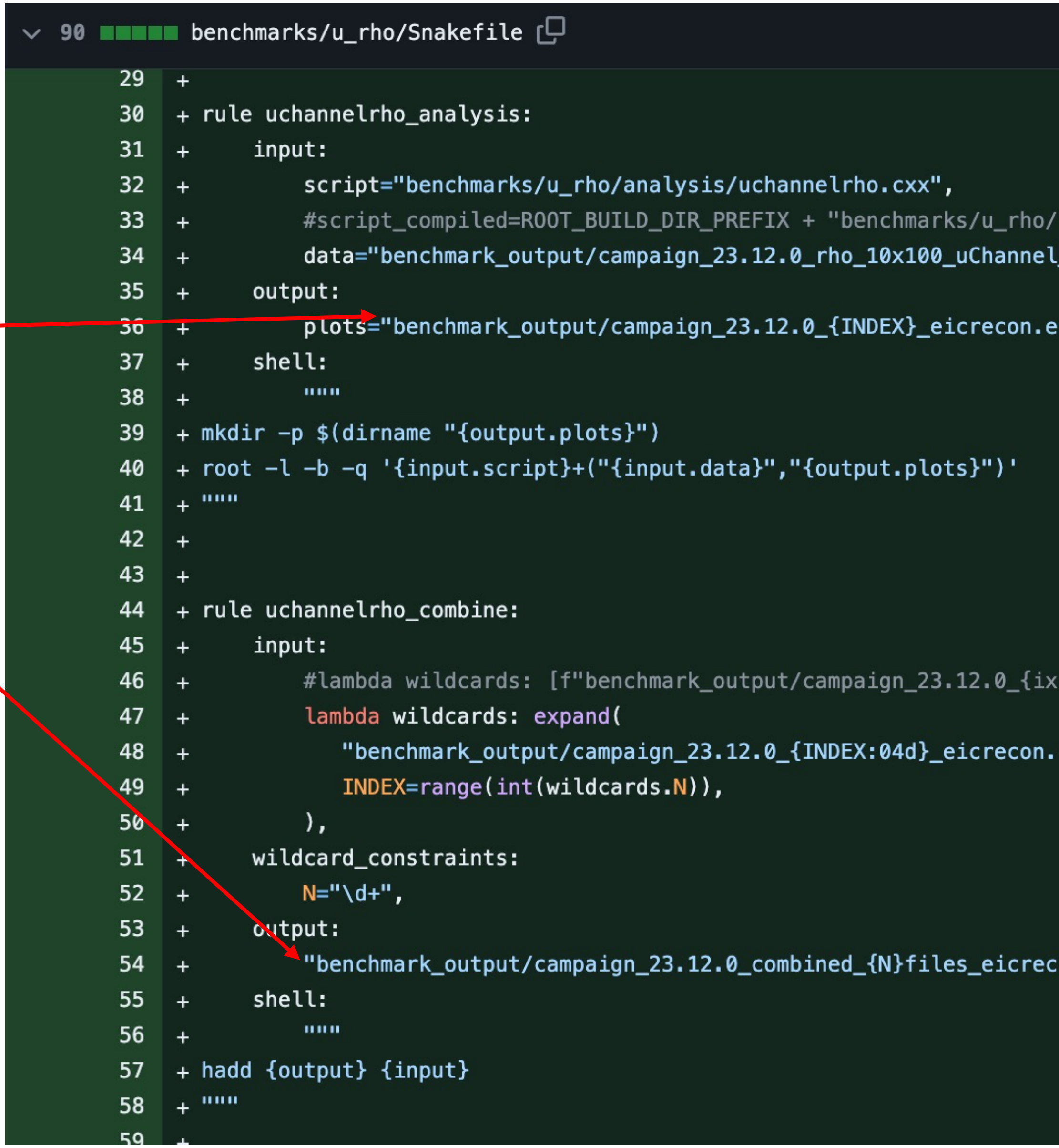
- `config.yml` can be used to call the `Snakefile` which contains instructions on how to generate the physics plots
- To create plots using the `Snakefile`, `config.yml` calls “snakemake” with the argument being the plot you want to produce
- Not mentioned in tutorial: To automatically generate physics plots as artifacts, you have to use `config.yml` to create a directory called “results” and copy the figures there



```
+ u_channel_rho:generate:
+   extends: .phy_benchmark
+   stage: generate
+   needs:
+     - ["u_channel_rho:compile"]
+   script:
+     - echo "I will be analyzing events here!"
+     - snakemake --cores 2 benchmark_output/campaign_23.12.0_combined_60files_eicrecon.edm4eic.plots_figures/benchmark_rho_mass.pdf
+     - echo "Finished, copying over figures now"
+     - mkdir results
+     - cp benchmark_output/campaign_23.12.0_combined_60files_eicrecon.edm4eic.plots_figures/*.pdf results/
+     - echo "Finished copying!"
```

