L1 Trigger Offline Analysis Demonstration

Ben Krikler Luke Kreczko

4th July 2017

Goals

- Understanding of cms-l1t-analysis
- Able to use and modify
- Identify where we need to improve it
- Develop some uses together:
  - From Alex Tapper: One nice example you could get going on would be to cross check plots like these:
  - https://indico.cern.ch/event/650692/contributions/2647648/attachments/1486851/2309370/Run2017B.pdf

Ben Krikler: b.krikler@cern.ch

Overview

- Installing
- Using
  - Out-of-the-box
  - Changing the config file
- Developing / adding things
  - Analyzers
  - New plot types
- Q&A, discussions
- Coding session

Some things quite fresh, others unpolished

Ben Krikler:
b.krikler@cern.ch

Getting Started

Downloading and Installing

Installing:

```
$ git clone -o upstream \
    https://github.com/cms-l1t-offline/cms-l1t-analysis.git
$ cd cms-l1t-analysis
$ source bin/env.sh
$ voms-proxy-init --voms cms
$ make setup
```

At the start of each session:

```
$ source bin/env.sh
```


Ben Krikler:
b.krikler@cern.ch

Basic Usage

Single command as main point of entry:

```
$ cmsl1t
```

Needs the name of a config file, eg:

```
config/demo.yaml
```

Typical command:

```
$ cmsl1t -c config/demo.yaml
```

Some limited control from the command-line, but most interaction via the config file

Ben Krikler:
b.krikler@cern.ch

Command line options

```
$ cmsl1t --help

Usage: cmsl1t [OPTIONS]


Options:

  -c, --config_file FILENAME  YAML style config file  [required]

  -n, --nevents INTEGER       Number of events to process.

  -r, --reload-histograms     Reload histograms from a file and skip the input
                              tuples

  -v, --verbosity LVL         Either CRITICAL, ERROR, WARNING, INFO or DEBUG

  --help                      Show this message and exit.
```


Ben Krikler:
b.krikler@cern.ch

The Config File

- See the online tutorials:
  - http://cms-l1t-analysis.readthedocs.io/en/latest/tutorial/configuration.html
  - https://github.com/cms-l1t-offline/cms-l1t-analysis/blob/master/docs/tutorial/configuration.rst

Using the Documentation

- Documentation is online and automatically kept up-to-date with code:
  - http://cms-l1t-analysis.readthedocs.io/en/latest/index.html
  - Uses comments in the code as well as dedicated documentation
- Things are still being fleshed out
  - Would be good to identify most urgent areas today

Ben Krikler: b.krikler@cern.ch

Under the Hood

- Long-term hope to be able to hide much of the next few slides from the typical user
- Still early stages, so will probably still be important

Ben Krikler:
b.krikler@cern.ch

Inside an Analyzer

- Main tasks of an analyzer:
  - Pull out the variables of interest from the tree and pass to the plotters
  - Tell the plotters when to actually plot, save to file, reload from a file, etc
- Key methods
  - Prepare_for_events - setup histograms
  - Fill_histograms - pass event information to hists
  - Write_histograms - Save histograms to a file
  - Make_plots - draw all histograms

Ben Krikler:
b.krikler@cern.ch

Inside an Analyzer: study_met.py

```python
"""
Study the MET distibutions and various PUS schemes
"""

from BaseAnalyzer import BaseAnalyzer
from cmsl1t.plotting.efficiency import EfficiencyPlot
from functools import partial
import cmsl1t.recalc.met as recalc
import numpy as np


class Analyzer(BaseAnalyzer):
    def __init__(self, config, **kwargs):
        super(Analyzer, self).__init__('study_met', config)

        self.eff_caloMET_BE = EfficiencyPlot()
        self.add_plotter(self.eff_caloMET_BE)

        file_format = config.try_get('output', 'plot_format', 'png')
        for hist in self.all_plots:
            hist.set_plot_output_cfg(self.output_folder, file_format)

    def prepare_for_events(self, reader):
        # TODO: Get these from a common place, and / or the config file
        puBins = range(0, 50, 10) + [999]
        thresholds = [70, 90, 110]

        self.eff_caloMET_BE.build("CaloMETBE", "OfflineMETBE",
                                  "CaloMET BE (GeV)", "Offline MET BE (GeV)",
                                  puBins, thresholds, 50, 0, 300)
        return True

    def fill_histograms(self, entry, event):
        pileup = event.nVertex
        if pileup < 5 or not event.passesMETFilter():
            return True

        if len(event.caloTowers) <= 0:
            return True

        offlineMetBE = event.sums.caloMetBE
        onlineMet = recalc.l1MetNot28(event.caloTowers).mag

        self.eff_caloMET_BE.fill(pileup, offlineMetBE, onlineMet)

        return True
```

