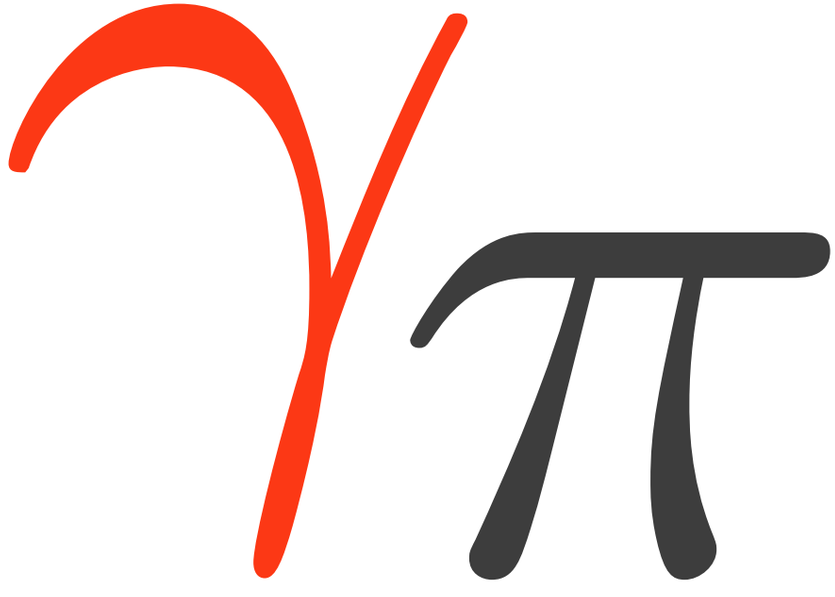
GAMMAPY

A PROTOTYPE FOR THE CTA SCIENCE TOOLS

Max-Planck-Institut für Kernphysik Heidelberg

Christoph Deil (MPIK Heidelberg)
for the Gammapy developers and the MPIK CTA group
CTA consortium meeting, Kashiwa, May 16, 2016

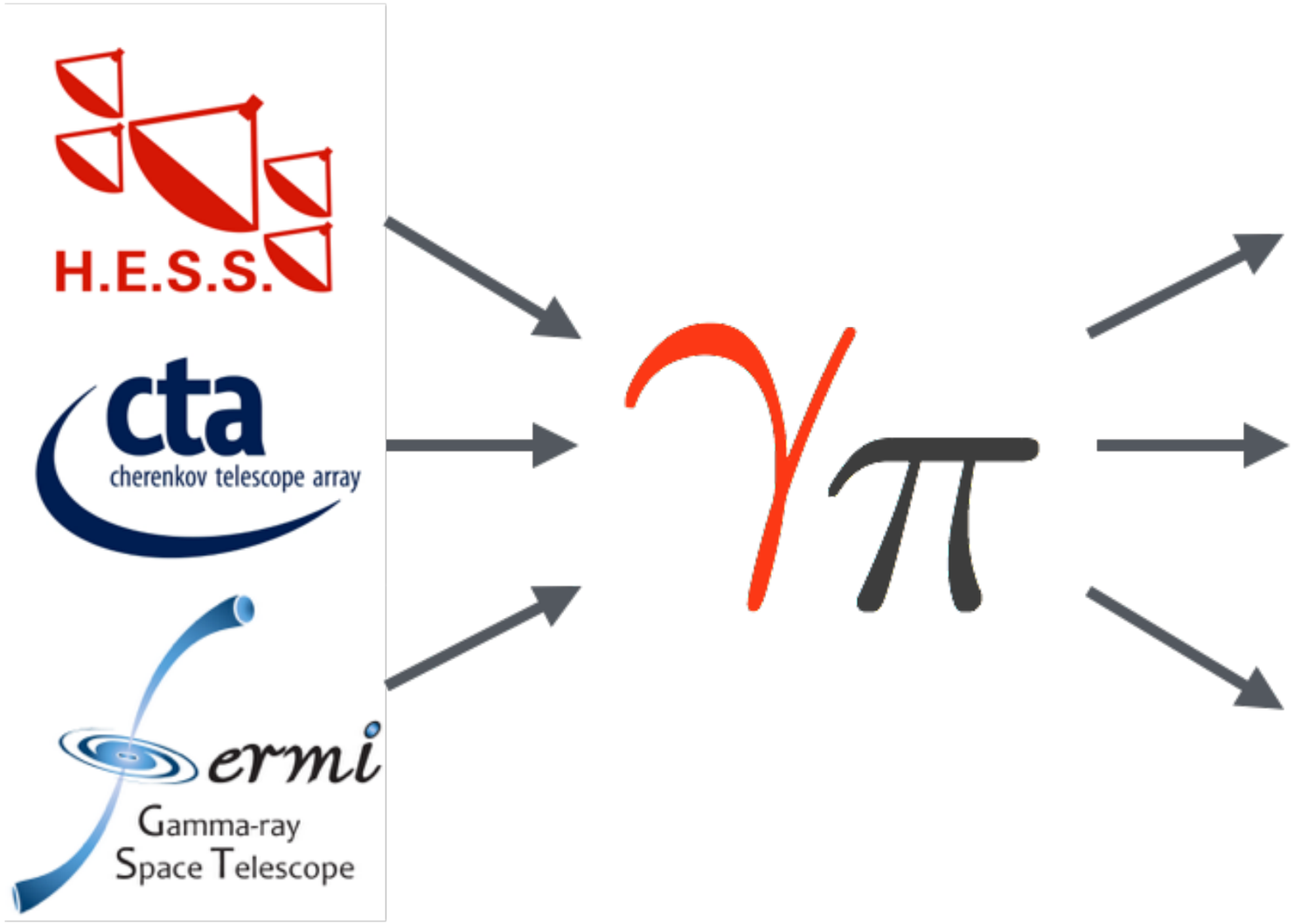
WHAT IS GAMMAPY?



