THE REMEIS ISISSCRIPTS

A HUGE COLLECTION OF USEFUL ISIS FUNCTIONS

V. GRINBERG (MIT), HIGH-RESOLUTION X-RAY SPECTROSCOPIC SOFTWARE AND TOOLS, MAY 12 2016

ISISscripts

http://www.sternwarte.uni-erlangen.de/isis/

The Remeis ISISscripts

A HUGE COLLECTION OF USEFUL ISIS FUNCTIONS

The `ISISscripts` are written and maintained by the Remeis observatory. They contain hundreds of useful functions, written for the X-ray data analysis software `isis` (Interactive Spectral Interpretation System, homepage), which is written in the scripting language `SLang` (homepage). In case you have questions or comments, you can write us an email.

These scripts were started by Manfred Hanke, but soon filled with more functions by many people at the Remeis observatory. Although some functions are quite specialized, the majority of the functions can be widely used. These function range from general `SLang` functions for array manipulation or adaptive integration routines, to parallel fitting functions (together with the Remeis `SLmpi` module) or powerful plotting routines (written by Mike Nowak, see homepage). Usually these functions are well documented with internal help functions. General tips, tricks, and examples can also be found on our wiki page

---

The `ISISscripts` contain functions for the following fields:

- X-ray data analysis
- Timing analysis
- Multiwavelength studies (including, e.g., radio and gamma-ray data)
- Advanced spectral fitting tools (e.g., simultaneous data fitting)
- Advanced plotting routines, employing SLXfig
- Parallel computation routines (uncertainties, contours, ...)
- and many more

A list of publications using the `ISISscripts` can be found here.

Download

The newest version of the isisscripts can be downloaded by clicking on the button to the right. It contains the latest stable release. Note that these scripts are updated almost daily, Hence, if you would like to stay up-to-date, it might be best to schedule a `cron` job and fetch the file via a `wget` command.

ISISSCRIPTS.TGZ

Last Update: 2016-May-11 12:23:12

ISISscripts

http://www.sternwarte.uni-erlangen.de/isis/

- Maintained by Remeis observatory/ECAP Bamberg (Jörn Wilms' group), with help from MIT (mainly me, Mike Nowak …)
- git-directory & downloadable zip; easy to use & update; git makes author of file easy to find out to ask questions …
- contributions and improvements welcome
- include Mike Nowak's isisrc routines (i.e., you can do all the cool things from Mike's talk - and MORE)
- many/most functions parallelized

ISISscripts & HighRes

(my somewhat biased pick …)

- confidence limits (`fit_pars`)
- saving (and re-loading) your fit results (`fits_save_fit`)
- parallel computing with MPI (`SLmpi` module)
- Monte Carlo based approaches (`emcee, mc_sig`)
- Bayesian blocks (`bayesian_blocks`)
- customized line functions (`init_line`, …)
- simultaneous fitting of multiple spectra (`simfit`)
- multiwavelength datasets (`define_counts`)
- easy, pretty plots (`SLxfig`)

fit_pars (cf. conf_loop)

```
isis> help fit_pars
fit_pars

 SYNOPSIS
    computes single-parameter confidence limits for several parameters

 USAGE
    Struct_Type results = fit_pars([Integer_Type pars[]]);

 QUALIFIERS
    ; strict[=1]: restarts the calculation if a new best fit was found
    ; saveoutput[=1]
    ; basefilename[=<date_time>]
    ; level[=1]: specifies the confidence level. Values of 0, 1, or 2
                 indicate 68%, 90%, or 99% confidence levels respectively.
                 By default, 90% confidence limits are computed.
    ; tolerance: convergence criterion for the calculation of the confidence
      limits (see help for the conf command). Default: 1e-3

 DESCRIPTION
    The return value results = struct { index, name, value, min, max, conf_min, conf_max, buf_be
low, buf_above, tex }
    is a table with the following information for each parameter:
    min and max are the minimum/maximum values allowed.
    conf_min and conf_max are the confidence limits.
    buf_below (buf_above) is the fraction of the allowed range [min:max]
    which separates the lower (upper) confidence limit from min (max).
    If one of these buffers is 0, your confidence interval has bounced.
```

fit_pars

```
isis> variable cont = fit_pars;
```

...

```
(  min.all. <=)  conf.min. <                          parameter value <  conf.max. (<=   max.all.)
(         0 <=) 0.00837953 <          powerlaw(1).norm = 0.00837953 < 0.00973384 (<=           1)
   [ 0.8% (conf) 99.0%]   $\left(8.3795300000^{+1.3543137875}_{-0.0000000016}\right)\times10^{-3
}$
(        -2 <=)   -1.52813 <      powerlaw(1).PhoIndex =   -1.25489 <  -0.977923 (<=           9)
   [ 4.3% (conf) 90.7%]   $-1.25\pm0.28$
(         0 <=) 3.87347e-05 <             gauss(1).area = 6.03972e-05 < 8.2274e-05 (<=
1)   [ 0.0% (conf)100.0%]   $\left(6.0\pm2.2\right)\times10^{-5}$
(     1e-06 <=)      1e-06 <            gauss(1).sigma =      1e-06 < 0.00120605 (<=           1)
   [ 0.0% (conf) 99.9%]   $\le1.3\times10^{-3}$
(         0 <=) 3.02621e-05 <             gauss(2).area = 5.25643e-05 < 7.62578e-05 (<=
 1)   [ 0.0% (conf)100.0%]   $\left(5.3^{+2.4}_{-2.3}\right)\times10^{-5}$
(     1e-06 <=)      1e-06 <            gauss(2).sigma = 0.00457944 <  0.0107828 (<=           1)
```

...

```
isis> print(cont);
{index=Integer_Type[19],
 name=String_Type[19],
 value=Double_Type[19],
 min=Double_Type[19],
 max=Double_Type[19],
 conf_min=Double_Type[19],
 conf_max=Double_Type[19],
 buf_below=Double_Type[19],
 buf_above=Double_Type[19],
 tex=String_Type[19]}
```

also as parallelized error calculation:

