PCAT Documentation

Release 0.1

Tansu Daylan

May 22, 2017

Contents

When testing hypotheses or inferring their free parameters, a recurring problem is to compare models that contain a number of elements whose multiplicity is itself unknown. Therefore, given some data, it is desirable to be able to compare models with different numbers of parameters. One way of achieving this is to obtain a point estimate (usually the most likely point) in the parameter space of each model and, then, rely on some information criterion to penalize more complex models for excess degrees of freedom. Another way is to sample from the parameter space of each model and compare their Bayesian evidences. Yet another is to take samples from the union of these models using a set of transdimensional jumps across models. This is what the Probabilistic Cataloger (PCAT) is designed for.

PCAT is a hierarchical, transdimensional, Bayesian inference framework. It's theoretical framework is introduced in Daylan, Portillo & Finkbeiner (2016), accepted to ApJ. In astrophysical applications, given the output of a photon counting experiment, it can be used to sample from **the catalog space**. In a more general context, it can be used as a mixture sampler to infer the posterior distribution of a metamodel given some Poisson distributed data.

In what follows, we assume that **the metamodel** is the union of models with different dimensionality. All such models have a certain number of common, **fixed-dimensional** parameters. In addition, each model has a different number of **elements**. An element is a collection of parameters that only exist together, and characterize an entity in the model. Examples are:

- A light source such as a star or galaxy, in an astrophysical emission model,
- A light deflecting dark matter subhalo in an gravitational lensing model,
- A term in the polynomial used to perform linear regression,

CHAPTER 1

Installation

To install PCAT you can use pip

```
pip install pcat
```

or download the latest release and run

```
python setup.py install
```

CHAPTER 2

Features

Compared to mainstream Bayesian inference methods, PCAT has a series of desirable features. It

- samples from the space of catalogs given some observation, unlike conventional cataloging, which estimates the most likely catalog,
- allows marginalization over all relevant nuisance parameters in the problem, including the dimensionality of the nuisance,
- reveals potentially non-Gaussian within and across model covariances,
- constrains element population characteristics via hierarchical priors,
- is a Bayesian framework, because point estimates fail in nearly degenerate likelihood topologies,
- implements Occam's razor (model parsimony) through natural priors on the number of degrees of freedom,
- strictly respects detailed across models,
- does not discard information contained in low-significance $(< 5\sigma)$ fluctuations in the observed dataset,
- reduces to a deterministic cataloger when the labeling degeneracy is explicitly broken,
- simultaneously infers the Point Spread Function (PSF) and the level of diffuse background.

Transdimensionality

PCAT takes steps across models by proposing to add elements whose parameters are drawn from the prior, or to kill randomly chosen elements, while respecting detailed balance in the metamodel. Apart from these elementary transdimensional operations, it can also optionally propose splits and merges of elements to efficiently sample typical across-model covariances.

Hierarchical priors

When there are multiple model elements, each with a set of parameters, it is natural to put priors on the **distribution** of these parameters, as opposed to placing individual priors separately on each element parameter. This assumes that a certain type of parameter of all elements, are drawn from a single probability distribution. This is particularly useful when such a parameter is subject to inference, and individual elements can be marginalized over. This results in a hierarchical prior structure, where the prior is placed on the distribution of element parameters, i.e., **hyperparameters**, and the prior on the individual element parameters are made conditional on these hyperparameters.

Labeling degeneracy

Due to **hairlessness** of the elements, the likelihood function is invariant to their permutations in the parameter vector, i.e., exchanging the labels of two elements leaves the likelihood invariant. This fact has consequences for **nonpersistent** elements, which get born or killed at least once during an MCMC run. Because of label changes, the posterior of these parameters look the same, which makes them useless for inferring their properties. In order to constrain such elements, the degeneracy must be broken in postprocessing of the samples. Note that, if the sampling is continued sufficiently long, e.g., for a Hubble time, the posterior of all transdimensional parameters will eventually look similar.

Proposal optimization

Construction of an MCMC chain requires a choice of proposal scale, which must remain constant (or strictly decrease) during sampling in order to respect detailed balance. In transdimensional inference, the choice of proposal scale includes both within and across model jumps. PCAT chooses the proposal scale for a particular sampling problem based on an initial calculation of the Fisher information at the maximum likelihood solution, in the beginning of each run, yielding an estimate of the model covariance. The tighter a parameter is constrained, the smaller the proposal scale for that parameter becomes. This ensures that the acceptance ratio is around 25% and minimizes the autocorrelation time of the resulting chain.

