softWAre for Dark matter ExpeRiments with Skippers

WADERS Framework

pronounced [v AI d e r s]

gitlab repo: https://gitlab.in2p3.fr/damicm/pysimdamicm

official web for documentation: https://ncastell.web.cern.ch/ncastell/

DQM Compton web: https://gev.uchicago.edu/compton/
(documentation **for analysis**)

Núria Castelló-Mor

IFCA Instituto de Física de Cantabria
CSIC Consejo Superior de Investigaciones Científicas
UC Universidad de Cantabria

1st DAMIC-M software school

11-13 January 2021
Online

https://indico.in2p3.fr/event/22866/

softWAre for Dark matter ExpeRiments with Skippers

WADERS Framework

pronounced [v AI d e r s]

the current name **pysimdamicm** became outdated!

Do you like WADERS?
Some alternatives ....

PROMISE	**P**ython sofwa**R**e f**O**r dark **M**atter **I**n **S**kipp**E**rs
SCAN	**S**kipper **C**CDs **An**alysis
PARANOIa	**P**ython softw**A**re fo**R** **A**nalysis of sk**I**ppers
PyDAMICM	
PARKER	**P**ython **A**nalysis fo**R** s**K**ipp**ER**s
SKRAP	**SK**ippe**R** **A**nalysis in **P**ython
ADAMS	**A**nalysis for **DA**rk **M**atter in **S**kippers

Do not like it .... suggestion?

Mariangela, Danielle, Rocio, Jordi, Carly thanks for your suggestions soft**WA**re for **D**ark matter **E**xpe**R**i

WADERS Framewo

pronounced [v AI d e r s]

the current name **pysimdamicm** became outdated!

Do you like WADERS?
Some alternatives ....

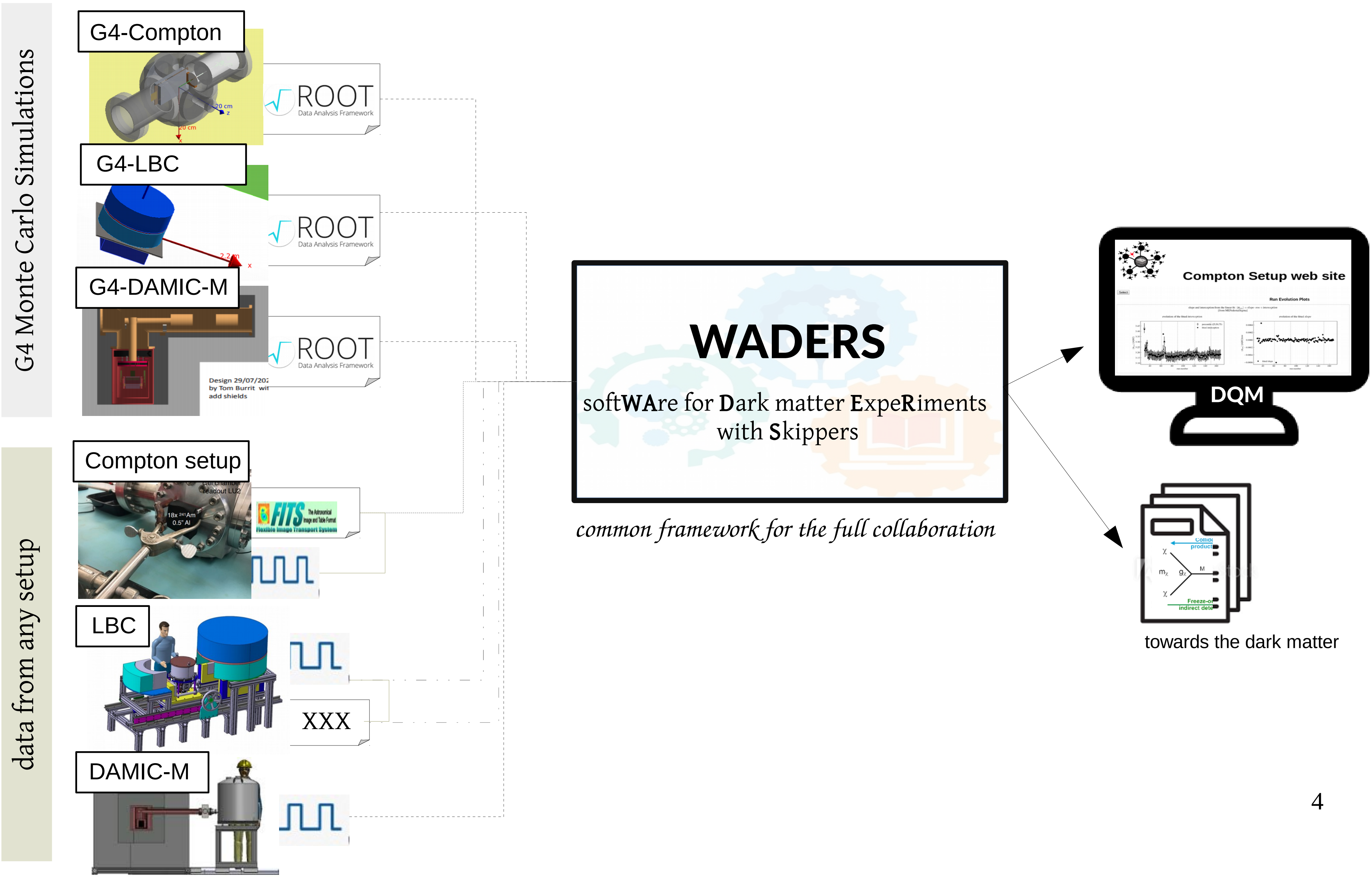
PROMISE	**P**ython sofwa**R**e f**O**r dark **M**atter **I**n **S**kipp**E**rs
SCAN	**S**kipper **C**CDs **An**alysis
PARANOIa	**P**ython softw**A**re fo**R** **A**nalysis of sk**I**ppers
PyDAMICM	
PARKER	**P**ython **A**nalysis fo**R** s**K**ipp**ER**s
SKRAP	**SK**ippe**R** **A**nalysis in **P**ython
ADAMS	**A**nalysis for **DA**rk **M**atter in **S**kippers

Do not like it .... suggestion?

Mariangela, Danielle, Rocio, Jordi, Carly thanks for your suggestions

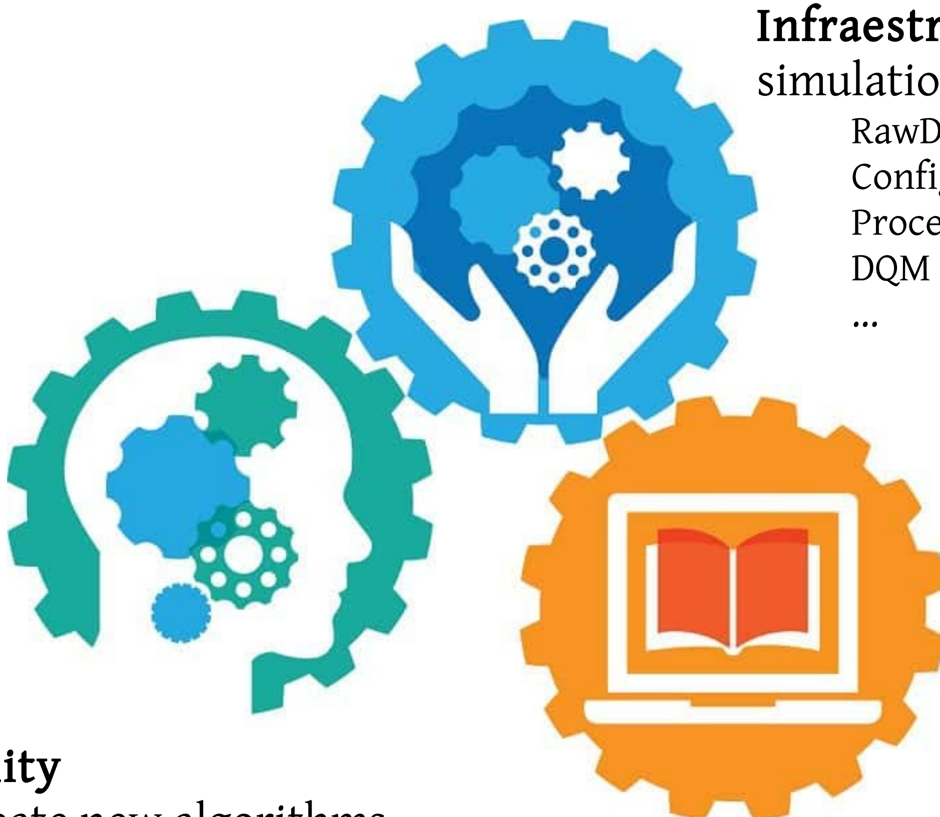
What is WADERS?

A framework which allows to post process data from skipper images [setup independent]



What is WADERS?

Is a framework



Infraestructure

simulations and reconstruction of skippper images

- RawData class (input file encapsulation)
- Config class (configuration)
- Process Manager
- DQM Manager
- ...

Library of processes

Collection of predefined algorithms (processes) to process the data

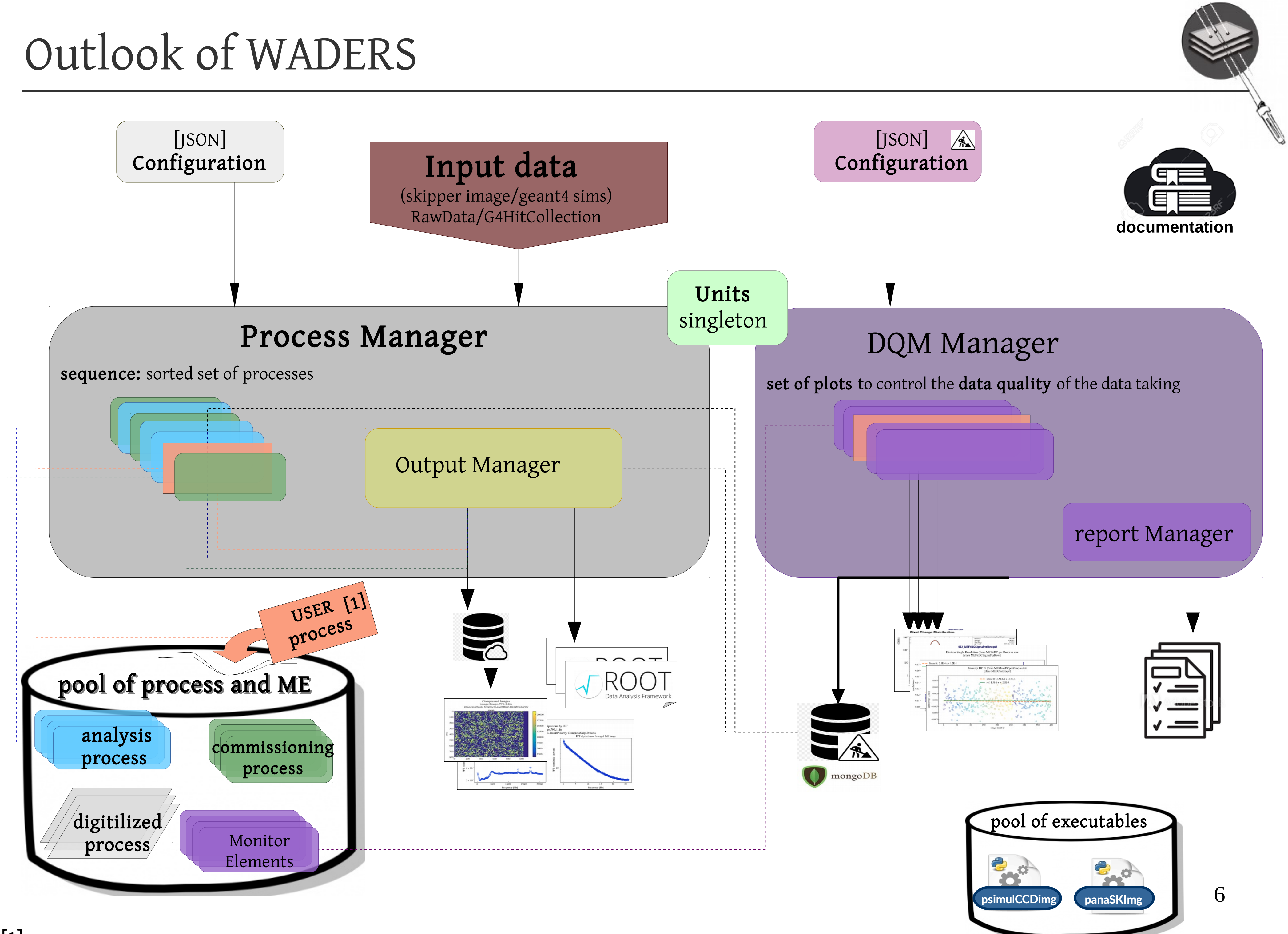
- CompressSkipperProcess
- PedestalSubtractProcess
- CalibrationProcess
- FitDarkCurrentProcess
- FitCalibrationConstant
- RnvsNskipsPlot
- FFTNoisePlot
- ChargeLossPlot
- SignalPatterRecognition
- ClusterFinder
- ...

(some processes from Paolo's CCDTestPython)

Plug-in ability

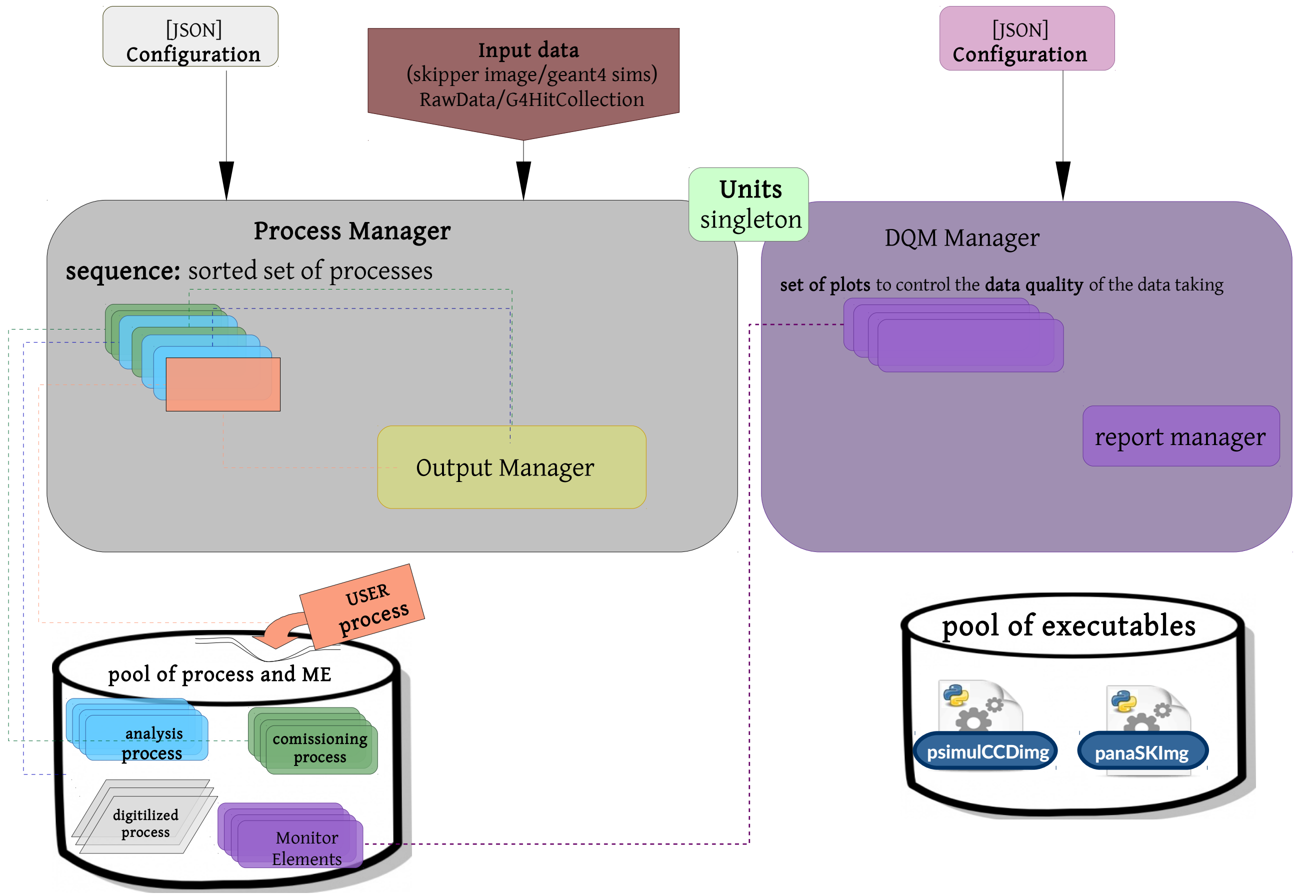
Allows to create new algorithms to process skipper images within this infrastructure

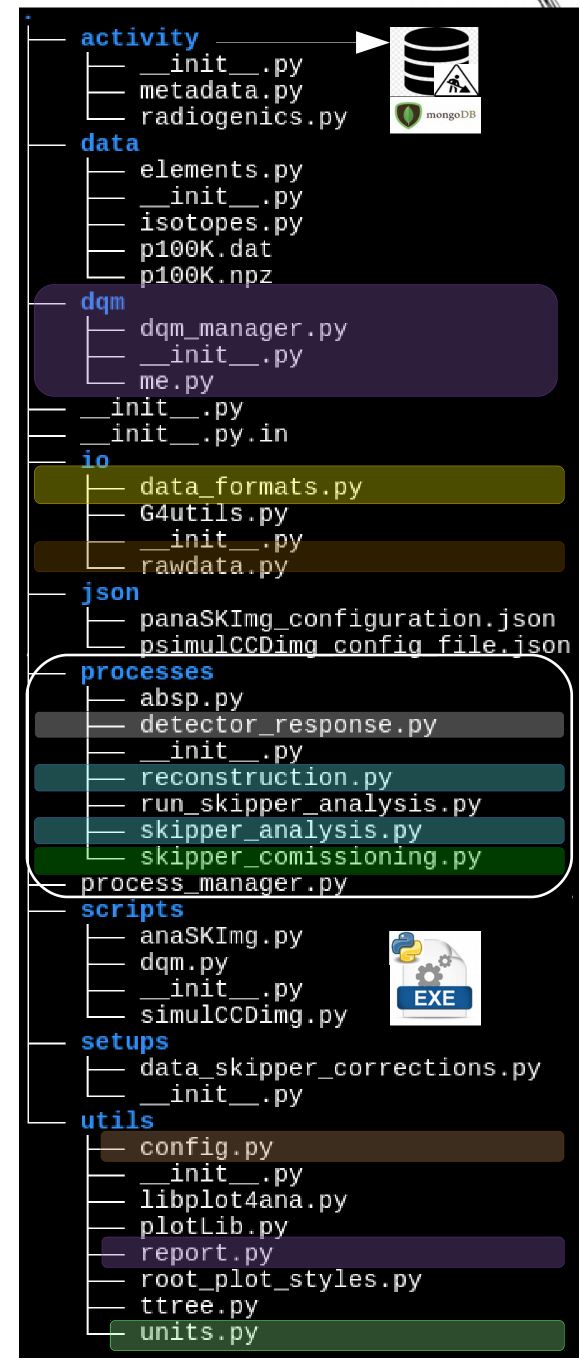
Outlook of WADERS



[1] new process and/or ME can be added for the user

Outlook of WADERS





documentation

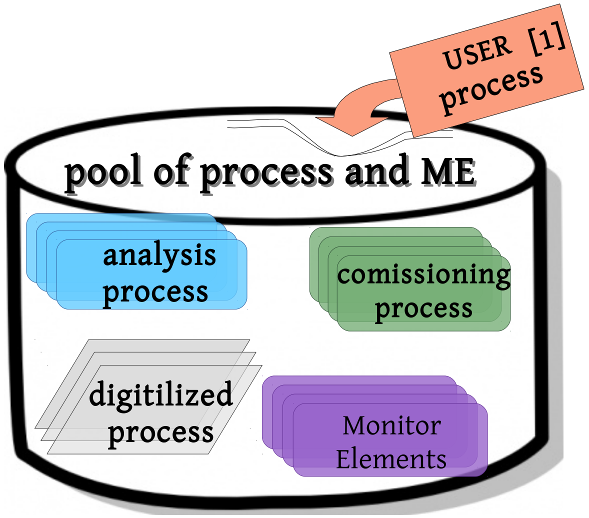
gitlab repo: https://gitlab.in2p3.fr/damicm/pysimdamicm

official web for documentation: https://ncastell.web.cern.ch/ncastell/
(mostly documentation for simulations)

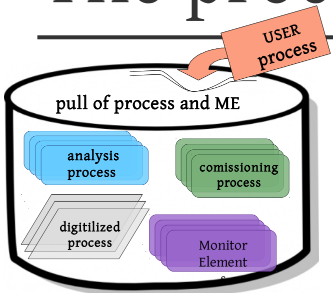
DQM Compton web: https://gev.uchicago.edu/compton/
(documentation for analysis)

This framework is built around

the **process** concept



The process concept

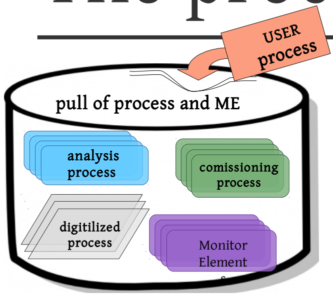


WADERS is built around the **process** concept

`process`: **algorithm implementation** which accesses, and/or modifies, and/or manipulates, and/or filter some input data, and probably adds some attributes to the original data.

