Datacards for ePIC

ePIC User Learning Workshop: Discoverability and Reusability for the pre-TDR and Beyond

Torri Jeske

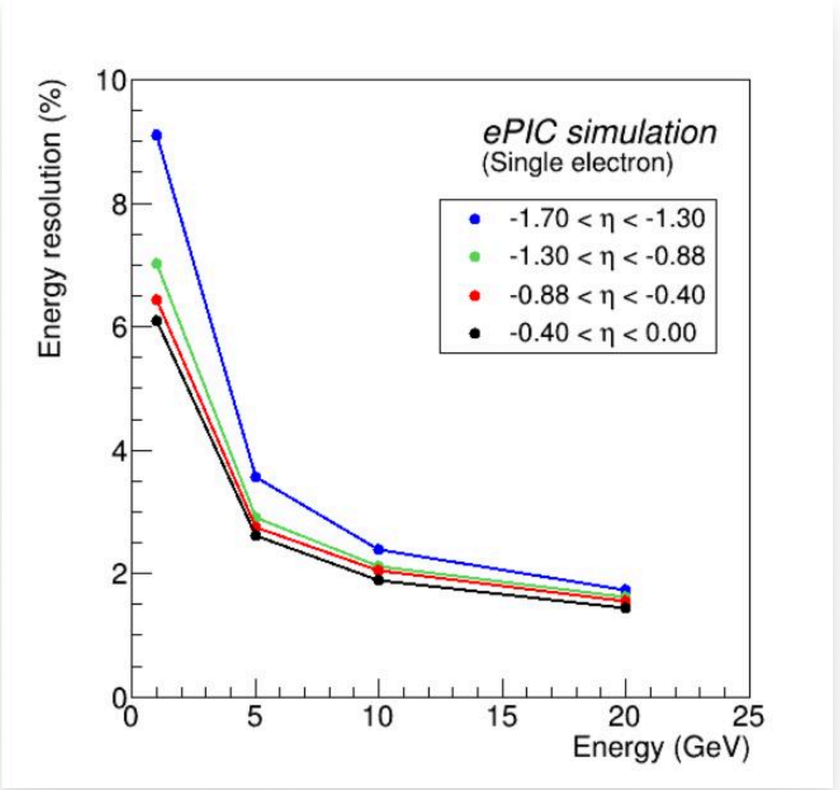
U.S. DEPARTMENT of ENERGY

Reproducibility

TDR plots

ePIC currently generates image-based data used for QA analysis, pre-TDR studies, etc

Storing macros in a repository is a good start, but is not completely sufficient to reproduce a given plot

