ctools "user" Introduction

Francesco Longo
Universita' di Trieste and INFN Trieste

Most of the material from J.Knödleseder

ctools

ctools

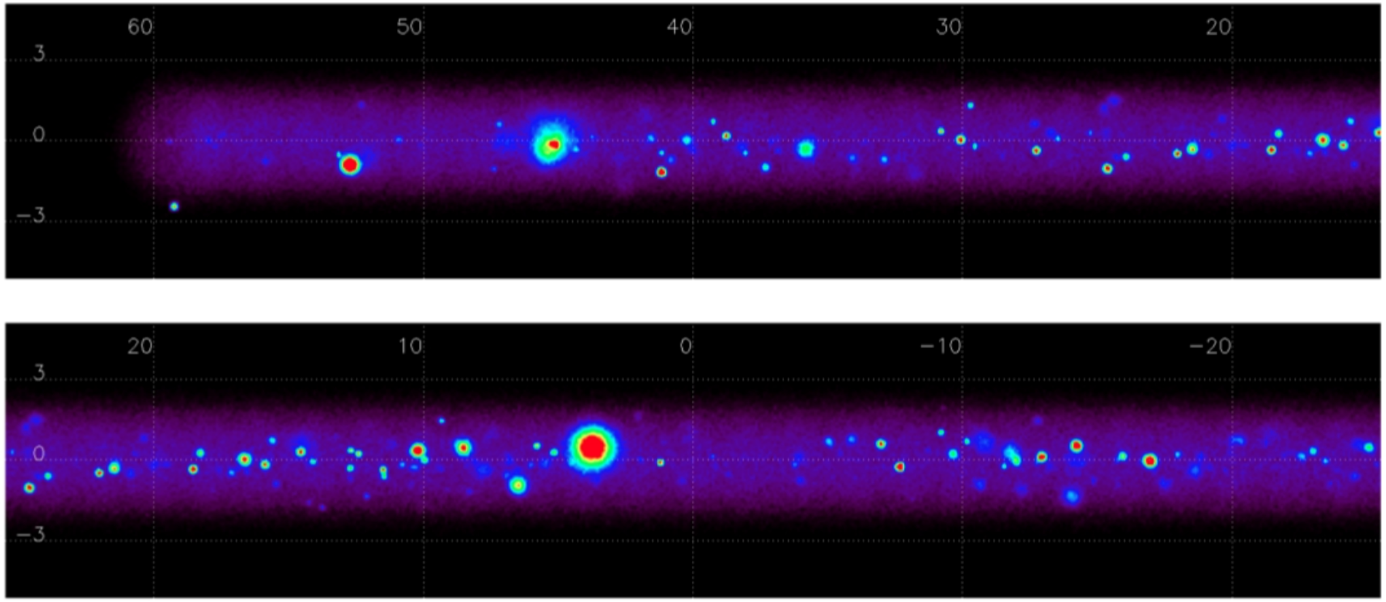
Overview
Activity
Roadmap
Issues
News
Documents
Wiki
Code status
Links
Forums
Files
Repository

ctools

ctools are a set of ftools-like executables needed for the scientific analysis of Cherenkov Telescope Array data. ctools is also a Python module that allows for interactive data analysis and building of analysis scripts and pipelines. ctools includes also an observation simulator to enable the scientific simulation of future CTA observations.

ctools are based on GammaLib, a versatile toolbox for the high-level analysis of astronomical gamma-ray data. Besides CTA, GammaLib supports also the analysis of Fermi-LAT data, and extensions to support further gamma-ray instruments are planned. This enables a simultaneous and coherent multi-instrument analysis of high-energy sources in the Universe.

ctools and GammaLib is free software distributed under the GPL license version 3.



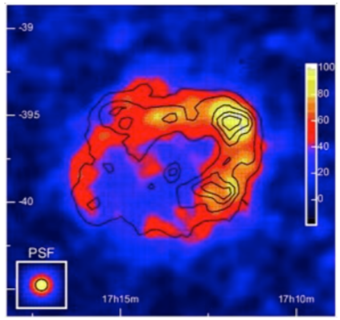
What are the ctools?

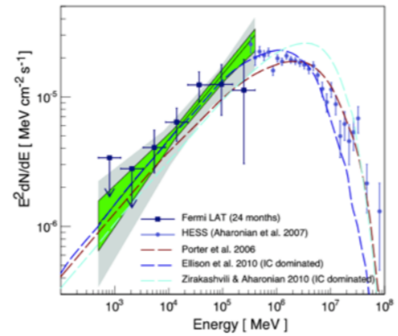
Tools for **end users** to extract science results from Cherenkov Telescope Array event lists and instrument response functions

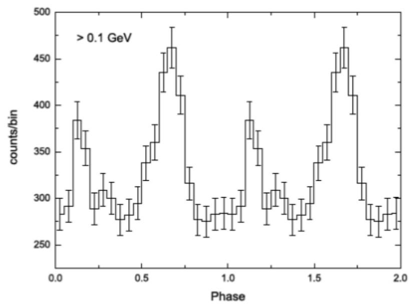
Background reduced event lists
Instrument response functions

Simulated event lists
Instrument response functions

ctools
cherenkov telescope array







Where do the ctools come from?

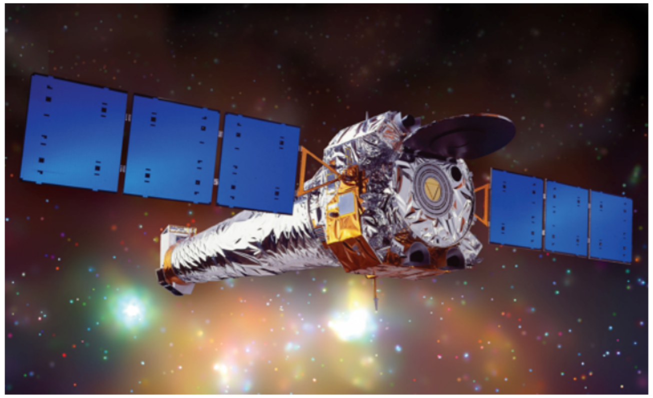
- **FITS** format invented in 1980'ies for the interchange of astronomical images on magnetic tapes (Wells, Greisen, Harten, 1981, A&AS, 44, 363)
- **HEASARC** established in 1990 as NASA's archive for **high-energy observatories** (support ROSAT, CGRO, ASCA, RXTE and achieve cost savings by reusing software and archive resources, today supports 32 HE missions and 25 CMB experiments)
- **FTOOLS** developed by HEASARC since 1992 as a generic set of software utilities to manipulate FITS files (current release: 6.20, 18/1/2017)

Where do the ctools come from?

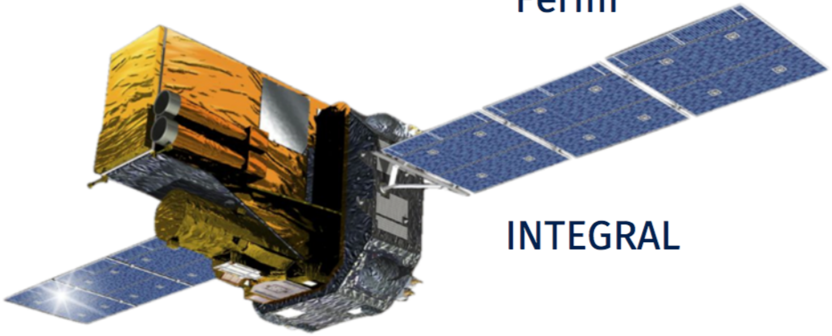
FTOOLS characteristics

- Modular workflow
- Use of IRAF parameter files

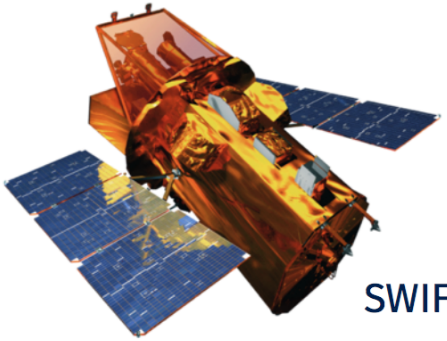
Fermi



Chandra



INTEGRAL



SWIFT

FTOOLS (like) analysis has become a **standard in high-energy astronomy**. Thousands of astronomers are today **familiar** with this standard.

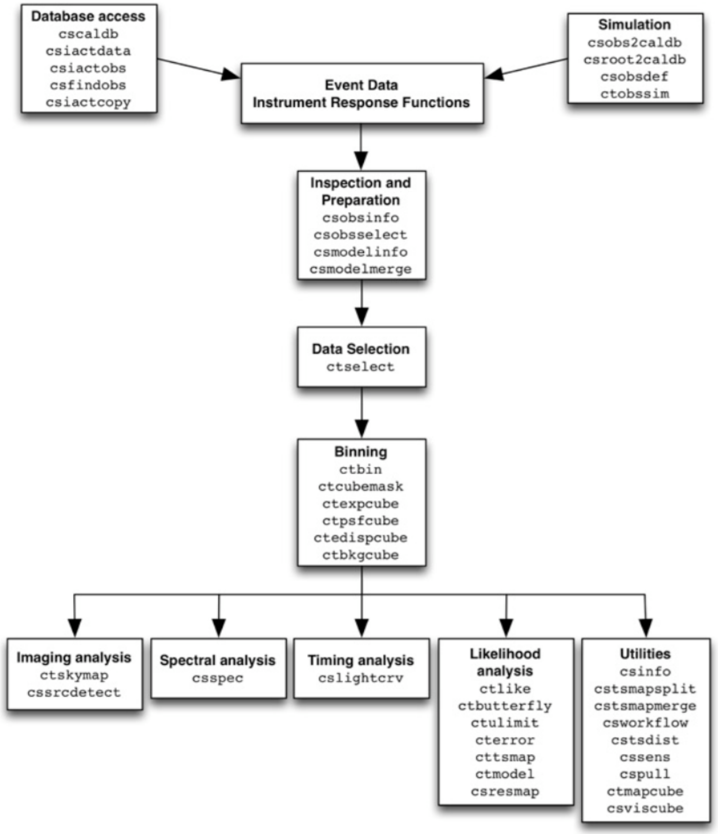
Where do the ctools come from?