- *A process should **focus in one, and just one concrete goal***

The process concept



WADERS is built around the **process** concept

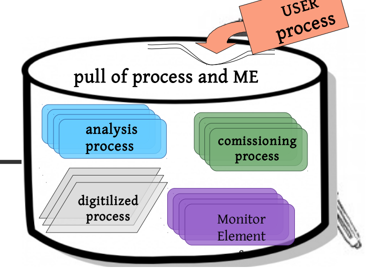
`process`: **algorithm implementation** which accesses, and/or modifies, and/or manipulates, and/or filter some input data, and probably adds some attributes to the original data.

- *A process should **focus in one, and just one concrete goal***

Each process is **configurable** at runtime by the user, through a JSON file.

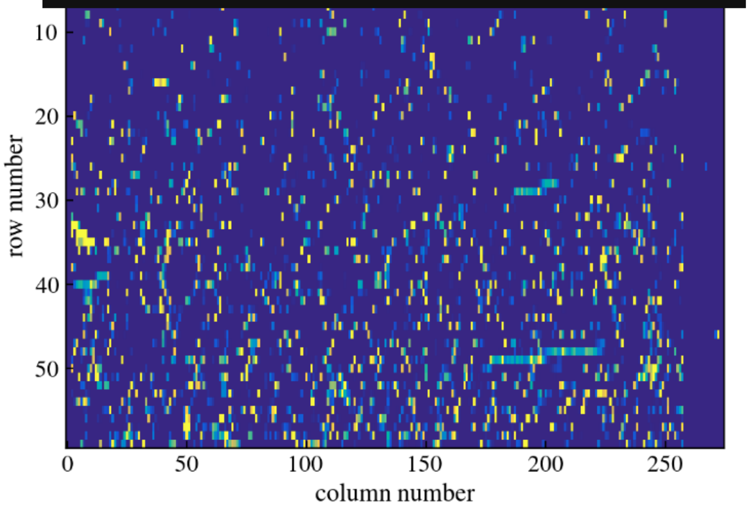
```
    "detector_response" :
    {
        "Diffusion":
        {
            "model"      : "gauss",
            "A"          : 216.2,
            "B"          : 0.000886,
            "z_offset"   : 0.0,
            "alpha"      : 0.0000597,
            "units"      : "A:um*um;B:1/um;z_offset:um;alpha:pixel/eVee",
            "fanofactor": 0.12,
            "active"     : 1
        },
        "ContinuousReadout":
        {
            "pixel_read_time" : 0.001,
            "units"           : "s/pixel",
            "ampli"           : 4,
            "active"          : 0
        },
        "PixelizeSignal":
        {
            "shift_pixel_in_x" : 0,
            "shift_pixel_in_y" : 0,
            "units"            : "N_pixels_in_x:pixel;pixel_size_in_x:mm;shift_pi
            "active"           : 1
        },
```

Configuration of a process via an example



Let’s see **PedestalSubtractionProcess** as an example

rawdata.image_mean_compressed

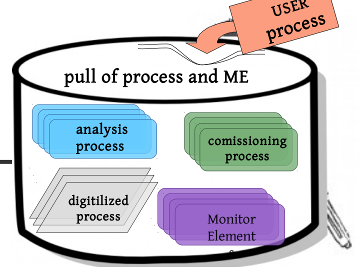


Algorithm to estimate and subtract the pedestal from an image.

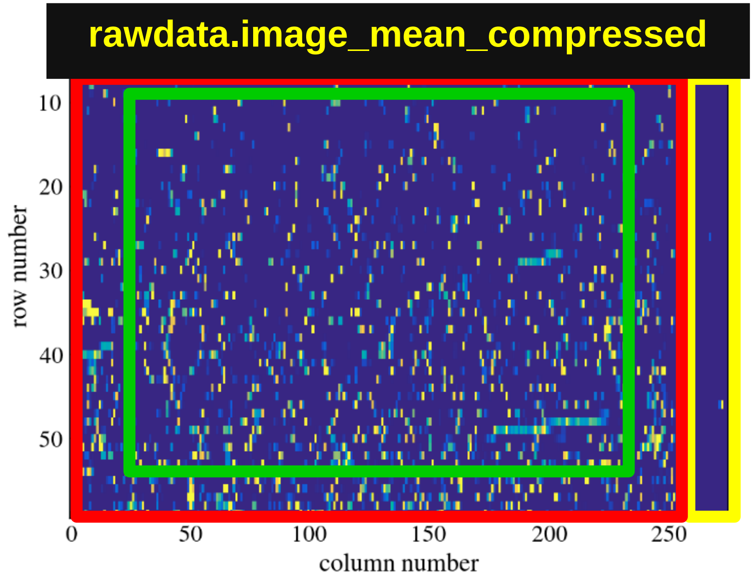
pedestal subtracted image

https://gev.uchicago.edu/compton/howto_notebooks/analysis/III_howto_PedestalSubtractedProcess.html

Configuration of a process via an example



Let's see **PedestalSubtractionProcess** as an example



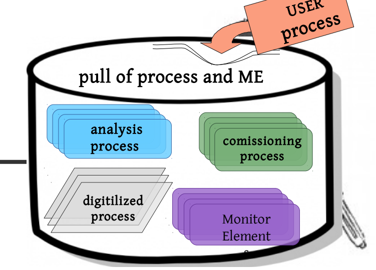
Algorithm to estimate and subtract the pedestal from an image.

The behaviour can be changed depending on the configured options

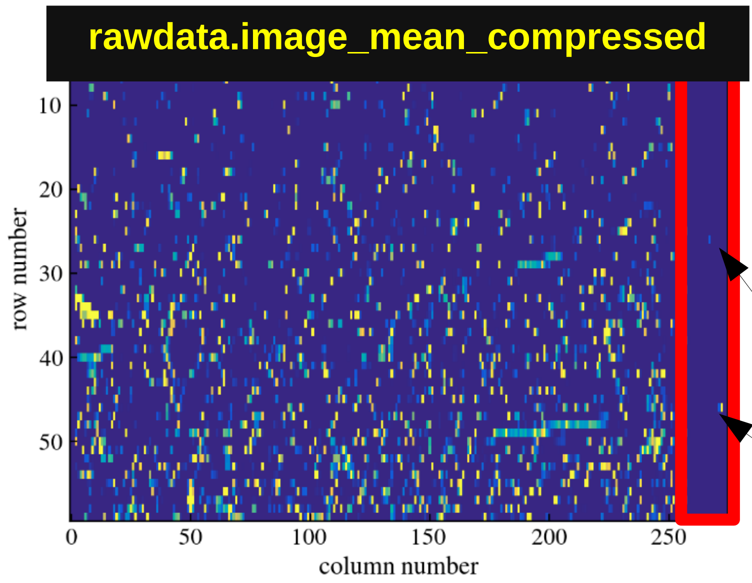
- use overscan or full image or just a region of the image
- mask pixels qith $q_{ij} > n\, q_{th}$
- estimated by row, or by column, or by both
- estimate by fitting a gaussian, or mean or a median
- for gaussian: estimate the pixels to consider by using MAD or STD
- .... other general parameters (related to the interactive mode)

https://gev.uchicago.edu/compton/howto_notebooks/analysis/III_howto_PedestalSubtractedProcess.html

Configuration of a process via an example



Let’s see **PedestalSubtractionProcess** as an example



Algorithm to estimate and subtract the pedestal from an image.

The behaviour can be changed depending on the configured options

- use overscan or full image or just a region of the image
- mask pixels qith $q_{ij} > n\ q_{th}$
- estimated by row, or by column, or by both
- estimate by fitting a gaussian, or mean or a median
- for gaussian: estimate the pixels to consider by using MAD or STD
- .... other general parameters (related to the interactive mode)

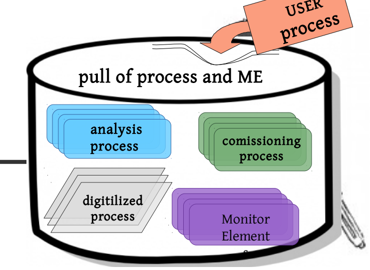
This algorithm include the posibility to mask some pixels in the selected region before to estimate the pedestal.
Given the value *n* all pixels with

$$q_{ij} > n\ q_{th}$$

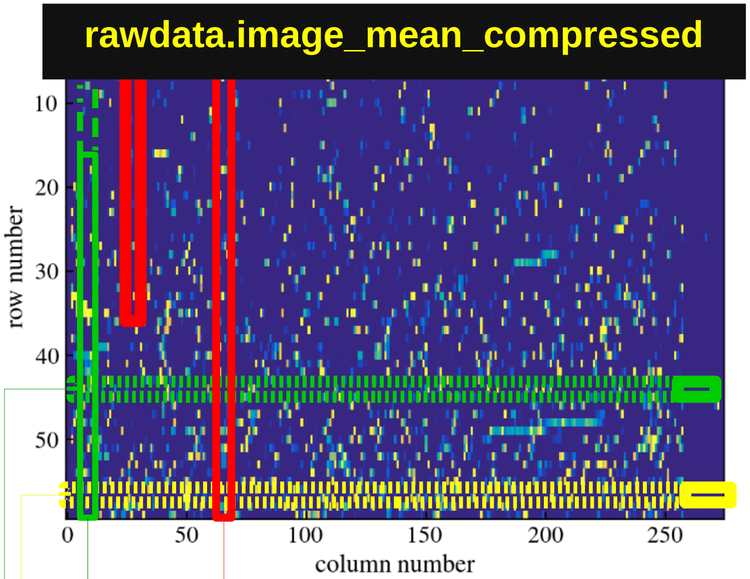
Will not be used to estimate the pedestal

https://gev.uchicago.edu/compton/howto_notebooks/analysis/III_howto_PedestalSubtractedProcess.html

Configuration of a process via an example



Let's see **PedestalSubtractionProcess** as an example



Algorithm to estimate and subtract the pedestal from an image.

The behaviour can be changed depending on the configured options

- use overscan or full image or just a region of the image
- mask pixels qith $q_{ij} > n\, q_{th}$
- estimated by row, or by column, or by both, or full
- estimate by fitting a gaussian, or mean or a median
- for gaussian: estimate the pixels to consider by using MAD or STD
- .... other general parameters (related to the interactive mode)

► pixels used to estimate the pedestal for column j

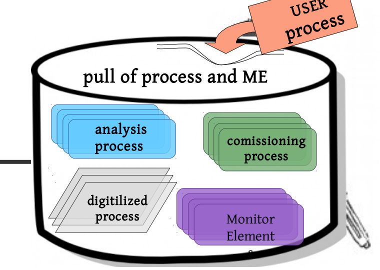
► pixels used to estimate the pedestal of row k

► pixels used to estimate the pedestal of row l

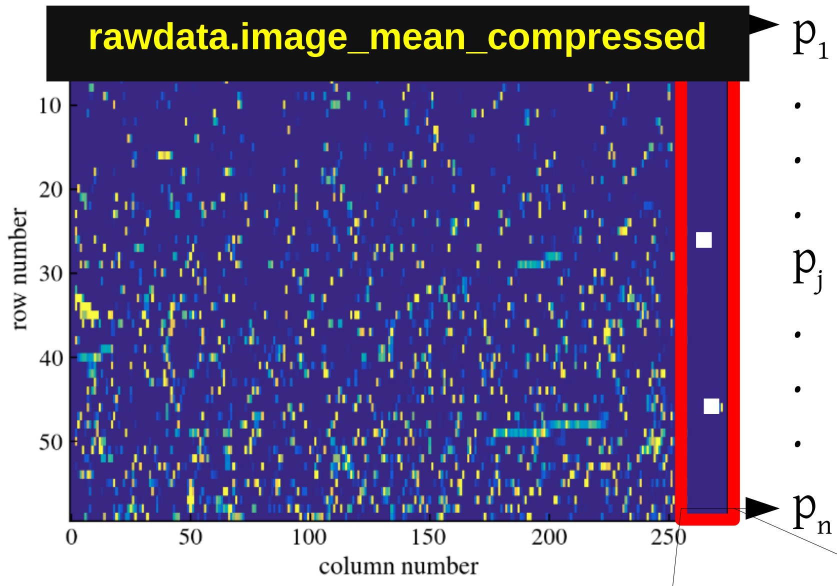
► pixels used to estimate the pedestal of column i (after subtraction estimated-row pedestals)

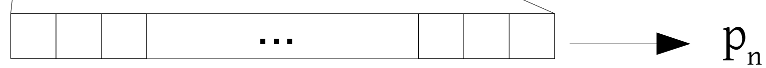
https://gev.uchicago.edu/compton/howto_notebooks/analysis/III_howto_PedestalSubtractedProcess.html

Configuration of a process via an example



Let’s see **PedestalSubtractionProcess** as an example





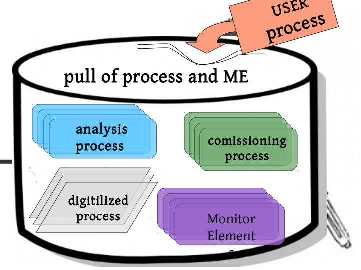
Algorithm to estimate and subtract the pedestal from an image.

The behaviour can be changed depending on the configured options

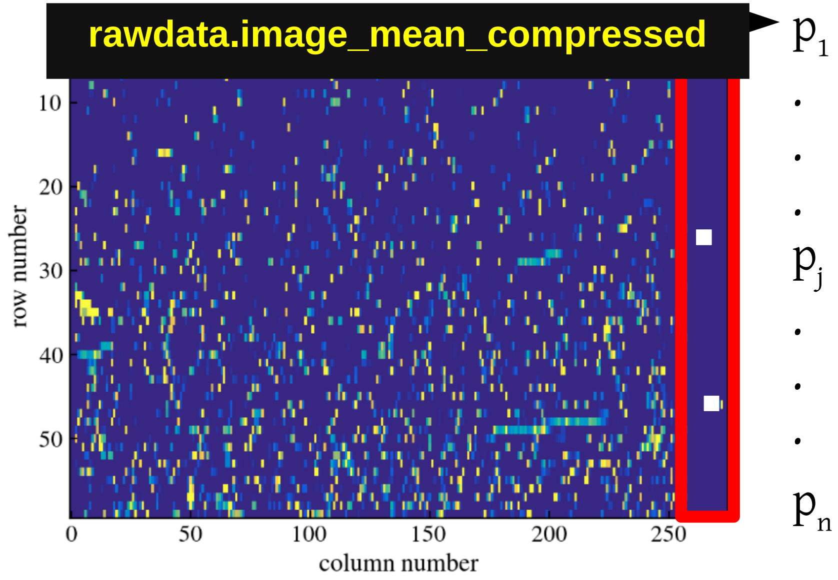
- use overscan or full image or just a region of the image
- mask pixels qith $q_{ij} > n\ q_{th}$
- estimated by row, or by column, or by both, or full
- estimate by fitting a gaussian, or mean or a median
- for gaussian: estimate the pixels to consider by using MAD or STD
- .... other general parameters (related to the interactive mode)

https://gev.uchicago.edu/compton/howto_notebooks/analysis/III_howto_PedestalSubtractedProcess.html

Configuration of a process via an example



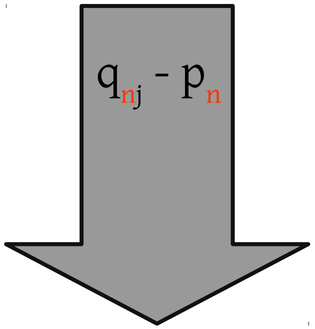
Let’s see **PedestalSubtractionProcess** as an example



Algorithm to estimate and subtract the pedestal from an image.

The behaviour can be changed depending on the configured options

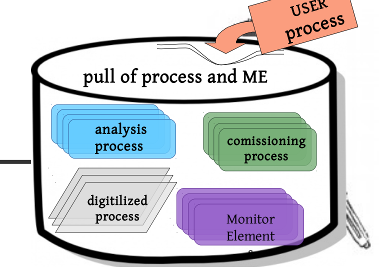
- use overscan or full image or just a region of the image
- mask pixels qith $q_{ij} > n\, q_{th}$
- estimated by row, or by column, or by both, or full
- estimate by fitting a gaussian, or mean or a median
- for gaussian: estimate the pixels to consider by using MAD or STD
- .... other general parameters (related to the interactive mode)



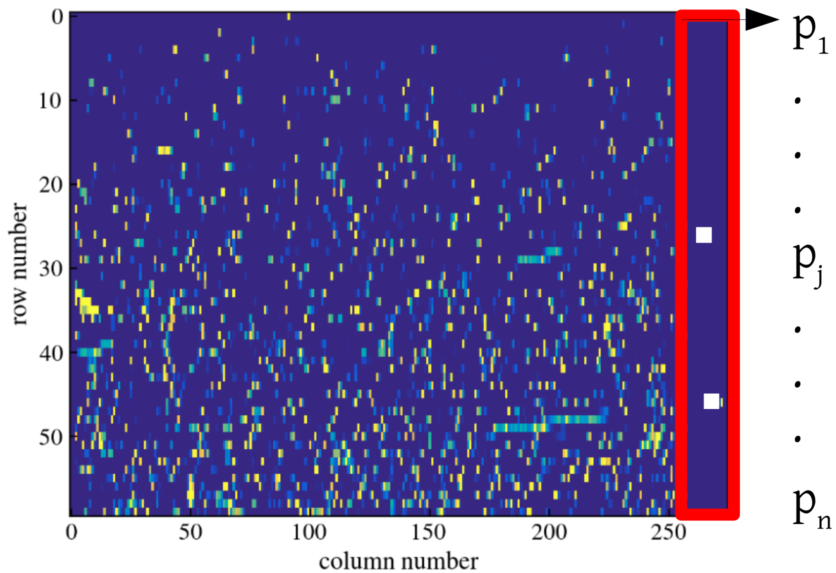
pedestal subtracted image

https://gev.uchicago.edu/compton/howto_notebooks/analysis/III_howto_PedestalSubtractedProcess.html

Configuration of a process via an example



Let's see **PedestalSubtractionProcess** as an example



Algorithm to estimate and subtract the pedestal from an image.

