A Deep Dive into PocketCoffea

PyHEP.dev 2024

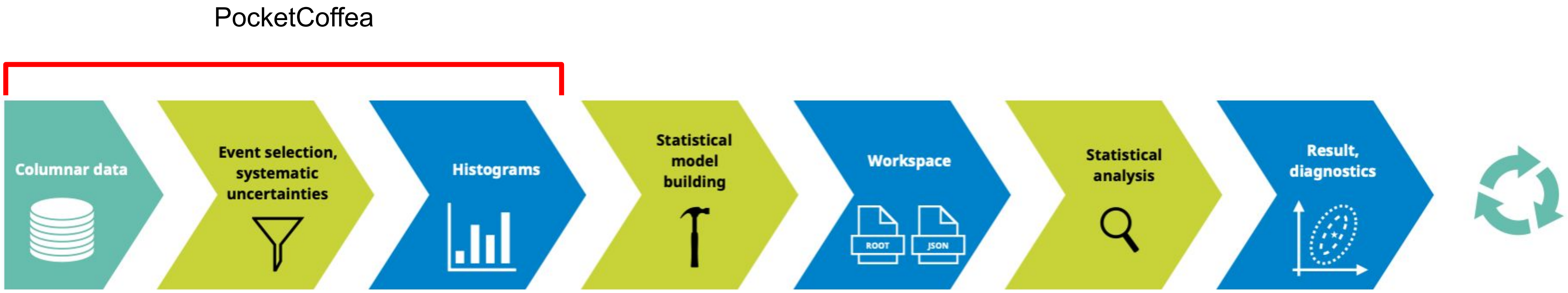
"Python in HEP" Developer's Workshop

Máté Zoltán Farkas

Physics Institute III | RWTH AACHEN UNIVERSITY

- What is PocketCoffea?
  - By Davide Valsecchi, Matteo Marchegiani
  - Framework for processing CMS NanoAODs
  - Based on Coffea v0.7.22

- Automatized workflow from dataset querying to histogramming



Adapted from [1]

What is PocketCoffea?

- It features…
  - Abstractions for your analysis workflow
  - Configuration files for phase space selection, corrections, triggers
    → Intuitive declaration of analysis configuration
  - Law pipeline from dataset preparation to plotting (new and undocumented)

- Where can you run it?
  - Lxplus, DESY NAF, Coffea Casa, RWTH T2 cluster etc.
    – freely extendable through executors
  - Container or local conda/mamba/micromamba environment

- https://pocketcoffea.readthedocs.io/en/stable/

The Pocket Coffea Workflow

- Default: python scripts or pocket-coffea commands
  - pocket-coffea build-datasets [args]
- New: law pipeline
  - Tasks wrap functionality of executor scripts
  - Rich CLI interface for parameter control

NEW

```
0 > Plotter(cfg=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/config.py, output_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_exam
│   _output=output.coffea, test=False, limit_files=None, limit_chunks=None, executor=iterative, scaleout=None, process_separately=False, plot_dir=plots, plot_verbose=0, plot_style=NO_STR, blind=True, plot_workers=4, plot_
│   _scale=False, plot_format=pdf, variables=)
│     LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/plots/blind/lin/.plots_done)
│       absent
│
└──1 > Runner(cfg=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/config.py, output_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output, coff
    │   ea_output=output.coffea, test=False, limit_files=None, limit_chunks=None, executor=iterative, scaleout=None, process_separately=False)
    │     coffea: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/output.coffea)
    │       absent
    │     parameters: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/parameters_dump.yaml)
    │       absent
    │     config: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/config.json)
    │       existent
    │     configurator: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/configurator.pkl)
    │       existent
    │
    ├──2 > CreateDatasets(dataset_definition=datasets/definition_*.json, dataset_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets, keys=[], download=False, overwrite=False, check=Fa
    │       lse, split_by_year=False, local_prefix=, allowlist_sites=[], blocklist_sites=[], regex_sites=T[123]_(FR|IT|DE|BE|CH|UK)_\w+, parallelize=4)
    │         0: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DATA_SingleMuon.json)
    │           existent
    │         1: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DATA_SingleMuon_redirector.json)
    │           existent
    │         2: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DYJetsToLL_M-50.json)
    │           existent
    │         3: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DYJetsToLL_M-50_redirector.json)
    │           existent
    │
    └──2 > JetCalibration(cfg=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/config.py, output_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/o
            utput)
              LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/jets_calibrator_JES_JER_Syst.pkl.gz)
                existent
```

The Pocket Coffea Workflow – law

Physics Institute III
RWTHAACHEN UNIVERSITY
Máté Zoltán Farkas, 29.08.2024

- Dependency management

  →task execution if output missing

Download datasets?