PCAT takes heavy-tailed within-model steps in a space, where the prior is uniformly distributed.

In order to ensure that the chains start from a well mixed state, the first `numbburn` samples are discarded and the resulting chain is thinned by a factor of `factthin`.

Performance

The above features are made possible by enlarging the hypothesis space so as to minimize mismodeling of the observed data. This is, however, at the expense of seriously slowing down inference.

PCAT alleviates the performance issues in two ways:

- Use of parallelism via bypassing Python's Global Interpreter Lock (GIL) by employing independent processes as opposed to threads. The parent process spawns multiple, (almost) noninteracting processes, which collect samples in parallel and report back to the parent process, which aggregates and postprocesses the samples.
- Locality approximation in the likelihood evaluation. The most time consuming part of the inference is the model evaluation, which can be prohibitively slow (for large datasets and many elements) if no approximations are made. PCAT assumes that the contribution of the elements to the model vanishes outside some circle around the element.

CHAPTER 3

Input

PCAT expects input data in the folder `pathbase/data/inpt/`. The typical inputs are the data and exposure data cubes, background template(s) (if applicable), element kernel template(s) and the PSF (if applicable).

Supplying data

Input dataset is provided through the `strgexprflux` argument. This should be the name of a FITS file (including the `.fits` extension), which contains a numpy array of dimension $N_e \times N_{pix} \times N_{psf}$, where N_e is the number of energy bins, N_{pix} is the number of spatial pixels and N_{psf} is the number of PSF classes. The units should be photons per cm ^{2} per seconds per GeV.

The exposure map is suppled via the `strgexpo` argument. This should be the name of a FITS file (including the `.fits` extension). The format should be the same as `strgexprflux`, whereas units should be cm ^{2} s.

Similarly, background templates can be provided via the `back` argument. `back` should be a list of FITS file names (including the `.fits` extension), whose format should be same as that of `strgexprflux`.

Specifying the model and placing priors

The prior structure of the model is set by the relevant arguments to `pcat.main.init()`. PCAT allows the following fixed dimensional parameters in each member of the metamodel:

- Background prediction

  Given an observed count map, the number of counts in each pixel can be modeled as a Poisson realization of the total counts from a number of elements. In practice, however, this number can be prohibitively large, when there are many more **faint** elements compared to bright ones. Therefore, it is computationally more favorable to represent the overall contribution of faint elements (those that negligibly affect the likelihood) with **background** templates. PCAT allows a list of spatially and spectrally distinct background templates to be in the model simultaneously.

- PSF

PSF defines a how a delta function in the position space projects onto the observed count maps. When the elements are count sources, PCAT assumes that the projection of all elements to the count maps are characterized by the same PSF. PSF convolution of the model prediction is only applicable for applications, where the observed count map has been collected with an instrument with a finite PSF.

Distributions of element features

All features of elements are assumed to have been drawn from an underlying probability distribution, and cross-correlations between element features are assumed to be zero. The 1-point function of the element features are controlled by the function arguments to `pcat.main.init()`. Furthermore, elements are divided into **populations**, where each population admits its own set of hyperparameters. Therefore the input to these arguments should be a `list` of strings.

- Spatial distribution (`spatdisttype`)
- `'unif'`

Both horizontal positions, θ_1, and vertical positions, θ_2, are uniformly distributed between $-\theta_{max}$ and θ_{max}, where $2\theta_{max}$ is the side of the square in which model elements can exist.

$$P(\theta_1) = \frac{1}{2\theta_{max}}$$
$$P(\theta_2) = \frac{1}{2\theta_{max}}$$

- `'disc'`

Horizontal positions, θ_1, are assumed to be uniformly distributed, whereas the vertical positions, θ_2, are drawn from the exponential distribution.

$$P(\theta_1) = \frac{1}{2\theta_{max}}$$
$$P(\theta_2) = \frac{1}{\theta_{2s}} \exp\left(-\frac{\theta_2}{\theta_{2s}} \right)$$

The scale of the exponential distribution, θ_{2s}, is a hyperparameter subject to inference and set by `bgaldistscal`.

