GammaLib and ctatools development status report

Jürgen Knödlseder
Institut de Recherche en Astrophysique et Planétologie (IRAP)

What is GammaLib ?

A self-contained, instrument independent, open source, multi-platform C++ library that implements all code required for high-level science analysis of astronomical gamma-ray data.

Self-contained: does not depend on external libraries*
*only the cfitsio library is required for FITS file access

Instrument independent: potentially supports any gamma-ray astronomy instrument; enables simultaneous multi-instrument analysis

Open source: latest code on http://sourceforge.net/projects/gammalib/

Multi-platform: (should) compile on any POSIX Unix platform*
*tested: Mac OS X 10.4, Redhat 9, Centos 5 (64 Bit), gentoo (64 Bit)

C++ library: makes heavily use of C++ classes

How does it work ?

Implement required functionnalities as C++ classes. Examples of C++ classes are:

GEnergy – unit independent energy value
GTime – time reference independent time value
GSkyDir – direction of sky (implicit conversion celestial/galactic)
GSkymap – skymap(s)
GFits – representation of FITS file
GXml – representation of XML file
GObservations – collection of observations / runs
GModel – Model describing a gamma-ray source or background
GMatrix – Matrix (supports, e.g., solution of linear equation)
GVector – Vector
GIntegral – Integration of functions
...

How does it work ?

Use abstract virtual base classes to define an instrument independent interface to instrument specific code. Examples of C++ classes are:

- GObservation – interface to a single observation / run
- GResponse – interface to response function
- GEventAtom – interface to an event (unbinned analysis)
- GEventBin – interface to an event bin (binned analysis)
- …

Implement derived classes for each instrument to implement the instrument specific code. Examples of C++ classes are:

- GCTAObservation – interface to a single CTA observation / run
- GCTAResponse – interface to CTA response function
- GCTAEventAtom – interface to an event (unbinned analysis)
- GCTAEventBin – interface to an event bin (binned analysis)
- …

How do I use it ?

Build a C++ application:

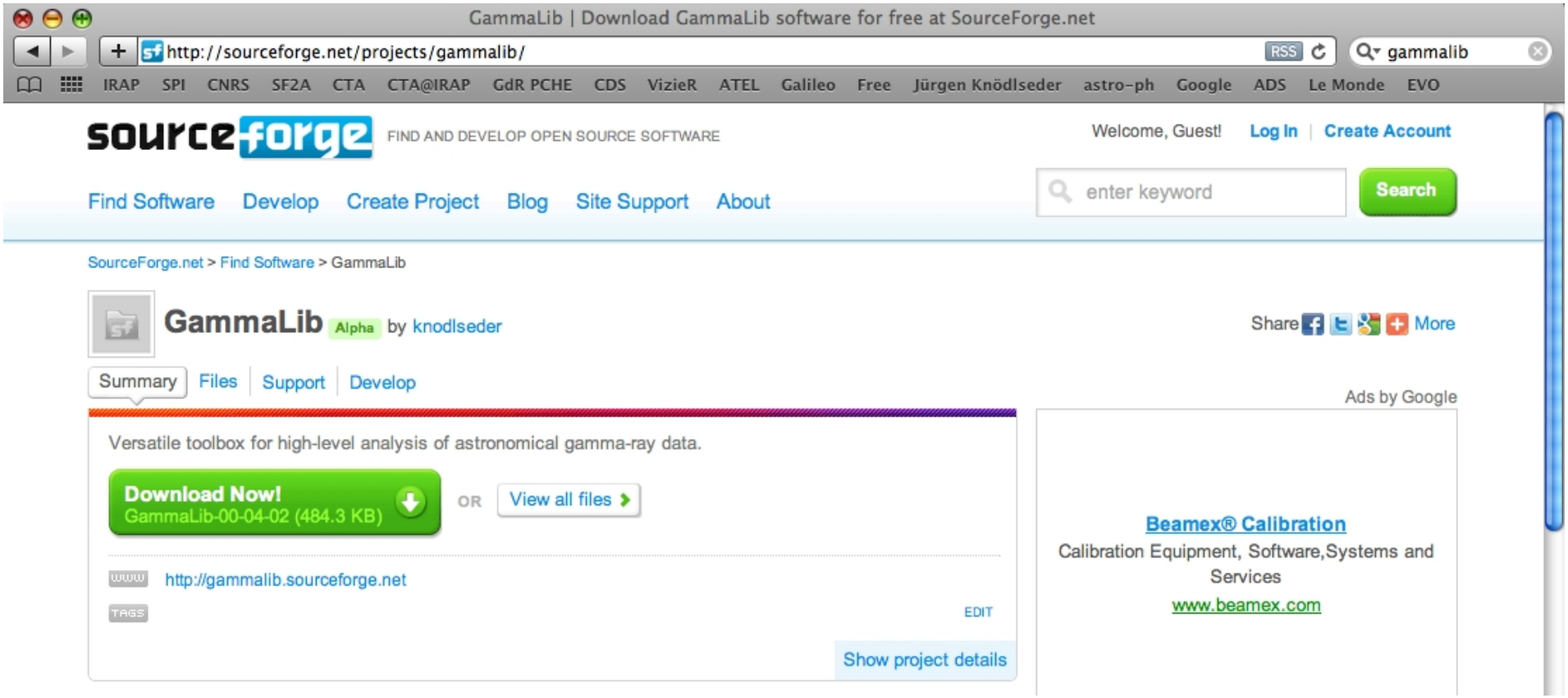
```
#include <iostream>
#include "GammaLib.hpp"
int main(void) {
    GObservations obs;
    std::cout << obs << std::endl;
    return 0;
}
```

