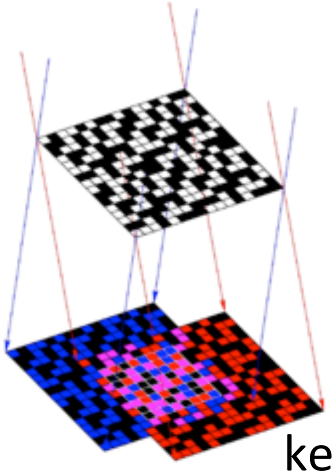
4th Coding Sprint

1. Short intro into GammaLib and ctools concepts
2. New features since last coding sprint
3. Instrument Response Functions
4. Goals of this sprint

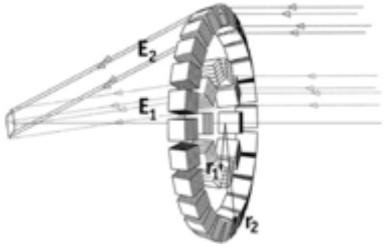
Jürgen Knödlseder (IRAP)

1. Short intro into GammaLib and ctools concepts

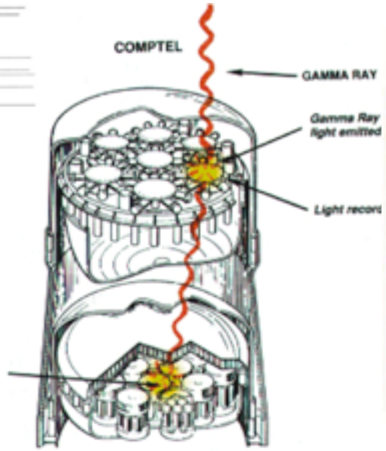
Concept

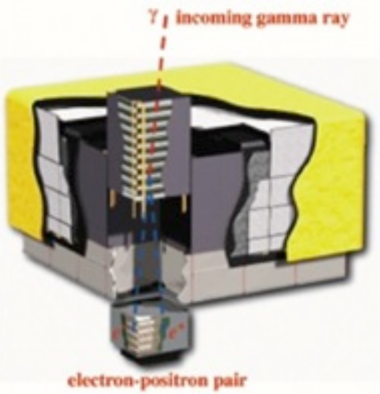


keV

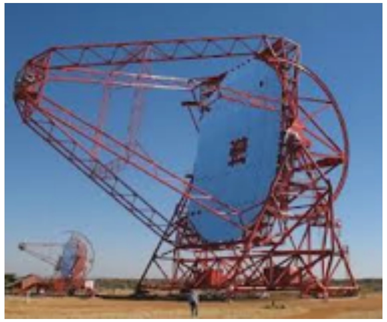


MeV





GeV



TeV

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).

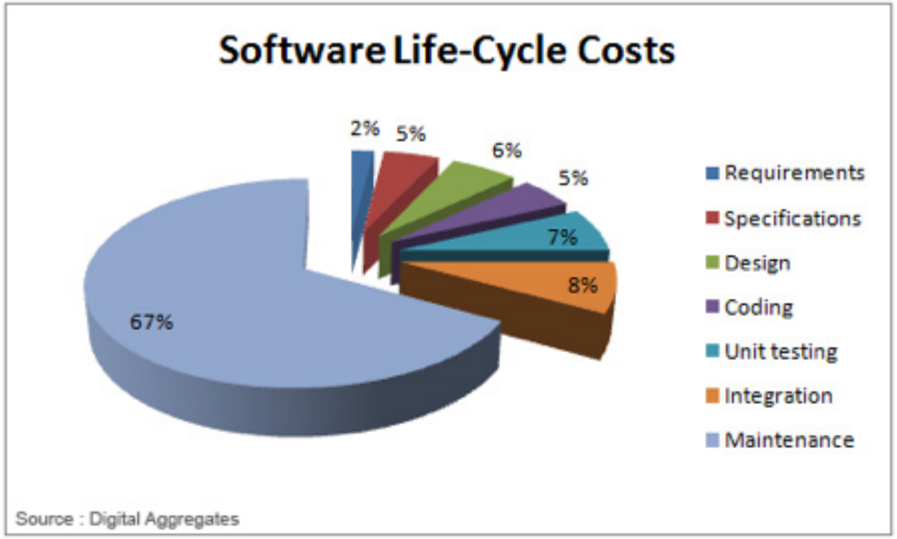
ctools cherenkov telescope array … is the client that uses the bricks provided by γLib

CTA specific *generic*

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

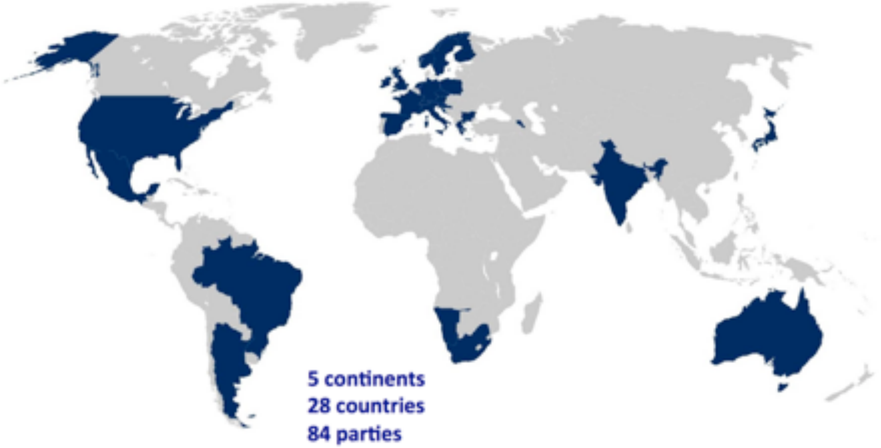
Design considerations

Minimise maintenance costs and maximise community involvement



Source : Digital Aggregates

CTA Consortium

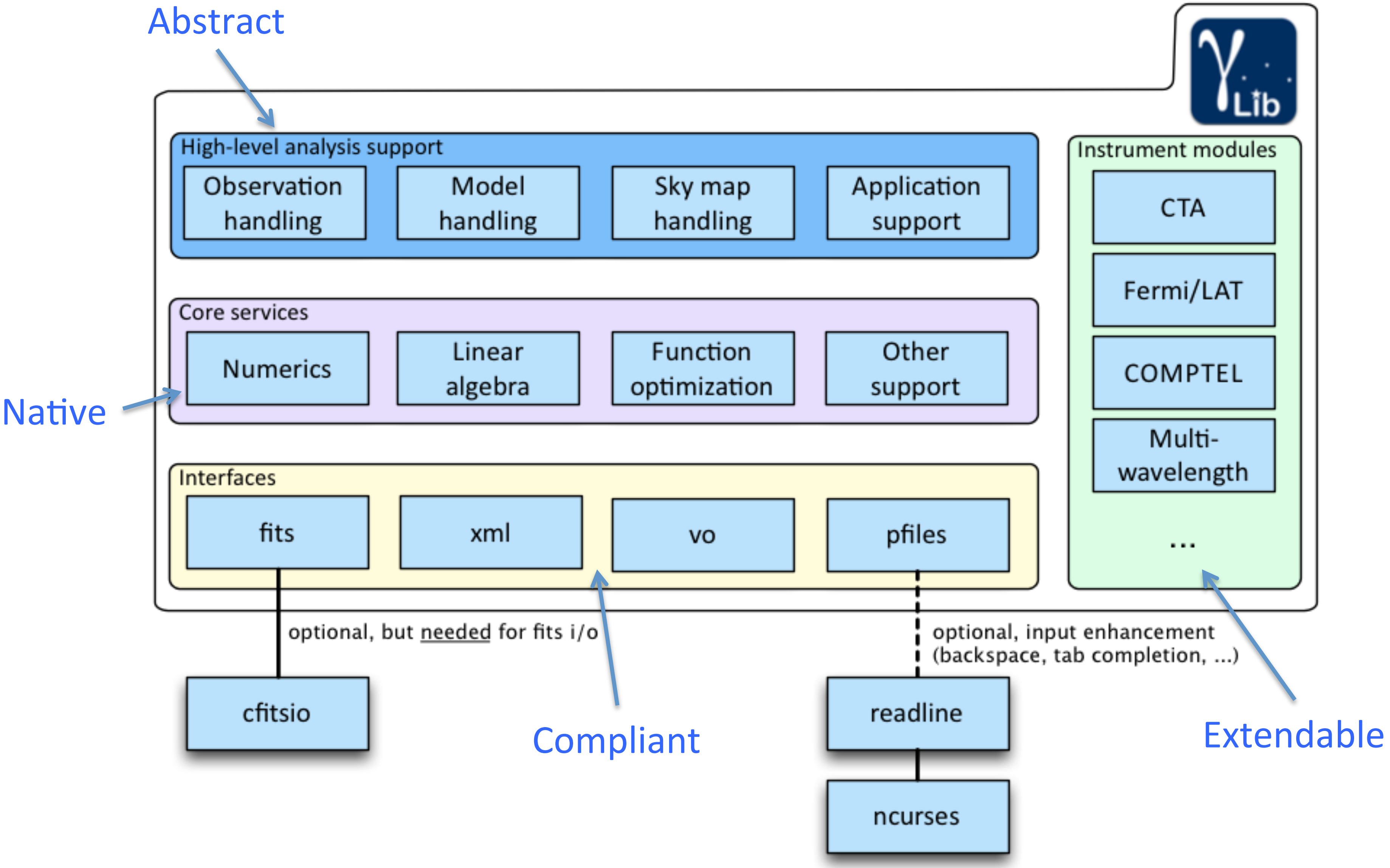


5 continents
28 countries
84 parties
173 institutes
1178 members (378 FTE)

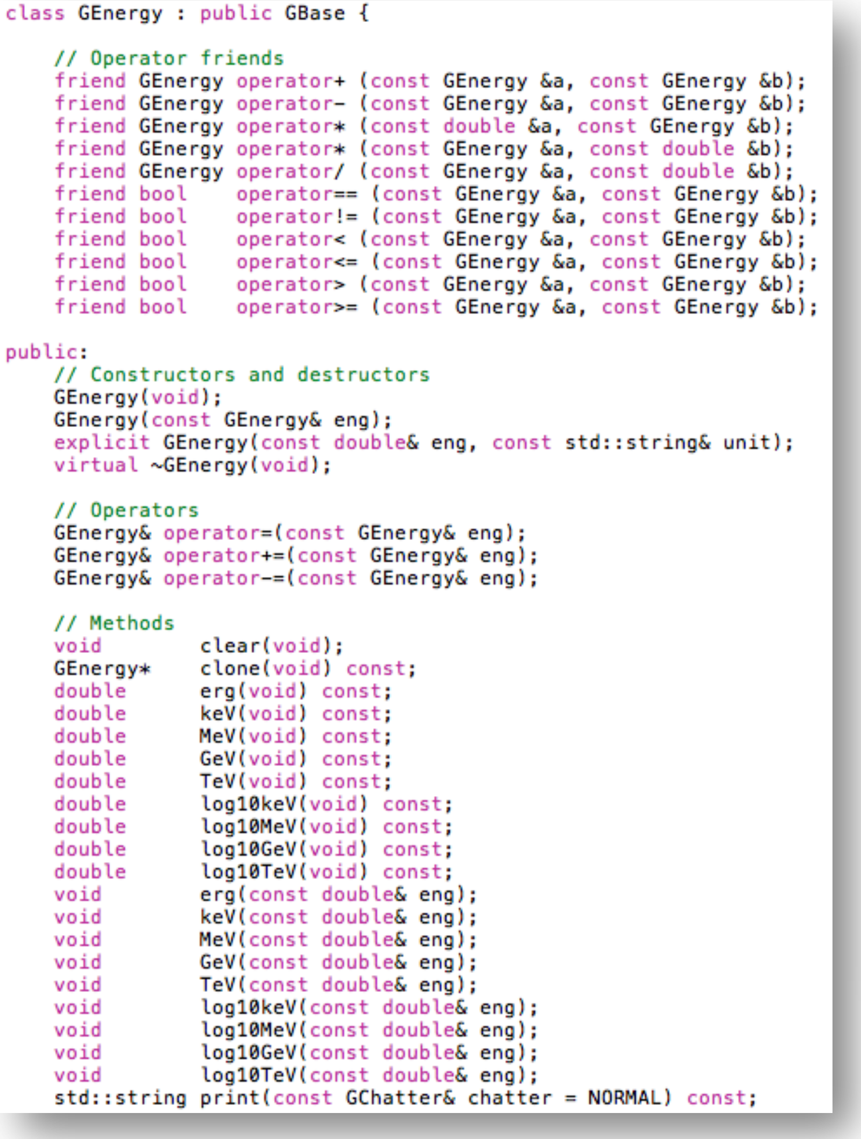
- Define and enforce coding rules (code quality)
- Avoid dependencies (full control over product)
- Support widely used platforms (Linux, Mac OS X, Solaris)
- Automatize unit testing, integration and deployment (continuous integration system & quality check)

- Open source development (end users develop the code)
- Follow an Agile development model (implement what end users need)
- Follow analysis models used in the high-energy astronomy domain (Fermi, INTEGRAL, XMM, Chandra, etc.)

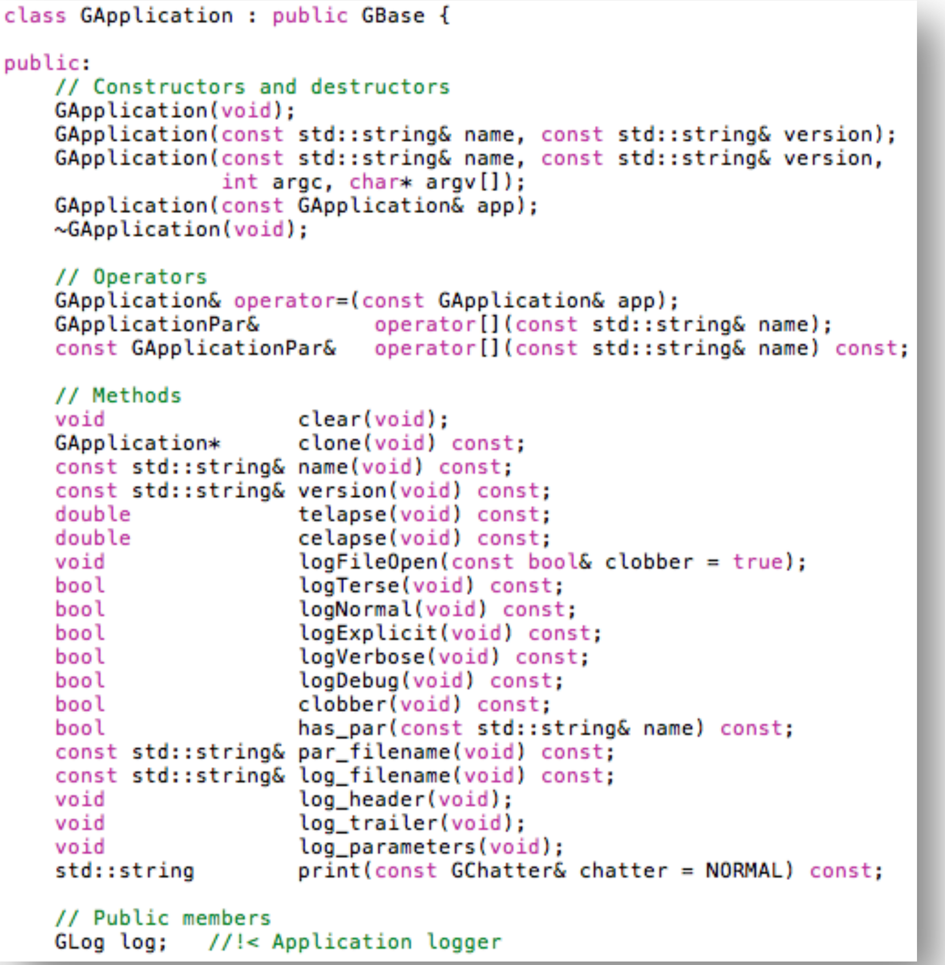
GammaLib overview



It’s all C++ classes

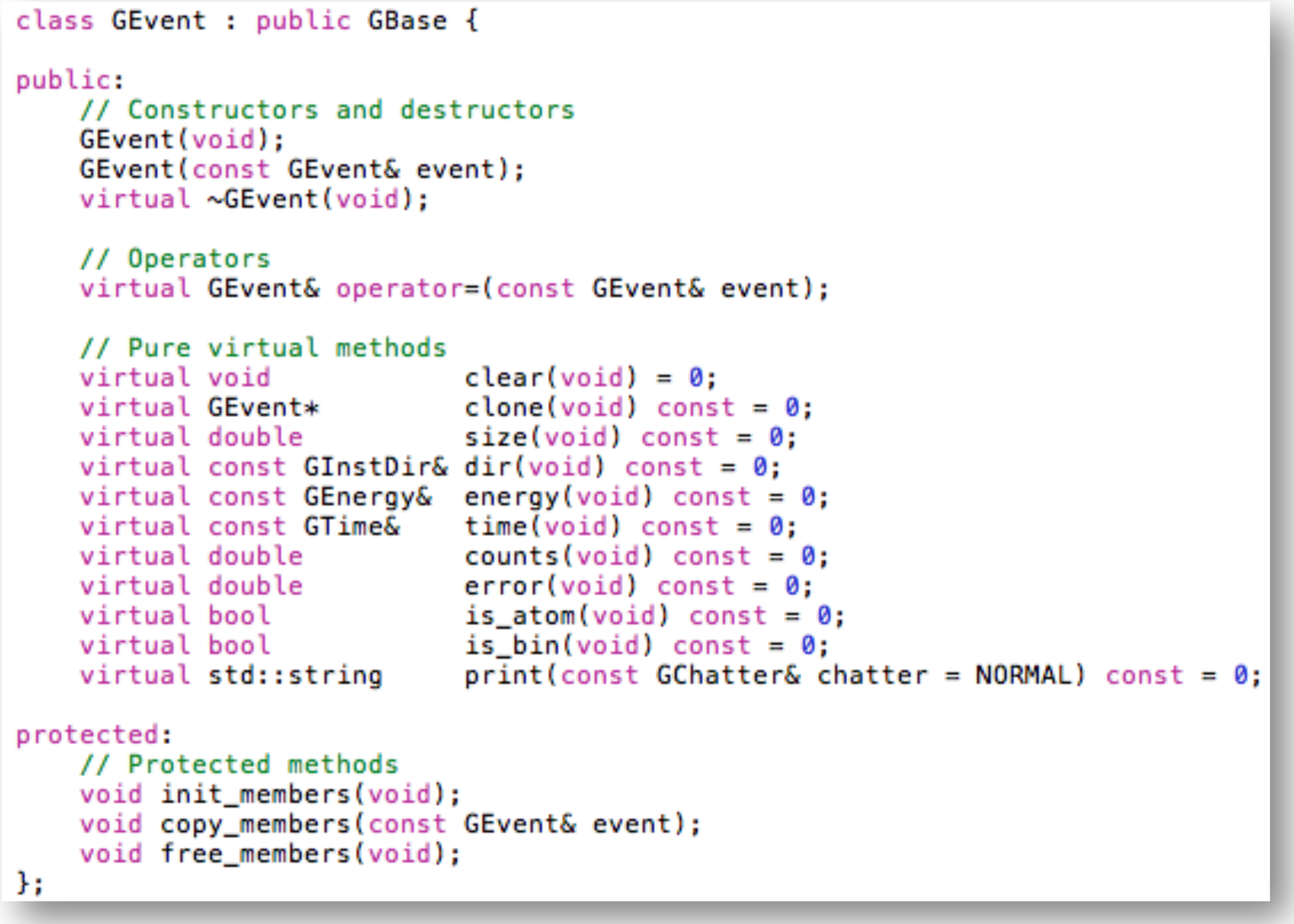


```
class GEnergy : public GBase {

    // Operator friends
    friend GEnergy operator+ (const GEnergy &a, const GEnergy &b);
    friend GEnergy operator- (const GEnergy &a, const GEnergy &b);
    friend GEnergy operator* (const double &a, const GEnergy &b);
    friend GEnergy operator* (const GEnergy &a, const double &b);
    friend GEnergy operator/ (const GEnergy &a, const double &b);
    friend bool    operator== (const GEnergy &a, const GEnergy &b);
    friend bool    operator!= (const GEnergy &a, const GEnergy &b);
    friend bool    operator< (const GEnergy &a, const GEnergy &b);
    friend bool    operator<= (const GEnergy &a, const GEnergy &b);
    friend bool    operator> (const GEnergy &a, const GEnergy &b);
    friend bool    operator>= (const GEnergy &a, const GEnergy &b);

public:
    // Constructors and destructors
    GEnergy(void);
    GEnergy(const GEnergy& eng);
    explicit GEnergy(const double& eng, const std::string& unit);
    virtual ~GEnergy(void);

    // Operators
    GEnergy& operator=(const GEnergy& eng);
    GEnergy& operator+=(const GEnergy& eng);
    GEnergy& operator-=(const GEnergy& eng);

    // Methods
    void        clear(void);
    GEnergy*    clone(void) const;
    double      erg(void) const;
    double      keV(void) const;
    double      MeV(void) const;
    double      GeV(void) const;
    double      TeV(void) const;
    double      log10keV(void) const;
    double      log10MeV(void) const;
    double      log10GeV(void) const;
    double      log10TeV(void) const;
    void        erg(const double& eng);
    void        keV(const double& eng);
    void        MeV(const double& eng);
    void        GeV(const double& eng);
    void        TeV(const double& eng);
    void        log10keV(const double& eng);
    void        log10MeV(const double& eng);
    void        log10GeV(const double& eng);
    void        log10TeV(const double& eng);
    std::string print(const GChatter& chatter = NORMAL) const;
```



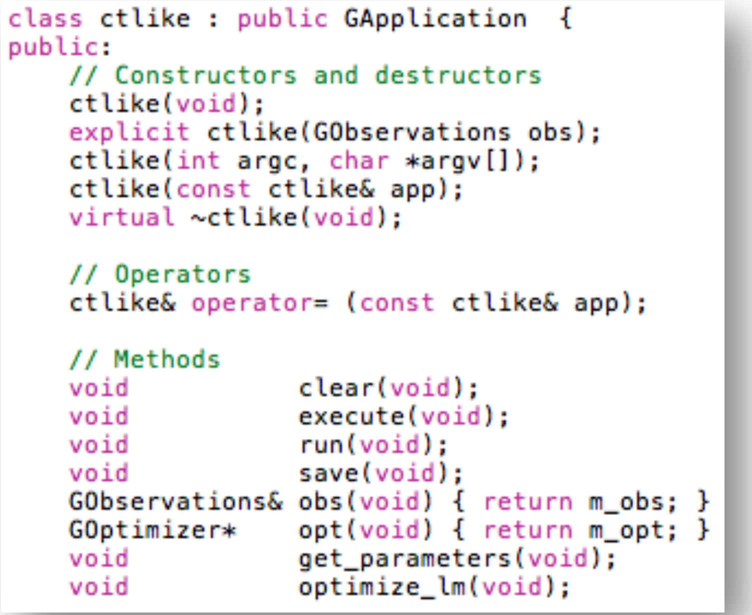
```
class GApplication : public GBase {

public:
    // Constructors and destructors
    GApplication(void);
    GApplication(const std::string& name, const std::string& version);
    GApplication(const std::string& name, const std::string& version,
                 int argc, char* argv[]);
    GApplication(const GApplication& app);
    ~GApplication(void);

    // Operators
    GApplication& operator=(const GApplication& app);
    GApplicationPar&         operator[](const std::string& name);
    const GApplicationPar&   operator[](const std::string& name) const;

    // Methods
    void               clear(void);
    GApplication*      clone(void) const;
    const std::string& name(void) const;
    const std::string& version(void) const;
    double             telapse(void) const;
    double             celapse(void) const;
    void               logFileOpen(const bool& clobber = true);
    bool               logTerse(void) const;
    bool               logNormal(void) const;
    bool               logExplicit(void) const;
    bool               logVerbose(void) const;
    bool               logDebug(void) const;
    bool               clobber(void) const;
    bool               has_par(const std::string& name) const;
    const std::string& par_filename(void) const;
    const std::string& log_filename(void) const;
    void               log_header(void);
    void               log_trailer(void);
    void               log_parameters(void);
    std::string        print(const GChatter& chatter = NORMAL) const;

    // Public members
    GLog log;   //!< Application logger
```

