cherenkov telescope array

INFN PADOVA
Istituto Nazionale di Fisica Nucleare
Sezione di Padova

CTA Software and dependencies

Rubén López-Coto, INFN Padova

University of Rijeka, 16/02/21

Fellini
Fellowship for Innovation at INFN

Funded by H2020 Marie Sklodowska Curie FELLINI - Grant 754496

Analysis Software

- The **low level** analysis software within the CTA philosophy falls within the so called Data Preparation and Preservation System (**DPPS**)
- **High level** analysis falls within the Science User Support (**SUSS**)

Previously...

Data management

Now:

pipelines, archive, etc.

computing resources, etc

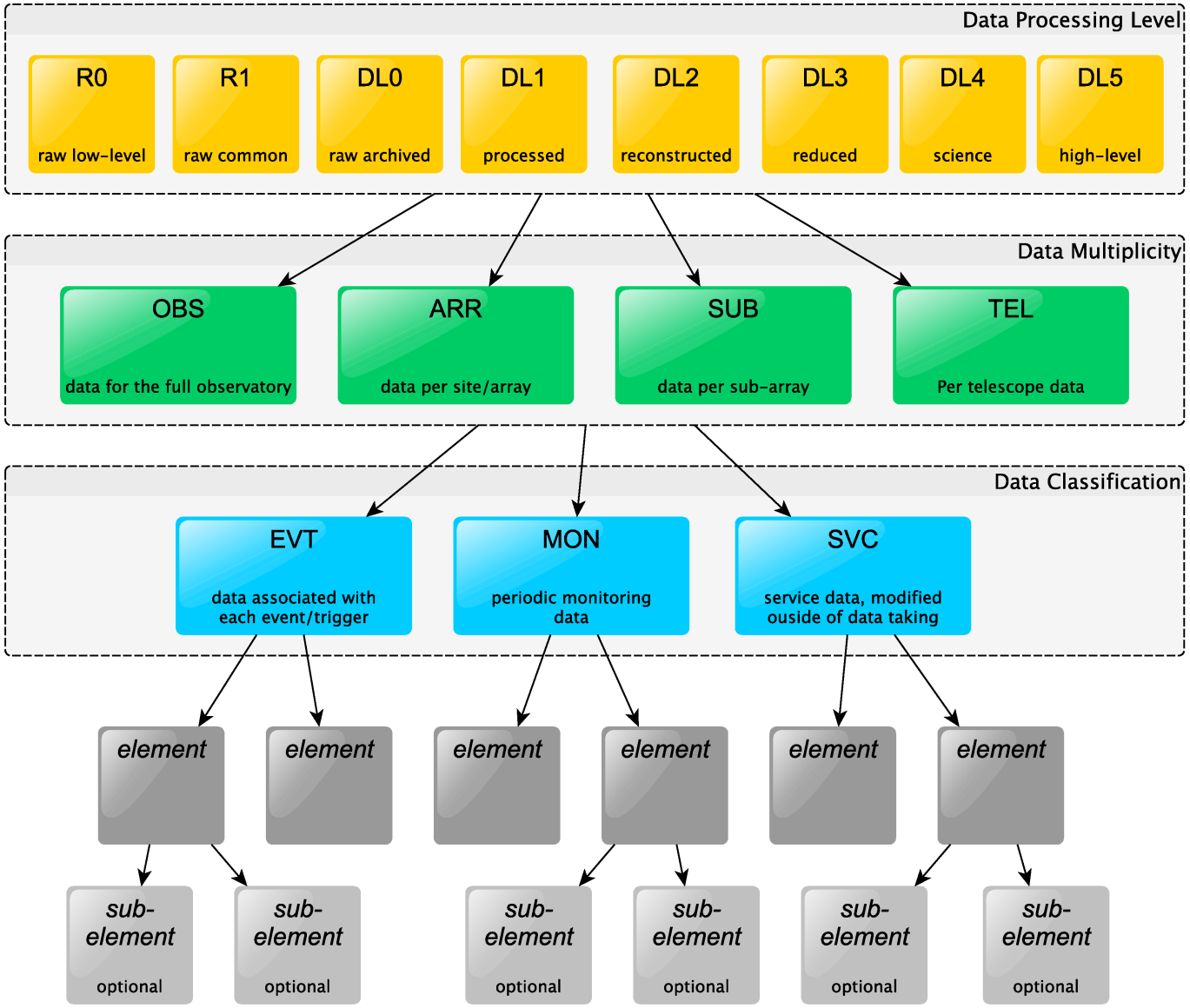
Data Processing and Preservation

Operations Support

Science User Support

science tools, user gateway etc.

Data Level definition



Data Levels: amount of processing

MCs are somewhere here right now

Pipeline starts here
Will need to eventually produce data here to be compatible with "real" CTA data

Reconstructed Events
(many reconstructions and parameters, no more telescopes)

Science Data:
Classified Events (final reconstruction), IRFs

R0 *(raw low-level)* camera data transmitted from telescope to central servers. R0 content and format is internal to each camera and is specified and coordinated between individual camera teams.

R1 *(raw common)* data output by an individual camera functional unit to the camera DAQ functional unit. This is the first level of data seen by the ACTL system and is therefore as common as possible between all cameras/hardware. Exceptionally, some R1 data may be stored for engineering purposes.

DL0 *(raw archived)* all archival data from the data acquisition hardware/software. This is the first level of data that are stored in the bulk archive. This includes both camera event data and technical data from other subsystems, such as non-camera devices or software.

DL1 *(processed)* processed DL0 data that may still include some TEL data and parameters derived from them. For example this includes calibrated image charge, Hillas parameters, and a usable telescope pattern. This is only optionally stored in the archive.

DL2 *(reconstructed)* reconstructed shower parameters such as energy, direction, particle ID, and related signal discrimination parameters. At this point, no TEL information is stored. For each event this information may be repeated for multiple reconstruction and discrimination methods. This is only optionally stored in the archive. At this point, telescope-wise info is generally dropped.

DL3 *(reduced)* Sets of selected (e.g. gamma-ray candidates, electron candidates, selected hadron candidates, etc.) events with a single final set of reconstruction and discrimination parameters, along with associated instrumental response characterizations and any technical data needed for science analysis.

DL4 *(science)* binned data products like spectra, sky maps, or light curves, along with associated data (source models, fit results, etc).

DL5 *(high-level)* high-level or “legacy” observatory data, such as CTA survey sky maps or the CTA source catalog.