Use python:

```
$ python
>>> from gammalib import *
>>> obs = GObservations()
>>> print obs
=== GObservations ===
 Number of observations ....: 0
 Number of predicted events : 0
=== GModels ===
 Number of models ..........: 0
 Number of parameters ......: 0
```

How do I get it ?

From here (latest version is GammaLib-00-04-04):



… or send me an e-mail (knodlseder@cesr.fr) or wait until the IRAP CTA www page gets online (within the next weeks).

How do I install it ?

After installing cfitsio (and swig if Python bindings are required):

1.4 Installing GammaLib

GammaLib is built on Unix systems by typing:

```
> ./autogen.sh
> ./configure
> make
> make install
```

Please try and report your feedback by specifying your machine type, your system, and the GammaLib version.

Test your installation:

1.5 Testing GammaLib

GammaLib should be tested by typing:

```
> make check
```

This will execute an extensive testing suite that should terminate with

```
==================
All 14 tests passed
==================
```

Also check http://gammalib.sourceforge.net/ for more information and documentation

Progress since last DAFA meeting

Actual version: GammaLib-00-04-04

Major changes since last summer:

- Add application logger (reporting interface)
- Add XML file support
- Add MWL instrument interface
- Implement Fermi/LAT instrument interface for binned analysis

Application logger

GammaLib provides support to build C++ applications:

Specification of application parameters through IRAF parameter files

New: Reporting through application logger (screen and/or log file)

Example:

```
// Get standard parameters
m_method = toupper(par("method")->value());
m_stat   = toupper(par("stat")->value());
m_refit  = par("refit")->boolean();
m_caldb  = par("caldb")->value();
m_irf    = par("irf")->value();
m_srcmdl = par("srcmdl")->value();
m_outmdl = par("outmdl")->value();

// Write parameters into logger
if (logTerse()) {
    log_parameters();
    log << std::endl;
}

// Write results into logger
if (logTerse()) {
    log << " Maximum log likelihood ....: ";
    log << m_logL << std::endl;
    log << " Npred .....................: ";
    log << m_obs.npred() << std::endl;
    log << m_models << std::endl << std::endl;
}
```