A **Python** package for **gamma-ray** astronomy

- Code: https://github.com/gammapy/gammapy
- Docs: https://gammapy.readthedocs.io/
- Mailing list: http://groups.google.com/group/gammapy
- License: BSD-3 (same as Numpy, Scipy, Astropy, …)

GAMMAPY SCOPE — SCIENCE TOOLS FOR GAMMA-RAY ASTRO



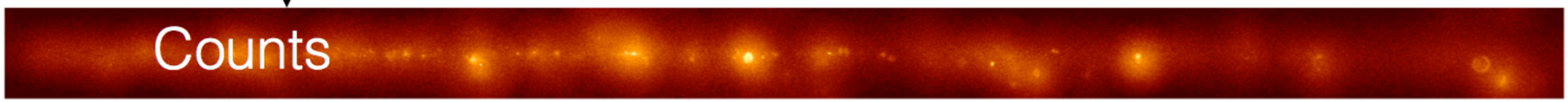
GAMMAPY APPLICATION EXAMPLE

With very little Python code, go from an event list to a simple Galactic plane survey source catalog.

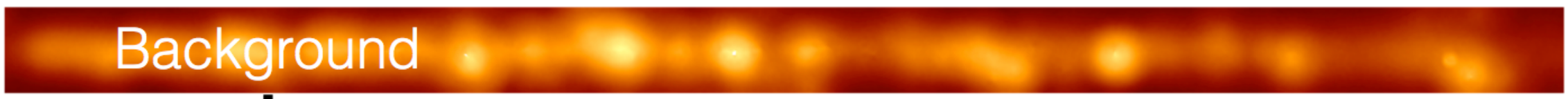
DL3

Input: Event list: | Energy | RA | DEC |

Coordinate transformations: ~ 1 minute

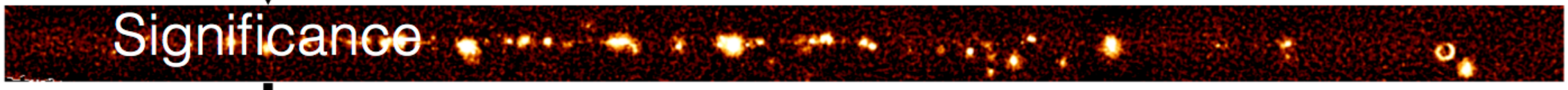


Iterative method: 2 minutes



DL4

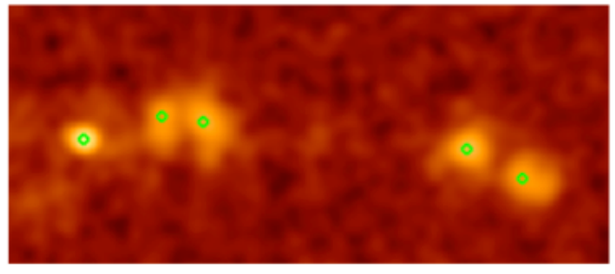
Very fast test statistic TS image computation: 5 minutes



Peak finder: ~ 1 minute

DL5

Source catalog: | Source Number | RA | DEC | ... |



WHAT IS GAMMAPY?

```python
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

- **Gammapy is a Python package** providing functions and classes to implement algorithms like the one shown here using Python scripts.
- Some (but not much yet) functionality has been wrapped in **command line tools** that are installed with Gammapy, e.g. in this case **gammapy-image-bin**.
- (This example is explained in detail in the backup slides.)

A **Python** package for
gamma-ray astronomy

General documentation

- About Gammapy
- Installation
- Getting Started
- Tutorials and Examples
- Data Formats
- References
- Developer documentation
- Changelog

The Gammapy toolbox