Run it on a small sample

Which workflow?

Submit jobs or run locally?

NEW

```
0 > Plotter(cfg=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/config.py, output_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_exam
    _output=output.coffea, test=False, limit_files=None, limit_chunks=None, executor=iterative, scaleout=None, process_separately=False, plot_dir=plots, plot_verbose=0, plot_style=NO_STR, blind=True, plot_workers=4, plot_
    _scale=False, plot_format=pdf, variables=)
      LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/plots/blind/lin/.plots_done)
        absent
│
└──1 > Runner(cfg=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/config.py, output_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output, coff
       ea_output=output.coffea, test=False, limit_files=None, limit_chunks=None, executor=iterative, scaleout=None, process_separately=False)
         coffea: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/output.coffea)
           absent
         parameters: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/parameters_dump.yaml)
           absent
         config: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/config.json)
           existent
         configurator: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/output/configurator.pkl)
           existent
   │
   ├──2 > CreateDatasets(dataset_definition=datasets/definition_*.json, dataset_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets, keys=[], download=False, overwrite=False, check=Fa
   │      lse, split_by_year=False, local_prefix=, allowlist_sites=[], blocklist_sites=[], regex_sites=T[123]_(FR|IT|DE|BE|CH|UK)_\w+, parallelize=4)
   │        0: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DATA_SingleMuon.json)
   │          existent
   │        1: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DATA_SingleMuon_redirector.json)
   │          existent
   │        2: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DYJetsToLL_M-50.json)
   │          existent
   │        3: LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/datasets/DYJetsToLL_M-50_redirector.json)
   │          existent
   │
   └──2 > JetCalibration(cfg=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/config.py, output_dir=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/o
          utput)
            LocalFileTarget(fs=local_fs, path=/.automount/net_rw/net__scratch_cms3a/farkas/analyse/felixconfigs/configs/zmumu/law_example/jets_calibrator_JES_JER_Syst.pkl.gz)
              existent
```

Dataset Querying

Physics Institute III
RWTHAACHEN UNIVERSITY
Máté Zoltán Farkas, 29.08.2024

- Dataset querying via configuration .json
  - human-readable and customizable format
  - Entries include generic information
- CreateDataset Task produces framework-ready json

Sample name

Various metadata

Sample

{} datasets_definitions.json ×

AnalysisConfigsPocketCoffea > configs > zmumu > datasets > {} datasets_definitions.json > {} DATA_SingleMuon > [ ] files > {} 1

```
    "DYJetsToLL_M-50":{
        "sample": "DYJetsToLL",
        "json_output"    : "datasets/DYJetsToLL_M-50.json",
        "files":[
            { "das_names": ["/DYJetsToLL_M-50_TuneCP5_13TeV-amcatnloFXFX-pythia8/RunIISummer20UL18NanoAODv9-106X_upgrade2018_realistic_v16_L1v1-v2/NANOAODSIM"],
              "metadata": {
                  "year":"2018",
                  "isMC": true,
          "xsec": 6077.22
              }
            }
        ]
    },


    "DATA_SingleMuon": {
        "sample": "DATA_SingleMuon",
        "json_output": "datasets/DATA_SingleMuon.json",
```

Dataset Processing

Physics Institute III
RWTH AACHEN UNIVERSITY
Máté Zoltán Farkas, 29.08.2024

- Runner Task:
  - Calls the coffea processor
  - Processing controlled by Configurator object

