cms-l1t-analysis Documentation

Release 0.1.1

kreczko,benkrikler

Contents

1	What	t is cms1lt?
	1.1	Goals of the package
	1.2	What is not included
2	CMS	L1T Analysis Tutorial 5
	2.1	Getting started
		2.1.1 cms11t requirements
		2.1.2 Install cmsl1t
		2.1.3 Run local examples
		2.1.4 Run examples on HTCondor
		2.1.5 Run examples on the grid
	2.2	Configuration
	2.3	Creating histograms
	2.4	Adding your own analyzer
	2.7	reduing your own unaryzer
3	Indic	es and tables 9
4	API I	Reference 11
	4.1	cmsl1t.analyzers: Analyzers
	4.2	cmsl1t.collections: Collections
	4.3	cmsl1t.config: ConfigParser
	4.4	cmsl1t.hist: Hist
	4.5	cmsl1t.io: IO
	4.6	cmsl1t.plotting: Plotting
	4.7	cmsl1t.playground: Playground
	4.8	
		cmslit.recalc: Recalculations
	4.9	cmsl1t.recalc: Recalculationscmsl1t.utils: Utils

Contents:

Contents 1

2 Contents

			-
\cap LI	ΛC		
LΠ	ΑH	' I 🗀	R I

What is cms1lt?

cmsl1t is a python package for Level 1 Trigger analysis for the Compact Muon Solenoid (CMS) experiment at the Large Hadron Collider (LHC).

Goals of the package

What is not included.

cms-l1t-analysis Documentation,	Pologoo 0 1 1	
cms-in-analysis documentation,	nelease U.T.T	

CHAPTER 2

CMS L1T Analysis Tutorial

Getting started

cmsl1t requirements

Install cmsl1t

Run local examples

Run examples on HTCondor

Run examples on the grid

Configuration

This project uses configuration files in the YAML format to define the workflow.

Disclaimer: We are still in alpha, this section is likely to change.

A config file consists of muliple sections. The general section describes the version and name of the config.

general:

version: 0.0.1 name: Benchmark

The input section describes the data that is to be processed and might be changed in the near future. The first subsection, files, is a list of files that can be either relative paths, absolute paths or global paths (e.g. xrootd) and can include wildcards.

```
input:
    files:
        - data/L1Ntuple_*.root
```

The second subsection, sample is used to describe the data: The name of the dataset, the title and the run number. The name is likely used in file and histogram names, while the title is meant to be used in string representations (e.g titles/legends of histograms). If pileup reweighting is required, the pileup_file parameter needs to be set.

```
input:
...
sample:
   name: Data
   title: 2016 Data
   pileup_file: ""
   run_number: 276243
```

The trigger subsection describes which trigger is to be used for this dataset. As the sample name and title, the trigger counterparts play a similar role.

```
input:
    ...
    trigger:
    name: SingleMu
    title: Single Muon
```

The analysis section describes which analyzers are to be run. Global parameters include flags and binning for the analyzers (do_fit, pu_bins). These can also be specified later separately for each analyzer if required.

```
analysis:
do_fit: False
pu_bins: 0,13,20,999
```

The analyzers subsection of analysis is a list of all analyzers to be run. These analyzers have to satisfy the same API as cmsllt.analyzers.BaseAnalyzer and be visible in the *PYTHONPATH*.

```
analysis:
...
analyzers:
- cmsl1t.analyzers.demo_analyzer
```

Modifiers are a way to enrich the event content by attaching objects to the event itself. E.g. cmsllt.recalc.met.llMetNot28 reads in event.caloTowers and creates a new object, event.llMetNot28, that can then be accessed by all analyzers.

```
analysis:
    ...
    modifiers:
    - cmsl1t.recalc.met.l1MetNot28:
        in: event.caloTowers
```

```
out: event.l1MetNot28
- cmsl1t.recalc.met.l1MetNot28HF:
   in: event.caloTowers
   out: event.l1MetNot28HF
```

Next, you can specify if you want progress information (e.g. a progress bar) and how often this information is updated (report_every in units of events).

```
analysis:
...
progress_bar:
   report_every: 1000
# or to switch it off
# progress_bar:
# enable: False
```

And finally the output section describes where the output, usually ROOT files, is stored. The `template entry is composed of a list of paths that are joined to create the full output file. The template expects the following named parameters:

- date
- sample_name
- run_number
- trigger_name

which are automatically filled by the config parser

```
output:
    # template is a list here that is joined (os.path.join) in the config
    # parser
    template:
    - benchmark/new
    - "{date}_{sample_name}_run-{run_number}_{trigger_name}"
```

So a complete example would look something like that:

```
version: 0.0.1
name: Benchmark

input:
    files:
        - data/L1Ntuple_*.root
    sample:
        name: Data
        title: 2016 Data
    trigger:
        name: SingleMu
        title: Single Muon
    pileup_file: ""
    run_number: 276243
```

2.2. Configuration 7

```
analysis:
 do_fit: False
 pu_type: 0PU12,13PU19,20PU
 pu_bins: 0,13,20,999
 analyzers:
    - cmsl1t.analyzers.demo_analyzer
 modifiers:
   - cmsl1t.recalc.met.l1MetNot28:
       in: event.caloTowers
       out: event.l1MetNot28
   - cmsl1t.recalc.met.l1MetNot28HF:
       in: event.caloTowers
       out: event.l1MetNot28HF
output:
  # template is a list here that is joined (os.path.join) in the config parser
 template:
     - benchmark/new
     - "{date}_{sample_name}_run-{run_number}_{trigger_name}"
```

Creating histograms

Adding your own analyzer

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

CHAPTER 4

API Reference

cmsl1t.analyzers: Analyzers

cmsl1t.collections: Collections

cmsl1t.config: ConfigParser

cmsl1t.hist: Hist

cmsl1t.io: IO

cmsl1t.plotting: Plotting

 ${\tt cmsl1t.playground:} \ {\tt Playground}$

cmsl1t.recalc: Recalculations

cmsl1t.utils: Utils

Python Module Index

```
C cmsl1t, 11 cmsl1t.analyzers, 11 cmsl1t.hist, 11 cmsl1t.playground, 11 cmsl1t.plotting, 11 cmsl1t.recalc, 11 cmsl1t.utils, 11
```

Index

С

```
cmsl1t (module), 11
cmsl1t.analyzers (module), 11
cmsl1t.hist (module), 11
cmsl1t.playground (module), 11
cmsl1t.plotting (module), 11
cmsl1t.recalc (module), 11
cmsl1t.utils (module), 11
```