eimsimbatch

January 12, 2017

Abstract

This is part of the package **eimsim**, which is a collection of routines used to make simulated XMM-Newton x-ray images, to perform source detection on them, and to assess the results.

1 Which documents to read

At last count, there are 25 tasks within the package **eimsim**, each with its own documentation (well, they will have, eventually). However, you don't need to read 25 sets of documentation! I recommend that you read the documentation for only 4 of the tasks, in the following order:

1. **eimsim**
2. **eimsimprep**
3. **eimsimbatch** (ie, the present document)
4. **eimsimreduce**

The documentation for **eimsim** contains all generic information relating to the package as a whole, such as the change history.

2 Description

As described in the **eimsim** documentation, the functionality of package **eimsim** is divided into things that need to be done only once and things which must be done N times for N simulation runs. The 'done once' things can be further divided into those things which must be done before the simulation runs, and those things which must be done afterward. The 'before' things are gathered into **eimsimprep**; the 'N simulation runs' things are to be found within the present task **eimsimbatch**; and the 'afterward' things are performed within **eimsimreduce**.

The task **eimsimbatch** is designed to be run an unlimited number of times, and also if desired simultaneously on separate processors, in order to accumulate many independent results and thus better statistics. Its parameters comprise all those for **eimsim**, plus the following: `nfields`, `startatn`, `nextonfail` and `cleanup`.

2.1 Basic functioning

The functions of **eimsimbatch** are best illustrated by example. Suppose you follow the cookbook example described in the **eimsim** documentation, which involves running **eimsimbatch** with all its parameters left at default. What this will achieve is a single invocation of **eimsim**, and thus a single output source list which records the detections in just 1 set of images. There are not likely to be more than 100 or so detections in this list unless you have used a large number of different image templates in the input set, which memory-wise is an inefficient thing to do. A better thing to do is specify some number of iterations via the `nfields` parameter, eg

```
eimsimbatch nfields=100
```

This will generate an ensemble of 100 lists of detections from the same starting set of template images, which will be merged together into one if you then subsequently run **eimsimreduce** as the cookbook advises. The **eimsimbatch** task achieves this by calling **eimsim** `nfields` times, with the supplied value of the –idnumber parameter of **eimsim** starting at 1 and incrementing to `nfields`. The –idnumber value is written by **eimsim** to the column `FIELD_N` of the output source lists.

Suppose you decide that you would like to add another 100 runs to the ensemble. You might be tempted to simply repeat the last **eimsimbatch** command. DO NOT DO THIS - you will just overwrite your previous 100 files. The correct way to do it is as follows:

```
eimsimbatch nfields=100 startatn=101
```

Ok, now you get serious - you decide that nothing less than 10^4 runs will satisfy you. Running 10^4 sequential invocations of **eimsim**, at about 10 min per, will take quite a while however. You can get around this if there is more than one processor available to you. Suppose you have 5 processors. Assign each processor a stream number, ssh to remote nodes if you need to, and on each one, set the following command going:

```
eimsimbatch nfields=2000 streamnumber=<1 to 5>
```

You may run all these 5 streams from the same directory if your hardware setup allows this: all tasks in package **eimsim** have been designed to avoid crosstalk by including the stream number and id (ie, field) number in all relevant file names.

2.2 Errors and cleaning up

Suppose, during one of the iterations of **eimsim**, **eimsim** fails with an error condition. What should **eimsimbatch** do? There are two alternatives, which you can select between via use of the parameter `nextonfail`. As the name implies, if this is set 'yes' (the current default value), a failure during one iteration of **eimsim** will only cause **eimsimbatch** to skip a beat, then proceed to the next iteration. If set to 'no', a single error will stop everything.

The `cleanup` parameter, if set to 'yes', calls **eimsim** twice: once with the parameters `entrystage` and `finalstage` retaining their passed-through values; the second time, both parameters are set to 'cleanup'. This makes **eimsim** delete any intermediate files. I recommend keeping `cleanup` set to 'yes' if you have a significant number of iterations to do.