Abstract C++ classes for abstract interfaces



```
class GEvent : public GBase {

public:
    // Constructors and destructors
    GEvent(void);
    GEvent(const GEvent& event);
    virtual ~GEvent(void);

    // Operators
    virtual GEvent& operator=(const GEvent& event);

    // Pure virtual methods
    virtual void            clear(void) = 0;
    virtual GEvent*         clone(void) const = 0;
    virtual double          size(void) const = 0;
    virtual const GInstDir& dir(void) const = 0;
    virtual const GEnergy&  energy(void) const = 0;
    virtual const GTime&    time(void) const = 0;
    virtual double          counts(void) const = 0;
    virtual double          error(void) const = 0;
    virtual bool            is_atom(void) const = 0;
    virtual bool            is_bin(void) const = 0;
    virtual std::string     print(const GChatter& chatter = NORMAL) const = 0;

protected:
    // Protected methods
    void init_members(void);
    void copy_members(const GEvent& event);
    void free_members(void);
};
```

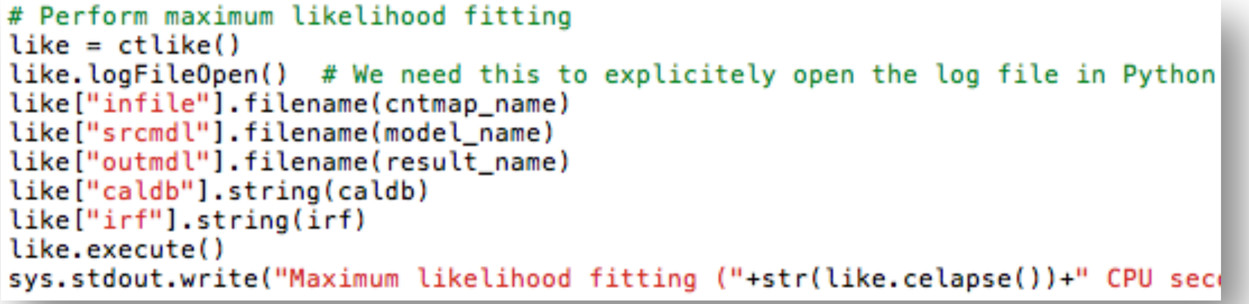
A *ctool* is an executable and a class



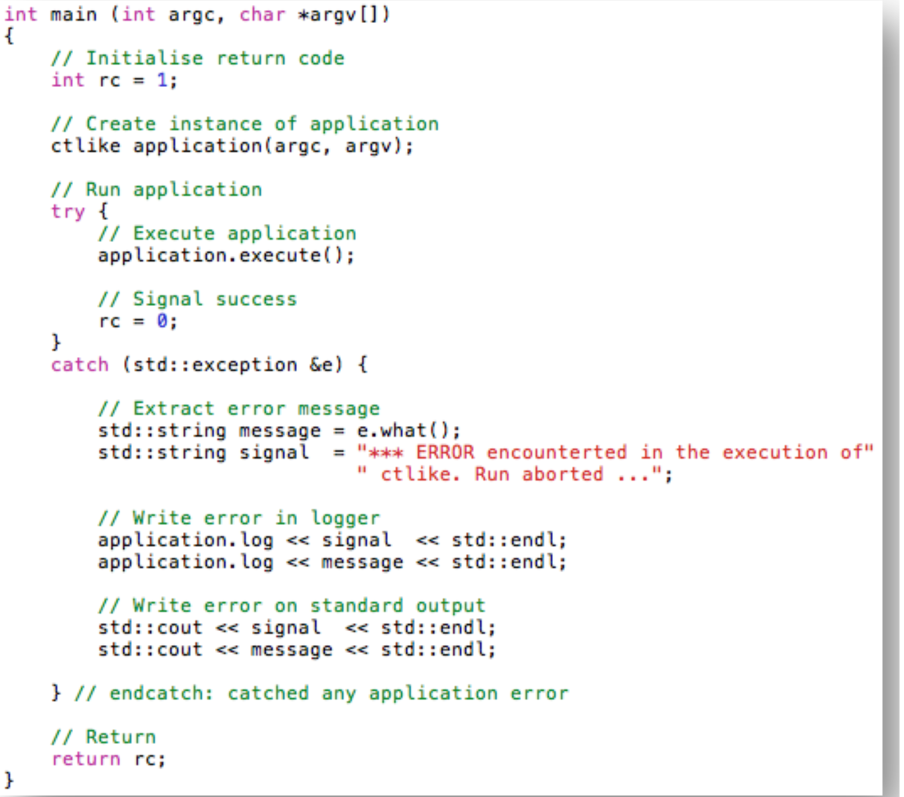
```
class ctlike : public GApplication  {
public:
    // Constructors and destructors
    ctlike(void);
    explicit ctlike(GObservations obs);
    ctlike(int argc, char *argv[]);
    ctlike(const ctlike& app);
    virtual ~ctlike(void);

    // Operators
    ctlike& operator= (const ctlike& app);

    // Methods
    void            clear(void);
    void            execute(void);
    void            run(void);
    void            save(void);
    GObservations&  obs(void) { return m_obs; }
    GOptimizer*     opt(void) { return m_opt; }
    void            get_parameters(void);
    void            optimize_lm(void);
```

ctlike is a C++ class …

… that can be used as a Python class in a script …



```
# Perform maximum likelihood fitting
like = ctlike()
like.logFileOpen()  # We need this to explicitely open the log file in Python
like["infile"].filename(cntmap_name)
like["srcmdl"].filename(model_name)
like["outmdl"].filename(result_name)
like["caldb"].string(caldb)
like["irf"].string(irf)
like.execute()
sys.stdout.write("Maximum likelihood fitting ("+str(like.celapse())+" CPU sec
```



```
int main (int argc, char *argv[])
{
    // Initialise return code
    int rc = 1;

    // Create instance of application
    ctlike application(argc, argv);

    // Run application
    try {
        // Execute application
        application.execute();

        // Signal success
        rc = 0;
    }
    catch (std::exception &e) {

        // Extract error message
        std::string message = e.what();
        std::string signal  = "*** ERROR encounterted in the execution of"
                              " ctlike. Run aborted ...";

        // Write error in logger
        application.log << signal  << std::endl;
        application.log << message << std::endl;

        // Write error on standard output
        std::cout << signal  << std::endl;
        std::cout << message << std::endl;

    } // endcatch: catched any application error

    // Return
    return rc;
}
```

… or as a C++ class in a C++ program (used to build the ctlike executable)

Running a *ctool* executable

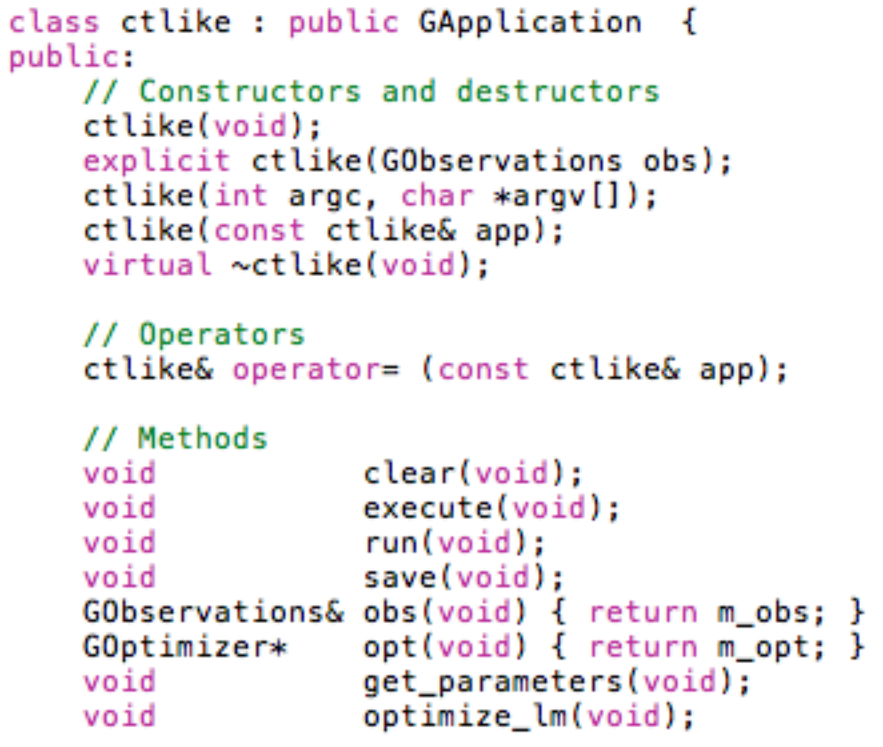
CTA event list simulator

```
$ ctobssim
Model [$CTOOLS/share/models/crab.xml]
Calibration database [aar]
Instrument response function [DESY20140105_50h]
RA of pointing (degrees) (0-360) [83.63]
Dec of pointing (degrees) (-90-90) [22.01]
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]
```

Wrapping C++ in Python: SWIG

http://www.swig.org/

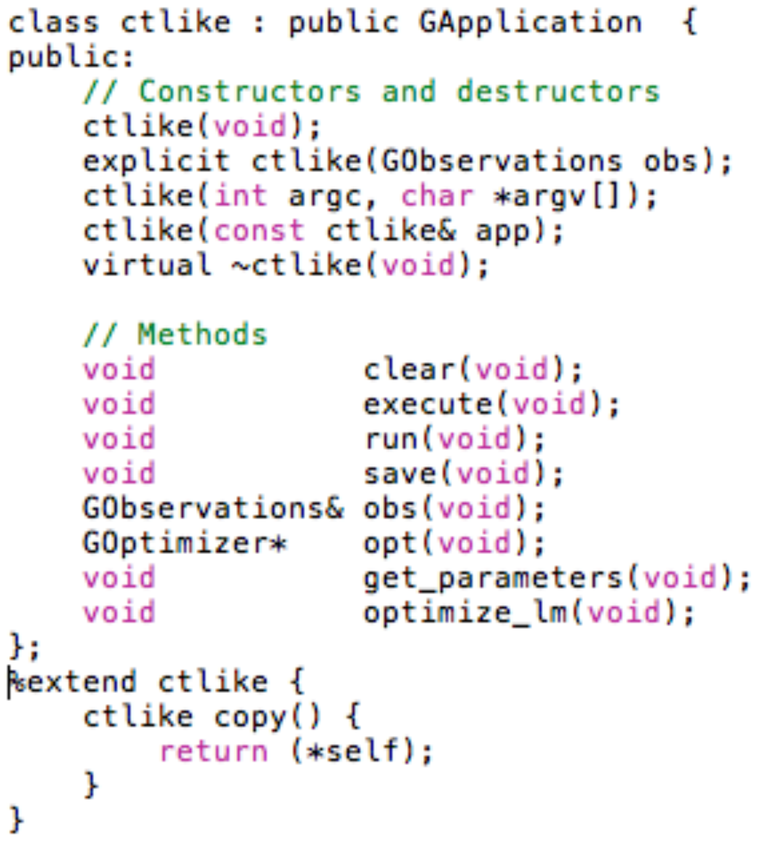
ctlike.hpp



```
class ctlike : public GApplication  {
public:
    // Constructors and destructors
    ctlike(void);
    explicit ctlike(GObservations obs);
    ctlike(int argc, char *argv[]);
    ctlike(const ctlike& app);
    virtual ~ctlike(void);

    // Operators
    ctlike& operator= (const ctlike& app);

    // Methods
    void            clear(void);
    void            execute(void);
    void            run(void);
    void            save(void);
    GObservations&  obs(void) { return m_obs; }
    GOptimizer*     opt(void) { return m_opt; }
    void            get_parameters(void);
    void            optimize_lm(void);
```

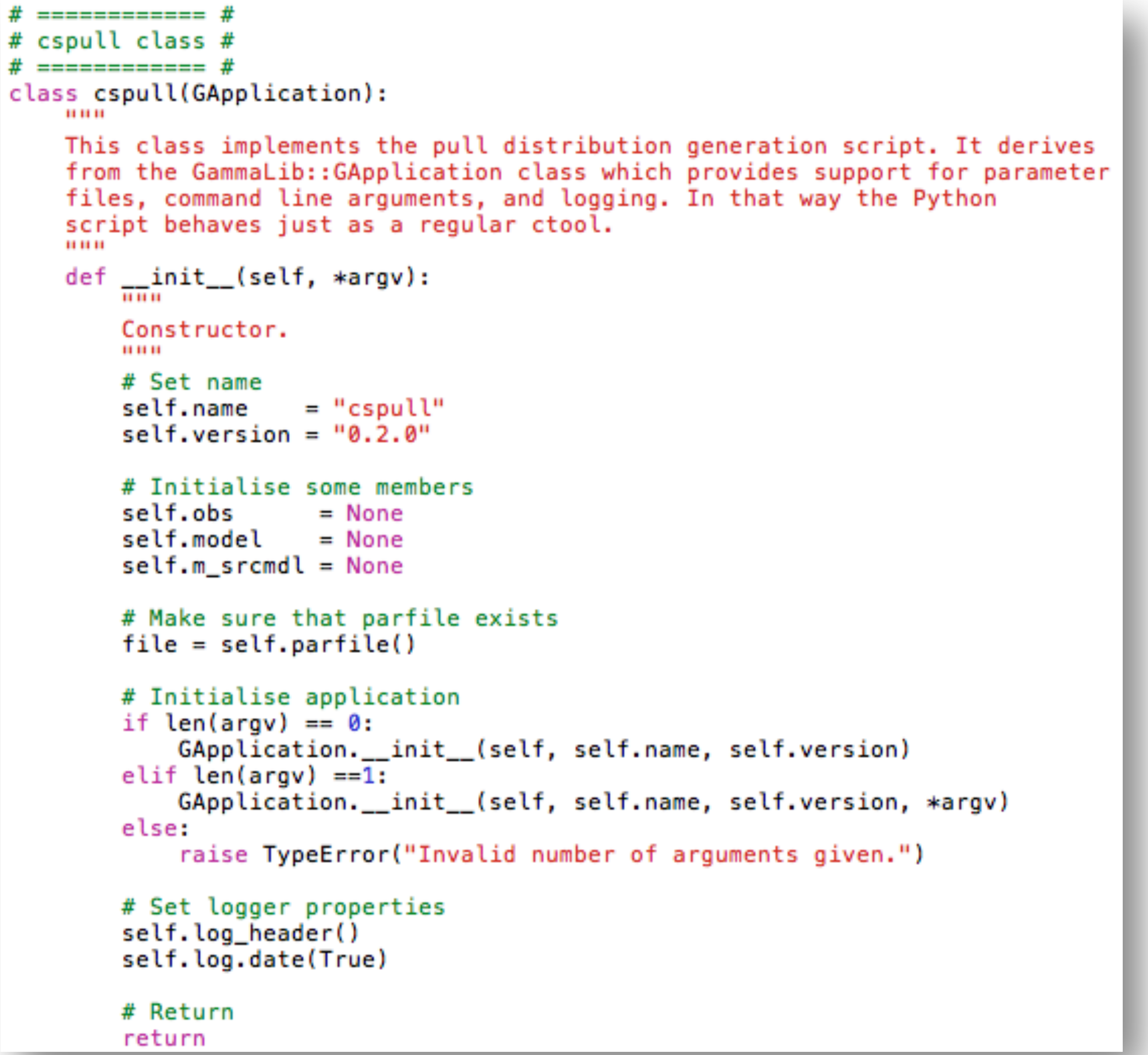
```
$ swig -c++ -python -Wall ctlike.i
ctlike.py
ctlike_wrap.cpp
$ gcc ctlike_wrap.cpp
```

ctlike.i



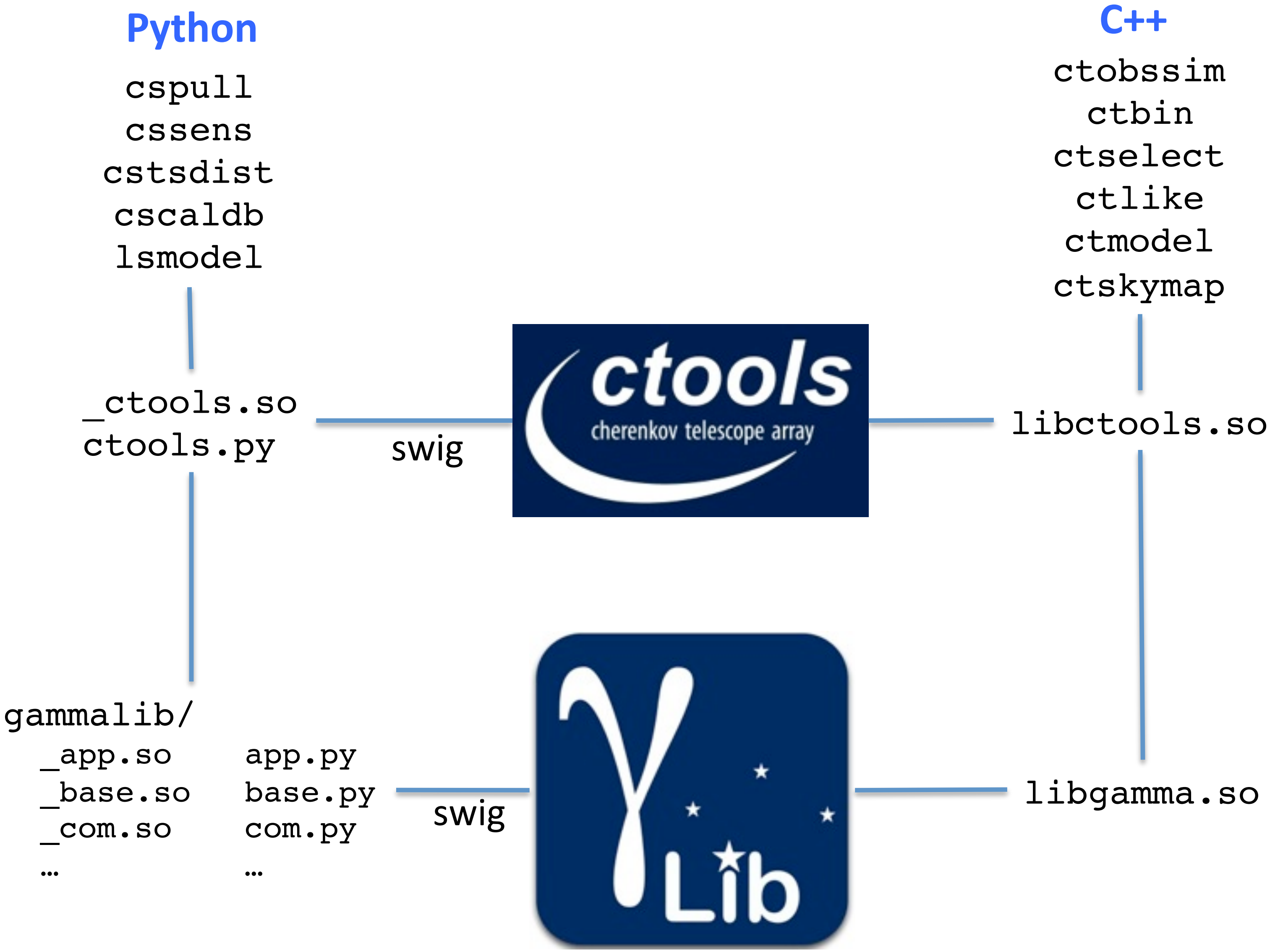
```
class ctlike : public GApplication  {
public:
    // Constructors and destructors
    ctlike(void);
    explicit ctlike(GObservations obs);
    ctlike(int argc, char *argv[]);
    ctlike(const ctlike& app);
    virtual ~ctlike(void);

    // Methods
    void            clear(void);
    void            execute(void);
    void            run(void);
    void            save(void);
    GObservations&  obs(void);
    GOptimizer*     opt(void);
    void            get_parameters(void);
    void            optimize_lm(void);
};
%extend ctlike {
    ctlike copy() {
        return (*self);
    }
}
```

A *cscript* is a Python script looking like a *ctool*



```
# ============ #
# cspull class #
# ============ #
class cspull(GApplication):
    """
    This class implements the pull distribution generation script. It derives
    from the GammaLib::GApplication class which provides support for parameter
    files, command line arguments, and logging. In that way the Python
    script behaves just as a regular ctool.
    """
    def __init__(self, *argv):
        """
        Constructor.
        """
        # Set name
        self.name    = "cspull"
        self.version = "0.2.0"

        # Initialise some members
        self.obs      = None
        self.model    = None
        self.m_srcmdl = None

        # Make sure that parfile exists
        file = self.parfile()

        # Initialise application
        if len(argv) == 0:
            GApplication.__init__(self, self.name, self.version)
        elif len(argv) ==1:
            GApplication.__init__(self, self.name, self.version, *argv)
        else:
            raise TypeError("Invalid number of arguments given.")

        # Set logger properties
        self.log_header()
        self.log.date(True)

        # Return
        return
```

The overall picture



Python

cspull
cssens
cstsdist
cscaldb
lsmodel

_ctools.so
ctools.py

swig

C++

ctobssim
ctbin
ctselect
ctlike
ctmodel
ctskymap

libctools.so

gammalib/

_app.so
_base.so
_com.so
…

app.py
base.py
com.py
…

swig

libgamma.so

What should I do if …

… I need a new spectral model?
Add a new spectral model class to the GammaLib model module.

… I need a new background model for CTA?
Add a new background model class to the GammaLib CTA interface module.

… I want a tool that generates CTA exposure maps?
Create a new ctool that uses the CTA response functions in GammaLib for exposure map computation.

… I want to implement an analysis workflow or pipeline?
Create a Python script that uses the ctools and gammalib Python modules.

… I want to test a new idea (e.g. create a ring background generator)?
Create a new cscript that uses the gammalib Python module.

General rule:
All generic and reusable code goes in GammaLib, code that is CTA specific and that is only needed for one specific task goes in ctools. Quick coding is better done by a cscript.