Writes text output in a standardized and formatted way onto screen and/or into a log file:

```
2010-12-19T23:40:57: +============+
2010-12-19T23:40:57: | Parameters |
2010-12-19T23:40:57: +============+
2010-12-19T23:40:57:  evfile ...................: eventlist.fits
2010-12-19T23:40:57:  cntmap ...................: cntmap.fits
2010-12-19T23:40:57:  method ...................: BINNED
2010-12-19T23:40:57:  stat .....................: POISSON
2010-12-19T23:40:57:  refit ....................: no
2010-12-19T23:40:57:  caldb ....................: irf
2010-12-19T23:40:57:  irf ......................: kb_E_50h_v3
2010-12-19T23:40:57:  srcmdl ...................: source.xml
2010-12-19T23:40:57:  outmdl ...................: results.xml
2010-12-19T23:40:57:  chatter ..................: 2
2010-12-19T23:40:57:  clobber ..................: yes
2010-12-19T23:40:57:  debug ....................: no
2010-12-19T23:40:57:  mode .....................: ql
```

XML file support

Use XML file to define a source model (Fermi/LAT format implemented):

```
<source_library title="source library">
  <source type="DiffuseSource" name="Extragal_diffuse" instrument="LAT">
    <spectrum type="FileFunction" file="month/isotropic_iem_v02.txt">
      <parameter scale="1.0" name="Normalization" min="0.0" max="1000.0" value="1.0" free="1"/>
    </spectrum>
    <spatialModel type="ConstantValue">
      <parameter scale="1" name="Value" min="0" max="10" value="1" free="0"/>
    </spatialModel>
  </source>

  <source type="DiffuseSource" name="Galactic_diffuse" instrument="LAT">
    <spectrum type="ConstantValue">
      <parameter scale="1.0" name="Value" min="0.0" max="1000.0" value="1.0" free="1"/>
    </spectrum>
    <spatialModel type="MapCubeFunction" file="/project-data/glast/skymaps/galprop/gll_iem_v02.fit">
      <parameter scale="1" name="Normalization" min="0.001" max="1000.0" value="1" free="0"/>
    </spatialModel>
  </source>

  <source type="PointSource" name="Crab">
    <spectrum type="PowerLaw2">
      <parameter scale="1e-07" name="Integral" min="1e-05" max="1000.0" value="1.0" free="1"/>
      <parameter scale="1.0" name="Index" min="-5.0" max="-0.1" 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>
    <spatialModel type="SkyDirFunction">
      <parameter scale="1" name="RA" min="-360" max="360" value="83.6331" free="0"/>
      <parameter scale="1" name="DEC" min="-90" max="90" value="22.0145" free="0"/>
    </spatialModel>
  </source>
</source_library>
```