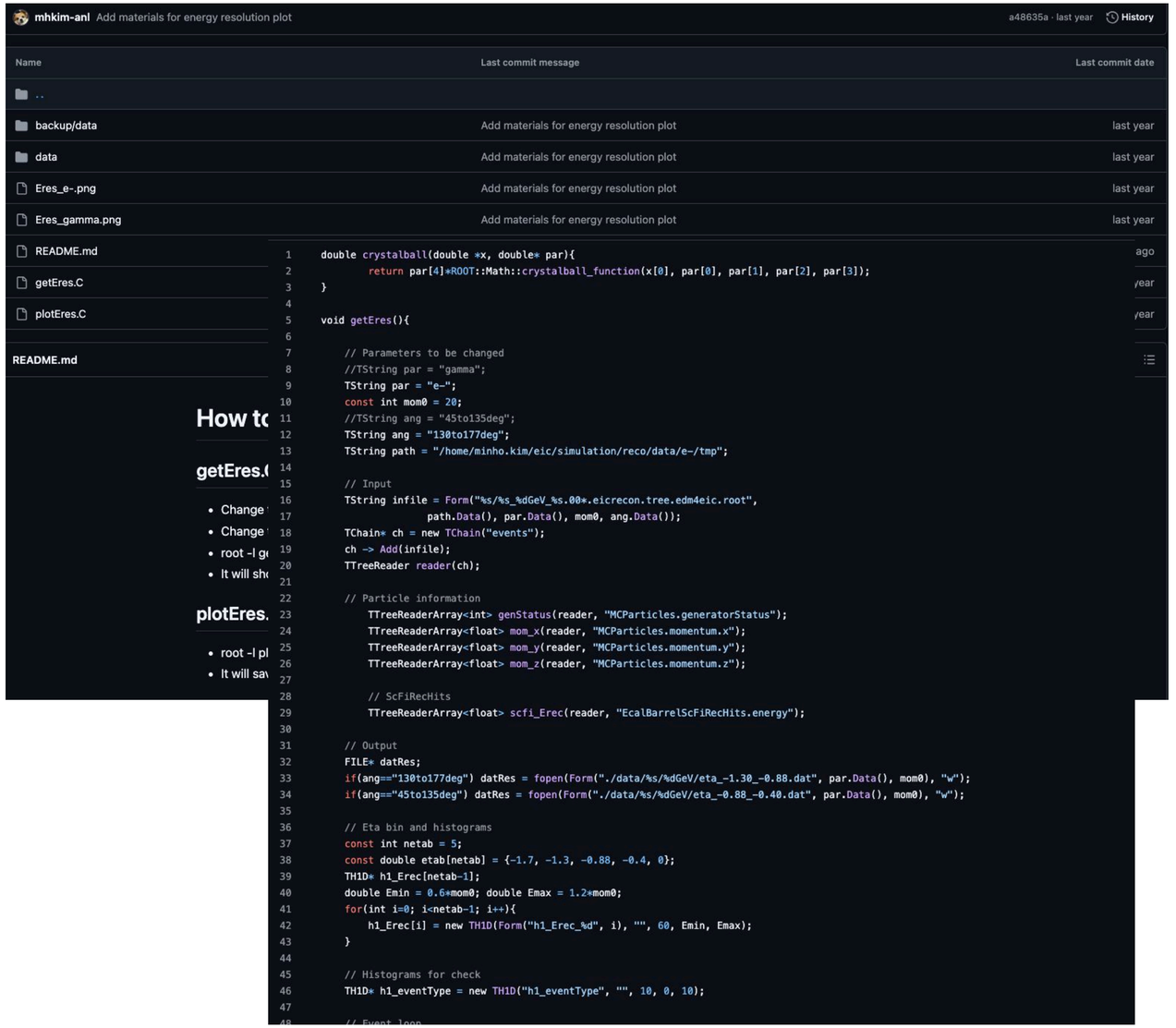


Reproducibility

BIC TDR Example

BIC Repository contains ROOT macros to generate plots incorporated into the TDR but:

- References directories on local user machine
- Doesn’t indicate what campaign it is valid for
- Doesn’t directly indicate which particular data set inside a given campaign is used
- Minho had to put up with about 8 different emails from me to regenerate 👎