The behaviour can be changed depending on the configured options

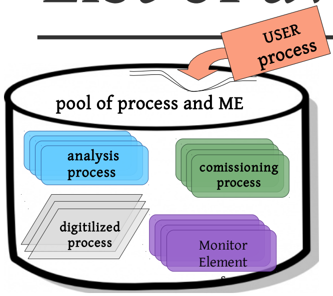
- use overscan or full image or just a region of the image
- mask pixels qith $q_{ij} > n\ q_{th}$
- estimated by row, or by column, or by both, or full
- estimate by fitting a gaussian, or mean or a median
- for gaussian: estimate the pixels to consider by using MAD or STD
- .... other general parameters (related to the interactive mode)

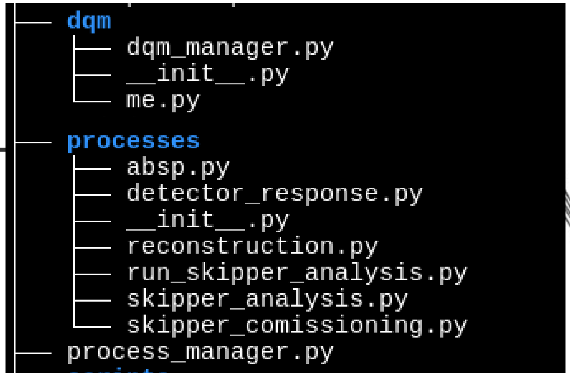
Set all options in the input JSON file:

```
        "PedestalSubtractionProcess":
        {
            "image":"mean_compressed",
            "method":"gauss_fit",
            "in_overscan":true,
            "use_mad":false,
            "axis":"row",
            "n_sigma_win_fit":3,
            "n_sigma_to_mask":-1,
            "show_fit":false,
            "histequ":false
        },
```


https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_Config.html

List of available processes





Depending on the working scope, the **processes** are grouped

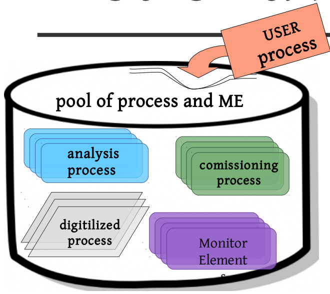
Abstract Class	
`DigitizeProcess`	**`detector response:`** to digitilize the geant4 monte carlo simulations (which have to be compatible with the one from DAMICG4) `pysimdamicm/processes/detector_response.py`
`SKImageProcess`	**`commissioning:`** for any [skipper] CCD image `pysimdamicm/processes/skipper_comissioning.py`
`SKImageProcess`	**`analysis:`** for any [skipper] CCD image `pysimdamicm/processes/reconstruction.py` `pysimdamicm/processes/skipper_analysis.py`
`MEabs`	**`Monitor Element:`** a plot or quantity as data quality of your data taking `pysimdamicm/dqm/me.py`

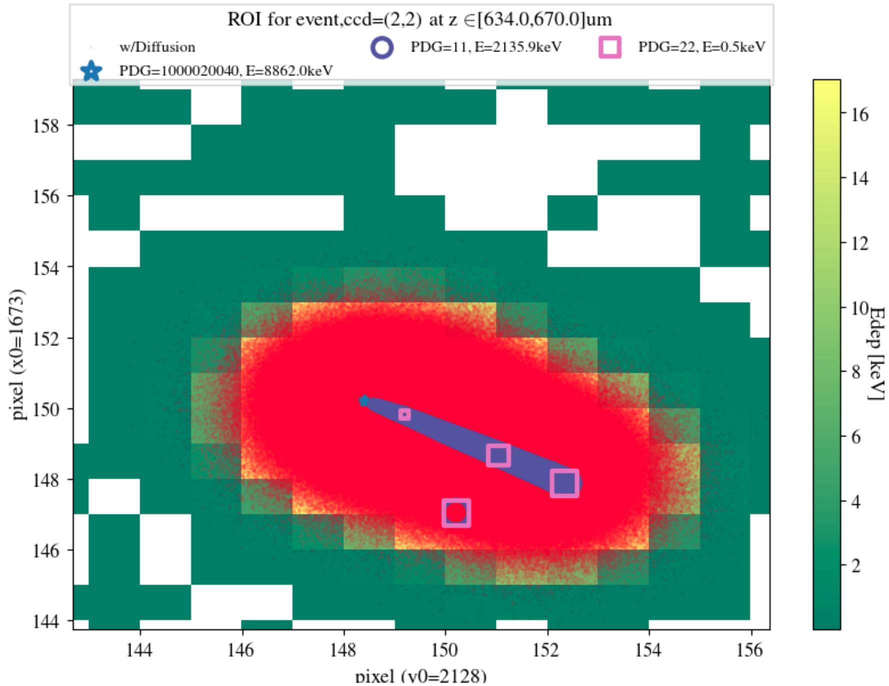
Visit the *Compton Setup web site* https://ncastell.web.cern.ch/ncastell/compton_web_site/
to see the full set of available ME

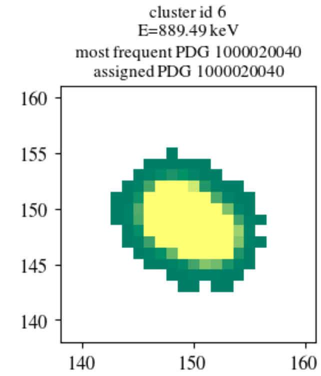
List of available processes

DigitizeProcess

detector response







Diffusion: Process to generate the creation of eh pairs and its diffusion

PixelizeSignal: Process to group the energy depositions into pixels

DarkCurrent: Process to add thermal generaion of eh pairs in the depletion region

ElectronicNoise: Process to emulate the electronic noise during readout

PixelSaturation: Process to emulate the saturation of one of the color channels of the sensor (which responds at its maximum value).
Note that the concequente overflow or blooming is not included

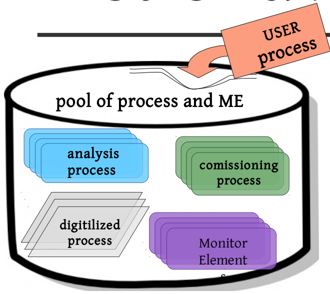
CreateFitsImage: Process to create a fits image to be used as DATA (allows to use a blank image as bkg)

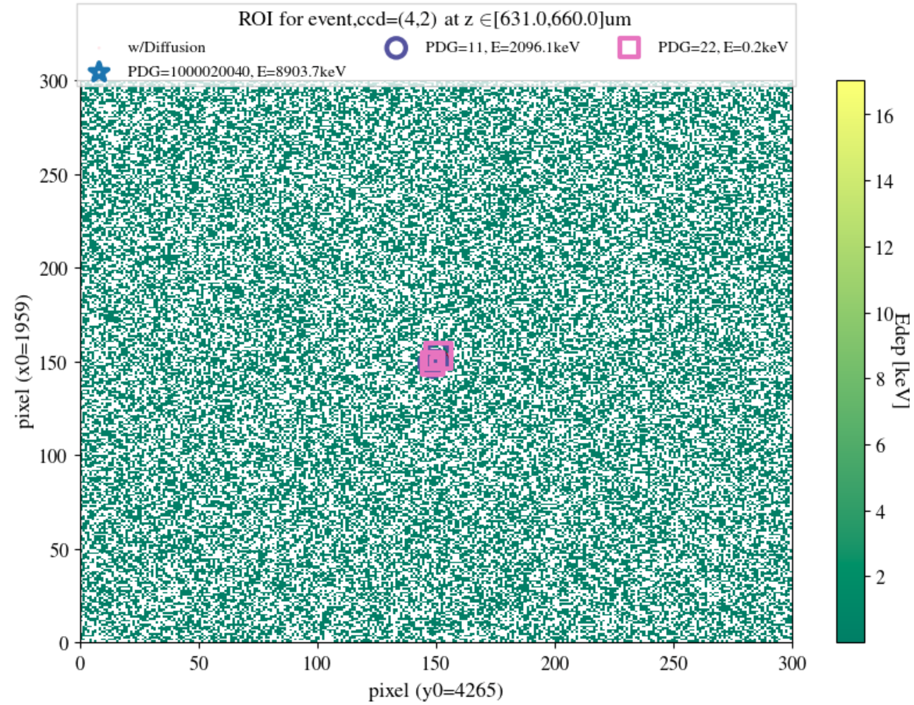
ContinuousReadout: Process to mimic the pixel charge loss due to continuous readout

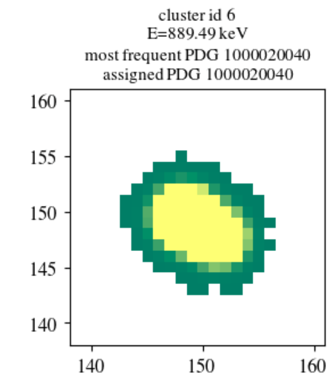
List of available processes

DigitizeProcess

detector response





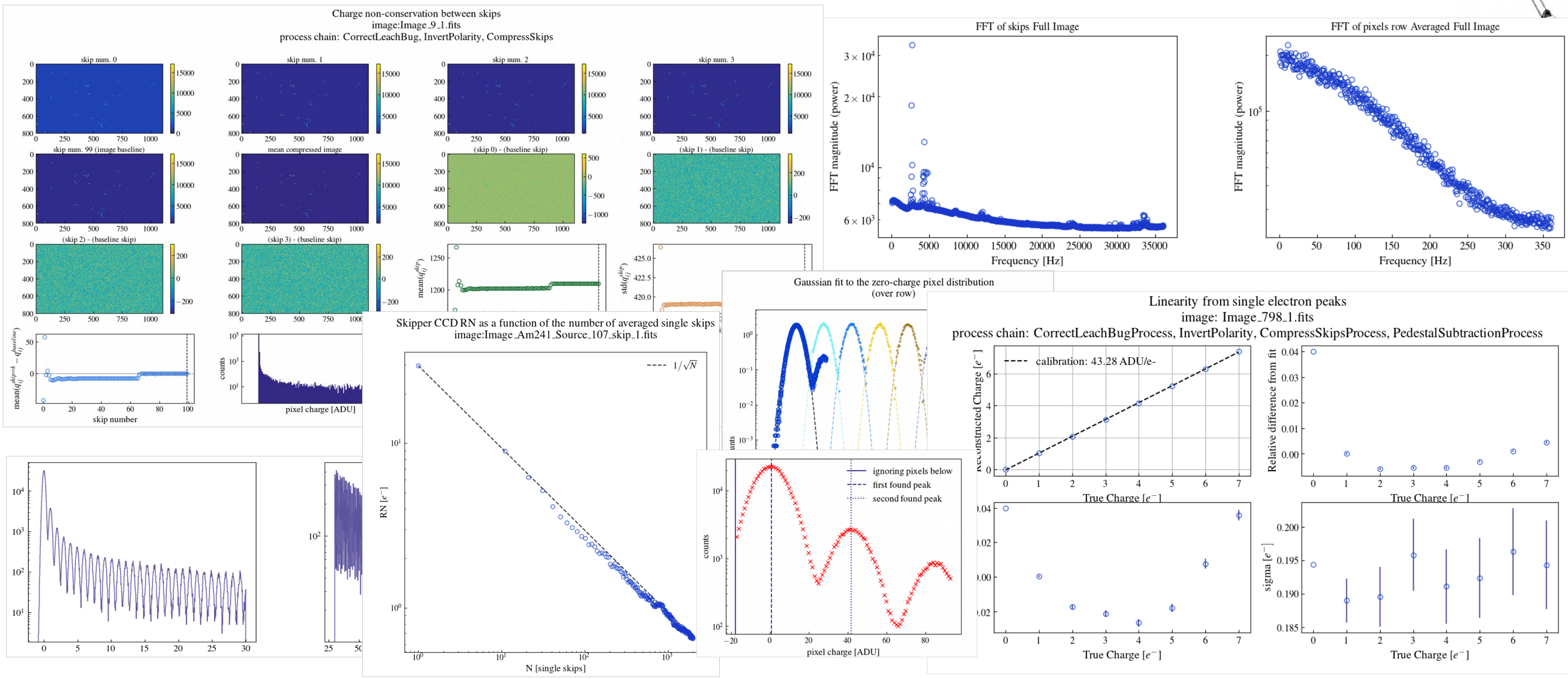


Diffusion:	Process to generate the creation of eh pairs and its diffusion
PixelizeSignal:	Process to group the energy depositions into pixels
DarkCurrent:	Process to add thermal generation of eh pairs in the depletion region
ElectronicNoise:	Process to emulate the electronic noise during readout
PixelSaturation:	Process to emulate the saturation of one of the color channels of the sensor (which responds at its maximum value). Note that the consecutively overflow or blooming is not included
CreateFitsImage:	Process to create a fits image to be used as DATA (allows to use a blank image as bkg)
ContinuousReadout:	Process to mimic the pixel charge loss due to continuous readout

List of available processes

SKImageProcess

commissioning



ChargeLossPlot: Process to create a set of plots to study charge loss between skips

FFTNoisePlot: Process to do the Fast Fourier Transform of an skipper image to study its noise in skips and pixels

FitCalibrationConstant: Process to find the calibration constant (or gain) by fitting n-consecutive single electron peaks.

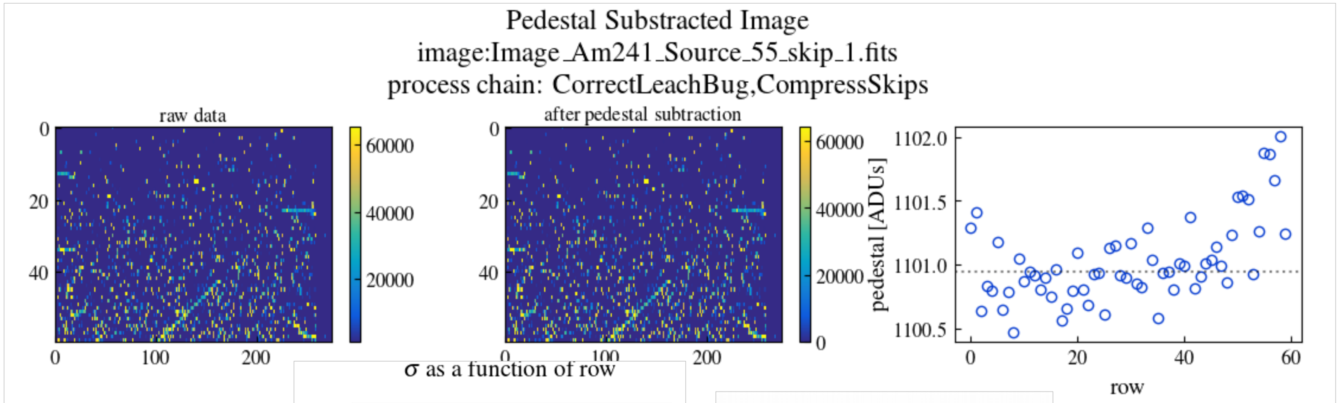
RNvsNskipsPlot: Process to study the signal-to-noise ratio as a function of N_{skips}

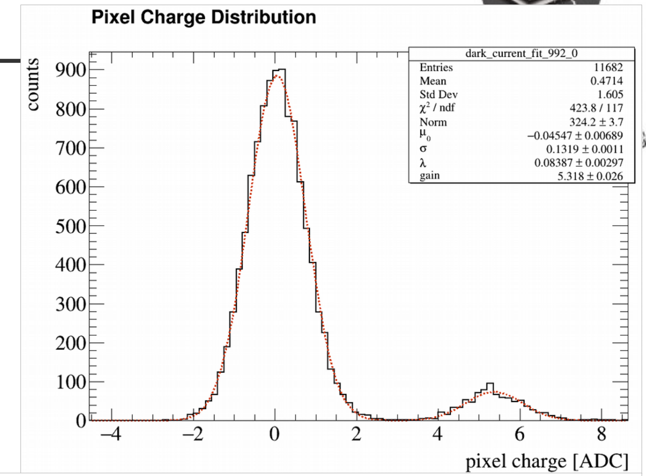
Mostly to create a set of plots to study skipper CCD images

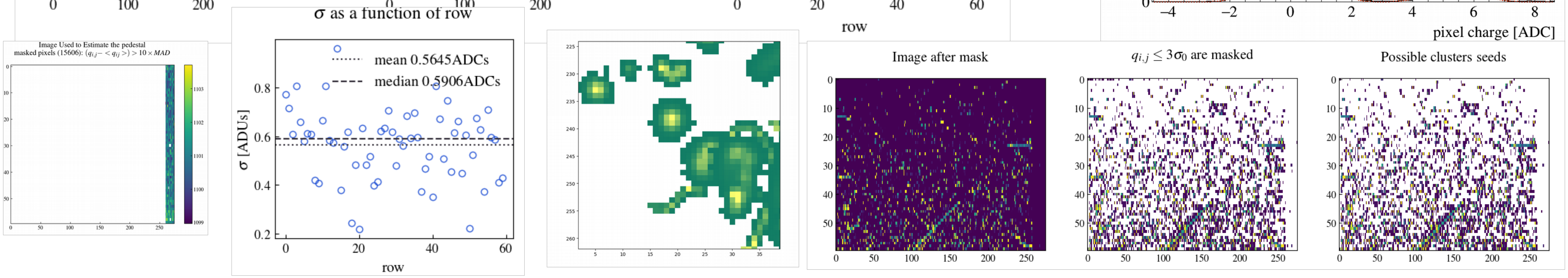
List of available processes

SKImageProcess

analysis







CompressSkipperProcess: Process to combine all single skips into a single value

PedestalSubtractionProcess: Process to get the pedestal-subtracted image (pedestal estimation and subtraction)

FitDarkCurrentProcess: Process to fit λ, μ_0, gain, single e$^-$ resolution

$$\sum_{n=0} P(n \cdot k|\lambda) \times G(q_{ij}, \mu_0 + (n \cdot k), \sigma_{pix})$$

CalibrationProcess: Process to apply the calibration to signal: convert ADU to → e-

[also used in simulations]

SignalPatterRecognition: Process for the discriminate between signal and noise

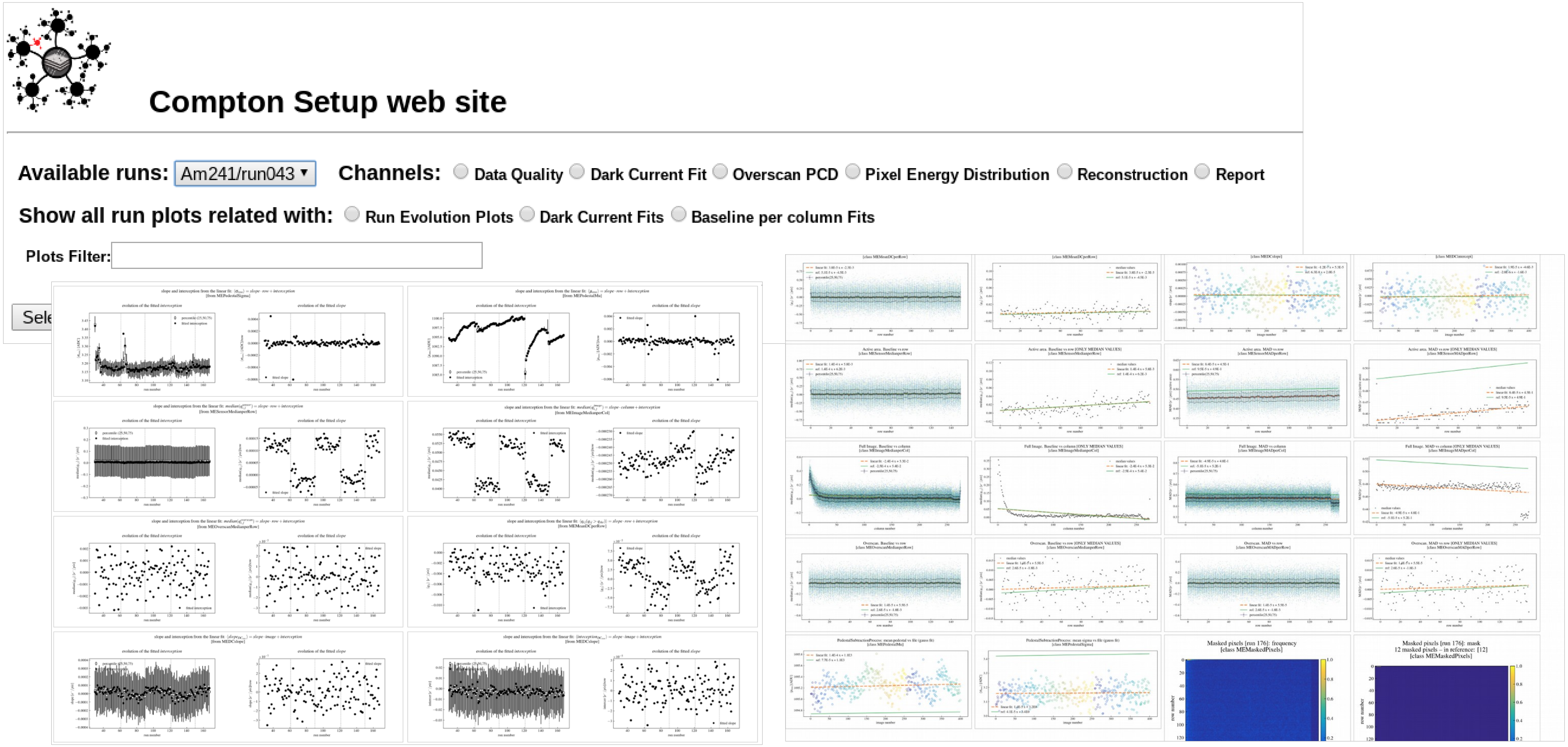
ClusterFinder: Process to find clusters sensor