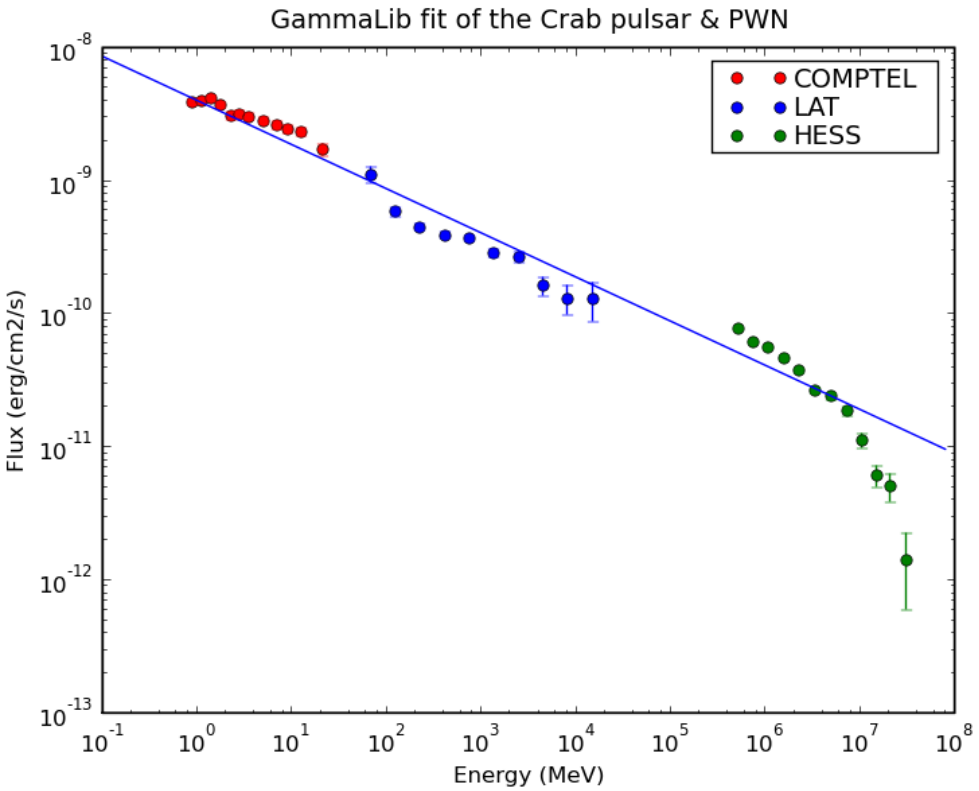
XML file support

Define a source model is just one line of code ...

```
$ python
>>> from gammalib import *
>>> models = GModels("data/source_model.xml")
>>> print models
=== GModels ===
 Number of models ..........: 3
 Number of parameters ......: 13
 Model name ................: Extragal_diffuse
 Model type ................: Constant FileFunction ConstantValue
 Instruments ...............: LAT
  Value ....................: 1 [0,10]  (fixed,scale=1)
  Normalization ............: 1 +/- 0 [0,1000]  (free,scale=1)
  Constant .................: 1 (relative value) (fixed,scale=1)
 Model name ................: Galactic_diffuse
 Model type ................: Constant ConstantValue MapCubeFunction
 Instruments ...............: LAT
  Normalization ............: 1 [0.001,1000]  (fixed,scale=1)
  Value ....................: 1 +/- 0 [0,1000]  (free,scale=1)
  Constant .................: 1 (relative value) (fixed,scale=1)
 Model name ................: Crab
 Model type ................: Constant PowerLaw2 PointSource
 Instruments ...............: all
  RA .......................: 83.6331 deg (fixed,scale=1)
  DEC ......................: 22.0145 deg (fixed,scale=1)
  Integral .................: 1e-07 +/- 0 [1e-12,0.0001] ph/cm2/s (free,scale=1e-07)
  Index ....................: -2 +/- 0 [-5,-0.1]  (free,scale=1)
  LowerLimit ...............: 100 [10,1e+06] MeV (fixed,scale=1)
  UpperLimit ...............: 500000 [10,1e+06] MeV (fixed,scale=1)
  Constant .................: 1 (relative value) (fixed,scale=1)
```

Multi-wavelength support

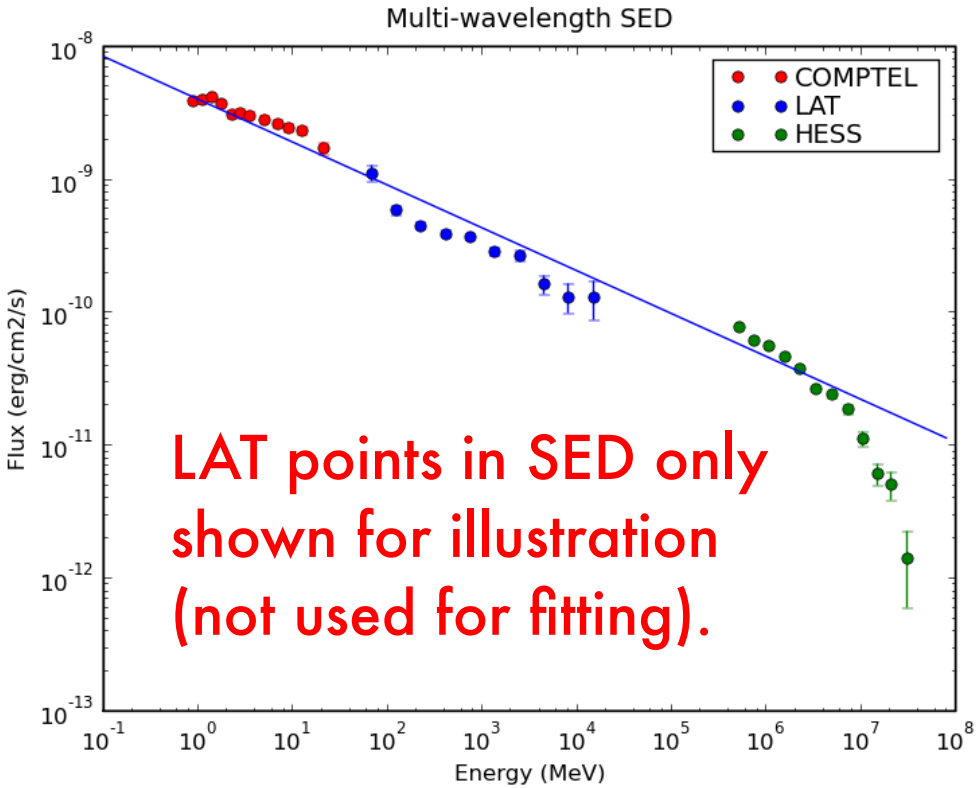
Extend GammaLib beyond gamma-rays by treating any SED (table of flux versus energy) as a special multi-wavelength instrument interface (MWL). Example: fitting the Crab from MeV to TeV energies:

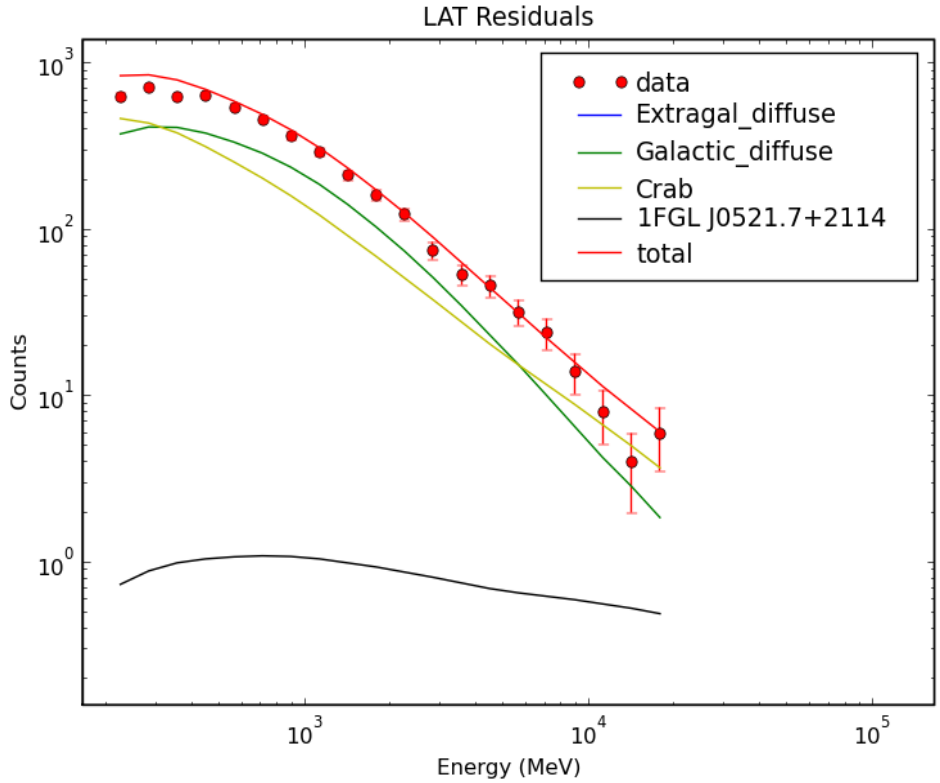


Please don't take this spectral fit seriously!
A simple power law has been fitted for illustrative purposes only.
Adequate fitting of the Crab would require (at least) 3 spectral components: pulsar, PWN synchrotron, PWN inverse Compton.