- `'gang'`

Radial positions, θ_r, are assumed to follow an exponential distribution with an angular scale, θ_{rs}, controlled by `gangdistscal`. The azimuthal positions, ϕ, are uniformly distributed.

$$P(\phi) = \frac{1}{2\pi}$$
$$P(\theta_r) = \frac{1}{\theta_{rs}} \exp\left(-\frac{\theta_r}{\theta_{rs}} \right)$$

- `'gaus'`

The prior spatial distribution is the sum of a certain number of Gaussians and a spatially constant floor.

$$P(\theta_1, \theta_2) \propto \alpha_c + c \sum_k \exp\left(-\frac{1}{2} \frac{(\theta_1 - \theta_k)^2}{\sigma_c^2} \right) \exp\left(-\frac{1}{2} \frac{(\theta_2 - \theta_k)^2}{\sigma_c^2} \right)$$

c is a normalizing constant that brings the maximum of the second term to unity. The amplitude of the floor, α_c, is set by the hyperparameter `spatdistcons`. It roughly parametrizes the degree of belief in the provided catalog. This spatial model is useful is there is a strong prior belief that elements exist at certain locations. An example is searching for elements at the positions of a earlier (deterministic) catalog.

- Flux distribution (`fluxdisttype`)
- `'powr'`

Power law between `minmflux` and `maxmflux` with the slope `fluxdistslop`.

- Spectral index distribution

Spectral index distribution of elements can be set with the argument `sinddisttype`.

- `'atan'`

Spectral indices s_a are distributed such that $\arctan(s_a)$ follow the uniform distribution between $\arctan(s_{min})$ and $\arctan(s_{max})$. s_{min} and s_{max} can be set by `minmsind` and `maxmflux`, respectively.

- `'gaus'`

Spectral indices s_a follow a Gaussian distribution with mean λ_s and variance σ_s^2.

Generating mock data

PCAT ships with a mock (simulated) data generator. Mock data is randomly drawn from a generative metamodel, not to be confused with the metamodel subject to inference, (hereafter, **the fitting metamodel**). Once the user configures the prior probability density of the fitting metamodel, PCAT samples from this metamodel given the simulated dataset. Some of the generative metamodel parameters can be fixed, i.e., assigned delta function priors. These are

- the number of mock elements, `truenumbpnts`,
- hyperparameters controlling the population characteristics of these elements.

All other generative metamodel parameters are fair draws from the hierarchical prior.

When working on variations of a certain problem or different analyses on the same dataset, it is useful to have default priors. PCAT allows unique defaults for different built-in experiments, controlled by the argument `exprtype`. Currently the built-in experimental types are

- `ferm`: Fermi-LAT (Default)
- `chan`: Chandra
- `hubb`: Hubble Space Telescope
- `sdss`: SDSS
- `gaia`: Gaia

By setting `exprtype` the user imposes the default prior structure for the chosen experimental type. However, it is also desirable to be able to change the prior of a specific parameter. This is accomplished by setting the *relevant argument(s)*.

Note: PCAT has a built-in fudicial (default) generative metamodel for each experiment type. The user can change parts of this model by providing arguments with the parameter names with a preceeding string `'true'` (indicating the fudicial model). The model subject to inference defaults to the resulting fudicial model. The user can also change parts of the fitting metamodel by setting arguments with the relevant parameter names (this time, without `'true'`, indicating the fitting metamodel). In other words, if the user does not specify any parameters, the fudicial model will be used to generate data, and the same model will be fitted to the data. In most cases, however, one is be interested in studying mismodeling, i.e., when the mock data is generated from a model different from that used to fit the data. This can be achieved by forcing the prior structure of the fitting metamodel to be different from the generator model.

Selecting the initial state

The initial state of the chain is drawn randomly from the prior.

CHAPTER 4

Output

A function call to `pcat.main.init()` returns the collected samples as well as postprocessed variables in an object that we will refer to as `gdat`. Any output (as well as many internal variables of the sampler) can be accessed via the attributes of this global object.

Furthermore, PCAT offers extensive routines to visualize the output chain. The output plots are placed in the relevant subfolders `pathbase/data/outp/rtag` and `pathbase/imag/rtag`, respectively, where `rtag` is the run tag. The `pathbase` folder is created if it does not already exist. It defaults to the value of the environment variable `$PCAT_DATA_PATH`. Therefore `pathbase` can be omitted by setting the environment variable `$PCAT_DATA_PATH`.

Plots

If not explicitly disabled by the user, PCAT produces plots in every stage of a run. Some plots are produced in the initial setup, frame plots are generated at predetermined times during the sampling and others are made in the postprocessing after all chains have run. The plot path, `pathbase/imag/rtag`, has the following subfolders:

- `init` Problem setup
- `prio` Plots regarding the sampling of the prior
- `post` Plots regarding the sampling of the posterior
- `info` Information gain

Note: Running the sampler on the prior probability distribution is optional and serves two purposes:

- diagnostic check that the imposed prior makes sense
- to calculate the KL divergence on all metamodel parameters and derived quantities

If sampling from the prior is disabled (default behaviour), then `prio` and `info` folders will be empty.