`mpi_fit_pars`

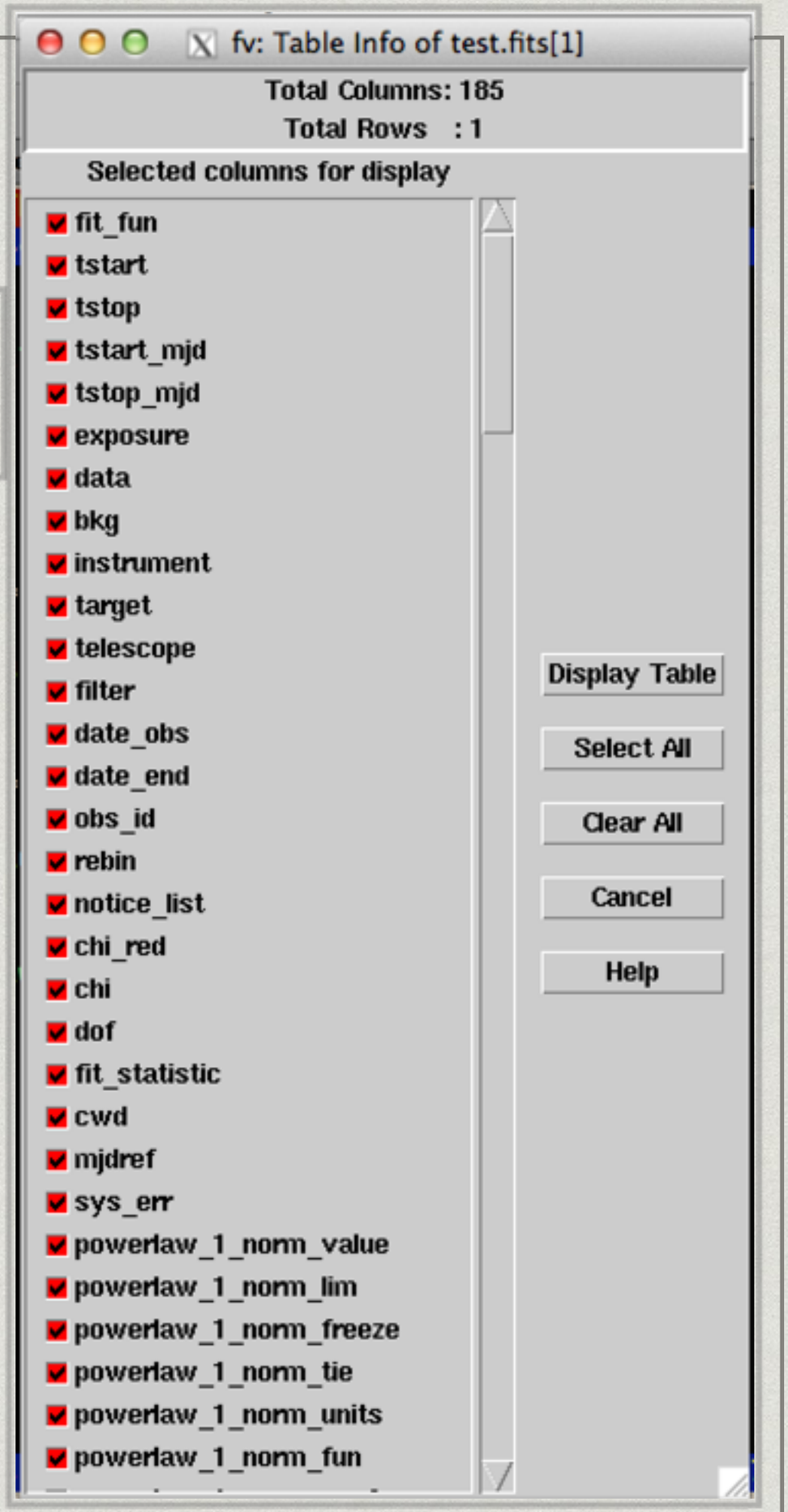
==> get SLANG-MPI-Module:
http://www.sternwarte.uni-erlangen.de/git.public/?p=slmpi.git

fits_save_fit

```
fits_save_fit

 SYNOPSIS
    saves the model and info of the observation to a FITS table
```

- saves bestfit model together with information on grouping, noticing, chi2, confidence intervals, etc.
- `fits_load_fit` uses `fits_save_fit` output to load data and model ==> full recovery
- works with all Xspec models
- more functions in the `fits_**_fit` suite!



mc_sig

Monte Carlo significance of spectral components

```
DESCRIPTION
   This function calculates the significance of a spectral component
   found in real data. During each Monte Carlo loop, spectra data
   without the component are simulated (for each detector) and this
   data are then fitted with a model containing the component (that
   needs to be tested for) and separately fitted with a model without
   it.
   The resulting simulated differences in chi square between these
   fits are returned and compared to the measured difference: the
   number of simulated chi squares below the measured one corresponds
   to the significance, that the spectral component is real (i.e. in
   80 cases out of 100 runs the simulated chi square difference is
   below the measured chi square difference, the significance is 80%).

   If two FITS-files created with fits_save_fit are provided, the
   first includes the model "with" the component to be tested and
   the second one "without" it.

   If only one argument is given, this has to be a temporary directory
   with results of a previous torque run (e.g. if "dontWait" was set).
   The function tries to collect and return the results as usual.

   The returned structure is defined as follows:
     readdchisqr  - the measured difference in chi square
     fakedchisqr  - an array of simulated differences in chi square
     significance - the resulting significance as defined above
```

- see also **emcee**

mc_sig

Monte Carlo significance of spectral components

DESCRIPTION

```
ent

is
at
hout

e

onds
 in
s
80%).
```

EXAMPLE

```
   % FITS-files created by fitting a cutoffpl and iron line to
   % RXTE-PCA, -HEXTE and Swift-XRT data (Rmf_OGIP_Compliance = 0 to
   % load XRT data, see help of fits_load_fit)
   sig = mc_sig("rxte_swift_cutoffpl_ironline.fits",
                "rxte_swift_cutoffpl.fits";
                mcruns = 1, ROC = [2,2,0],
                beforeData = "defineMyModels.sl",
                afterData = "setDataHooks.sl",
                chatty = 2);
```

```
first includes the model "with" the component to be tested and
the second one "without" it.

If only one argument is given, this has to be a temporary directory
with results of a previous torque run (e.g. if "dontWait" was set).
The function tries to collect and return the results as usual.

The returned structure is defined as follows:
  readdchisqr  - the measured difference in chi square
  fakedchisqr  - an array of simulated differences in chi square
  significance - the resulting significance as defined above
```