high-level functionality

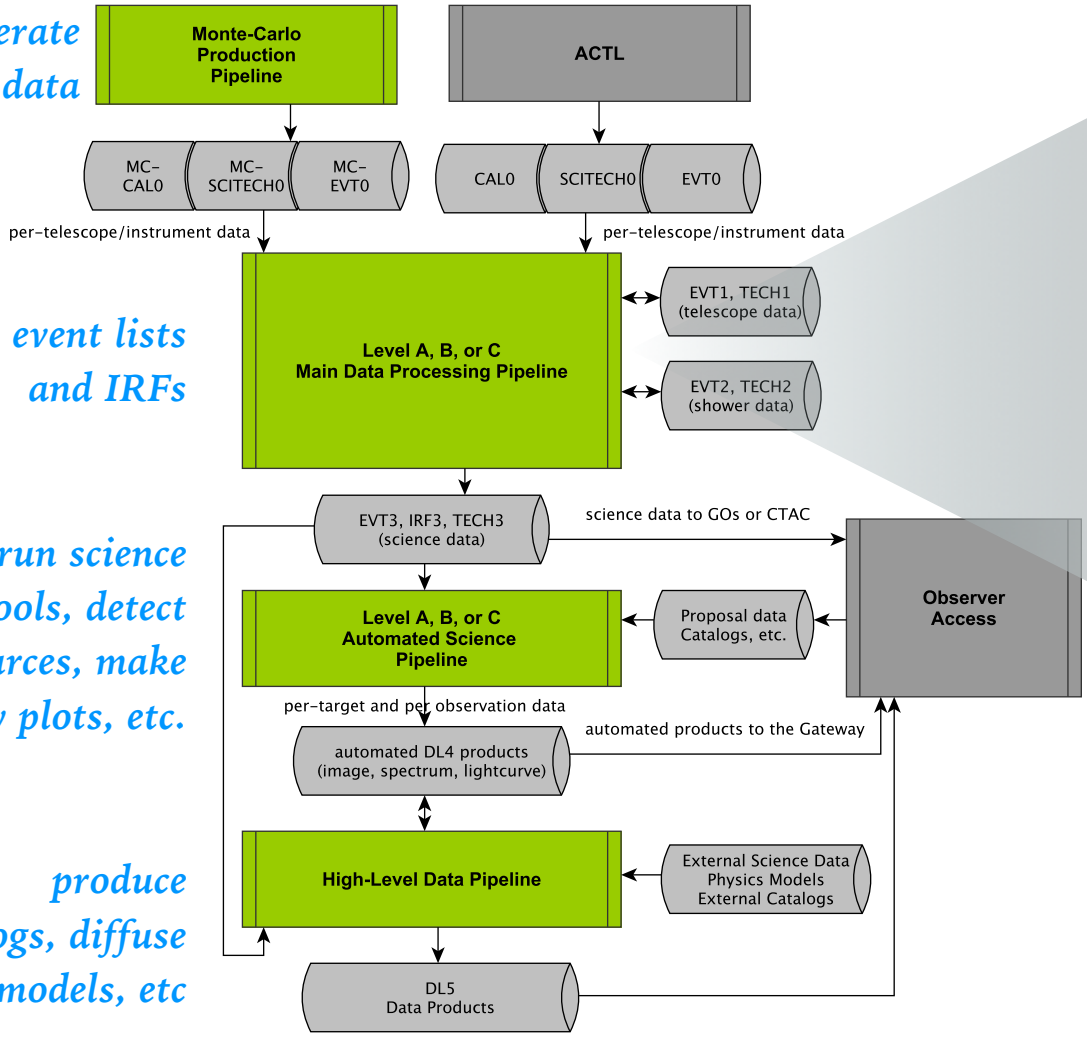
source: TDR, names a bit out of date

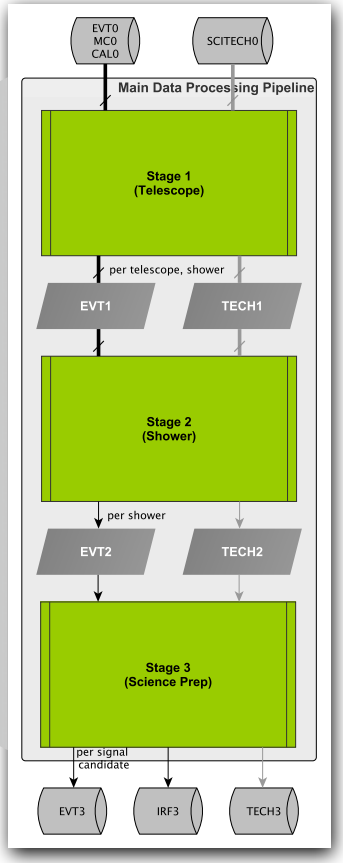
generate MC data

make event lists and IRFs

run science tools, detect sources, make sky plots, etc.

produce catalogs, diffuse models, etc



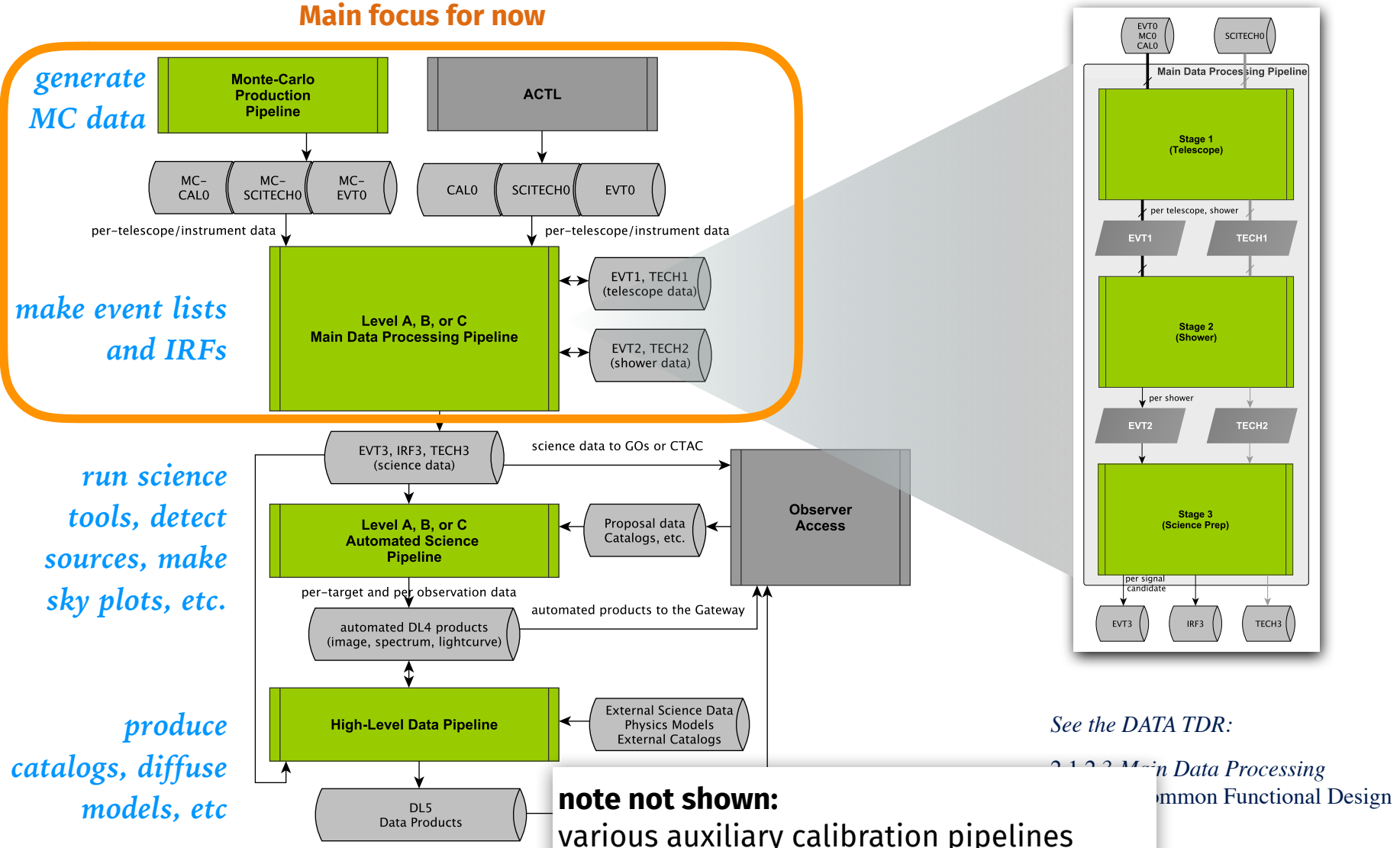


See the DATA TDR:

2.1.2.3 *Main Data Processing Pipeline* Common Functional Design

high-level functionality

source: TDR, names a bit out of date

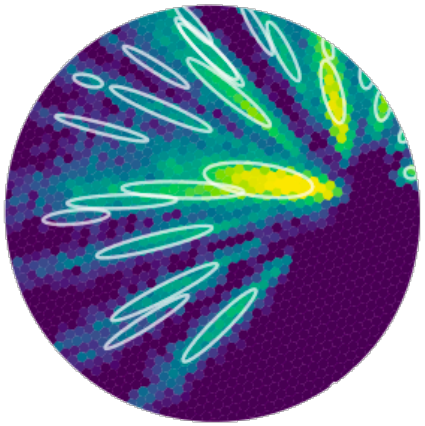


See the DATA TDR:

2.1.2.3 Main Data Processing

Common Functional Design

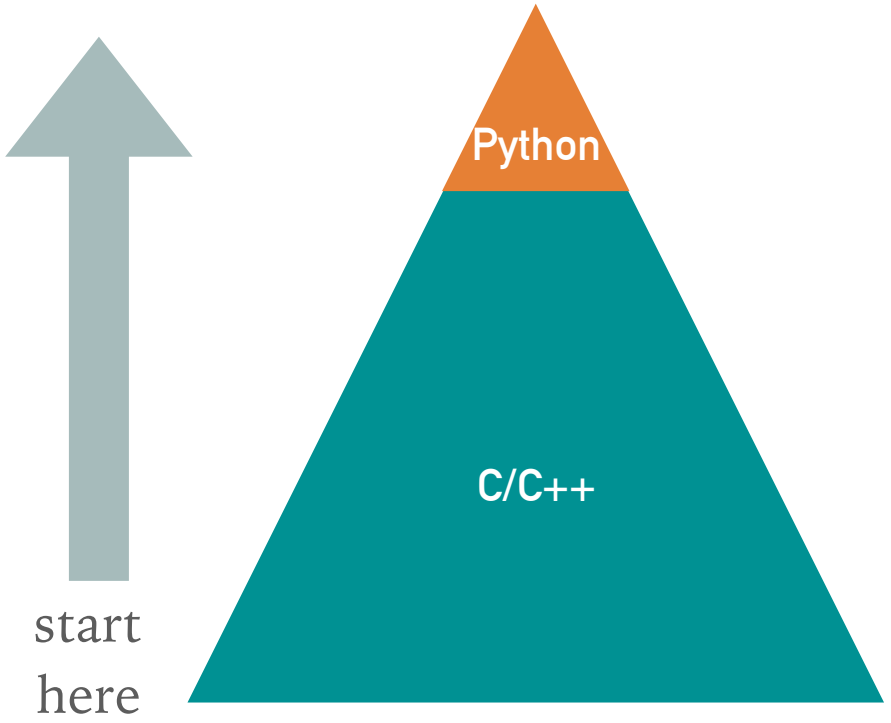
note not shown:
various auxiliary calibration pipelines
(run separately from main data processing)