Integrating Benchmarks: Middle Level

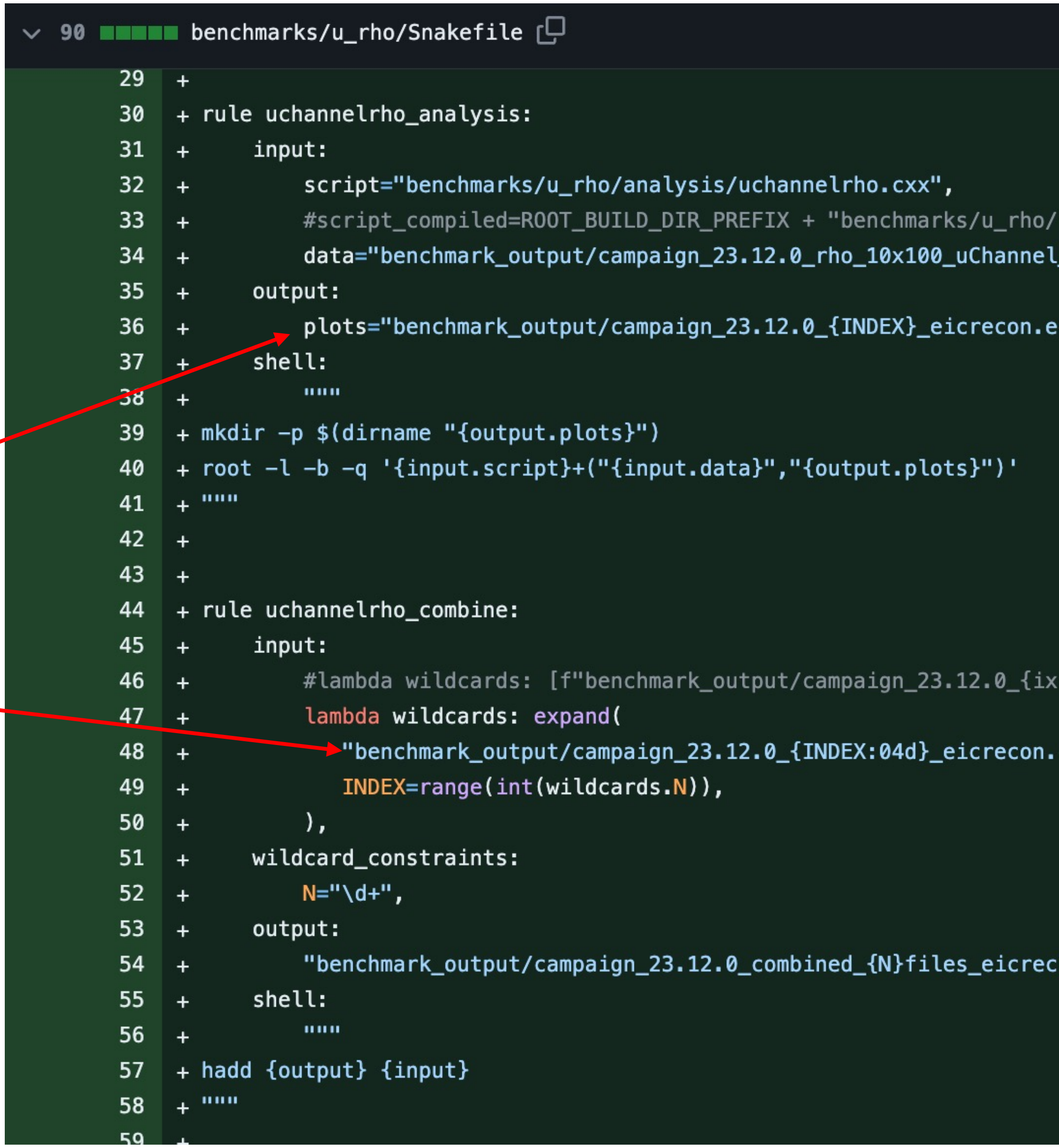
- `Snakefile` contains sets of rules like grabbing reco files, running the analysis, and hadding the output
- Whichever rule you want to execute, run
  `snakemake –cores 2 OUTPUTFILE`
- Where OUTPUTFILE is one of these file names



```
90 ■■■■■ benchmarks/u_rho/Snakefile
+
+ rule uchannelrho_analysis:
+     input:
+         script="benchmarks/u_rho/analysis/uchannelrho.cxx",
+         #script_compiled=ROOT_BUILD_DIR_PREFIX + "benchmarks/u_rho/
+         data="benchmark_output/campaign_23.12.0_rho_10x100_uChannel
+     output:
+         plots="benchmark_output/campaign_23.12.0_{INDEX}_eicrecon.e
+     shell:
+         """
+ mkdir -p $(dirname "{output.plots}")
+ root -l -b -q '{input.script}+("{input.data}","{output.plots}")'
+ """
+
+
+ rule uchannelrho_combine:
+     input:
+         #lambda wildcards: [f"benchmark_output/campaign_23.12.0_{ix
+         lambda wildcards: expand(
+            "benchmark_output/campaign_23.12.0_{INDEX:04d}_eicrecon.
+            INDEX=range(int(wildcards.N)),
+         ),
+     wildcard_constraints:
+         N="\d+",
+     output:
+         "benchmark_output/campaign_23.12.0_combined_{N}files_eicrec
+     shell:
+         """
+ hadd {output} {input}
+ """
+
```