2. New features since last coding sprint

New tools & scripts
Stacked analysis

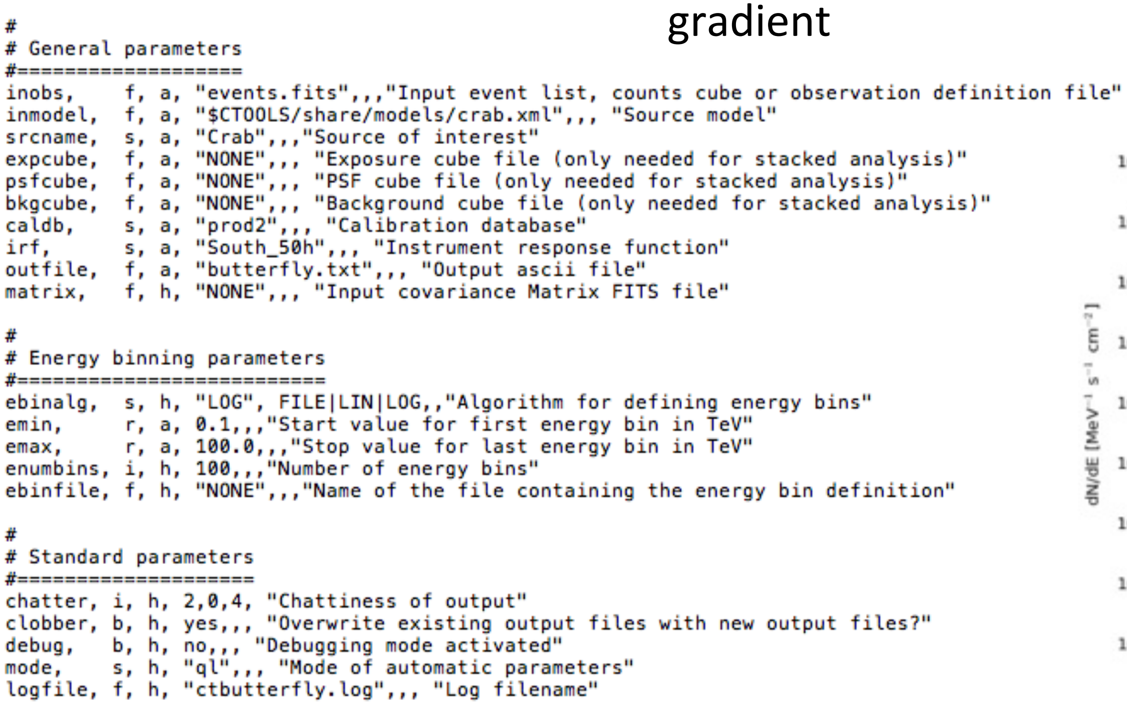
New tools

ctbutterfly cttsmap ctulimit cterror	likelihood related
ctcubemask	binned or stacked analysis
ctbkgcube ctpsfcube ctexpcube	stacked analysis

ctbutterfly

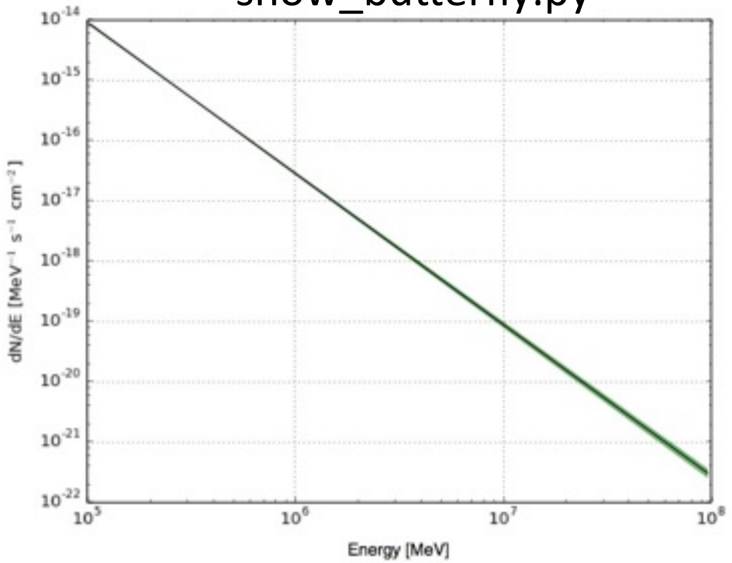
Purpose: produce a butterfly diagram for a spectral model M

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



```
#
# 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"
```

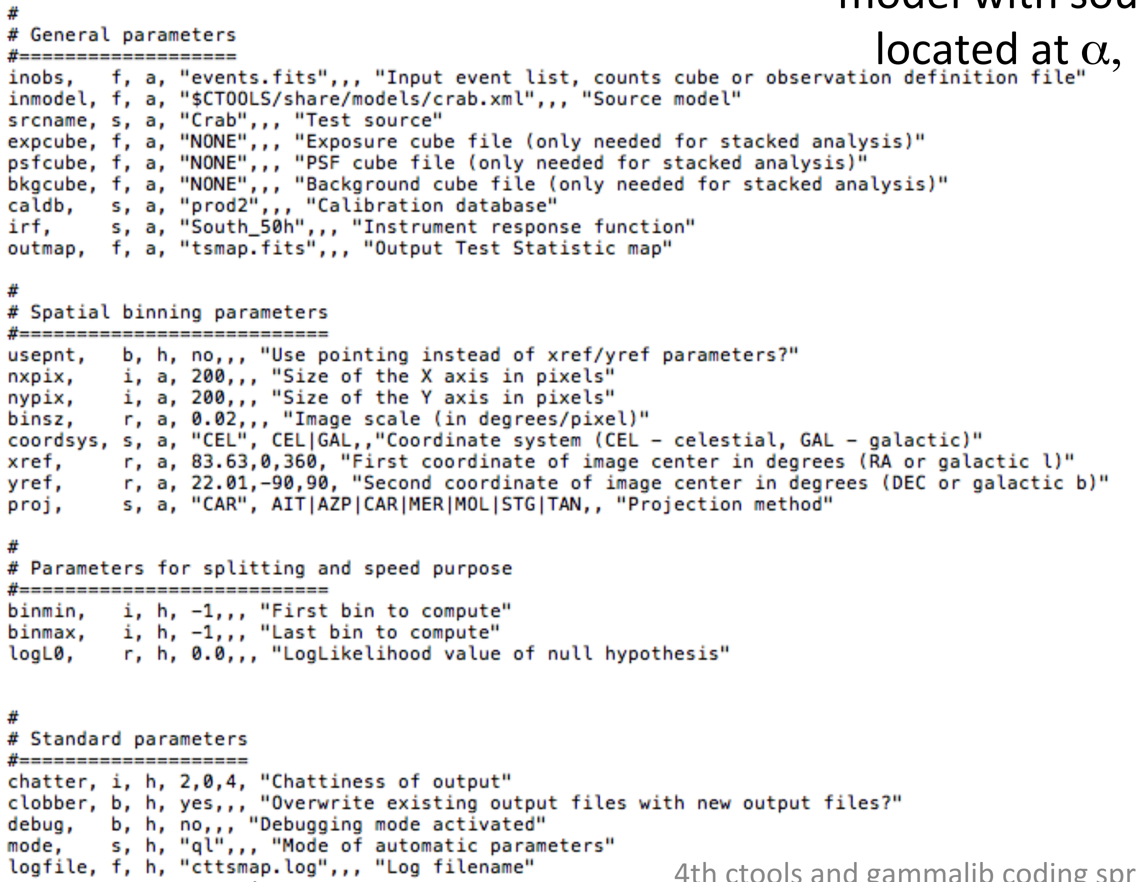
show_butterfly.py



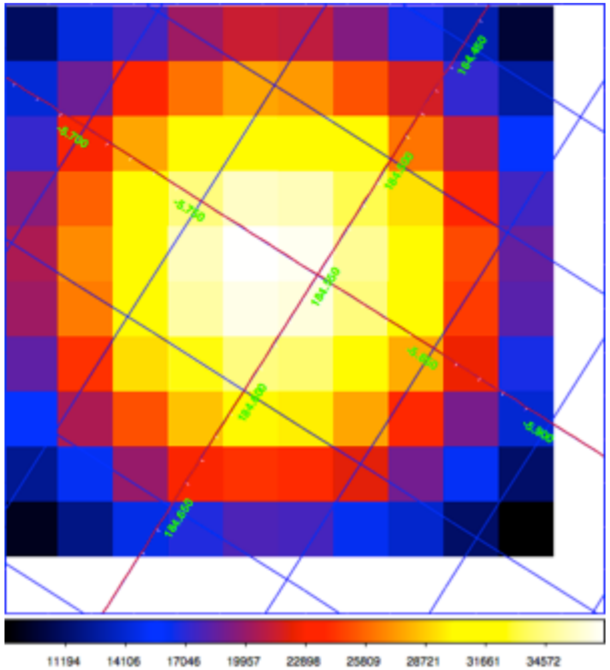
cttsmap

Purpose: produce a Test Statistics map

$$TS(\alpha,\delta) = 2\,(\underbrace{\ln\mathcal{L}(\alpha,\delta)}_{\text{model with source located at }\alpha,\,\delta} - \underbrace{\ln\mathcal{L}_0}_{\text{model without source}})$$

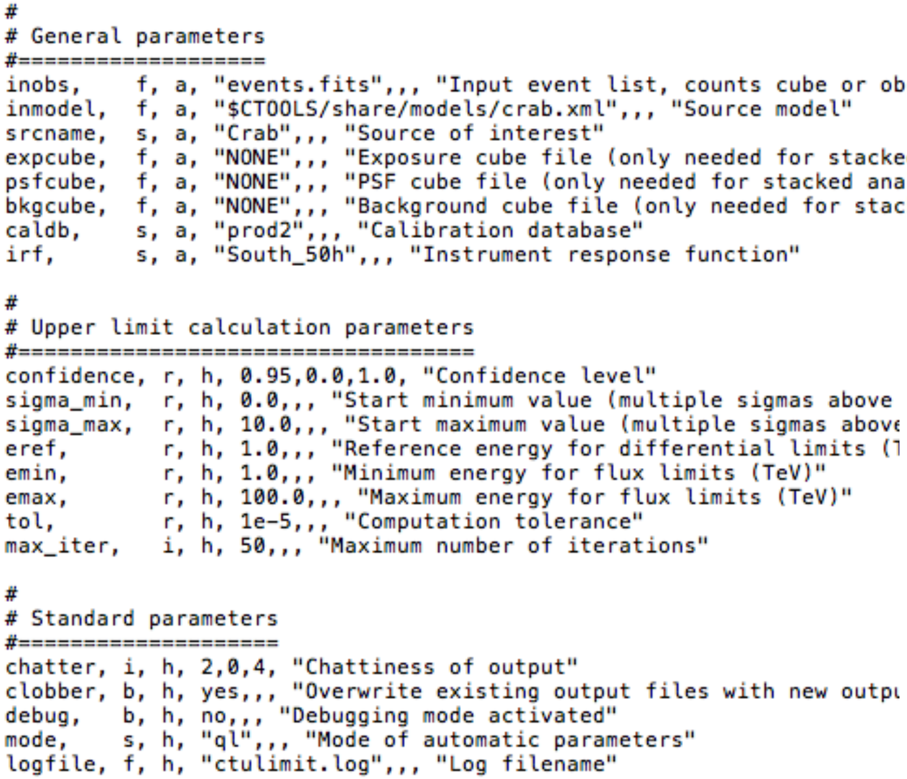


```
#
# General parameters
#===================
inobs,   f, a, "events.fits",,, "Input event list, counts cube or observation definition file"
inmodel, f, a, "$CTOOLS/share/models/crab.xml",,, "Source model"
srcname, s, a, "Crab",,, "Test source"
expcube, f, a, "NONE",,, "Exposure cube file (only needed 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"
```

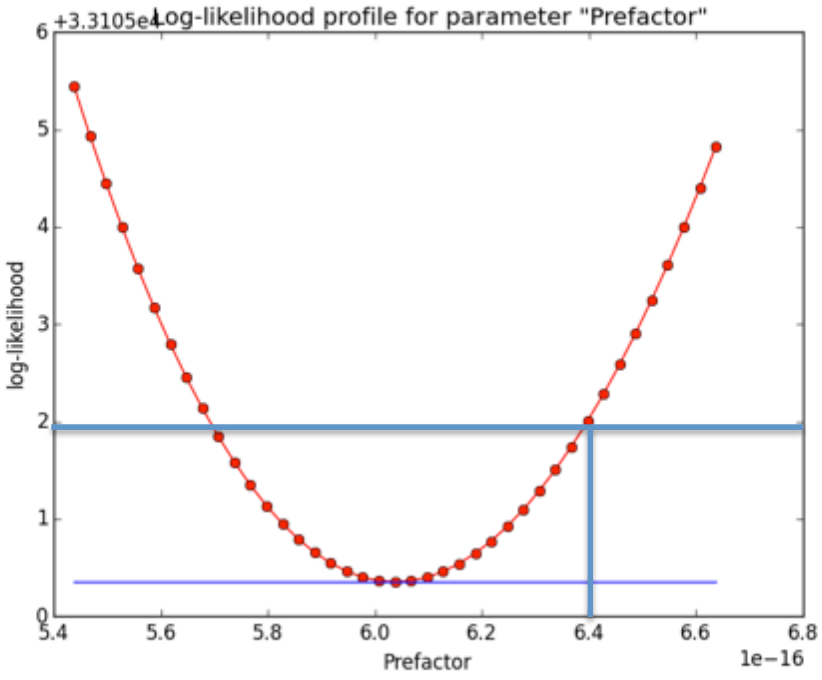


ctulimit

Purpose: produce upper flux limits for a source

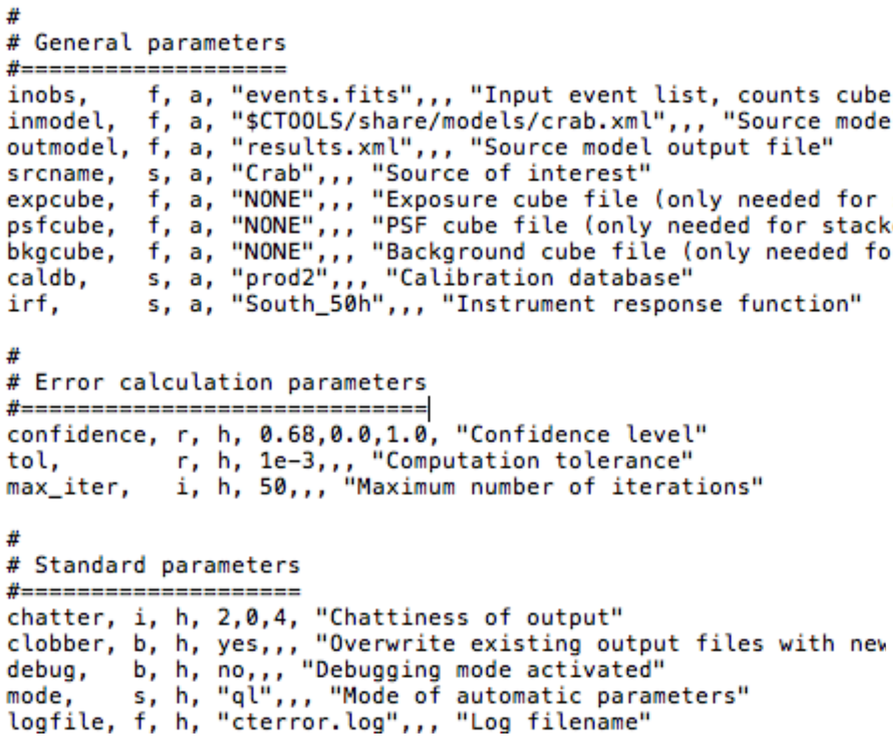


```
#
# General parameters
#====================
inobs,    f, a, "events.fits",,, "Input event list, counts cube or observation definition file"
inmodel,  f, a, "$CTOOLS/share/models/crab.xml",,, "Source model"
srcname,  s, a, "Crab",,, "Source of interest"
expcube,  f, a, "NONE",,, "Exposure cube file (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"

#
# Upper limit calculation parameters
#====================================
confidence, r, h, 0.95,0.0,1.0, "Confidence level"
sigma_min,  r, h, 0.0,,, "Start minimum value (multiple sigmas above
sigma_max,  r, h, 10.0,,, "Start maximum value (multiple sigmas above
eref,       r, h, 1.0,,, "Reference energy for differential limits (T
emin,       r, h, 1.0,,, "Minimum energy for flux limits (TeV)"
emax,       r, h, 100.0,,, "Maximum energy for flux limits (TeV)"
tol,        r, h, 1e-5,,, "Computation tolerance"
max_iter,   i, h, 50,,, "Maximum number of iterations"

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

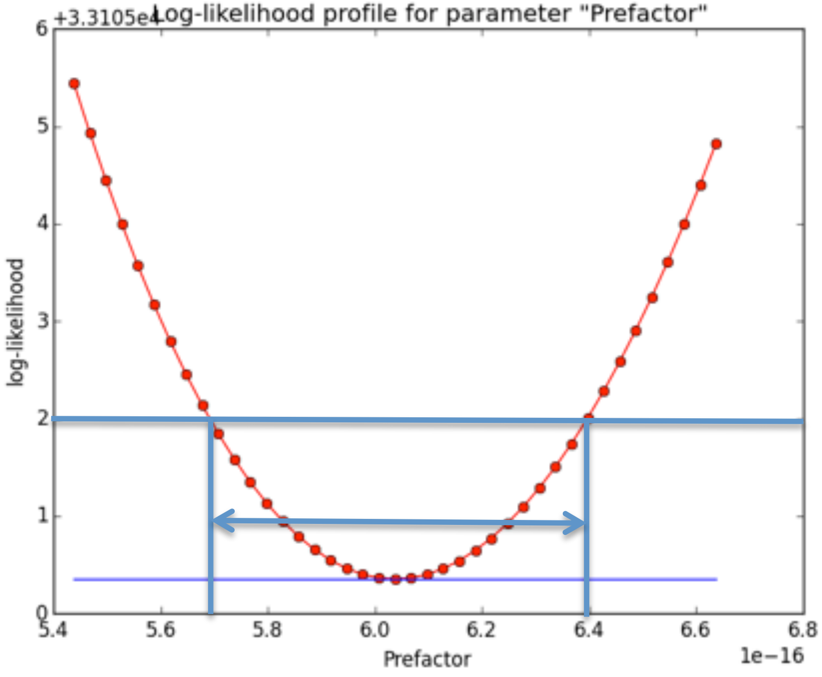


cterror

Purpose: evaluate errors using the likelihood profile

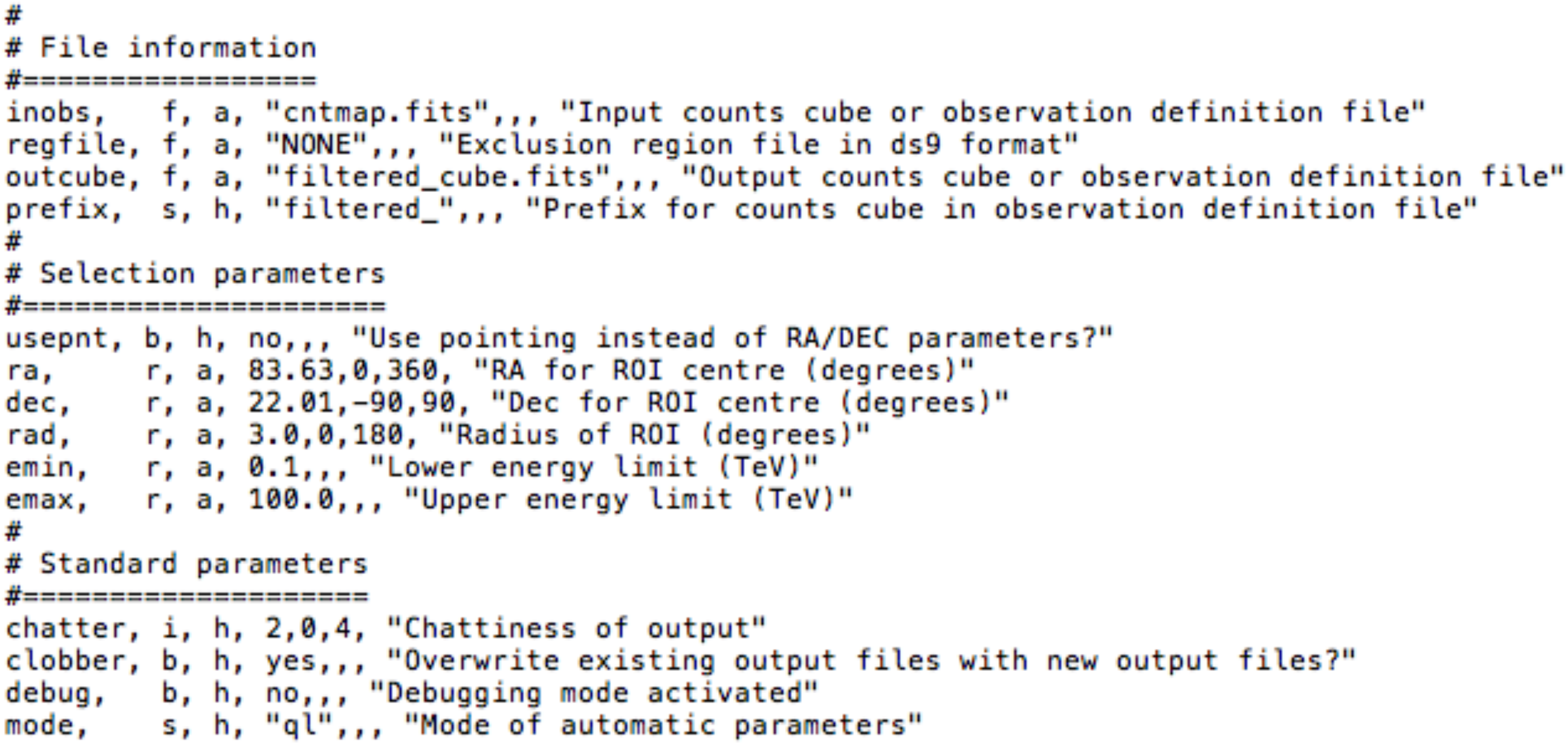


```
#
# General parameters
#====================
inobs,    f, a, "events.fits",,, "Input event list, counts cube or observation definition file"
inmodel,  f, a, "$CTOOLS/share/models/crab.xml",,, "Source model"
outmodel, f, a, "results.xml",,, "Source model output file"
srcname,  s, a, "Crab",,, "Source 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"

#
# Error calculation parameters
#============================
confidence, r, h, 0.68,0.0,1.0, "Confidence level"
tol,        r, h, 1e-3,,, "Computation tolerance"
max_iter,   i, h, 50,,, "Maximum number of iterations"

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



ctcubemask

Purpose: mask out bins in an events cube (by setting them to negative value)



```
#
# File information
#=================
inobs,   f, a, "cntmap.fits",,, "Input counts cube or observation definition file"
regfile, f, a, "NONE",,, "Exclusion region file in ds9 format"
outcube, f, a, "filtered_cube.fits",,, "Output counts cube or observation definition file"
prefix,  s, h, "filtered_",,, "Prefix for counts cube in observation definition file"
#
# Selection parameters
#=====================
usepnt, b, h, no,,, "Use pointing instead of RA/DEC parameters?"
ra,     r, a, 83.63,0,360, "RA for ROI centre (degrees)"
dec,    r, a, 22.01,-90,90, "Dec for ROI centre (degrees)"
rad,    r, a, 3.0,0,180, "Radius of ROI (degrees)"
emin,   r, a, 0.1,,, "Lower energy limit (TeV)"
emax,   r, a, 100.0,,, "Upper energy limit (TeV)"
#
# 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"
```

Stacked analysis

Principle

Combine data for multiple observations (aka runs) into a single events cube.