ctapipe

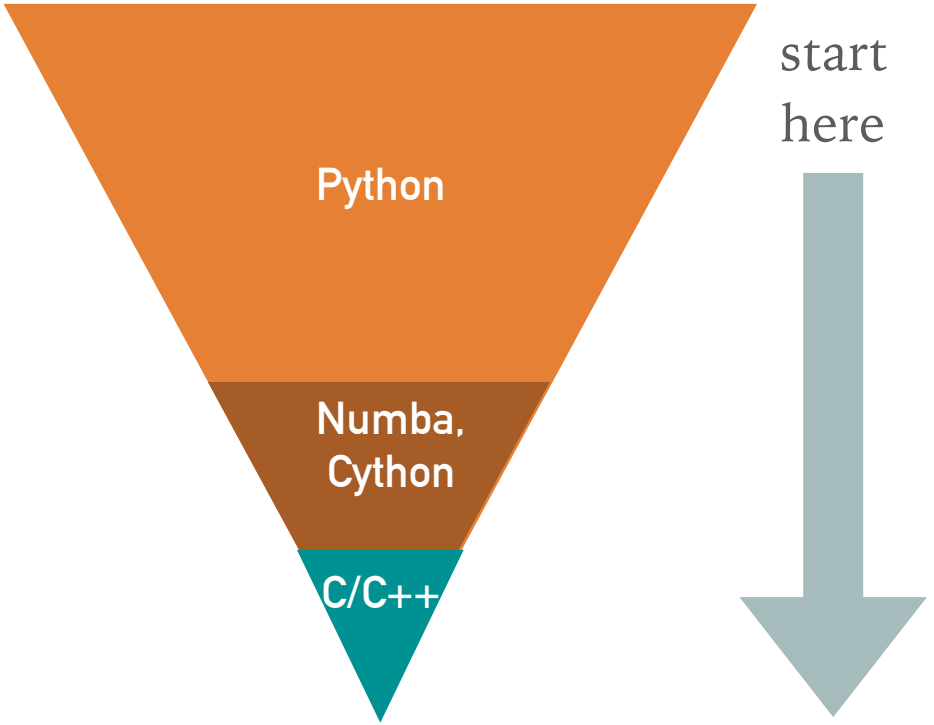
Building a Framework

Bottom-Up approach



Most current frameworks did it this way (if they use python at all)

Top-Down approach



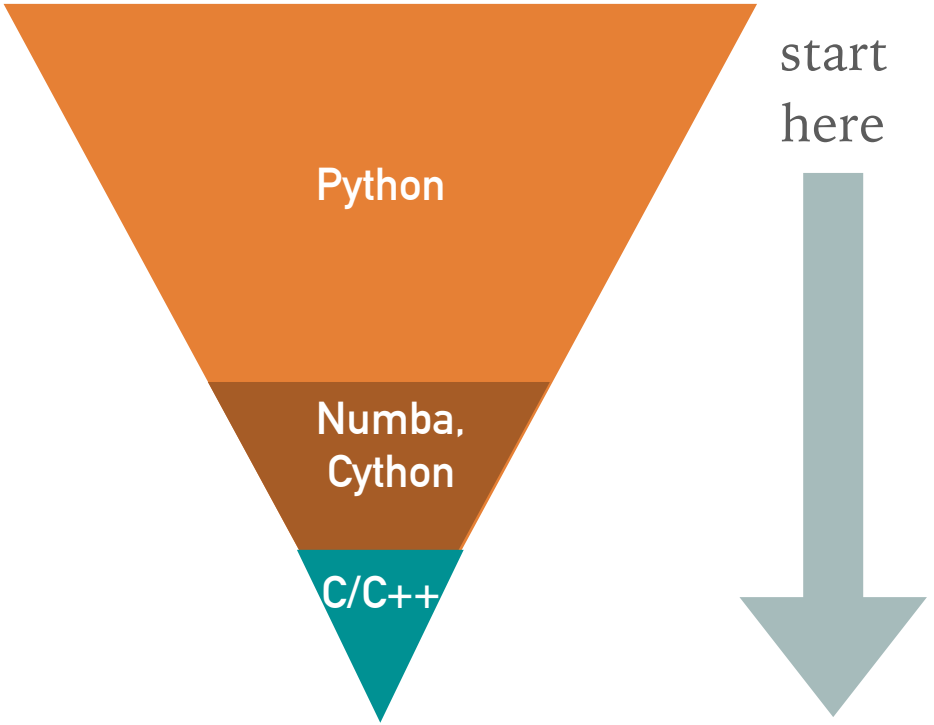
Our approach: start early with python and high-level API

Building a Framework

Bottom-Up approach

Most current frameworks did it this way (if they use python at all)