- Command line tools (`gammapy.scripts`)
- Astrophysical source and population models (`gammapy.astro`)
- Background estimation and modeling (`gammapy.background`)
- Catalog (`gammapy.catalog`)
- Cube Style Analysis (`gammapy.cube`)
- Data and observation handling (`gammapy.data`)
- Access datasets (`gammapy.datasets`)
- Source detection tools (`gammapy.detect`)
- Image processing and analysis tools (`gammapy.image`)
- Instrument response function (IRF) functionality (`gammapy.irf`)
- Morphology and PSF methods (`gammapy.morphology`)
- Regions (`gammapy.region`)
- Spectrum estimation and modeling (`gammapy.spectrum`)
- Statistics tools (`gammapy.stats`)
- Time handling and analysis (`gammapy.time`)
- Utility functions and classes (`gammapy.utils`)

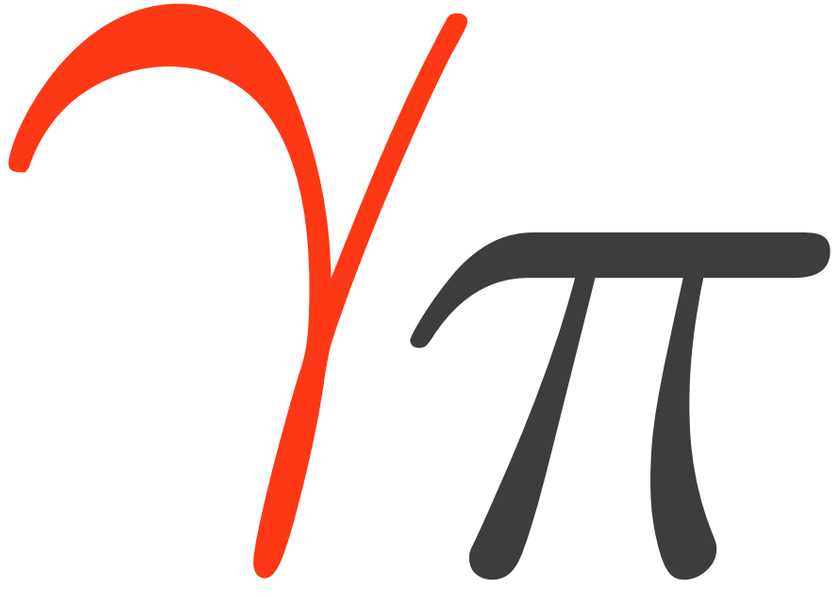
GAMMAPY FEATURES

- Classical analysis available (2D images and 1D spectra)
- Cube analysis work in progress (3D lon-lat-energy)
- Data and IRF handling (work in progress, see Gammapy presentation at recent IACT DL3 meeting)
- Background modeling
- Source detection

THIS PRESENTATION

- Gammapy is a very young project, with very high development activity.
- The API is not stable yet (partly because the IACT DL3 data model and formats are still work in progress).
- This presentation will focus on the Gammapy approach (architecture, implementation) and development.
- We will present an overview of features and applications to real HESS data and CTA simulations later.

GAMMAPY APPROACH



GAMMAPY APPROACH

GAMMAPY APPROACH