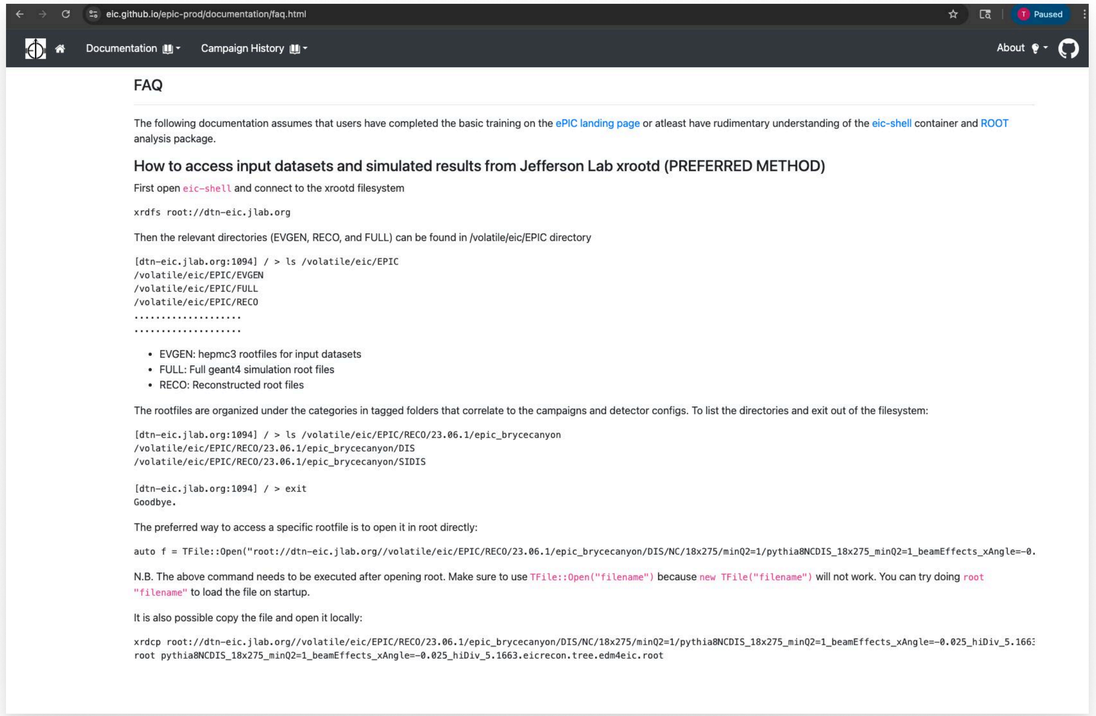


Reproducibility

Reproducing BIC plots

I have ROOT macro from GitHub, now:

- How do I access the simulation campaigns from JLab?
- Production Working Group maintains an up-to-date web page with a lot of useful information (if you know where to find it!)



Reproducing BIC plots

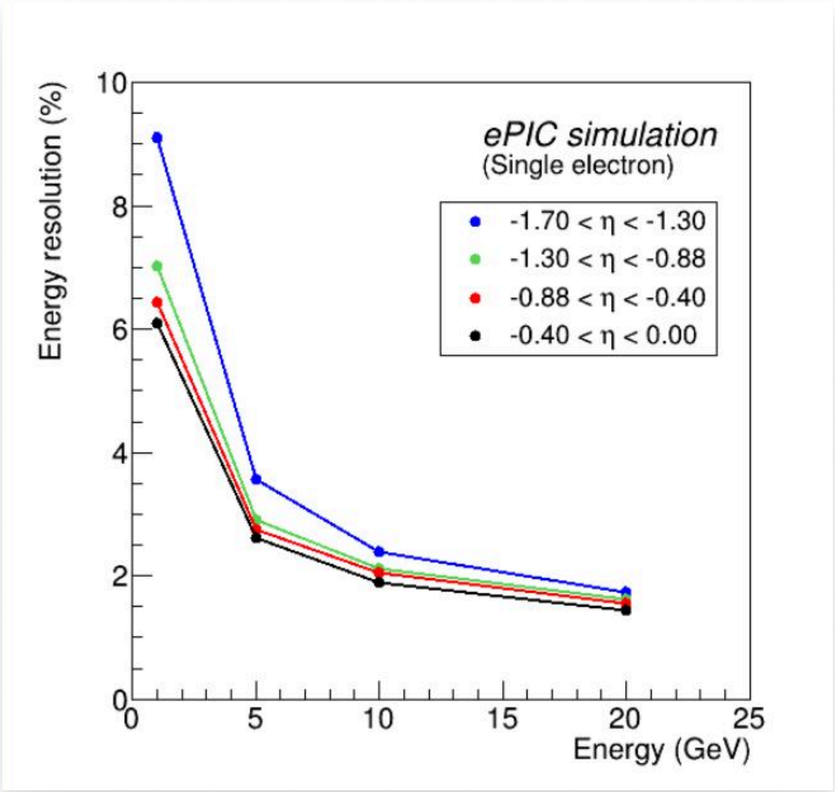
I have ROOT macro from GitHub, and the necessary campaign files now:

- What environment do I use (eic-shell)
- Where can I run this macro to produce the plots? (JLab, BNL, local machine)
- What I did: wrote a very basic script to submit jobs to JLab farm (clearly this doesn’t scale)
- What we would like: Use a Snakemake workflow to be consistent with benchmarks