In turn, both `prio` and `post` contain the following subfolders:

- `opti` Proposal scale optimization
- `fram` Frame plots, giving snapshots of the MCMC state during the sampling
- `anim` GIF animations made from the frame plots in `fram` that are produced during sampling.
- `finl` Posterior distribution of model parameters and derived quantities
- `diag` Diagnostic plots

`fram` and `post` paths organize the plots into subfolders:

- `assc` Associations of the sample elements with the reference element
- `cmpl` Completeness as a function of reference element features
- `fdis` False discovery rate as a function of model element features
- `histodim` One dimensional histograms of element features
- `histtdim` Two dimensional histograms of pairs of element features
- `scattdim` Scatter plots of pairs of element features

Note: A reference sample is defined as a sample that is overplotted on the metamodel samples. Reference samples always use the green color. If the data is simulated, the true metamodel automatically becomes the reference sample. If the data is supplied by the user, the reference sample is also expected from the user, and is optional.

Last, `finl` paths are the folders that hold the posterior and prior distributions. They offer additional subfolders:

- `cond` Condensed catalog related plots
- `deltllik` Log-likelihood difference for all proposals
- `lpri` Log-prior and other terms in the acceptance ratio for all proposals
- `spmr` Split and merge related plots
- `varbscal` Marginal and joint distributions of all quantities, i.e., model parameters and derived variables.
- `varbscalproc` Same as above, but individually for each chain.

Chain

`pcat.main.init()` returns a pointer to the object that contains the output chain. It also writes the output chain to `$PCAT_DATA_PATH/outp/rtag/pcat.h5`, which is an HDF5 file. Each dataset (or object attribute as in the former case) is an `ndarray` of samples, either of the parameters or of quantities derived from the parameters, as well as diagnostic and utility variables.

In overall, the output folder contains the following files:

- `args.txt`

The list of arguments to `pcat.main.init()`.

- `comp.txt`

A small text file produced at the very end of the run to indicate that the run was completed successfully.

- `opti.h5`

HDF5 file containing the proposal scale optimization data.

- `stdo.txt`

The log of the standard output collected during the run.

- `pcat.h5`

HDF5 file containing samples from the metamodel, with the following fields:

`sampvarb` Parameter vector

`samp` Scaled parameter vector (uniformly distributed with respect to the prior)

`deltllikpopl` Delta log-likelihood of the l^{th} population

`lgalpopl` Horizontal coordinates of the l^{th} population

`bgalpopl` Vertical coordinates of the l^{th} population

`fluxpopl` Flux of the l^{th} population

`sindpopl` Spectral index of the l^{th} population

`curvpopl` Spectral curvature of the l^{th} population

`expopopl` Spectral cutoff energy of the l^{th} population

In order to reduce inter-process communication, PCAT writes its internal state to the disc before child processes are spawned and after individual workers have finished their tasks. These intermediate files, in the form of python pickle objects, are temporarily written to the output folder, but deleted before the run ends.

CHAPTER 5

Diagnostics

Autocorrelation

A chain of states needs to be Markovian (memoryless) in order to be interpreted as fair draws from a target probability density. The autocorrelation of the chain shows whether the chain is self-similar along the simulation time (either due to low acceptance rate or small step size). Therefore the autocorrelation plots should be monitored after each run.

In a transdimensional setting, the autocorrelation of a parameter is ill-defined, since parameters can be born, killed or change identity. Therefore, for such parameters, we calculate the autocorrelation of the model count map.

The acceptance rate of the birth and death moves shows whether the prior on the element parameters are appropriate. If the acceptance rate of birth and deaths proposals is too low, this indicates that an element randomly drawn from the prior is very unlikely to fit the data. In contrast, if it is too high, it means that most of the prior volume is insensitive to the data. Therefore the prior should be adjusted such that the birth and death acceptance rate is between ~ 5

Gelman-Rubin test

PCAT nominally runs multiple, noninteracting chains, whose samples are aggregated at the end. In order to ensure convergence, therefore, one can compare within-chain variance with across-chain variance. This is known as the Gelman-Rubin test. PCAT outputs the GR test statistics in `gdat.gmrb` and plots the relevant diagnostics in `$PCAT_DATA_PATH/imag/rtag/diag/`.