- see also **emcee**

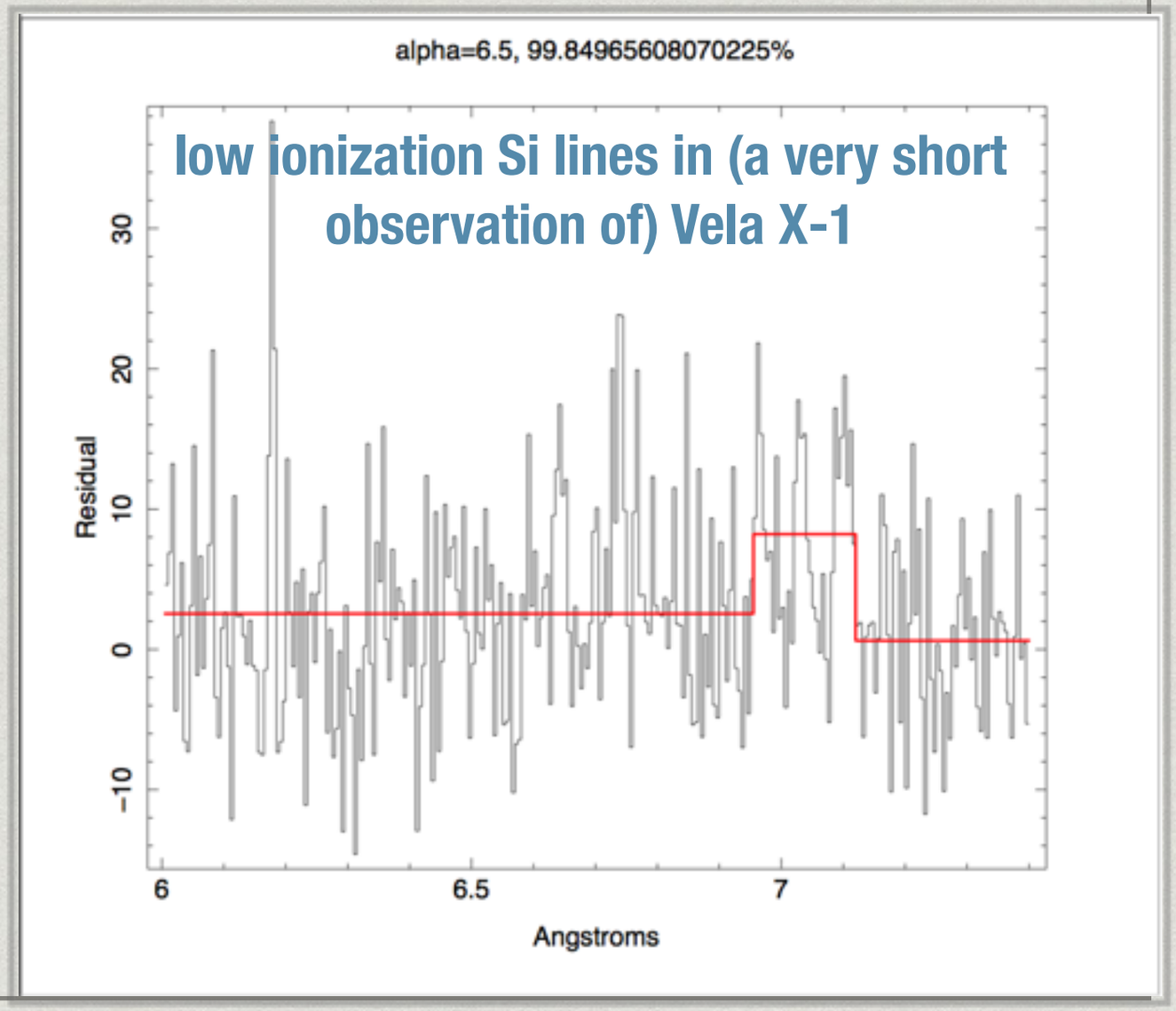
bayesian_blocks

find Bayesian blocks in data

Scargle et al. (Arxiv:1207.5578)
—> the problem of detecting and characterizing local variability in time series and **other forms of sequential data**

==> apply algorithm to residuals of spectral data for a "blind" line search

- http://space.mit.edu/cxc/analysis/SITAR/bb_experiment.html
- see Young et al. 2007 for example on HETGS spectrum of M81



custom line functions

(in development)

aim —> user-friendliness: clear, flexible function containing a sum of easily-accessible lines

`“gauss(1)+gauss(2)+…+gauss(15)” ==> “lines()”`

balance between too simple vs. not flexible enough

==> **`init_line, add_line, rolling_fit,`** etc.

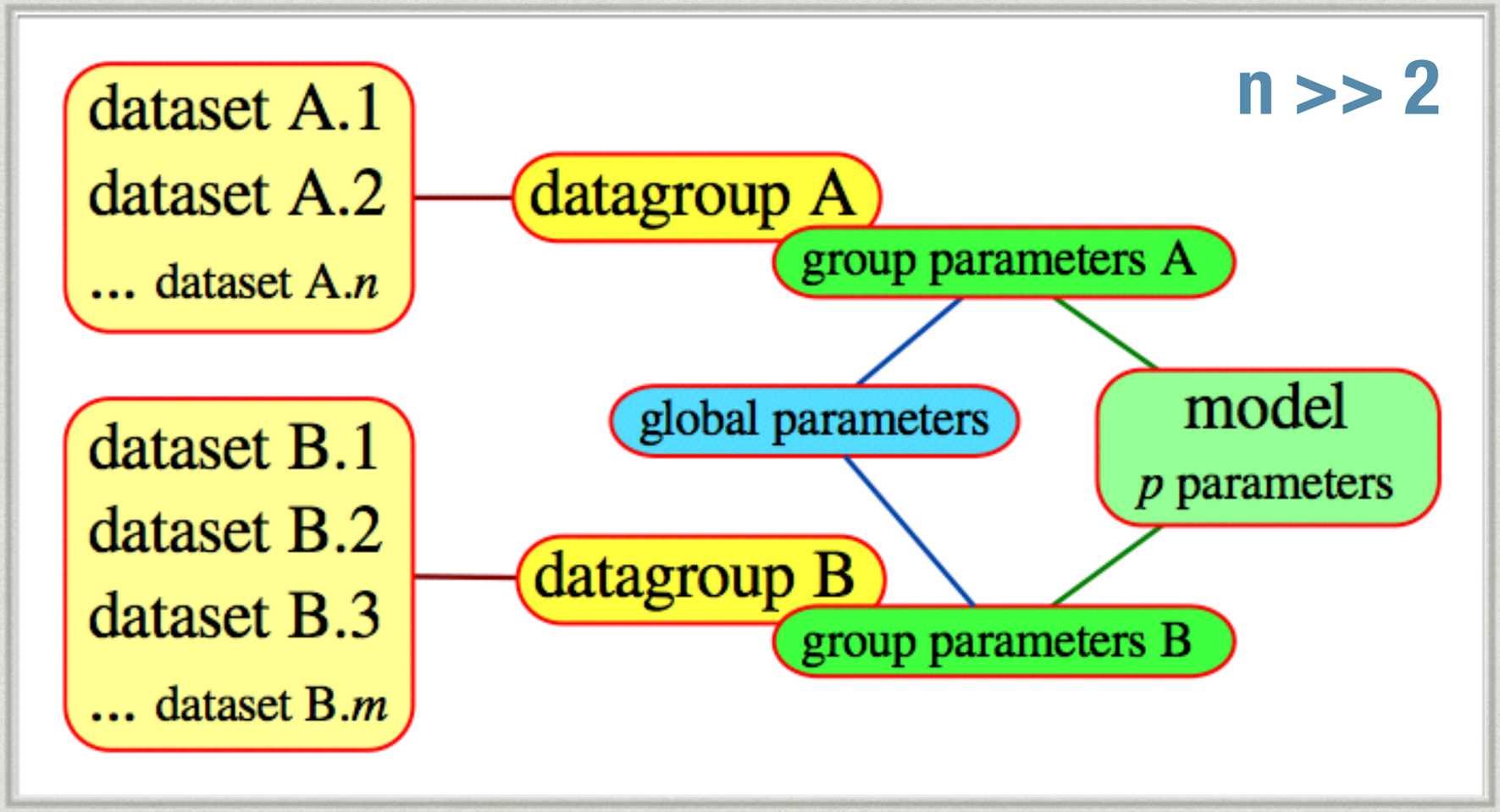
line “database”

```
variable lmbda={
    {"Fe_XXVIa",1.77985,1.65,1.91},
    {"Fe_XXVr",1.85051,1.75,1.95},
    {"Fe_XXVi",1.85744,1.75,1.95},
    {"Fe_XXVf",1.86808,1.75,1.95},
    {"Ca_XXa",3.020,3.015,3.025},
    {"Ca_XXa1",3.018,3.015,3.021},
    {"Ca_XXa2",3.024,3.021,3.027},
```