Requires:

- computation of an effective exposure (exposure cube)
- computation of a (exposure) averaged point spread function (Psf cube)
- computation of a (exposure) averaged background rate (Background cube)

As opposed to …

Unbinned analysis (events are not binned, observations are not combined, one IRF per observation)

Binned analysis (events are binned, observations are not combined, one IRF per observation)

Stacked analysis

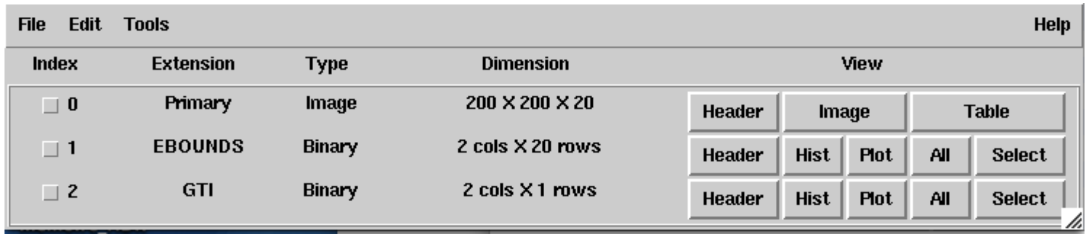
Implementation

ctbin* does now **always** combines multiple observations into a single event cube (before: ctbin generated one events cube per observation)
To get former behaviour requires an external loop over observations

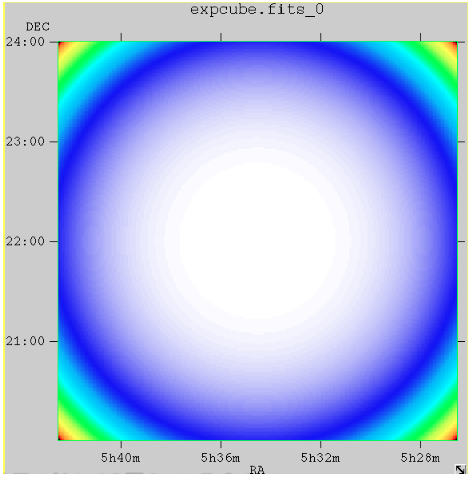
ctexpcube* computes exposure

GCTACubeExposure

$$\mathrm{Exposure}(\alpha,\delta,E) = \sum_i \underbrace{A_i(\alpha,\delta,E)}_{\text{effective area}} \times \underbrace{\tau_i}_{\text{livetime}}$$



Index	Extension	Type	Dimension	View
□ 0	Primary	Image	200 X 200 X 20	Header · Image · Table
□ 1	EBOUNDS	Binary	2 cols X 20 rows	Header · Hist · Plot · All · Select
□ 2	GTI	Binary	2 cols X 1 rows	Header · Hist · Plot · All · Select



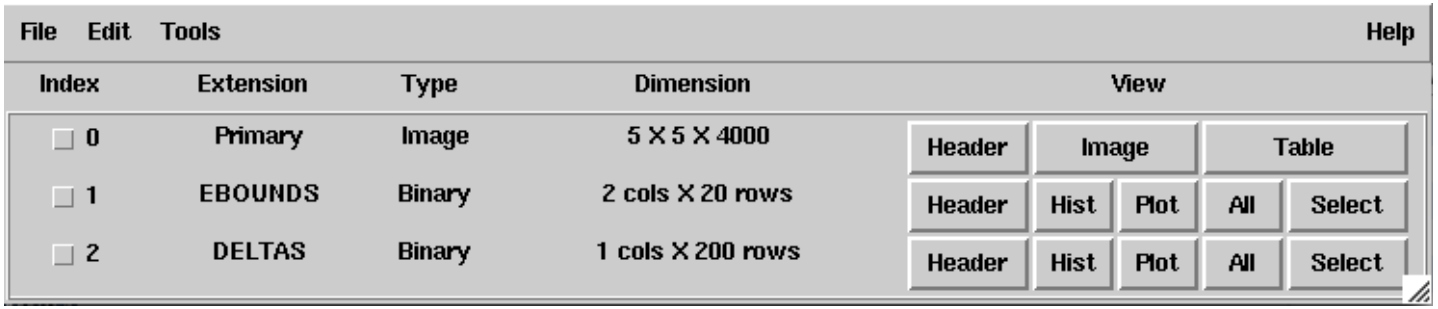
*now only take into account bins within ROI

Stacked analysis

Implementation

ctpsfcube* computes average point spread function GCTACubePsf

$$\mathrm{PSF}(\alpha,\delta,E,\theta) = \frac{\sum_i \overbrace{\mathrm{PSF}_i(\alpha,\delta,E,\theta)}^{\text{point spread function}} \times A_i(\alpha,\delta,E) \times \tau_i}{\sum_i \underbrace{A_i(\alpha,\delta,E)}_{\text{effective area}} \times \underbrace{\tau_i}_{\text{livetime}}}$$



Index	Extension	Type	Dimension
□ 0	Primary	Image	5 X 5 X 4000
□ 1	EBOUNDS	Binary	2 cols X 20 rows
□ 2	DELTAS	Binary	1 cols X 200 rows

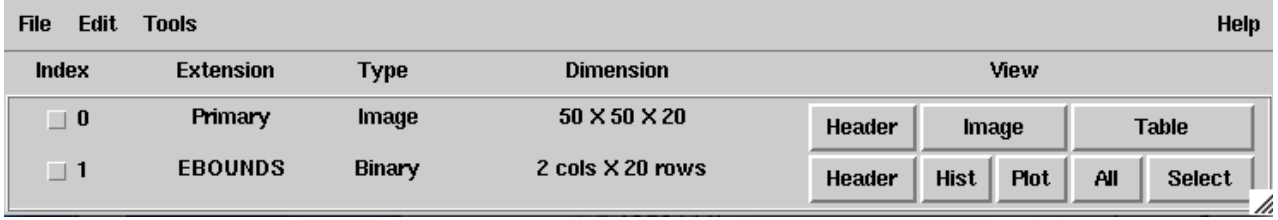
*now only take into account bins within ROI

Stacked analysis

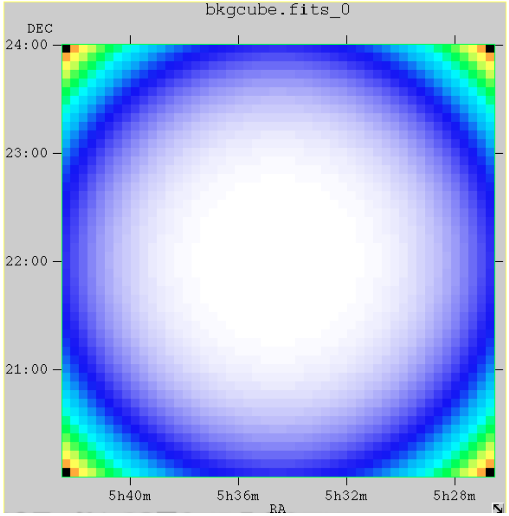
Implementation

ctbkgcube* computes average background rate GCTACubeBackground

$$\mathrm{BKG}(\alpha,\delta,E) = \frac{\sum_i \overbrace{\mathrm{BKG}_i(\alpha,\delta,E)}^{\text{background rate}} \times \overbrace{\tau_i}^{\text{livetime}}}{\sum_i \tau_i}$$



Index	Extension	Type	Dimension
□ 0	Primary	Image	50 X 50 X 20
□ 1	EBOUNDS	Binary	2 cols X 20 rows

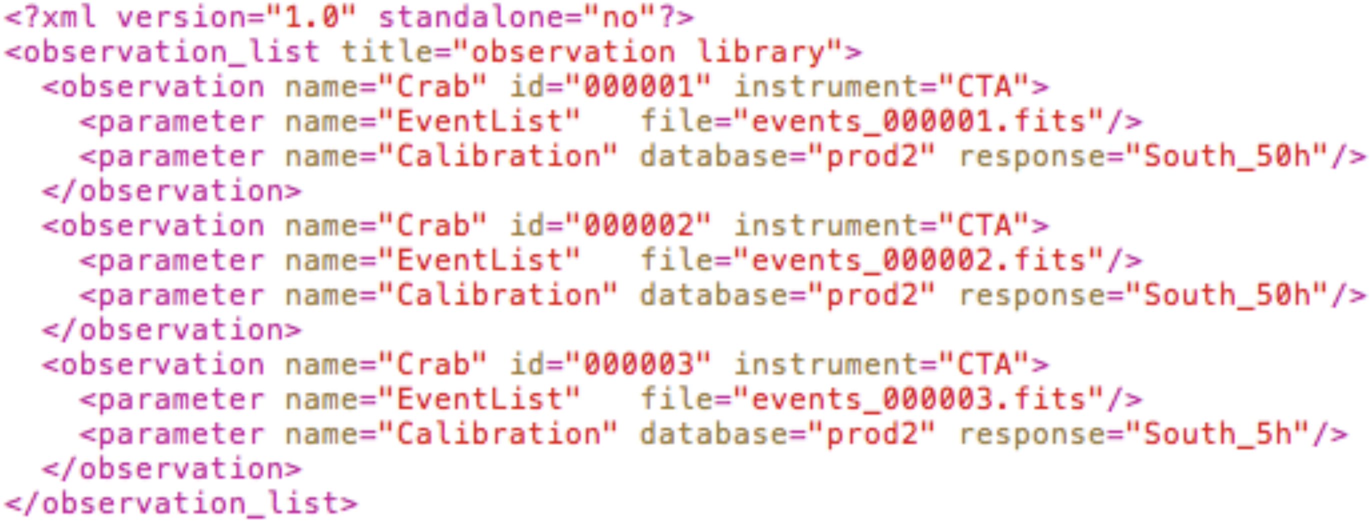


*now only take into account bins within ROI

Stacked analysis

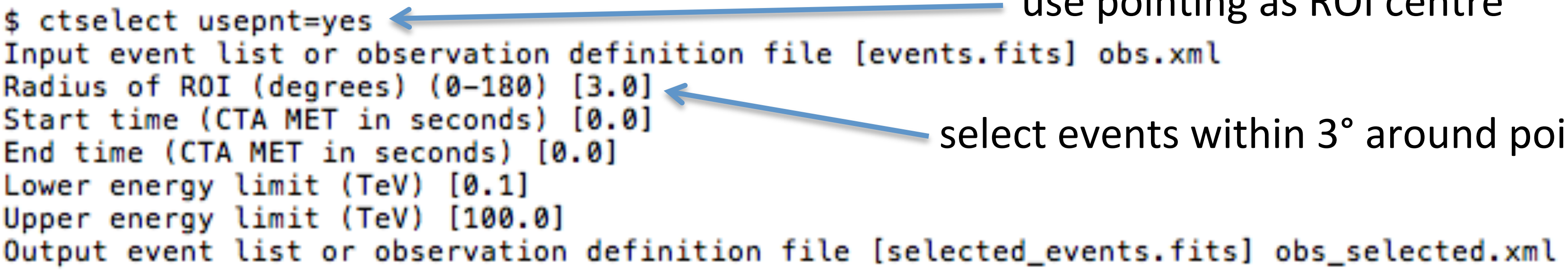
Analysis steps

1. Define your list of observations in an XML file



```
<?xml version="1.0" standalone="no"?>
<observation_list title="observation library">
  <observation name="Crab" id="000001" instrument="CTA">
    <parameter name="EventList"   file="events_000001.fits"/>
    <parameter name="Calibration" database="prod2" response="South_50h"/>
  </observation>
  <observation name="Crab" id="000002" instrument="CTA">
    <parameter name="EventList"   file="events_000002.fits"/>
    <parameter name="Calibration" database="prod2" response="South_50h"/>
  </observation>
  <observation name="Crab" id="000003" instrument="CTA">
    <parameter name="EventList"   file="events_000003.fits"/>
    <parameter name="Calibration" database="prod2" response="South_5h"/>
  </observation>
</observation_list>
```

2. Run ctselect on this list to define the ROIs



```
$ ctselect usepnt=yes
Input event list or observation definition file [events.fits] obs.xml
Radius of ROI (degrees) (0-180) [3.0]
Start time (CTA MET in seconds) [0.0]
End time (CTA MET in seconds) [0.0]
Lower energy limit (TeV) [0.1]
Upper energy limit (TeV) [100.0]
Output event list or observation definition file [selected_events.fits] obs_selected.xml
```

use pointing as ROI centre

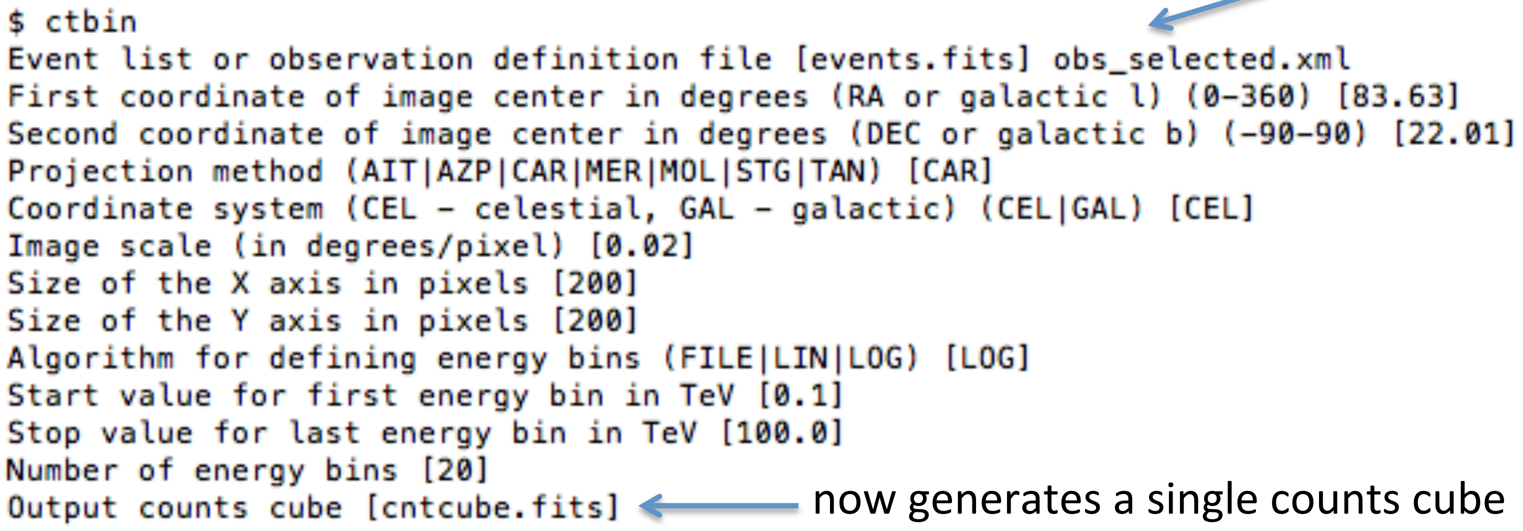
select events within 3° around pointing

Stacked analysis

Analysis steps

3. Run ctbin

use selected events

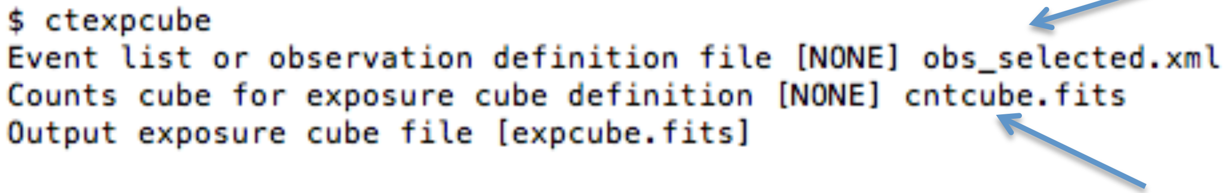


```
$ ctbin
Event list or observation definition file [events.fits] obs_selected.xml
First coordinate of image center in degrees (RA or galactic l) (0-360) [83.63]
Second coordinate of image center in degrees (DEC or galactic b) (-90-90) [22.01]
Projection method (AIT|AZP|CAR|MER|MOL|STG|TAN) [CAR]
Coordinate 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 [cntcube.fits]
```

now generates a single counts cube

4. Run ctexpcube

needs definition of observations



```
$ ctexpcube
Event list or observation definition file [NONE] obs_selected.xml
Counts cube for exposure cube definition [NONE] cntcube.fits
Output exposure cube file [expcube.fits]
```

take exposure cube definition from counts cube

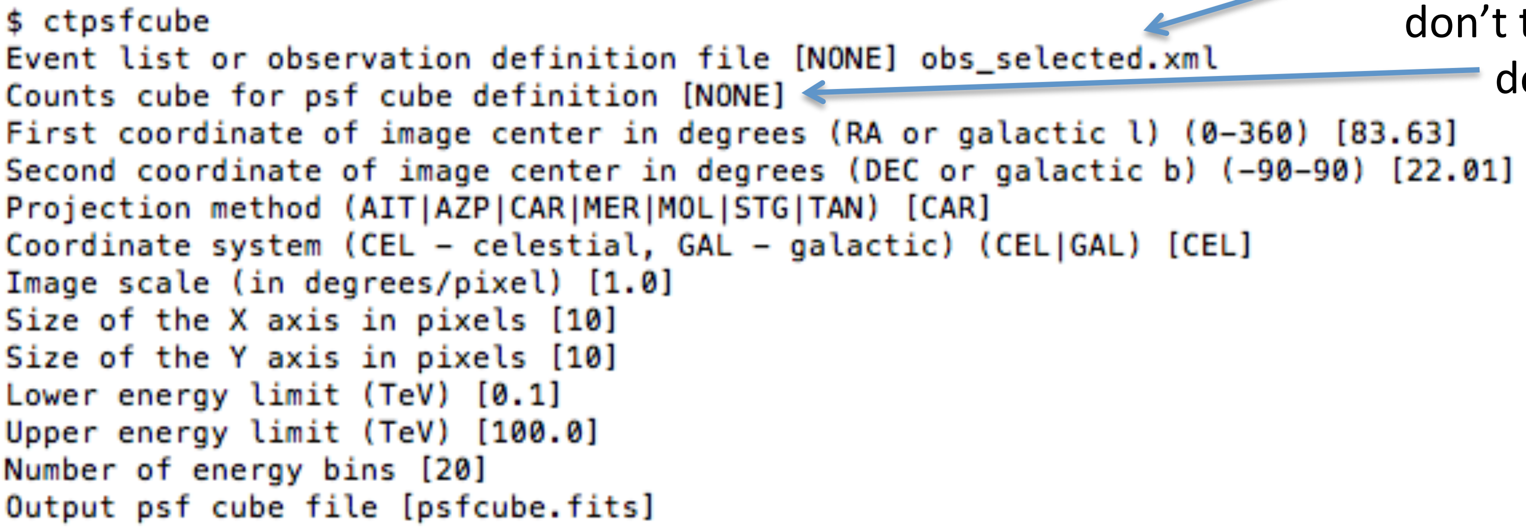
Stacked analysis

Analysis steps

5. Run ctpsfcube

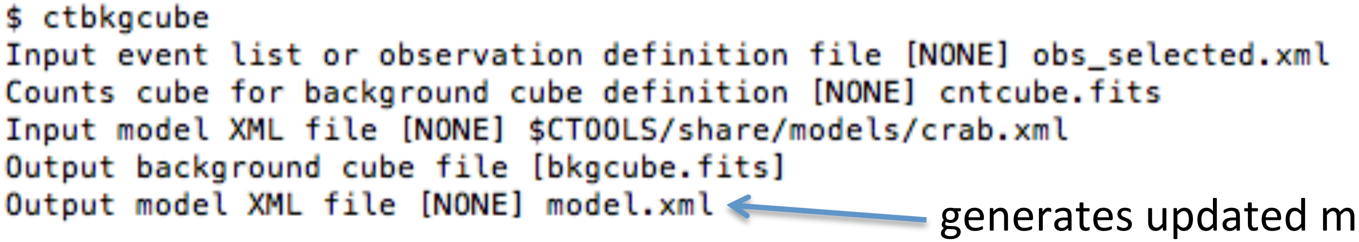
needs definition of observations

don't take PSF cube definition from counts cube



```
$ ctpsfcube
Event list or observation definition file [NONE] obs_selected.xml
Counts cube for psf cube definition [NONE]
First coordinate of image center in degrees (RA or galactic l) (0-360) [83.63]
Second coordinate of image center in degrees (DEC or galactic b) (-90-90) [22.01]
Projection method (AIT|AZP|CAR|MER|MOL|STG|TAN) [CAR]
Coordinate system (CEL - celestial, GAL - galactic) (CEL|GAL) [CEL]
Image scale (in degrees/pixel) [1.0]
Size of the X axis in pixels [10]
Size of the Y axis in pixels [10]
Lower energy limit (TeV) [0.1]
Upper energy limit (TeV) [100.0]
Number of energy bins [20]
Output psf cube file [psfcube.fits]
```

6. Run ctkbgcube



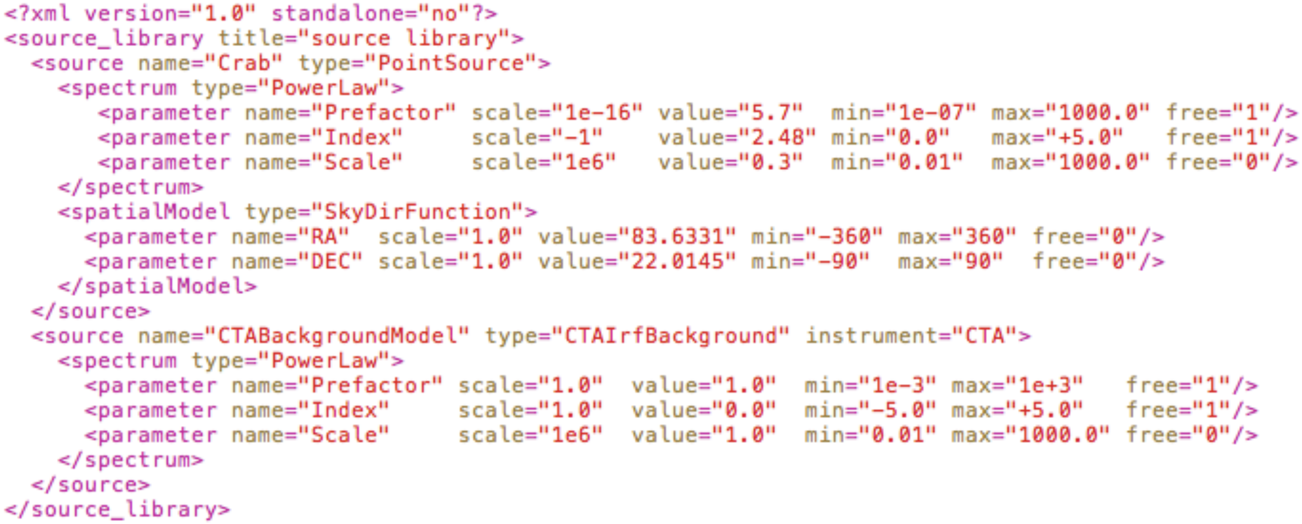
```
$ ctbkgcube
Input event list or observation definition file [NONE] obs_selected.xml
Counts cube for background cube definition [NONE] cntcube.fits
Input model XML file [NONE] $CTOOLS/share/models/crab.xml
Output background cube file [bkgcube.fits]
Output model XML file [NONE] model.xml
```

generates updated model

Stacked analysis

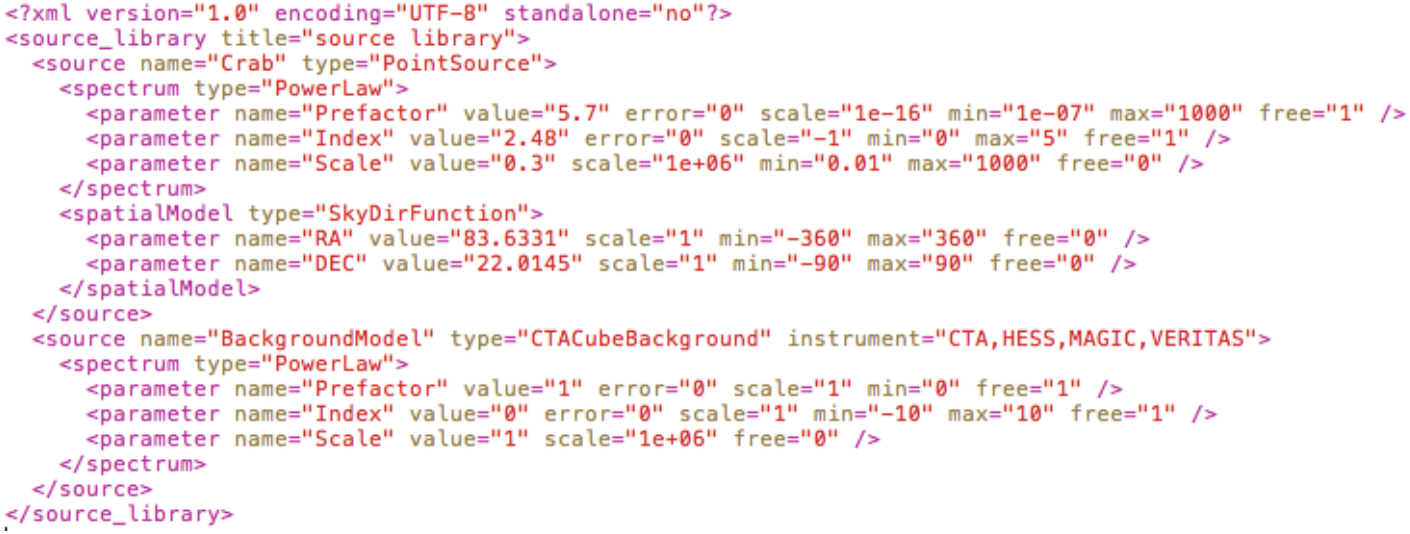
Analysis steps

input model



```
<?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="CTABackgroundModel" type="CTAIrfBackground" instrument="CTA">
    <spectrum type="PowerLaw">
      <parameter name="Prefactor" scale="1.0" value="1.0"  min="1e-3" max="1e+3"   free="1"/>
      <parameter name="Index"     scale="1.0" value="0.0"  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>
  </source>
</source_library>
```