List of available processes

MEabs

Monitor Elements

Go to https://ncastell.web.cern.ch/ncastell/compton_web_site/



MEMeanDCperRow, MEDCslope, MEDCintercept, MESensorMedianperRow, MESensorMADperRow, MEImageMedianperCol, MEImageMADperCol, MEOverscanMedianperRow, MEOverscanMADperRow, MEPedestalMu, MEPedestalSigma, MEMaskedPixels, MESinglePED, MEOverscanPCDMu, MEOverscanPCDSigma, MEFitDC, MEFitDCMu0PerRow, MEFitDCSigmaPerRow,MEFitDCLambdaPerRow, MEFitDCMu0, MEFitDCSigma, MEFitDCLambda, METempShift

In collaboration with the Compton setup group

All processes are well documented

Dedicated documentation for each process can be found

documentation

→ USING THE EXECUTABLES PROVIDED BY WADERS

- for a short description

```
[user @dm ~]$  psimulCCDimg --json
```

```
[user @dm ~]$  panaSKImg --help
```

All processes are well documented

Dedicated documentation for each process can be f

→ USING THE EXECUTABLES PROVIDED BY WADER

```
[user @dm ~]$  psimulCCDimg --json
```

```
(venv_analysis) [castello@gev pysimdamicm]$ psimulCCDimg --json
Valid options are:
"in_root":
    "in_dir_path" : Path for <in_root.root_file_name> (by default  .)
    "root_file_name" : ROOT file name of the geant4-based DAMICG4 simulation
"out_root":
    "out_dir_path" : Path for the output root file (by default  .)
    "prefix_root_name" : Output file name prefix, the response of the detecto
                will be sotored as <dir_path>/<prefix_root_name>_<root_file_
    "store_pixel_info" : If set, the properties of each pixel in the cluster
    "save_CCDimg" : Boolean option to store the image for each cluster in a n
           format with two independent arrays: `energy` and `noise`
           for the pixel energy of the geant4 simulation and the pixel
           noise distribution due to dark current and/or electronic noise. (by d
"detector_response":
    "Diffusion":
      "model" : Use karthick to apply probability distributions to get the nu
      "A" : Coefficient for the xy dispersion measured by the
                detector as a function of the depth (by default  216.2)
      "B" : Coefficient for the xy dispersion measured by the
                detector as a function of the depth (by default  0.000886)
      "z_offset" : Starting z-position in the silicon bulk (by default  0.0)
      "alpha" : Parameter for the second term on the diffusion model, the pro
      "units" : informative keyword to highlight the units of each process pa
      "fanofactor" : Fano factor to model e-h pair creation
                within the silicon bulk (by default  0.12)
      "active" : Include the detector response and/or reconstruction process
    "ContinuousReadout":
      "pixel_read_time" : Time to read a pixel in units of seconds (by defaul
      "units" : informative keyword to highlight the units of each process pa
      "ampli" : Readout: number of amplifiers used for skipper
                continous readout: 1/2/4 (by default  4)
      "active" : Include the detector response and/or reconstruction process
    "PixelizeSignal":
      "shift_pixel_in_x" : Skip the first `shift_pixel_in_x` pixels from the
      "shift_pixel_in_y" : skip the first `shift_pixel_in_y` pixels from the
      "units" : informative keyword to highlight the units of each process pa
      "active" : Include the detector response and/or reconstruction process
    "ElectronicNoise":
      "mode" : Full-image or cropped-image for intrinsic detector noise simul
      "pedestal" : Mean value in units of electrons/pixel (by default  0.0)
      "sigma" : STD value in units of electrons/pixel (by default  0.25)
      "units" : informative keyword to highlight the units of each process pa
      "img_size" : x and y-axis CCD region size. Both noises, dark current an
                are simulated in a reduced CCD region for computational reas
      "min_enlarge" : Minimum extra-pixel size when the elongation of the clu
      "active" : Include the detector response and/or reconstruction process
    "DarkCurrent":
      "mode" : Full-image or cropped-image for intrinsic detector noise simul
      "darkcurrent" : Dark current value in units of e-/pixel/day (by default
      "exp_time" : Time for the darkcurrent mesurement in units of days, bein
                the lambda parameter in the poisson distribution (by default
      "img_size" : x and y-axis CCD region size. Both noises, dark current an
                are simulated in a reduced CCD region for computational reas
      "min_enlarge" : Minimum extra-pixel size when the elongation of the clu
      "units" : informative keyword to highlight the units of each process pa
      "active" : Include the detector response and/or reconstruction process
    "PixelSaturation":
      "saturation" : Number of ADC units for a saturated pixel (by default  6
      "units" : informative keyword to highlight the units of each process pa
      "active" : Include the detector response and/or reconstruction process
"reconstruction":
    "SignalPatternRecognition":
      "method" : Algorithm number for cluster finder (by default  1)
      "threshold" : Minimum pixel charge as signal (by default  15)
```

continue ...

documentation

All processes are well documente

Dedicated documentation for each process can be

→ USING THE EXECUTABLES PROVIDED BY WADER

```
[user @dm ~]$  panaSKImg --help
```

```
                        id_skip_start in its scope in the json file)
  --skip-end ID_SKIP_END
                        First skip to start with (for all processes, if a
                        process use an specific starting point include
                        id_skip_start in its scope in the json file). Note
                        that -1 means last value
  --row-start ID_ROW_START
                        First row to start with (for all processes, if a
                        process use an specific starting point include
                        id_row_start in its scope in the json file)
  --row-end ID_ROW_END  First row to start with (for all processes, if a
                        process use an specific starting point include
                        id_row_start in its scope in the json file). Note that
                        -1 means last value
  --col-start ID_COL_START
                        First col to start with (for all processes, if a
                        process use an specific starting point include
                        id_col_start in its scope in the json file)
  --col-end ID_COL_END  First col to start with (for all processes, if a
                        process use an specific starting point include
                        id_col_start in its scope in the json file). Note that
                        -1 means last value

** Common to all PROCESS ************************** :
  -s SEQUENCE, --sequence SEQUENCE
                        Coma-separated list of process names (in this case,
                        sequence from json file will be ignored)
  --list-processes      List all available process names
  --save-img            [BOOL] Set to save intermediate images as fits files
  --save-plots          [BOOL] Set to save plots as eps and pdf files
  --display             [BOOL] Running in debug mode (all plots will be also
                        display)
  --verbose             [BOOL] Report extra information/plots during execution
                        (for all booked processes)
  --cal CALIBRATION     Calibration constant to start with (several process
                        has this parameter)

** For Process CompressSkipperProcess ************************* :
  --func-to-compress FUNC_TO_COMPRESS [FUNC_TO_COMPRESS ...]
                        CompressSkipperProcess. List of functions to reduce
                        the single skipper images into a single one (functions
                        must exist in the numpy package)

** For Process PedestalSubtractionProcess ************************** :
  --method METHOD       Method to use to compute the pedestal
  --in-overscan         Set to use the full image to estimate the pedestal,
                        instead of only the overscan region
  --axis AXIS           Axis in which the overscan should be computed:
                        row/col/both/none
  --n-sigma-win-fit N_SIGMA_WIN_FIT
                        Number of sigmas to define the spectral window to fit
                        a gaussian to single electron peaks
  --n-sigma-to-mask N_SIGMA_TO_MASK
                        Number of sigmas to define the maximum pixel charge to
                        take into account to estimate the pedestal
  --show-fit            Set to show up several extra plots and information

** For Process ChargeLossPlot ************************ :
  --skip-id-list SKIP_ID_LIST [SKIP_ID_LIST ...]
                        List of skip index to be display to search for charge
                        loss/gain
  --skip-id-baseline SKIP_ID_BASELINE
                        Index of the single skip image to be used as baseline
                        image
```

documentation

continue ...

All processes are well documented

documentation

Dedicated documentation for each process can be found at

→ ONLINE

- for a more detailed explanation (params, algorithm,examples)

https://ncastell.web.cern.ch/ncastell/pysimdamicm/getting_started.html

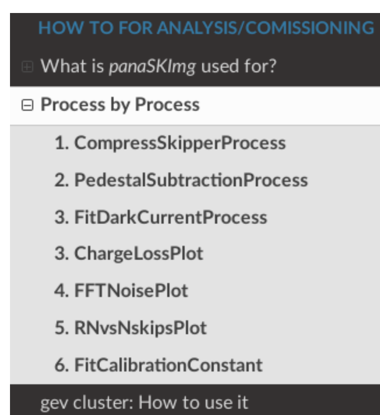
https://gev.uchicago.edu/compton/

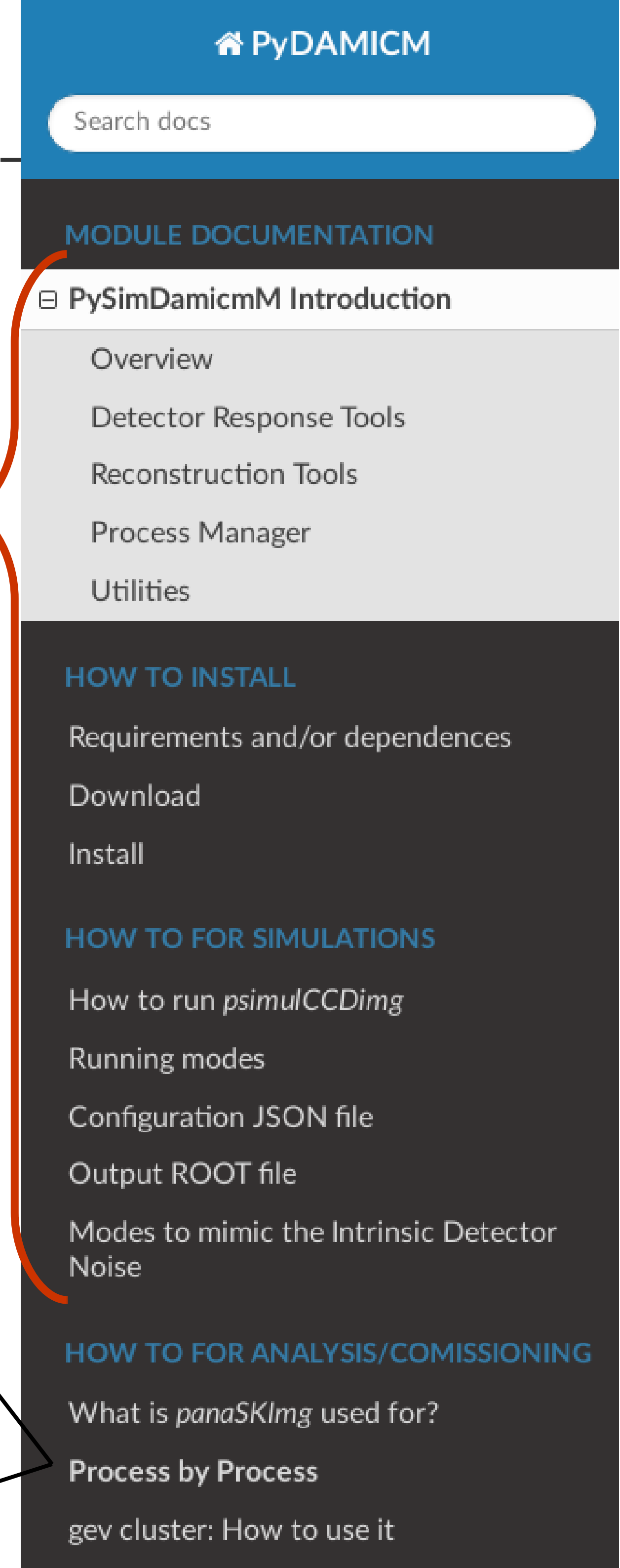
A gallery of Notebooks on how to use pysimdamicm for analysis:

- **How to I:** [Install]
- **How to II:** [Introduction]
- **How to III:** [Configuration JSON file]
- **How to IV:** [RawData class]
- **How to V:** [CompressSkipperProcess class]
- **How to VI:** [PedestalSubtractedProcess class]
- **How to VII:** [FitDarkCurrentProcess class]

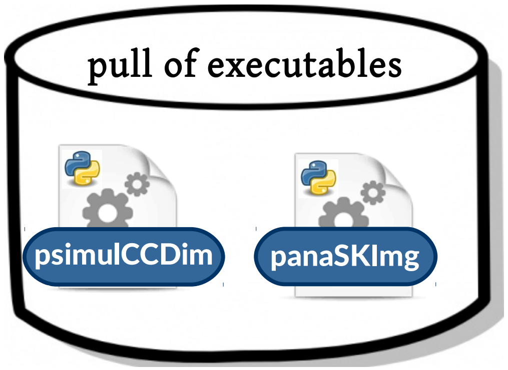
Find all notebooks (.html, .ipynb) also within the repository under the folder

```
pysimdamicm/notebooks/howto/analysis
```

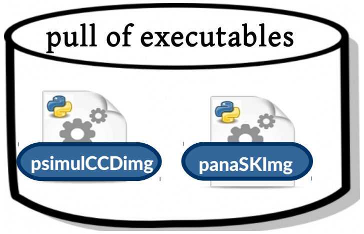




How to use it?



How to use WADERS



WADERS provides two executables

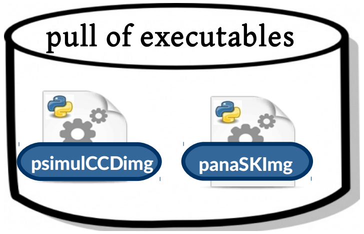
- configured via JSON file and/or command line options

`psimulCCDimg`**:** using a Geant4-created input file (ROOT file compatible with DAMICG4's output), simulates the CCD **sensor response** and, emulates the **electronic response** and performs **cluster reconstruction**
It can also provide a CCD image (fits format) to be processes with `panaSKImg` (as if it was real data).

`panaSKImg`**:** using (n) input image file(s) (fits,ROOT,...), performs data analysis (both **commissioning/operational** and high level data **reconstruction**).

- Comissioning: study the linearity with single electron peaks, noise studies, skip-charge losses studies, ...
- Operational: Data Quality Monitor
- High level data: pedestal subtraction, dark current fit, cluster reconstruction, ...

How to use WADERS: Configuration JSON file



WADERS provides two executables

- configured via JSON file and/or command line options

`psimulCCDimg`: main sections for the configuration JSON file

`output`: mostly to define the prefix of the output ROOT file
`detector_response`: define the **list of processes** and its parameters
`reconstruction`: define the **list of processes** and its parameters

> The user **has no control over the order** in which the process will be executed (these are internally sorted)

`panaSKImg`: main sections for the configuration JSON file

`input`: parameters to define the input image (overscan and prescan regions, active area, number of skip measurements, ...)
`process`: define the **list of ordered processes** and its parameters

> The **user must define the order** of execution of the process. This is done in the JSON file, setting the parameter `sequence`.

How to use WADERS: JSON file example for **psimulCCDimg**

pysimdamicm/json/psimulCCDimg_config_file.json

units –> ONLY INFORMATIOVE

```
{
    "in_root" :
    {
        "in_dir_path" : ".",
        "root_file_name": "None"
    },
    "out_root":
    {
        "out_dir_path" : ".",
        "prefix_root_name"  : "out_simulCCDimg",
        "store_pixel_info"  : 1,
        "save_CCDimg"       : 0
    },
    "detector_response" :
    {
        "Diffusion":
        {
            "model"     : "gauss",
            "A"         : 216.2,
            "B"         : 0.000886,
            "z_offset"  : 0.0,
            "alpha"     : 0.0000597,
            "units"     : "A:um*um;B:1/um;z_offset:um;alpha:p
            "fanofactor": 0.12,
            "active"    : 1
        },
        "ContinuousReadout":
        {
            "pixel_read_time" : 0.001,
            "units"           : "s/pixel",
            "ampli"           : 4,
            "active"          : 0
        },
        "PixelizeSignal":
        {
            "shift_pixel_in_x" : 0,
            "shift_pixel_in_y" : 0,
            "units"            : "N_pixels_in_x:pixel;pixel_s
            "active"           : 1
        },
        "ElectronicNoise":
        {
            "mode"     : "cropped-image",
            "pedestal" : 0.0,
            "sigma"    : 0.25,
            "units"    : "pedesta:e/pixel;img_size:pixel;min_
            "img_size" : 300,
            "min_enlarge":30,
            "active"   : 0
        },
```

deprecated (applies to "in_root")

Output root file naming
out_simulCCDimg_<input_file_name>.root

PROCESS SECTION
Configurable options

```
        "DarkCurrent":
        {
            "mode"        : "cropped-image",
            "darkcurrent" : 0.001,
            "exp_time"    : 0.3333333333,
            "img_size"    : 300,
            "min_enlarge" : 30,
            "units"       : "darkcurrent:e/pixel/day;exp_t
            "active"      : 0
        },
        "PixelSaturation":
        {
            "saturation" : 65535,
            "units"      : "ADC",
            "active"     : 1
        }
    },
    "reconstruction":
    {
        "SignalPatternRecognition":
        {
            "method"    : 1,
            "threshold" : 15,
            "units"     : "ADC",
            "active"    : 0
        },
        "ClusterFinder":
        {
            "method"               : 1,
            "max_nearest_neighbor" : 2,
            "active"               : 1
        }
    },
    "CF":
    {
        "ccd_shape":[1000,4000],
        "ccd_pixel_size_x":0.015,
        "ccd_pixel_size_y":0.015,
        "ccd_thickness":0.675,
        "ADC2eV": 0.72,
        "e2eV"  : 3.77,
        "detector_name":"CCDSensor"
    }
}
```

to define
- CCD size
- pixel size (and binning)
- CCD thickness
- ADC → eV
- mean E to create eh pair
- detector name

How to use WADERS: JSON file example for **psimulCCDimg**

pysimdamicm/json/psimulCCDimg_config_file.json

```
{
    "in_root" :
    {
        "in_dir_path" : ".",
        "root_file_name": "None"
    },
    "out_root":
    {
        "out_dir_path" : ".",
        "prefix_root_name"  : "out_simulCCDimg",
        "store_pixel_info"  : 1,
        "save_CCDimg"       : 0
    },
    "detector_response" :
    {
        "Diffusion":
        {
            "model"     : "gauss",
            "A"         : 216.2,
            "B"         : 0.000886,
            "z_offset"  : 0.0,
            "alpha"     : 0.0000597,
            "units"     : "A:um*um;B:1/um;z_offset:um;alpha:p
            "fanofactor": 0.12,
            "active"    : 1
        },
        "ContinuousReadout":
        {
            "pixel_read_time" : 0.001,
            "units"           : "s/pixel",
            "ampli"           : 4,
            "active"          : 0
        },
        "PixelizeSignal":
        {
            "shift_pixel_in_x" : 0,
            "shift_pixel_in_y" : 0,
            "units"            : "N_pixels_in_x:pixel;pixel_s
            "active"           : 1
        },
        "ElectronicNoise":
        {
            "mode"     : "cropped-image",
            "pedestal" : 0.0,
            "sigma"    : 0.25,
            "units"    : "pedesta:e/pixel;img_size:pixel;min_
            "img_size" : 300,
            "min_enlarge":30,
            "active"   : 0
        },
```

Use “`active`” to switch on/off a process, the process manager will executed secuentially the list of process internally sorted.

```
        "DarkCurrent":
        {
            "mode"        : "cropped-image",
            "darkcurrent" : 0.001,
            "exp_time"    : 0.3333333333,
            "img_size"    : 300,
            "min_enlarge" : 30,
            "units"       : "darkcurrent:e/pixel/day;exp_t
            "active"      : 0
        },
        "PixelSaturation":
        {
            "saturation" : 65535,
            "units"      : "ADC",
            "active"     : 1
        }
    },
    "reconstruction":
    {
        "SignalPatternRecognition":
        {
            "method"    : 1,
            "threshold" : 15,
            "units"     : "ADC",
            "active"    : 0
        },
        "ClusterFinder":
        {
            "method"               : 1,
            "max_nearest_neighbor" : 2,
            "active"               : 1
        }
    },
    "CF":
    {
        "ccd_shape":[1000,4000],
        "ccd_pixel_size_x":0.015,
        "ccd_pixel_size_y":0.015,
        "ccd_thickness":0.675,
        "ADC2eV": 0.72,
        "e2eV"  : 3.77,
        "detector_name":"CCDSensor"
    }
}
```

