Scripts and my editing workflow.

Keith Shortridge, AAO

1st April 2016

These are notes to accompany the Python scripts I've been using for the ADASS 2015 Proceedings, in the hope that others may find them useful. They've been tuned to work the way I do, so they really only make sense if I explain a bit about the workflow I've developed. It more or less follows Nuria's guidelines.

Caveat: These scripts work on the relatively few papers I’ve handled so far. Authors find all sorts of ways to confuse scripts like these, so I make no promises that they will handle any other papers properly. But if they don’t, let me know.

This description is for the ADASS 2015 Proceedings, but the scripts should work for subsequent conferences as well, and I hope they might be found useful. The only thing here that is 2015-specific is the name of the adassXXVreference.bib file, and changing this requires only a one-line change to one of the scripts.

# File layout.

I have a directory called Proceedings, which contains:



a) The Python scripts.

b) A directory called Work which contains:

1) My personal master reference file, adassXXVreferences.bib (this has all the references from the papers I've finished processing). This needs to exist, at least as an empty file, or it will not be updated by the scripts. (I should fix that.)

2) A directory called Volume, which contains:

The files asp2014.bst and asp2014.sty.

3) A copy of Nuria’s subjectKeywords.txt file, with all the keywords from previous ADASS proceedings – although I removed the first part, so it just has the hierarchical keyword list (see the section on subject indexes, later).

4) newKeywords.txt contains all the new keywords I’ve added to the papers I’ve finished processing (and which weren’t in the original subjectKeywords.txt file). If this doesn’t exist, the scripts will create it.

5) A directory called Authors, which contains one directory for each paper I’ve worked on so far, named with the paper ID and author, eg O1-2\_Aloisi, O6-3\_Lorente, etc. Obviously, this started out empty.

c) A directory called UploadFiles which contains all the files uploaded from CloudStor, so it contains all the dated folders with the submitted archive files (mostly tar files, some zip files). Note that the scripts do not assume this layout for UploadFiles – they just assume that the submitted archive files will be somewhere in the UploadFiles directory tree.

So UpLoadFiles looks like:



and Authors looks like:



(B3\_Shortridge isn’t on my editing list, but I use it as a paper to experiment on when I test the scripts.)

The scripts assume this layout. I work entirely from either the Proceedings directory itself, or from the individual Paper directories in Work/Authors. I tend to keep one terminal window open in Proceedings and another open in the relevant directory in Work/Authors. I have the Proceedings directory in my execution path, so the Python scripts can be found easily.

# Getting the files for the paper. (Setup.py)

With my default set to the Proceedings directory, I use Setup.py to fetch the files for the next paper on my list. For the sake of this demo, I’ll use my own Birds of a Feather (session B3) paper. This isn’t necessarily typical: it has a lot of references, and no graphics. But it will do. (I’ve trimmed down some of the list of references in the output that ‘s shown here, just to save space.)

~/Proceedings > ./Setup.py B3 Shortridge

The script looks in UploadFiles to find any archives with the paper ID (in this case B3) in their name, then looks for the one containing the file with the latest date. It can handle .tar, .tar.gz or .zip archives, and can handle a mix of them. (It looks into the contents of the archives to get the dates, and ignores the file and directory names.)

Found file ./UploadFiles/321378619452579185/B3\_v1.tar.gz

Latest file date/time = 2015-10-23 13:59

Found file ./UploadFiles/323615474452781851/B3\_v2.tar.gz

Latest file date/time = 2015-11-18 10:49

Assuming B3\_v2.tar.gz is later than B3\_v1.tar.gz

Archive file to be used is ./UploadFiles/323615474452781851/B3\_v2.tar.gz

Now it unpacks the latest dated archive into a new directory called, in this case B3\_Shortridge in Work/Authors. (If this already exists, the old one is renamed and saved.) It then looks at what’s there. It looks for the main .tex file, and a copyright form, which it renames as required. There are no asp2014.sty or .bst files supplied, so it creates symbolic links to the files in Proceedings/Work/Volume, which (for me) is the easiest way of handling these files. It looks for the .bib file, if any, and renames it as adassXXVreferences.bib.

Found B3.tex in Work/Authors/B3\_Shortridge

Installing symbolic link to asp2014.sty

Installing symbolic link to asp2014.bst

B3\_Copyright\_Form.pdf appears to be a copyright form

Renamed B3\_Copyright\_Form.pdf as copyrightForm\_B3\_Shortridge.pdf

Renaming B3.bib to adassXVVreferences.bib

Now it looks at the main .tex file. It fixes the unnecessary ./ in any \usepackage{./asp2014} line, and fixes the any \bibliography line that used the now renamed .bib file (if there was one) so it now uses adassXXVreferences.