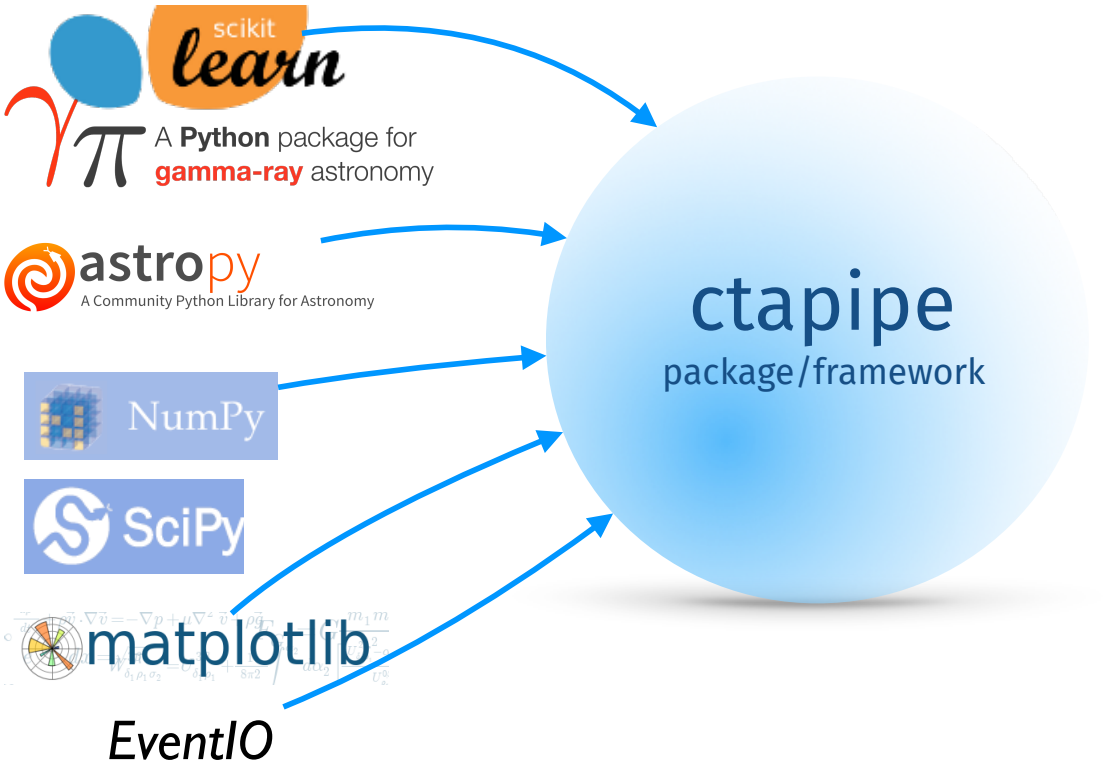
Top-Down approach



Our approach: start early with python and high-level API

common “core” package

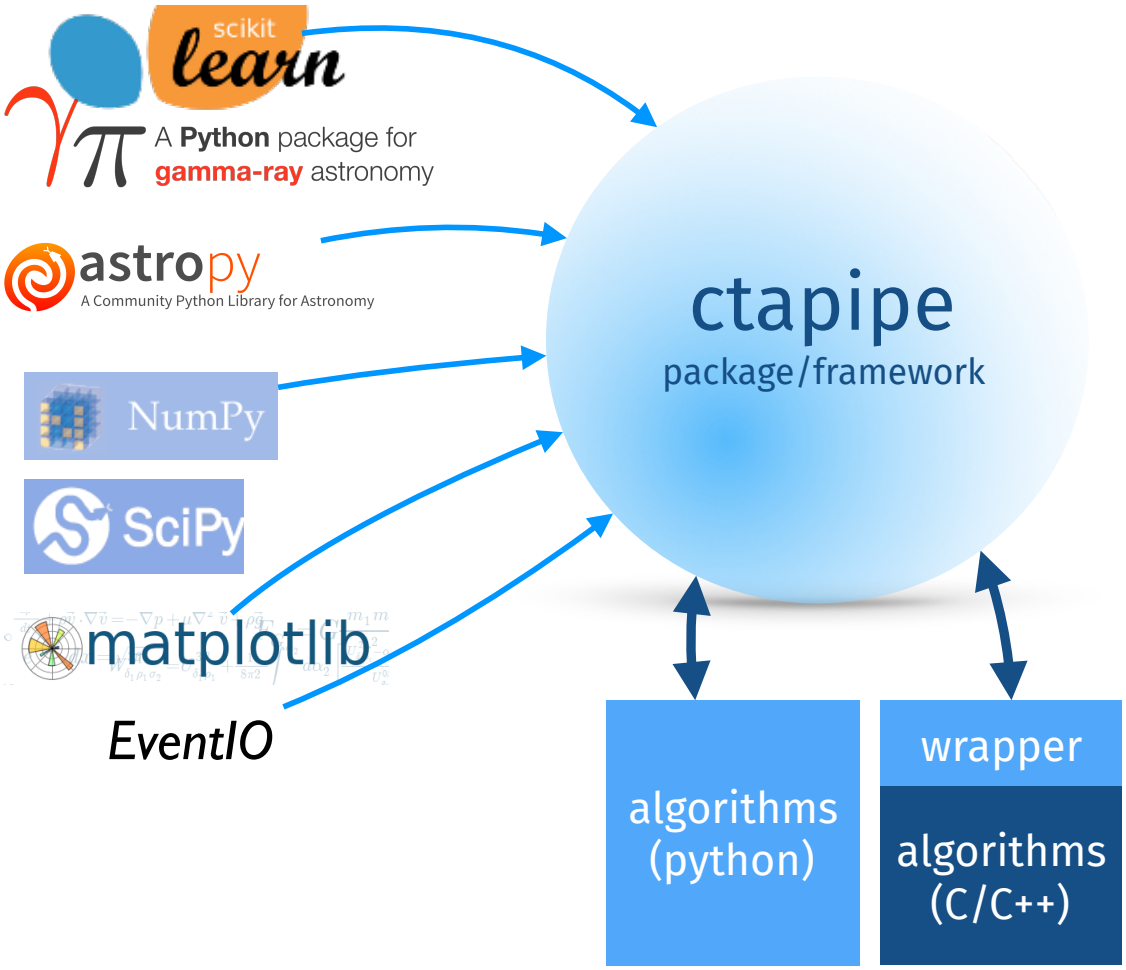
ctapipe will be **glue** between various components.
Provides common APIs and user interfaces
packaging, etc.



pytest

common “core” package

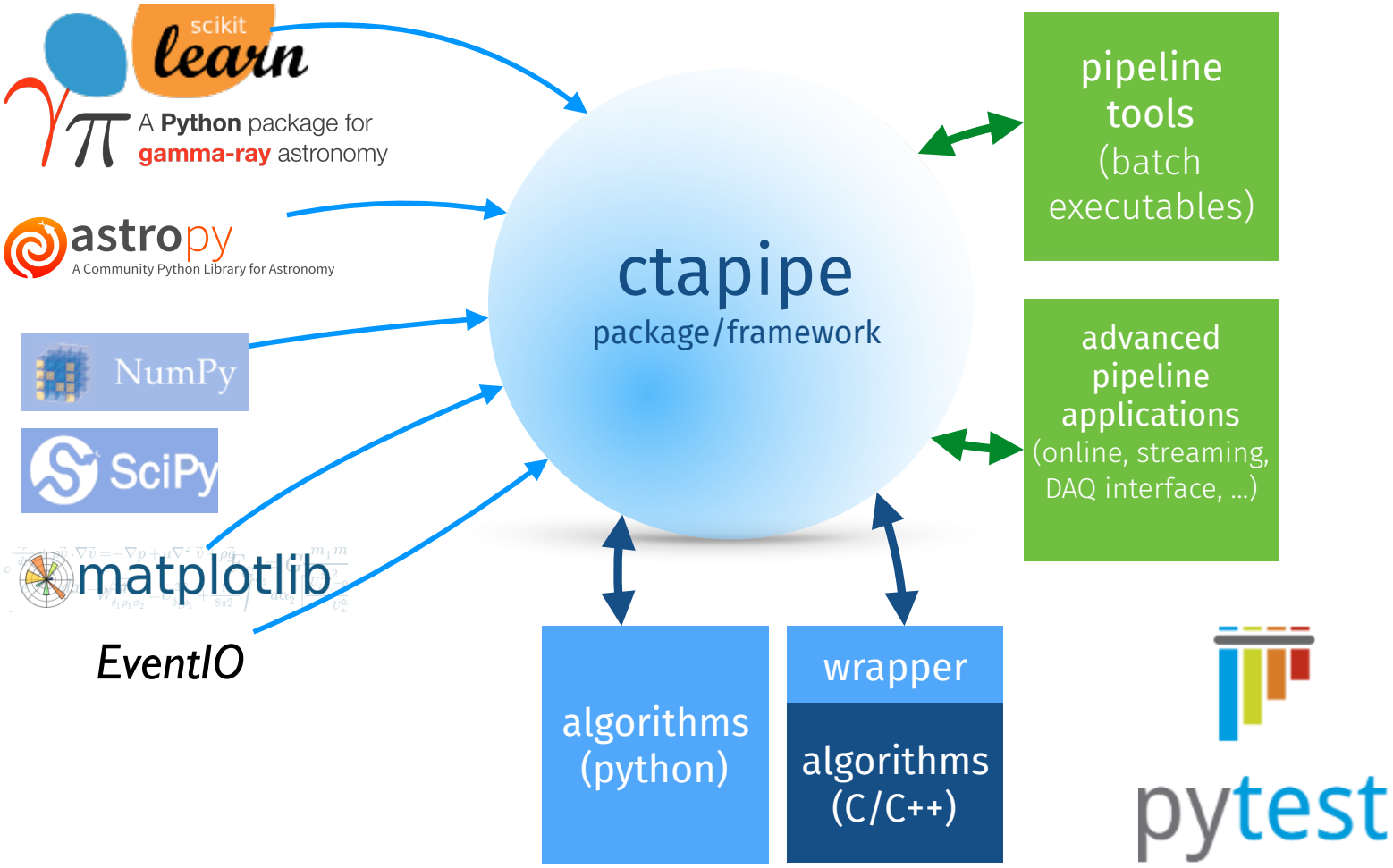
ctapipe will be **glue** between various components.
Provides common APIs and user interfaces
packaging, etc.