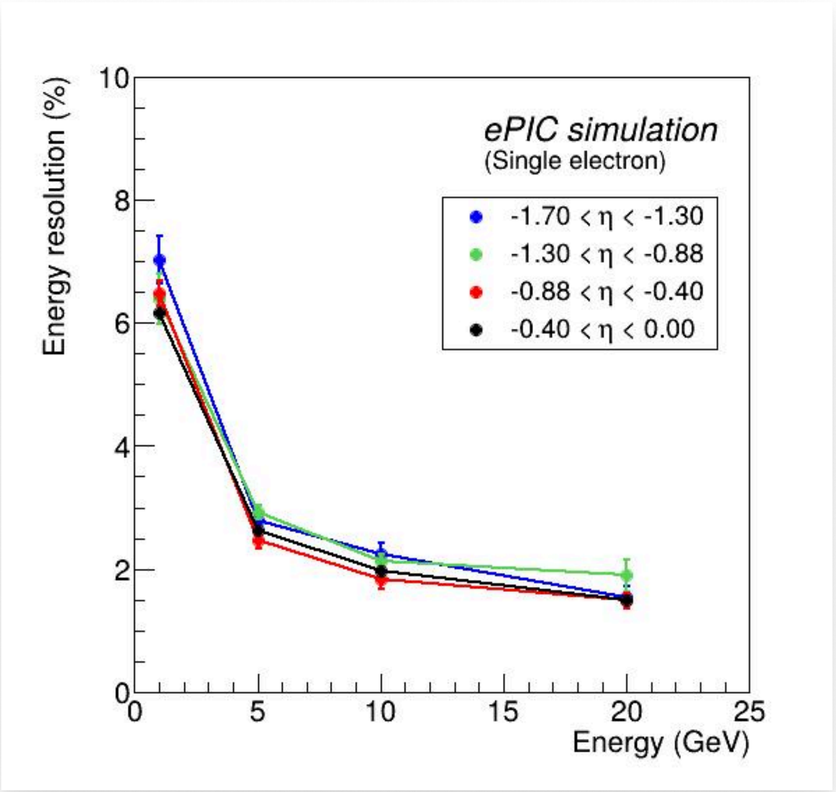
```
for CAMPAIGN in $CAMPAIGNS; do
    # Validate explicitly provided campaigns.
    if [ $# -gt 0 ]; then
        echo "Verifying campaign $CAMPAIGN..."
        if ! check_campaign_data "$CAMPAIGN"; then
            echo "Skipping invalid campaign: $CAMPAIGN"
            echo ""
            continue
        fi
    fi

    echo "Submitting EnergyResolution for campaign $CAMPAIGN..."
    sbatch \
        --job-name=er_${CAMPAIGN} \
        --output=$LOGDIR/er_${CAMPAIGN}_%j.out \
        --error=$LOGDIR/er_${CAMPAIGN}_%j.err \
        --time=02:00:00 \
        --partition=production \
        --mem=8000 \
        --gres=disk:1G \
        --account=eic \
        --export=ALL,CAMPAIGN=$CAMPAIGN,DETECTOR=$DETECTOR \
        --wrap="/volatile/eic/roark/epic-tdr-bic/run_analysis.sh EnergyResolution $CAMPAIGN $DETECTOR"
    echo ""
done
```

Do they actually match?



From BIC TDR repo



What I generated

I haven't had time to figure out where I messed up but that is coming!

Caution!

This is absolutely a work in progress that Diana and I just started last week. Seriously. *Work in progress*.

Genesis mission datacards have been under development for much longer.

Datacards

“Let’s just do it” -- Torri’s famous last words

Data Cards

What’s a datacard?

A structured metadata document that describes a dataset -- what it is, where it came from, who created it, how it can be accessed, and how it can be used.

Data cards serve both humans (who need to understand a dataset before using it) and machines (automated pipelines that ingest, catalog, and validate datasets).

There is no expectation that the ePIC community should already be familiar with datacards. This requires us to think about the data we produce in a way we most likely haven’t before.

Datacard Profiles

Profile	When to use	Time to complete
core	In-workflow, draft, or simple datasets not yet being shared	10-15 minutes
extended	Datasets being shared with partners or submitted to OSTI, Zenodo, or similar	30-45 minutes
ai_ready	Datasets intended for AI/ML training, inference, or evaluation	45-60 minutes
sensitive	CUI, export-controlled, PII, or classified datasets	45-60 minutes

Field Annotations

Annotation	Meaning
core	Required for all profiles
extended	Required for extended, ai_ready, and sensitive profiles
ai_ready	Required for ai_ready profile only
sensitive	Required for the sensitive profile only
pub	Required for when release_status=approved or published
if_applicable	Fill in if it applies to your dataset; else skip
system	Do not fill in -- the repository system populates this at ingest

Datacards

Datacard Metadata

Describes the datacard itself, *NOT* the dataset.