Fermi/LAT support (binned)

Implementation of Fermi/LAT interface for binned analysis (GammaLib can now be used instead of gtlike). Example: replacing the LAT SED by LAT data (allows proper treatment of diffuse emission and nearby sources):





CTA interface status and next steps

CTA prototype interface implemented for binned and unbinned analysis.

Limitations:

- so far handles only point sources (PSF approximated by Gaussian)
- no zenith angle or offset angle dependence implemented
- no background handling implemented

Next CTA related steps:

- implement angular dependencies in response function
- implement energy redistribution
- improve PSF prescription (e.g., double Gaussian)
- implement background handling / modelling

Next GammaLib steps:

- implement event simulator
- implement more complex source models

What are the ctatools?

ctatools is a set of high-level analysis executables used to analyse CTA data. They make use of GammaLib, and are inspired by tools existing for other open high-energy observatories (e.g., Fermi/LAT, INTEGRAL), and follow standards developed by HEASARC (implemented, e.g., in ftools). Existing tools are:

ctselect – Select events from CTA event file (region of interest, time range, energy range); required for unbinned analysis

ctbin – Bin CTA event file in a data cube used for binned analysis

ctlike – Maximum likelihood fitting of source & background models (binned & unbinned)

ctselect – Select events

Event selection using acceptance cone (RA/DEC, radius), energy range, and time range (required for unbinned analysis).

Log file (excerpt):

```
+=============+
| Parameters |
+=============+
 infile ....................: events.fits
 outfile ...................: evfile.fits
 ra ........................: 117.02
 dec .......................: -33.35
 rad .......................: 2.5
 tmin ......................: 0.0
 tmax ......................: 10000.0
 emin ......................: 0.02
 emax ......................: 100.0
 chatter ...................: 3
 clobber ...................: yes
 debug .....................: no
 mode ......................: ql

+=================+
| Event selection |
+=================+
 Energy range ..............: 0.02-100 TeV
 Acceptance cone centre ....: RA=117.02, DEC=-33.35 deg
 Acceptance cone radius ....: 2.5 deg
 cfitsio selection .........: ENERGY >= 0.02 && ENERGY <= 100 && ANGSEP(117.02,-33.35,RA,DEC) <= 2.5
 FITS filename .............: events.fits[EVENTS][ENERGY >= 0.02 && ENERGY <= 100 && ANGSEP(117.02,-33.35,RA,DEC) <= 2.5]
```

Parameter file:

```
infile,  f, a, "events.fits",,,   "Input event data file"
Outfile, f, a, "evfile.fits",,,   "Output event data file"
ra,      r, a,   117.02,   0,360, "RA for ROI centre (deg)"
dec,     r, a,   -33.35, -90, 90, "Dec for ROI centre (deg)"
rad,     r, a,     2.5,    0,180, "Radius of ROI (deg)"
tmin,    r, a,     0.0,    0,   , "Start time (MET in s)"
tmax,    r, a, 10000.0,    0,   , "End time (MET in s)"
emin,    r, a,     0.02,   0,   , "Lower energy limit (TeV)"
emax,    r, a,   100.0,    0,   , "Upper energy limit (TeV)"
chatter, i, h,     3,      0,  4, "Chattiness of output"
clobber, b, h,   yes,      ,    , "Overwrite existing files?"
debug,   b, h,    no,      ,    , "Debugging mode activated?"
mode,    s, h,  "ql",      ,    , "Mode of automatic parameters"
```

ctbin – Bin events into counts map

Event selection using acceptance cone (RA/DEC, radius), energy range, and time range (required for unbinned analysis).