Fixed \usepackage{./asp2014} directive

Modified \bibliography{B3} to use adassXXVreferences.bib

It checks to see if any non-standard packages are being used (this example doesn’t use any).

It looks for the references defined in the .bib file:

References in adassXXVreferences.bib :

2011ASPC..442...53A

2012ASPC..461..283A

2012arXiv1204.3055M

2015ASPC..495..551R

O11-4\_adassxxv

P092\_adassxxv

P056\_adassxxv

2015ASPC..495..527S

1999ASPC..172..487P

1995ASPC...77..199A

Starlink\_SSN\_20.3W

and those cited in the main .tex file. It also checks at this point for use of directives such as \cite, and for the use of \bibitem references and warns if it finds any.

References cited in /Users/ks/Proceedings/Work/Authors/B3\_Shortridge/B3.tex :

1999ASPC..172..487P

Starlink\_SSN\_20.3W

2011ASPC..442...53A

1995ASPC...77..199A

2015ASPC..495..527S

P056\_adassxxv

2011ASPC..442...53A

2012ASPC..461..283A

O11-4\_adassxxv

2012ASPC..461..283A

2012arXiv1204.3055M

2015ASPC..495..551R

P092\_adassxxv

Then it checks to see if these are consistent. Are all the references in the .bib file actually cited in the .tex file? Are all the references in the .tex file defined in the .bib file?

All Bib file references used in .tex file

All .tex file citations defined

Finally, the script checks to see what .eps files have been supplied, and looks to see what graphics files are used by the .tex file. It checks these to see if there are any missing or unused files. This paper doesn’t have any.

.eps files supplied:

No .eps files found

No graphics files used by the .tex file

At the end of this, the files for the paper are in the required subdirectory of Work/Authors, pretty much anything that can be done automatically to the file has been done, and the real editing can start.

## Tricks

If I ‘ve been editing for a while and want to see the original version, I can just repeat the Setup.py command but with a dummy name, eg “Setup.py B3 Dummy” and the script will fetch the files again, but this time into Work/Authors/B3\_Dummy. Of course, it has made its automatic changes to the files, but if you need the actual raw files you can look at the terminal output to find where it prints the absolute pathname of the archive file it used, cut that out and untar or unzip it explicitly into a temporary directory.

Occasionally, Setup.py won’t find an archive file supplied by an author. This is usually because they’ve not followed the naming convention properly – something like P0123.tar instead of P123.tar – and the easiest thing is simply to go into the UploadFiles directory and correct the file name. Setup.py will handle a number of failures to follow the conventions (it will find P87.tar even if it’s looking for P087.tar, and it will find O1.4.tar even if it’s looking for O1-4.tar), but some things still get past it.

# Editing steps

This is my sequence.

Set my default (usually in a second terminal window) to the Work/Authors directory containing the files for the paper.

Open the copyright file and see if it really is a copyright file, and if it’s been signed properly (not all have).

Run LaTeX on the file. (I use TeXworks on the Mac, and actually run pdflatex

If a .bib file has been supplied (it will now have been renamed as adassXXVreferences.bib) run BibTeX.

Run LaTeX a couple more times to sort out any citations.

Open the resulting PDF file (since I was running pdflatex) and print a copy (I need something to scribble on when I edit).

Check that it has the right number of pages.

Check that the author list is correct and the presenting author is first.

See if there were any LaTeX errors (overfull \hbox etc, undefined references) or warnings (underful \hbox etc).

Read the paper looking for dodgy grammar and unclear text or illustrations.

Run a spelling checker (I don’t have a good LaTeX-aware checker, so I just read the .tex file into Textedit and use that. I find I can ignore the errors from LaTeX directives easily enough, but a good LaTeX-aware checker would be nice.)

Fix any trivial problems, send non-trivial ones back to the author. (I define non-trivial as ones I can’t fix in less time than it takes to write a detailed e-mail explaining the problem, with an allowance for the time spent waiting for a reply etc.)

Check the names of the graphics files used – these are printed out by Setup.py, or by the RefCheck.py script described in the section on references – and make sure they follow the required convention. If necessary, change them.

Add %\aindex entries, one for each author.

Add %\ssindex entries, as required.

The next sections go into a bit more detail about references (which I’ve found the most time-consuming part of the process), and adding index entries.

After that, I run a Finish.py script to tidy things up. That’s described after the sections on references and indexes.

# Fixing references

This is worth a section to itself. The Setup.py script warns of references that aren’t defined in either a .bib file or using \bibitem. It also warns of usages such as \cite rather than \citep and \citet. And it warns of .bib and \bibitem entries that aren’t actually used, and prints out the reference IDs that the author has used. This gives a pretty good idea of the scope of any problems with references.