Note: Make sure to run PCAT with the argument `diagmode=False` for reasonable time performance. `diagmode=True` option puts the sampler in a conservative diagnostic mode and performs extensive checks on the state of critical data structures to ensure that the model and proposals are self consistent, largely slowing down execution.

CHAPTER 6

Tutorial

In this tutorial, we will illustrate how to run PCAT during a typical science analysis. Assuming that you have *installed* PCAT, let us run it on mock data.

All user interaction with PCAT can be performed through the `pcat.main.init()` function. Because arguments to this function have hierarchically defined defaults, even the default call (without arguments) starts a valid PCAT run. The default call generates mock Fermi-LAT data with an isotropic background and point source distribution and takes samples from the catalog space of the generated data. Therefore it assumes that you have the necessary Instrument Response Function (IRF) file as well as exposure, data and background flux files in `pathbase/data/inpt/`.

The default run collects a single chain of 100000 samples, discards the initial 20% of the samples and thins the remaining samples to obtain a chain of 1000 samples. The number of processes, total number of samples per process, the number of samples to be discarded and the factor by which to thin the chain can be set using the arguments `numbproc`, `numbswep`, `numbburn` and `factthin`, respectively. After initialization, PCAT collects samples, produces frame plots (snapshots of the sampler state during the execution) and postprocesses the samples at the end. The run should finish in under half an hour with the message

```
>> The ensemble of catalogs is at $PCAT_DATA_PATH/data/outp/rtag/
```

While the sampler is running, you can check `$PCAT_DATA_PATH/imag/rtag/` to inspect *the output plots*.

Although PCAT visualizes the ensemble of catalogs in various projections to the data and model spaces, the user can also work directly on *the output chain*.

CHAPTER 7

API

All user interaction with PCAT is accomplished through the `pcat.main.init()` function. Below is a list of its function arguments.

`pcat.main.`**`init`**`(...)`

Given an observed dataset, sample from the metamodel.

Sampler settings

Parameters

- **`numbswep`** (*`int`*) – Number of samples to be taken by each process
- **`numbburn`** (*`int`*) – Number of samples to be discarded from the beginning of each chain
- **`factthin`** (*`int`*) – Factor by which to thin each chain. Only one sample out of `factthin` samples is saved.
- **`numbproc`** (*`int`*) – Number of processes. The total number of samples before thinning and burn-in, will be `numbproc` times `numbswep`.

Input

Parameters

- **`indxenerincl`** (*`ndarray int`*) – Indices of energy bins to be taken into account. It is only effective if data is provided by the user, i.e., for non-simulation runs, where it defaults to all available energy bins. Other energy bins are discarded.
- **`indxevttincl`** (*`ndarray int`*) – Indices of PSF class bins to be taken into account. Works similar to `indxenerincl`.

Output

Parameters

- **`verbtype`** (*`int`*) – Verbosity level
  - `0` No standard output
  - `1` Minimal standard output including status of progress (Default)

  - `2` Diagnostic verbose standard output

- **`pathbase`** (*`str`*) – Data path of PCAT. See *Output*.

Associations with the reference elements

Parameters

- **`anglassc`** (*`float`*) – Radius of the circle within which sample catalog elements can be associated with the elements in the reference elements.
- **`margfactcomp`** (*`float`*) – The ratio of the side of the square in which the sample elements are associated with the reference elements, to the size of the image.
- **`nameexpr`** (*`str`*) – A string that describes the provided reference element to be shown in the plot legends.
- **`cntrpnts`** (*`bool`*) – Force the mock data to a single PS at the center of the image. Defaults to `False`.

General

Parameters

- **`elemtype`** (*`str`*) – Functional type of elements.
  - `'lght'` Elements are light sources
  - `'lens'` Elements are lenses.
- **`evalcirc`** (*`str`*) – Flag to evaluate the likelihood only inside a circle of a certain radius around elements.

Initial state

Parameters **`randinit`** (*`bool`*) – Force the initial state to be randomly drawn from the prior. Default behavior for mock data is to initialize the chain with the true state.

Adaptivity

Parameters **`optiprop`** (*`bool`*) – Optimize the scale of each proposal by acceptance rate feedback. All samples during the tuning are discarded.

