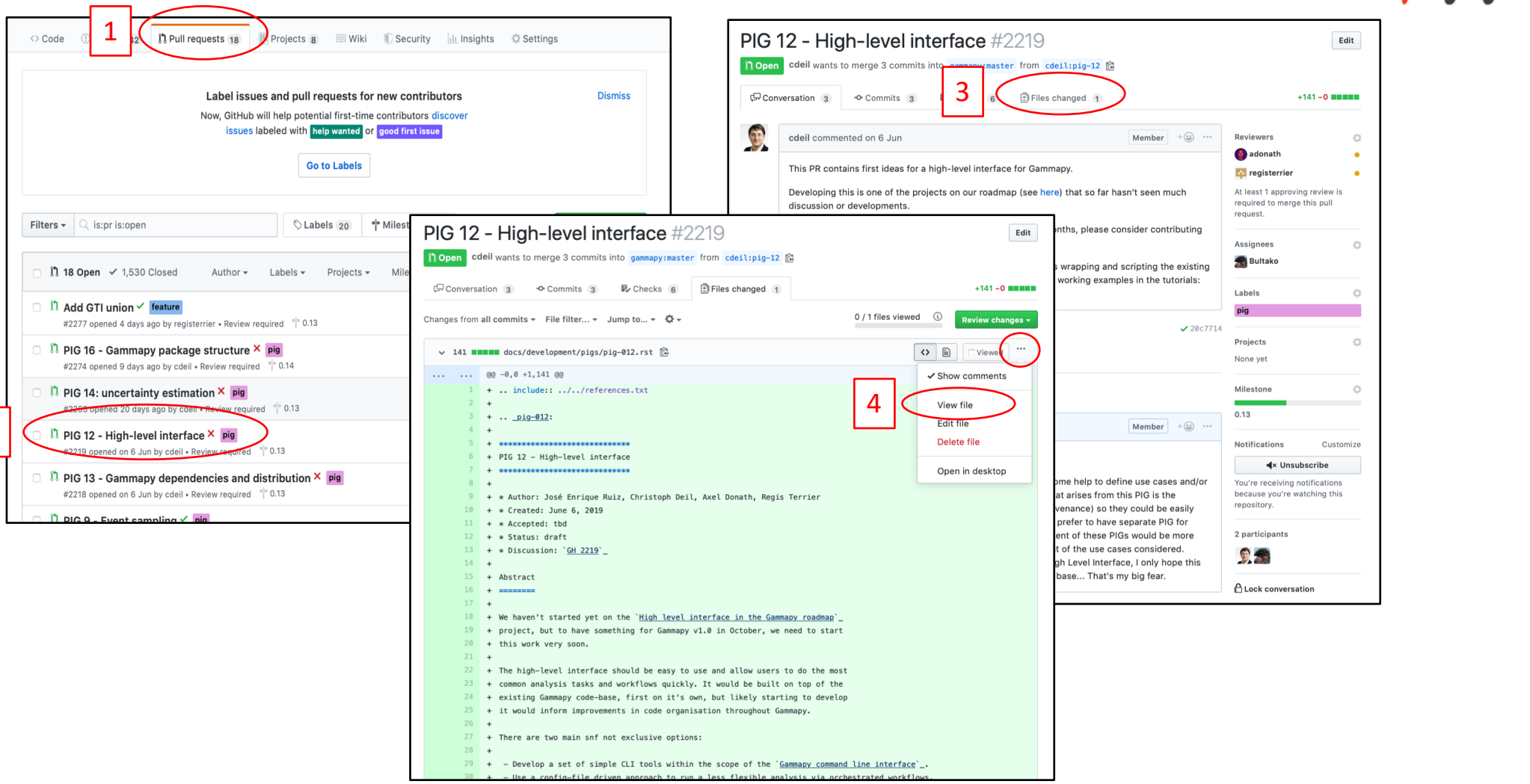
PIG 12
High Level Interface

Jose Enrique Ruiz – IAA / CSIC

Gammapy Coding Sprint

Erlangen - July 15th 2019


1
Code
Pull requests 18
Projects 8
Wiki
Security
Insights
Settings
Label issues and pull requests for new contributors
Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue
Dismiss
Go to Labels
Filters
is:pr is:open
Labels 20
18 Open
1,530 Closed
Author
Labels
Projects
Add GTI union
feature
#2277 opened 4 days ago by registerrier • Review required
0.13
PIG 16 - Gammapy package structure
pig
#2274 opened 9 days ago by cdeil • Review required
0.14
PIG 14: uncertainty estimation
pig
PIG 12 - High-level interface
pig
#2219 opened on 6 Jun by cdeil • Review required
0.13
PIG 13 - Gammapy dependencies and distribution
pig
#2218 opened on 6 Jun by cdeil • Review required
0.13
2
PIG 12 - High-level interface #2219
Edit
Open
cdeil wants to merge 3 commits into gammapy:master from cdeil:pig-12
Conversation 3
Commits 3
Checks 6
Files changed 1
+141 −0
Changes from all commits
File filter...
Jump to...
0 / 1 files viewed
Review changes
141 docs/development/pigs/pig-012.rst
Viewed
Show comments
View file
Edit file
Delete file
Open in desktop
3
4
@@ −0,0 +1,141 @@
+ .. include:: ../../references.txt
+ .. _pig-012:
+ ********************************
+ PIG 12 — High-level interface
+ ********************************
+ * Author: José Enrique Ruiz, Christoph Deil, Axel Donath, Regis Terrier
+ * Created: June 6, 2019
+ * Accepted: tbd
+ * Status: draft
+ * Discussion: `GH 2219`_
+ Abstract
+ ========
+ We haven't started yet on the `High level interface in the Gammapy roadmap`_
+ project, but to have something for Gammapy v1.0 in October, we need to start
+ this work very soon.
+ The high-level interface should be easy to use and allow users to do the most
+ common analysis tasks and workflows quickly. It would be built on top of the
+ existing Gammapy code-base, first on it's own, but likely starting to develop
+ it would inform improvements in code organisation throughout Gammapy.
+ There are two main snf not exclusive options:
+ - Develop a set of simple CLI tools within the scope of the `Gammapy command line interface`_.
+ - Use a config-file driven approach to run a less flexible analysis via orchestrated workflows.
PIG 12 - High-level interface #2219
Edit
Open
cdeil wants to merge 3 commits into gammapy:master from cdeil:pig-12
Conversation 3
Commits 3
Files changed 1