ctools are **intentionally** very similar to the Fermi/LAT Science Tools

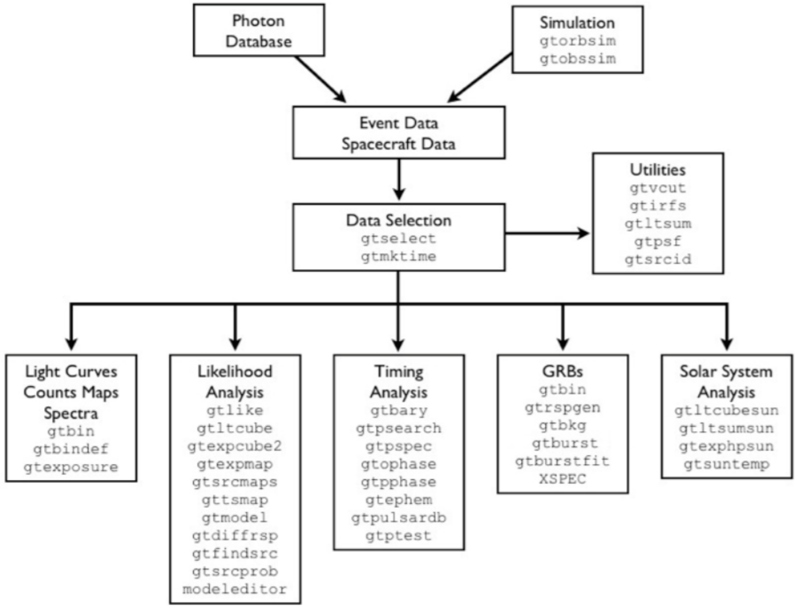
- Fermi/LAT Science Tools proven successful
- Basically **no learning** curve for **Fermi/LAT users**
- **Low learning curve** for users of other **HE observatories**
- But admittedly **some learning curve** for people from the **VHE community**

Status of ctools

ctools



Fermi/LAT Science Tools



- In terms of functionality and number of tools equivalent to the Fermi/LAT Science Tools
- More models
- Fitting of spatial and temporal parameters possible

ctools developers

ctools
cherenkov telescope array

CTA

Cherenkov Telescope Array Science Analysis Software

Home Download

Home | Documentation »

Develop

ctools is an open source projet and you are highly welcome to contribute to the development. Contributions can come in any areas: writing C++ code, contributing Python scripts, writing documentation, testing code, etc.

The ctools development is managed by a development platform that is accessible at https://cta-redmine.irap.omp.eu/projects/ctools/. Please check out the Wiki on that platform that contains information on how to contribute to the ctools development.

You may also want to get in the ctools information flow by subscribing to the ctools@irap.omp.eu mailing list. To subscribe simply send an e-mail to ctools-subscribe@irap.omp.eu.

We are organising regular Coding sprints to allow newcomers to get familiar with the code base and the coding practices. You are highly invited to join one of the next coding sprints.

You can also follow @gammalib on twitter to get informed about new release of GammaLib and ctools.

Behind the scenes

ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

Next sprint: 3-7 April 2017

ctools

ctools CTA
Cherenkov Telescope Array Science Analysis Software

Home Download

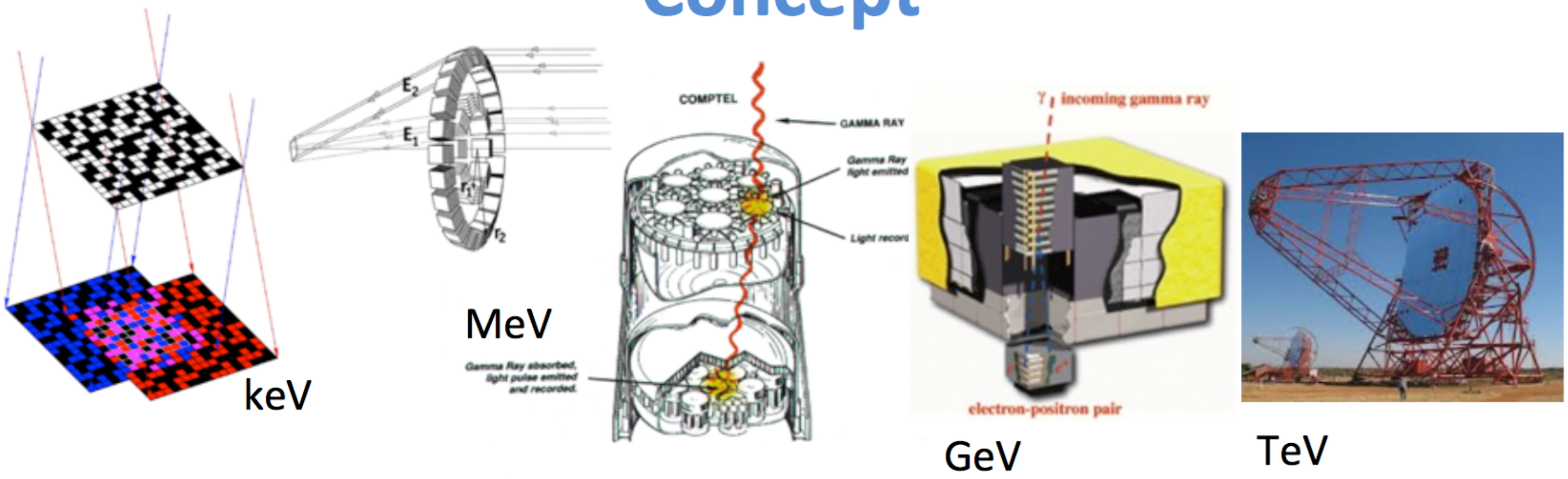
Home | Documentation » User Manual »

Introduction

ctools is a highly modular collection of utilities for processing and analysing CTA reconstructed event data in the FITS (Flexible Image Transport System) data format. Each utility presents itself as a FTOOL (see http://heasarc.gsfc.nasa.gov/ftools/) and performs a single simple task such as event binning, event selection or model fitting. Individual utilities can easily be chained together in scripts to achieve more complex operations, either by using the command line interface, or by using the Python scripting language. The ctools user interface is controlled by standard IRAF-style parameter files. Software is written in C++ to provide portability across most computer systems. The data format dependencies between hardware platforms are isolated through the cfitsio library package from HEASARC (http://heasarc.gsfc.nasa.gov/fitsio/).

This User Manual describes the use of the ctools software.

Concept



All gamma-ray telescopes measure individual photons as events =>
Handle events from gamma-ray telescopes in an abstract and common software framework.
Existing high-energy analysis frameworks share a number of **common features** (FITS files, likelihood fitting, modular design).

… is the client that uses the bricks provided by

CTA specific

generic

… to build a set of analysis executables for CTA (and alike)

Summary

- Status of ctools SW
- GammaLib introduction
- ctools introduction
- CTA response functions

ctools introduction

CTA

Cherenkov Telescope Array Science Analysis Software

Home Get it

ctools

About

ctools is a software package developed for the scientific analysis of Cherenkov Telescope Array (CTA) data. Analysis of data from existing Imaging Air Cherenkov Telescopes (such as H.E.S.S., MAGIC or VERITAS) is also supported, provided that the data and response functions are available in the format defined for CTA.

ctools comprises a set of ftools-like binary executables with a command-line interface allowing for interactive step-wise data analysis. ctools comprises also a Python module allowing to control all executables. Creation of shell or Python scripts and pipelines is supported. ctools comprises also cscripts, which are Python scripts that behave like binary ftools executables. Extensions of the ctools package by user defined binary executable or Python scripts is supported.

ctools are based on GammaLib, a versatile toolbox for the high-level analysis of astronomical gamma-ray data. Besides CTA, GammaLib supports also the analysis of Fermi/LAT and COMPTEL data, and extensions to support further gamma-ray instruments are planned. An interface to virtual observatory ressources is also in preparation. By making use of the GammaLib multi-instrument capabilities, ctools supports the joint analysis of CTA (or any IACT providing data in the CTA format), Fermi/LAT and COMPTEL data.

ctools is free software distributed under the GNU GPL license version 3

http://cta.irap.omp.eu/ctools/

ctools paper

arXiv.org > astro-ph > arXiv:1606.00393

Search or Article I

(Help | Advanced searc

Astrophysics > Instrumentation and Methods for Astrophysics

GammaLib and ctools: A software framework for the analysis of astronomical gamma-ray data

J. Knödlseder, M. Mayer, C. Deil, J.-B. Cayrou, E. Owen, N. Kelley-Hoskins, C.-C. Lu, R. Buehler, F. Forest, T. Louge, H. Siejkowski, K. Kosack, L. Gerard, A. Schulz, P. Martin, D. Sanchez, S. Ohm, T. Hassan, S. Brau-Nogué

(Submitted on 1 Jun 2016 (v1), last revised 22 Jul 2016 (this version, v2))

The field of gamma-ray astronomy has seen important progress during the last decade, yet there exists so far no common software framework for the scientific analysis of gamma-ray telescope data. We propose to fill this gap by means of the GammaLib software, a generic library that we have developed to support the analysis of gamma-ray event data. GammaLib has been written in C++ and all functionality is available in Python through an extension module. On top of this framework we have developed the ctools software package, a suite of software tools that enables building of flexible workflows for the analysis of Imaging Air Cherenkov Telescope event data. The ctools are inspired by science analysis software available for existing high-energy astronomy instruments, and they follow the modular ftools model developed by the High Energy Astrophysics Science Archive Research Center. The ctools have been written in Python and C++, and can be either used from the command line, via shell scripts, or directly from Python. In this paper we present the GammaLib and ctools software versions 1.0 that have been released end of 2015. GammaLib and ctools are ready for the science analysis of Imaging Air Cherenkov Telescope event data, and also support the analysis of Fermi-LAT data and the exploitation of the COMPTEL legacy data archive. We propose to use ctools as the Science Tools software for the Cherenkov Telescope Array Observatory.