```python
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

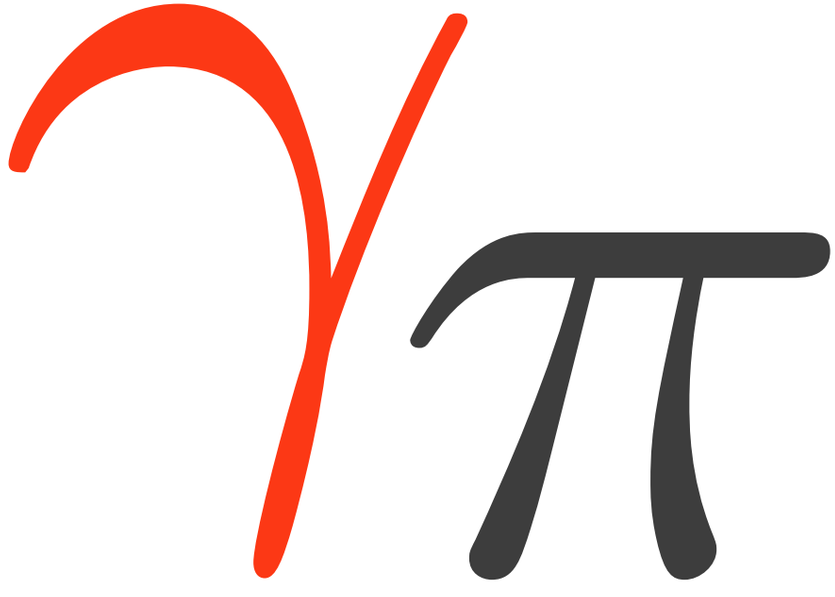
- In Gammapy (and Astropy), the data structures (e.g. EventList or SkyMap) are Python objects (native, not wrapped) that at the core consist of **Numpy arrays**.
- Algorithms are implemented in Python using Numpy array expressions and by calling functions and methods from Numpy, Scipy, Astropy.
- These other packages often call into powerful, fast, well-tested numerics and astro libraries.

GAMMAPY APPROACH

- Gammapy approach:
  - Use Python, build on Numpy, Scipy, Astropy
- Consequences:
  - Codebase is focused on gamma-ray astronomy, single-language, high-level, small, simple.
  - Objects consist of Numpy array and Astropy objects (e.g. SkyCoord, Time, WCS, Table) and are thus compatible with other scientific Python and astro packages.
  - Nice to use and extend in Python (even efficient for algorithms that operate on events or pixels)

GAMMAPY DEVELOPMENT

and deployment



GAMMAPY DEVELOPMENT OVERVIEW

Gammapy is at the moment set up like Astropy and most other open-source Python packages …

GitHub Version control, issue tracker,
contributions via pull requests & code review

Tests automatically run on Linux & Mac
on each pull request and master branch
Travis CI

pytest Python testing framework
(makes it easy to write and run tests)

Python documentation generator
API and narrative docs pages
cross-linked, full-text search
SPHINX