+141 −0
cdeil commented on 6 Jun
Member
This PR contains first ideas for a high-level interface for Gammapy.
Developing this is one of the projects on our roadmap (see here) that so far hasn't seen much discussion or developments.
nths, please consider contributing
s wrapping and scripting the existing
working examples in the tutorials:
20c7714
Member
me help to define use cases and/or
at arises from this PIG is the
venance) so they could be easily
prefer to have separate PIG for
ent of these PIGs would be more
t of the use cases considered.
gh Level Interface, I only hope this
base... That's my big fear.
Reviewers
adonath
registerrier
At least 1 approving review is required to merge this pull request.
Assignees
Bultako
Labels
pig
Projects
None yet
Milestone
0.13
Notifications
Customize
Unsubscribe
You're receiving notifications because you're watching this repository.
2 participants
Lock conversation

PIG 12 - High-level interface

- Author: José Enrique Ruiz, Christoph Deil, Axel Donath, Regis Terrier
- Created: June 6, 2019
- Accepted: tbd
- Status: draft
- Discussion: GH 2219

Abstract

We haven't started yet on the High level interface in the Gammapy roadmap project, but to have something for Gammapy v1.0 in October, we need to start this work very soon.

The high-level interface should be easy to use and allow users to do the most common analysis tasks and workflows quickly. It would be built on top of the existing Gammapy code-base, first on it's own, but likely starting to develop it would inform improvements in code organisation throughout Gammapy.

There are two main snf not exclusive options:

> - Develop a set of simple CLI tools within the scope of the Gammapy command line interface.
> - Use a config-file driven approach to run a less flexible analysis via orchestrated workflows.

The first option is what Fermitools and ctools already use, where each tool is simple/atomic enough to allow users to inspect the output results before taking a decission on how to run and set the parameter values for the next tool.

The second option is what is implemented in Enrico, or HAP in HESS. It involves basic orchestrated analysis workflows using a set of input parameters that the user provides via a configuration file. We would need to define a structured syntax for declaration and orchestration of parameters and tasks (probably using YAML format) and also provide config files as templates for users.