Comments:	19 pages, 10 figures, accepted for publication in A&A, corrected X axis units in Figure 10
Subjects:	**Instrumentation and Methods for Astrophysics (astro-ph.IM)**; High Energy Astrophysical Phenomena (astro-ph.HE)
Journal reference:	A&A 593, A1 (2016)
DOI:	10.1051/0004-6361/201628822
Cite as:	**arXiv:1606.00393 [astro-ph.IM]**
	(or **arXiv:1606.00393v2 [astro-ph.IM]** for this version)

Status of ctools

ctools CTA
Cherenkov Telescope Array Science Analysis Software

About | Getting ctools | For Users | For Developers | Help

Home | Documentation » Getting ctools » previous | next | index

Download

ctools can be obtained in form of releases or directly from the git development repository. Prefer a release if you intend using ctools for production (and publications). Clone the code from git if you need the most recent code that implements new features and corrects known bugs.

Releases

The latest ctools release is `ctools-1.3.1` (20 July 2017).

Below a table of ctools releases. Please read the *Release History* to learn more about new features and corrected bugs in a given release.

Warning
At this stage of the project there is a strict link between the ctools and gammalib versions. Please make sure that you have the corresponding gammalib version installed before installing ctools. The Mac OS X packages comprise both ctools and gammalib.

ctools	gammalib	Mac OS X package
1.3.1	1.3.1	ctools-1.3.1-macosx10.7.dmg
1.3.0	1.3.0	ctools-1.3.0-macosx10.7.dmg
1.2.1	1.2.0	ctools-1.2.1-macosx10.7.dmg
1.2.0	1.2.0	ctools-1.2.0-macosx10.7.dmg
1.1.0	1.1.0	ctools-1.1.0-macosx10.3.dmg
1.0.1	1.0.1	ctools-1.0.1-macosx10.3.dmg
1.0.0	1.0.0	ctools-1.0.0-macosx10.3.dmg
0.10.0	0.11.0	ctools-0.10.0-macosx10.3.dmg
0.9.0	0.10.0	ctools-0.9.1-macosx10.3.dmg
0.8.1	0.9.1	ctools-00-08-01-macosx10.3.dmg
0.8.0	0.9.0	ctools-00-08-00-macosx10.3.dmg

Table Of Contents

Previous topic

Getting ctools

Next topic

Installing ctools

Quick search

Go

Enter search terms or a module, class or function name.

GammaLib

Gammalib

GammaLib
A versatile toolbox for scientific analysis of astronomical gamma-ray data

Home Get it

GammaLib

About

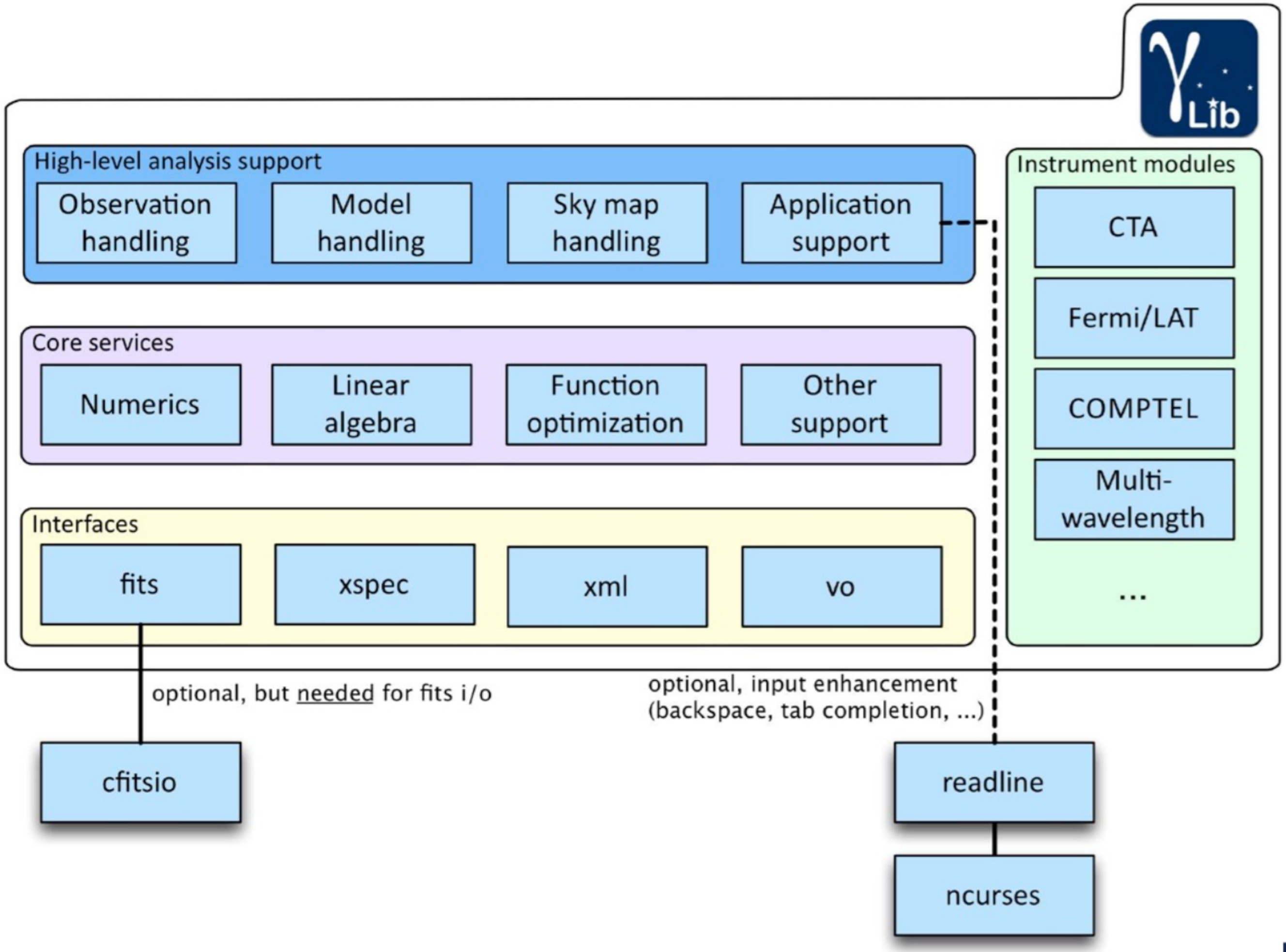
The GammaLib is a versatile toolbox for the high-level analysis of astronomical gamma-ray data. It is implemented as a C++ library that is fully scriptable in the Python scripting language. The library provides core functionalities such as data input and output, interfaces for parameter specifications, and a reporting and logging interface. It implements instruments specific functionalities such as instrument response functions and data formats. Instrument specific functionalities share a common interface to allow for extension of the GammaLib to include new gamma-ray instruments. The GammaLib provides an abstract data analysis framework that enables simultaneous multi-mission analysis.

GammaLib does not rely on any third-party software, except of HEASARC's cfitsio library that is used to implement the FITS interface. Large parts of the code treat gamma-ray observations in an abstract representation, and do neither depend on the characteristics of the employed instrument, nor on the particular formats in which data and instrument response functions are delivered. Instrument specific aspects are implemented as isolated and well defined modules that interact with the rest of the library through a common interface. This philosophy also enables the joint analysis of data from different instruments, providing a framework that allows for consistent broad-band spectral fitting or imaging. So far, GammaLib supports analysis of COMPTEL, Fermi/LAT, and Cherenkov telescope data (CTA, H.E.S.S., MAGIC, VERITAS).