Version controlled with patch, minor, and major. For each update, add corresponding entry to change_log

Profile determines the completeness level you are committing to for this datacard.
For right now, we are completing [core].

```yaml
datacard:
  template_version: "1.0"
  datacard_version: "1.0"

  profile: core
  filename: "genesis_datacard_epic_simulation_campaign_march_2026.md"
  language: en

  id:
    type: __TYPE__
    value: __VALUE__

  sensitivity_tier: tier0_open
  access_level: open

  creation_method: manual
  created_date: "${2026-04-24}"
  updated_date: "${YYYY-MM-DD}"

  change_log:
    - date: "${YYYY-MM-DD}"
      datacard_version: "1.0"
      summary: "Initial creation"

  created_by:
    - role: editor
      date: "2026-04-24"
      description: "Generated first version of datacard"
      creator:
        type: person
        person:
          given_name: Torri
          family_name: Jeske
          orcid: 0000-0002-5191-0501
          email: roark@jlab.org
          affiliation:
            name: Thomas Jefferson National Accelerator Facility
            ror_id: https://ror.org/02vwzrd76
        organization:
```

Datacards

Level 1: Basic and Discoverable

Required for all datasets regardless of publication state or profile.

Define a single human-readable name for this dataset.

Define a project the data set belongs to, a version of the dataset

Identify a primary persistent identifier for this dataset → TO DO!

```yaml
identification:
  name: "ePIC simulation campaign root trees from March 2026"
  project: "epic"
  version: "1.0"

  primary_id:
    type: other
    value: 26.04


  additional_ids:
    - type: __TYPE__
      value: __VALUE__

  supersedes:

    type: __TYPE__
    value: __VALUE__

  superseded_by:

    type: __TYPE__
    value: __VALUE__

  parent_collection:
    name: __NAME__
    identifier:
      type: __TYPE__
      value: __VALUE__
```

Datacards

Description

Defines human-readable, high-level description of the data set, collection methodology, intended use, current use, out of scope use, limitations, and keywords.

```yaml
description:
  summary: "This dataset contains reconstructed outputs in the format of ROOT trees, produced in March of 2026. This dataset is intended for simulation studies requested by the physics working groups (PWGs) and DSCs. Each data set contains hepmc2 root files for input datasets, full geant4 simulation root files, and reconstructed root files."
  purpose: __PURPOSE__
  collection_methodology: computational simulation

  data_characteristics: __CHARACTERISTICS__

  intended_use: simulation studies

  current_use: __CURRENT_USE__

  out_of_scope_use: __OUT_OF_SCOPE__

  limitations: __LIMITATIONS__

  keywords:
    - simulation, ePIC, campaign

# --- Object & Dataset Type ------------------------------------
object_type: dataset
dataset_type: ND
# --- Release Status -------------------------------------------
release_status: approved
# --- Workflow & Lifecycle -------------------------------------

  state: raw

  is_intermediate: True

  pipeline_stage: "post-reconstruction"
```

Datacards

Dataset Readiness

An assessment of the dataset’s usability and interoperability level against the Genesis Dataset Readiness Model. Readiness describes usability, not scientific quality or value.

As of right now, ePIC simulation data is categorized as “discoverable”

```yaml
dataset_readiness:
  level: 1
  evaluated_against: __MODEL__
  evaluated_at: __YYYY-MM-DD__
  evaluated_by:

    type: __TYPE__
    person:
      given_name: __GIVEN_NAME__
      family_name: __FAMILY_NAME__
      orcid: __ORCID__
      email: __EMAIL__
      affiliation:
        name: __ORG_NAME__
        ror_id: __ROR_ID__
    organization:
      name: __ORG_NAME__
      ror_id: __ROR_ID__
  confidence: __CONF__
```

Access Policy

Describes who can access the dataset and under what conditions.