- Configurator:
  - Provides abstraction for commonly used parameters
  - Datasets, year, corrections, categories, variables to store and export

```python
cfg = Configurator(
    parameters = parameters,
    datasets = {
        "jsons": [f"{localdir}/datasets/DATA_SingleMuon.json",
                  f"{localdir}/datasets/DYJetsToLL_M-50.json"
                  ],
        "filter" : {
            "samples": ["DATA_SingleMuon",
                        "DYJetsToLL"],
            "samples_exclude" : [],
            "year": ['2018']
        }
    },

    workflow = ZmumuBaseProcessor,

    skim = [get_nObj_min(1, 18., "Muon"),
            # Asking only SingleMuon triggers since we are only using SingleMuon PD data
            get_HLTsel(primaryDatasets=["SingleMuon"])],

    preselections = [dimuon_presel],
    categories = {
        "baseline": [passthrough],
    },

    weights = {
        "common": {
            "inclusive": ["genWeight","lumi","XS",
                          "pileup"
```

Declarative Parameter Definitions

- `yaml` configuration files
  - Phase-space region definition
  - Triggers
  - Plotting styles
  - Custom run options
  - Easy to read format
  - …and redefinable in the coffea processor workflow!

object_preselection.yaml

AnalysisConfigsPocketCoffea > configs > zmumu >

Davide Valsecchi, 4 months ago | 2 authors (

```yaml
object_preselection:
  Muon:
    pt: 15
    eta: 2.4
    iso: 0.25  #PFIsoLoose
    id: tightId

  Electron:
    pt: 15
    eta: 2.4
    iso: 0.06
    id: mvaFall17V2Iso_WP80

  Jet:
    dr_lepton: 0.4
    pt: 30
    eta: 2.4
    jetId: 2
    puId:
      wp: L
      value: 4
      maxpt: 50.0
    btag:
      wp: M
```

The Runner & Processor

- Runner:
  - Calls executor to execute coffea
- BaseProcessorABC:
  - Abstract class inheriting coffea's ProcessorABC
  - Implements processing steps
  - Takes care of corrections, histogramming
  - Provides overrides for the analysis workflow
    - 2 abstract methods
  - Takes care of variable storage
- One workflow per analysis