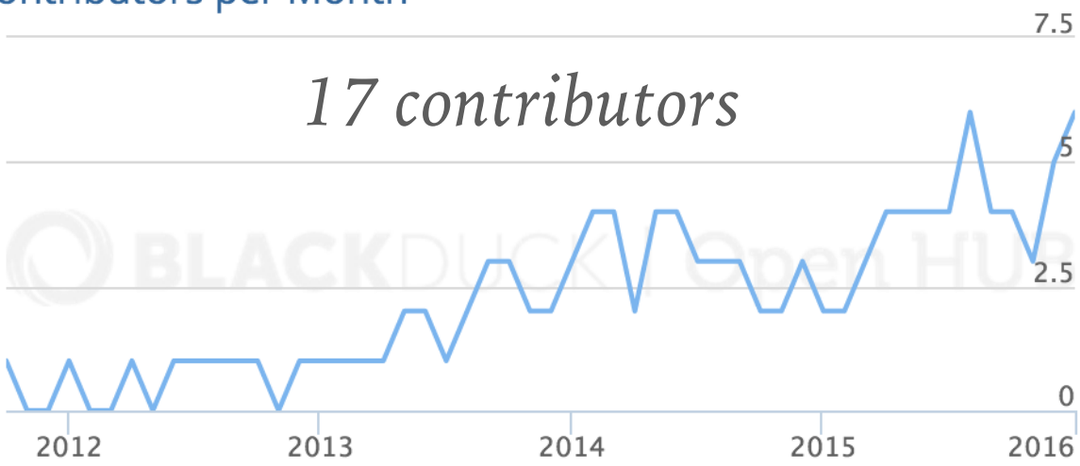
Anaconda Binary cross-platform package manager.
Install Gammapy and all dependencies on any
Linux & Mac box in $HOME in 10 min.

GAMMAPY DEVELOPMENT

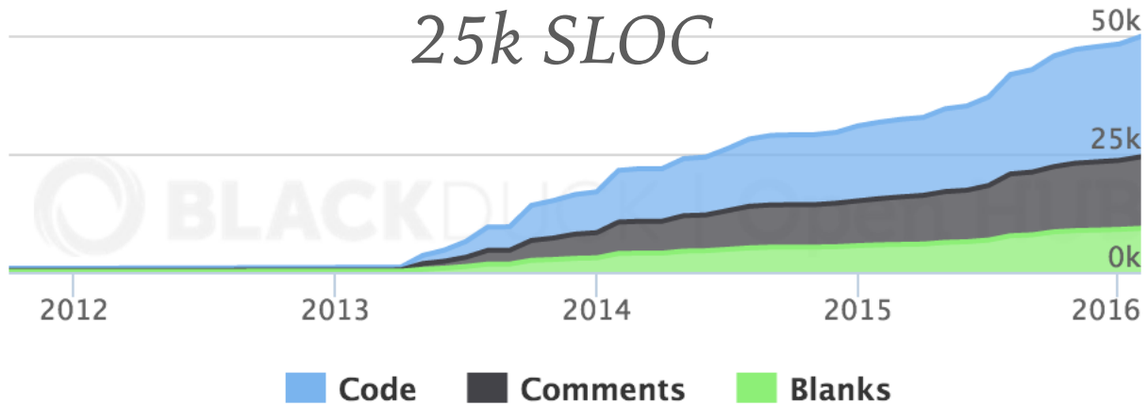
- Young project, very active development (see below)
- Last stable release: Gammapy 0.4 in April 2016
- First coding sprint June 6-10 at MPIK, Heidelberg
- 1.0 release planned for summer, paper in the fall.

https://openhub.net/p/gammapy

Contributors per Month



Lines of Code



PYTEST FOR CTA ST?