GammaLib is free software distributed under the GNU GPL license version 3

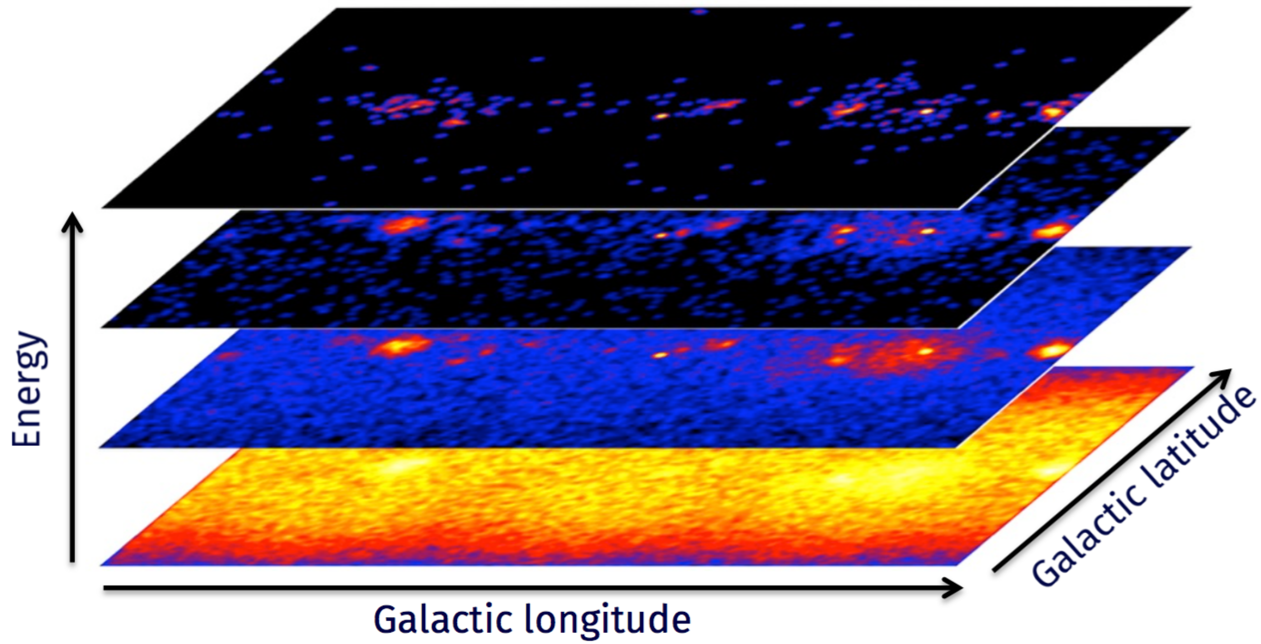
http://gammalib.sourceforge.net/

GammaLib: the technology under the hood



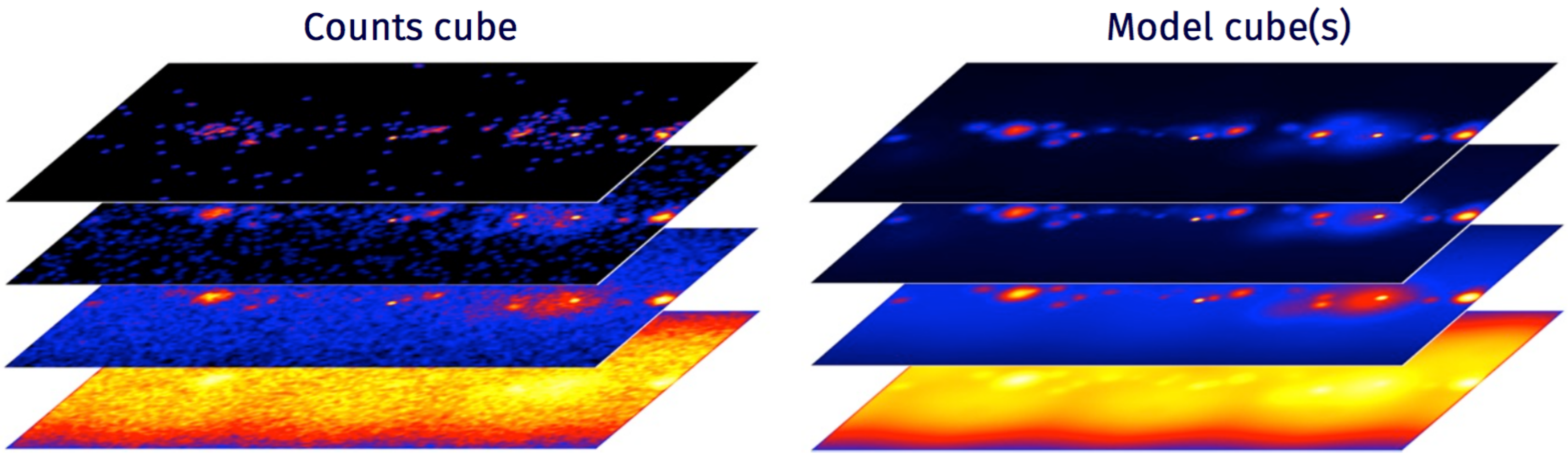
How does the analysis work?

- CTA data live in a 4 dimensional world
  - Reconstructed arrival direction (2d)
  - Reconstructed energy (1d)
  - Time (1d)
- For a given time interval, events can be binned in a 3 dimensional data space



How does the analysis work?

- ctools fits the 3d data space to extract spatial and spectral information about the gamma-ray sources
  - Parametrised model components
  - Simultaneous fitting of spatial and spectral parameters



- Classical VHE analysis techniques (on/off fitting, aperture photometry, etc.) are under development

Model summary

- Spatial
  - Point source
  - Radial symmetric models
    - Gaussian
    - Disk
    - Shell
  - Elliptical models
    - Gaussian
    - Disk
  - "Diffuse" models
    - Map
    - Map cubes (energy dependent maps)
    - Isotropic
  - Composite
- Temporal
  - Constant
  - Light curve
  - Phase curve
- Spectral
  - Power law
  - Broken power law
  - Exponentially cut off power law
  - Super exponentially cut off power law
  - Log parabola
  - Gaussian (line)
  - File function (arbitrary spectrum)
  - Node function (arbitrary fit)
  - Constant
  - Composite
  - Multiplicative (useful for EBL)

And more to come (e.g. Dark Matter Halo)

Conclusions

- ctools package allows simulating and analysing CTA & IACT event data
- Analysis approach and tools intentionally very similar to Fermi/LAT Science Tools
- Existing functionality equivalent to Fermi/LAT Science Tools
- In addition
  - More models (in particular for extended source, but also for temporal variations)
  - Spatial and temporal model fitting
- Support for joint multi-mission analysis (CTA, IACT, Fermi/LAT, COMPTEL, MWL)

ctools examples

ctools

Reference Manual

This manual provides reference information for all ctools and csripts. General information on ctools usage can be found here.

Below you find links to the command line reference for the tools and scripts that are available.

ctools

- ctbin — Generates counts cube
- ctbkgcube — Generates background cube
- ctbutterfly — Compute butterfly
- ctcubemask — Filter counts cube
- ctexpcube — Generates exposure cube
- ctlike — Performs maximum likelihood fitting
- ctmodel — Computes model counts cube
- ctobssim — Simulate CTA observations
- ctpsfcube — Generates point spread function cube
- ctselect — Selects event data
- ctskymap — Generates sky map
- cttsmap — Generates Test Statistics map
- ctulimit — Calculates upper limit

How to use ctobssim?

Simulating CTA data

CTA data are simulated by the executable ctobssim. To start the executable, type ctobssim at the console prompt (which is denoted by `$`). ctobssim will query for a number of parameters:

```
$ ctobssim
Model [$CTOOLS/share/models/crab.xml]
RA of pointing (degrees) (0-360) [83.63]
Dec of pointing (degrees) (-90-90) [22.01]
Calibration database [dummy]
Instrument response function [cta_dummy_irf]
Radius of FOV (degrees) (0-180) [5.0]
Start time (MET in s) (0) [0.0]
End time (MET in s) (0) [1800.0]
Lower energy limit (TeV) (0) [0.1]
Upper energy limit (TeV) (0) [100.0]
Output event data file or observation definition file [events.fits]
```

Each line represents a query for one parameter. The line starts with a short description of the parameter, followed by the default parameter value proposed by ctobssim in squared brackets `[ ]`. If no parameter is entered (which is the case for the majority of parameters shown here), the default parameter will be used. Otherwise, the specified parameter will overwrite the default parameter.

You may have recognised that the environment variable `$CTOOLS` has been used in the path names of the first two parameters. ctools will automatically expand the environment variables.

The CTA instrument properties (effective area, PSF width) are taken for the moment from a dummy performance table that is located in `$CTOOLS/share/caldb/cta`.

Events are simulated based on the instrument properties and based on a source and background model. Only events that fall within the specified region of interest (ROI), defined as a circle around a sky position in Right Ascension and Declination (in degrees), will be stored in the output event data file. The duration of the simulation is taken here to 30 minutes (or 1800 seconds). Events are simulated for energies between 0.1 and 100 TeV.

http://cta.irap.omp.eu/ctools/

How to?

Implementation

The general model is describe in ctools using a model definition XML file. Below is a simple example of such a file comprising one source and one background model. Each model is factorised in a spectral (tag `<spectrum>`) and a spatial component (tags `<spatialModel>` and `<radialModel>`):

$$M(x, y, E) = M_{\rm spectral}(E) \times M_{\rm spatial}(x, y)$$

In this specific example, the source component `Crab` describes a point source at the location of the Crab nebula with a power law spectral shape. The background component `Background` is modelled as a radial Gaussian function in offset angle squared (with the offset angle being defined as the angle between pointing and measured event direction) and a spectral function that is tabulated in an ASCII file.

```
<?xml version="1.0" standalone="no"?>
<source_library title="source library">
  <source name="Crab" type="PointSource">
    <spectrum type="PowerLaw">
       <parameter name="Prefactor" scale="1e-16" value="5.7"  min="1e-07" max="1000.0" free="1"/>
       <parameter name="Index"     scale="-1"    value="2.48" min="0.0"   max="+5.0"   free="1"/>
       <parameter name="Scale"     scale="1e6"   value="0.3"  min="0.01"  max="1000.0" free="0"/>
    </spectrum>
    <spatialModel type="SkyDirFunction">
      <parameter name="RA"  scale="1.0" value="83.6331" min="-360" max="360" free="1"/>
      <parameter name="DEC" scale="1.0" value="22.0145" min="-90"  max="90"  free="1"/>
    </spatialModel>
  </source>
  <source name="Background" type="RadialAcceptance" instrument="CTA">
    <spectrum type="FileFunction" file="$CTOOLS/share/models/bkg_dummy.txt">
      <parameter name="Normalization" scale="1.0" value="1.0" min="0.0" max="1000.0" free="1"/>
    </spectrum>
    <radialModel type="Gaussian">
       <parameter name="Sigma" scale="1.0" value="3.0" min="0.01" max="10.0" free="1"/>
    </radialModel>
  </source>
</source_library>
```

How to use?

The source and background model is defined by the XML file `$CTOOLS/share/models/crab.xml`:

```
<?xml version="1.0" standalone="no"?>
<source_library title="source library">
  <source name="Crab" type="PointSource">
    <spectrum type="PowerLaw">
       <parameter name="Prefactor" scale="1e-16" value="5.7"  min="1e-07" max="1000.0" free="1"/>
       <parameter name="Index"     scale="-1"    value="2.48" min="0.0"   max="+5.0"   free="1"/>
       <parameter name="Scale"     scale="1e6"   value="0.3"  min="0.01"  max="1000.0" free="0"/>
    </spectrum>
    <spatialModel type="SkyDirFunction">
      <parameter name="RA"  scale="1.0" value="83.6331" min="-360" max="360" free="0"/>
      <parameter name="DEC" scale="1.0" value="22.0145" min="-90"  max="90"  free="0"/>
    </spatialModel>
  </source>
  <source name="Background" type="RadialAcceptance" instrument="CTA">
    <spectrum type="FileFunction" file="$CTOOLS/share/models/bkg_dummy.txt">
      <parameter scale="1.0" name="Normalization" min="0.0" max="1000.0" value="1.0" free="1"/>
    </spectrum>
    <radialModel type="Gaussian">
       <parameter name="Sigma" scale="1.0" value="3.0" min="0.01" max="10.0" free="1"/>
    </radialModel>
  </source>
</source_library>
```

The model consists of a source library that contains 2 “sources”: the Crab nebula and an instrumental background model.