so far lines accessed individually, next step: line groups?

your chance to give input how things should work!

simultaneous_fit
(Kuehnel et al. 2015, 2016)



possible without this dedicated structure, but highly complex

simultaneous_fit
(Kuehnel et al. 2015, 2016)

Advantages

- Fixed parameters can be determined correctly
- Complicated parameter correlations can be implemented and tested
- Different types of data can be combined
- Parameter degeneracies can be broken
- Reduced number of degrees of freedom

Disadvantages

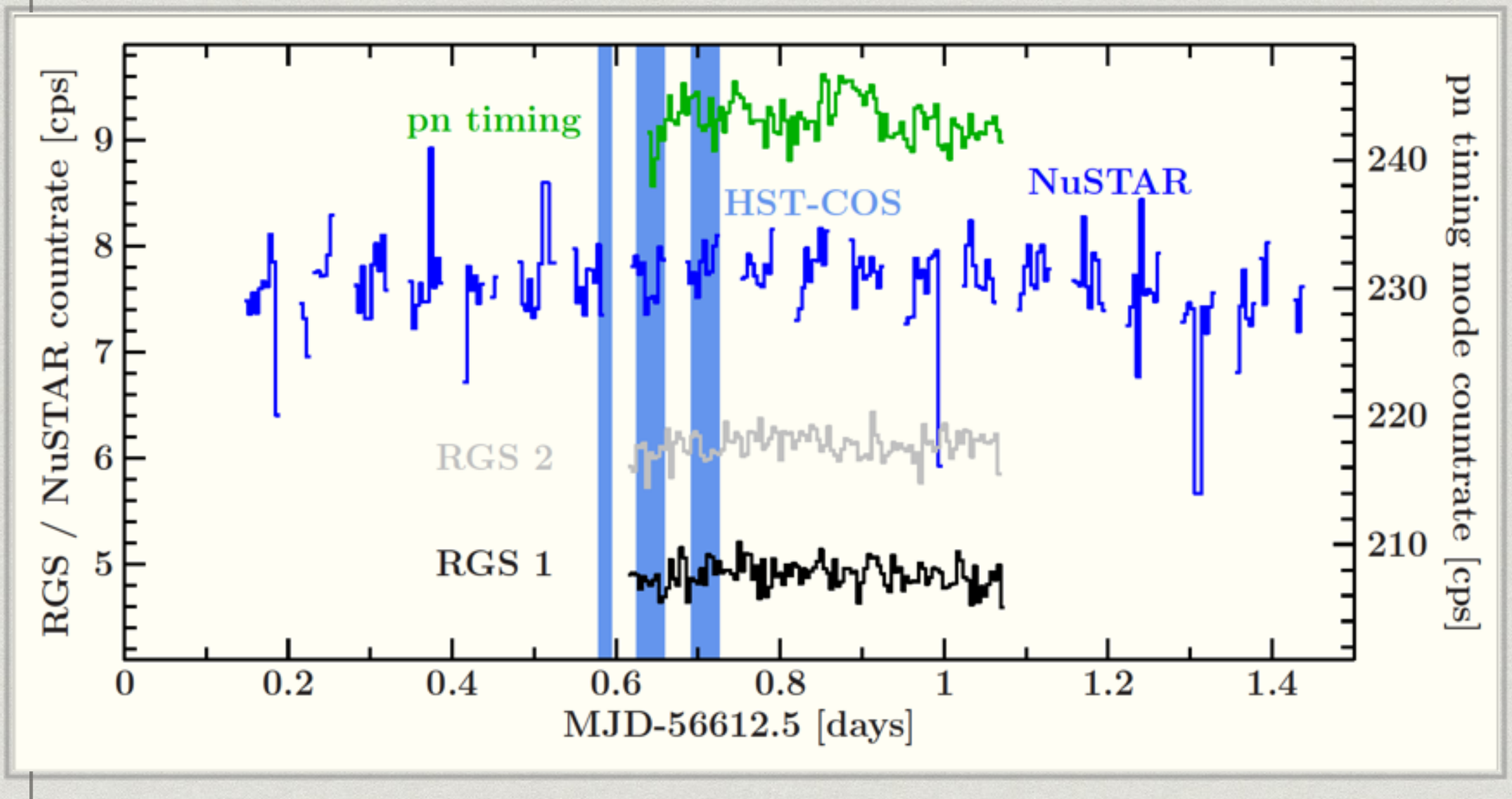
- Increased runtime of fits and uncertainty calculations
- Large memory is needed → multi-CPU calculations required
- Statistical weights of datasets have to be chosen
- Careful handling of fit-parameters required

simultaneous_fit
(Kuehnel et al. 2015, 2016)