Indicate the sensitivity tier for access control purposes, the access level required to obtain the dataset, and the access policy for the specific type of authorization required to access this dataset.

```yaml
access_policy:
  sensitivity_tier: open
  access_level: open
  access_restrictions: None - publicly accessible

  authorization_required: none
  policy_url: __URL__
  policy_text: __TEXT__
```

Datacards

Contacts

Used to establish the primary point of contact for questions about this dataset. Required for all profiles -- every dataset must have a reachable contact person or organization.

Defaulted to Production Working Group as an organizational contact

```
# --- Contacts ----------------------------------------------------
contact:
  type: organization
  person:
    given_name: ${GIVEN_NAME}
    family_name: ${FAMILY_NAME}
    orcid: __ORCID__
    email: ${EMAIL}
    affiliation:
      name: ${ORG_NAME}
      ror_id: __ROR_ID__
  organization:
    name: Production WOrking Group
    ror_id: __ROR_ID__
  valid_until: __YYYY-MM-DD__
  succession_note: __NOTE__
additional_contacts: []
```

Authorship and Credits

Establish authorship via person or organization, can provide list of contributors.

```
# --- Authorship & Credit ---------------------------------------

authors:
  - type: ${TYPE}
    person:
      given_name: ${GIVEN_NAME}
      family_name: ${FAMILY_NAME}
      orcid: __ORCID__
      email: __EMAIL__
      affiliation:
        name: ${ORG_NAME}
        ror_id: __ROR_ID__
    organization:
      name: __ORG_NAME__
      ror_id: __ROR_ID__
    role: ${ROLE}
contributors: []
```

Categorization

Indicate the high-level scientific domain or discipline associated with the dataset, structured tags for catalog filtering, the object type, and risk level.

In the future, for ai_ready datacards, indicate task category (e.g., classification) and relevant subcategories (e.g., binary classification, multi_label_classification)

```yaml
# --- Categorization ------------------------------------------------
categorization:
  science_domain: "physics"

  tags:
    project: epic
    science: physics
    type: dataset
    risk: general
  task_category: []
  task_subcategory: []
```

Dataset Characteristics

Define a number of characteristics about your dataset.

```yaml
# --- Dates ----------------------------------------------------
dates:
  data_collection_start: __YYYY-MM-DD__
  data_collection_end: __YYYY-MM-DD__
  issued: __YYYY-MM-DD__
  modified: __YYYY-MM-DD__
```

```yaml
# --- Dataset Characteristics ----------------------------------
dataset_info:
  formats: ["ROOT"]
  encoding: __ENCODING__
  schema_version: __VERSION__

  modalities: ["tree"]

  features: []
  splits: []
  language: __LANG__
  spatial_coverage:
    description: __DESC__
    bounding_box:
      west: __DECIMAL_DEG__
      east: __DECIMAL_DEG__
      south: __DECIMAL_DEG__
      north: __DECIMAL_DEG__
  temporal_coverage:

    start: __YYYY-MM-DD__
    end: __YYYY-MM-DD__
    description: __DESC__

dataset_scale:
  record_count: __COUNT__
  record_unit: __UNIT__
  compressed_bytes: __BYTES__
  uncompressed_bytes: __BYTES__
```

```yaml
# --- Access Endpoints -----------------------------------------
--

access:
  current_location: "dtn-eic.jlab.org//volatile/eic/EPIC/"
  intended_repositories:
    - name: __NAME__
      access_level: __LEVEL__
      is_primary: __BOOL__
      date_deposited: __YYYY-MM-DD__
      api:
        endpoint: __URL__
        documentation_url: __URL__
        authentication: __AUTH__
        version: __VERSION__
        rate_limit: __LIMIT__
```

Datacards

Provenance

Describes how this dataset was created, what it was derived from, and what processing was applied.

Clear provenance sections in the datacards can significantly aid reproducibility

This section could capture a lot of the relevant information for Snakemake

```yaml
# --- Provenance ----------------------------------------------

provenance:
  was_generated_by: "ePIC production and simulation campaign"
  source_data:
    - name: __NAME__
      identifier:
        type: __TYPE__
        value: __VALUE__
      relationship: __REL__
  processing_steps: __DESCRIPTION__
  instrumentation: __DESCRIPTION__
  simulation_details: __DESC__
  software_environment:
    os: __OS__
    compiler: __COMPILER__
    container: __CONTAINER__
    hpc_environment: __ENV__
    notes: __NOTES__
```

Campaign-level datacard summary

Pick a profile

As a first attempt, I decided to go with [core]

Fill out datacards with [core] annotations

Remember the distinction between placeholder conventions (blank → information not yet known!)