How to use WADERS: JSON file example for **panaSKImg**

pysimdamicm/json/panaSKImg_config_file.json

to define
- reading skips axis
- overscan/prescan
- skip/col/row to ignore
- set setup-dependent

mandatory! (for "Nskips", "Ncols", "Nrows")

```json
{
    "input":
    {
        "image":
        {
            "extensions" :0,
            "skip_image":true,
            "axis_to_compress":1,
            "correct_leach_bug":true,
            "correct_polarity":false,
            "id_skip_start":3,
            "id_skip_end":-1,
            "id_row_start":0,
            "id_row_end":-1,
            "id_col_start":0,
            "id_col_end":-1,
            "n_rows_overscan":0,
            "n_rows_prescan" :0,
            "n_cols_overscan":15,
            "n_cols_prescan" :2,
            "active_region_rows":null,
            "active_region_cols":null
            },
        "scp":
        {
            },
        "convention":
        {
            "Nskips":"NDCMS",
            "Ncols":"NAXIS1",
            "Nrows":"NAXIS2",
            "Npbin":"NPBIN",
            "Nsbin":"NSBIN",
            "ampl":"AMPL",
            "exposure_time":"MEXP",
            "read_time":"MREAD"
            }
    },
    "process":
    {
        "sequence":"CompressSkipperProcess;RNvsNskipsPlot",
        "CompressSkipperProcess":
        {
            "func_to_compress":["mean"]
        },
        "PedestalSubtractionProcess":
        {
            "image":"mean_compressed",
            "method":"gauss_fit",
            "in_overscan":true,
            "use_mad":false,
            "axis":"row",
            "n_sigma_win_fit":3,
            "n_sigma_to_mask":-1,
            "show_fit":false,
            "histequ":false
        },
        "CalibrationProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "gain":5.3
        },
        "FitCalibrationConstant":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "n_peaks":3,
            "calibration":5,
            "n_sigma_win_fit":3
        },
        "FitDarkCurrentProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "method":"root",
            "do_calibration":true,
            "n_peaks":2,
            "n_sigma_fit":2,
            "mu_gauss":0.0,
            "sigma_gauss":0.2,
            "lambda_poisson":0.05,
            "binning_size":0.25,
            "fit_options":"QS"
        },
        "ChargeLossPlot":
        {
            "skip_id_list":[0,1,2],
            "skip_id_baseline":-1,
            "histequ":false,
            "gray_palette":false
        },
        "FFTNoisePlot":
        {
        },
        "RNvsNskipsPlot":
        {
            "n_skips_per_block":10,
            "is_blank":false
        },

        "SignalPatternRecognition":
        {
            "method"    : 1,
            "image"     : "mean_compressed_pedestal_subtracted_e",
            "isdata"    : true,
            "mask"      : true,
            "threshold" : [null,3,4]
        },
        "ClusterFinder":
        {
            "method"              : 1,
            "max_nearest_neighbor" : 1
        }
    }
}
```

How to use WADERS: JSON file example for **panaSKImg**

pysimdamicm/json/panaSKImg_config_file.json

Corrections to be applied to data before processing
Invert polarity: max → min
leach_bug → for COMPTON setup

```json
{
    "input":
    {
        "image":
        {
            "extensions" :0,
            "skip_image":true,
            "axis_to_compress":1,
            "correct_leach_bug":true,
            "correct_polarity":false,
            "id_skip_start":3,
            "id_skip_end":-1,
            "id_row_start":0,
            "id_row_end":-1,
            "id_col_start":0,
            "id_col_end":-1,
            "n_rows_overscan":0,
            "n_rows_prescan" :0,
            "n_cols_overscan":15,
            "n_cols_prescan" :2,
            "active_region_rows":null,
            "active_region_cols":null
            },
        "scp":
        {
            },
        "convention":
        {
            "Nskips":"NDCMS",
            "Ncols":"NAXIS1",
            "Nrows":"NAXIS2",
            "Npbin":"NPBIN",
            "Nsbin":"NSBIN",
            "ampl":"AMPL",
            "exposure_time":"MEXP",
            "read_time":"MREAD"
            }
    },
    "process":
    {
        "sequence":"CompressSkipperProcess;RNvsNskipsPlot",
        "CompressSkipperProcess":
        {
            "func_to_compress":["mean"]
        },
        "PedestalSubtractionProcess":
        {
            "image":"mean_compressed",
            "method":"gauss_fit",
            "in_overscan":true,
            "use_mad":false,
            "axis":"row",
            "n_sigma_win_fit":3,
            "n_sigma_to_mask":-1,
            "show_fit":false,
            "histequ":false
        },
        "CalibrationProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "gain":5.3
        },
        "FitCalibrationConstant":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "n_peaks":3,
            "calibration":5,
            "n_sigma_win_fit":3
        },
        "FitDarkCurrentProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "method":"root",
            "do_calibration":true,
            "n_peaks":2,
            "n_sigma_fit":2,
            "mu_gauss":0.0,
            "sigma_gauss":0.2,
            "lambda_poisson":0.05,
            "binning_size":0.25,
            "fit_options":"QS"
        },
        "ChargeLossPlot":
        {
            "skip_id_list":[0,1,2],
            "skip_id_baseline":-1,
            "histequ":false,
            "gray_palette":false
        },
        "FFTNoisePlot":
        {
        },
        "RNvsNskipsPlot":
        {
            "n_skips_per_block":10,
            "is_blank":false
        },

        "SignalPatternRecognition":
        {
            "method"    : 1,
            "image"     : "mean_compressed_pedestal_subtracted_e",
            "isdata"    : true,
            "mask"      : true,
            "threshold" : [null,3,4]
        },
        "ClusterFinder":
        {
            "method"               : 1,
            "max_nearest_neighbor" : 1
        }
    }
}
```

How to use WADERS: JSON file example for **panaSKImg**

`pysimdamicm/json/panaSKImg_config_file.json`

```
{
    "input":
    {
        "image":
        {
            "extensions" :0,
            "skip_image":true,
            "axis_to_compress":1,
            "correct_leach_bug":true,
            "correct_polarity":false,
            "id_skip_start":3,
            "id_skip_end":-1,
            "id_row_start":0,
            "id_row_end":-1,
            "id_col_start":0,
            "id_col_end":-1,
            "n_rows_overscan":0,
            "n_rows_prescan" :0,
            "n_cols_overscan":15,
            "n_cols_prescan" :2,
            "active_region_rows":null,
            "active_region_cols":null
            },
        "scp":
        {
            },
        "convention":
        {
            "Nskips":"NDCMS",
            "Ncols":"NAXIS1",
            "Nrows":"NAXIS2",
            "Npbin":"NPBIN",
            "Nsbin":"NSBIN",
            "ampl":"AMPL",
            "exposure_time":"MEXP",
            "read_time":"MREAD"
            }
    },
    "process":
    {
        "sequence":"CompressSkipperProcess;RNvsNskipsPlot",
        "CompressSkipperProcess":
        {
            "func_to_compress":["mean"]
        },
        "PedestalSubtractionProcess":
        {
            "image":"mean_compressed",
            "method":"gauss_fit",
            "in_overscan":true,
            "use_mad":false,
            "axis":"row",
            "n_sigma_win_fit":3,
            "n_sigma_to_mask":-1,
            "show_fit":false,
            "histequ":false
        },
```

~~active~~

Only process listed in "**sequence**" will be secuentially executed and in order of appearance!

```
        "CalibrationProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "gain":5.3
        },
        "FitCalibrationConstant":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "n_peaks":3,
            "calibration":5,
            "n_sigma_win_fit":3
        },
        "FitDarkCurrentProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "method":"root",
            "do_calibration":true,
            "n_peaks":2,
            "n_sigma_fit":2,
            "mu_gauss":0.0,
            "sigma_gauss":0.2,
            "lambda_poisson":0.05,
            "binning_size":0.25,
            "fit_options":"QS"
        },
        "ChargeLossPlot":
        {
            "skip_id_list":[0,1,2],
            "skip_id_baseline":-1,
            "histequ":false,
            "gray_palette":false
        },
        "FFTNoisePlot":
        {
        },
        "RNvsNskipsPlot":
        {
            "n_skips_per_block":10,
            "is_blank":false
        },

        "SignalPatternRecognition":
        {
            "method"    : 1,
            "image"     : "mean_compressed_pedestal_subtracted_e",
            "isdata"    : true,
            "mask"      : true,
            "threshold" : [null,3,4]
        },
        "ClusterFinder":
        {
            "method"               : 1,
            "max_nearest_neighbor" : 1
        }
    }
}
```

PROCESS SECTION

Configurable options

How to use WADERS: JSON file example for **panaSKImg**

pysimdamicm/json/panaSKImg_config_file.json

Any of these options (column, row or skips range) can be parameters of any Process
If one process requires a different region, just add the correspondent parameters into its section

```
{
    "input":
    {
        "image":
        {
            "extensions" :0,
            "skip_image":true,
            "axis_to_compress":1,
            "correct_leach_bug":true,
            "correct_polarity":false,
            "id_skip_start":3,
            "id_skip_end":-1,
            "id_row_start":0,
            "id_row_end":-1,
            "id_col_start":0,
            "id_col_end":-1,
            "n_rows_overscan":0,
            "n_rows_prescan" :0,
            "n_cols_overscan":15,
            "n_cols_prescan" :2,
            "active_region_rows":null,
            "active_region_cols":null
            },
        "scp":
        {
            },
        "convention":
        {
            "Nskips":"NDCMS",
            "Ncols":"NAXIS1",
            "Nrows":"NAXIS2",
            "Npbin":"NPBIN",
            "Nsbin":"NSBIN",
            "ampl":"AMPL",
            "exposure_time":"MEXP",
            "read_time":"MREAD"
            }
    },
    "process":
    {
        "sequence":"CompressSkipperProcess;RNvsNskipsPlot",
        "CompressSkipperProcess":
        {
            "func_to_compress":["mean"]
        },
        "PedestalSubtractionProcess":
        {
            "image":"mean_compressed",
            "method":"gauss_fit",
            "in_overscan":true,
            "use_mad":false,
            "axis":"row",
            "n_sigma_win_fit":3,
            "n_sigma_to_mask":-1,
            "show_fit":false,
            "histequ":false
        },
```

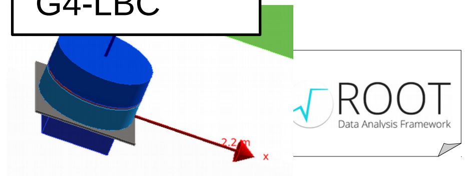
```
        "CalibrationProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "gain":5.3
        },
        "FitCalibrationConstant":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "n_peaks":3,
            "calibration":5,
            "n_sigma_win_fit":3
        },
        "FitDarkCurrentProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "method":"root",
            "do_calibration":true,
            "n_peaks":2,
            "n_sigma_fit":2,
            "mu_gauss":0.0,
            "sigma_gauss":0.2,
            "lambda_poisson":0.05,
            "binning_size":0.25,
            "fit_options":"QS"
        },
        "ChargeLossPlot":
        {
            "skip_id_list":[0,1,2],
            "skip_id_baseline":-1,
            "histequ":false,
            "gray_palette":false
        },
        "FFTNoisePlot":
        {
        },
        "RNvsNskipsPlot":
        {
            "n_skips_per_block":10,
            "is_blank":false
        },

        "SignalPatternRecognition":
        {
            "method"    : 1,
            "image"     : "mean_compressed_pedestal_subtracted_e",
            "isdata"    : true,
            "mask"      : true,
            "threshold" : [null,3,4]
        },
        "ClusterFinder":
        {
            "method"               : 1,
            "max_nearest_neighbor" : 1
        }
    }
}
```

How to use WADERS: running **psimulCCDimg**

For simulations

G4-LBC



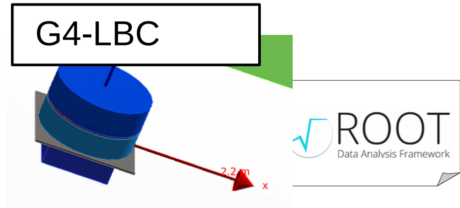

```
psimulCCDimg   psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
```

MANDATORY Configuration JSON file

https://ncastell.web.cern.ch/ncastell/pysimdamicm/howtouse.html#

How to use WADERS: running **psimulCCDimg**

For simulations



DAMICG4 (or compatible)

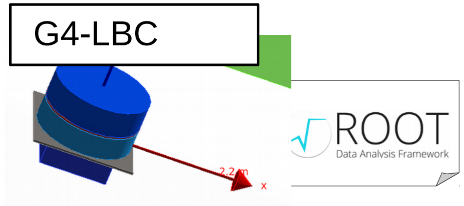
```
psimulCCDimg   psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
```

[absolute/relative path] input ROOT file

https://ncastell.web.cern.ch/ncastell/pysimdamicm/howtouse.html#

How to use WADERS: running **psimulCCDimg**

For simulations




```
psimulCCDimg   psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
```

absolute or relative path for the outputs
(automatic naming for the ROOT file)

<prefix_from_json>_log**ab87863**_input_file_name.root

```
commit 526863b
Author: Nuria Castello Mor <nuria.castello.mor@cern.ch>
Date:   Fri Jan 8 14:20:06 2021 +0100

    Updapte uproot required version

commit ab87863
Author: Nuria Castello Mor <nuria.castello.mor@cern.ch>
Date:   Fri Jan 8 11:54:02 2021 +0100

    Fix some requirements
```

How to use WADERS: running **psimulCCDimg**

For simulations

```
psimulCCDimg    psimul                              /DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
```

```
>>> d.detector_config.Show(0)
======> EVENT:0
 Diffusion_A      = 0.0002162
 Diffusion_B      = 0.886
 Diffusion_z_offset = 0
 Diffusion_fanofactor = 0.16
 Diffusion_alpha = 5.97e-05
 Diffusion_z_max = 0.669
 PixelizeSignal_shift_pixel_in_x = 0
 PixelizeSignal_N_pixels_in_x = 4000
 PixelizeSignal_pixel_size_in_x = 0.015
 PixelizeSignal_shift_pixel_in_y = 0
 PixelizeSignal_N_pixels_in_y = 6000
 PixelizeSignal_pixel_size_in_y = 0.015
 DarkCurrent_full_image_counter = 0
 DarkCurrent_darkcurrent = 1.15741e-08
 DarkCurrent_exp_time = 28800
 DarkCurrent_img_size = 300
 DarkCurrent_min_enlarge = 30
 ElectronicNoise_full_image_counter = 0
 ElectronicNoise_nSamples = 1
 ElectronicNoise_sigma = 0.25
 ElectronicNoise_pedestal = 0
 ElectronicNoise_img_size = 300
 ElectronicNoise_min_enlarge = 30
 SignalPatternRecognition_threshold = 30
 SignalPatternRecognition_method = 1
 ClusterFinder_method = 1
 ClusterFinder_max_nearest_neighbor = 2
 e2eV             = 3.77
 ADC2eV           = 0.72
```

Detailed documentation for the output ROOT file at https://ncastell.web.cern.ch/ncastell/pysimdamicm/howtouse.html#

PyDAMICM

Search docs

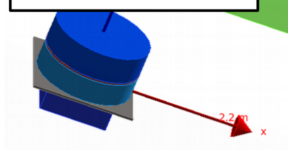
⊟ **Output ROOT file**

- detector_config TTree — Parameters for the simulated process with their configuration values
- geant4_config TTree — Some parameters from the geant4 ROOT file: properties of the simulated volumes, simulated SS [CCDs] simulated PP, ...
- pixelizedEvent TTree — Properties of the pixel (evt,ccd,x,y,z,pdg,Edep, ...)
- clustersRec TTree — Properties of all found clusters (evt,ccd,x,y,z,Energy,Qmax,...)

How to use WADERS: running **psimulCCDimg**

For simulations

G4-LBC



ROOT
Data Analysis Framework

```
>>> d.geant4_config.Show(0)
======> EVENT:0
 NEvts             = 30000
 NCCDs             = 2
 Seed              = 722
 primary           = ion
 ion               = 60a27z
 ccd_mass          = 0.0169857
 ccd_vol           = 7.29
 ccd_density       = 2.33
 ccd_surface       = 220.05
 sim_vol_pvnames = KConccd_PV
 n_vols            = 1
 sim_vol_density = 3.27918
 sim_vol_mass      = 0.00580722
 sim_vol_volume    = 1.77094
 sim_vol_surface = 121.256
 Nclusters         = 542
```

```
psimulCCDimg    psimulCCDim ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
```

Detailed documentation for the output ROOT file at https://ncastell.web.cern.ch/ncastell/pysimdamicm/howtouse.html#

PyDAMICM

Search docs

⊟ Output ROOT file

- detector_config TTree — Parameters for the simulated process with their configuration values
- geant4_config TTree — Some parameters from the geant4 ROOT file: properties of the simulated volumes, simulated SS [CCDs] simulated PP, ...
- pixelizedEvent TTree — Properties of the pixel (evt,ccd,x,y,z,pdg,Edep, ...)
- clustersRec TTree — Properties of all found clusters (evt,ccd,x,y,z,Energy,Qmax,...)

How to use WADERS: running **psimulCCDimg**

For simulations

```
>>> d.pixelizedEvent.Show(0)
======> EVENT:0
 event              = 5
 Npix               = (vector<int>*)0x3b45a10
 ccd                = (vector<int>*)0x3b46500
 pixels_Edep        = (vector<vector<float> >*)0x3ac4860
 pixels_N_carried_charges = (vector<vector<int> >*)0x3ac5b00
 pixels_pdg         = (vector<vector<float> >*)0x3a84290
 pixels_sigma_xy = (vector<vector<float> >*)0x3ac6f20
 pixels_time        = (vector<vector<float> >*)0x3ea6120
 pixels_x           = (vector<vector<int> >*)0x3ea4080
 pixels_y           = (vector<vector<int> >*)0x3ea3350
 pixels_z           = (vector<vector<float> >*)0x3ecfd50
 pp_energy          = (vector<vector<float> >*)0x3ea3f70
 pp_momx            = (vector<vector<float> >*)0x3ea0d00
 pp_momy            = (vector<vector<float> >*)0x3b0b280
 pp_momz            = (vector<vector<float> >*)0x208b5b0
 pp_posx            = (vector<vector<float> >*)0x3f560f0
 pp_posy            = (vector<vector<float> >*)0x3e9cc70
 pp_posz            = (vector<vector<float> >*)0x3f16e30
```

`psimulCCDimg psim` … `oot --g4out ../outputs`

Detailed documentation for the output ROOT file at https://ncastell.web.cern.ch/ncastell/pysimdamicm/howtouse.html#

PyDAMICM

Search docs

Output ROOT file

- detector_config TTree — Parameters for the simulated process with their configuration values
- geant4_config TTree — Some parameters from the geant4 ROOT file: properties of the simulated volumes, simulated SS [CCDs] simulated PP, ...
- pixelizedEvent TTree — Properties of the pixel (evt,ccd,x,y,z,pdg,Edep, ...)
- clustersRec TTree — Properties of all found clusters (evt,ccd,x,y,z,Energy,Qmax,...)

How to use WADERS: running **psimulCCDimg**

For simulations

psimulCCDimg psim 1Am.root --g4out ../outputs

```
>>> d.clustersRec.Show(0)
======> EVENT:0
 event             = 5
 Nclusters         = 1
 DX                = (vector<float>*)0x3695cf0
 DY                = (vector<float>*)0x36e5700
 Energy            = (vector<float>*)0x2532a60
 Npix              = (vector<int>*)0x367f050
 PosX              = (vector<float>*)0x36f9a50
 PosY              = (vector<float>*)0x36da780
 Qmax              = (vector<float>*)0x36d9f90
 QmaxX             = (vector<float>*)0x368c9f0
 QmaxY             = (vector<float>*)0x367ef10
 RMSX              = (vector<float>*)0x2531ec0
 RMSY              = (vector<float>*)0x3690810
 ccd               = (vector<int>*)0x36916e0
 cluster_id        = (vector<int>*)0x36912c0
 maxX              = (vector<float>*)0x36f6e90
 maxY              = (vector<float>*)0x36e4bb0
 meanX             = (vector<float>*)0x3684730
 meanY             = (vector<float>*)0x3681c20
 minX              = (vector<float>*)0x37e1820
 minY              = (vector<float>*)0x36ea8b0
 pixels_E          = (vector<vector<float> >*)0x2ecf670
 pixels_time       = (vector<vector<float> >*)0x3a84370
 pixels_x          = (vector<vector<float> >*)0x3a8f800
 pixels_y          = (vector<vector<float> >*)0x3a29160
 pixels_z          = (vector<vector<float> >*)0x3ab2370
 pp_energy         = (vector<vector<float> >*)0x3ab2d30
 pp_momx           = (vector<vector<float> >*)0x3a8b390
 pp_momy           = (vector<vector<float> >*)0x3acc0c0
 pp_momz           = (vector<vector<float> >*)0x3acb4f0
 pp_posx           = (vector<vector<float> >*)0x3aa85d0
 pp_posy           = (vector<vector<float> >*)0x3a727c0
 pp_posz           = (vector<vector<float> >*)0x3a61820
 primary_part      = (vector<vector<float> >*)0x36edfb0
```

Detailed documentat stell/pysimdamicm/howtouse.html#

PyDAMI

Search docs

⊟ Output ROOT fi

detector_config uration values

geant4_config s of the simulated volumes, simulated SS [CCDs]

pixelizedEvent simulated PP, ...

clustersRec TTree — Properties of all found clusters (evt,ccd,x,y,z,Energy,Qmax,...)