- pytest is great:
  - Nice to write tests (plain assertions, test fixtures, parametrisation, markers, …)
  - Good reporting on the console as well as to continuous integration systems.
  - Flexible test runner (test collection and selection)
  - Many plugins available (e.g. for coverage reports)
- We recommend pytest for the CTA ST
  (actually any project with a Python interface)

GAMMAPY DISTRIBUTION AND INSTALLATION

- Gammapy works with Python 2 or 3, on Mac, Linux, Windows.
  (Sherpa doesn't work with Python 3 and on Windows yet.)
- Source distribution can be installed with pip:

  pip install gammapy
- Binary distribution can be installed with conda:

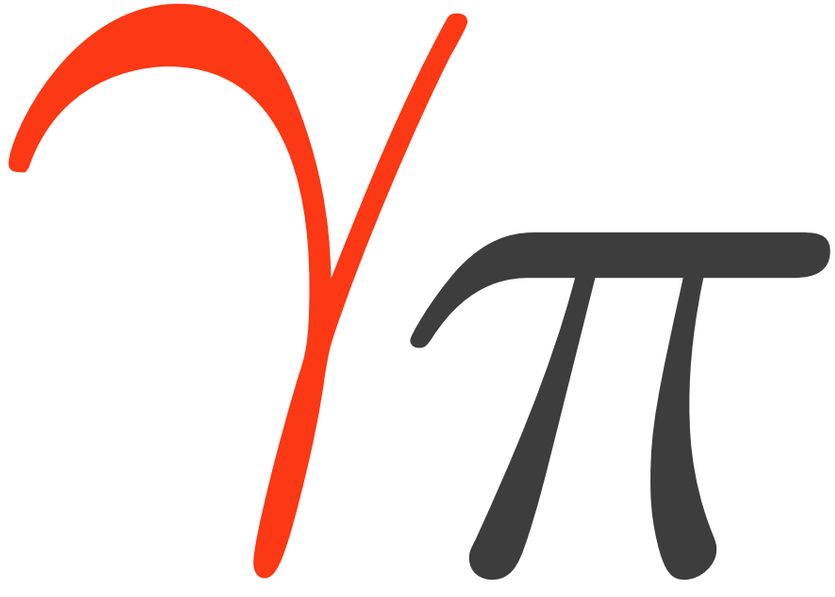
  conda install -c openastronomy gammapy
- Developers use setup.py
  (setuptools, the standard for Python packages):

  git clone https://github.com/gammapy/gammapy.git
  cd gammapy
  python setup.py install
- No other binary packages (Linux, Mac) available yet.

CONDA FOR CTA ST?

- Conda is great:
  - A cross-platform package manager (Linux, Mac, Windows)
  - Not just for Python (e.g. QT and R are distributed this way)
  - Easy to set up, to install and update packages, works in $HOME, multiple versions possible.
  - Simple to build and distribute packages (compared to other package managers)
  - An extensible solution (compared to distributing binaries without a package manager, like CIAO or the Fermi ST do).
- We recommend conda for the CTA ST binary distribution (as one first solution that works for everyone on any platform)

SUMMARY
OUTLOOK



SUMMARY

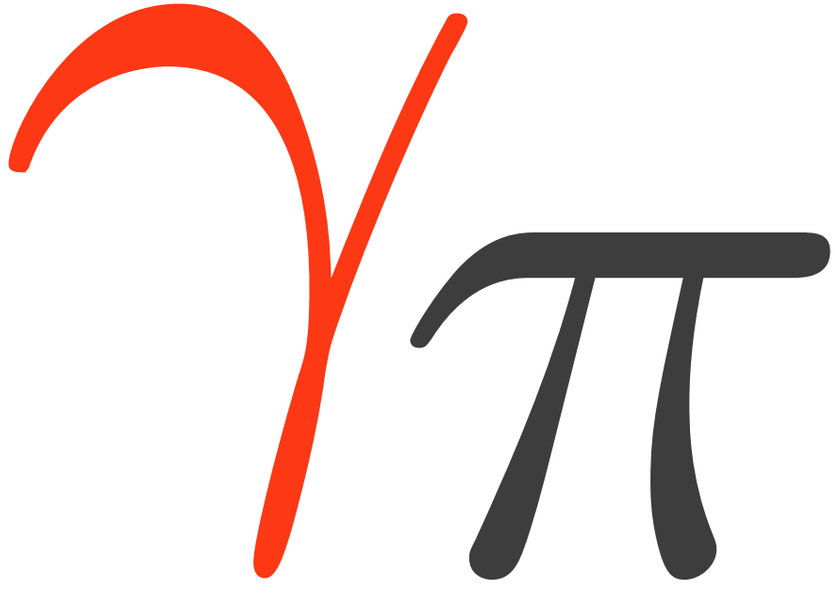
- Gammapy is a Python package for gamma-ray astronomy
- A prototype for the CTA science tools (a document with a more detailed description will be circulated soon).
- We think the following aspects being prototyped in Gammapy are worth considering for the CTA project for the ST:
  - Approach
    - — Python, Numpy, Scipy, Astropy
    - — Focused on gamma-ray astronomy
    - — Algorithms in Python
  - Testing and deployment: pytest and conda are excellent.