pytest

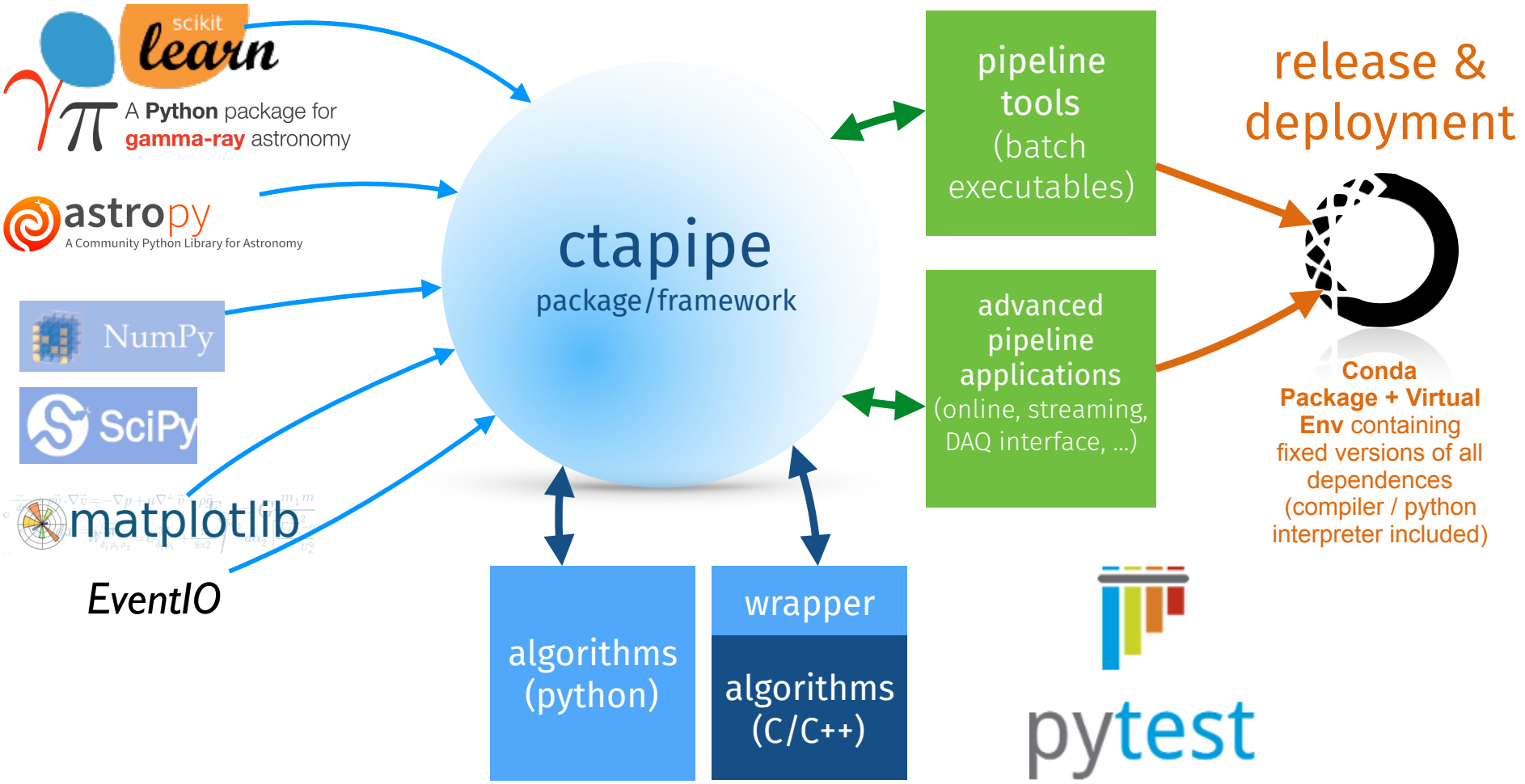
common “core” package

ctapipe will be **glue** between various components.
Provides common APIs and user interfaces
packaging, etc.



common “core” package

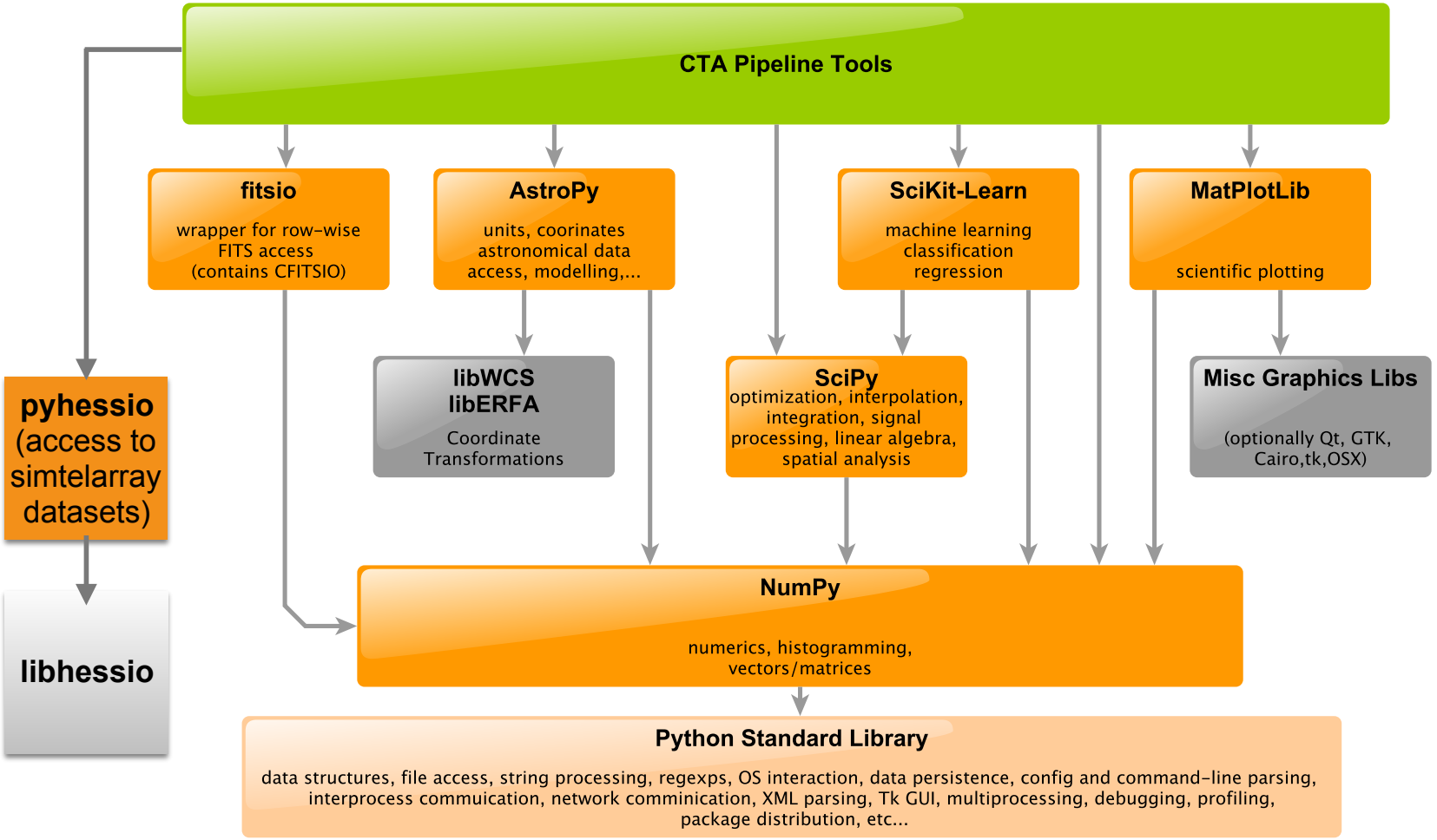
ctapipe will be **glue** between various components.
Provides common APIs and user interfaces
packaging, etc.



Design

- ctapipe is the “CTA Low-level Data Processing Pipeline Framework Prototype”
  - it means that it is a library for building pipelines, **not a pipeline**
- Libraries to perform the DL0 → DL3 analysis
- ctapipe is built upon the Scientific Python Stack, core dependencies are:
  - numpy
  - scipy
  - matplotlib
  - astropy
  - …