after ctbkgcube

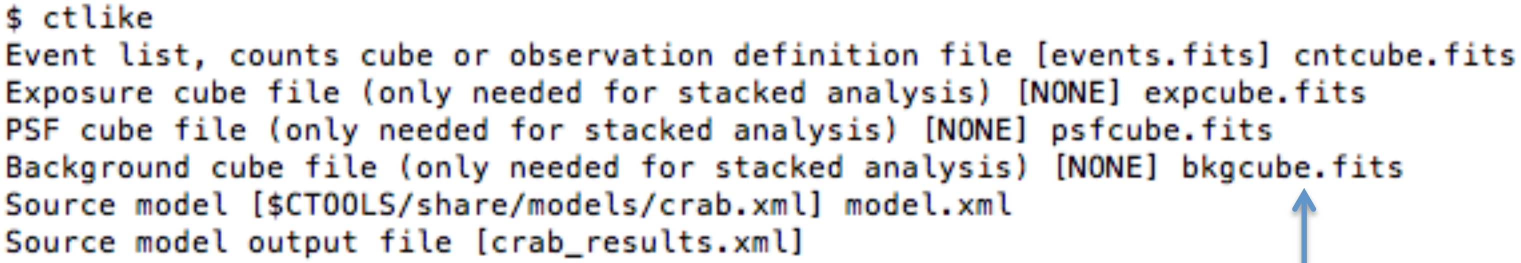


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<source_library title="source library">
  <source name="Crab" type="PointSource">
    <spectrum type="PowerLaw">
      <parameter name="Prefactor" value="5.7" error="0" scale="1e-16" min="1e-07" max="1000" free="1" />
      <parameter name="Index" value="2.48" error="0" 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>
  <source name="BackgroundModel" type="CTACubeBackground" instrument="CTA,HESS,MAGIC,VERITAS">
    <spectrum type="PowerLaw">
      <parameter name="Prefactor" value="1" error="0" scale="1" min="0" free="1" />
      <parameter name="Index" value="0" error="0" scale="1" min="-10" max="10" free="1" />
      <parameter name="Scale" value="1" scale="1e+06" free="0" />
    </spectrum>
  </source>
</source_library>
```

Stacked analysis

Analysis steps

7. Run ctlike

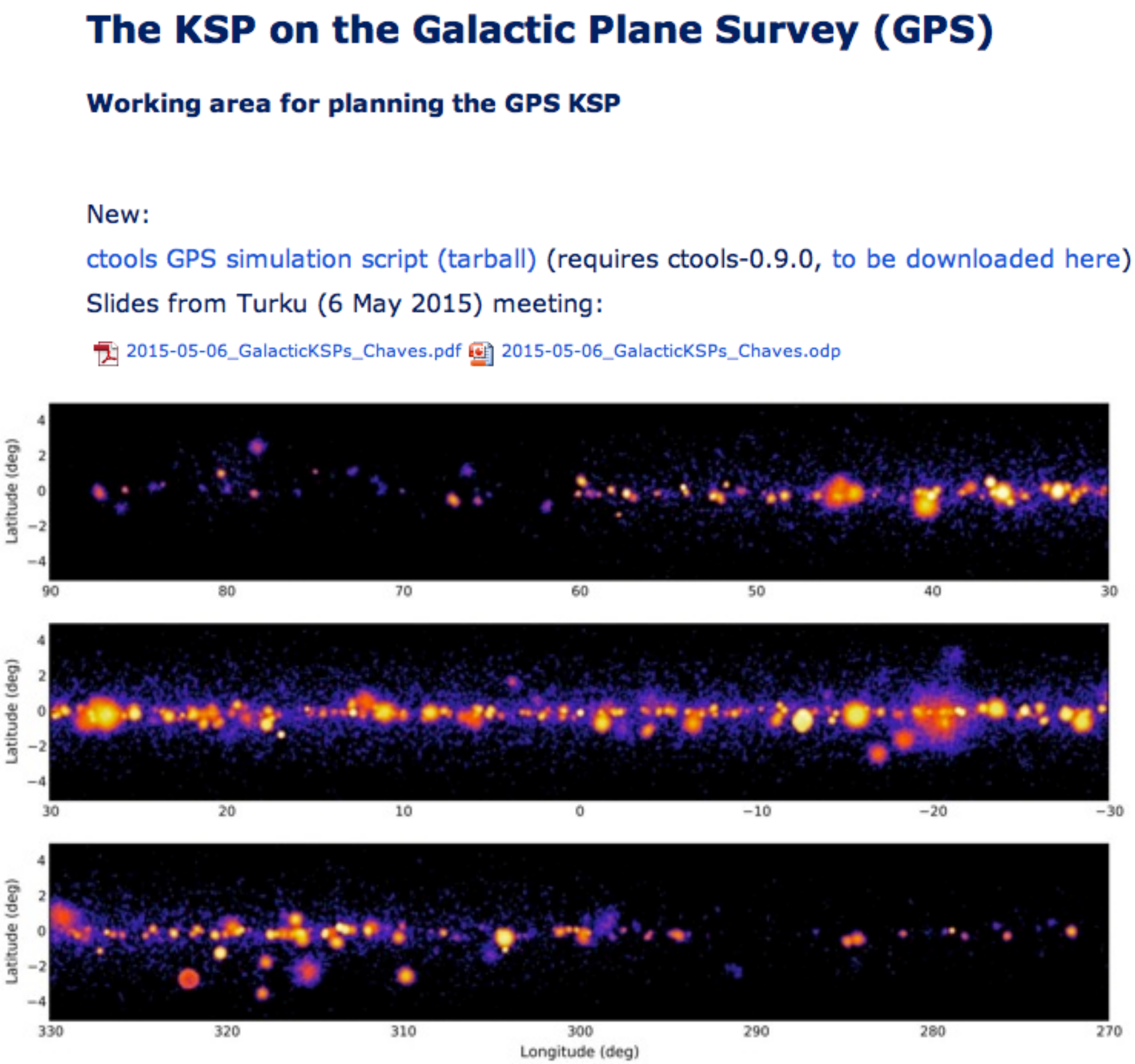


```
$ ctlike
Event list, counts cube or observation definition file [events.fits] cntcube.fits
Exposure cube file (only needed for stacked analysis) [NONE] expcube.fits
PSF cube file (only needed for stacked analysis) [NONE] psfcube.fits
Background cube file (only needed for stacked analysis) [NONE] bkgcube.fits
Source model [$CTOOLS/share/models/crab.xml] model.xml
Source model output file [crab_results.xml]
```

provide outputs from
ctbin, ctexpcube, ctpsfcube and ctbkgcube

Stacked analysis

https://portal.cta-observatory.org/WG/PHYS/SitePages/PHYS-KSP-GPS.aspx



The KSP on the Galactic Plane Survey (GPS)

Working area for planning the GPS KSP

New:

ctools GPS simulation script (tarball) (requires ctools-0.9.0, to be downloaded here)

Slides from Turku (6 May 2015) meeting:

2015-05-06_GalacticKSPs_Chaves.pdf 2015-05-06_GalacticKSPs_Chaves.odp

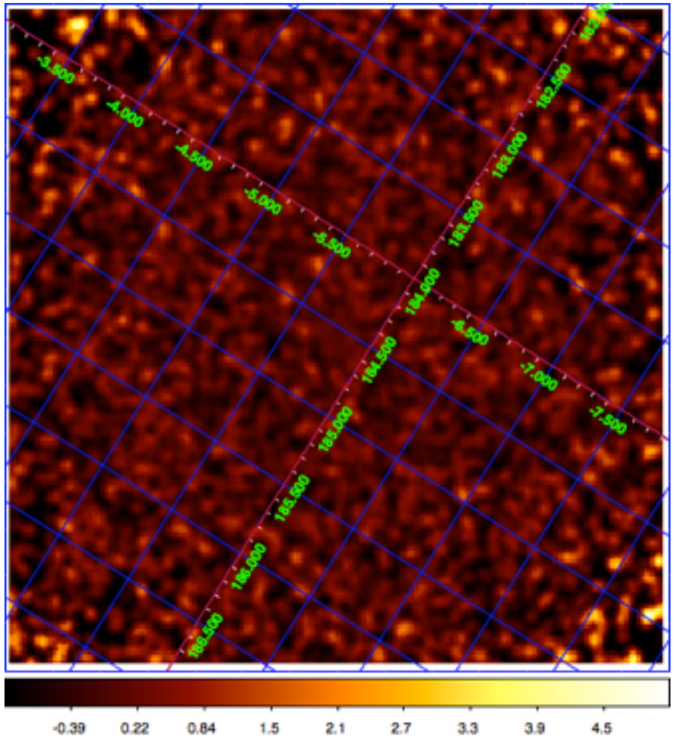
New scripts

csresmap	fit quality
csspec cslightcrv	likelihood related
csobsdef	observation definition

csresmap

Purpose: produce a residual counts map*

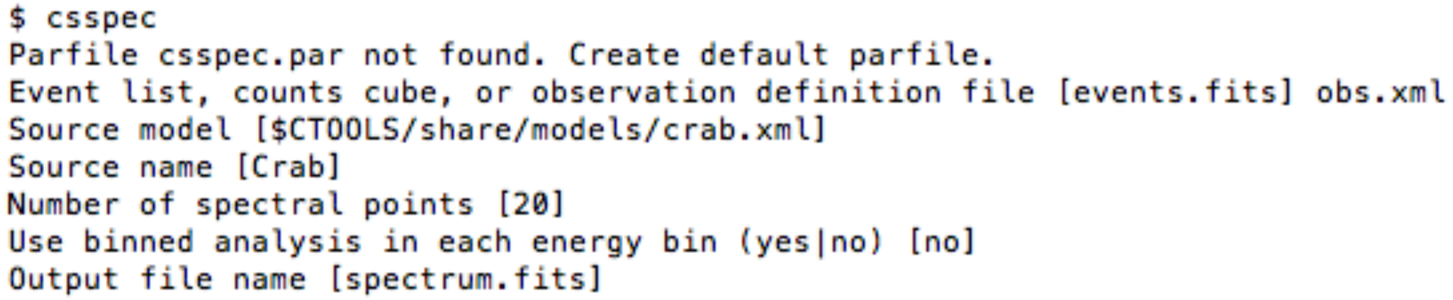
$$\underbrace{\mathrm{R}(\alpha,\delta)}_{\text{residual map}} = \frac{\sum_i \overbrace{\mathrm{N}(\alpha,\delta,E_i)}^{\text{counts}} - \overbrace{\mathrm{M}(\alpha,\delta,E_i)}^{\text{model}}}{\sum_i \mathrm{M}(\alpha,\delta,E_i)}$$



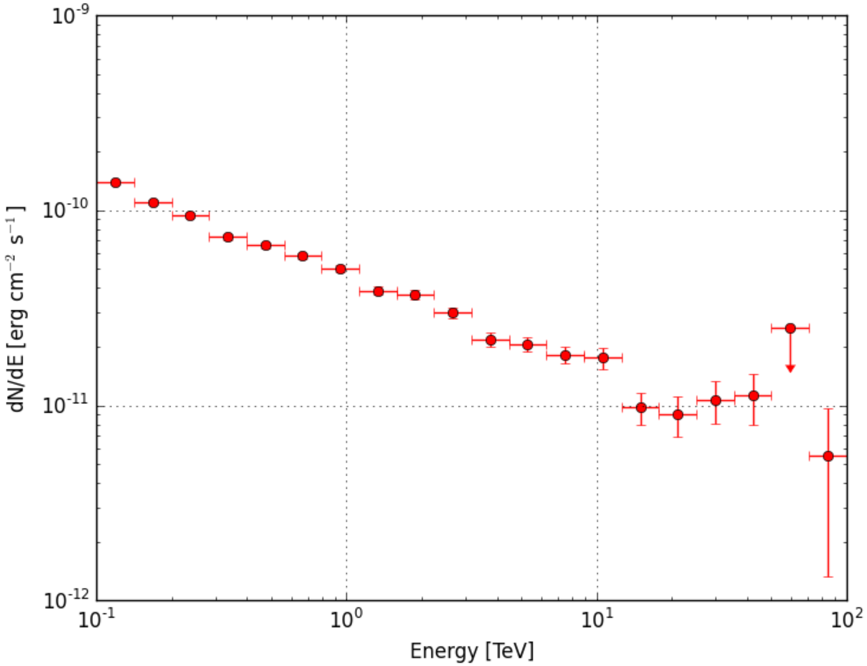
*does not yet work for stacked analysis

csspec

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



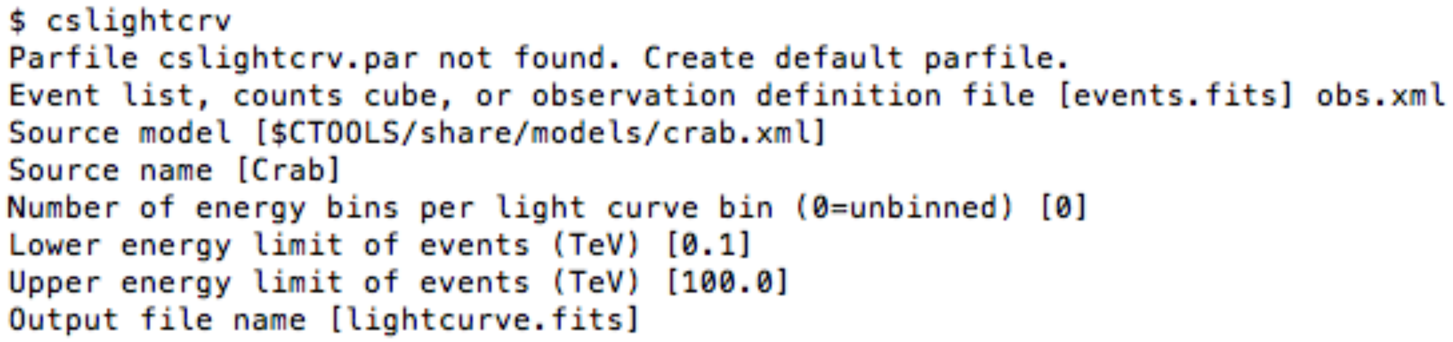
```
$ csspec
Parfile csspec.par not found. Create default parfile.
Event list, counts cube, or observation definition file [events.fits] obs.xml
Source model [$CTOOLS/share/models/crab.xml]
Source name [Crab]
Number of spectral points [20]
Use binned analysis in each energy bin (yes|no) [no]
Output file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



show_spectrum.py

cslightcrv

Purpose: runs ctlike in various temporal bands to produce a light curve (including upper limits)

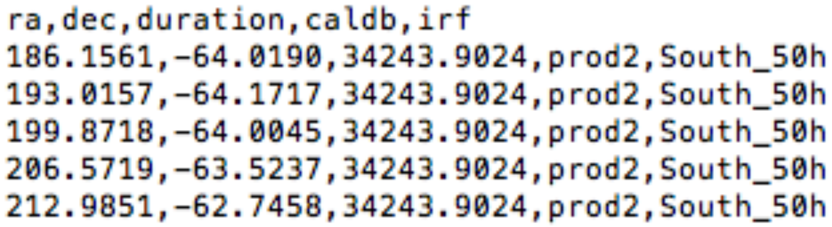


```
$ cslightcrv
Parfile cslightcrv.par not found. Create default parfile.
Event list, counts cube, or observation definition file [events.fits] obs.xml
Source model [$CTOOLS/share/models/crab.xml]
Source name [Crab]
Number of energy bins per light curve bin (0=unbinned) [0]
Lower energy limit of events (TeV) [0.1]
Upper energy limit of events (TeV) [100.0]
Output file name [lightcurve.fits]
```

no plotting script yet (maybe something to do at the sprint)

csobsdef

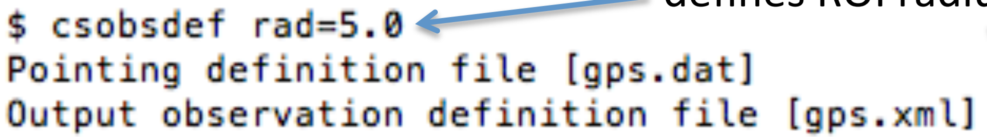
Purpose: generation of an observation definition XML file



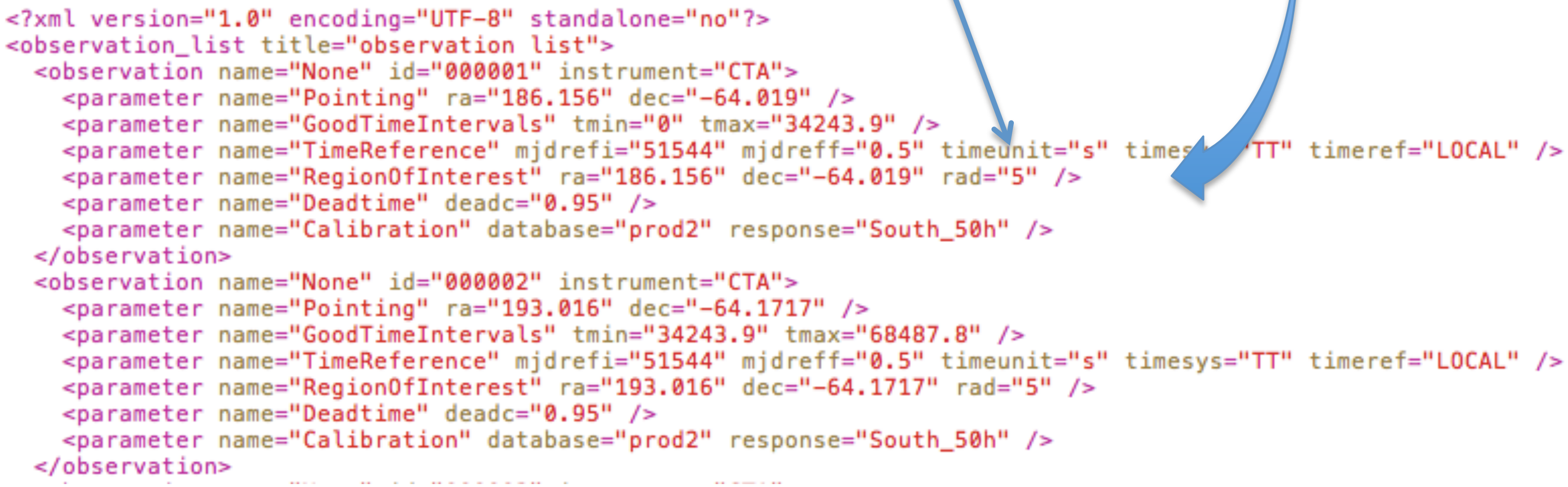
```
ra,dec,duration,caldb,irf
186.1561,-64.0190,34243.9024,prod2,South_50h
193.0157,-64.1717,34243.9024,prod2,South_50h
199.8718,-64.0045,34243.9024,prod2,South_50h
206.5719,-63.5237,34243.9024,prod2,South_50h
212.9851,-62.7458,34243.9024,prod2,South_50h
```

see script make_pointings.py for survey KSPs

defines ROI radius



```
$ csobsdef rad=5.0
Pointing definition file [gps.dat]
Output observation definition file [gps.xml]
```



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<observation_list title="observation list">
  <observation name="None" id="000001" instrument="CTA">
    <parameter name="Pointing" ra="186.156" dec="-64.019" />
    <parameter name="GoodTimeIntervals" tmin="0" tmax="34243.9" />
    <parameter name="TimeReference" mjdrefi="51544" mjdreff="0.5" timeunit="s" timesys="TT" timeref="LOCAL" />
    <parameter name="RegionOfInterest" ra="186.156" dec="-64.019" rad="5" />
    <parameter name="Deadtime" deadc="0.95" />
    <parameter name="Calibration" database="prod2" response="South_50h" />
  </observation>
  <observation name="None" id="000002" instrument="CTA">
    <parameter name="Pointing" ra="193.016" dec="-64.1717" />
    <parameter name="GoodTimeIntervals" tmin="34243.9" tmax="68487.8" />
    <parameter name="TimeReference" mjdrefi="51544" mjdreff="0.5" timeunit="s" timesys="TT" timeref="LOCAL" />
    <parameter name="RegionOfInterest" ra="193.016" dec="-64.1717" rad="5" />
    <parameter name="Deadtime" deadc="0.95" />
    <parameter name="Calibration" database="prod2" response="South_50h" />
  </observation>
```

Output can be directly fed into ctobssim …

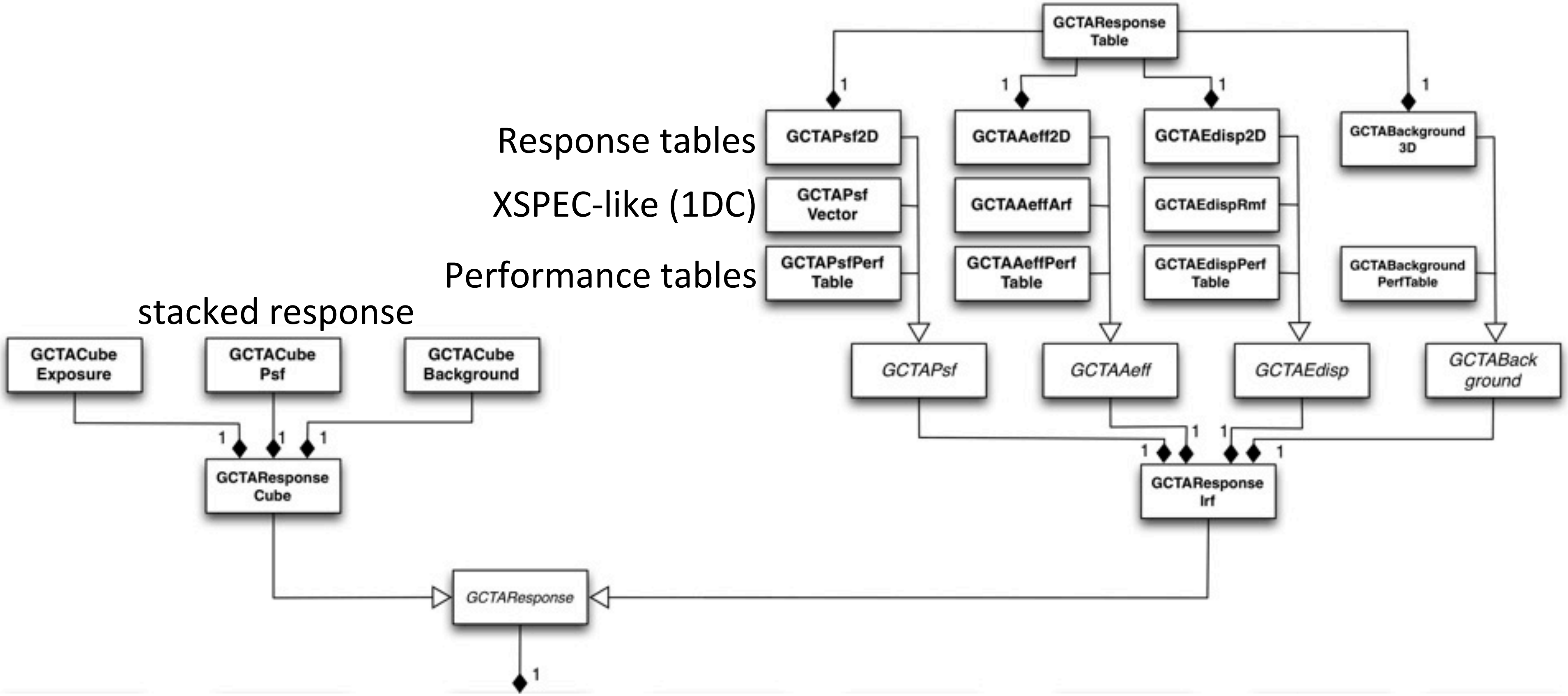
3. Instrument Response Functions

Current IRF handling

$$R_\gamma(\alpha',\delta',E'|\alpha,\delta,E,\vec{a}) = \underbrace{A_\gamma(\alpha,\delta,E,\vec{a})}_{\text{effective area (cm2)}} \times \underbrace{PSF(\alpha',\delta'|\alpha,\delta,E,\vec{a})}_{\text{point spread function}} \times \underbrace{D(E'|\alpha,\delta,E,\vec{a})}_{\text{energy dispersion}}$$

Full area, no angle cuts

$$1 = \int d\alpha'\, d\delta' PSF(\alpha',\delta'|\alpha,\delta,E,\vec{a}) \qquad 1 = \int dE' D(E'|\alpha,\delta,E,\vec{a})$$



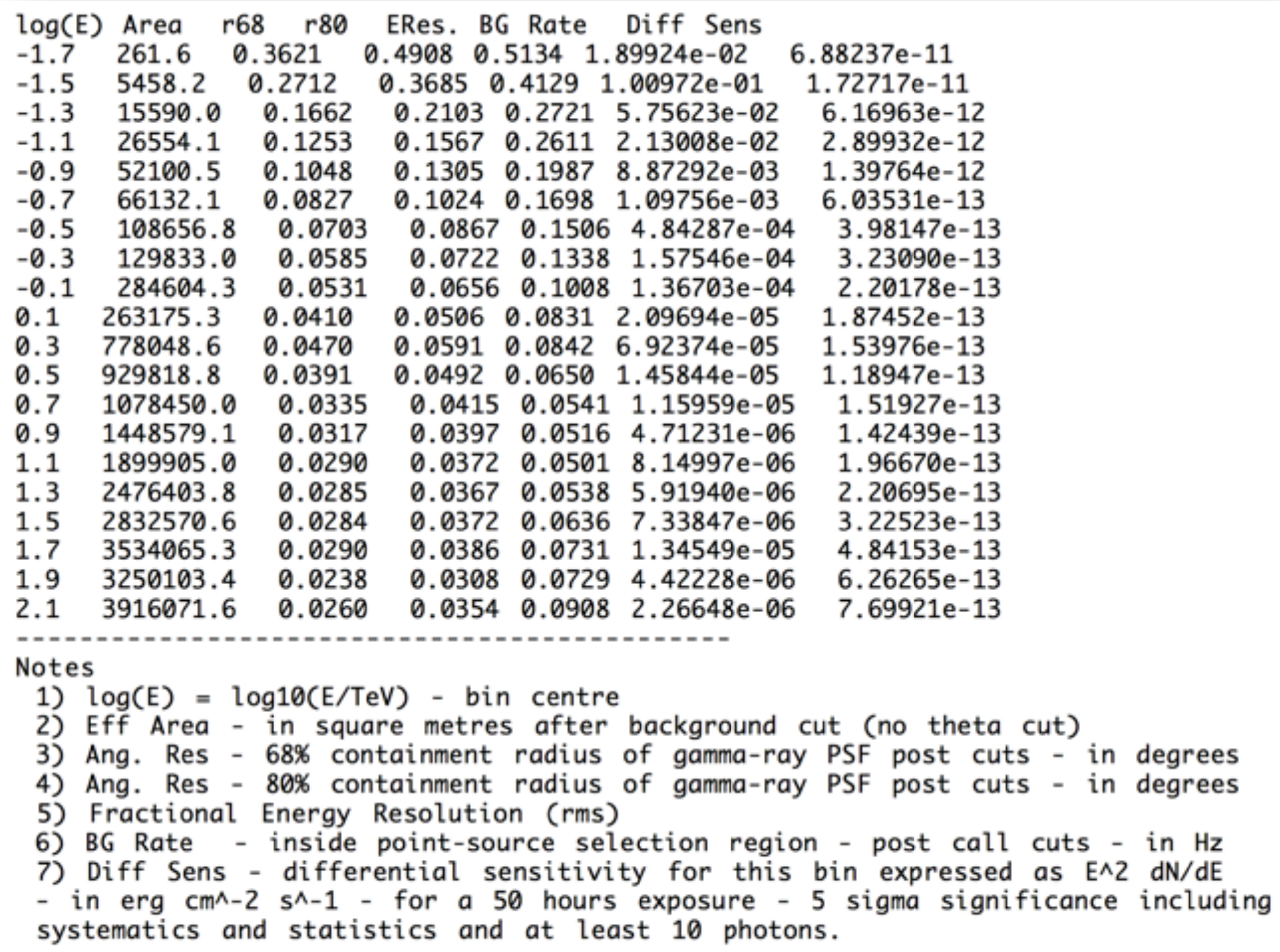
Response tables

XSPEC-like (1DC)

Performance tables

stacked response

Performance tables