If the author has supplied just a few \bibitem references, it usually isn’t a huge job to generate a new adassXXVreferences.bib file from them, especially if they can be found in ADS. URL references should probably be footnotes, but can be @online references in the .bib file. You can wait quite a while for an author to do this!

\cite references need to be reworked. If the author hasn’t used ADS-style reference IDs (things like 2011ASPC..442...53A) what they have used needs to be reviewed and possibly changed, both in the .bib and the .ref.

Given the changes that tend to be made, I found it convenient to be able to re-run the reference check done by the Setup.py script on the latest versions of the .bib and .tex files. There is a RefCheck.py script that does this. It has to be run with the default set to the directory containing these files. Here it is being run on a different paper:

> cd Work/Authors/P009\_Ballester

> RefCheck.py P009

References in adassXXVreferences.bib :

ballester2011

freudling2013

2014ASPC..485..277B

2010SPIE.7737E..19P

References cited in /Users/ks/Proceedings/Work/Authors/P009\_Ballester/P009.tex :

2010SPIE.7737E..19P

2014ASPC..485..277B

freudling2013

ballester2011

All Bib file references used in .tex file

All .tex file citations defined

Graphics files used by /Users/ks/Proceedings/Work/Authors/P009\_Ballester/P009.tex :

P009\_f4.eps

P009\_f1.eps

P009\_f2.eps

P009\_f3.eps

.eps files supplied:

P009\_f1.eps

P009\_f2.eps

P009\_f3.eps

P009\_f4.eps

All .eps files in the directory are used by the .tex file

All graphics files used by the .tex file are supplied

Note that RefCheck.py checks the graphics files as well as the references. This paper now passes these checks, and you can see the reference IDs that have been used. These are now a mixture of ADS standard references and references in the form <author><year>, which we are assuming is acceptable. (RefCheck.py also checks for the use of non-standard packages, but this example paper doesn’t use any packages other than the standard asp2014 package.)

One question is: what to do with .bib files that have unused references? In particular, the case where we have obviously been sent the author’s master .bib file used for many different papers, including this one.

For this, there is the TrimBibFile.py script. Run from the directory containing the .tex and .bib files, this simply comments out all the unused references in the .bib file. Here it is being run on a different paper, one with just one unused reference in the .bib file:

TrimBibFile.py O12-6

References in adassXXVreferences.bib :

Bowden2009

1974A&AS...15..417H

2007ASPC..376..127M

Momjian2011

Momjian2012

2011ApJ...739L...1P

Satterthwaite2015

Torvalds1992

Keeping reference Bowden2009

Keeping reference 1974A&AS...15..417H

Keeping reference 2007ASPC..376..127M

Keeping reference Momjian2011

Keeping reference Momjian2012

Keeping reference 2011ApJ...739L...1P

Commenting out unused reference Satterthwaite2015

Keeping reference Torvalds1992

Note that BibTeX has an odd attitude to comments. It treats anything outside an @stylename{…} block as a comment, and you can’t just stick a ‘%’ in front of the ‘@stylename’ to comment it out. It treats the ‘%’ as a comment (not because it’s a ‘%’, which *isn’t* a comment character to BibTeX, which doesn’t *have* a comment character, but because it’s outside the @stylename{…} block), and processes the @stylename{…} block as before. What TrimBibFile.py does is replace the ‘@’ in the block in question with ‘\_AT\_’. This is all that’s needed to comment out the block, but to make it stand out, it also puts a ‘%’ in front of every line in the block. (TeXworks then colour-codes these as comments, apparently being unaware of BibTeX’s odd commenting conventions.) At a later stage, another trivial script could remove these commented blocks if necessary.

# Author index entries

I eventually wrote a script to handle the author index entries. This is Aindex.py. You run it from the directory containing the files for the paper (although the only one it looks at is the main .tex file). It looks at the \author directive near the start of the .tex file and parses it to generate the required commented-out \aindex entries.

Here it is running on a sample paper with reasonable number of authors:

MacBook Work/Authors/P071\_McGlynn > Aindex.py P071

Generating author index entries for paper P071

%\aindex{McGlynn,~T.}

%\aindex{Smale,~A.}

%\aindex{White,~R.}

%\aindex{Donaldson,~T.}

%\aindex{Aloisi,~A.}

%\aindex{Mazzarella,~J.}

%\aindex{Ebert,~R.}

%\aindex{Imel,~D.}

%\aindex{Berriman,~G.~B.}

%\aindex{Teplitz,~H.}

%\aindex{Groom,~S.}

%\aindex{Desai,~V.}

The script doesn’t attempt to insert these entries automatically into the .tex file. This is because parsing the author list turns out to be quite a tricky exercise, complicated mainly by lists that don’t *quite* follow the rules about serial commas and the use of ‘and’ between each author. So I just print out the names, and you can copy these from the terminal window and paste them into the .tex file. As you do so, have a look to see if any look odd. It’s quite a good idea to count the number of authors the script finds and compare them with the author list in the file.