Post processing

Parameters

- **`strgexprflux`** (*`str`*) – Name of the FITS file (without the extension) in `pathdata` containing the observed data as an `ndarray`.
- **`strgcatl`** (*`str`*) – A descriptive name for the provided reference elements to be shown in the plot legends.
- **`strgback`** (*`list of str or int`*) – A list of FITS file names (without the extension) in `pathdata` each containing a spatial template for the background prediction as an `ndarray`. See `strgexprflux` for the content of the file and its unit. One element of the list can be a float, indicating an isotropic template with the provided amplitude.
- **`lablback`** (*`list of str`*) – a list of axis labels for the spatial background templates to be shown in plots.
- **`strgexpo`** (*`str or float`*) – Name of the FITS file (without the extension) in `pathdata` containing the exposure map. See `strgexprflux` for the format of the numpy array. `strgexpo` can also be a float, in which case the exposure map will be assumed to be uniform across along all data dimensions.

- **liketype** (*strg*) – Type of the likelihood.
  - 'pois' Poisson probability of getting the observed number of counts given the model prediction (default).
  - 'gaus' Gaussian approximation of the above. This may accelerate the execution in cases, where the bottle neck of the sampler time budget is likelihood evaluation.
- **exprtype** (*str*) – Name of the experiment used to collect the observed data. exprtype can be used to set other options to their default values for the particular experiment. - 'ferm' Fermi-LAT - 'chan' Chandra - 'hubb' HST - 'sdss' SDSS
- **lgalcntr** (*float*) – Galactic longitude of the image center. lgalcntr and bgalcntr are used to rotate the observed data, exposure and background maps as well as the provided reference elements to the center of the ROI. They are only effective when pixelization is HealPix, i.e, pixltype='heal'.
- **bgalcntr** (*float*) – Galactic latitude of the image center. See lgalcntr.
- **maxmangl** (*float*) – Maximum angular separation at which PSF can be interpolated. It defaults to three times the diagonal legth of the image, enough to evaluate the PSF across the whole image.
- **pixltype** – Type of the pixelization.
  - heal HealPix
  - chan Cartesian

Plotting

Parameters

- **makeplot** (*bool*) – Make output plots, which is the default behavior. If False, no output plots are produced.
- **numbswepplot** (*int*) – Number of samples (before thinning and burn-in) for which one set of frame plots will be produced. Frame plots reveal individual samples in detail and are later used for producing animations.
- **scalmaps** (*str*) – A string that sets the stretch of the count maps
  - 'asnh' Arcsinh (default)
  - 'self' Linear
  - 'logt' Log 10
- **satumaps** (*bool*) – Saturate the count maps
- **exprinfo** (*bool*) – Overplot the provided reference elements on the output plots.
- **makeanim** (*bool*) – Make animations of the frame plots. Defaults to True.
- **anotcatl** (*bool*) – Anotate the catalog members on the plots, if an annotation text is provided along with the reference element. (Default: False)
- **strgbinsener** (*str*) – A string holding the label for the energy axis.
- **asscmetrtype** (*str*) – Type of metric used to associate the sample catalogs with the reference element
- **strgexprname** (*str*) – A string describing the experiment used to collect the observed data.
- **strganglunit** (*str*) – Label for the spatial axes.

- **labllgal** (*str*) – Label for the horizontal axis
- **lablbgal** (*str*) – Label for the vertical axis

Diagnostics

Parameters

- **emptsamp** (*bool*) – Perform a futile run without collecting any samples, but creating all data structures and producing all visualizations as if in a normal run. Defaults to `False`.
- **diagmode** (*bool*) – Start the run in diagnostic mode. Defaults to `False`.

Model

Parameters

- **spatdisttype** (*list of str*) – Type of spatial distribution of elements for each population
- **fluxdisttype** (*list of str*) – Type of flux distribution of elements for each population
- **spectype** (*list of str*) – Type of energy spectrum of elements for each population
- **psfntype** (*str*) – Type of PSF radial profile
- **oaxitype** (*str*) – Type of PSF off-axis profile

Note: The generative metamodel parameters can be set by preceeding the parameter name with `true`. For example, in order to set the mock number of elements, you can specify `truenumbpnts=array([10])`.

CHAPTER 8

Garbage collection

PCAT produces two folders for the output of each run, one for plots and the other to contain the chain saved to the disc. Given that many test and intermediate runs may be needed before each science run, the number of folders (and files therein) may increase quickly. In order to avoid this, the script `gcol.py` is provided to the user as a convenience. When executed, it erases the output from all runs that

- have not run to completion, i.e., does not have the animations, which are produced at the very end, or
- has collected less than 100000 samples per chain.

Index