How to use WADERS: running **psimulCCDimg**

For simulations

expected ouput

```
WARNING: Parameter <PixelSaturation.units> will be ignored (Not implemented)
WARNING: Parameter <ContinuousReadout.units> will be ignored (Not implemented)
WARNING: Parameter <ccd_pixel_size_y> will be ignored (Not implemented)
Config.activate_configuration --- Add Diffusion process to the simulated process chain.
Config.activate_configuration --- Add PixelizeSignal process to the simulated process chain.
Config.activate_configuration --- Add ClusterFinder process to the simulated process chain.

-- INFO. Input root file:  /pbs/home/n/ncastell/repos/pysimdamicm/notebooks/software_school_2021/
-- INFO. Output root file:  /pbs/home/n/ncastell/repos/pysimdamicm/notebooks/software_school_202
Setting value of  ccd_shape  to  [4000, 6000]
Setting value of  ccd_pixel_size_x  to  0.015
Setting value of  ccd_pixel_size_y  to  0.015
Setting value of  ccd_thickness  to  0.675
Setting value of  e2eV  to  3.77
Setting value of  ADC2eV  to  0.72
Setting value of  detector_name  to  CCDSensor_PV
main INFO. Process sequence (objects):
 [<pysimdamicm.processes.detector_response.Diffusion object at 0x7f890b7fae80>, <pysimdamicm.proce
onstruction.ClusterFinder object at 0x7f8957aa1b00>]


--- Loading geant4 data
     * CCDOut loaded (2.3 sec),
     * EventOut loaded (0.22 sec)

--- Reconstruction process starts:
     * 431/30000 events  to process, 1.437%


.......................................... event 5 (0% processed)

.......................................... event 3369 (10% processed)

.......................................... event 5669 (20% processed)

.......................................... event 8793 (30% processed)

.......................................... event 11908 (40% processed)

.......................................... event 15804 (50% processed)

.......................................... event 17936 (60% processed)

.......................................... event 20493 (70% processed)

.............................

.............................
Time statistics:
 - Reconstruction processes d

-- Found volume  CCDSensor_PV
-- Found volume  KConccd_PV

Output Root file: /pbs/home/n/ncastell/repos/pysimdamicm/notebooks/software_school_2021/ex2_3/60Co


 ------------- Done!
```

```
-- Not found volume  CCDSensor
-- Found volume  KConccd_PV
```

reading from JSON file

reading input ROOT

reading input ROOT

reconstruction starts

...

Extracting info from:
Sensitive detector
where PP were placed (if used)

...

How to use WADERS: running **psimulCCDimg**

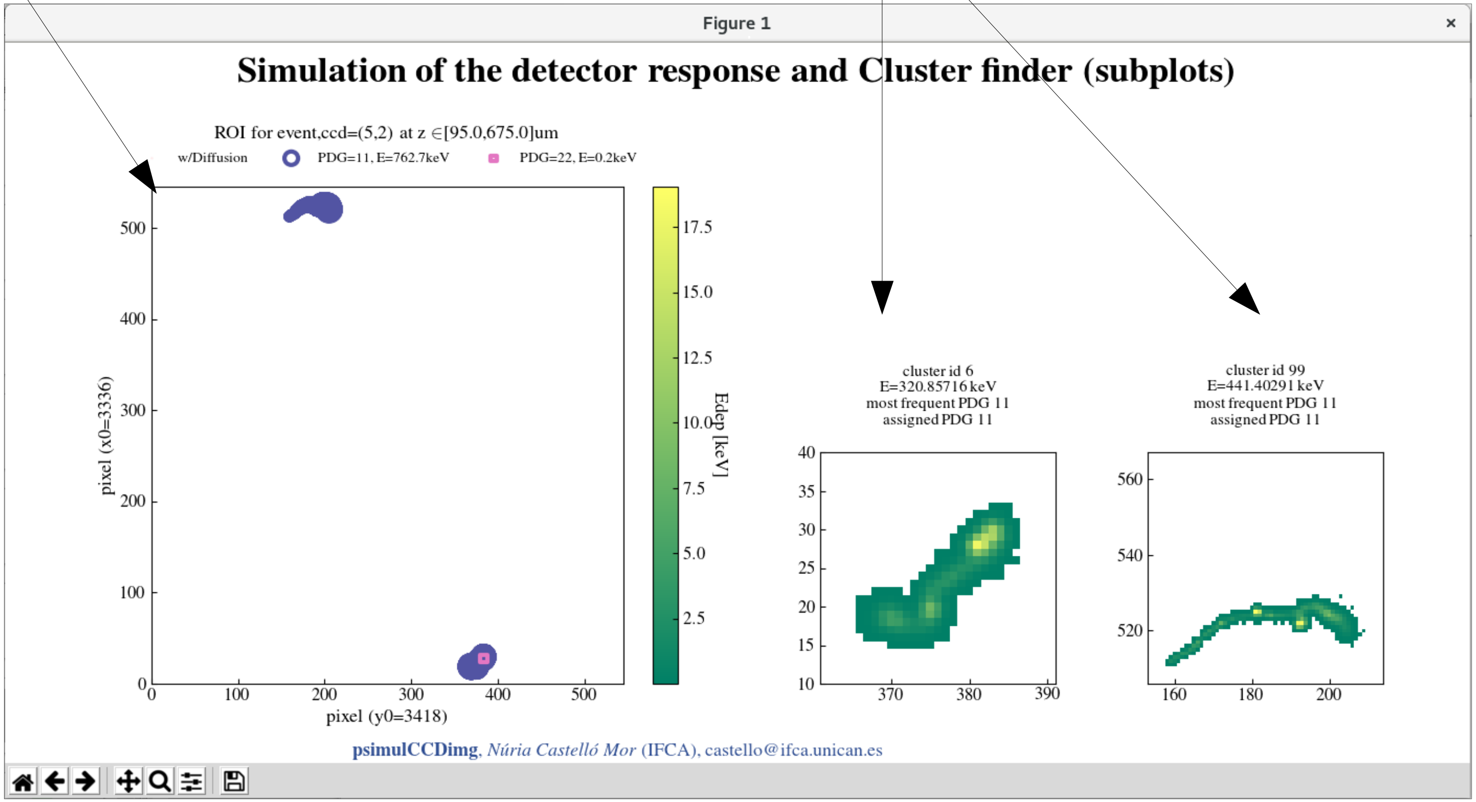
For simulations

debug mode

```
psimulCCDimg   psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
         --debug      --event 5    --cls-pix-min 3
```

all energy depositions in CCD id 2
event with id number 5

zoom into
clusters found with $N_{pix} > 3$



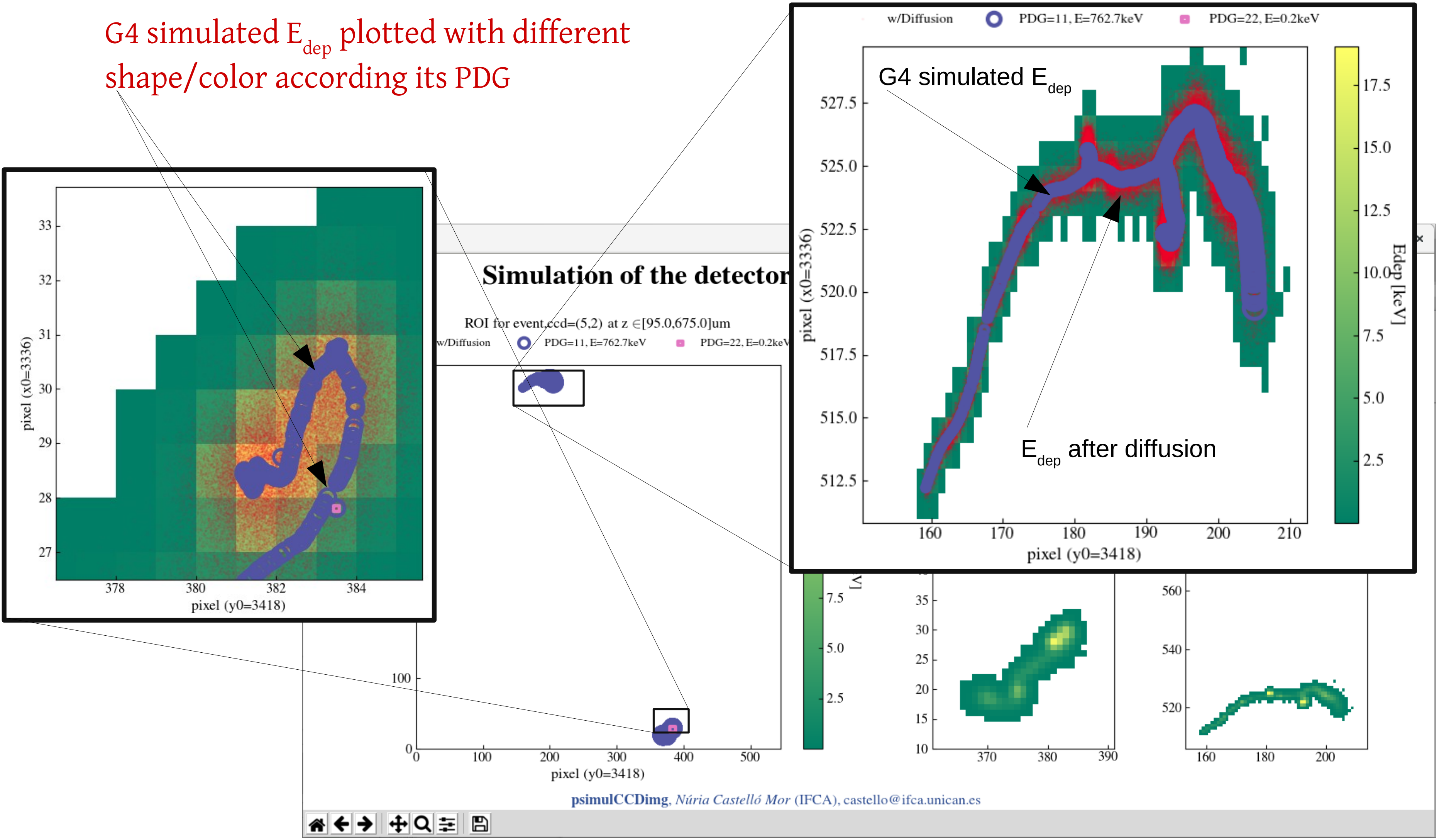
How to use WADERS: running **psimulCCDimg**

For simulations

debug mode

```
psimulCCDimg   psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
                 --debug     --event 5   --cls-pix-min 3
```

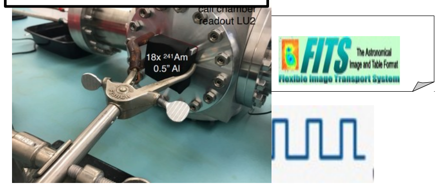
G4 simulated E_{dep} plotted with different shape/color according its PDG



How to use WADERS: running **panaSKImg**

For skipper CCD images

Compton setup




```
panaSKImg   --json panaSKImg_config_compton.json “/data/compton/data/*Image*Source*fits” --skip _598 full_1.fits -o .
```

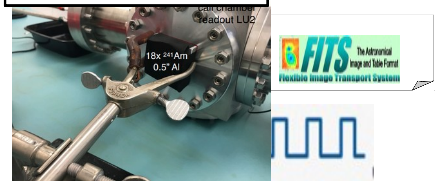
configuration JSON file

https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_panaSKImg.html

How to use WADERS: running **panaSKImg**

For skipper CCD images

Compton setup




```
panaSKImg   --json panaSKImg_config_compton.json “/data/compton/data/*Image*Source*fits” --skip _598 full_1.fits -o .
```

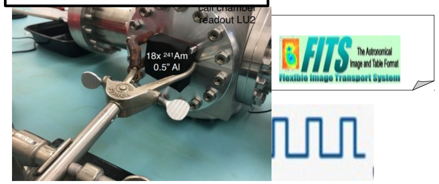
mandatory input file(s)
(accept wildcards, you need “”)

https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_panaSKImg.html

How to use WADERS: running **panaSKImg**

For skipper CCD images

Compton setup




```
panaSKImg   --json panaSKImg_config_compton.json “/data/compton/data/*Image*Source*fits” --skip _598 full_1.fits -o .
```

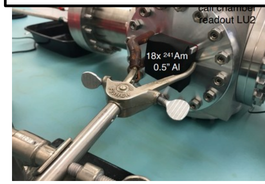
Image(s) to be ignore
(those containing this pattern)

https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_panaSKImg.html

How to use WADERS: running **panaSKImg**

For skipper CCD images

Compton setup



```
panaSKImg   --json panaSKImg_config_compton.json “/data/compton/data/*Image*Source*fits” --skip _598 full_1.fits -o .
```

output directory

automatic naming for output plots and images

(using input file name, process class, sub-string if needed)

https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_panaSKImg.html

How to use WADERS: running **panaSKImg**

For skipper CCD images

```
panaSKImg   --json panaSKImg_config_compton.json "/data/compton/data/*Image*Source*fits" --skip _598 full_1.fits -o .
            --save-plots --save-img  --display --verbose
            -s ChargeLossPlot,CompressSkipperProcess,PedestalSubtractionProcess,SignalPatternRecognition,ClusterFinder
            ...
```

```
(venv_software_school) /(0)>panaSKImg --help
usage: panaSKImg [-h] [--skip SKIP [SKIP ...]] [-o OUTPUT] [-e EXTENSION]
                 [-j JSONFILE] [--mask MASK] [--dqm]
                 [--dqm-data-dir DQM_DATA_DIR] [--image-HR IMAGE_HR]
                 [--run RUN] [--me-ref ME_REF] [--all-me] [--invert]
                 [--skip-start ID_SKIP_START] [--skip-end ID_SKIP_END]
                 [--row-start ID_ROW_START] [--row-end ID_ROW_END]
                 [--col-start ID_COL_START] [--col-end ID_COL_END]
                 [-s SEQUENCE] [--list-processes] [--save-img] [--save-plots]
                 [--display] [--verbose] [--cal CALIBRATION]
                 [--func-to-compress FUNC_TO_COMPRESS [FUNC_TO_COMPRESS ...]]
                 [--method METHOD] [--in-overscan] [--axis AXIS]
                 [--n-sigma-win-fit N_SIGMA_WIN_FIT]
                 [--n-sigma-to-mask N_SIGMA_TO_MASK] [--show-fit]
                 [--skip-id-list SKIP_ID_LIST [SKIP_ID_LIST ...]]
                 [--skip-id-baseline SKIP_ID_BASELINE] [--histequ]
                 [--gray-palette] [--n-skips N_SKIPS_PER_BLOCK] [--is-blank]
                 [--n-peaks N_PEAKS] [--calibration CALIBRATION]
                 [--dc-axis DC_AXIS] [--n-elec N_ELEC]
                 [--n-sigma-fit N_SIGMA_FIT] [--mu-gauss MU_GAUSS]
                 [--sigma-gauss SIGMA_GAUSS] [--lambda-poisson LAMBDA_POISSON]
                 [--fit-options FIT_OPTIONS] [--do-calibration]
                 infile

positional arguments:
  infile                Input CCD Image or a pattern file name for multiple
                        inputs, in this case use "" to quote the expression.
                        If extension is not 0, see -e
```

Useful command lines

- to store plots
- to store images
- to display the final plots before ending
- to display intermediate plots and print more msm
- to set the sequence of process

All command lines has preference over the JSON file values

https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_panaSKImg.html

How to use WADERS: summary

```
psimulCCDimg psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
                              --event 5 --cls-pix-min 5
```

executable **JSON config file** **input file[s]** **output directory**

```
panaSKImg --json panaSKImg_compton_config.json "/data/compton/data/*Image*Source*fits" -o ../Run145/avg
            --skip _598 full_1.fits --save-plots --save-img --verbose
            -s CompressedSkipperProcess,PedestalSubtractionProcess,SignalPatterRecognition,ClusterFinder
            --skip-start 3 --skip-end 1569 --func-to-compress mean
```

How to use WADERS: summary

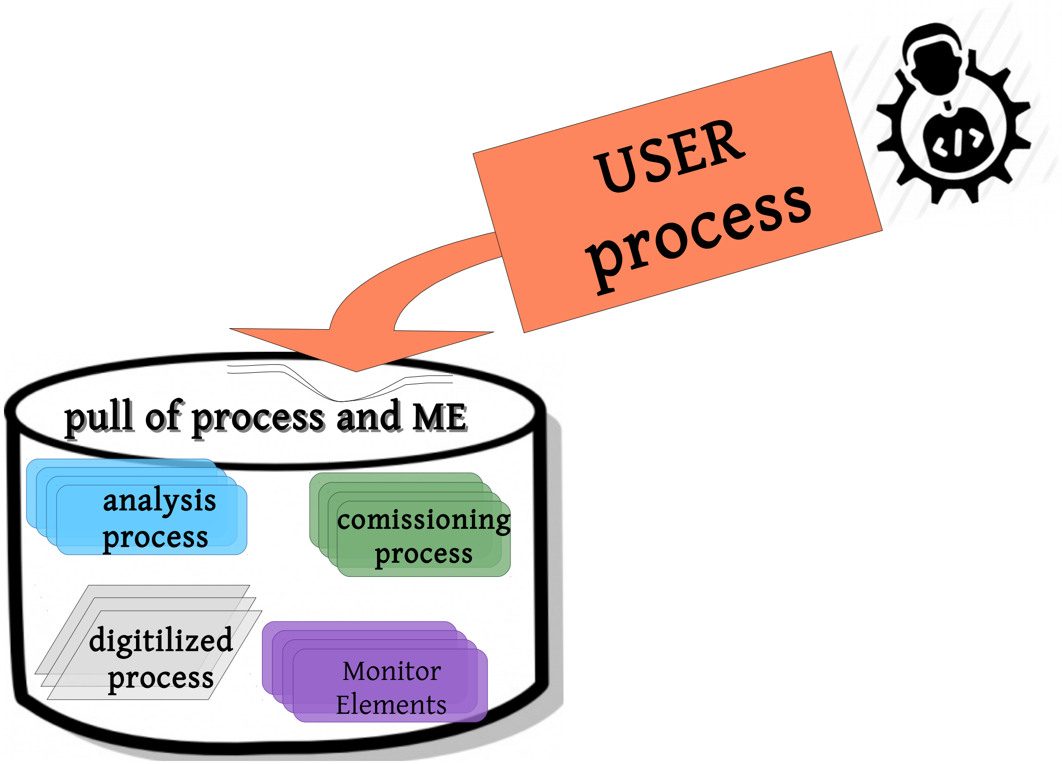
```
psimulCCDimg   psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
                        --debug --event 5 --cls-pix-min 5
```

Optional command lines related to debug mode

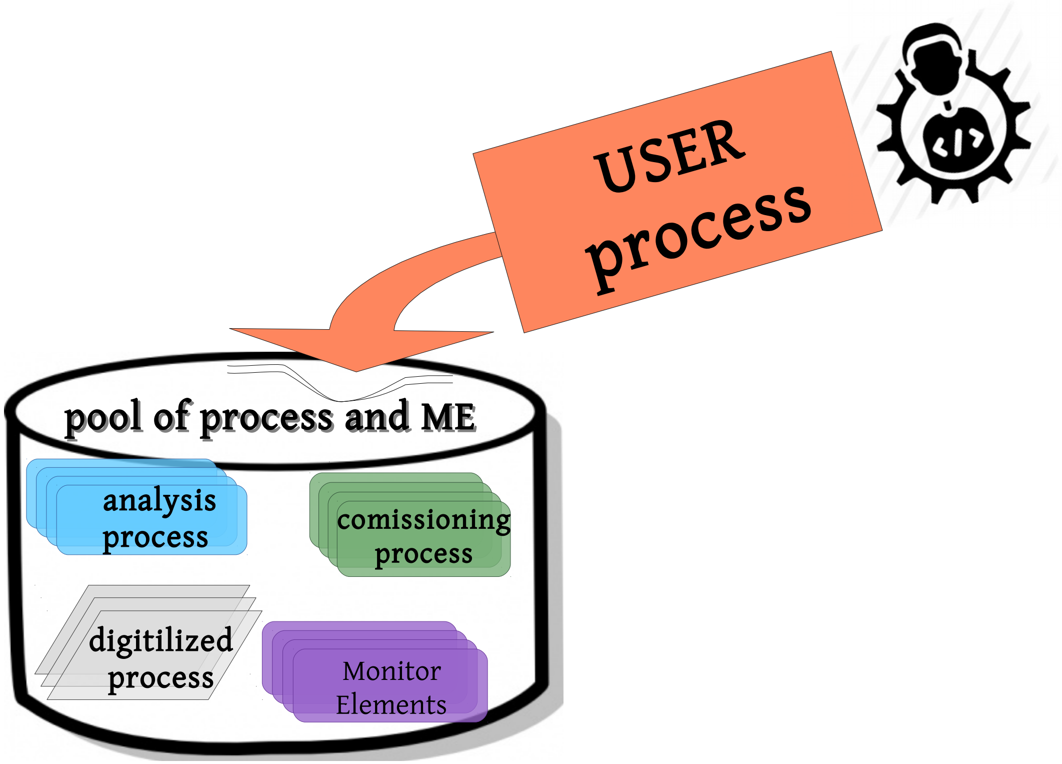
```
panaSKImg --json panaSKImg_compton_config.json "/data/compton/data/*Image*Source*fits" -o ../Run145/avg
            --skip _598 full_1.fits --save-plots --save-img --verbose
            -s CompressedSkipperProcess,PedestalSubtractionProcess,SignalPatterRecognition,ClusterFinder
            --skip-start 3 --skip-end 1569 --func-to-compress mean
            ...
```

A lot of optional command lines
→ mostly to launch without changing the JSON file
So far, all configurable parameters have a command line which has preference over the json file one

For developers only



Do you need a process that is not available?