Temporarily store datacards in eic/datacards

This is a *private* repository for now while these cards are under active development

Datacards → Snakemake workflows

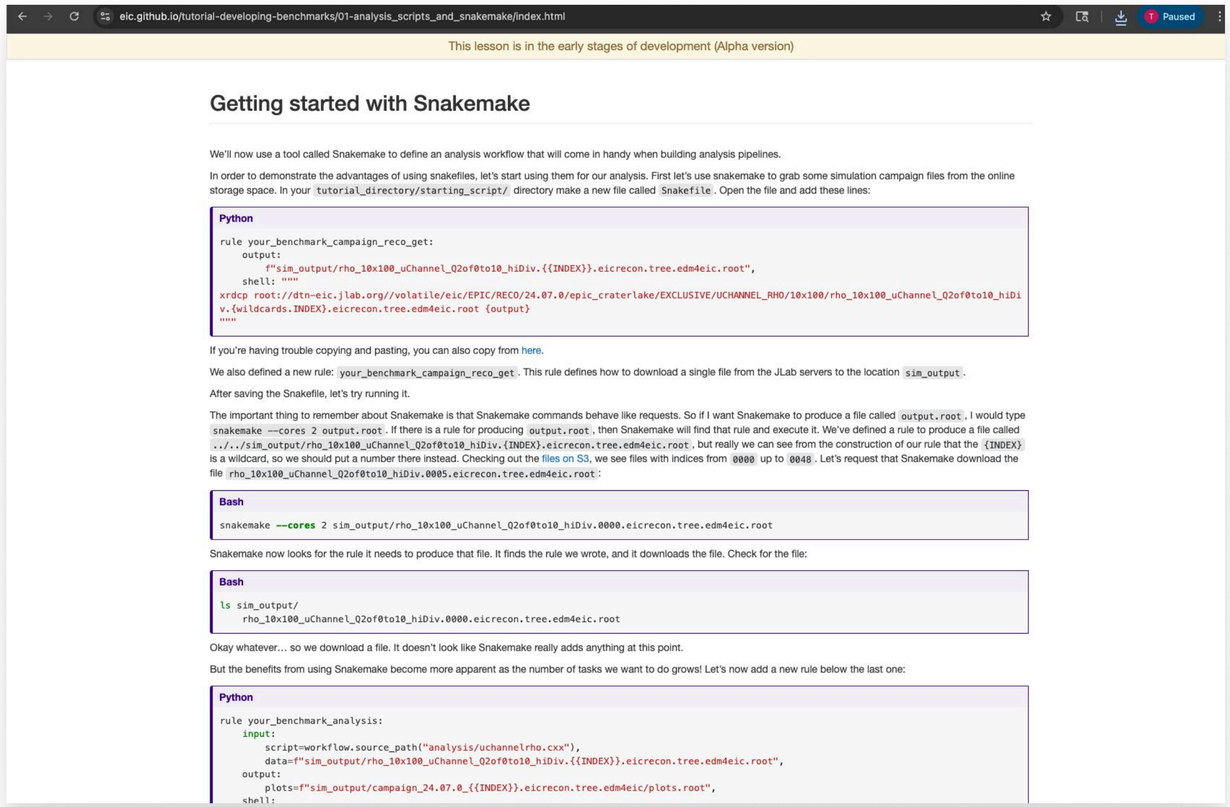
How can we use datacards and tools to generate Snakemake workflows that reproduce QA plots?

Important context: Most ePIC users are probably not familiar with Snakemake

What is Snakemake?

Snakemake is an open-source workflow management system used to create reproducible and scalable data analysis pipelines. Users define “rules” that specify input files, output files, and the command(s) to generate them.

Dmitry Kalinkin led the development of Snakemake workflows for physics and detector benchmark plots.