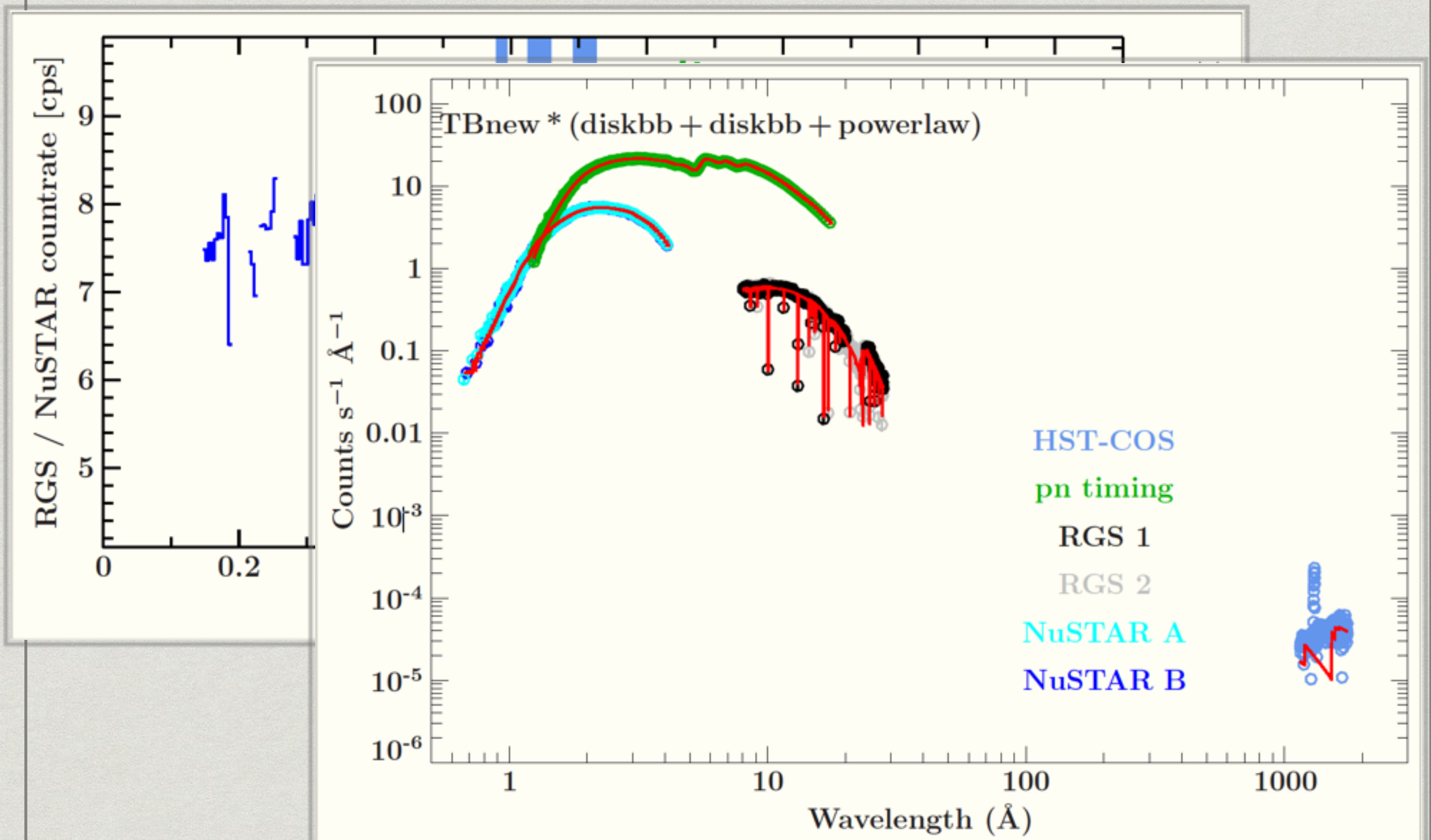
```
isis> simfit = simultaneous_fit();
NOTE: a new version of the add_data-function has been implemented.
      You may set the "old"-qualifier to fall back to the old
      function for compatibility reasons.
isis> print(simfit);
{add_data=&simultaneous_fit_add_data,
 delete_data=&simultaneous_fit_delete_data,
 sort_params=&simultaneous_fit_sort_params,
 apply_logic=&simultaneous_fit_apply_logic,
fit_fun=&simultaneous_fit_fit_fun,
get_par=&simultaneous_fit_get_par,
set_par=&simultaneous_fit_set_par,
freeze=&simultaneous_fit_freeze,
thaw=&simultaneous_fit_thaw,
copy_par=&simultaneous_fit_copy_par,
set_par_fun=&simultaneous_fit_set_par_fun,
set_par_fun_history=&simultaneous_fit_set_par_fun_history
```

for a very careful consideration of fit statistics/goodness of fit, see Kuehnel et al. 2016!

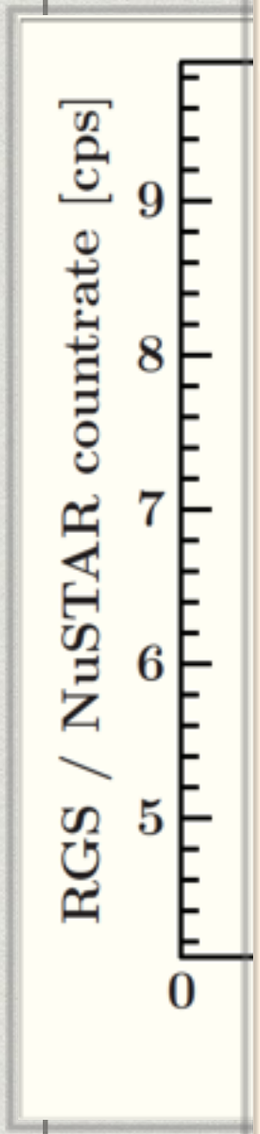
Multiwavelength Datasets



Multiwavelength Datasets



Multiwavelength Datasets




```
isis> help define_counts
define_counts

 SYNOPSIS
    Define a counts-histogram using slang arrays

 USAGE
    s = define_counts (Struct_Type | bins | [lo, hi,] counts [, err])

 DESCRIPTION
    This function provides a way to define a new data-set using
    S-Lang arrays.  As input, it accepts 1) a Struct_Type with
    fields bin_lo, bin_hi, value, err or 2) a list of four
    equal-length arrays with the same data or 3) a single array
    containing only the bin values.  The new data-set is added to
    the internal list just as though the data had been loaded from
    an ascii or FITS data file.  Normally, the function returns the
    integer index of the new data-set.  If the function fails, the
    return value is -1.

    The wavelength grid arrays (bin_lo, bin_hi) and the uncertainty
    (err) arrays are optional.  If the wavelength grid arrays are
    shorter than the counts array (or are missing), they are
    ignored, and the data grid is assumed to be supplied by an RMF.
    If the uncertainty array is shorter than the counts array (or
```