Integrating Benchmarks: Middle Level

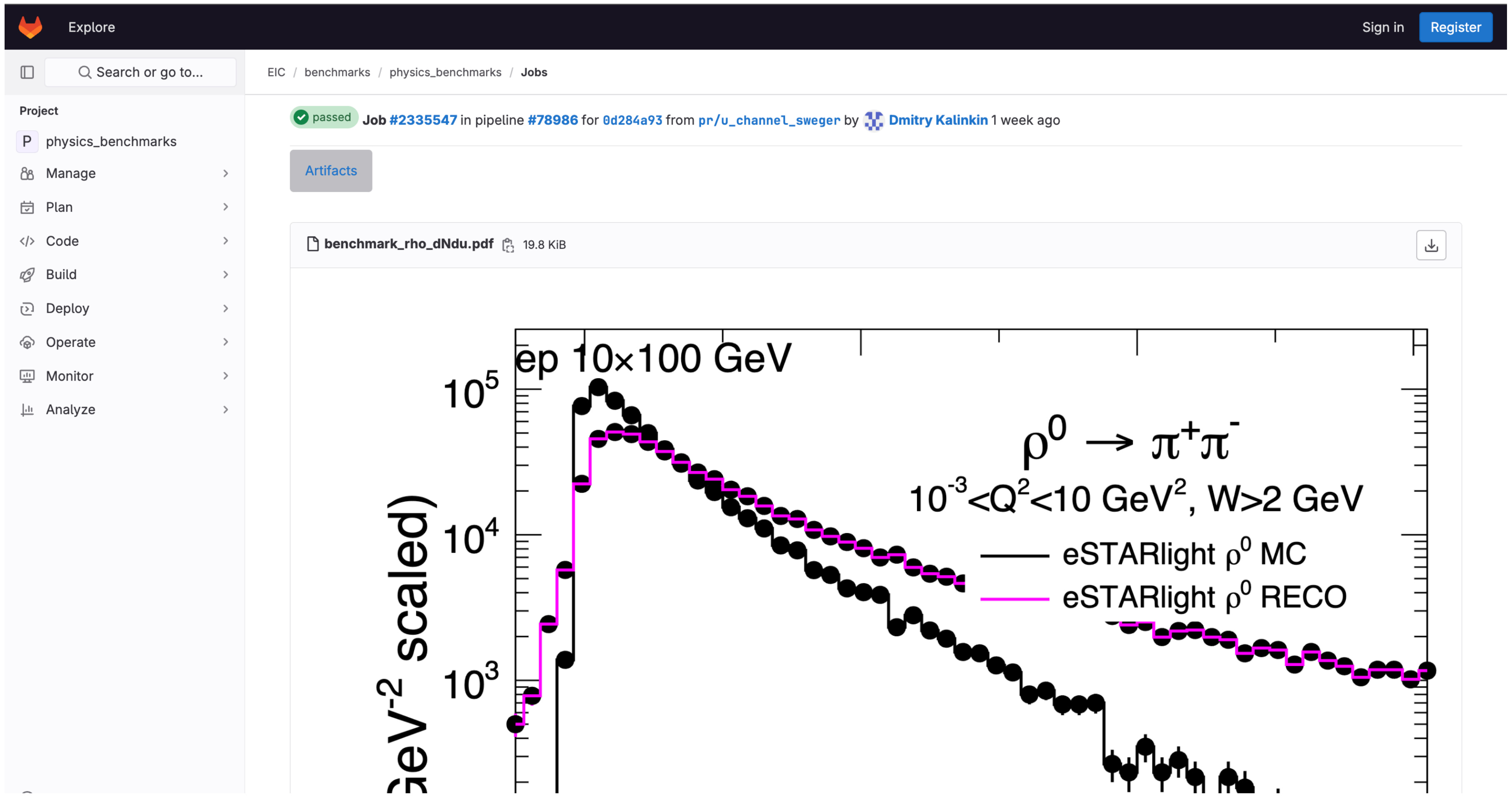
- `Snakefile` contains sets of rules like grabbing reco files, running the analysis, and hadding the output
- Whichever rule you want to execute, run
  `snakemake –cores 2 OUTPUTFILE`
- Where OUTPUTFILE is one of these file names
- Snakemake figures out which rule to run in order to produce that file
- It also runs any other rules which produce outputs that this rule requires as input
- So here if you include a line like
  `snakemake –cores 2 combined_analysis.root`
- snakemake will take this to mean you want to run the uchannelrho_combine rule, but it will first see that it has to run the uchannelrho_analysis rule first in order to have files to combine



```
90 benchmarks/u_rho/Snakefile
+
+ rule uchannelrho_analysis:
+     input:
+         script="benchmarks/u_rho/analysis/uchannelrho.cxx",
+         #script_compiled=ROOT_BUILD_DIR_PREFIX + "benchmarks/u_rho/
+         data="benchmark_output/campaign_23.12.0_rho_10x100_uChannel
+     output:
+         plots="benchmark_output/campaign_23.12.0_{INDEX}_eicrecon.e
+     shell:
+         """
+ mkdir -p $(dirname "{output.plots}")
+ root -l -b -q '{input.script}+("{input.data}","{output.plots}")'
+ """
+
+