Achieving a stable high-level interface with one or both of these options would allow us to continue improving the Gammapy code-base without breaking user-defined workflows or reciped based on CLI tools.

PIG 5 - Gammapy 1.0 Roadmap

- Author: Axel Donath (editor), Régis Terrier & Christoph Deil
- Created: September 28, 2018
- Accepted: January 31, 2019
- Status: accepted
- Discussion: GH 1841

High-level interface

Implement a config-file based high level analysis interface (e.g. as used in `fermipy`) and command line tool. It gives access to limited, pre-scripted standard analysis workflows. Alternatively the high level analysis interface could generate pre-filled Python scripts or notebooks, that can be edited and executed by users.

What is a High Level Interface ?

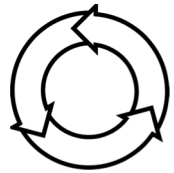
An **interface** to access and run most common Gammapy **tasks and workflows**

Built on top of existing Python API which will be hidden under the hood

Interface / Working methodology

- Easy to use for those using a **commands-in-terminal** approach
- Provide means to build → run **simple workflows based on configuration files**

Tasks and Workflows / Granularity / Iterative Fine-Tuning

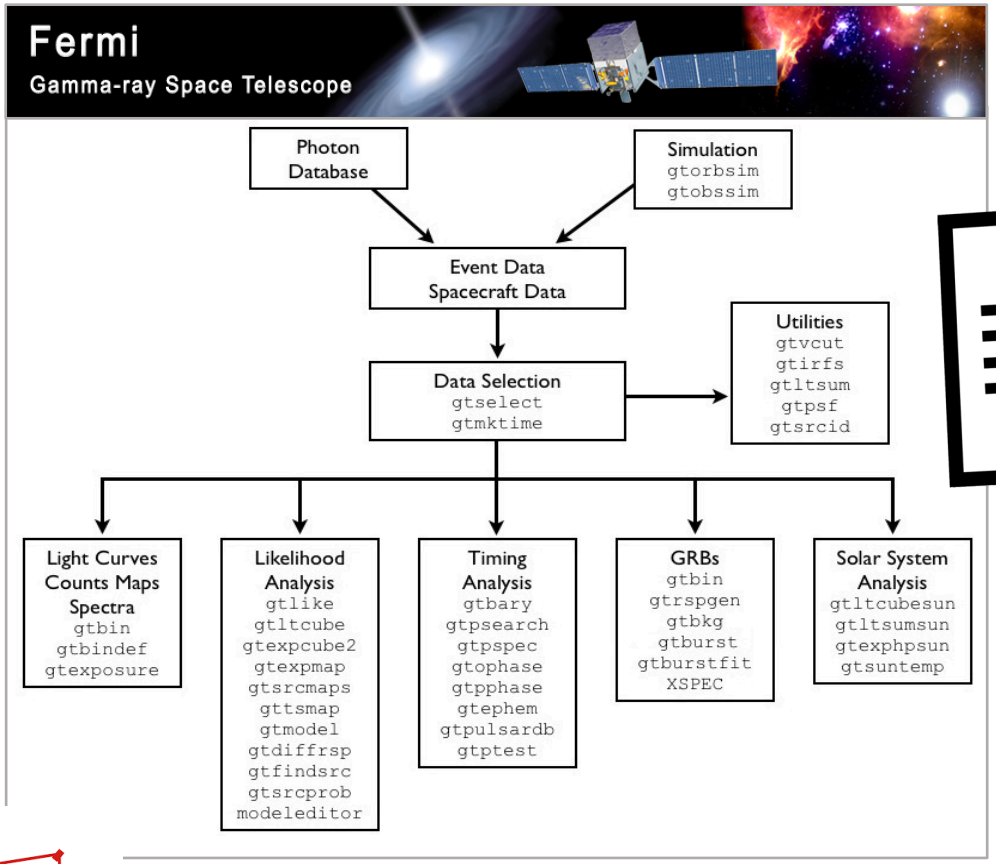


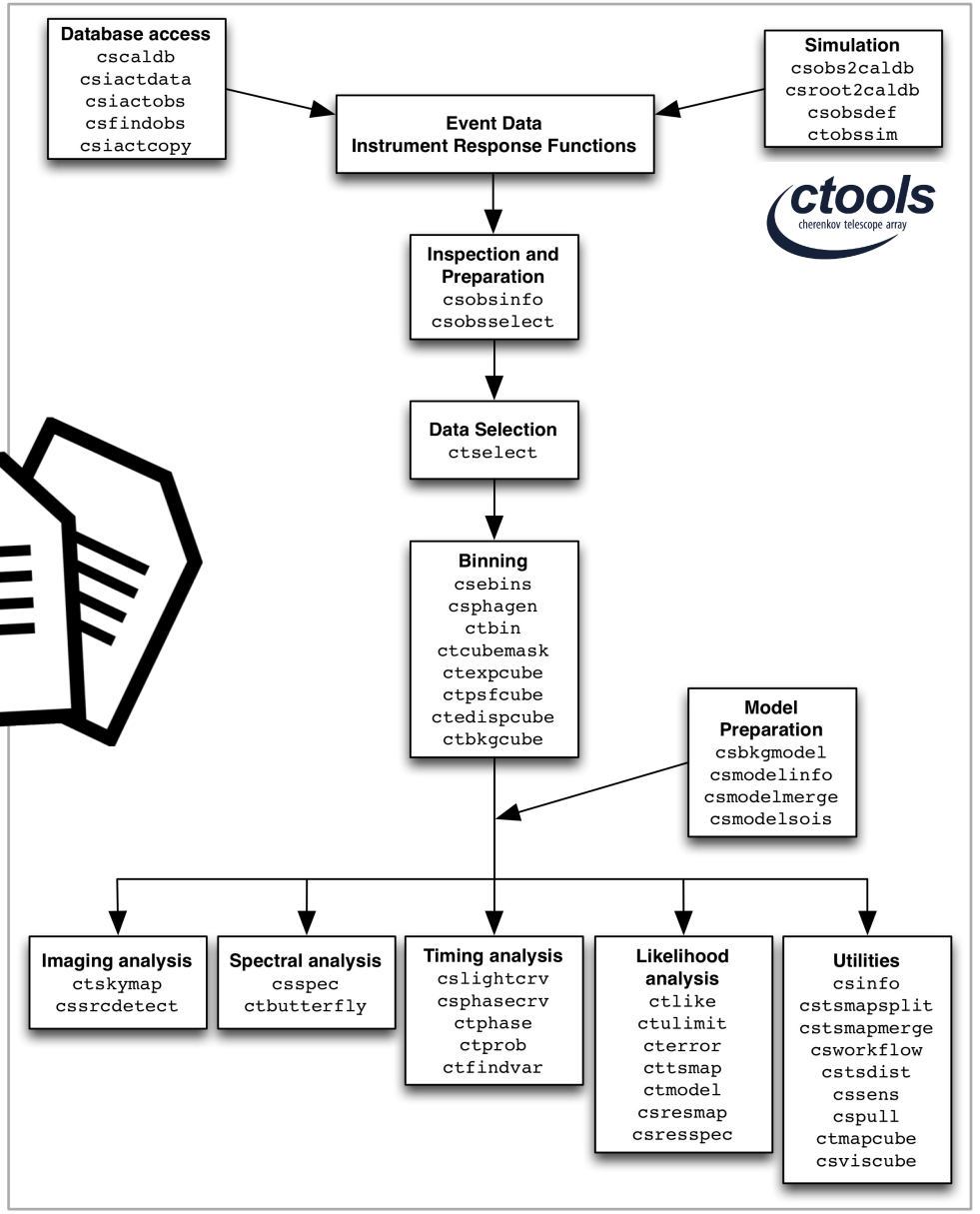
- **Atomic tasks** that need visual inspection of results for the user to take follow-up decisions
- Simple common workflows to get a **rough estimated result** of i.e. data reduction (map, spectrum,..), model fitting, spectral points, light curves, imaging, source detection, etc...

CTA will likely run an automated analysis of all data providing quicklook products

What do others have ?

Command Line Tools





H.E.S.S.

HAP - H.E.S.S. Analysis Program

What do others have ?