The Crab nebula is modelled by a factorized sky model that has a spectral and a spatial component (tags `<spectrum>` and `<spatialModel>`, respectively). The spectrum is modelled by a power law, which is defined by 3 parameters: the `Prefactor`, the `Index` and the `Scale`. The spatial model has 2 parameters: Right Ascension in degrees (RA), and Declination in degrees (DEC). Each parameter has a value and a scale factor, the real value of the parameter being the product value * scale. Typically, scale is chosen so that value is of the order of 1 (this is relevant for model fitting later). In addition, value is bound by a minimum (min) and maximum (max) value, and a parameter may be free (`free="1"`) or fixed (`free="0"`). The min, max, and free attributes are not relevant here for the simulations, but they will be important for the model fitting later.

http://cta.irap.omp.eu/ctools/

How to use?

The spectral intensity I(E) (in units of photons/cm2/s/MeV) of the power law is given by

$$\frac{dN}{dE} = N_0 \left(\frac{E}{E_0} \right)^{\gamma}$$

where the parameters in the XML definition have the following mappings:

- N_0 = `Prefactor`
- γ = `Index`
- E_0 = `Scale`

Note that energies are given in MeV.

The instrumental background of CTA is modelled by a factorized data model that has a spectral and a radial component (tags `<spectrum>` and `<radialModel>`, respectively). The spectral component describes the on-axis background counting rate of CTA as function of energy in units of counts/s/sr/TeV. The radial component describes the variation of the background rate with offset angle squared, (i.e. square of the offset angle with respect to the pointing direction) which is modelled here by a Gaussian. The only parameter of the radial component is the width of the Gaussian Sigma, which is here set to 3 degrees squared.

http://cta.irap.omp.eu/ctools/

Spectral Models

Power law

The GModelSpectralPlaw class implements the power law function

$$\frac{dN}{dE} = k_0 \left(\frac{E}{E_0} \right)^{\gamma}$$

where the parameters in the XML definition have the following mappings:

- k_0 = `Prefactor`
- γ = `Index`
- E_0 = `Scale`

The XML format for specifying a power law is:

```
<spectrum type="PowerLaw">
 <parameter name="Prefactor" scale="1e-16" value="5.7"  min="1e-07" max="1000.0" free="1"/>
 <parameter name="Index"     scale="-1"    value="2.48" min="0.0"   max="+5.0"   free="1"/>
 <parameter name="Scale"     scale="1e6"   value="0.3"  min="0.01"  max="1000.0" free="0"/>
</spectrum>
```

An alternative power law function is defined by the GModelSpectralPlaw2 class that uses the integral flux as parameter rather than the Prefactor:

$$\frac{dN}{dE} = \frac{N(\gamma+1)E^{\gamma}}{E_{\rm max}^{\gamma+1} - E_{\rm min}^{\gamma+1}}$$

where the parameters in the XML definition have the following mappings:

- N = `Integral`
- γ = `Index`
- $E_{\rm min}$ = `LowerLimit`
- $E_{\rm max}$ = `UpperLimit`

The XML format for specifying a power law defined by the integral flux is:

```
<spectrum type="PowerLaw2">
 <parameter scale="1e-07" name="Integral"   min="1e-07" max="1000.0"    value="1.0" free="1"/>
 <parameter scale="1.0"   name="Index"      min="-5.0"  max="+5.0"      value="-2.0" free="1"/>
 <parameter scale="1.0"   name="LowerLimit" min="10.0"  max="1000000.0" value="100.0" free="0"/>
 <parameter scale="1.0"   name="UpperLimit" min="10.0"  max="1000000.0" value="500000.0" free="0"/>
</spectrum>
```

NOTE: The UpperLimit and LowerLimit parameters are always treated as fixed and, as should be apparent from this definition, the flux given by the Integral parameter is over the range (LowerLimit, UpperLimit). Use of this model allows the errors on the integrated flux to be evaluated directly by likelihood, obviating the need to propagate the errors if one is using the PowerLaw form.

http://gammalib.sourceforge.net/user_manual/modules/model.html#sec-model

Spectral Models

Exponentially cut-off power law

The GModelSpectralExpPlaw class implements the exponentially cut-off power law function

$$\frac{dN}{dE} = k_0 \left(\frac{E}{E_0}\right)^{\gamma} \exp\left(\frac{-E}{E_{cut}}\right)$$

where the parameters in the XML definition have the following mappings:

- k_0 = `Prefactor`
- γ = `Index`
- E_0 = `Scale`
- E_{cut} = `Cutoff`

The XML format for specifying an exponentially cut-off power law is:

```
<spectrum type="ExpCutoff">
 <parameter name="Prefactor" scale="1e-16" value="5.7"  min="1e-07" max="1000.0" free="1"/>
 <parameter name="Index"     scale="-1"    value="2.48" min="0.0"   max="+5.0"   free="1"/>
 <parameter name="Cutoff"    scale="1e6"   value="1.0"  min="0.01"  max="1000.0" free="1"/>
 <parameter name="Scale"     scale="1e6"   value="0.3"  min="0.01"  max="1000.0" free="0"/>
</spectrum>
```

http://gammalib.sourceforge.net/user_manual/modules/model.html#sec-model

Spectral Models

Broken power law

The GModelSpectralBrokenPlaw class implements the broken power law function

$$\frac{dN}{dE} = k_0 \times \begin{cases} \left(\frac{E}{E_b}\right)^{\gamma_1} & \text{if } E < E_b \\ \left(\frac{E}{E_b}\right)^{\gamma_2} & \text{otherwise} \end{cases}$$

where the parameters in the XML definition have the following mappings:

- k_0 = `Prefactor`
- γ_1 = `Index1`
- γ_2 = `Index2`
- E_b = `BreakValue`

The XML format for specifying a broken power law is:

```
<spectrum type="BrokenPowerLaw">
 <parameter name="Prefactor"  scale="1e-16" value="5.7"  min="1e-07" max="1000.0" free="1"/>
 <parameter name="Index1"     scale="-1"    value="2.48" min="0.0"   max="+5.0"   free="1"/>
 <parameter name="BreakValue" scale="1e6"   value="0.3"  min="0.01"  max="1000.0" free="1"/>
 <parameter name="Index2"     scale="-1"    value="2.70" min="0.01"  max="1000.0" free="1"/>
</spectrum>
```

http://gammalib.sourceforge.net/user_manual/modules/model.html#sec-model

Spectral Models

Log parabola

The GModelSpectralLogParabola class implements the log parabola function

$$\frac{dN}{dE} = k_0 \left(\frac{E}{E_0} \right)^{\gamma + \eta \ln(E/E_0)}$$

where the parameters in the XML definition have the following mappings:

- k_0 = `Prefactor`
- γ = `Index`
- η = `Curvature`
- E_0 = `Scale`

The XML format for specifying a log parabola spectrum is:

```
<spectrum type="LogParabola">
 <parameter name="Prefactor" scale="1e-17" value="5.878"   min="1e-07" max="1000.0" free="1"/>
 <parameter name="Index"     scale="-1"    value="2.32473" min="0.0"   max="+5.0"   free="1"/>
 <parameter name="Curvature" scale="-1"    value="0.074"   min="-5.0"  max="+5.0"   free="1"/>
 <parameter name="Scale"     scale="1e6"   value="1.0"     min="0.01"  max="1000.0" free="0"/>
</spectrum>
```

An alternative XML format is supported for compatibility with the Fermi/LAT XML format:

```
<spectrum type="LogParabola">
 <parameter name="Prefactor" scale="1e-17" value="5.878"   min="1e-07" max="1000.0" free="1"/>
 <parameter name="alpha"     scale="1"     value="2.32473" min="0.0"   max="+5.0"   free="1"/>
 <parameter name="beta"      scale="1"     value="0.074"   min="-5.0"  max="+5.0"   free="1"/>
 <parameter name="Scale"     scale="1e6"   value="1.0"     min="0.01"  max="1000.0" free="0"/>
</spectrum>
```

where

- `alpha` = -`Index`
- `beta` = -`Curvature`

http://gammalib.sourceforge.net/user_manual/modules/model.html#sec-model

Spectral Models

File function

A function defined using an input ASCII file with columns of energy and differential flux values. The energy units are assumed to be MeV and the flux values are assumed to $\mathrm{cm^{-2} s^{-1} MeV^{-1}}$ (the only exception being a model for which the spatial component is a constant diffuse model GModelSpatialDiffuseConst; in this case, the units are $\mathrm{cm^{-2} s^{-1} MeV^{-1} sr^{-1}}$). The sole parameter is a multiplicative normalization:

$$\frac{dN}{dE} = N_0 \frac{dN}{dE}\bigg|_{\mathrm{file}}$$

where the parameters in the XML definition have the following mappings:

- N_0 = `Normalization`

The XML format for specifying a file function is:

```
<spectrum type="FileFunction" file="data/filefunction.txt">
 <parameter scale="1.0" name="Normalization" min="0.0" max="1000.0" value="1.0" free="1"/>
</spectrum>
```

If the file is given as relative path, the path is relative to the working directory of the executable. Alternatively, an absolute path may be specified. Any environment variable present in the path name will be expanded.

http://gammalib.sourceforge.net/user_manual/modules/model.html#sec-model

How to use?

ctobssim will write 2 files in the working directory: `events.fits` and `ctobssim.log`. The first file contains the simulated events in FITS format and can be inspected using `fv` or `ds9`. The FITS file will contain 3 extensions: an empty primary image, a binary table named `EVENTS` that holds the events (one row per event), and a binary table named `GTI` holding the Good Time Intervals (for the moment a single row with 2 columns providing the start and the stop time of the simulated time interval).