Non spectral data

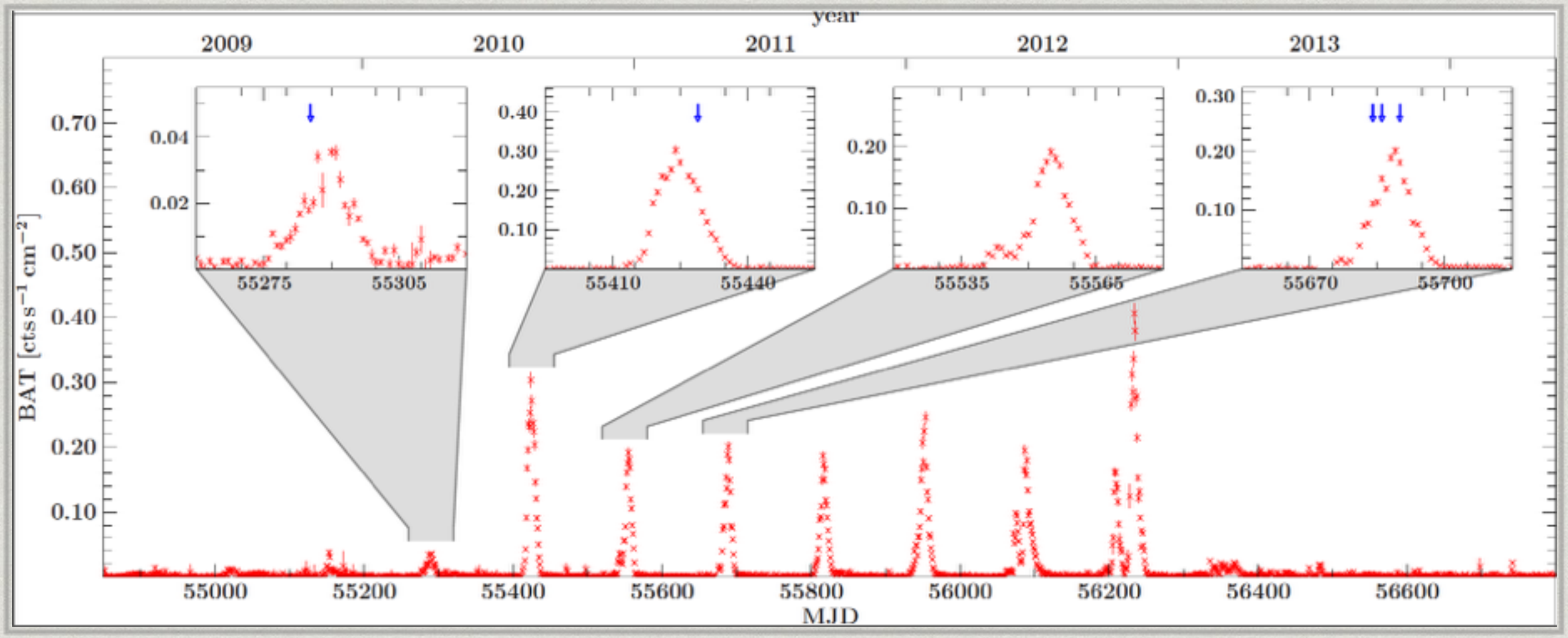
- two suited of X-ray timing tools: *SITAR* & *foucalc*
- arrival time analysis for X-ray pulsars (*atime*_***)
- simple linear fit functions build it
- for everything else: spectra are also just arrays, so define the data as a “spectrum” and fit e.g. a Gaussian

```
%%assume h is the histogram over the (lo,hi) grid
%%and we expect the distribution of the values to be Gaussian

variable hid = define_counts(lo,hi,hist,sqrt(hist));
fit_fun("gauss");
()=fit_pars
```

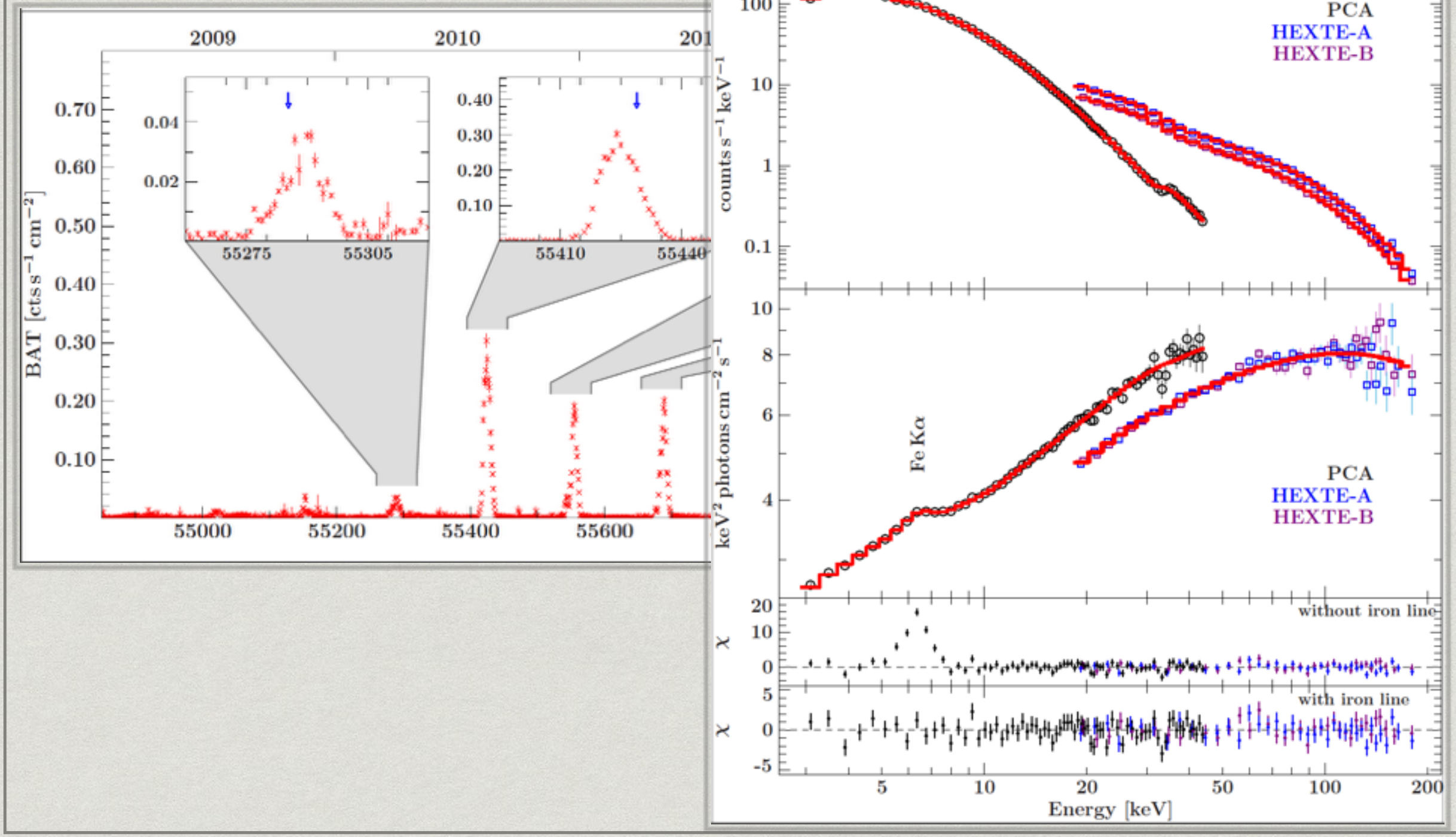
SLxfig

S-Lang functions that automatically run Xfig's fig2dev and LaTeX to produce pdf, (e)ps, png, …