See also

Config files + Python API

Fermipy https://fermipy.readthedocs.io/en/latest/quickstart.html

Configuration files are used in Python analysis scripts written by the user to build the workflow and run it executing the Python script.

```
from fermipy.gtanalysis import GTAnalysis

gta = GTAnalysis('config.yaml',logging={'verbosity' : 3})
gta.setup()
# Free Normalization of all Sources within 3 deg of ROI center
gta.free_sources(distance=3.0,pars='norm')

# Free all parameters of isotropic and galactic diffuse components
gta.free_source('galdiff')
gta.free_source('isodiff')

# Free sources with TS > 10
gta.free_sources(minmax_ts=[10,None],pars='norm')

# Fix sources with TS < 10
gta.free_sources(minmax_ts=[None,10],free=False,pars='norm')

# Fix sources with 10 < Npred < 100
gta.free_sources(minmax_npred=[10,100],free=False,pars='norm')
```

```
data:
  evfile : ft1.lst
  scfile : ft2.fits
  ltcube : ltcube.fits

binning:
  roiwidth   : 10.0
  binsz      : 0.1
  binsperdec : 8

selection :
  emin : 100
  emax : 316227.76
  zmax    : 90
  evclass : 128
  evtype  : 3
  tmin    : 239557414
  tmax    : 428903014
  filter  : null
  target : 'mkn421'

gtlike:
  edisp : True
  irfs : 'P8R2_SOURCE_V6'
  edisp_disable : ['isodiff','galdiff']

model:
  src_roiwidth : 15.0
  galdiff  : '$FERMI_DIFFUSE_DIR/gll_iem_v06.fits'
  isodiff  : 'iso_P8R2_SOURCE_V6_v06.txt'
  catalogs : ['3FGL']
```

What do others have ?

Workflow Configuration

Enrico https://enrico.readthedocs.io

1. Create configuration file

```
enrico_config Myconfig.conf <answer a few questions like the position of your target>
```

2. Create XML file for sky model

```
enrico_xml Myconfig.conf
```

3. Run likelihood analysis for full spectrum

```
enrico_sed Myconfig.conf
```

4. Compute Lightcurve or TSMap

```
enrico_lc Myconfig.conf
enrico_tsmap Myconfig.conf
```

5. Plot results

```
enrico_plot_sed Myconfig.conf (plot the SED)
enrico_plot_lc Myconfig.conf (plot the lightcurve)
enrico_plot_tsmap Myconfig.conf (generate a fits file for the TS map)
```