The second file produced by ctobssim is a human readable log file that contains information about the job execution. As example, the last lines from this file are shown here:

```
2014-10-30T22:35:06: +======================+
2014-10-30T22:35:06: | Simulate observation |
2014-10-30T22:35:06: +======================+
2014-10-30T22:35:06: === Observation ===
2014-10-30T22:35:06:  Simulation area ...........: 1.9635e+11 cm2
2014-10-30T22:35:06:  Simulation cone ...........: RA=83.63 deg, Dec=22.01 deg, r=5.5 deg
2014-10-30T22:35:06:  Time interval .............: 0 - 1800 s
2014-10-30T22:35:06:  Photon energy range .......: 100 GeV - 100 TeV
2014-10-30T22:35:06:  Event energy range ........: 100 GeV - 100 TeV
2014-10-30T22:35:06:  MC source photons .........: 207547 [Crab]
2014-10-30T22:35:06:  MC source events ..........: 995 [Crab]
2014-10-30T22:35:06:  MC source events ..........: 995 (all source models)
2014-10-30T22:35:06:  MC background events ......: 5146
2014-10-30T22:35:06:  MC events .................: 6141 (all models)
2014-10-30T22:35:06:
2014-10-30T22:35:06: +==================+
2014-10-30T22:35:06: | Save observation |
2014-10-30T22:35:06: +==================+
2014-10-30T22:35:06:
2014-10-30T22:35:06: Application "ctobssim" terminated after 10 wall clock seconds, consuming 0.3604 seconds o
```

Each line starts with the UTC time at which the line has been written. In this run, 207547 Crab photons have been thrown over an area of 19.6 square kilometres during a time interval of 1800 seconds. 995 of these photons have been registered by CTA as events. In the same time interval, 5146 background events have been registred by CTA.

http://cta.irap.omp.eu/ctools/

How to use?

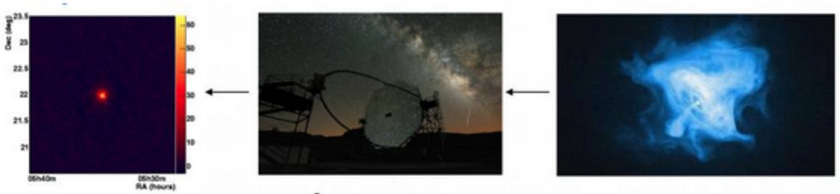
CTA Instrument Response Functions

Note

Instrument response functions are formally not part of ctools as they should be provided by the instrument teams. The GammaLib framework on which ctools are built comes with a set of basic response functions, but for getting the latest instrument response functions you should contact the relevant instrument teams. CTA consortium members should download the latest calibration database containing Prod1 and Prod2 instrument response functions from the consortium Sharepoint site (requires password).

What are instrument response functions?

The instrument response functions provide a mathematical description that links the measured quantities $\vec{d}$ of an event to the physical quantities $\vec{p}$ of the incident photon. The following figure illustrates this relationship:



$$e(\vec{d}) = \int d\vec{p}\, R(\vec{d}|\vec{p},\vec{a}) \times I(\vec{p})$$

$I(\vec{p})$ is the gamma-ray intensity arriving at Earth as a function of photon properties $\vec{p}$ (which usually are true photon energy, true photon incident direction, and true photon arrival time), while $e(\vec{d})$ is the expected event rate as function of event properties $\vec{d}$ (which usually are the measured photon energy, measured or reconstructued photon incident direction, and measured photon arrival time). The expected event rate is obtained by integrating the product of the instrumental response function $R(\vec{d}\,|\,\vec{p},\vec{a})$ and the emitted intensity $I(\vec{p})$ over the photon properties $\vec{p}$. The argument $\vec{a}$ in the response function comprises any auxiliary parameter on which the response function may depend on (e.g. pointing direction, triggered telescopes, optical efficiencies, atmospheric conditions, etc.). All these quantities and hence the instrument response function may depend on time.

http://cta.irap.omp.eu/ctools/user_manual/getting_started/response.html

Response Functions

CTA response functions

The instrument response functions for CTA are factorised into the effective area $A_{\mathrm{eff}}(d, p, E, t)$ (units cm^2), the point spread function $PSF(p'|d, p, E, t)$, and the energy dispersion $E_{\mathrm{disp}}(E'|d, p, E, t)$ following:

$$R(p', E', t'|d, p, E, t) = A_{\mathrm{eff}}(d, p, E, t) \times PSF(p'|d, p, E, t) \times E_{\mathrm{disp}}(E'|d, p, E, t)$$

ctools are shipped with response functions for the northern and southern arrays, and variants are available that have been optimised for exposure times of 0.5 hours, 5 hours and 50 hours. In total, the following six instrument response functions are available: `North_0.5h`, `North_5h`, `North_50h`, `South_0.5h`, `South_5h`, and `South_50h`.

Each response is stored in a single FITS file, and each component of the response factorisation is stored in a binary table of that FITS file. In addition, the response files contain an additional table that describes the background rate as function of energy and position in the field of view. An example of a CTA response file is shown below:

Index	Extension	Type	Dimension	View				
0	Primary	Image	0	Header	Image		Table	
1	EFFECTIVE AREA	Binary	6 cols X 1 rows	Header	Hist	Plot	All	Select
2	POINT SPREAD FUNCTION	Binary	10 cols X 1 rows	Header	Hist	Plot	All	Select
3	ENERGY DISPERSION	Binary	7 cols X 1 rows	Header	Hist	Plot	All	Select
4	BACKGROUND	Binary	7 cols X 1 rows	Header	Hist	Plot	All	Select

Each table in the response file is in a standardised format that is the one that is also used for the Fermi/LAT telescope. As an example, the effective area component of the response file is shown below. Response information is stored in a n-dimensional cube, and each axis of this cube is described by the lower and upper edges of the axis bins. In this example the effective area is stored as a 2D matrix with the first axis being energy and the second axis being offaxis angle. Effective area information is stored for true (`EFFAREA`) and reconstructed (`EFFAREA_RECO`) energy. Vector columns are used to store all information.

http://cta.irap.omp.eu/ctools/user_manual/getting_started/response.html

How to use?

Binning CTA data

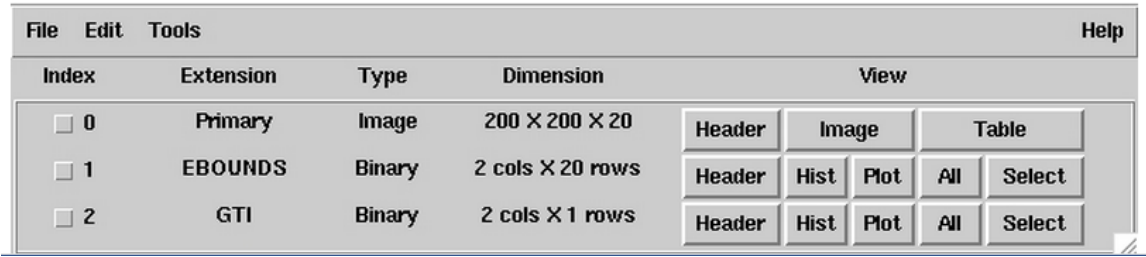
As next analysis step you will bin the data in a counts cube using the executable ctbin. A counts cube is a 3 dimensional data cube, spanned by Right Ascension (or Galactic longitude), Declination (or Galactic latitude), and the logarithm (base 10) of energy.

ctbin is executed by typing:

```
$ ctbin
Input event list or observation definition file [events.fits]
First coordinate of image center in degrees (RA or galactic l) [83.63]
Second coordinate of image center in degrees (DEC or galactic b) [22.01]
Projection method e.g. AIT|AZP|CAR|MER|STG|TAN (AIT|AZP|CAR|MER|STG|TAN) [CAR]
Coordinate system (CEL - celestial, GAL - galactic) (CEL|GAL) [CEL]
Image scale (in degrees/pixel) [0.02]
Size of the X axis in pixels [200]
Size of the Y axis in pixels [200]
Algorithm for defining energy bins (FILE|LIN|LOG) [LOG]
Start value for first energy bin in TeV [0.1]
Stop value for last energy bin in TeV [100.0]
Number of energy bins [20]
Output counts cube [cntmap.fits]
```

In this example we adjust the event data file name and accept all the remaining parameter defaults as they perfectly satisfy our needs. The counts cube will be centred on the location of the Crab (Right Ascension 83.63 degrees, Declination 22.01 degrees) and will be aligned in celestial coordinates. A cartesian projection has been selected. The counts cube has 200 x 200 spatial pixels of 0.02 x 0.02 degrees in size, hence it covers a total area of 4 x 4 degrees.

The counts cube will contain 20 maps, which are logarithmically spaced in energy, and which cover the energy range from 0.1 TeV to 100 TeV. In this example, the counts cube will be saved as `cntmap.fits` in the working directory. In addition to the counts cube, that is stored as the primary image extension, the FITS file also contains an extension named `EBOUNDS` that defines the energy boundaries that were used, and an extension `GTI` that defines the Good Time Intervals that have been used. The following image shows the resulting FITS file. The `EBOUNDS` table has 20 rows, one for each energy bin, while the `GTI` table has just a single row, indicating the start and stop time of the simulated data.



How to use?