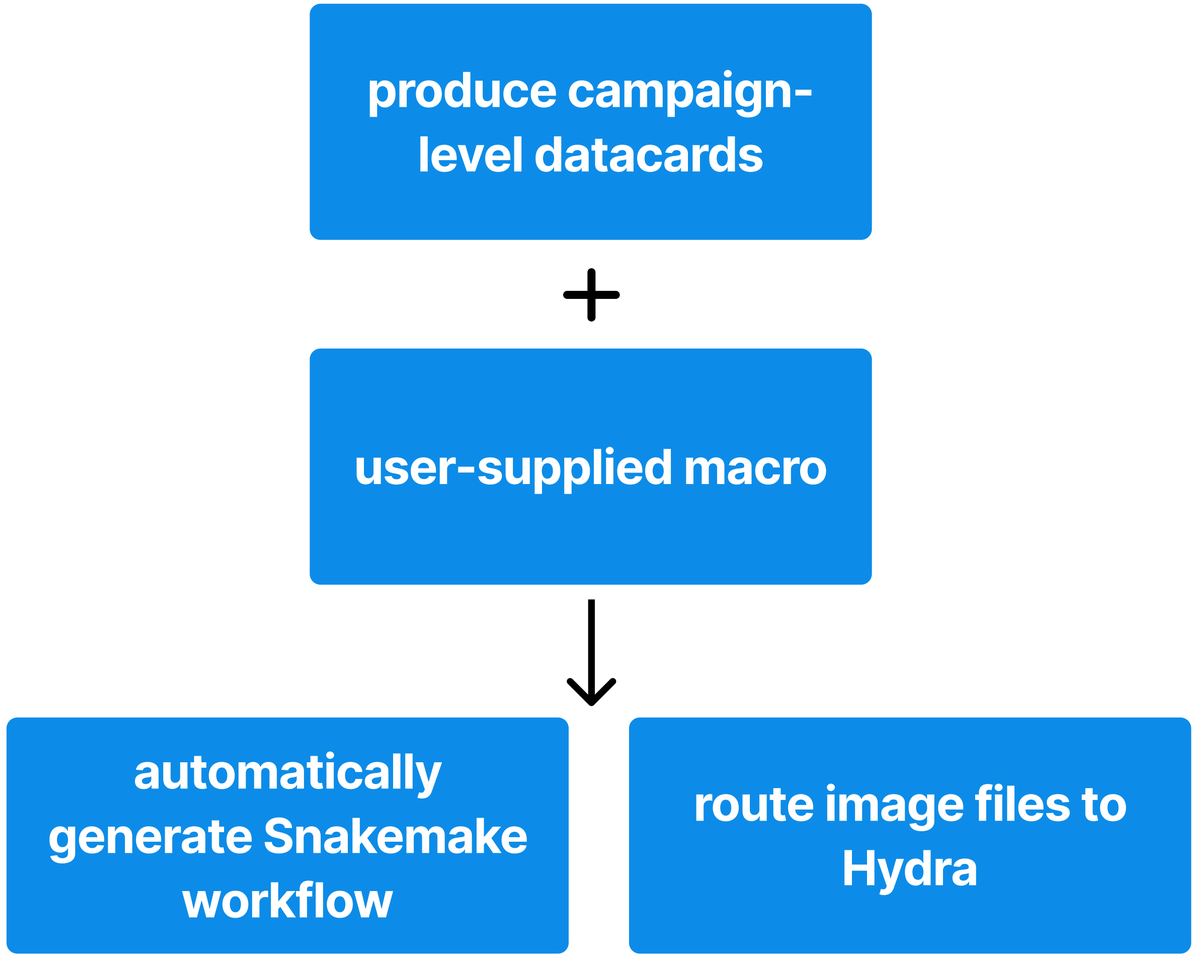


Datacards → Snakemake

Snakefile generation

With modifications, datacards could be used to automatically generate Snakefiles to aid in production of QA plots.

There is ongoing work in Genesis to develop agents with tools and skills that can interpret and generate workflows from datacards



Conclusions + Future Outlook + Action Items

Datacards

Structured metadata document that describes a dataset
Can be generated for campaign data as well as corresponding benchmark/QA plots

Datacard → Snakemake workflows

The ePIC community should have access to tools that can generate Snakemake workflows on their behalf using datacards

Input and collaboration

Datacards can help point out where ePIC production and analysis workflows are insufficient in capturing important aspects of the data, ePIC datasets and workflows can inform structure of datacards