Log file (excerpt):

```
=== GCTAObservation ===
 Name .....................:
 Instrument ...............: CTA
 Statistics ...............: Poisson
 Time range ...............: 51910 - 51910.1 days
 Energy range .............: 20 GeV - 100 TeV
 Region of interest .......: undefined
 CTA response .............: undefined
=== GCTAEventList ===
 Number of events .........: 8510

=== GSkymap ===
 Number of pixels .........: 40000
 Number of maps ...........: 20
 X axis dimension .........: 200
 Y axis dimension .........: 200
=== GWcsCAR ===
 Coordinate system ........: EQU
 Reference coordinate .....: (117.02, -33.35) deg
 Reference pixel ..........: (100.5, 100.5)
 Increment at reference ...: (0.02, 0.02) deg
 Coordinate of North Pole ..: (180, 0) deg
 CD matrix ................: [0.02 0][0 0.02]
 Inverse CD matrix ........: [50 -0][-0 50]
 Coordinate of native pole .: (117.02, 56.65)
```

Parameter file:

```
evfile,    f, a, "events.fits",,, "Event data file name"
outfile,   f, a, "cntmap.fits",,, "Output file name"
ebinalg,   s, a, "LOG", FILE|LIN|LOG,, "Energy bins"
emin,      r, a, 0.02,,, "Start value for first energy bin in TeV"
emax,      r, a, 100.0,,, "Stop value for last energy bin in TeV"
enumbins,  i, a, 20,,, "Number of energy bins"
ebinfile,  f, a, "NONE",,, "Energy bin definition file (optional)"
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"
xref,      r, a, 117.02,,, "First coordinate of image center"
yref,      r, a, -33.35,,, "Second coordinate of image center"
axisrot,   r, a, 0.0,,, "Rotation angle of image axis, in degrees"
proj,      s, a, "CAR",,, "Projection method"
chatter,   i, h, 1,0,4, "Chattiness of output"
clobber,   b, h, yes,,, "Overwrite existing file?"
debug,     b, h, no,,, "Debugging mode activated?"
mode,      s, h, "ql",,, "Mode of automatic parameters"
```

ctlike – Maximum likelihood fitting

Adjusts a parametrized source model defined by an XML file to CTA data using either binned or unbinned maximum likelihood fitting.

Log file (excerpt):

```
+=========================================+
| Maximum likelihood optimisation results |
+=========================================+
=== GOptimizerLM ===
 Optimized function value ..: 344486
 Absolute precision ........: 1e-06
 Optimization status .......: converged
 Number of parameters ......: 6
 Number of free parameters .: 2
 Number of iterations ......: 611
 Lambda ....................: 0.002
 Maximum log likelihood ....: -344486
 Npred .....................: 8295.97
=== GModels ===
 Number of models ..........: 1
 Number of parameters ......: 6
 Model name ................: Source1
 Model type ................: Constant PowerLaw PointSource
  RA .......................: 117 deg (fixed,scale=1)
  DEC ......................: -33 deg (fixed,scale=1)
  Prefactor ................: 2.12933e-08 +/- 1.78219e-09 [1e-14,0.0001] ph/cm2/s/MeV (free,scale=1e-07)
  Index ....................: -2.1413 +/- 0.0100224 [-5,5]  (free,scale=1)
  PivotEnergy ..............: 100 [10,1e+06] MeV (fixed,scale=1)
  Constant .................: 1 (relative value) (fixed,scale=1)
```

Parameter file:

```
evfile, f,a,"events.fits",,, "Event data (for unbinned)"
cntmap, f,a,"cntmap.fits",,, "Counts map (for binned)"
method, s,a,"UNBINNED",,, "Maximum likelihood method"
stat,   s,a,"POISSON",,, "Optimization statistics"
refit,  b,a,no,,, "Do refitting?"
caldb,  s,a,"irf",,, "Calibration database"
irf,    s,a,"kb_E_50h_v3",,, "Instrument response"
srcmdl, f,a,"model.xml",,, "Source model"
outmdl, f,h,"results.xml",,, "Source model result"
chatter,i,h,2,0,4, "Chattiness of output"
clobber,b,h,yes,,, "Overwrite existing files?"
debug,  b,h,no,,, "Debugging mode activated"
mode,   s,h,"ql",,, "Mode of automatic parameters"
```

ctatools – next steps

Add missing tools to have a first toy high-level analysis system for the CTA consortium to play with:

ctobssim – simulate CTA observation (Monte Carlo based on response function and specified source model; not a full shower / array simulation)

ctbkgmod – compute CTA background model to be used in likelihood analysis