Extending the library of process

Do you need a process that is not available?

WADERS allows any new algorithm to be added as a new process

But before ....

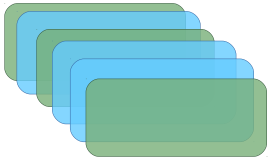
A few words on the operational outline

WADERS: operational outline

Input file encapsulated into runtime containers

RawData	**G4HitCollection**
.header	.x
.ncols	.y
.nskips	.z
.image	.pdg
...	...
.image	.Edep

the data we want to manipulate



sequence, set of processes
→ sorted and
→ configured
by the **Process Manager**

[JSON]
configuration

WADERS: operational outline

Input file encapsulated into runtime containers

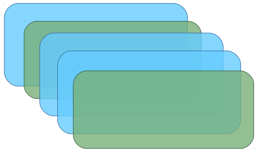
RawData	**G4HitCollection**
.header	.x
.ncols	.y
.nskips	.z
.image	.pdg
...	...
.image	.Edep

the data we want to manipulate

Process Manager

process1
execute_process

.image_mean_compressed	.Edep
create plot	.x
.A	.y
	.z



sequence, set of processes
→ sorted and
→ configured
by the Process Manager

[JSON]
configuration

WADERS: operational outline

Input file encapsulated into runtime containers

RawData
.header
.ncols
.nskips
.image
...
.image

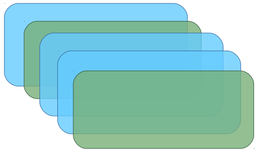
G4HitCollection
.x
.y
.z
.pdg
...
.Edep

the data we want to manipulate

Process Manager
process1
execute_process

.image_mean_compressed
.gain

.Edep
.x
.y
.z



sequence, set of processes
→ sorted and
→ configured
by the Process Manager

[JSON]
configuration

or maniupate existing ones

create new data members
from existing ones [or news]

WADERS: operational outline

Input file encapsulated into runtime containers

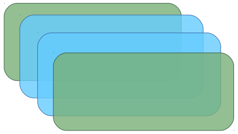
RawData	**G4HitCollection**
.header	.x
.ncols	.y
.nskips	.z
.image	.pdg
...	...
.image	.Edep

the data we want to manipulate

Process Manager

process1
execute_process

.image_mean_compressed	.Edep .x
.gain	.y .z



sequence, set of processes
→ sorted and
→ configured
by the Process Manager

[JSON]
configuration

process2
execute_process

create some plots(rawdata.gain)	**.pixel_Edep** **.pixel_x** **.pixel_y** **.pixel_z** ...

Goes to plots container to be displayed at the end (if booked)

WADERS: operational outline

Input file encapsulated into runtime containers

RawData	**G4HitCollection**
.header	.x
.ncols	.y
.nskips	.z
.image	.pdg
...	...
.image	.Edep

the data we want to manipulate

Process Manager

process1
execute_process

.image_mean_compressed

.gain

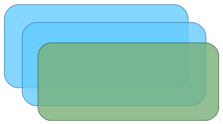
.Edep
.x
.y
.z

process2
execute_process

create some plots(rawdata.gain)

.pixel_Edep
.pixel_x
.pixel_y
.pixel_z
...

. . . .



sequence, set of processes
→ sorted and
→ configured
by the Process Manager

[JSON]
configuration

WADERS: operational outline

Input file encapsulated into runtime containers

RawData	**G4HitCollection**
.header	.x
.ncols	.y
.nskips	.z
.image	.pdg
...	...
.image	.Edep

the data we want to manipulate

Process Manager

process1
execute_process

- .image_mean_compressed
- .gain
- .Edep
- .x
- .y
- .z

process2
execute_process

- **create some plots(rawdata.gain)**
- **.pixel_Edep**
- **.pixel_x**
- **.pixel_y**
- **.pixel_z**
- ...

....

sequence, set of processes
→ sorted and
→ configured
by the Process Manager

[JSON]
configuration

WADERS: operational outline

Input file encapsulated into runtime containers

RawData
.header
.ncols
.nskips
.image
...
.image

G4HitCollection
.x
.y
.z
.pdg
...
.Edep

the data we want to manipulate

Process Manager

process1
execute_process

.image_mean_compressed
.gain

.Edep
.x
.y
.z

process2
execute_process

create some plots(rawdata.gain)

.pixel_Edep
.pixel_x
.pixel_y
.pixel_z
...

....

sequence, set of processes
→ sorted and
→ configured
by the Process Manager

[JSON]
configuration

WADERS: operational outline

Input file encapsulated into runtime containers

RawData	**G4HitCollection**
.header	.x
.ncols	.y
.nskips	.z
.image	.pdg
...	...
.image	.Edep

the data we want to manipulate

Process Manager

process1
execute_process

- **.image_mean_compressed**
- **.gain**
- .Edep
- .x
- .y
- .z

process2
execute_process

- **create some plots(rawdata.gain)**
- **.pixel_Edep**
- **.pixel_x**
- **.pixel_y**
- **.pixel_z**
- ...

....

processN
execute_process

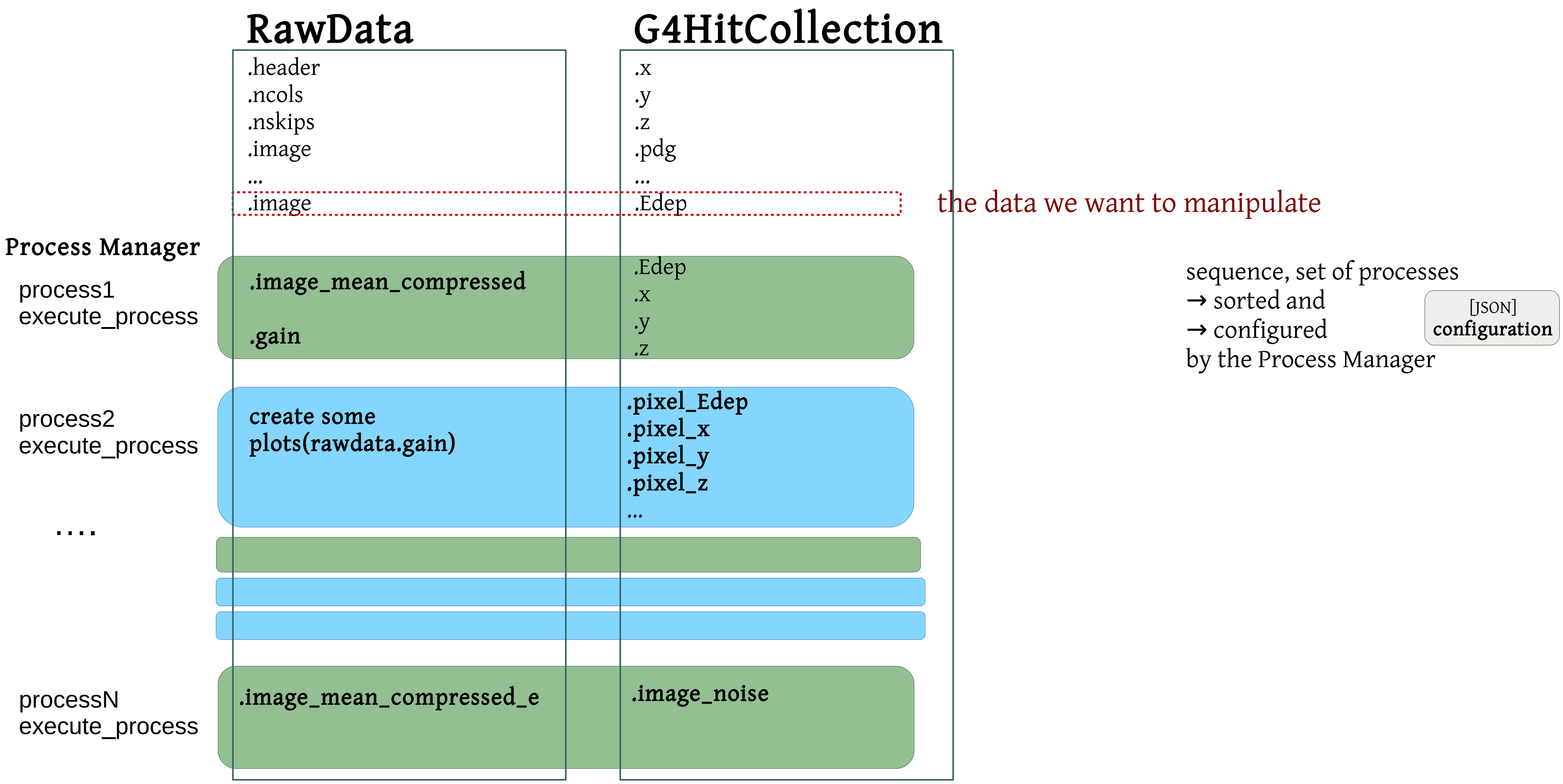
- **.image_mean_compressed_e**
- **.image_noise**

sequence, set of processes
→ sorted and
→ configured
by the Process Manager

[JSON]
configuration

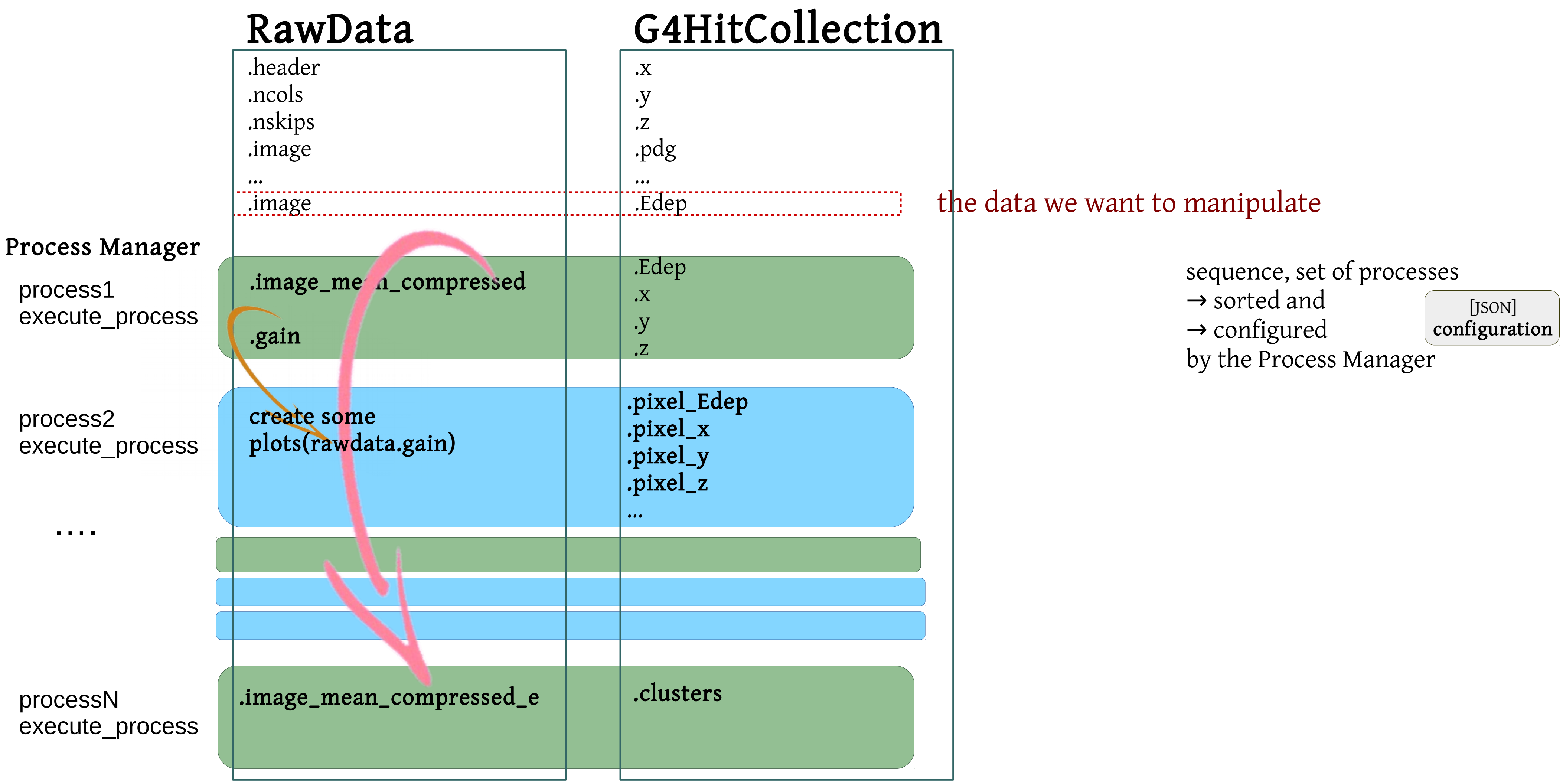
WADERS: operational outline

Input file encapsulated into runtime containers



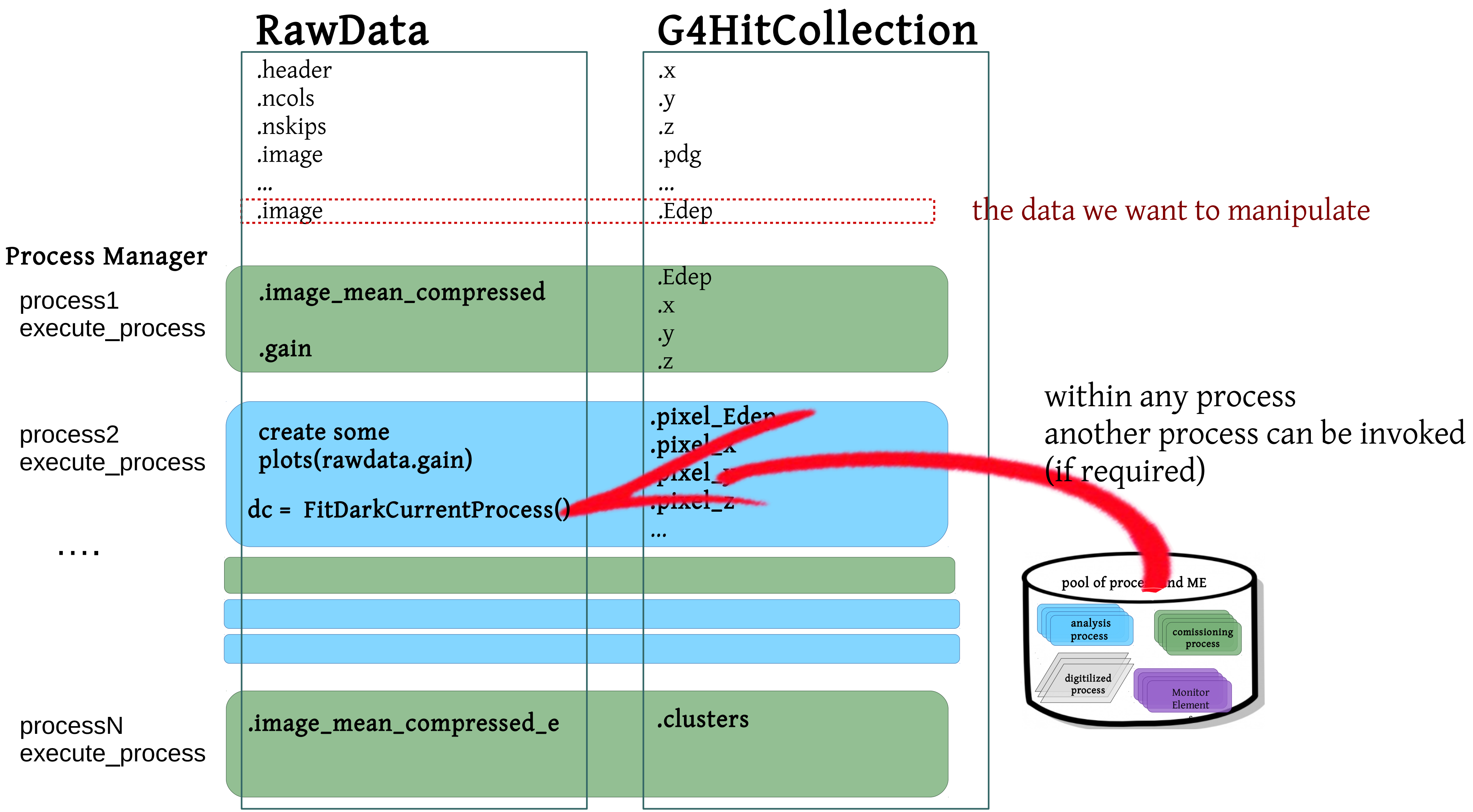
WADERS: operational outline

Input file encapsulated into runtime containers



WADERS: operational outline

Input file encapsulated into runtime containers



Be very careful: a process is an algorithm with a single objective, If this objective is the product of a two-process sequence means your process is not well define within this framework!!!

Extending the library of process

Do you need a process that is not available?

WADERS allows any new algorithm to be added as a new process

How to

Extending the library of process

Do you need a process that is not available?

WADERS allows any new algorithm to be added as a new process

An extension of an available process? Or new?

Abstract class

…the working scope, the **processes** are grouped

Abstract class	
DigitizeProcess	**detector response:** to digitilize the geant4 monte carlo simulations (which have to be compatible with the one from DAMICG4) pysimdamicm/processes/detector_response.py
SKImageProcess	**comissioning:** for any [skipper] CCD image pysimdamicm/processes/skipper_comissioning.py
SKImageProcess	**analysis:** for any [skipper] CCD image pysimdamicm/processes/reconstruction.py pysimdamicm/processes/skipper_analysis.py
MEabs	**Monitor Element:** a plot or quantity as data quality of your data taking pysimdamicm/dqm/me.py

Visit the *Compton Setup web site* https://ncastell.web.cern.ch/ncastell/compton_web_site/
to see the full set of available ME 18

Extending the library of process: how to

Building a new process

It has **mandatory ingredients**

- (class attribute name) string uniquely identifying the process
- (class attribute sequence id) integer defining the per default order in the sequence of the process to be executed
- any parameter to be configured from the json file must be added to the object dictionary `__units__`
- the algorithm must be implemented in the method `execute_process(self,rawdata)`
- The process does not return nothing:
  - any needed parameter must be appended into the rawdata object
- The *automatic plug-in mechanism* is still work in progress. So far, the process manager must be informed ad hoc

  add the process in the `ProcessManager.__valid_processes__` class attribute

must inherit from abstract class
DigitizeProcess, SKImageProcess or MEabs

```
class NameOfThePROCESS(ABSTRACTCLASSNAME):
    """DOCUMENT IS REALLY IMPORTANT!!

        blablabla blablalbalbalbla

          xx = y


        Attributes
        ----------
            A :: int
                blab

    """
    __sequence_id__ = 1111
    __name__ = 'NameOfThePROCESS'

    def __init__(self):
        """
        """
        super().__init__()

        self.A

        ###
        self.__units__.update({'A':u.ADC/u.pixel})

    def execute_process(self,rawdata):
        """
        """
        print("Process <NameOfThePROCESS> INFO. Bla bla bla bla usin ")
        print("  using {} ADC/pix".format(self.A))

        #### CODE HERE YOUR ALGORITHM
        ################################################################################

        # get image

        # create blab bla bla

        ################################################################################

        #### if required, add attribute to RawData object
        setattr(rawdata,"attr_name",obj_attr_name)
```

Documentation is really important!!