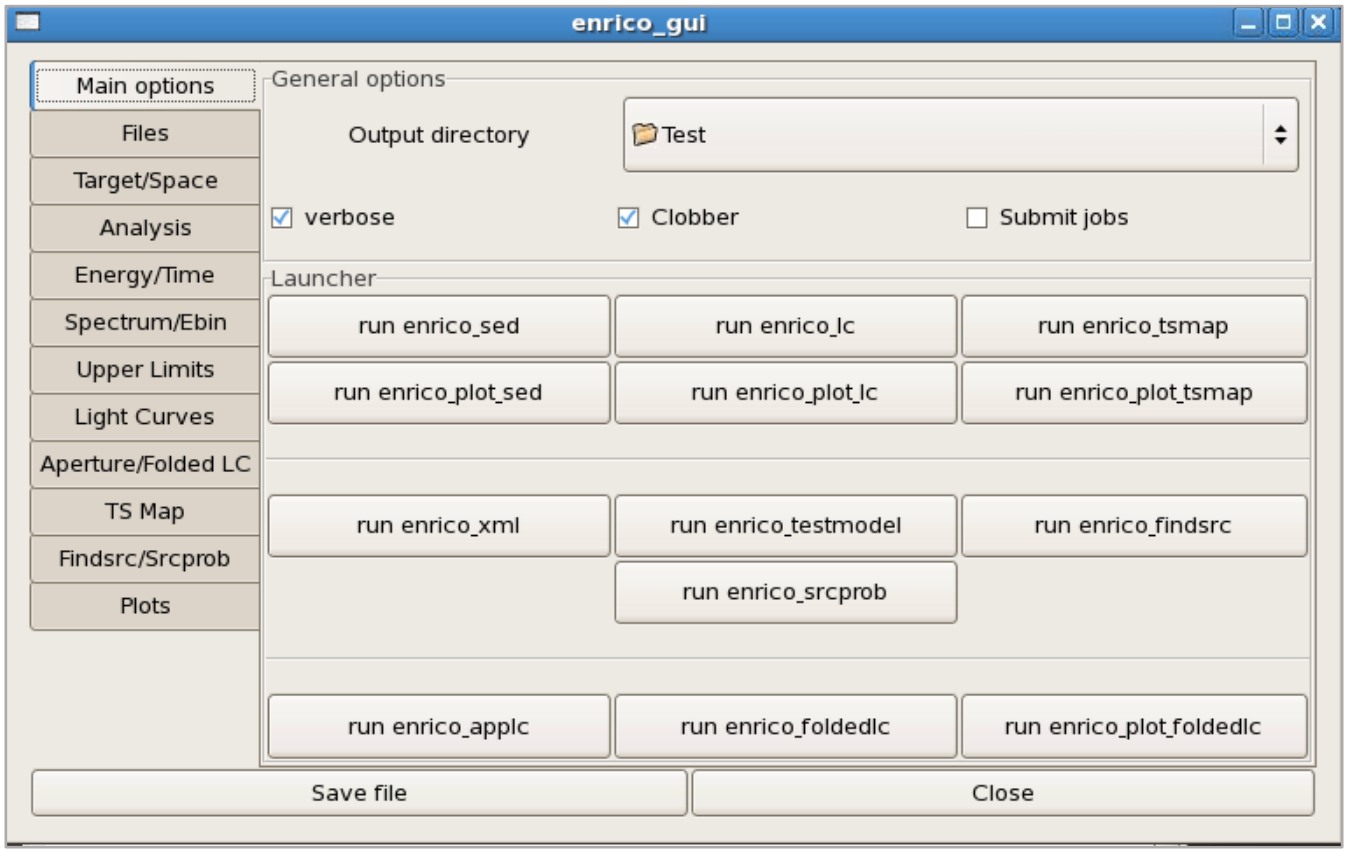
```
[target]
    name = PG155+113
    ra = 238.92935
    dec = 11.190102
    spectrum = PowerLaw2
    redshift = 0.5
    ebl_model = 4
    fit_tau = no
[space]
    xref = 238.92935
    yref = 11.190102
    rad = 10.0
    binsz = 0.1
    coordsys = CEL
    proj = AIT
    phibins = 0
[file]
    spacecraft = ~/myanalysis/FT2.fits
    event = ~/myanalysis/data.list
    xml = ~/myanalysis/XML_model.xml
    tag = MyTag
[time]
    tmin = 239557417.0
    tmax = 256970880.0
    file = ""
    type = 'MET'
[energy]
    emin = 200.0
    emax = 300000.0
    enumbins_per_decade = 10
[analysis]
    # General analysis options
```

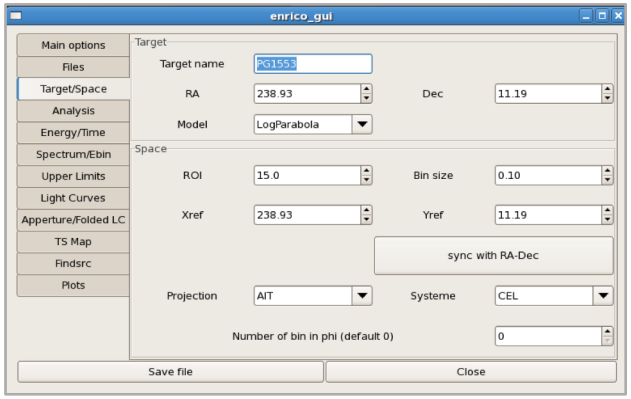
What do others have ?

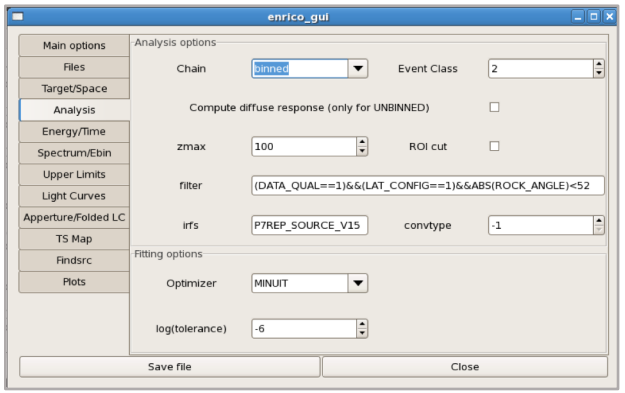
Workflow Configuration

Enrico https://enrico.readthedocs.io/en/latest/gui.html

GUI to build config file and run commands







What do others have ?