Dependencies



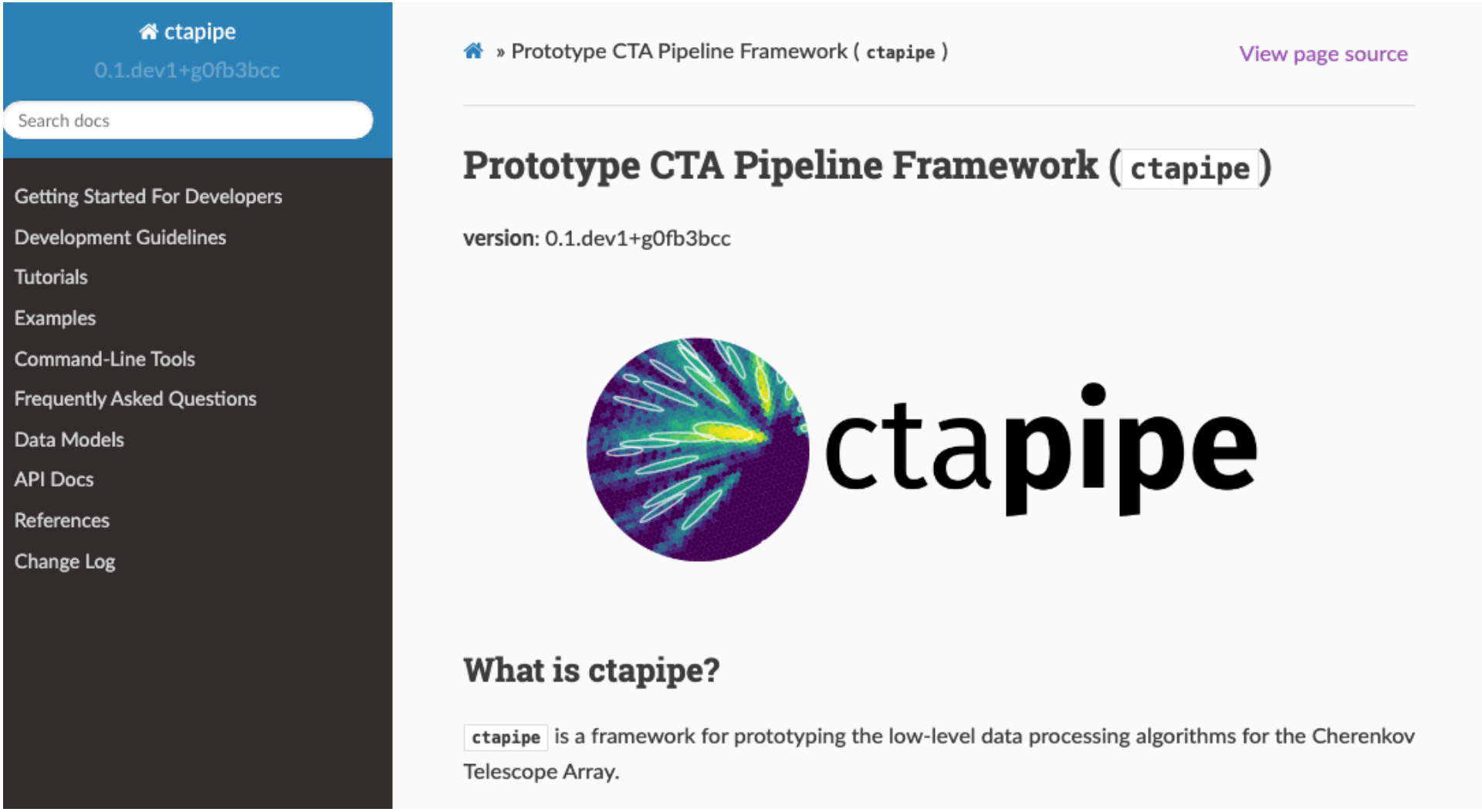
Developement

- ctapipe is developed as Open Source Software (Currently under MIT License)
  - https://github.com/cta-observatory/ctapipe
- We use the "Github-Workflow":
  - Few people (core developers) have write access to the main repository
  - Contributors fork the main repository and work on branches
  - Pull Requests are merged after Code Review and automatic execution of the test suite
- Early development stage ⇒ backwards-incompatible API changes might and will happen

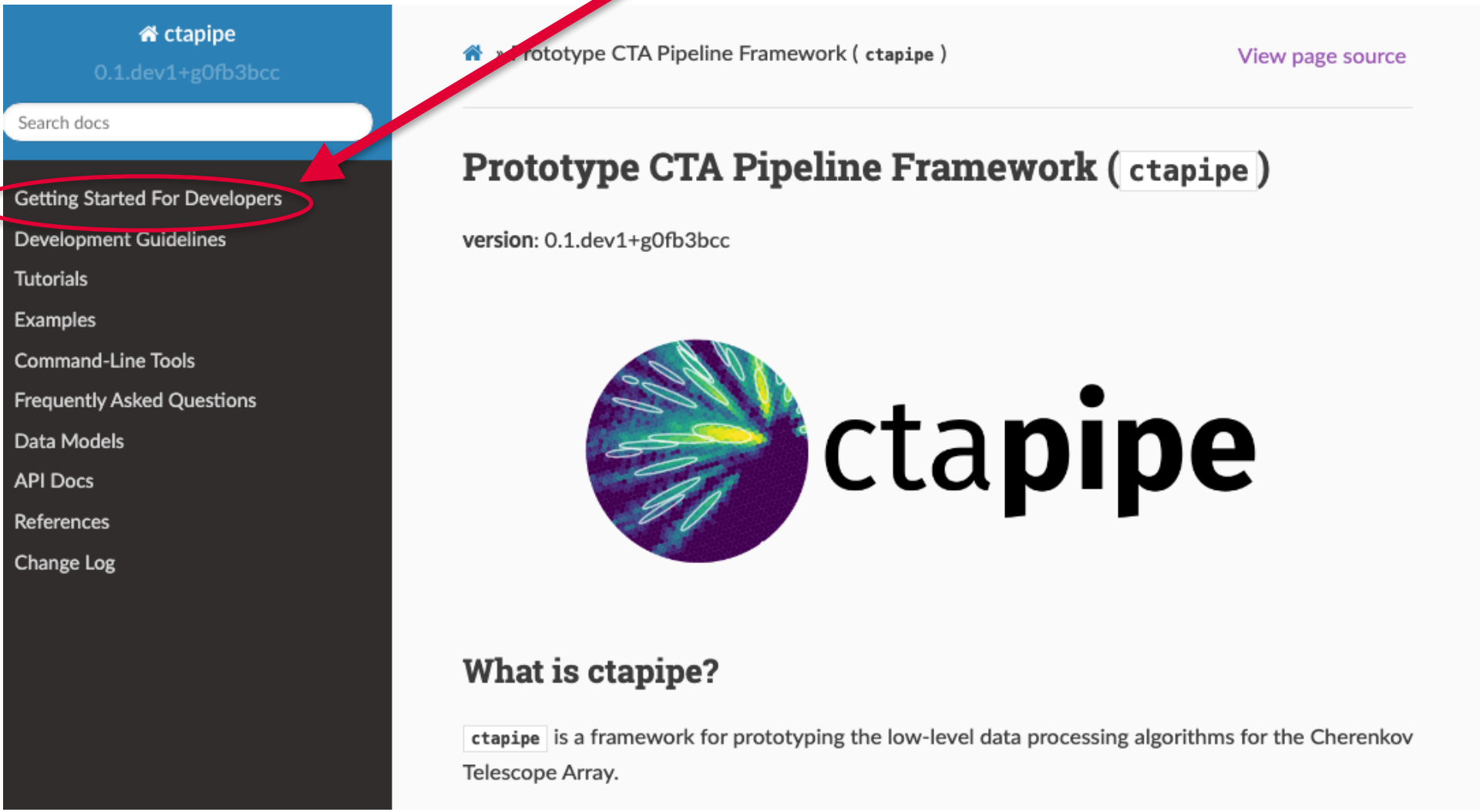
What's there

- Reading simtel simulation files
- Simple calibration, cleaning and feature extraction functions
- Camera and Array plotting
- Coordinate frames and transformations
- Stereo-reconstruction using line intersections