Fitting CTA data

Now we are ready to fit the simulated data with a model. For simplicity we use in this example the same model that we used to simulate the data with ctobssim. Model fitting is done using the executable ctlike, and we do the fit by typing:

```
$ ctlike
Event list, counts cube or observation definition file [events.fits] cntmap.fits
Calibration database [dummy]
Instrument response function [cta_dummy_irf]
Source model [$CTOOLS/share/models/crab.xml]
Source model output file [crab_results.xml]
```

Fitting of the data is done in *binned* mode, which means that the events have been binned into a counts cube and the fit computes the log-likelihood function by summing over all 200 x 200 x 20 bins of the counts cube. There is an alternative method, the so called *unbinned* mode, where the events are not binned into a counts cube and the log-likelihood is computed directly by summing over all events. We will explore the *unbinned* mode later.

One of the parameters given to ctlike is a source model output file (we specified `crab_results.xml` in the example), and this file will be a copy of the model XML file where the parameter values have been replaced by the fit results. In addition, the statistical uncertainties are added for each fitted parameter using the attribute error. Below we show the XML result file that has been produced by the run:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<source_library title="source library">
  <source name="Crab" type="PointSource">
    <spectrum type="PowerLaw">
      <parameter name="Prefactor" value="6.07928" error="0.204582" scale="1e-16" min="1e-07" max="1000" free="
      <parameter name="Index" value="2.5009" error="0.0252057" scale="-1" min="0" max="5" free="1" />
      <parameter name="Scale" value="0.3" scale="1e+06" min="0.01" max="1000" free="0" />
    </spectrum>
    <spatialModel type="SkyDirFunction">
      <parameter name="RA" value="83.6331" scale="1" min="-360" max="360" free="0" />
      <parameter name="DEC" value="22.0145" scale="1" min="-90" max="90" free="0" />
    </spatialModel>
  </source>
```

Likelihood

Doing an unbinned analysis

As gamma-ray events are rare, the counts cubes generated by ctbin will in general be sparse, having many empty pixels, in particular at high energies. An alternative analysis technique consists of working directly on the event list without binning the events in a counts cube. We will see the benefit of such an analysis later once you re-run ctlike in unbinned mode.

For unbinned analysis you first have to define the data space region over which the analysis is done. This is similiar to the ctbin step in binned analysis where you defined the size of the counts cube, the energy range, and the time interval. For unbinned analysis you have no such thing as a counts cube, but you have to define over which region of the data space the selected events are spread (because the ctools have to integrate over this region to compute the total number of predicted events in the data space that you analyse). Furthermore, you have to define what energy range is covered, and what time interval is spanned by the data. All this is done by the executable ctselect, which replaces the ctbin step in an unbinned analysis.

ctselect performs an event selection by choosing only events within a given region-of-interest (ROI), within a given energy band, and within a given time interval from the input event list. The ROI is a circular region on the sky, for which you define the centre (in celestial coordinates) and the radius. Such a circular ROI is sometimes also called an acceptance cone. The following example shows how to run ctselect:

```
$ ctselect
Input event list or observation definition file [events.fits]
RA for ROI centre (degrees) (0-360) [83.63]
Dec for ROI centre (degrees) (-90-90) [22.01]
Radius of ROI (degrees) (0-180) [3.0]
Start time (CTA MET in seconds) (0) [0.0]
End time (CTA MET in seconds) (0) [0.0]
Lower energy limit (TeV) (0) [0.1]
Upper energy limit (TeV) (0) [100.0]
Output event list or observation definition file [selected_events.fits]
```

ctlike

```
$ ctlike
Event list, counts cube or observation definition file [cntmap.fits] selected_events.fits
Calibration database [dummy]
Instrument response function [cta_dummy_irf]
Source model [$CTOOLS/share/models/crab.xml]
Source model output file [crab_results.xml]
```

You will recognise that ctlike runs much faster in unbinned mode compared to binned mode. This is understandable as the selected event list contains only 6127 events, while the binned counts cube we used before had 200 x 200 x 20 = 800000 pixels. As unbinned maximum likelihood fitting loops over the events (while binned maximum likelihood loops over the pixels), there are much less operations to perform in unbinned than in binned mode (there is some additional overhead in unbinned mode that comes from integrating the models over the region of interest, yet this is negligible compared to the operations needed when looping over the pixels). So as long as you work with short event lists, unbinned mode is faster. Unbinned ctlike should also be more precise as no binning is performed, hence there is no loss of information due to histogramming.

TS calculation

Note

The *ctlike* tool has the ability to estimate the detection significance for sources in the XML model. This is done by computing the Test Statistic value which is defined as twice the log-likelihood difference between fitting a source at a given position on top of a (background) model or fitting no source. Roughly speaken, the square root of the Test Statistic value gives the source detection significance in Gaussian sigmas, although the exact relation depends somewhat on the formulation of the statistical problem.

To instruct *ctlike* to compute the Test Statistic value for a given source you need to add the attribute `tscalc="1"` to the XML file:

```
<source name="Crab" type="PointSource" tscalc="1">
```

ctlike will then compute the Test Statistic value for that source and dump the result in the log file:

```
2015-05-22T19:58:43: === GModelSky ===
2015-05-22T19:58:43:  Name ......................: Crab
2015-05-22T19:58:43:  Instruments ...............: all
2015-05-22T19:58:43:  Test Statistic ............: 18662.6
```

The Test Statistic value will also be added as new attribute `ts` to the XML result file:

```
<source name="Crab" type="PointSource" ts="18662.576" tscalc="1">
```

Butterfly

Calculate and visualise butterfly

To visualise the analysis results retrieved above, one can calculate the confidence band of the spectral fit. The tool ctbutterfly takes the optimised source model as input. It takes the covariance matrix from the fit to conduct a Gaussian error propagation for each energy value. It will write the butterfly information into an ASCII file. The following example shows how to compute such a butterfly from the command line.

```
$ ctbutterfly
Input event list, cube or observation definition file [events.fits]
Calibration database [dummy]
Instrument response function [cta_dummy_irf]
Source model [$CTOOLS/share/models/crab.xml] crab_results.xml
Source of interest [Crab]
Start value for first energy bin in TeV [0.1]
Stop value for last energy bin in TeV [100.0]
Output ascii file [butterfly.txt]
```

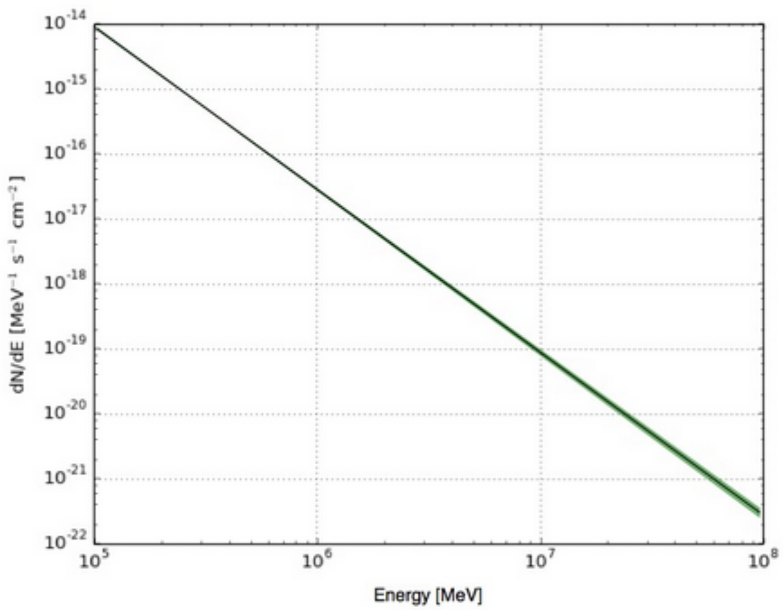
Below some lines of the `ctbutterfly.log`:

```
2014-10-30T17:51:39: +===========================+
2014-10-30T17:51:39: | Compute covariance matrix |
2014-10-30T17:51:39: +===========================+
2014-10-30T17:51:39:
2014-10-30T17:51:39: +===================+
2014-10-30T17:51:39: | Generate butterfly |
2014-10-30T17:51:39: +===================+
2014-10-30T17:51:39:
2014-10-30T17:51:39: +=======================+
2014-10-30T17:51:39: | Save Butterfly to file |
2014-10-30T17:51:39: +=======================+
2014-10-30T17:51:39:
2014-10-30T17:51:39: Application "ctbutterfly" terminated after 15 wall clock seconds, consuming 0.051253
```

Butterfly

```
python $CTOOLS/share/examples/python/show_butterfly.py butterfly.txt
```

This will result in a canvas which should look like the following:



Confidence band of the fit

ctbutterfly

Purpose: produce a butterfly diagram for a spectral model M

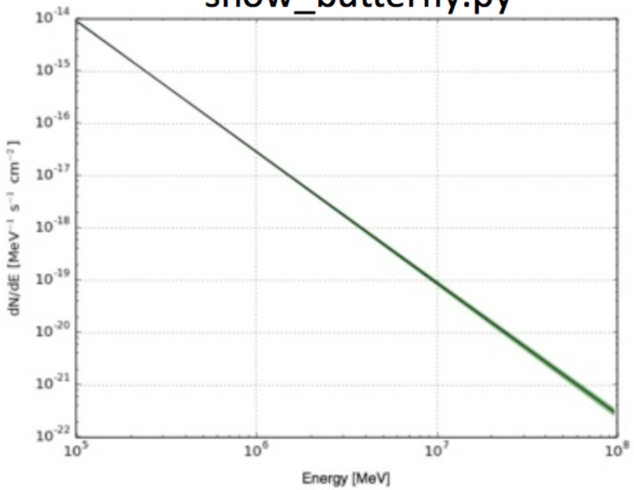
$$\underbrace{\sigma(E)}_{\text{flux error}} = \sqrt{\sum_{i,k} \underbrace{\frac{\delta M(E)}{\delta p_i}}_{\text{spectral model gradient}} \underbrace{\left(\frac{\delta^2 \ln \mathcal{L}(E)}{\delta p_i \delta p_k}\right)^{-1}}_{\text{covariance matrix}} \underbrace{\frac{\delta M(E)}{\delta p_k}}_{\text{spectral model gradient}}}$$