```
log(E) Area    r68    r80    ERes.  BG Rate    Diff  Sens
-1.7    261.6    0.3621   0.4908 0.5134 1.89924e-02   6.88237e-11
-1.5    5458.2    0.2712   0.3685 0.4129 1.00972e-01   1.72717e-11
-1.3    15590.0    0.1662   0.2103 0.2721 5.75623e-02   6.16963e-12
-1.1    26554.1    0.1253   0.1567 0.2611 2.13008e-02   2.89932e-12
-0.9    52100.5    0.1048   0.1305 0.1987 8.87292e-03   1.39764e-12
-0.7    66132.1    0.0827   0.1024 0.1698 1.09756e-03   6.03531e-13
-0.5    108656.8    0.0703   0.0867 0.1506 4.84287e-04   3.98147e-13
-0.3    129833.0    0.0585   0.0722 0.1338 1.57546e-04   3.23090e-13
-0.1    284604.3    0.0531   0.0656 0.1008 1.36703e-04   2.20178e-13
0.1   263175.3    0.0410   0.0506 0.0831 2.09694e-05   1.87452e-13
0.3   778048.6    0.0470   0.0591 0.0842 6.92374e-05   1.53976e-13
0.5   929818.8    0.0391   0.0492 0.0650 1.45844e-05   1.18947e-13
0.7   1078450.0    0.0335   0.0415 0.0541 1.15959e-05   1.51927e-13
0.9   1448579.1    0.0317   0.0397 0.0516 4.71231e-06   1.42439e-13
1.1   1899905.0    0.0290   0.0372 0.0501 8.14997e-06   1.96670e-13
1.3   2476403.8    0.0285   0.0367 0.0538 5.91940e-06   2.20695e-13
1.5   2832570.6    0.0284   0.0372 0.0636 7.33847e-06   3.22523e-13
1.7   3534065.3    0.0290   0.0386 0.0731 1.34549e-05   4.84153e-13
1.9   3250103.4    0.0238   0.0308 0.0729 4.42228e-06   6.26265e-13
2.1   3916071.6    0.0260   0.0354 0.0908 2.26648e-06   7.69921e-13
--------------------------------------------
Notes
  1) log(E) = log10(E/TeV) - bin centre
  2) Eff Area - in square metres after background cut (no theta cut)
  3) Ang. Res - 68% containment radius of gamma-ray PSF post cuts - in degrees
  4) Ang. Res - 80% containment radius of gamma-ray PSF post cuts - in degrees
  5) Fractional Energy Resolution (rms)
  6) BG Rate  - inside point-source selection region - post call cuts - in Hz
  7) Diff Sens - differential sensitivity for this bin expressed as E^2 dN/dE
  - in erg cm^-2 s^-1 - for a 50 hours exposure - 5 sigma significance including
  systematics and statistics and at least 10 photons.
```

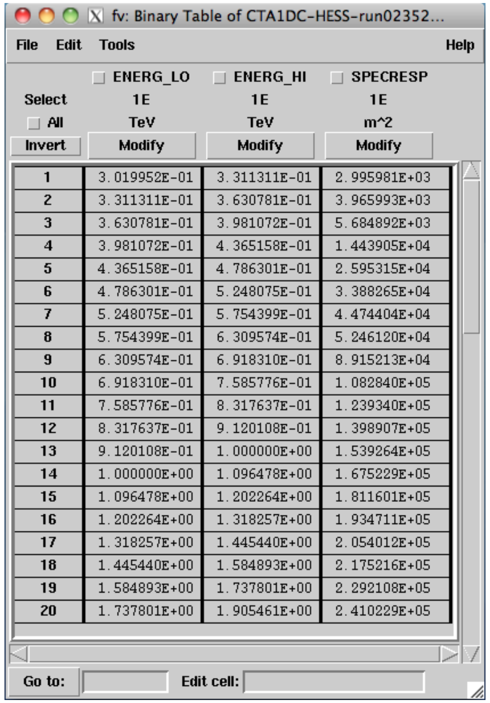
GCTAAeffPerfTable
GCTAPsfPerfTable
GCTAEdispPerfTable
GCTABackgroundPerfTable

Only on-axis information

A_{eff} and B_{rate} off-axis dependence modelled using $B(\theta) \propto \exp\left(-\frac{1}{2}\frac{\theta^4}{\sigma^2}\right)$

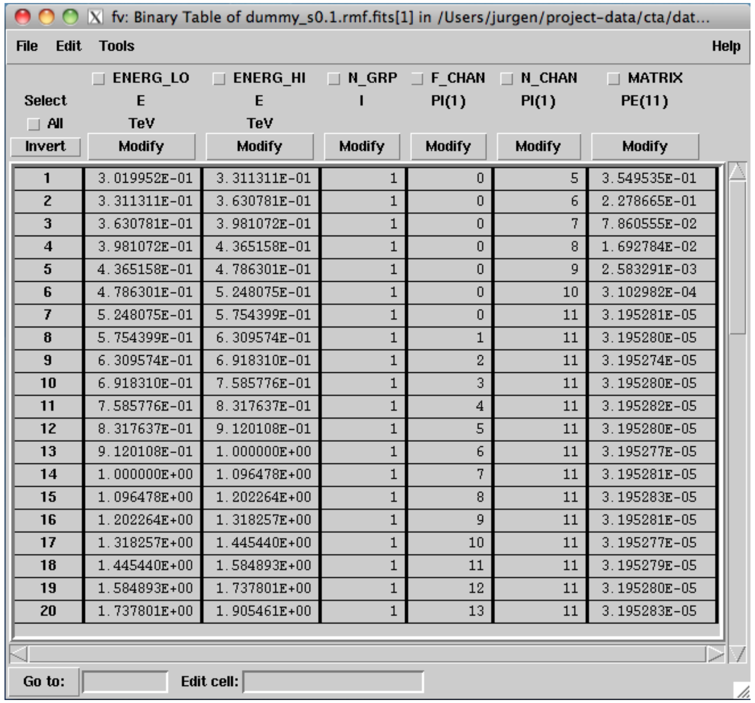
Gaussians assumed for PSF and energy dispersion

ARF, RMF, PSF vectors



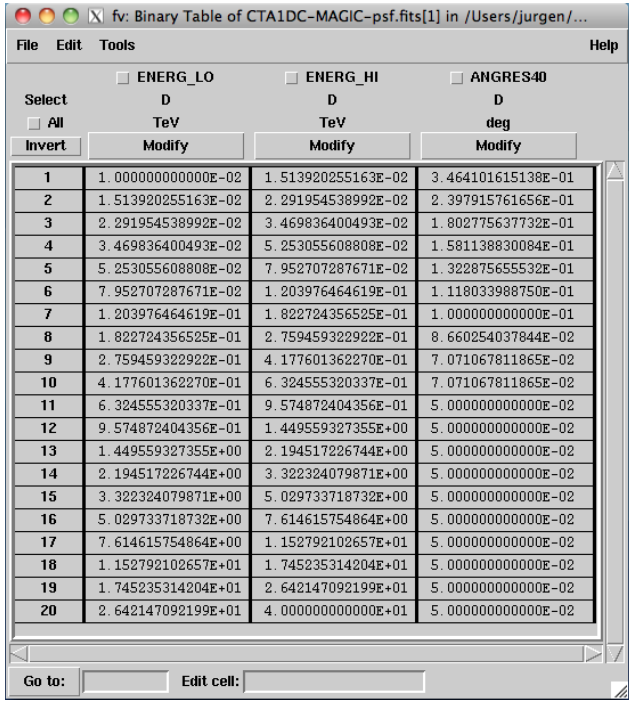
fv: Binary Table of CTA1DC-HESS-run02352...

	□ ENERG_LO	□ ENERG_HI	□ SPECRESP
Select	1E	1E	1E
□ All	TeV	TeV	m^2
Invert	Modify	Modify	Modify
1	3.019952E-01	3.311311E-01	2.995981E+03
2	3.311311E-01	3.630781E-01	3.965993E+03
3	3.630781E-01	3.981072E-01	5.684892E+03
4	3.981072E-01	4.365158E-01	1.443905E+04
5	4.365158E-01	4.786301E-01	2.595315E+04
6	4.786301E-01	5.248075E-01	3.388265E+04
7	5.248075E-01	5.754399E-01	4.474404E+04
8	5.754399E-01	6.309574E-01	5.246120E+04
9	6.309574E-01	6.918310E-01	8.915213E+04
10	6.918310E-01	7.585776E-01	1.082840E+05
11	7.585776E-01	8.317637E-01	1.239340E+05
12	8.317637E-01	9.120108E-01	1.398907E+05
13	9.120108E-01	1.000000E+00	1.539264E+05
14	1.000000E+00	1.096478E+00	1.675229E+05
15	1.096478E+00	1.202264E+00	1.811601E+05
16	1.202264E+00	1.318257E+00	1.934711E+05
17	1.318257E+00	1.445440E+00	2.054012E+05
18	1.445440E+00	1.584893E+00	2.175216E+05
19	1.584893E+00	1.737801E+00	2.292108E+05
20	1.737801E+00	1.905461E+00	2.410229E+05



fv: Binary Table of dummy_s0.1.rmf.fits[1] in /Users/jurgen/project-data/cta/dat...

	□ ENERG_LO	□ ENERG_HI	□ N_GRP	□ F_CHAN	□ N_CHAN	□ MATRIX
Select	E	E	I	PI(1)	PI(1)	PE(11)
□ All	TeV	TeV				
Invert	Modify	Modify	Modify	Modify	Modify	Modify
1	3.019952E-01	3.311311E-01	1	0	5	3.549535E-01
2	3.311311E-01	3.630781E-01	1	0	6	2.278665E-01
3	3.630781E-01	3.981072E-01	1	0	7	7.860555E-02
4	3.981072E-01	4.365158E-01	1	0	8	1.692784E-02
5	4.365158E-01	4.786301E-01	1	0	9	2.583291E-03
6	4.786301E-01	5.248075E-01	1	0	10	3.102982E-04
7	5.248075E-01	5.754399E-01	1	0	11	3.195281E-05
8	5.754399E-01	6.309574E-01	1	1	11	3.195280E-05
9	6.309574E-01	6.918310E-01	1	2	11	3.195274E-05
10	6.918310E-01	7.585776E-01	1	3	11	3.195280E-05
11	7.585776E-01	8.317637E-01	1	4	11	3.195282E-05
12	8.317637E-01	9.120108E-01	1	5	11	3.195280E-05
13	9.120108E-01	1.000000E+00	1	6	11	3.195277E-05
14	1.000000E+00	1.096478E+00	1	7	11	3.195281E-05
15	1.096478E+00	1.202264E+00	1	8	11	3.195283E-05
16	1.202264E+00	1.318257E+00	1	9	11	3.195281E-05
17	1.318257E+00	1.445440E+00	1	10	11	3.195277E-05
18	1.445440E+00	1.584893E+00	1	11	11	3.195279E-05
19	1.584893E+00	1.737801E+00	1	12	11	3.195280E-05
20	1.737801E+00	1.905461E+00	1	13	11	3.195283E-05



fv: Binary Table of CTA1DC-MAGIC-psf.fits[1] in /Users/jurgen/...

	□ ENERG_LO	□ ENERG_HI	□ ANGRES40
Select	D	D	D
□ All	TeV	TeV	deg
Invert	Modify	Modify	Modify
1	1.000000000000E-02	1.513920255163E-02	3.464101615138E-01
2	1.513920255163E-02	2.291954538992E-02	2.397915761656E-01
3	2.291954538992E-02	3.469836400493E-02	1.802775637732E-01
4	3.469836400493E-02	5.253055608808E-02	1.581138830084E-01
5	5.253055608808E-02	7.952707287671E-02	1.322875655532E-01
6	7.952707287671E-02	1.203976464619E-01	1.118033988750E-01
7	1.203976464619E-01	1.822724356525E-01	1.000000000000E-01
8	1.822724356525E-01	2.759459322922E-01	8.660254037844E-02
9	2.759459322922E-01	4.177601362270E-01	7.071067811865E-02
10	4.177601362270E-01	6.324555320337E-01	7.071067811865E-02
11	6.324555320337E-01	9.574872404356E-01	5.000000000000E-02
12	9.574872404356E-01	1.449559327355E+00	5.000000000000E-02
13	1.449559327355E+00	2.194517226744E+00	5.000000000000E-02
14	2.194517226744E+00	3.322324079871E+00	5.000000000000E-02
15	3.322324079871E+00	5.029733718732E+00	5.000000000000E-02
16	5.029733718732E+00	7.614615754864E+00	5.000000000000E-02
17	7.614615754864E+00	1.152792102657E+01	5.000000000000E-02
18	1.152792102657E+01	1.745235314204E+01	5.000000000000E-02
19	1.745235314204E+01	2.642147092199E+01	5.000000000000E-02
20	2.642147092199E+01	4.000000000000E+01	5.000000000000E-02

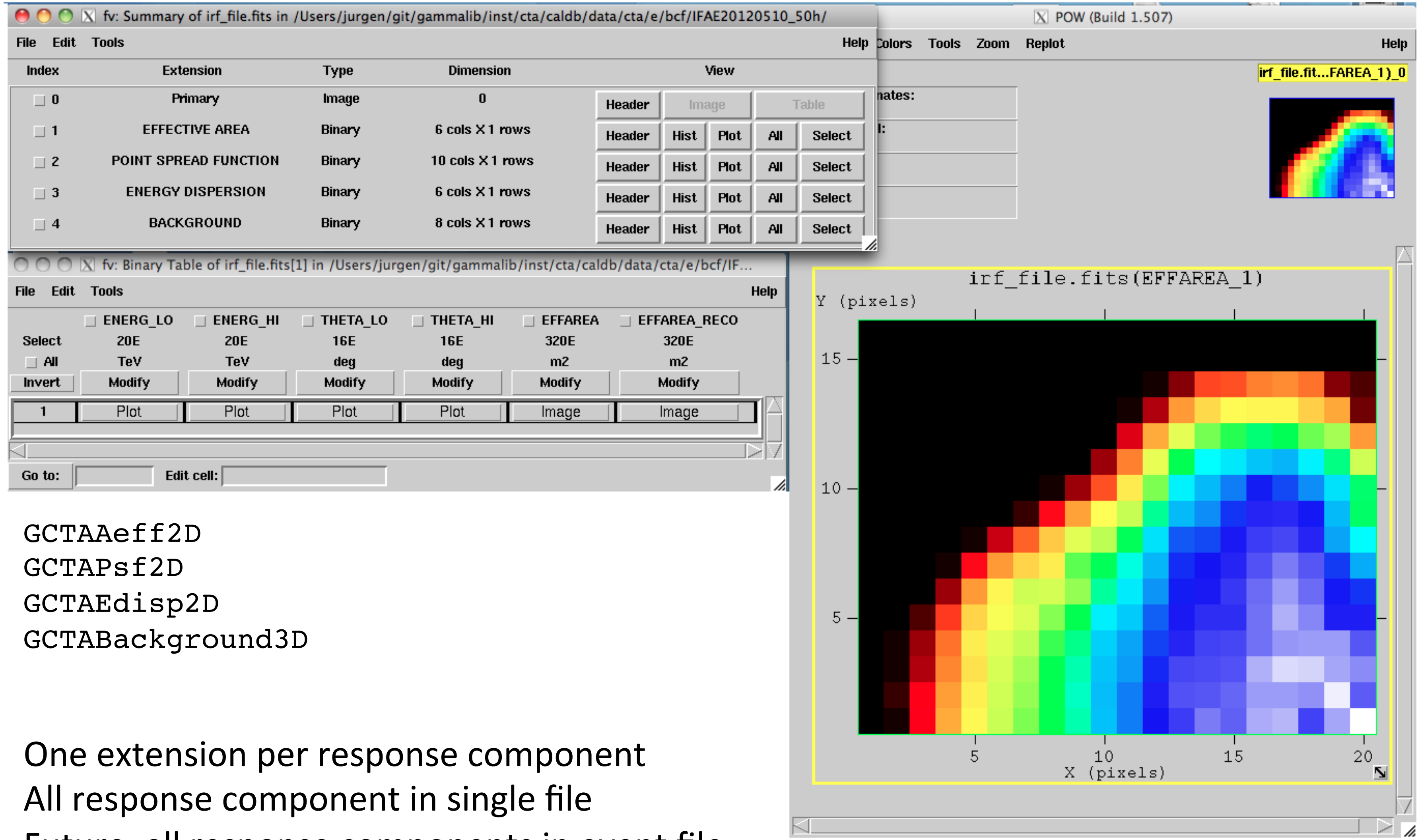
```
GCTAAeffArf
GCTAPsfVector
GCTAEdispRmf
```

Only on-axis information

A_{eff} off-axis dependence modelled using $B(\theta) \propto \exp\left(-\frac{1}{2}\frac{\theta^4}{\sigma^2}\right)$

Gaussian assumed for PSF

Response tables



fv: Summary of irf_file.fits in /Users/jurgen/git/gammalib/inst/cta/caldb/data/cta/e/bcf/IFAE20120510_50h/

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	6 cols X 1 rows	Header Hist Plot All Select
□ 4	BACKGROUND	Binary	8 cols X 1 rows	Header Hist Plot All Select

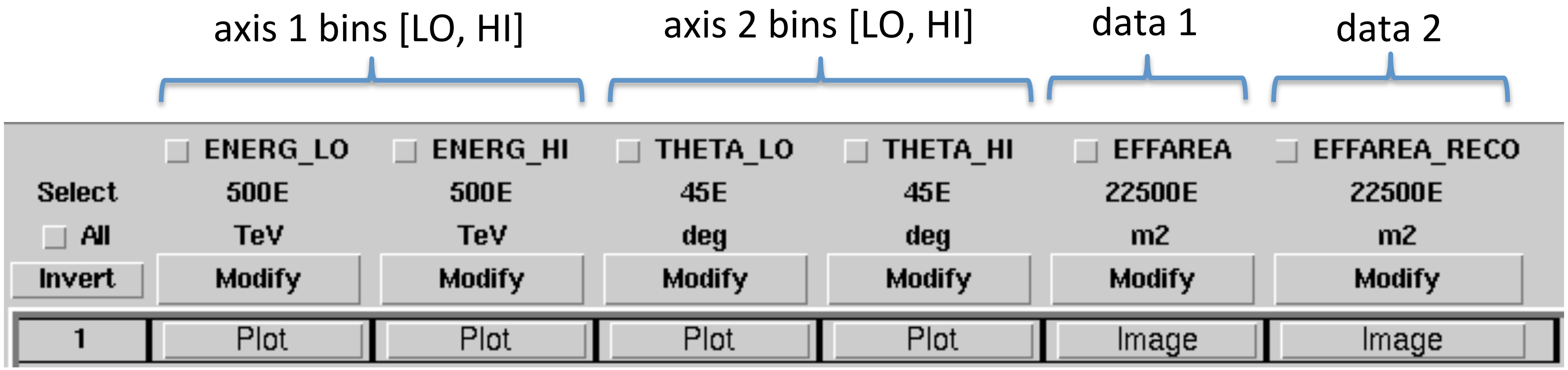
fv: Binary Table of irf_file.fits[1] in /Users/jurgen/git/gammalib/inst/cta/caldb/data/cta/e/bcf/IF...

Select	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ EFFAREA	□ EFFAREA_RECO
	20E	20E	16E	16E	320E	320E
□ All	TeV	TeV	deg	deg	m2	m2
Invert	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image

GCTAAeff2D
GCTAPsf2D
GCTAEdisp2D
GCTABackground3D

One extension per response component
All response component in single file
Future: all response components in event file

Response tables



axis 1 bins [LO, HI]

axis 2 bins [LO, HI]

data 1

data 2

	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ EFFAREA	□ EFFAREA_RECO
Select	500E	500E	45E	45E	22500E	22500E
□ All	TeV	TeV	deg	deg	m2	m2
Invert	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image

Format of Fermi/LAT instrument response files

Can handle n-dimensional cubes (don't need to be contiguous)

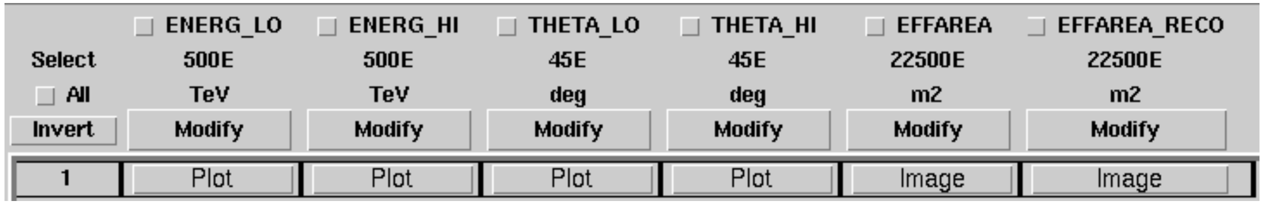
Can handle arbitrary number of data blocks

Can handle parametric models (each data block is a parameter)

Handling of this format implement by class GCTAResponseTable

Response tables

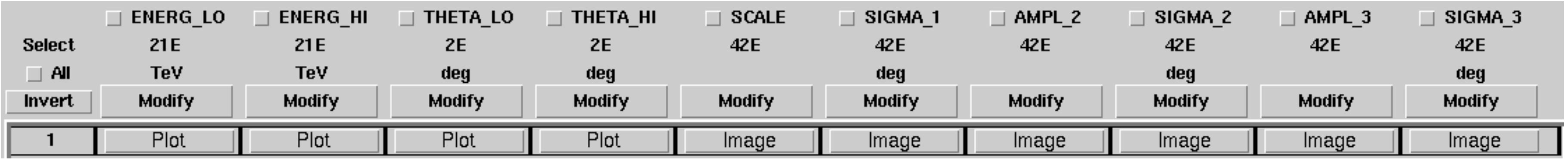
Effective area



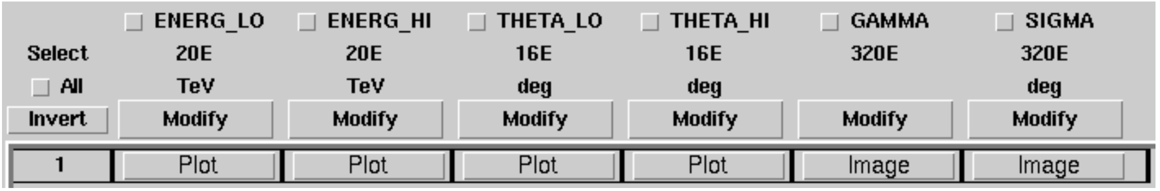
	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ EFFAREA	□ EFFAREA_RECO
Select	500E	500E	45E	45E	22500E	22500E
□ All	TeV	TeV	deg	deg	m2	m2
Invert	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image

- Function of energy and off axis angle
- Store A_{eff} values as function or true or measured energy

Point spread function



	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ SCALE	□ SIGMA_1	□ AMPL_2	□ SIGMA_2	□ AMPL_3	□ SIGMA_3
Select	21E	21E	2E	2E	42E	42E	42E	42E	42E	42E
□ All	TeV	TeV	deg	deg		deg		deg		deg
Invert	Modify	Modify	Modify	Modify	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image	Image	Image	Image	Image

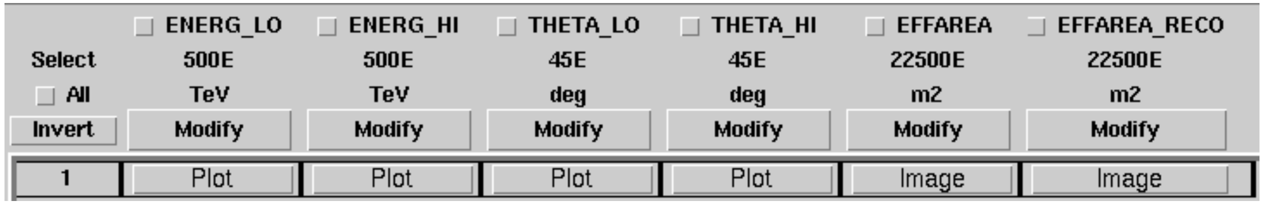


	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ GAMMA	□ SIGMA
Select	20E	20E	16E	16E	320E	320E
□ All	TeV	TeV	deg	deg		deg
Invert	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image

- Function of energy and off axis angle
- Two parametric variants: 3-Gaussians (6 parameters), King function (2 parameters)

Response tables

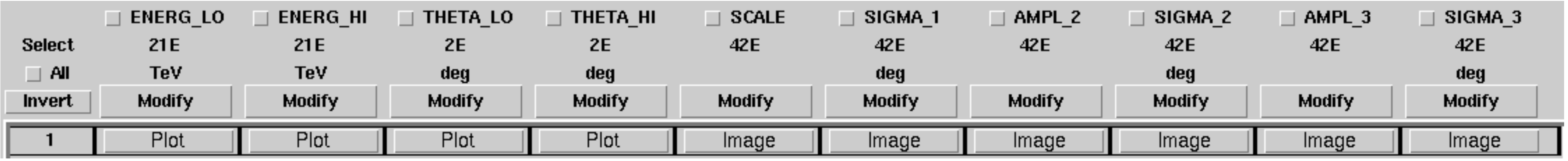
Effective area



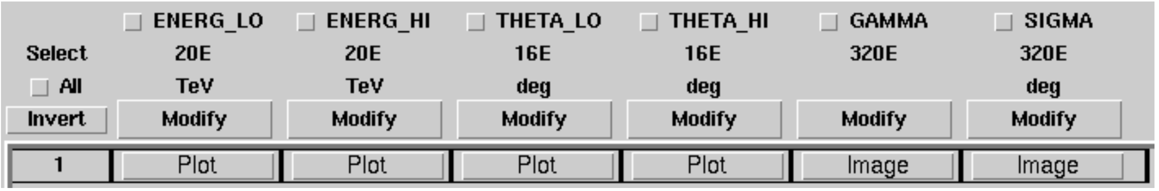
	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ EFFAREA	□ EFFAREA_RECO
Select	500E	500E	45E	45E	22500E	22500E
□ All	TeV	TeV	deg	deg	m2	m2
Invert	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image

- Function of energy and off axis angle
- Store A_{eff} values as function or true or measured energy

Point spread function



	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ SCALE	□ SIGMA_1	□ AMPL_2	□ SIGMA_2	□ AMPL_3	□ SIGMA_3
Select	21E	21E	2E	2E	42E	42E	42E	42E	42E	42E
□ All	TeV	TeV	deg	deg		deg		deg		deg
Invert	Modify	Modify	Modify	Modify	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image	Image	Image	Image	Image

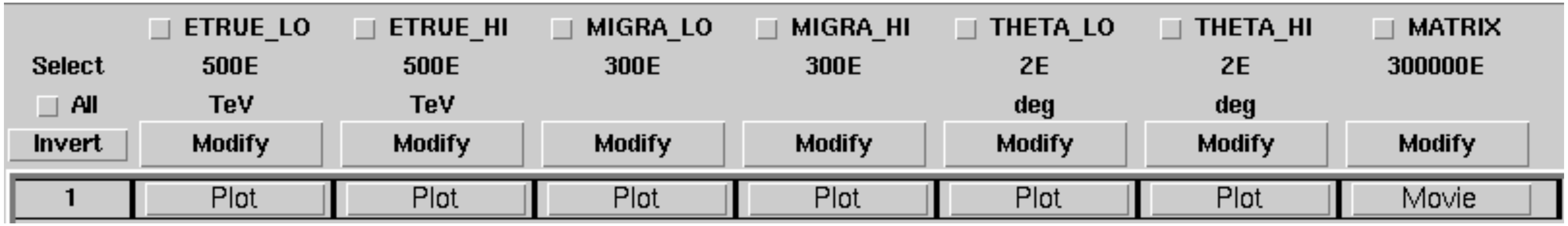


	□ ENERG_LO	□ ENERG_HI	□ THETA_LO	□ THETA_HI	□ GAMMA	□ SIGMA
Select	20E	20E	16E	16E	320E	320E
□ All	TeV	TeV	deg	deg		deg
Invert	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Image	Image

- Function of energy and off axis angle
- Two parametric variants: 3-Gaussians (6 parameters), King function (2 parameters)

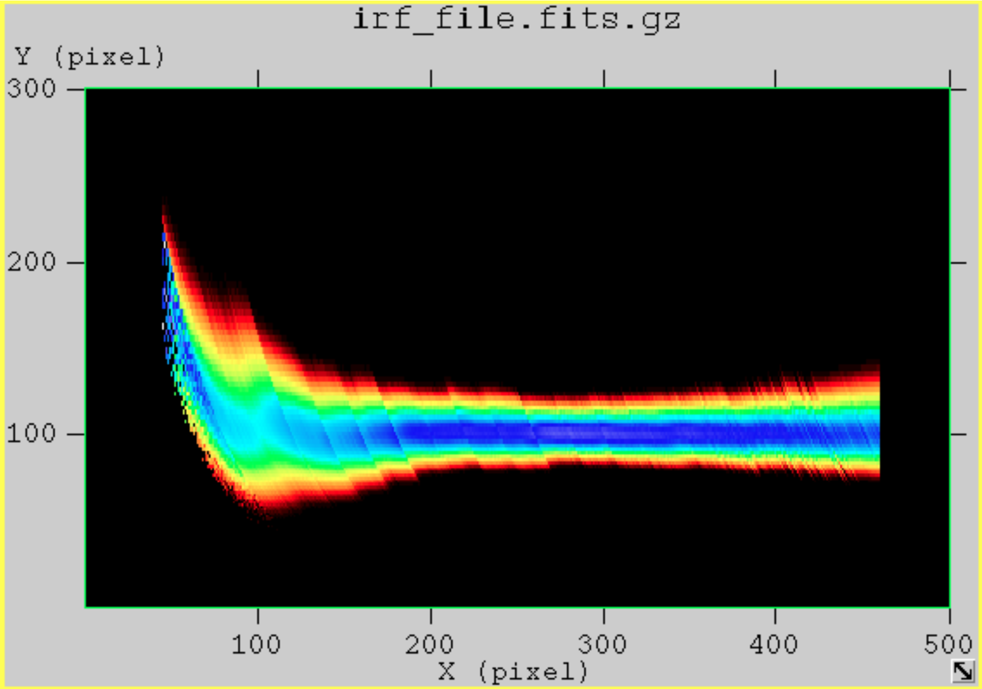
Response tables

Energy dispersion



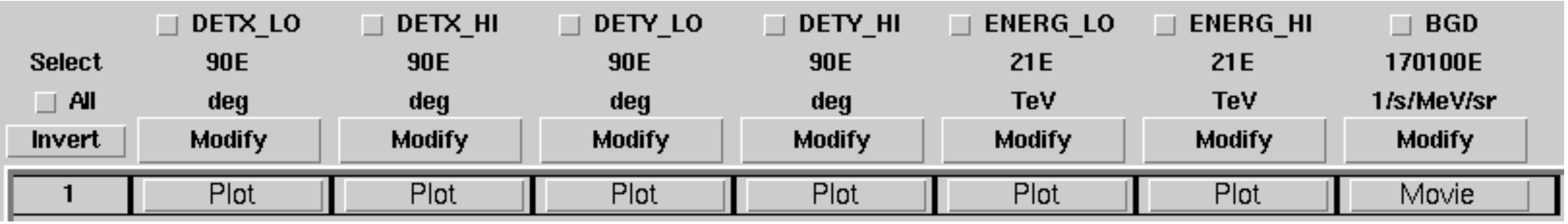
	□ ETRUE_LO	□ ETRUE_HI	□ MIGRA_LO	□ MIGRA_HI	□ THETA_LO	□ THETA_HI	□ MATRIX
Select	500E	500E	300E	300E	2E	2E	300000E
□ All	TeV	TeV			deg	deg	
Invert	Modify	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Plot	Plot	Movie

- Function of true energy, E_{reco}/E_{true} and off axis angle
- Store migration matrix (3D)



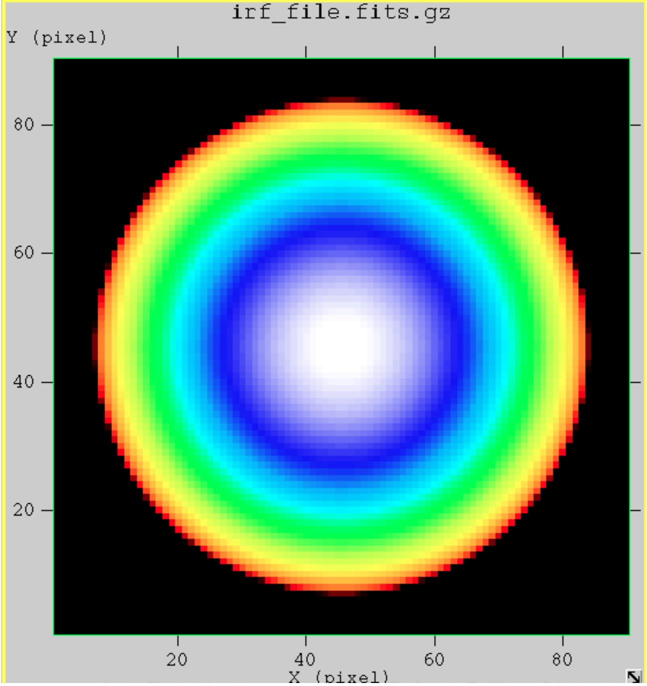
Response tables

Background rates

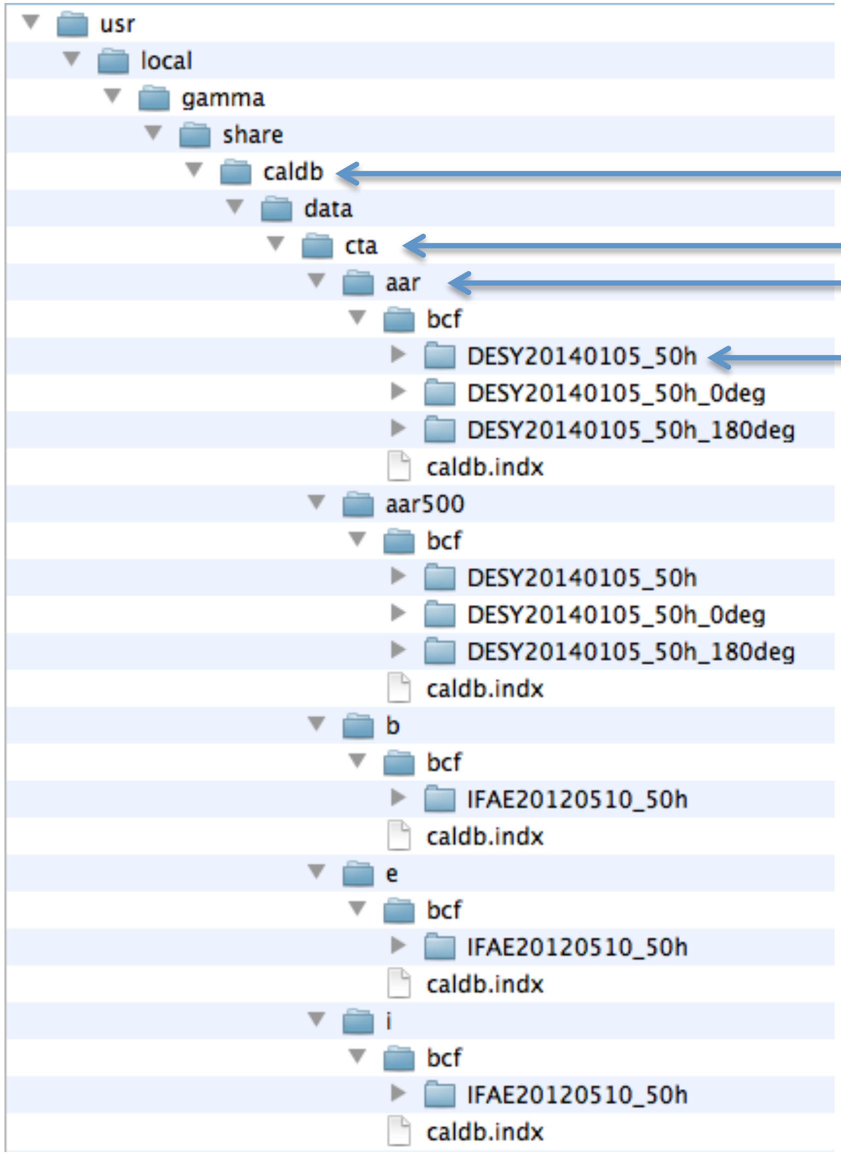


	□ DETX_LO	□ DETX_HI	□ DETY_LO	□ DETY_HI	□ ENERG_LO	□ ENERG_HI	□ BGD
Select	90E	90E	90E	90E	21E	21E	170100E
□ All	deg	deg	deg	deg	TeV	TeV	1/s/MeV/sr
Invert	Modify	Modify	Modify	Modify	Modify	Modify	Modify
1	Plot	Plot	Plot	Plot	Plot	Plot	Movie

- Function of DETX, DETY and measured energy
- Store rates per energy and solid angle (3D)



Calibration database usage