OUTLOOK

- Gammapy development activity is increasing.
- First coding sprint June 6-10, 2016 at MPIK, Heidelberg
- Focus in the coming months will be on
  - updating open IACT DL3 spec and Gammapy code
  - classical and cube IACT analysis methods
  - CTA response functions and simulation capabilities
  - improving test coverage and documentation
- 1.0 release planned for summer, paper in the fall.

BACKUP SLIDES

HOW GAMMAPY WORKS



Behind the scenes …

HOW GAMMAPY WORKS (SLIDE 1 OF 6)

```python
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

- Let's use this code example to explain how Gammapy works.
- Gammapy is a **Python package** providing functions and classes to implement algorithms like the one shown here using Python scripts.
- Some (but not much yet) functionality has been wrapped in command line tools that are installed with Gammapy, e.g. in this case **gammapy-image-bin**.

HOW GAMMAPY WORKS (SLIDE 2 OF 6)

```python
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

- In Gammapy (and Astropy), the data structures (e.g. EventList or SkyMap) are Python objects (native, not wrapped) that at the core consist of **Numpy arrays**.
- Algorithms are implemented in Python using Numpy array expressions and by calling functions and methods from Numpy, Scipy, Astropy.
- These other packages often call into powerful, fast, well-tested numerics and astro libraries.

HOW GAMMAPY WORKS (SLIDE 3 OF 6)

```
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

- gammapy.data.EventList is an astropy.table.Table subclass.
- Table is a collection of Column objects with a lot of convenient methods.
- A Column object usually a 1-dim Numpy array (although Quantity, SkyCoord, Time objects can be columns as well).
- Row-wise access less efficient because of the column-first memory layout.
- This turns out to be the right choice, most of the time tables are accessed by column.

HOW GAMMAPY WORKS (SLIDE 4 OF 6)

```python
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

- gammapy.image.SkyMap has a data attribute (a 2-dim Numpy array) and a wcs attribute (a (astropy.wcs.WCS object).
- Scientific Python packages (Numpy, Scipy, scikit-image, Astropy, ...) all operate on Numpy arrays.
- All their functionality is available efficiently (no copy) for Gammapy objects.

HOW GAMMAPY WORKS (SLIDE 5 OF 6)

```
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

- EventList.read loads the data from a FITS file using astropy.io.fits and creates an EventList object.
- SkyMap.empty creates a zero-filled SkyMap object with a given WCS specification.
- This is a general pattern in Gammapy:
  - In-memory objects aren't coupled to the FITS serialisation format
  - The __init__ constructor takes inputs directly (often Numpy arrays or Astropy objects).
  - A classmethod called read is used to construct objects from FITS files.

HOW GAMMAPY WORKS (SLIDE 6 OF 6)

```
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyMap

events = EventList.read('events.fits')
image = SkyMap.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN'
)
image.fill(events)
image.write('counts.fits')
```

- The SkyMap.fill method implements the algorithm:
  - Access “RA” and “DEC” columns from the event list.
  - If image WCS is GALACTIC, use SkyCoord to transform.
  - Use WCS to transform event to pixel coordinates.
  - Use numpy.histogramdd to fill the counts histogram.
- This executed efficiently because all steps use Numpy arrays.