```
#
# General parameters
#===================
inobs,    f, a, "events.fits",,,"Input event list, counts cube or observation definition file"
inmodel,  f, a, "$CTOOLS/share/models/crab.xml",,, "Source model"
srcname,  s, a, "Crab",,,"Source of interest"
expcube,  f, a, "NONE",,, "Exposure cube file (only needed for stacked analysis)"
psfcube,  f, a, "NONE",,, "PSF cube file (only needed for stacked analysis)"
bkgcube,  f, a, "NONE",,, "Background cube file (only needed for stacked analysis)"
caldb,    s, a, "prod2",,, "Calibration database"
irf,      s, a, "South_50h",,, "Instrument response function"
outfile,  f, a, "butterfly.txt",,, "Output ascii file"
matrix,   f, h, "NONE",,, "Input covariance Matrix FITS file"

#
# Energy binning parameters
#==========================
ebinalg,  s, h, "LOG", FILE|LIN|LOG,,"Algorithm for defining energy bins"
emin,     r, a, 0.1,,,"Start value for first energy bin in TeV"
emax,     r, a, 100.0,,,"Stop value for last energy bin in TeV"
enumbins, i, h, 100,,,"Number of energy bins"
ebinfile, f, h, "NONE",,,"Name of the file containing the energy bin definition"

#
# Standard parameters
#====================
chatter, i, h, 2,0,4, "Chattiness of output"
clobber, b, h, yes,,, "Overwrite existing output files with new output files?"
debug,   b, h, no,,, "Debugging mode activated"
mode,    s, h, "ql",,, "Mode of automatic parameters"
logfile, f, h, "ctbutterfly.log",,, "Log filename"
```

show_butterfly.py



Further steps

ctools
cherenkov telescope array

CTA
Cherenkov Telescope Array Science Analysis Software

Home Download

Home | Documentation » User Manual » Getting started »

Beyond the first steps

Assuming that you have read the *quickstart* tutorial, here now some chapters to read if you intend to do more complex analyses. Once you have gone through these chapters you should have a pretty complete picture about the capabilities of ctools. And you should be ready for doing your own cutting edge analyses.

- Combining observations
- Performing a stacked analysis
- Generating a Test Statistic map
- Generating a residual map
- Connecting observations to dedicated instrument response functions
- Connecting observations to specific models

cttsmap

Purpose: produce a Test Statistics map

$$TS(\alpha,\delta) = 2\left(\ln \mathcal{L}(\alpha,\delta) - \ln \mathcal{L}_0\right)$$

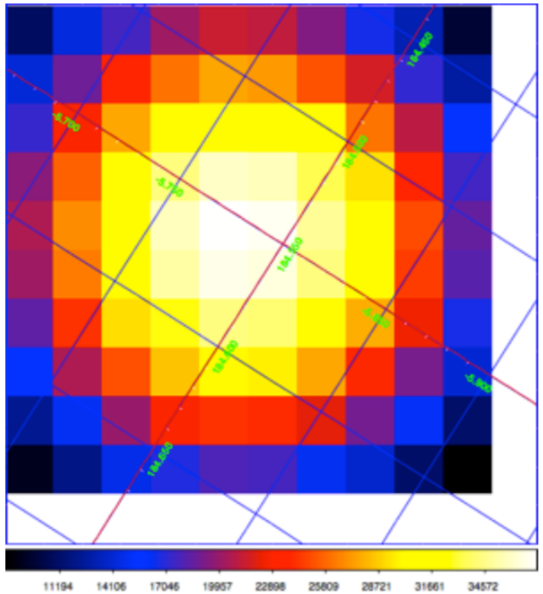
model with source located at α, δ — model without source

```
#
# General parameters
#====================
inobs,   f, a, "events.fits",,, "Input event list, counts cube or observation definition file"
inmodel, f, a, "$CTOOLS/share/models/crab.xml",,, "Source model"
srcname, s, a, "Crab",,, "Test source"
expcube, f, a, "NONE",,, "Exposure cube file (only needed for stacked analysis)"
psfcube, f, a, "NONE",,, "PSF cube file (only needed for stacked analysis)"
bkgcube, f, a, "NONE",,, "Background cube file (only needed for stacked analysis)"
caldb,   s, a, "prod2",,, "Calibration database"
irf,     s, a, "South_50h",,, "Instrument response function"
outmap,  f, a, "tsmap.fits",,, "Output Test Statistic map"

#
# Spatial binning parameters
#============================
usepnt,   b, h, no,,, "Use pointing instead of xref/yref parameters?"
nxpix,    i, a, 200,,, "Size of the X axis in pixels"
nypix,    i, a, 200,,, "Size of the Y axis in pixels"
binsz,    r, a, 0.02,,, "Image scale (in degrees/pixel)"
coordsys, s, a, "CEL", CEL|GAL,,"Coordinate system (CEL - celestial, GAL - galactic)"
xref,     r, a, 83.63,0,360, "First coordinate of image center in degrees (RA or galactic l)"
yref,     r, a, 22.01,-90,90, "Second coordinate of image center in degrees (DEC or galactic b)"
proj,     s, a, "CAR", AIT|AZP|CAR|MER|MOL|STG|TAN,, "Projection method"

#
# Parameters for splitting and speed purpose
#============================
binmin,   i, h, -1,,, "First bin to compute"
binmax,   i, h, -1,,, "Last bin to compute"
logL0,    r, h, 0.0,,, "LogLikelihood value of null hypothesis"


#
# Standard parameters
#====================
chatter, i, h, 2,0,4, "Chattiness of output"
clobber, b, h, yes,,, "Overwrite existing output files with new output files?"
debug,   b, h, no,,, "Debugging mode activated"
mode,    s, h, "ql",,, "Mode of automatic parameters"
logfile, f, h, "cttsmap.log",,, "Log filename"
```



cscripts

cscripts

- cscaldb — Lists available instrument response functions
- csobsdef — Generates observation definition file
- cslightcrv — Computes lightcurve
- cspull — Generates pull distribution
- cssens — Computes CTA sensitivity
- csspec — Computes spectral points
- csresmap — Generates residual map
- cstsdist — Generates TS distribution

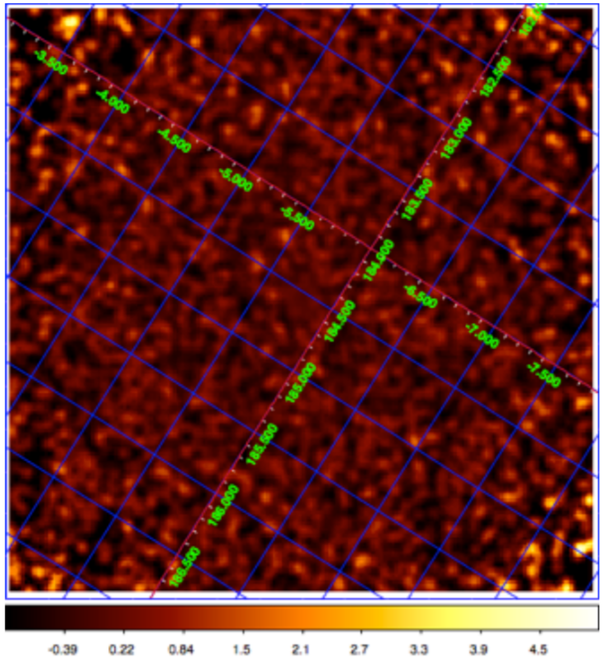
csresmap

Purpose: produce a residual counts map*

counts — model

$$\mathrm{R}(\alpha, \delta) = \frac{\sum_i \mathrm{N}(\alpha, \delta, E_i) - \mathrm{M}(\alpha, \delta, E_i)}{\sum_i \mathrm{M}(\alpha, \delta, E_i)}$$

residual map

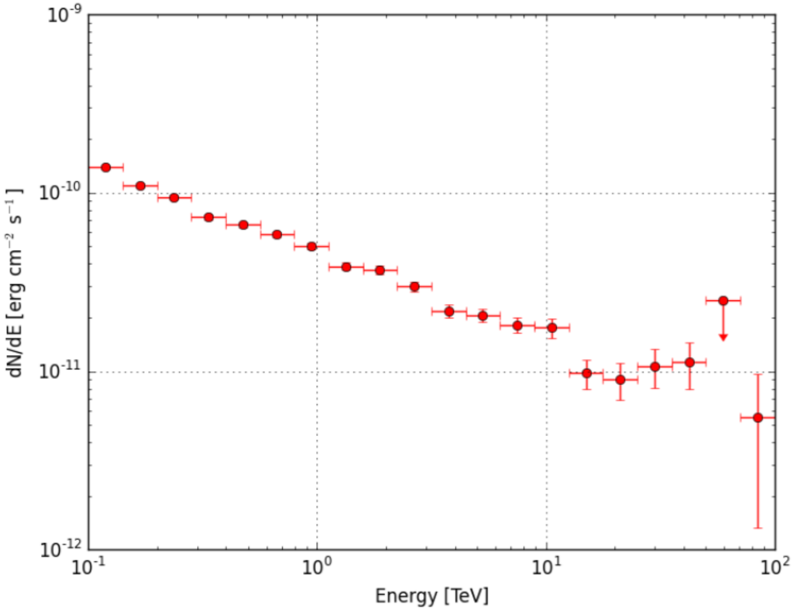


*does not yet work for stacked analysis

csspec

Purpose: runs ctlike in various energy bands to produce a spectrum (including upper limits)

```
$ csspec
Parfile csspec.par not found. Create default parfile.
Event list, counts cube, or observation definition file [events.fits] obs.xml
Source model [$CTOOLS/share/models/crab.xml]
Source name [Crab]
Number of spectral points [20]
Use binned analysis in each energy bin (yes|no) [no]
Output file name [spectrum.fits]
```



show_spectrum.py