+ rule uchannelrho_combine:
+     input:
+         #lambda wildcards: [f"benchmark_output/campaign_23.12.0_{ix
+         lambda wildcards: expand(
+           "benchmark_output/campaign_23.12.0_{INDEX:04d}_eicrecon.
+           INDEX=range(int(wildcards.N)),
+         ),
+     wildcard_constraints:
+         N="\d+",
+     output:
+         "benchmark_output/campaign_23.12.0_combined_{N}files_eicrec
+     shell:
+         """
+ hadd {output} {input}
+ """
+
```

Artifacts

- If everything goes well, the benchmark plots should be visible as artifacts: https://eicweb.phy.anl.gov/EIC/benchmarks/physics_benchmarks/-/artifacts



EIC / benchmarks / physics_benchmarks / Jobs

passed Job #2335547 in pipeline #78986 for 0d284a93 from pr/u_channel_sweger by Dmitry Kalinkin 1 week ago

benchmark_rho_dNdu.pdf 19.8 KiB

To Do

- Continue to investigate rho reconstruction quality
- In addition to finalizing the analysis plots, this benchmark is not complete until
  - ❑ It re-runs Geant simulations when detectors are updated
  - ❑ It re-runs reconstruction when algorithms are updated
  - ❑ Gives a failure or success statement based on output plots

Backward rho benchmark:

https://github.com/eic/physics_benchmarks/pull/3

Thank you for your attention!

zwsweger@ucdavis.edu