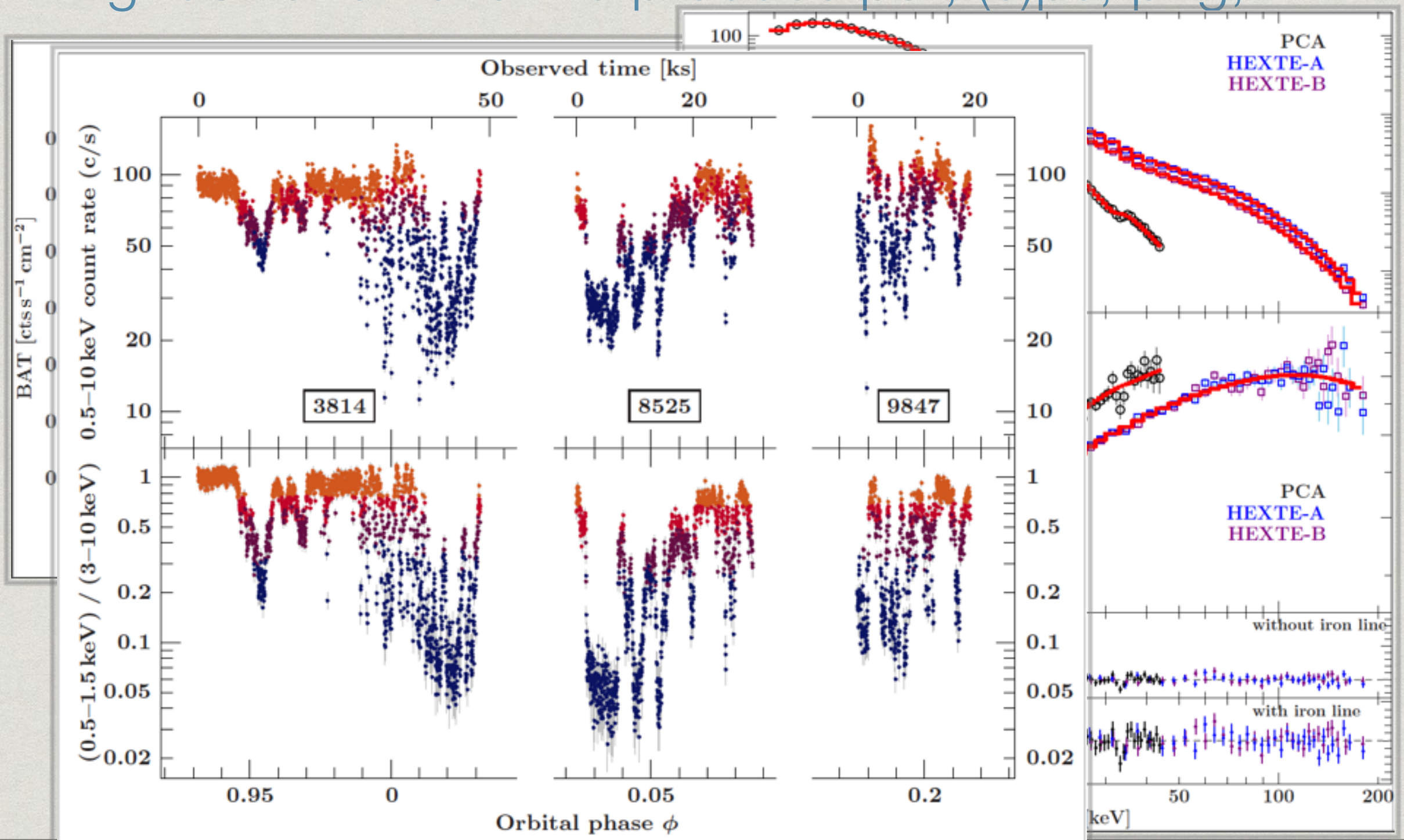
SLxfig

S-Lang functions that automatically run Xfig's fig2dev and LaTeX to produce pdf, (e)ps, png, …