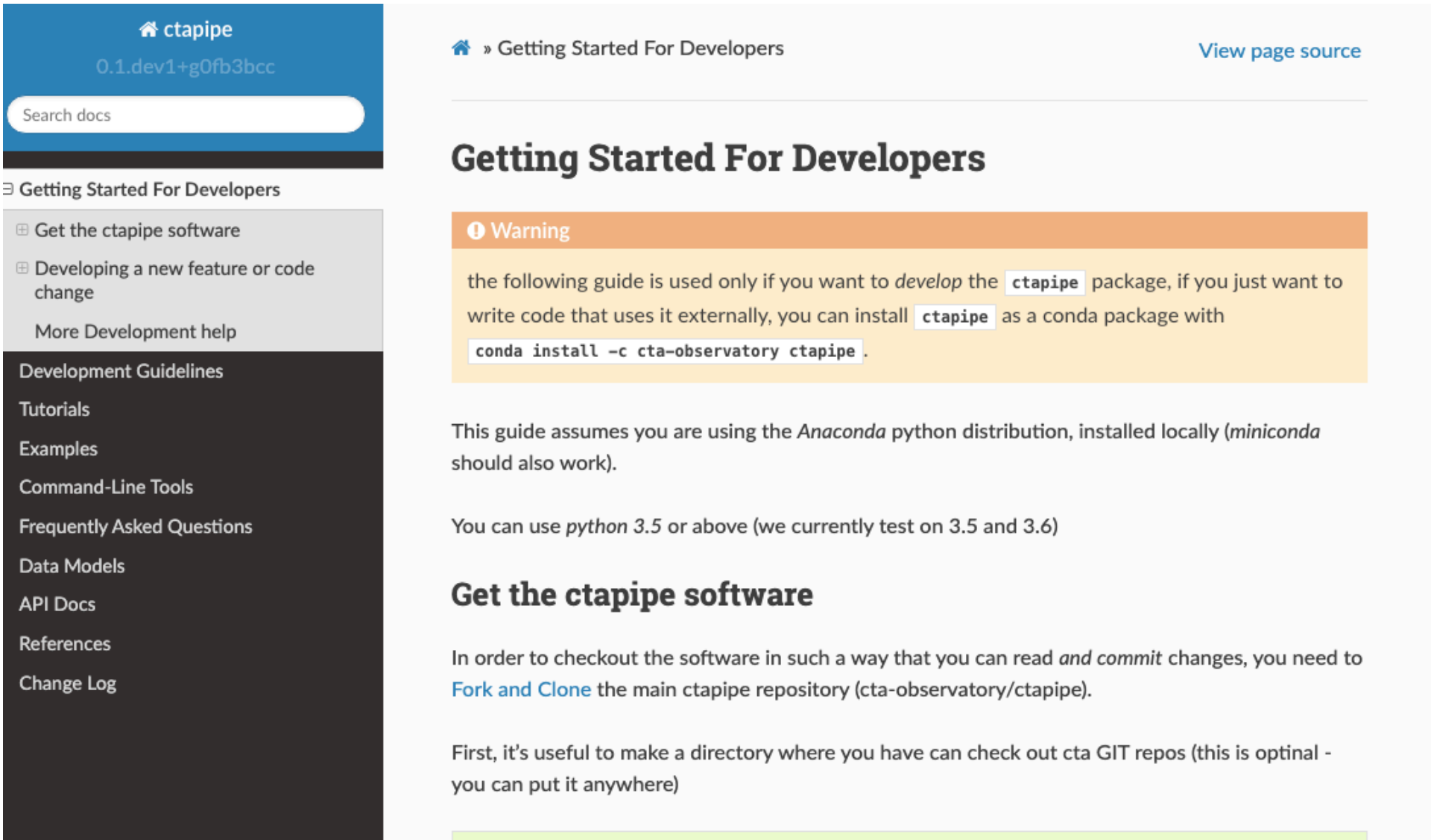
For more info... Read the docs



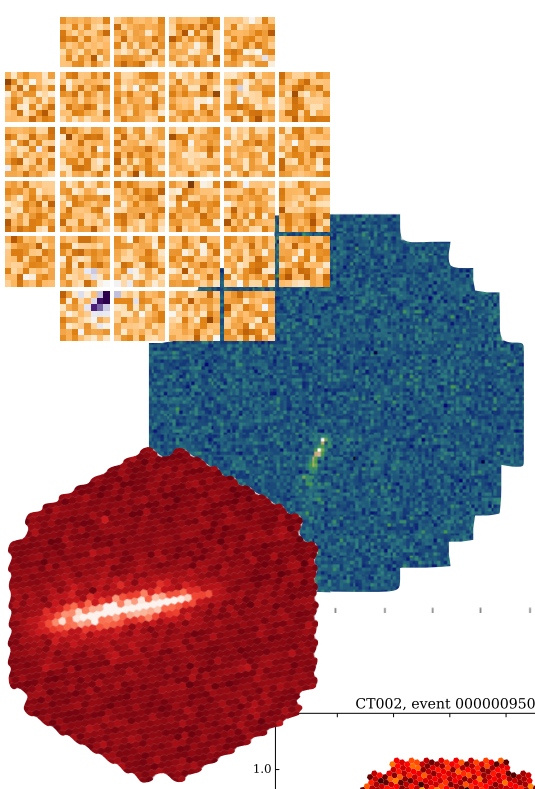
For more info... Read the docs

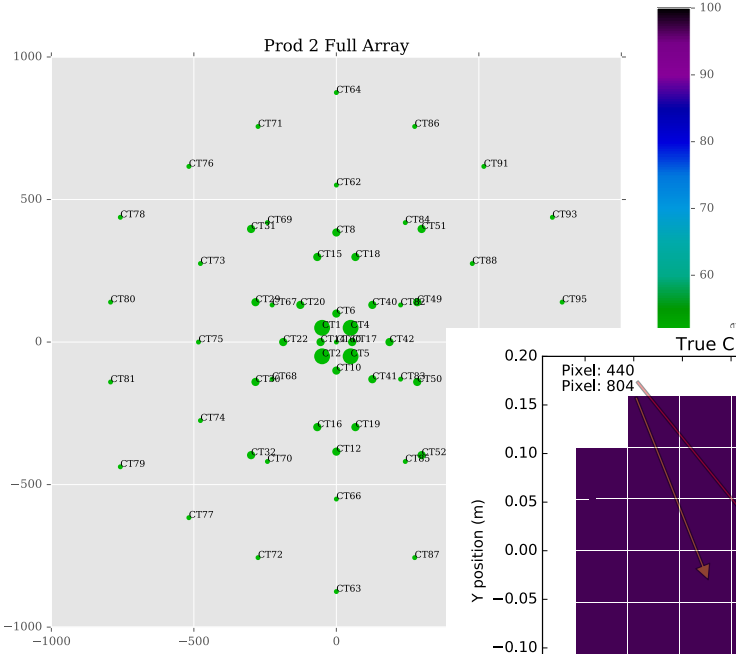


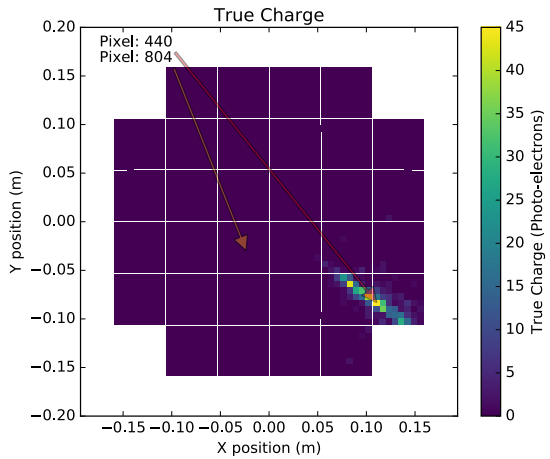
For more info... Read the docs

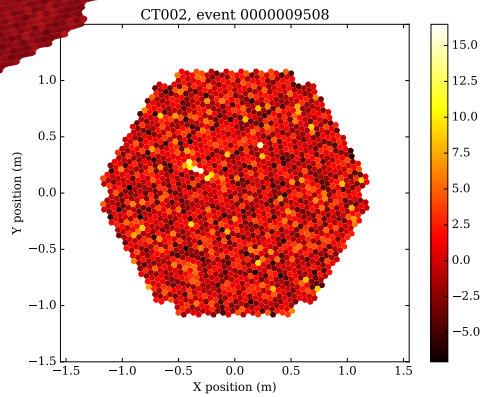


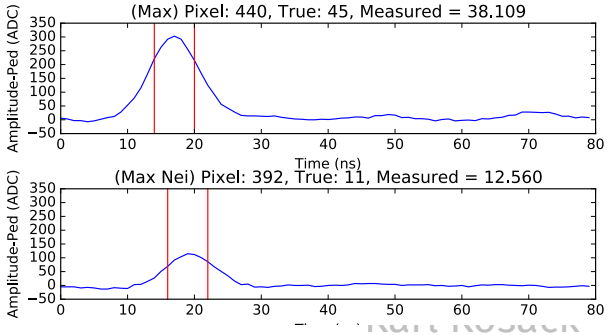
Visualization

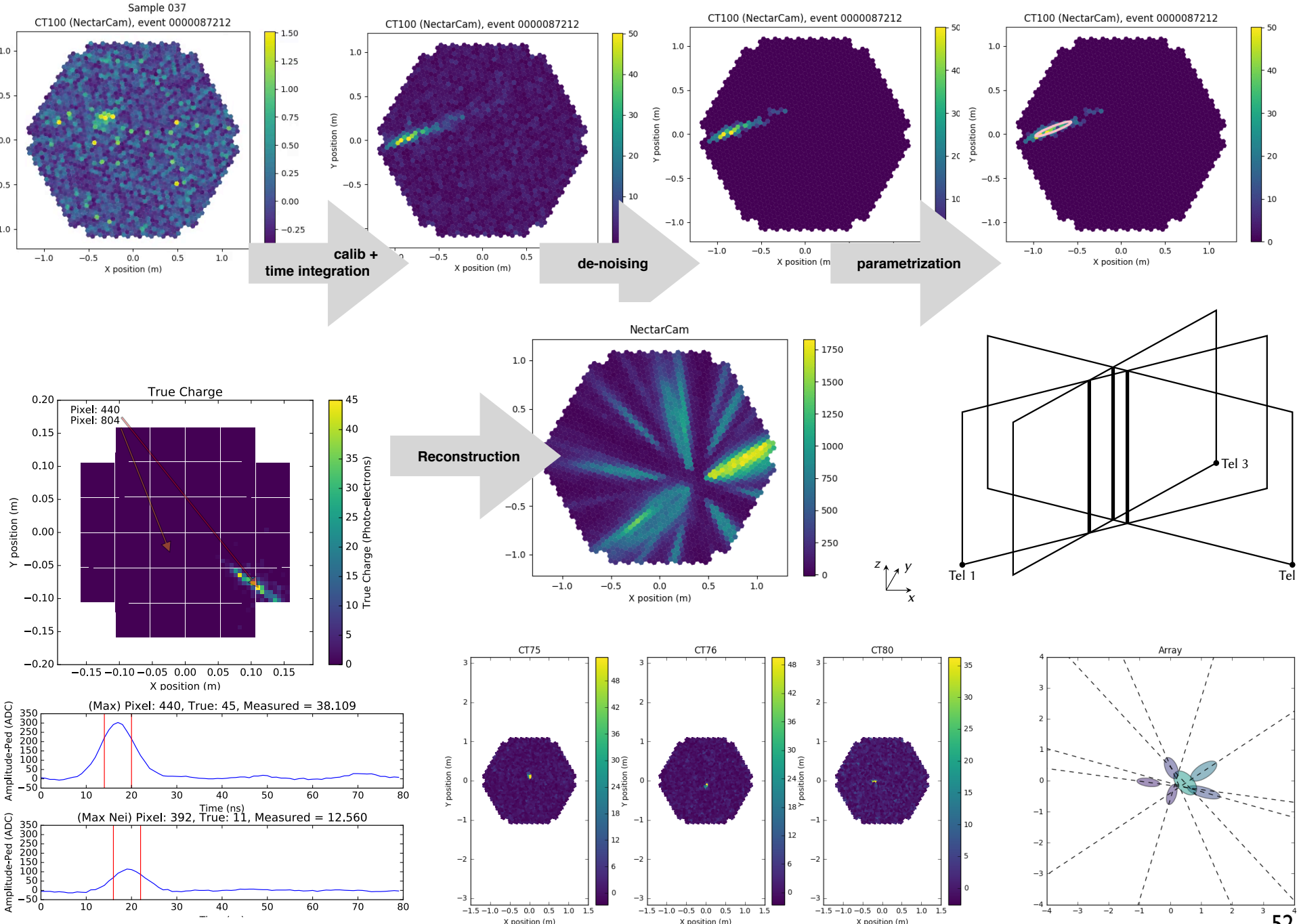











Sample 037
CT100 (NectarCam), event 0000087212
CT100 (NectarCam), event 0000087212
CT100 (NectarCam), event 0000087212
CT100 (NectarCam), event 0000087212
X position (m)
Y position (m)
calib +
time integration
de-noising
parametrization
True Charge
Pixel: 440
Pixel: 804
True Charge (Photo-electrons)
NectarCam
Reconstruction
Tel 1
Tel 3
z
y
x
(Max) Pixel: 440, True: 45, Measured = 38.109
(Max Nei) Pixel: 392, True: 11, Measured = 12.560
Amplitude-Ped (ADC)
Time (ns)
CT75
CT76
CT80
Array

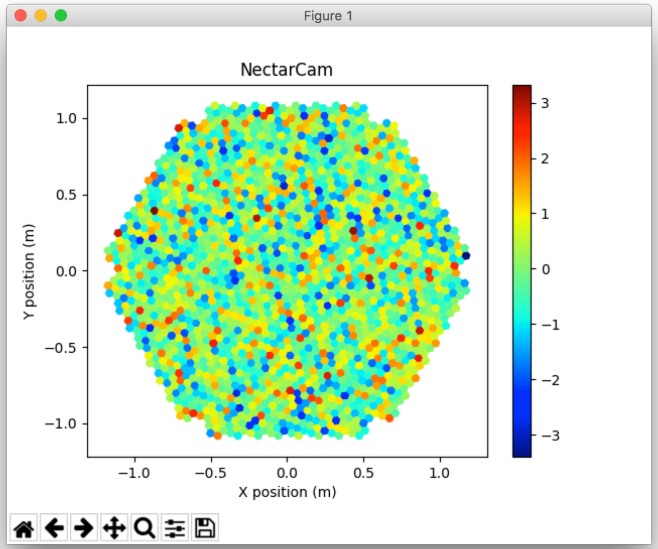
CameraDisplay

```
from ctapipe.visualization import CameraDisplay
from ctapipe.instrument import CameraGeometry

camera = CameraGeometry.from_name("NectarCam")
disp = CameraDisplay(camera)

disp.image = np.random.normal(size=camera.pix_x.shape)

disp.cmap = 'jet'
disp.add_colorbar()
```