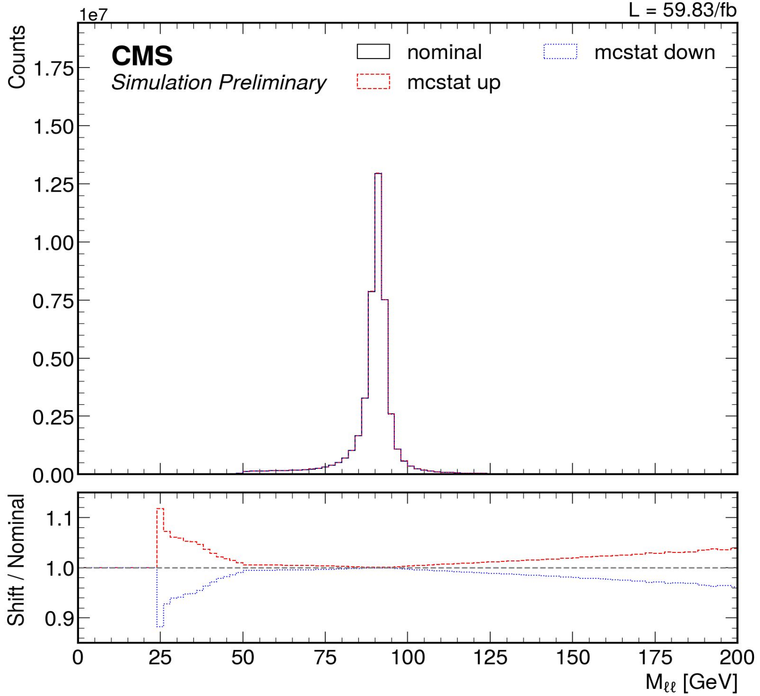
workflow.py 5 ×

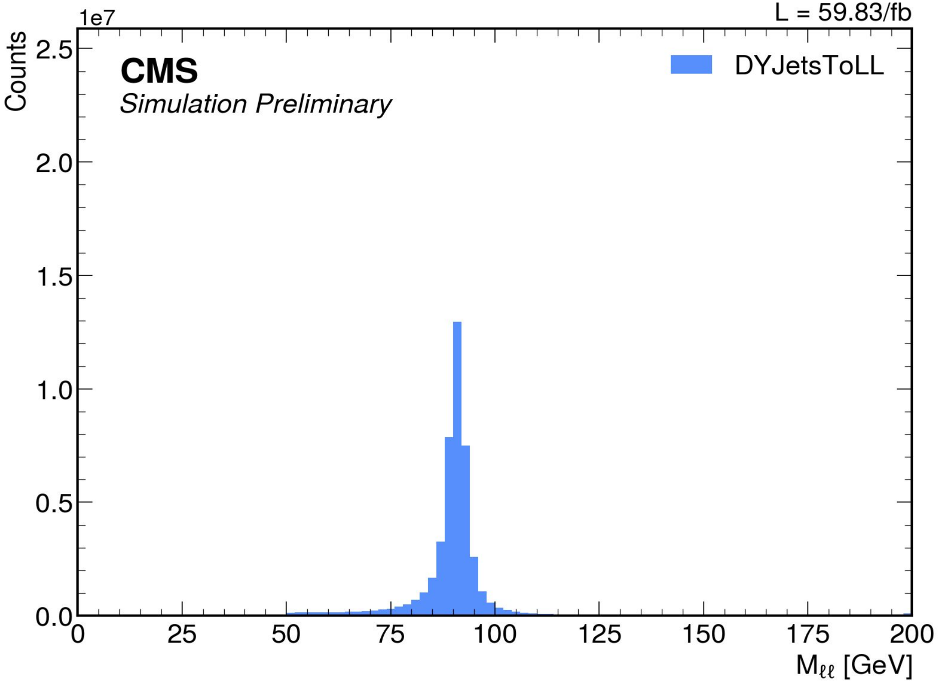
AnalysisConfigsPocketCoffea > configs > zmumu > workflow.py > ...

```python
class ZmumuBaseProcessor(BaseProcessorABC):
    def __init__(self, cfg: Configurator):
        super().__init__(cfg)


    def apply_object_preselection(self, variation):
        '''...
        # Include the supercluster pseudorapidity variable
        electron_etaSC = self.events.Electron.eta + self.events.Electron.del
        self.events["Electron"] = ak.with_field(...
        # Build masks for selection of muons, electrons, jets, fatjets
        self.events["MuonGood"] = lepton_selection(...
        self.events["ElectronGood"] = lepton_selection(
            self.events, "Electron", self.params
        )
        leptons = ak.with_name(...
        self.events["LeptonGood"] = leptons[ak.argsort(leptons.pt, ascending

        self.events["JetGood"], self.jetGoodMask = jet_selection(
            self.events, "Jet", self.params, "LeptonGood"
        )
        self.events["BJetGood"] = btagging(...

        self.events["ll"] = get_dilepton(...

    def count_objects(self, variation):
        self.events["nMuonGood"] = ak.num(self.events.MuonGood)
        self.events["nElectronGood"] = ak.num(self.events.ElectronGood)
        self.events["nLeptonGood"] = (
            self.events["nMuonGood"] + self.events["nElectronGood"]
        )
        self.events["nJetGood"] = ak.num(self.events.JetGood)
        self.events["nBJetGood"] = ak.num(self.events.BJetGood)
        # self.events["nfatjet"]   = ak.num(self.events.FatJetGood)

    # Function that defines common variables employed in analyses and save t
    def define_common_variables_before_presel(self, variation):
        self.events["JetGood_Ht"] = ak.sum(abs(self.events.JetGood.pt), axis
```

Plotsystematics & Plotter

- matplotlib & mplhep backend
- Default style: CMS standard
- Support for systematic uncertainty shifts (new and undocumented)





What's not there/coming

Máté Zoltán Farkas, 29.08.2024

- B-hive integration/ML extension
  - Support for ML workflows (training and inference)
- Statistical model building:
  - Datacard creation
  - Statistical inference with evermore
- Development on newer coffea version support
- …

PocketCoffea