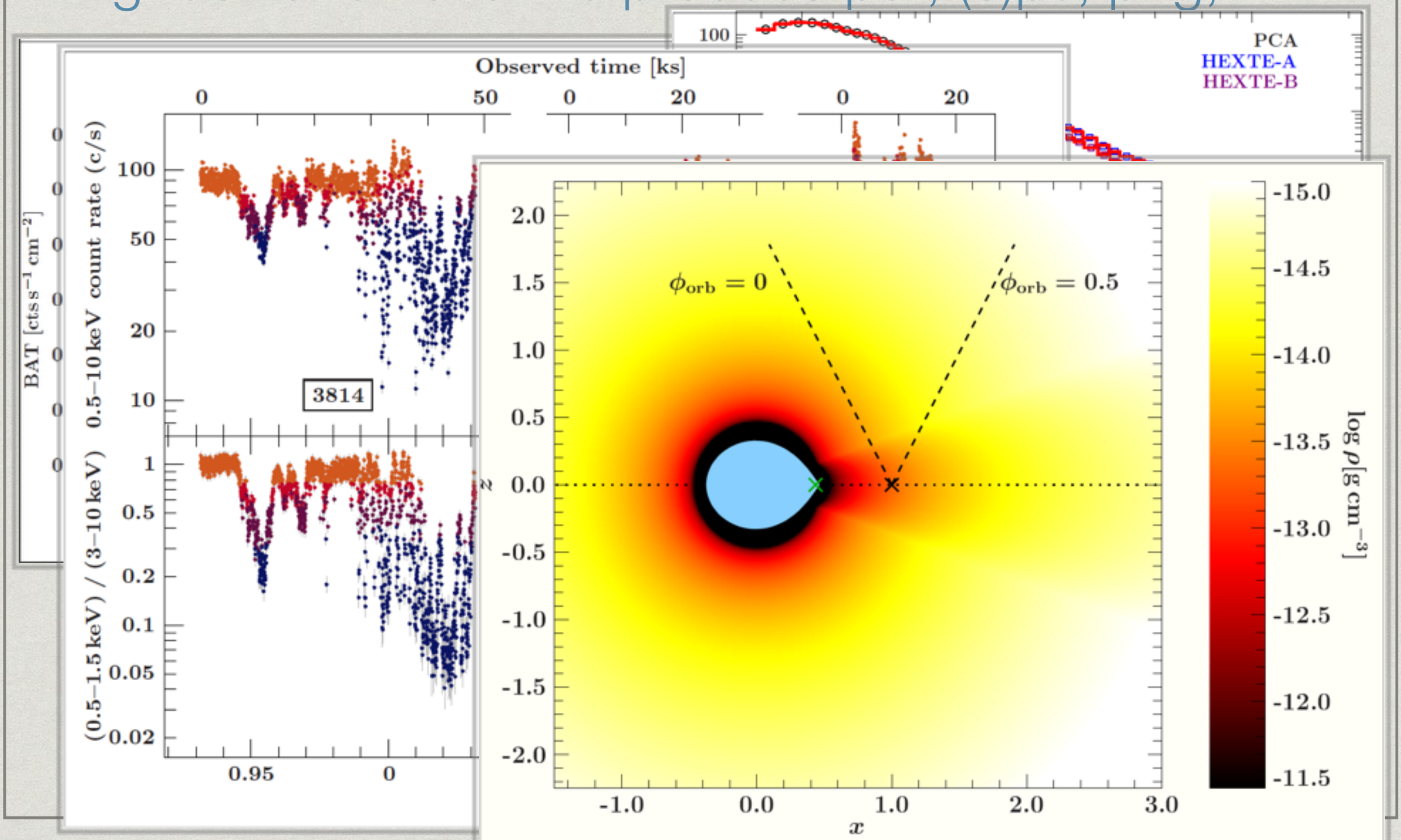
SLxfig

S-Lang functions that automatically run Xfig's fig2dev and LaTeX to produce pdf, (e)ps, png, …



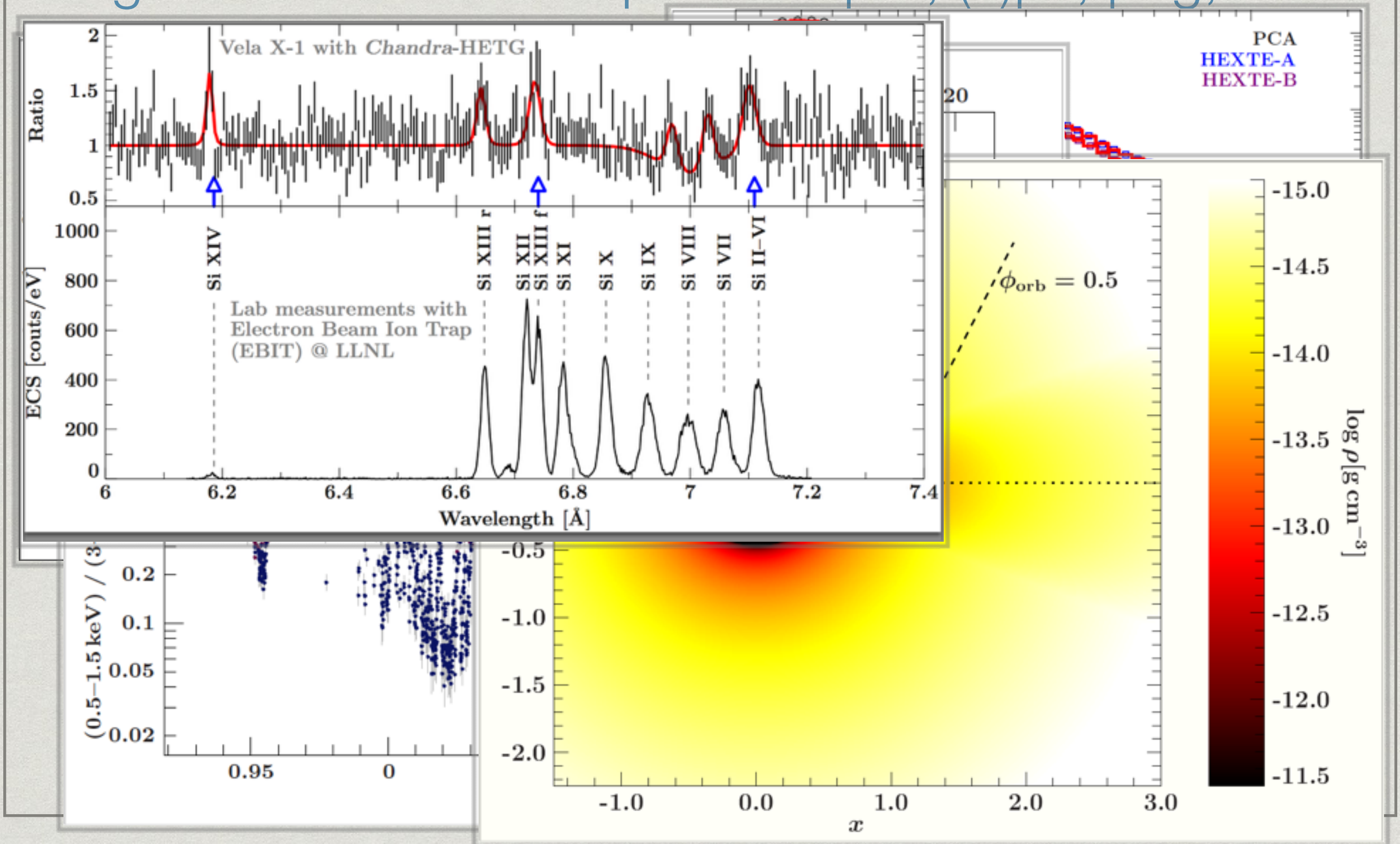
SLxfig

S-Lang functions that automatically run Xfig's fig2dev and LaTeX to produce pdf, (e)ps, png, …



SLxfig

S-Lang functions that automatically run Xfig's fig2dev and LaTeX to produce pdf, (e)ps, png, …



SLxfig

S-Lang functions that automatically run Xfig's fig2dev and LaTeX to produce pdf, (e)ps, png, ...

- plotting in the same environment as your analysis - seamless transition and easy changes
- paper-quality plots that interact with quick plotting routines for spectra
- comprehensive library of plots & code & refs to more pages with both:
  http://www.sternwarte.uni-erlangen.de/wiki/doku.php?id=isis:slxfig
- two examples in the tutorials: light curve & spectra
  ==> should enable an easy start into xfig
  http://space.mit.edu/ASC/isis2015/tutorials.html

ISIScripts & non-HiRes

(random survey of useful/fun functions)

- `SITAR` & `foucalc`: two independent timing toolkits
- `(radius,luminosity,temperature)=zams(mass;z=metallicity);`
- `radio_mod2img`: radio image using a model + beam
- `MJDofDate / dateOfMJD` (Felix's favorites)
- `lumdist`: luminosity distance given z
- `RXTE_ASM_lightcurve`