Call class **`Analyzer`** and derive from **`BaseAnalyzer`** class

Bookmark the plotters this analyzer uses. Use the method **`add_plotter`**

Prepare plotters for event data for when we run off NTuples

Pass event data through to plotters

Ben Krikler:
b.krikler@cern.ch

Accessing Event Data

- All known trees in NTuples are loaded
- Can access objects in the event by....
  - event.nVertex
  - event.caloTowers
  - event.emuCaloTowers
  - event.sums
- Some special objects eg. sums
- Modifiers (recalc)
  - Creates new event.<var> accessors

Ben Krikler:
b.krikler@cern.ch

Inside a Plotter

- Main tasks of a plotter
  - Maintain a list of histograms
  - Receive a set of values for each event and fill the contained histograms appropriately
  - Lay out the histograms on a canvas
  - Merge histograms from multiple input histogram files when asked
- Key methods of a plotter
  - fill - fill histograms with values passed from trees
  - Draw - turn histograms into plots
  - Build - Create the internal histograms etc
  - to_root, from_root - write and read histograms from a file
- Note: We're still converting from an intermediate approach, so not all existing plotters look the same

Ben Krikler: b.krikler@cern.ch

Inside a Plotter: efficiency.py

```python
from __future__ import print_function
from cmsllt.plotting.base import BasePlotter
from cmsllt.hist.hist_collection import HistogramCollection
from cmsllt.hist.factory import HistFactory
import cmsllt.hist.binning as bn
from cmsllt.utils.draw import draw, label_canvas
from cmsllt.utils.fit_efficiency import fit_efficiency
from cmsllt.io import to_root

from rootpy.plotting import Legend, HistStack
from rootpy.context import preserve_current_style


class EfficiencyPlot(BasePlotter):
    def build(self,
              online_name, offline_name,
              online_title, offline_title,
              pileup_bins, thresholds, n_bins, low, high):
        """ This is not in an init function so that we can by-pass this in the
        case where we reload things from disk """
        self.online_name = online_name
        self.offline_name = offline_name
        self.online_title = online_title
        self.offline_title = offline_title
        self.pileup_bins = bn.Sorted(pileup_bins, 'pileup',
                                     use_everything_bin=True)
        self.thresholds = bn.GreaterThan(thresholds, "threshold",
                                         use_everything_bin=True)

        name = [online_name, offline_name,
                "thresh_{threshold}", "pu_{pileup}"]
        name = "__".join(name)
        title = " ".join([online_name, " in PU bin: {pileup}",
                          "and passing threshold: {threshold}"])
        self.yields = HistogramCollection([self.pileup_bins, self.thresholds],
                                          "HistID", n_bins, low, high,
                                          name="yield" + name, title=title)
        self.filename_format = "{type}" + name

    def fill(self, pileup, online, offline):
        self.yields[pileup, online].fill(offline)

    def draw(self, with_fits=True):
        # Calculate the efficiency for each threshold
        self.__fill_efficiencies()
        if with_fits:
            self.__fit_efficiencies()

        # Overlay the "all" pile-up bin for each threshold
        all_pileup_effs = self.efficiencies.get_bin_contents([bn.Base.everything])
        hists = []
        labels = []
```

Prepare the binning and histograms

Creates binning objects: Online Thresholds, Pileup

Fill histograms with the given data from the event

Turn the histograms into plots

Ben Krikler:
b.krikler@cern.ch

Inside a Plotter: efficiency.py

```
43    def draw(self, with_fits=True):
44        # Calclate the efficiency for each threshold
45        self.__fill_efficiencies()
46        if with_fits:
47            self.__fit_efficiencies()
48
49        # Overlay the "all" pile-up bin for each threshold
50        all_pileup_effs = self.efficiencies.get_bin_contents([bn.Base.everything])
51        hists = []
52        labels = []
53        fits = []
54        for threshold in all_pileup_effs.iter_all():
55            if not isinstance(threshold, int):
56                continue
57            hists.append(all_pileup_effs.get_bin_contents(threshold))
58            labels.append("> " + str(self.thresholds.bins[threshold]))
59            if with_fits:
60                fits.append(self.fits.get_bin_contents([bn.Base.everything, threshold]))
61        self.__make_overlay("all", "all", hists, fits, labels, self.online_title)
62
63        # Overlay individual pile-up bins for each threshold
64        for threshold in self.thresholds:
65            hists = []
66            labels = []
67            fits = []
68            for pileup in self.pileup_bins.iter_all():
69                if not isinstance(pileup, int):
70                    continue
71                hists.append(self.efficiencies.get_bin_contents([pileup, threshold]))
72                if with_fits:
73                    fits.append(self.fits.get_bin_contents([pileup, threshold]))
74                labels.append(str(self.pileup_bins.bins[pileup]))
75            self.__make_overlay(pileup, threshold, hists, fits, labels, "PU bin")
76
77        # Produce the fit summary plot
78        if with_fits:
79            self.__summarize_fits()
80
81    def to_root(self, filename):
82        """ Write histograms to disk """
83        to_write = [self, self.yields]
84        if hasattr(self, "efficiencies"):
85            to_write += [self.efficiencies]
86        to_root(to_write, filename)
87
88
89    def __fill_efficiencies(self):
90        # Boiler plate to convert a given distribution to a efficiency
```