Workflow Configuration

csworkflow.py

```
# ================ #
# csworkflow class #
# ================ #
class csworkflow(ctools.cscript):
    """
    Executes an analysis workflow.

    The ``csworkflow`` script executes an analysis workflow defined in an
    XML file.
    """
```

Command line tools (`tools`)

ctapipe https://cta-observatory.github.io/ctapipe/ctapipe_api/tools/index.html

- Based on Python Traitlets framework
- Options for the command-tools may be specified in the command line or in a *config file*
- Config file formats supported JSON or Python

Also provides a framework to develop command line tools from Python classes.

What do we have ?

Command Line Interface

- Small set of tools for an embryonic CLI
- Developed with **Click** placed in `gammapy.scripts`
- Only `gammapy image` may be potentially needed in a data analysis process

Jupyter notebooks

- ~25 tutorials /recipes in notebook format

Python scripts

- Very few scripts and all need to be fixed and readapted

High-level analysis Python Classes in API

- SpectralAnalysisIACT
- LightCurveEstimator
- FluxPointEstimator
- SigmaVEstimator

Granularity

https://docs.gammapy.org/0.12/scripts/index.html

```
$ gammapy --help
Usage: gammapy [OPTIONS] COMMAND [ARGS]...

  Gammapy command line interface (CLI).

  Gammapy is a Python package for gamma-ray astronomy.

  Use ``--help`` to see available sub-commands, as well as the available
  arguments and options for each sub-command.

  For further information, see https://gammapy.org/ and
  https://docs.gammapy.org/

  Examples
  --------

  $ gammapy --help
  $ gammapy --version
  $ gammapy info --help
  $ gammapy info

Options:
  --log-level [debug|info|warning|error]
                                  Logging verbosity level.
  --ignore-warnings               Ignore warnings?
  --version                       Print version and exit.
  -h, --help                      Show this message and exit.

Commands:
  check     Run checks for Gammapy
  download  Download datasets and notebooks
  image     Analysis - 2D images
  info      Display information about Gammapy
  jupyter   Perform actions on notebooks
```

What should we do?

Develop a set of simple CLI tools

- Start prototyping with a small subset that could be extended as use-cases for analysis recipes are identified.
- Provide well established classes and code-structure to build the tools.

Use a config-file driven approach to run a less flexible analysis via orchestrated workflows

- First inspection /quicklook analysis via automated workflows.
- Provide **templated configuration files** for user editing.
- Skeleton **parametrized jupyter notebooks** filled and executed under the hood → pretty execution reports.

Format for declaration of parameters /values: YAML, JSON, others..

Structured syntax for workflow building and orchestration of tasks.

papermill

https://github.com/nteract/papermill

YAML files for declaration of parameters /values

Use cases design

- In which cases is serialization needed?

Development framework

- Take advantage of existing code set-up for CLI tools in `gammapy.scripts` `$ click_`
- Other frameworks i.e. Python-Fire https://github.com/google/python-fire
- **ctapipe tools**

Related activities / PIGs

Improving the code-base without breaking workflows/recipes made with High Level Interface

not stable API ⇏ not stable High Level Interface

PIG 6 Observations handling

PIG 7 Models

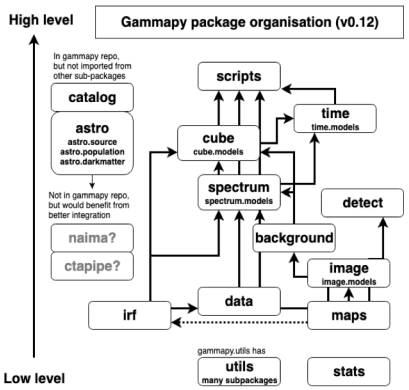
PIG 8 Datasets

PIG 10 Regions

PIG 11 Lightcurves

PIG 14 Uncertainties

PIG 16 Gammapy package structure



Future PIG Provenance / Structured logging

Seamless tracking and recording of software actions.
In the form of **structured** log files.
Provenance metadata model.

Enhanced inspection and **forensic studies** of the research analysis process.
Improving reproducibility and reuse by the community.