- updating the image is fast, but constructing the display is slow (reuse the same display when possible)

CameraDisplay
autoscale : bool autoupdate : bool axes cmap cmap : str colorbar : NoneType geom image image : NoneType norm norm : str pixel_highlighting pixels : NoneType, PatchCollection
add_colorbar() add_ellipse() clear_overlays() enable_pixel_picker() highlight_pixels() on_pixel_clicked() overlay_moments() set_limits_minmax() set_limits_percent() show() update()

EventSources

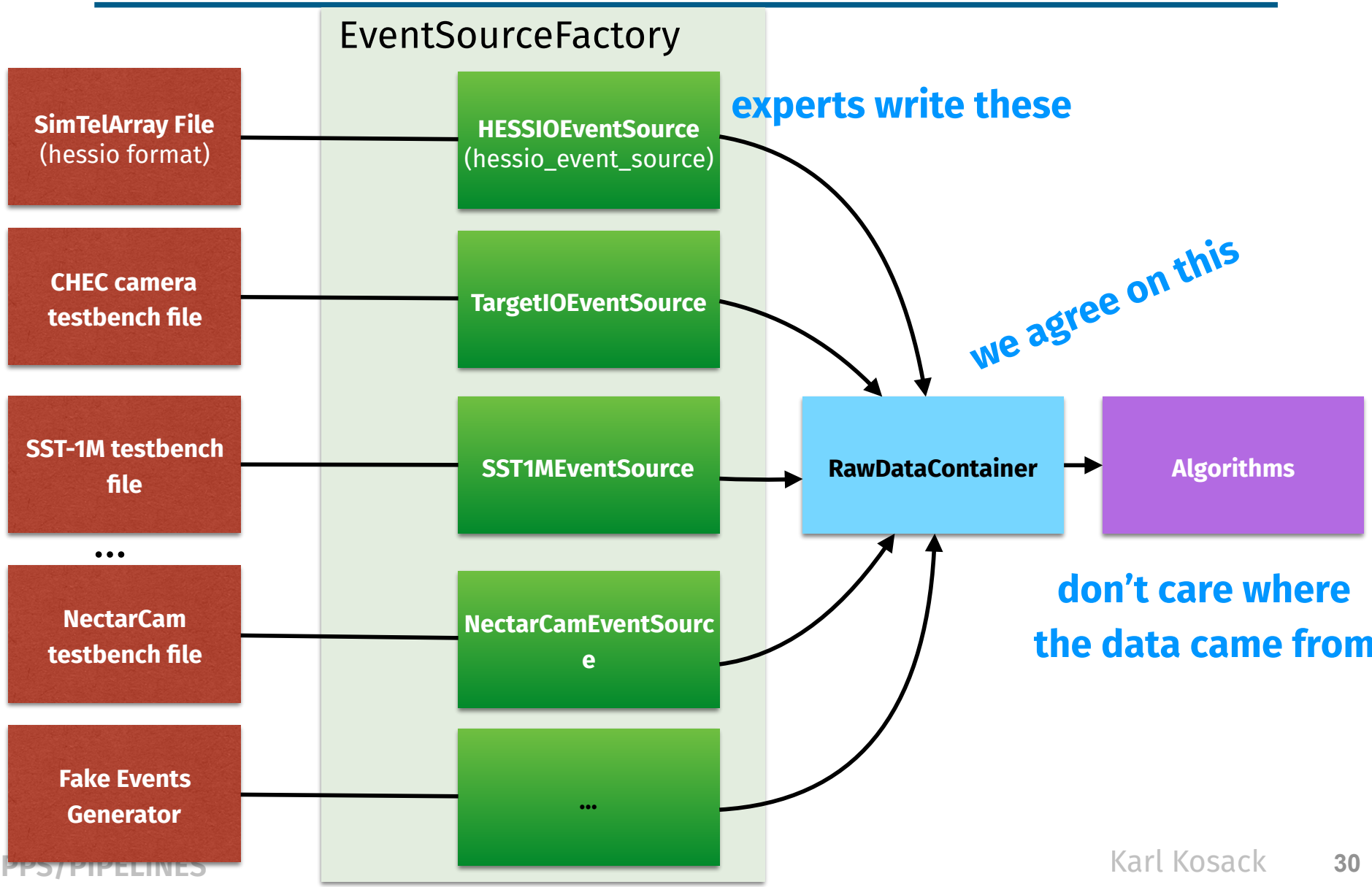
Interface:

- An *iterable* object, that emits DataContainers
- You can now loop more than once over the same EventSource (it will start from the beginning each time)
- **IMPORTANT**: by default the container returned is not a new container, but the same one with new information (no copy is made)
  - This is to prevent more than one event in memory at a time, for efficiency
  - if you want to store/cache several Containers, you must *deepcopy* them.

Useful options:

- allowed_tels = [1,2,3,4]:
  - specify a list of telescopes that are to be read, others will be skipped
  - *evt.r0.tels_with_data* will only show telescopes in this list
- max_events=n:
  - stop iteration after n events

EventSources



Hands on

- Run notebooks in the **notebooks/ctapipe** folder
  - ctapipe_handson.ipynb