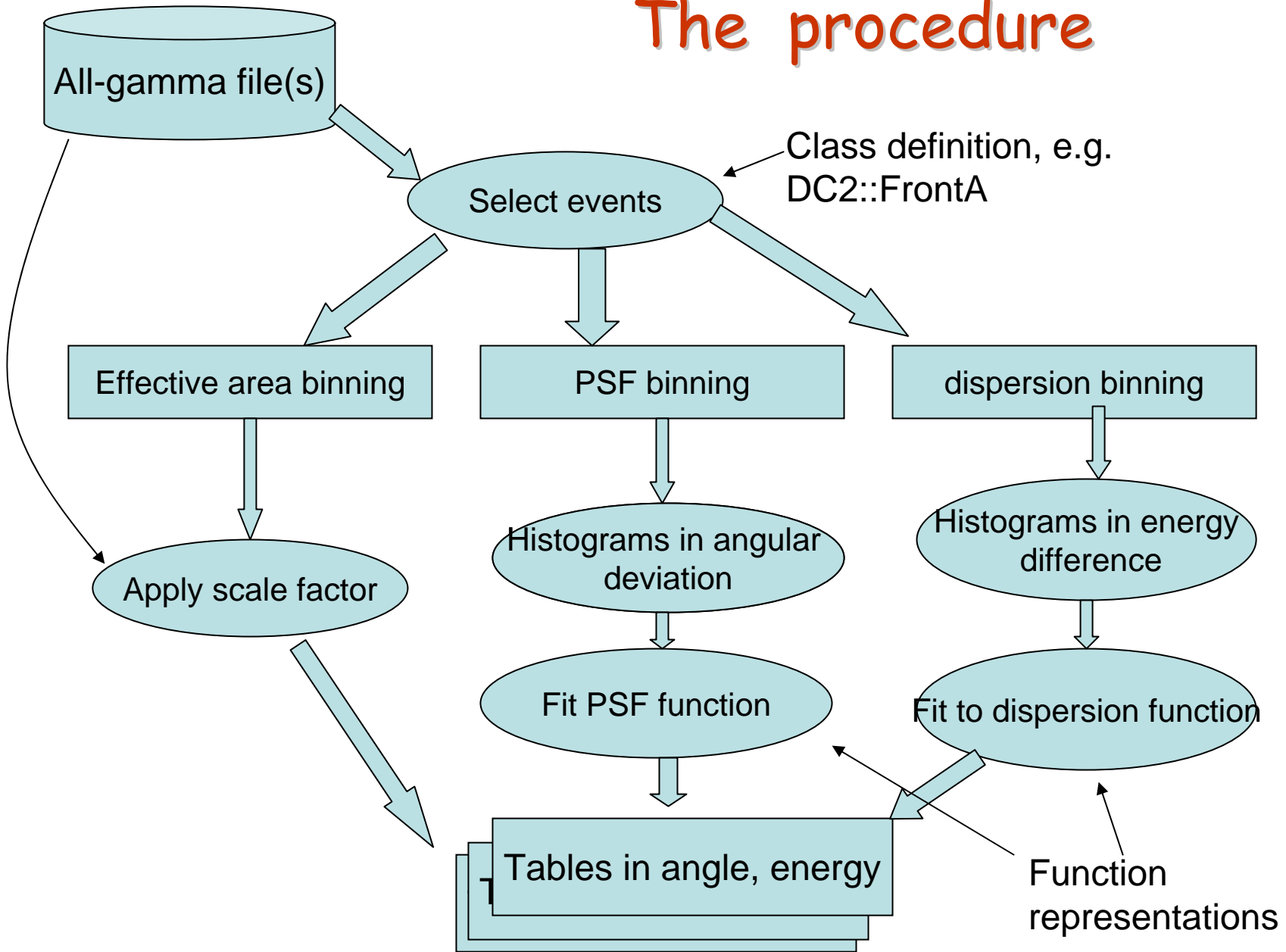


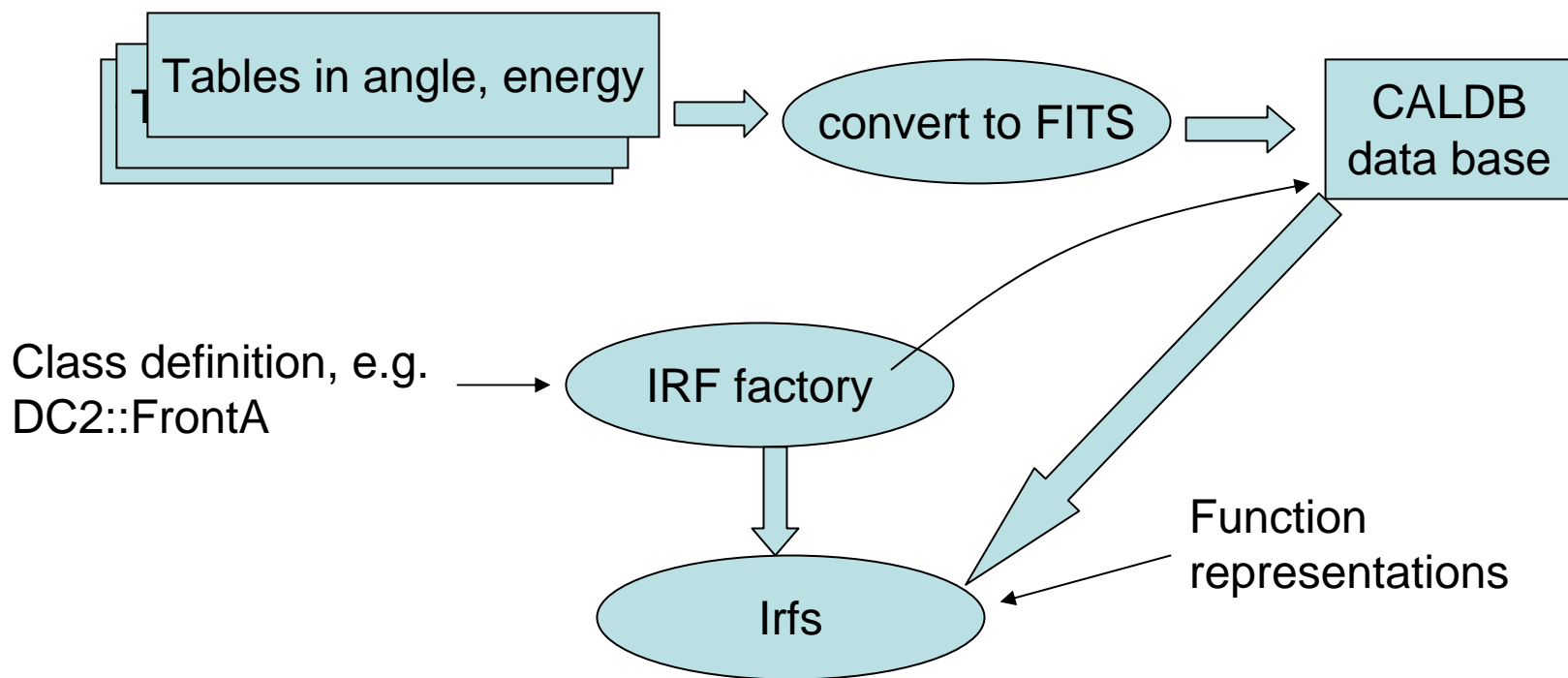
Generating post-DC2 IRFs: a proposal

Toby Burnett

The procedure



Procedure, cont.



A new package demonstrates the proposal

- All code (generation, reading/writing database, access) in one package
 - Irf creation in two stages: prune, then fit. Fitting stage at most 30 s.
 - Same function definitions used for both fitting and evaluation
 - Same $(\log E, \cos\theta)$ binning for PSF and dispersion: (0.5×0.1) effective area subdivides: (0.25×0.02)
 - Reproduces DC2 fits, a_{eff} .
- Data base is not (for now) CALDB: it uses ROOT instead
 - One file structured with TDirectory: top level is the class name, then next level currently “front” or “back”.
 - Each final directory has TH2F objects representing tables of the effective area and each of the parameters
 - Use ROOT to create the tables, and look up a value. (no interpolation yet)

Next, if accepted

- Get help!!!
- Need to implement the integrals, hopefully by a superclass
- Documentation, especially for procedures to create new IRF's
 - Currently pretty easy, but could be streamlined, maybe use STapp framework.
- More testing, especially gtlikelihood failure with DC2 Irf's.
- Connect to CALDB if needed

Example simple test

- Use Jim's pylrfLoader (thanks, Jim!)
- Expose his Loader() and the new (added) irfLoader(): run both to load old and new.
- Create old and new Irfs objects:
 - `dc2FA = IrfsFactory.instance().create('DC2::FrontA') #old`
 - `afront = IrfsFactory.instance().create('classA/front') #new`
- Write simple loop to generate data for plot at right.
 - Note that the tail on the new classA/front does not agree: the new functional form allows for a different powerlaw tail (see the slight kink at 0.9 deg.)
 - `psf_bin4` is `CTBCORE>0.9`.

PSF functions at 1000 GeV, normal