Extending the library of process: how to

Building a new process

must inherit from abstract class
DigitizeProcess, SKImageProcess or MEabs

```python
class NameOfThePROCESS(ABSTRACTCLASSNAME):
    """DOCUMENT IS REALLY IMPORTANT!!

        blablabla blablalbalbalbla


            xx = y


        Attributes
        ----------
            A :: int
                blab

    """
    __sequence_id__ = 1111
    __name__ = 'NameOfThePROCESS'

    def __init__(self):
        """
        """
        super().__init__()

        self.A

        ###
        self.__units__.update({'A':u.ADC/u.pixel})

    def execute_process(self,rawdata):
        """
        """
        print("Process <NameOfThePROCESS> INFO. Bla bla bla bla usin ")
        print("  using {} ADC/pix".format(self.A))

        #### CODE HERE YOUR ALGORITHM
        ############################################################################

        # get image

        # create blab bla bla

        ############################################################################

        #### if required, add attribute to RawData object
        setattr(rawdata,"attr_name",obj_attr_name)
```

Documentation is really important!!

It has **mandatory ingredients**

- (class attribute name) string uniquely identifying the process
- (class attribute sequence id) integer defining the per default order in the sequence of the process to be executed
- any parameter to be configured from the json file must be added to the object dictionary `__units__`
- the algorithm must be implemented in the method `execute_process(self,rawdata)`
- The process does not return nothing:
  - any needed parameter must be appended into the rawdata object
- The *automatic plug-in mechanism* is still work in progress. So far, the process manager must be informed ad hoc

  add the process in the `ProcessManager.__valid_processes__` class attribute

Extending the library of process: how to

Building a new process

must inherit from abstract class
DigitizeProcess, SKImageProcess or MEabs

It has **mandatory ingredients**

- (class attribute name) string uniquely identifying the process
- (class attribute sequence id) integer defining the per default order in the sequence of the process to be executed
- any parameter to be configured from the json file must be added to the object dictionary `__units__`
- the algorithm must be implemented in the method `execute_process(self,rawdata)`
- The process does not return nothing:
  - any needed parameter must be appended into the rawdata object
- The *automatic plug-in mechanism* is still work in progress. So far, the process manager must be informed ad hoc
  add the process in the `ProcessManager.__valid_processes__` class attribute

```
class NameOfThePROCESS(ABSTRACTCLASSNAME):
    """DOCUMENT IS REALLY IMPORTANT!!

        blablabla blablalbalbalbla

            xx = y

        Attributes
        ----------
            A :: int
                blab

    """
    __sequence_id__ = 1111
    __name__ = 'NameOfThePROCESS'

    def __init__(self):
        """
        """
        super().__init__()

        self.A

        ###
        self.__units__.update({'A':u.ADC/u.pixel})

    def execute_process(self,rawdata):
        """
        """
        print("Process <NameOfThePROCESS> INFO. Bla bla bla bla usin ")
        print("  using {} ADC/pix".format(self.A))

        #### CODE HERE YOUR ALGORITHM
        ###########################################################################

        # get image

        # create blab bla bla

        ###########################################################################

        #### if required, add attribute to RawData object
        setattr(rawdata,"attr_name",obj_attr_name)
```

Documentation is really important!!

Extending the library of process: how to

Building a new process

must inherit from abstract class
DigitizeProcess, SKImageProcess or MEabs

```
class NameOfThePROCESS(ABSTRACTCLASSNAME):
    """DOCUMENT IS REALLY IMPORTANT!!

        blablabla blablalbalbalbla


          xx = y


        Attributes
        ----------
            A :: int
                blab

    """
    __sequence_id__ = 1111
    __name__ = 'NameOfThePROCESS'

    def __init__(self):
        """
        """
        super().__init__()

        self.A

        ###
        self.__units__.update({'A':u.ADC/u.pixel})

    def execute_process(self,rawdata):
        """
        """
        print("Process <NameOfThePROCESS> INFO. Bla bla bla bla usin ")
        print("  using {} ADC/pix".format(self.A))

        #### CODE HERE YOUR ALGORITHM
        ##########################################################################

        # get image

        # create blab bla bla

        ##########################################################################

        #### if required, add attribute to RawData object
        setattr(rawdata,"attr_name",obj_attr_name)
```

Documentation is really important!!

It has **mandatory ingredients**

- (class attribute name) string uniquely identifying the process
- (class attribute sequence id) integer defining the per default order in the sequence of the process to be executed
- any parameter to be configured from the json file must be added to the object dictionary `__units__`
- the algorithm must be implemented in the method `execute_process(self,rawdata)`
- The process does not return nothing:
  - any needed parameter must be appended into the rawdata object
- The *automatic plug-in mechanism* is still work in progress. So far, the process manager must be informed ad hoc

  add the process in the `ProcessManager.__valid_processes__` class attribute

Extending the library of process: how to

Building a new process

must inherit from abstract class
DigitizeProcess, SKImageProcess or MEabs

It has **mandatory ingredients**

- (class attribute name) string uniquely identifying the process
- (class attribute sequence id) integer defining the per default order in the sequence of the process to be executed
- any parameter to be configured from the json file must be added to the object dictionary `__units__`
- the algorithm must be implemented in the method `execute_process(self,rawdata)`
- The process does not return nothing:
  - any needed parameter must be appended into the rawdata object
- The *automatic plug-in mechanism* is still work in progress. So far, the process manager must be informed ad hoc
  - add the process in the `ProcessManager.__valid_processes__` class attribute

```python
class NameOfThePROCESS(ABSTRACTCLASSNAME):
    """DOCUMENT IS REALLY IMPORTANT!!

        blablabla blablalbalbalbla

        ..math:
            xx = y + x


        Attributes
        ----------
            A :: int
                blab

    """
    __sequence_id__ = 1111
    __name__ = 'NameOfThePROCESS'

    def __init__(self):
        """
        """
        super().__init__()

        self.A

        ###
        self.__units__.update({'A':u.ADC/u.pixel})

    def execute_process(self,rawdata):
        """
        """
        print("Process <NameOfThePROCESS> INFO. Bla bla bla bla usin ")
        print("  using {} ADC/pix".format(self.A))

        #### CODE HERE YOUR ALGORITHM
        ###########################################################################

        # get image

        # create blab bla bla

        ###########################################################################

        #### if required, add attribute to RawData object
        setattr(rawdata,"attr_name",obj_attr_name)
```

Extending the library of process: how to

```python
class SKImageProcess(metaclass=ABCMeta):
    """Abstract Class for the digitilize process.

    Each process involved in the response of the DAMIC-M detector (based on CCD
    be created from this abstract class.

    """
    __axis_cartesian__ = {'x':0, 'y':1}
    __axis_name__      = {1:'col',0:'row'}
    __axis_id__        = {'col':1,'row':0,'both':[0,1]}
    __verbose__        = False
    __use_mad__        = True
    __fig_num__        = 5000
    def __init__(self):
        """
        """
        ### Add some attributes common for all process

        self.image = "raw"
        self.save_image = False
        self.save_plots = False

        ### define range for the 3 axis
        self.id_skip_start = np.nan
        self.id_skip_end   = np.nan

        self.id_col_start = np.nan
        self.id_col_end   = np.nan

        self.id_row_start = np.nan
        self.id_row_end   = np.nan

        self.exit = 0

        self.__units__ = {"__verbose__":1,
                "image":1,"__sequence_id__":1,
                "save_image":1,"save_plots":1,
                "id_skip_start":1,"id_skip_end":1,
                "id_row_start":1,"id_row_end":1,
                "id_col_start":1,"id_col_end":1}

    def info(self):
        """ Show all attributes of the process
        """
        print("<{}> with sequence id {}.\n List of public data members: ".forma
        for attr in sorted(self.__units__.keys()):
            print("\t * {} = {} ".format(attr, getattr(self,attr)))

    def print_warning(self,msm):
        print("\x1b[93m    WARNING. {} \x1b[m".format(msm))

    @staticmethod
    def find_peak_position(image,n_moving_avg=50, min_distance=1,in_ADCs=True,
        """
        """
```

Each abstract class
(DigitizeProcess, SKImageProcess or Meabs)
has already a pre-defined configuration parameters.
Use the **super mechanism** to call the constructor

```python
class NameOfThePROCESS(ABSTRACTCLASSNAME):
    """DOCUMENT IS REALLY IMPORTANT!!

        blablabla blablalbalbalbla

        ..math:
            xx = y + x


        Attributes
        ----------
            A :: int
                blab

    """
    __sequence_id__ = 1111
    __name__ = 'NameOfThePROCESS'

    def __init__(self):
        """
        """
        super().__init__()

        self.A

        ###
        self.__units__.update({'A':u.ADC/u.pixel})

    def execute_process(self,rawdata):
        """
        """
        print("Process <NameOfThePROCESS> INFO. Bla bla bla bla usin ")
        print("  using {} ADC/pix".format(self.A))

        #### CODE HERE YOUR ALGORITHM
        ##########################################################################

        # get image

        # create blab bla bla

        ##########################################################################

        #### if required, add attribute to RawData object
        setattr(rawdata,"attr_name",obj_attr_name)
```

Extending the library of process: example

Building a new process

Let’s see a very simple example

pysimdamicm/processes/skipper_analysis.py

mandatory

```
class CalibrationProcess(SKImageProcess):
    """The image will be calibrated: from ADC to electrons using the gain parameter

    """
    __sequence_id__ = 25
    __name__ = 'CalibrationProcess'

    def __init__(self):
        """
        """
        super().__init__()

        self.image = "mean_compressed_pedestal_subtracted"
        self.gain  = 5.3*u.ADC

        ###
        self.__units__.update({'gain':u.ADC})

    def execute_process(self,rawdata):
        """
        """
        print("Process <CalibrationProcess> INFO. The image will be calibrated using a gain of {} ADC/e-".format(self.gain))

        #### GET IMAGE
        ################################################################################
        itype = rawdata.get_image_attribute_name(self.image)
        image = getattr(rawdata,itype)

        setattr(rawdata,"{}_e".format(itype),image/self.gain)

```

Extending the library of process: example

Plug-in ability

Building a new process

Let's see a very simple example

pysimdamicm/processes/skipper_analysis.py

```python
class CalibrationProcess(SKImageProcess):
    """The image will be calibrated: from ADC to electrons using the gain parameter

    """
    __sequence_id__ = 25
    __name__ = 'CalibrationProcess'

    def __init__(self):
        """
        """
        super().__init__()

        self.image = "mean_compressed_pedestal_subtracted"
        self.gain  = 5.3*u.ADC

        ###
        self.__units__.update({'gain':u.ADC})

    def execute_process(self,rawdata):
        """
        """
        print("Process <CalibrationProcess> INFO. The image will be calibrated using a gain of {} ADC/e-".format(self.gain))

        #### GET IMAGE
        ################################################################################
        itype = rawdata.get_image_attribute_name(self.image)
        image = getattr(rawdata,itype)

        setattr(rawdata,"{}_e".format(itype),image/self.gain)

```

Any JSON configurable param must be defined here

Extending the library of process: example

Building a new process

Let's see a very simple example

pysimdamicm/processes/skipper_analysis.py

```python
class CalibrationProcess(SKImageProcess):
    """The image will be calibrated: from ADC to electrons using the gain parameter

    """
    __sequence_id__ = 25
    __name__ = 'CalibrationProcess'

    def __init__(self):
        """
        """
        super().__init__()

        self.image = "mean_compressed_pedestal_subtracted"
        self.gain  = 5.3*u.ADC

        ###
        self.__units__.update({'gain':u.ADC})

    def execute_process(self,rawdata):
        """
        """
        print("Process <CalibrationProcess> INFO. The image will be calibrated using a gain of {} ADC/e-".format(self.gain))

        #### GET IMAGE
        ################################################################################
        itype = rawdata.get_image_attribute_name(self.image)
        image = getattr(rawdata,itype)

        setattr(rawdata,"{}_e".format(itype),image/self.gain)
```

This is the pure virtual method were your algorithm must be implemented

https://gev.uchicago.edu/compton/howto_notebooks/analysis/I_howto_RawData.html

The configuration parameter: image

The **'image'** in the JSON file [only for panaSKImg]

- is the name of the image to be manipulated
- some processes (as this one) append a new image into the RawData object as a data member, which can be used for any further process

```
class CalibrationProcess(SKImageProcess):
    """The image will be calibrated: from ADC to electrons using the gain parameter

    """
    __sequence_id__ = 25
    __name__ = 'CalibrationProcess'

    def __init__(self):
        """
        """
        super().__init__()

        self.image = "mean_compressed_pedestal_subtracted"
        self.gain  = 5.3*u.ADC

        ###
        self.__units__.update({'gain':u.ADC})
```

The configuration parameter: image

The **'image'** in the JSON file [only for panaSKImg]

- is the name of the image to be manipulated
- some processes (as this one) append a new image into the RawData object as a data member, which can be used for any further process

process name	image param	`RawData` attribute	appended `RawData` attribute
`CompressSkipperProcess`	raw	image	image_mean_compressed
`PedestalSubtractionProcess`	mean_compressed	image_mean_compressed	image_mean_compressed_pedestal_subtracted
`CalibrationProcess`	mean_compressed_pedestal_subtracted	image_mean_compressed_pedestal_subtracted	image_mean_compressed_pedestal_subtracted_e

https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_Config.html

The configuration parameter: image

Plug-in ability

The **'image'** in the JSON file [only for panaSKImg]

- is the name of the image to be manipulated
- some processes (as this one) append a new image into the RawData object as a data member, which can be used for any further process

process name	image param	RawData attribute	appended RawData attribute
CompressSkipperProcess	raw	image	image_mean_compressed
PedestalSubtractionProcess	mean_compressed	image_mean_compressed	image_mean_compressed_pedestal_subtracted
CalibrationProcess	mean_compressed_pedestal_subtracted	image_mean_compressed_pedestal_subtracted	image_mean_compressed_pedestal_subtracted_e

```
        "CompressSkipperProcess":
        {
            "func_to_compress":["mean"]
        },
        "PedestalSubtractionProcess":
        {
            "image":"mean_compressed",
            "method":"gauss_fit",
            "in_overscan":true,
            "use_mad":false,
            "axis":"row",
            "n_sigma_win_fit":3,
            "n_sigma_to_mask":-1,
            "show_fit":false,
            "histequ":false
        },
        "CalibrationProcess":
        {
            "image":"mean_compressed_pedestal_subtractedi_e",
            "gain":5.3
        },
```

```
        "FitCalibrationConstant":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "n_peaks":3,
            "calibration":5,
            "n_sigma_win_fit":3
        },
        "FitDarkCurrentProcess":
        {
            "image":"mean_compressed_pedestal_subtracted",
            "method":"root",
```


https://gev.uchicago.edu/compton/howto_notebooks/analysis/howto_Config.html

The Units Class

pysimdamicm/processes/skipper_analysis.py

```python
class CalibrationProcess(SKImageProcess):
    """The image will be calibrated: from ADC to electrons using the gain parameter

    """
    __sequence_id__ = 25
    __name__ = 'CalibrationProcess'

    def __init__(self):
        """
        """
        super().__init__()

        self.image = "mean_compressed_pedestal_subtracted"
        self.gain  = 5.3*u.ADC

        ###
        self.__units__.update({'gain':u.ADC})

    def execute_process(self,rawdata):
        """
        """
        print("Process <CalibrationProcess> INFO. The image will be calibrated using a gain of {} ADC/e-".format(self.gain))

        #### GET IMAGE
        ##########################################################################
        itype = rawdata.get_image_attribute_name(self.image)
        image = getattr(rawdata,itype)

        setattr(rawdata,"{}_e".format(itype),image/self.gain)

```

The Units Class

The singleton class Units defines *the unit system* and *its default*

- Any configurable parameter has units
  - 1: for strings, boolean and addimensional
- `execute_process`: the algorithm implementation must be done in the default units
- Be careful with the units when adding an attribute to RawData object. For instance, if you just want to have something in units of keV

```
something_keV = something/u.keV
```

pysimdamicm/utils/units.py

```python
@Singleton
class Units(object):
    """Define the system units by default and its conversion factors


    Attributes
    ----------
    mm : float
        milimeter, default unit for coordenates

        ...

   ADC2eV : float
        Value to convert ADC (analog-to-digital converter) units to keV,
        default is 2.6E-4.

    """

    def __init__(self):

        ### default units as the ones from geant4 simulations
        self.mm = float(1.0)
        self.eV = float(1.0)
        self.s  = float(1.0)
        self.pixel = int(1)
        self.e = int(1)
        self.eVee = float(1.0)

        ### ADCu
        self.ADC = 1

        #### Conversion factors
        self.e2ADC = 14.5*self.ADC
        self.e2eV = 3.77*self.eV
        self.eV2ADC = 1/self.e2eV * self.e2ADC
        self.ADC2eV = 1/self.e2ADC * self.e2eV
        #### added for comissioning
        self.ADC2e  = 5.3

```

```python
        #### CCD dimensions
        self.ccd_shape=(int(4000),int(6000))*self.pixe
        self.ccd_pixel_size_x=float(15./1000.)*self.mm
        self.ccd_pixel_size_y=float(15.0/1000.)*self.m
        self.ccd_thickness=0.675*self.mm
        ### overscan region
        self.n_rows_overscan = 0
        self.n_rows_prescan = 0
        self.n_cols_overscan = 0
        self.n_cols_prescan = 0

        self.pix2mm=self.ccd_pixel_size_x
        self.mm2pix=1/self.ccd_pixel_size_x

        ### define conversions for all the others
        ### long
        self.um = 1e-3 * self.mm
        self.cm = 10 * self.mm
        self.m  = 1e3 * self.mm
        ### energy
        self.keV = 1e3 *self.eV
        self.MeV = 1e6 *self.eV
        self.GeV = 1e9 *self.eV
        ### time
        self.us = 1e-6*self.s
        self.ms = 1/1000*self.s
        self.minute = 60.0*self.s
        self.hour = 60.0*self.minute
        self.day = 24.0*self.hour

        ### pixel saturation
        self.pix_saturation = 0.0*self.ADC*self.ADC2eV

        ### add sensitive detector name
        self.detector_name = "CCDSensor_PV"

        ### non related with units
        self.n_figure = 1
```

Overview

- WADERS is a framework for both simulations and reconstruction of skipper images
  - defines and provides the infraestructure
  - provides an extense library of processes
  - and can be extended
  - the latest version v3.0.1 is functional and stable
    - extensively used for simulations, Compton setup, ...

- Can be used
  - for comissioning purposes
  - for high level analysis
  - as a data quality monitor
  - as a plotter
  - ...

- Extensively documented
  - Implementation details (design, classes, definitions, interfaces, ...)
  - *howto* notebooks (algorithm description, running examples, ...)
- Huge effort, *but still needs to be improved, feedback will be welcome*

https://ncastell.web.cern.ch/ncastell/pysimdamicm/

https://gev.uchicago.edu/compton/

But in continuous development

Working on

- In the Simulations Scope
  - allow Saturation process to be applied even when noise is not booked [Nuria]
  - paste clusters into blank image (coming soon, needed now for Compton) [Nuria/Nick]
  - Karthik Diffusion model [Nick]
  - allow to record clusterTrue and clusterRec (now goes all into clusterRec) [Nuria]
  - add library of plots for background mitigation studies (the one I have is part of another module) [Nuria]
  - link to a data base for contamination materials

- In the analysis scope
  - still to premature to plan improvements … we just started to use it!

- DQM
  - set/unset ME via JSON [Agustin]
  - write documentation for each ME process [Agustin]
  - output as a pickle object → migrate to mongoDB [Alvaro/Jordi/Nuria]
  - create an interactive web [Jordi/Nuria]
  - include automatic publication to a web [Jordi/Nuria]

- [general] Framework
  - process library on runtime to avoid to add the process into the process_manages [Nuria]
  - analysis code documentation [Agustin/Nuria]
  - integrate all documentation into the official web [Nuria]
  - clean code from deprecated options [Nuria]

The code will improve as the people start to use it!

softWAre for Dark matter ExpeRiments with Skippers

WADERS Framework

pronounced [v AI d e r s]

Questions?

The code will improve as the people start to use it!

Any bug? Any suggestion?

contact me on slack, by e-mail

or **preferently open an issue** on gitlab.in2p3.fr

https://gitlab.in2p3.fr/damicm/pysimdamicm/-/issues