2.2.1 Lock and stop files

Task **eimsimbatch** writes a lock file in the PWD when it commences running. The filename of this lock file has the format 'lock_<streamnumber>'. This helps to prevent any other invocation of **eimsimbatch** with the same `streamnumber` running in the same PWD. It isn't foolproof, since it sets up a 'race condition'. (Google if curious.) The task deletes the lock file at the end of operations. If the task fails with an error, this lock file remains undeleted and will prevent renewed invocation of the task with the same `streamnumber` until it is deleted by hand.

If you want to halt **eimsimbatch** at the end of a processing loop, halt it 'nicely' that is, without resorting to control-Cs or 'kills', this may be accomplished by writing a stop file, of filename format 'stop_<streamnumber>', into the PWD. Doing 'touch stop_<streamnumber>' is sufficient. The task deletes the stop file before exiting.

3 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints
streamnumber	no	int	1	

If you want to run several instances of **eimsimbatch** (thus **eimsim**) at the same time from the same PWD, you should give them different stream numbers via this parameter. The `streamnumber` is included in all intermediate file names.

nfields	no	int	1	

If you want each invocation of **eimsimbatch** to accumulate an ensemble of N statistically independent simulations, set `nfields` to N. The task will then call **eimsim** N times, incrementing the value of the parameter –idnumber on each occasion. The starting value of –idnumber is taken from parameter `startatn`. The –idnumber is included in all intermediate file names.

startatn	no	int	1	

This is useful if you have run **eimsimbatch** at least once before, and now wish to run it again so as to add further simulation runs to the ensemble. Suppose your previous runs at the specified value of `streamnumber` have accumulated N simulations: you should therefore set `startatn` to $N + 1$ for your next run at the same `streamnumber`. WARNING! If you do *not* set `startatn` to an appropriate value before restarting **eimsimbatch**, your previous data will be silently overwritten.

nextonfail	no	bool	yes	

If 'yes', a failure in any iteration of **eimsim** will cause **eimsimbatch** to proceed to the next iteration; if set to 'no', a failure of **eimsim** will cause **eimsimbatch** also to fail.

cleanup	no	bool	no	

Set this to 'yes' if you want all intermediate files except the source lists to be cleared away at the end of processing.

obsidroots	no	string	.	

See **eimsim** documentation.

entrystage	no	string	makesimlist	

See **eimsim** documentation.

finalstage	no	string	compare	

See **eimsim** documentation.

refband	no	string	1	

See **eimsim** documentation.

prdssubdir	no	string	product	

See **eimsim** documentation.

simopsubdir	no	string	sim_output	

See **eimsim** documentation.

simgensubdir	no	string	sim_generic	

See **eimsim** documentation.

srcspecset	no	dataset	srcspec.fits	

See **eimsim** documentation.

withsimsources	no	bool	yes	

See **eimsim** documentation.

energyfraction	no	real	0.95	

See **eimsim** documentation.

fluxcutoff	no	real	2.0e-15	

See **eimsim** documentation.

withfluxoffset	no	bool	no	

See **eimsim** documentation.

fluxoffset	no	real	0	

See **eimsim** documentation.

dettaskstyle	no	string	auto	user—auto

See **eimsim** documentation.

dettask	no	string	eimsimdetect1xmm	

See **eimsim** documentation.

withdetentrystage	no	bool	no	

See **eimsim** documentation.

detentrystage	yes	string		

See **eimsim** documentation.

withdetfinalstage	no	bool	no	

See **eimsim** documentation.

detfinalstage	yes	string		

See **eimsim** documentation.

astest	no	bool	no	

See **eimsim** documentation.

4 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

code *(error)*
message

code *(warning)*
message
corrective action: action

5 Input and Output Files

See the relevant sections in the **eimsim** documentation.

References