```
usr
  local
    gamma
      share
        caldb
          data
            cta
              aar
                bcf
                  DESY20140105_50h
                  DESY20140105_50h_0deg
                  DESY20140105_50h_180deg
                caldb.indx
              aar500
                bcf
                  DESY20140105_50h
                  DESY20140105_50h_0deg
                  DESY20140105_50h_180deg
                caldb.indx
              b
                bcf
                  IFAE20120510_50h
                caldb.indx
              e
                bcf
                  IFAE20120510_50h
                caldb.indx
              i
                bcf
                  IFAE20120510_50h
                caldb.indx
```

Set calibration database root:

```
export CALDB=/usr/local/gamma/share/caldb
```

mission

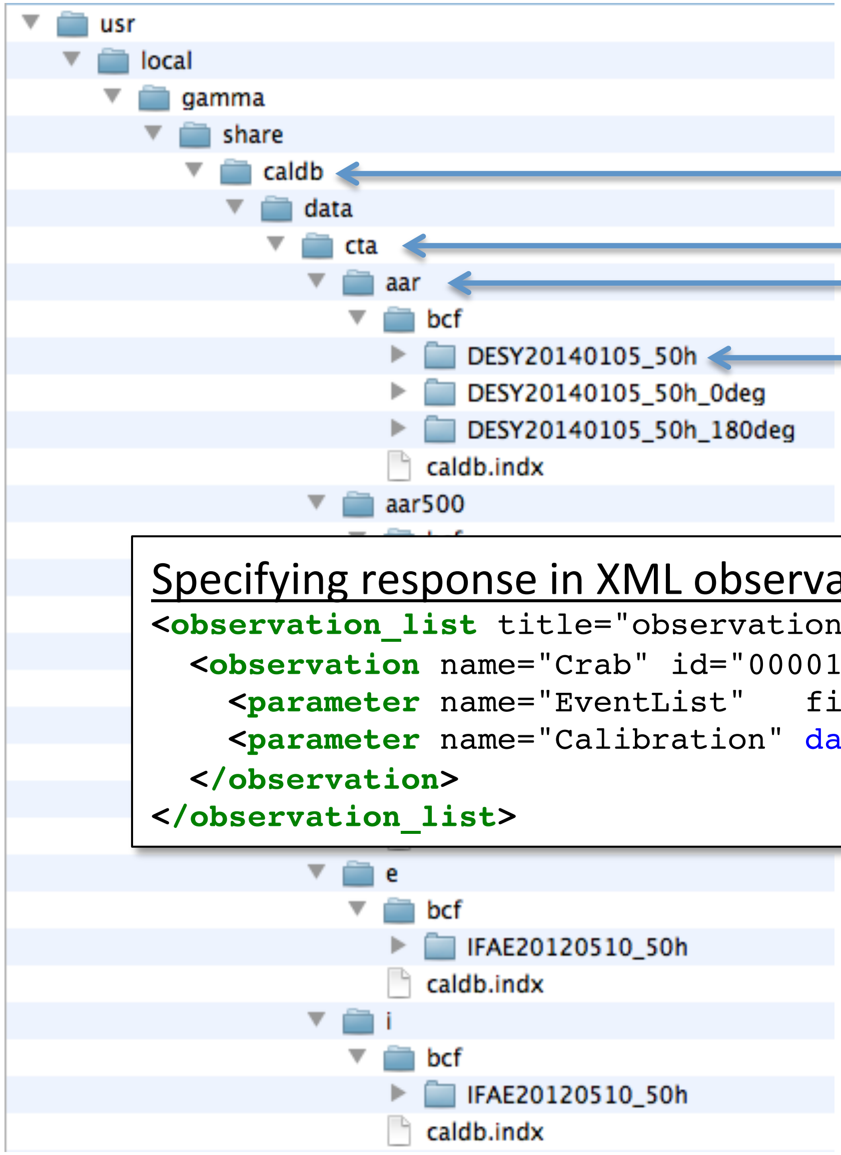
instrument

rspname

Specifying response as input parameters:

```
$ ctobssim
Model [$CTOOLS/share/models/crab.xml]
Calibration database [aar]
Instrument response function [DESY20140105_50h]
RA of pointing (degrees) (0-360) [83.63]
Dec of pointing (degrees) (-90-90) [22.01]
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]
```

Calibration database usage



```
usr
  local
    gamma
      share
        caldb
          data
            cta
              aar
                bcf
                  DESY20140105_50h
                  DESY20140105_50h_0deg
                  DESY20140105_50h_180deg
                caldb.indx
              aar500
              e
                bcf
                  IFAE20120510_50h
                caldb.indx
              i
                bcf
                  IFAE20120510_50h
                caldb.indx
```

Set calibration database root:

```
export CALDB=/usr/local/gamma/share/caldb
```

mission

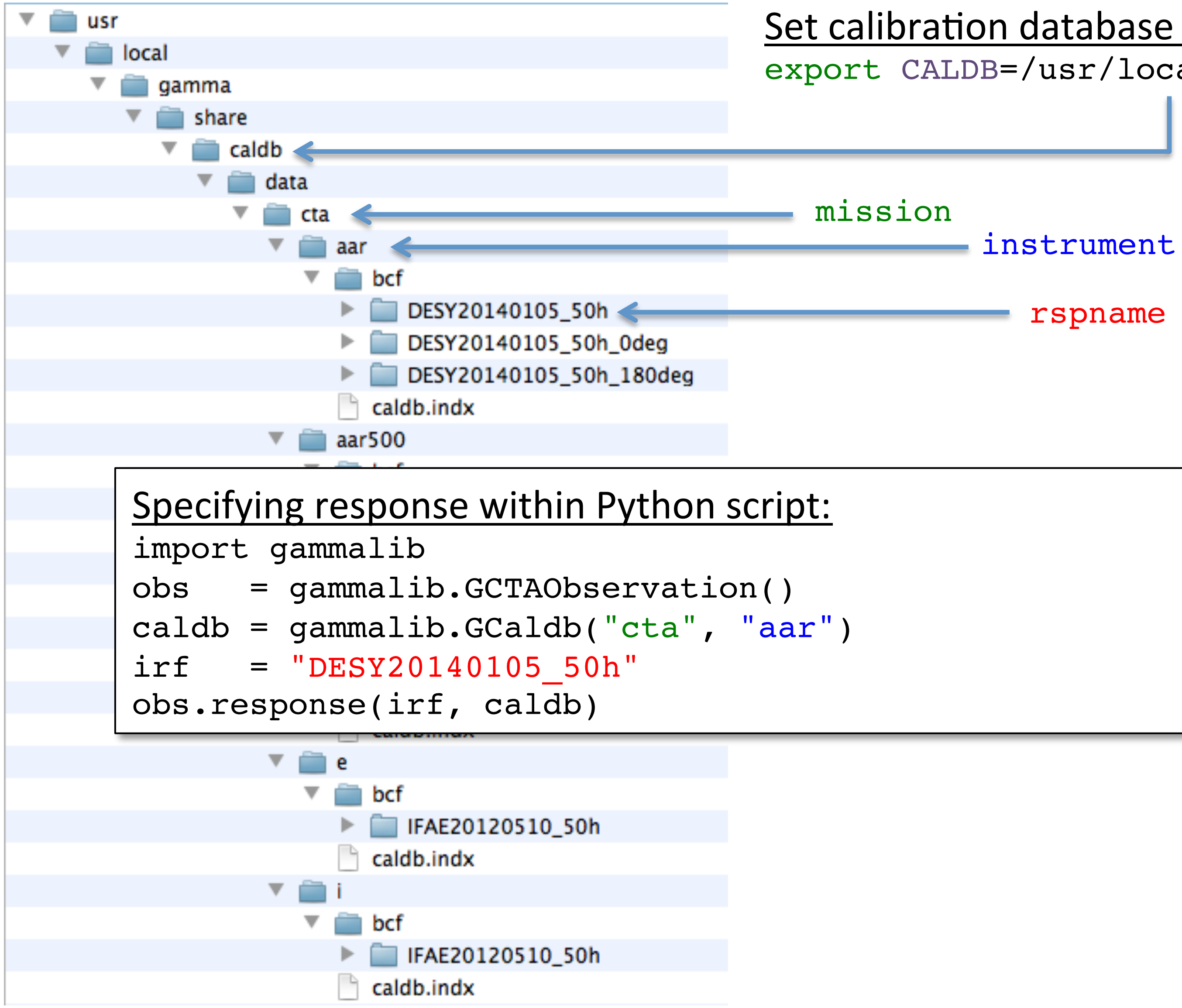
instrument

rspname

Specifying response in XML observation definition file:

```
<observation_list title="observation library">
  <observation name="Crab" id="00001" instrument="CTA">
    <parameter name="EventList"   file="events.fits"/>
    <parameter name="Calibration" database="aar" response="DESY20140105_50h"/>
  </observation>
</observation_list>
```

Calibration database usage



```
usr
  local
    gamma
      share
        caldb
          data
            cta
              aar
                bcf
                  DESY20140105_50h
                  DESY20140105_50h_0deg
                  DESY20140105_50h_180deg
                caldb.indx
              aar500
              ...
              e
                bcf
                  IFAE20120510_50h
                caldb.indx
              i
                bcf
                  IFAE20120510_50h
                caldb.indx
```

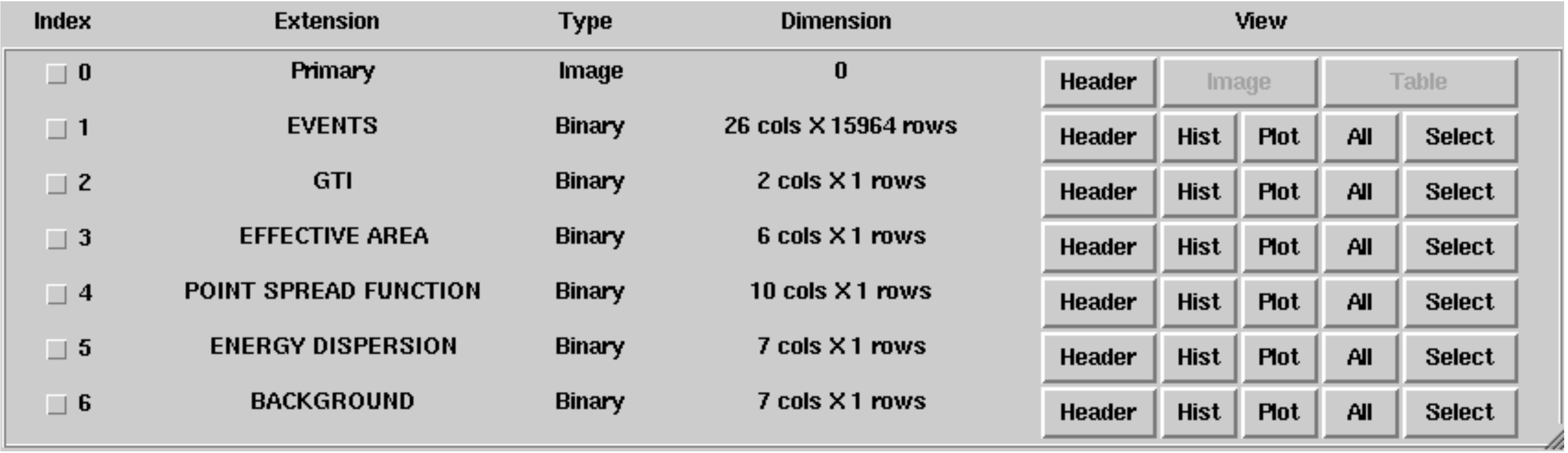
Set calibration database root:

```
export CALDB=/usr/local/gamma/share/caldb
```

Specifying response within Python script:

```
import gammalib
obs   = gammalib.GCTAObservation()
caldb = gammalib.GCaldb("cta", "aar")
irf   = "DESY20140105_50h"
obs.response(irf, caldb)
```

Evolutions

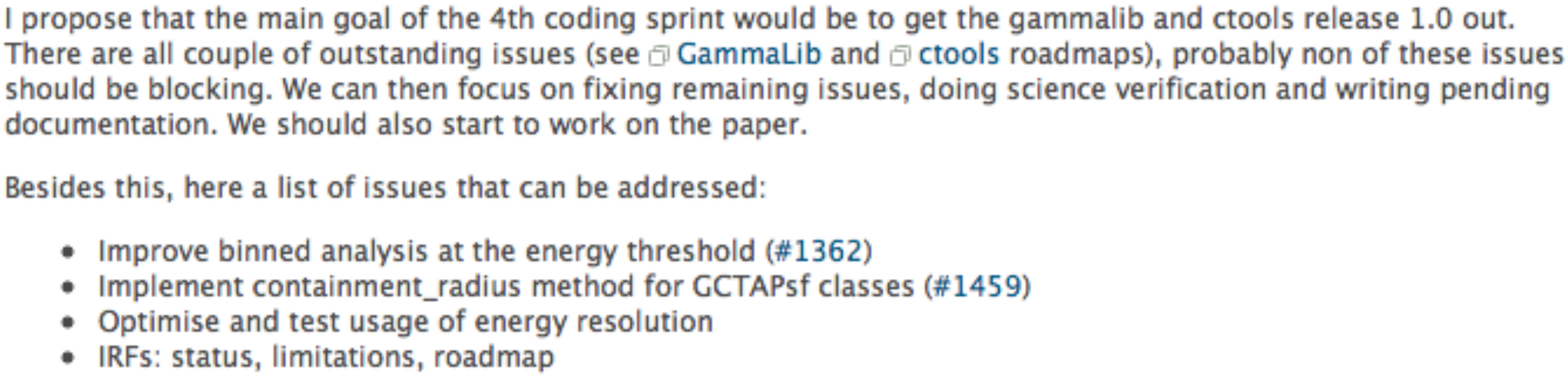


Index	Extension	Type	Dimension	View
□ 0	Primary	Image	0	Header Image Table
□ 1	EVENTS	Binary	26 cols X 15964 rows	Header Hist Plot All Select
□ 2	GTI	Binary	2 cols X 1 rows	Header Hist Plot All Select
□ 3	EFFECTIVE AREA	Binary	6 cols X 1 rows	Header Hist Plot All Select
□ 4	POINT SPREAD FUNCTION	Binary	10 cols X 1 rows	Header Hist Plot All Select
□ 5	ENERGY DISPERSION	Binary	7 cols X 1 rows	Header Hist Plot All Select
□ 6	BACKGROUND	Binary	7 cols X 1 rows	Header Hist Plot All Select

- Ultimately we would like to have a CTA events file that looks like this (event selection and instrument response functions are tied together; it makes sense to store them together).
- Auxiliary information could be added (for example pointing or weather information)
- For each observation (aka run) the user would get a single, self contained file. This considerably simplifies data distribution and data handling and minimizes the danger of improper usage.
- This would make the usage of a calibration database obsolete.

4. Goals of this sprint

Goals of this sprint

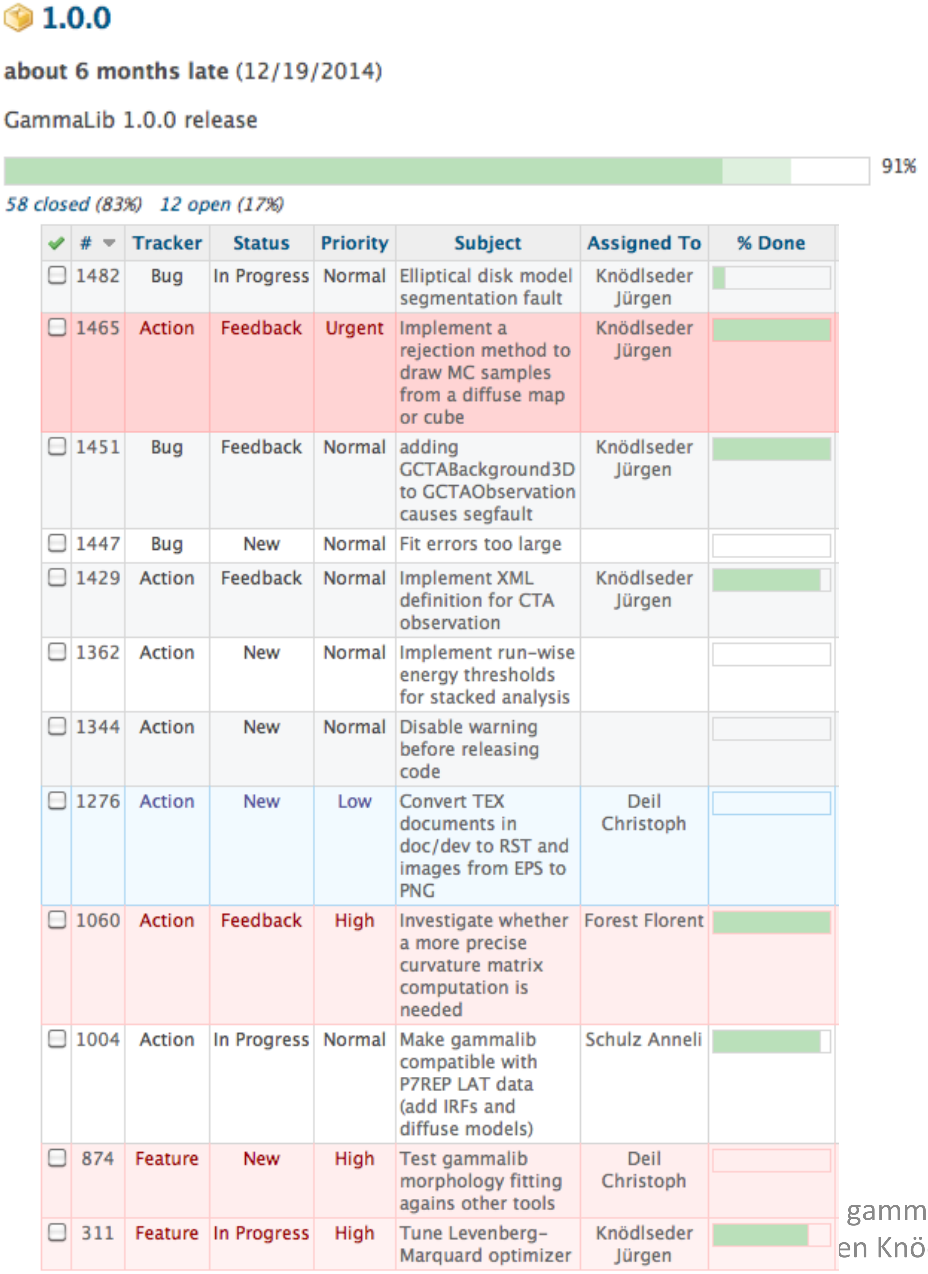


I propose that the main goal of the 4th coding sprint would be to get the gammalib and ctools release 1.0 out. There are all couple of outstanding issues (see GammaLib and ctools roadmaps), probably non of these issues should be blocking. We can then focus on fixing remaining issues, doing science verification and writing pending documentation. We should also start to work on the paper.

Besides this, here a list of issues that can be addressed:

- Improve binned analysis at the energy threshold (#1362)
- Implement containment_radius method for GCTAPsf classes (#1459)
- Optimise and test usage of energy resolution
- IRFs: status, limitations, roadmap

Roadmap towards gammalib release 1.0



1.0.0

about 6 months late (12/19/2014)

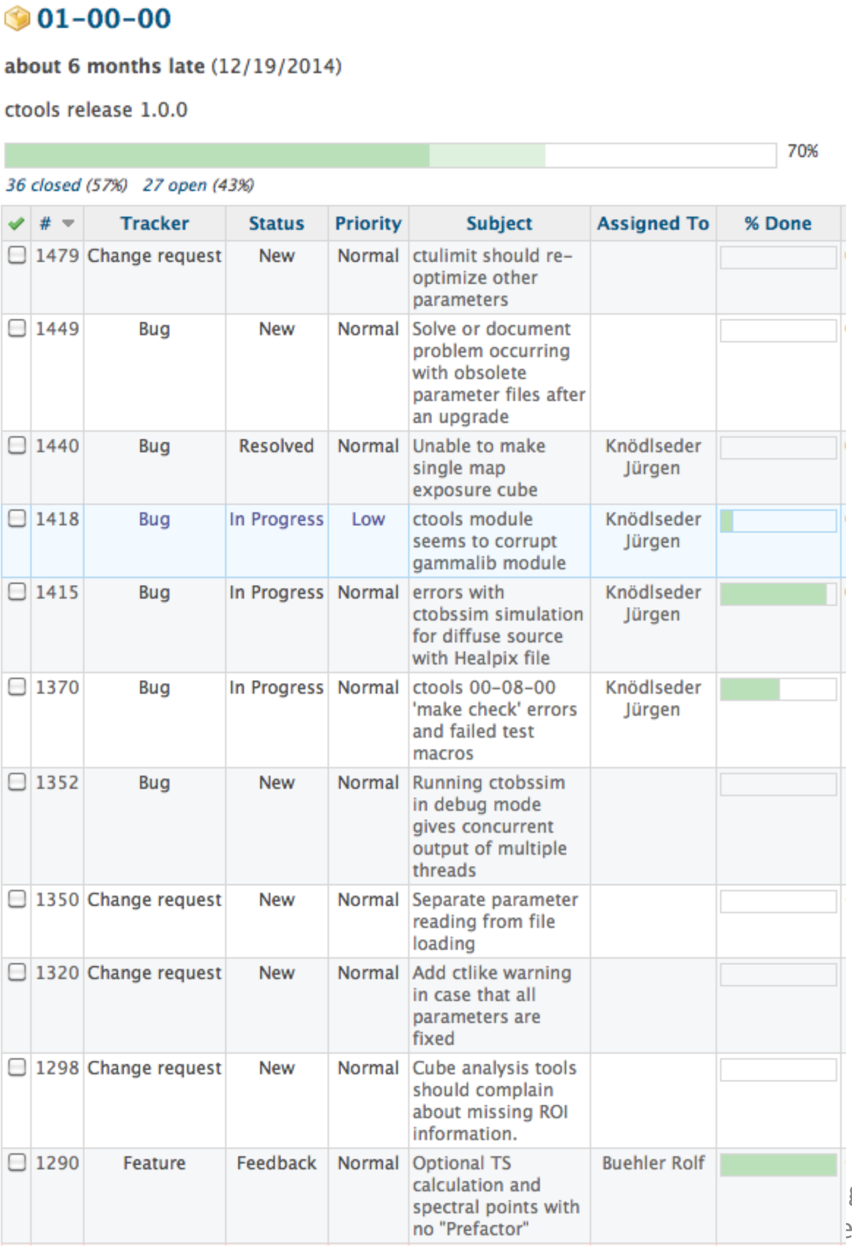
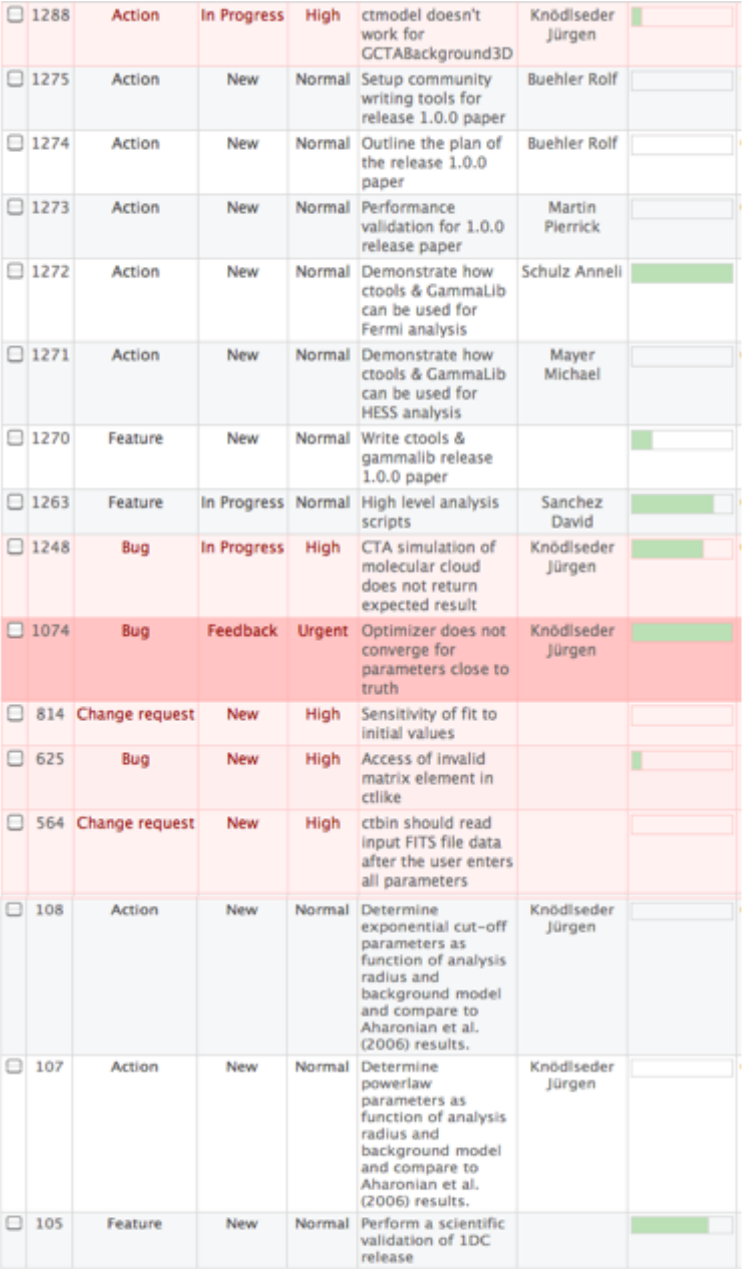
GammaLib 1.0.0 release

91%

58 closed (83%) *12 open (17%)*

✓	#	Tracker	Status	Priority	Subject	Assigned To	% Done
□	1482	Bug	In Progress	Normal	Elliptical disk model segmentation fault	Knödlseder Jürgen	
□	1465	Action	Feedback	Urgent	Implement a rejection method to draw MC samples from a diffuse map or cube	Knödlseder Jürgen	
□	1451	Bug	Feedback	Normal	adding GCTABackground3D to GCTAObservation causes segfault	Knödlseder Jürgen	
□	1447	Bug	New	Normal	Fit errors too large		
□	1429	Action	Feedback	Normal	Implement XML definition for CTA observation	Knödlseder Jürgen	
□	1362	Action	New	Normal	Implement run-wise energy thresholds for stacked analysis		
□	1344	Action	New	Normal	Disable warning before releasing code		
□	1276	Action	New	Low	Convert TEX documents in doc/dev to RST and images from EPS to PNG	Deil Christoph	
□	1060	Action	Feedback	High	Investigate whether a more precise curvature matrix computation is needed	Forest Florent	
□	1004	Action	In Progress	Normal	Make gammalib compatible with P7REP LAT data (add IRFs and diffuse models)	Schulz Anneli	
□	874	Feature	New	High	Test gammalib morphology fitting agains other tools	Deil Christoph	
□	311	Feature	In Progress	High	Tune Levenberg-Marquard optimizer	Knödlseder Jürgen	

Roadmap towards ctools release 1.0

01-00-00

about 6 months late (12/19/2014)

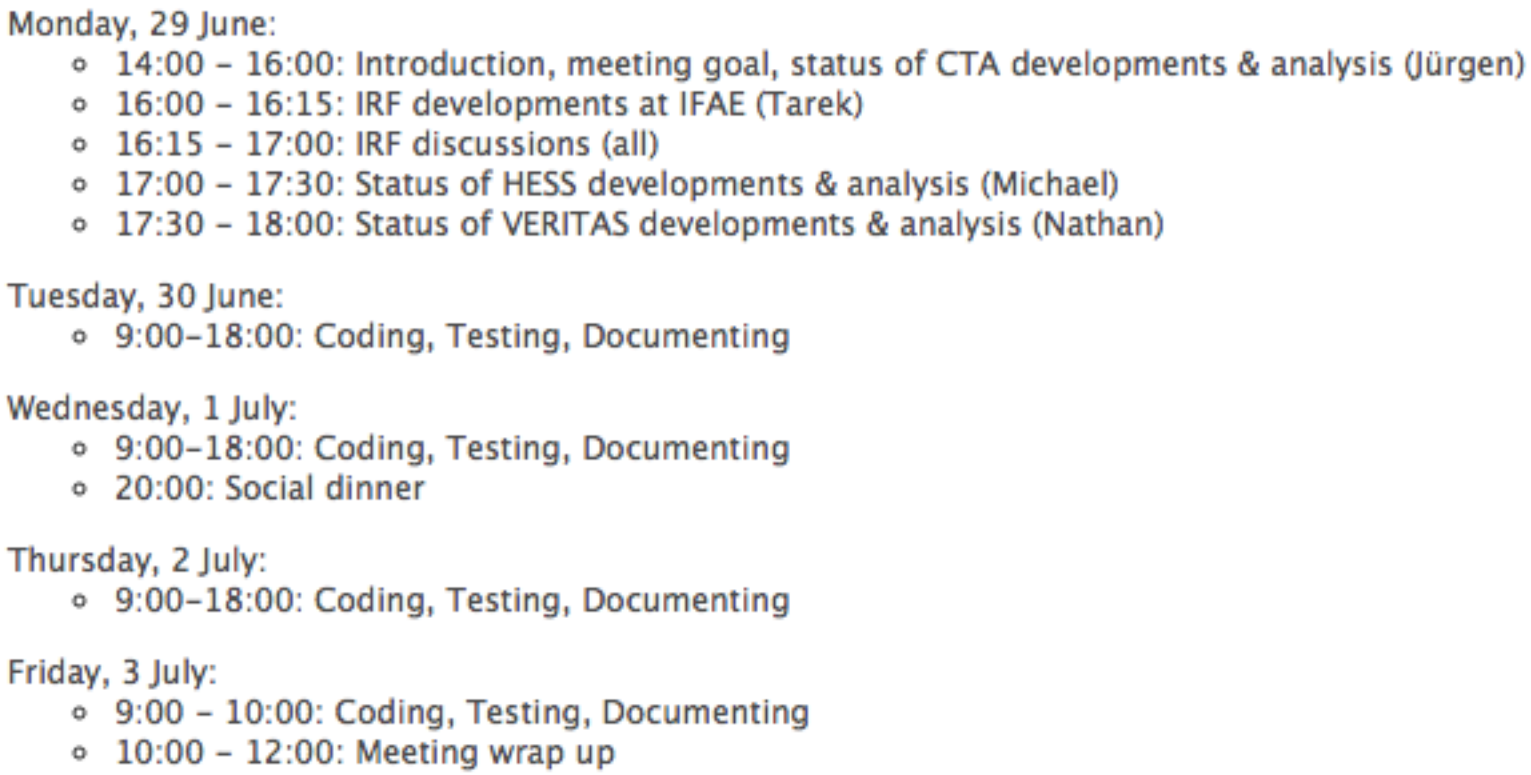
ctools release 1.0.0

70%

36 closed (57%) *27 open (43%)*

✓	#	Tracker	Status	Priority	Subject	Assigned To	% Done
□	1479	Change request	New	Normal	ctulimit should re-optimize other parameters		
□	1449	Bug	New	Normal	Solve or document problem occurring with obsolete parameter files after an upgrade		
□	1440	Bug	Resolved	Normal	Unable to make single map exposure cube	Knödlseder Jürgen	
□	1418	Bug	In Progress	Low	ctools module seems to corrupt gammalib module	Knödlseder Jürgen	
□	1415	Bug	In Progress	Normal	errors with ctobssim simulation for diffuse source with Healpix file	Knödlseder Jürgen	
□	1370	Bug	In Progress	Normal	ctools 00-08-00 'make check' errors and failed test macros	Knödlseder Jürgen	
□	1352	Bug	New	Normal	Running ctobssim in debug mode gives concurrent output of multiple threads		
□	1350	Change request	New	Normal	Separate parameter reading from file loading		
□	1320	Change request	New	Normal	Add ctlike warning in case that all parameters are fixed		
□	1298	Change request	New	Normal	Cube analysis tools should complain about missing ROI information.		
□	1290	Feature	Feedback	Normal	Optional TS calculation and spectral points with no "Prefactor"	Buehler Rolf	
□	1288	Action	In Progress	High	ctmodel doesn't work for GCTABackground3D	Knödlseder Jürgen	
□	1275	Action	New	Normal	Setup community writing tools for release 1.0.0 paper	Buehler Rolf	
□	1274	Action	New	Normal	Outline the plan of the release 1.0.0 paper	Buehler Rolf	
□	1273	Action	New	Normal	Performance validation for 1.0.0 release paper	Martin Pierrick	
□	1272	Action	New	Normal	Demonstrate how ctools & GammaLib can be used for Fermi analysis	Schulz Anneli	
□	1271	Action	New	Normal	Demonstrate how ctools & GammaLib can be used for HESS analysis	Mayer Michael	
□	1270	Feature	New	Normal	Write ctools & gammalib release 1.0.0 paper		
□	1263	Feature	In Progress	Normal	High level analysis scripts	Sanchez David	
□	1248	Bug	In Progress	High	CTA simulation of molecular cloud does not return expected result	Knödlseder Jürgen	
□	1074	Bug	Feedback	Urgent	Optimizer does not converge for parameters close to truth	Knödlseder Jürgen	
□	814	Change request	New	High	Sensitivity of fit to initial values		
□	625	Bug	New	High	Access of invalid matrix element in ctlike		
□	564	Change request	New	High	ctbin should read input FITS file data after the user enters all parameters		
□	108	Action	New	Normal	Determine exponential cut-off parameters as function of analysis radius and background model and compare to Aharonian et al. (2006) results.	Knödlseder Jürgen	
□	107	Action	New	Normal	Determine powerlaw parameters as function of analysis radius and background model and compare to Aharonian et al. (2006) results.	Knödlseder Jürgen	
□	105	Feature	New	Normal	Perform a scientific validation of 1DC release		

Agenda



Monday, 29 June:

- 14:00 – 16:00: Introduction, meeting goal, status of CTA developments & analysis (Jürgen)
- 16:00 – 16:15: IRF developments at IFAE (Tarek)
- 16:15 – 17:00: IRF discussions (all)
- 17:00 – 17:30: Status of HESS developments & analysis (Michael)
- 17:30 – 18:00: Status of VERITAS developments & analysis (Nathan)

Tuesday, 30 June:

- 9:00–18:00: Coding, Testing, Documenting

Wednesday, 1 July:

- 9:00–18:00: Coding, Testing, Documenting
- 20:00: Social dinner

Thursday, 2 July:

- 9:00–18:00: Coding, Testing, Documenting

Friday, 3 July:

- 9:00 – 10:00: Coding, Testing, Documenting
- 10:00 – 12:00: Meeting wrap up