Convert yields into efficiencies and optionally apply the fitting

Make the plot for the inclusive pile-up bin with different thresholds

Make one plot for each threshold, separated into pile-up bins

Save the contained histograms to the given root file

Ben Krikler:
b.krikler@cern.ch

Inside a Plotter: efficiency.py

```
89      def __fill_efficiencies(self):
90          # Boiler plate to convert a given distribution to a efficiency
91          def make_eff(labels):
92              pileup_bin = labels["pileup"]
93              threshold_bin = labels["threshold"]
94              total = self.yields.get_bin_contents([pileup_bin, bn.Base.everything])
95              passed = self.yields.get_bin_contents([pileup_bin, threshold_bin])
96              efficiency = passed.Clone(passed.name.replace("yield", "efficiency"))
97              efficiency.Divide(total)
98              return efficiency
99
100         # Actually make the efficiencies
101         self.efficiencies = HistogramCollection([self.pileup_bins, self.thresholds],
102                                                 make_eff)
103
104     def __fit_efficiencies(self):
105         def make_fit(labels):
106             pileup_bin = labels["pileup"]
107             threshold_bin = labels["threshold"]
108             efficiency = self.efficiencies.get_bin_contents([pileup_bin, threshold_bin])
109             params = fit_efficiency(efficiency, self.thresholds.get_bin_center(threshold_bin))
110             return params
111
112         # Actually make the efficiencies
113         self.fits = HistogramCollection([self.pileup_bins, self.thresholds],
114                                         make_fit)
115
116     def __make_overlay(self, pileup, threshold, hists, fits, labels, header):
117         with preserve_current_style():
118             # Draw each efficiency (with fit)
119             canvas = draw(hists, draw_args={'xtitle': self.offline_title,
120                                             'ytitle': 'Efficiency'})
121             if len(fits) > 0:
122                 for fit, hist in zip(fits, hists):
123                     fit["asymmetric"].linecolor = hist.GetLineColor()
124                     fit["asymmetric"].Draw("same")
125
126             # Add labels
127             label_canvas()
128
129             # Add a legend
130             legend = Legend(len(hists), header=header)
131             for hist, label in zip(hists, labels):
132                 legend.AddEntry(hist, label)
133             legend.Draw()
134
135             # Save canvas to file
136             name = self.filename_format.format(type="efficiency_",
137                                                pileup=pileup,
138                                                threshold=threshold)
139             self.save_canvas(canvas, name)
```

Convert yields to efficiency

Apply the efficiency curve fitting.
Uses the fit_efficiency utility method

Actually make a plot given a list of histograms.
Might move method into BasePlotter if wanted elsewhere

Save the canvas using method from base class

Ben Krikler:
b.krikler@cern.ch

Some utility methods

- Make life easier
- cmsl1t/utils
- Drawing and labelling plots
- Jet matching algorithm
- Efficiency fitting
- Time code

Batch running: Coming soon

- Wrapper script to submit jobs to batch
- Run normal tool with `-r` option and set the input file list to the output
- Input files can be stored on xrootd

Ben Krikler: b.krikler@cern.ch

Contributing and the Repository

- Typical procedure:
  - Fork the official repository to your account
  - Open a PR from your fork to the main repository
  - Wait or ask for people to review
- Unit tests, and pep8 compliance
  - All unit tests must pass
  - Any framework-level code should be unit-tested
  - All code must comply with PEP-8
    - http://legacy.python.org/dev/peps/pep-0008/
    - Eg: No more than 100 characters per line, tab = 4 spaces, space around operators, after commas, etc
- Can check all of this by running:
  - **`$ make test`**

Getting Dirty

Questions and discussion

- Can you go away and use it?
- What do you think is missing?
- Is anything unclear?
- Anything else....

Ben Krikler:
b.krikler@cern.ch

Coding session

- Play with the config file
- Implement an analyzer, eg. resolutions
- Help us benchmark
- Add feature requests to GitHub