To date, whenever there has been a problem with the author list, the script does at least say something, usually suggesting there may be a missing serial comma or a missing ‘and’. It isn’t always right about just what the problem is. A lower case initial is also a give-away. Here’s an example of one it gets wrong:

MacBook Work/Authors/P056\_Jenness > Aindex.py P056

Generating author index entries for paper P056

%\aindex{Jenness,~T.}

%\aindex{Team,~f.~t.~L.~D.~M.}

\* Note: 'and' missing from last of multiple authors \*

\* Initial letter in 'for' is in lower case \*

\* Initial letter in 'the' is in lower case \*

In this case, the author list was actually:

\author{Tim~Jenness,$^1$ for the LSST Data Management Team

The script does know about a number of things I didn’t think I’d have to bother with when I first wrote it, like people with hyphenated first names like Jean-Luc Picard, people with names like van der Waals, and people with Jr. as a suffix, and people who are the nth of their line, but it can still be caught out.

# Subject index entries

I have a fairly trivial but very useful script to simplify the insertion of subject index entries.

In the section on files, I described having in the Work directory two files called subjectKeywords.txt and newKeywords.txt. The first is the list of keywords supplied by Nuria based on the previous ADASS proceedings, with the heading section removed, so it starts like this:

> more Work/subjectKeywords.txt

adaptive optics

algorithm

analysis

artificial skepticism

A'Trous wavelet reconstruction

automatic

basket-weaving

CLEAN

ClumpFind

clustering

spatial

control

and so on.

The newKeywords.txt file contains all the subject keywords I’ve used that weren’t in that subjectKeywords.txt file. I don’t change subjectKeywords.txt file at all (other than the initial trimming of the header section).

I have a simple script called Index.py that searches both these files for entries that match strings supplied on the command line. So, if, for example, you think a paper needs a subject index entry for the VLA, you can try (with your default set to the Proceedings directory):

> Index.py vla

Looking for index entries matching 'vla'

%\ssindex{archives!individual!EVLA}

%\ssindex{observatories!ground-based!EVLA}

%\ssindex{observatories!ground-based!VLA}

3 entries in master index

No entries in new index

What I find most useful about this script is that it supplies the strings you need to put into the text as commented-out \ssindex entries. You select the one you want in the terminal window and paste it into the .tex file. Note that it needs to go in as a separate line, otherwise having it commented is going to cause problems. It also makes it much easier to find automatically later when it has to be uncommented.

If the entry you want isn’t there, which will happen if your search string is completely new, you can use a similar string to see what similar entries have been used. Say you want an entry for the TAIPAN instrument:

> Index.py TAIPAN

Looking for index entries matching 'TAIPAN'

No entries in master index

No entries in new index

OK, nothing. But it is an instrument, so what about:

> Index.py instrument

Looking for index entries matching 'instrument'

%\ssindex{instruments}

%\ssindex{instruments!bolometer}

%\ssindex{instruments!camera}

%\ssindex{instruments!camera!submillimeter array}

%\ssindex{instruments!individual}

%\ssindex{instruments!individual!2dF}

…..

%\ssindex{instruments!individual!WFC3}

%\ssindex{instruments!individual!WFPC2}

%\ssindex{instruments!individual!X-shooter}

%\ssindex{instruments!interferometer}

%\ssindex{instruments!multibeam}

%\ssindex{instruments!polarimeter}

50 entries in master index

%\ssindex{instruments|individual|HAWK-I}

%\ssindex{instruments|individual|VIMOS}

%\ssindex{instruments|individual|VISIR}

3 entries in new index

I had to edit out most of the individual instruments. There are a lot of entries. From that, you can cut and paste one and edit it, probably as

%\ssindex{instruments!individual!TAIPAN}

of course, if you mean TAIPAN the survey, that’s different.

# Finishing a paper

Once the editing for a paper is complete, a number of things need to be done. Those \aindex and \ssindex entries need to be uncommented, and a number of other items such as \begin{document} and \end{document} need to be commented.

What’s more, the entries in adassXXVreferences.bib for this paper need to be added to the master version of this file higher up in the directory structure, and (amongst other things, for the purposes of the Index.py script) the new subject entries need to be added to the newKeywords.txt file.

It’s much easier to do this with a script, and I use one called Finish.py.

Here’s Finish.py at work on yet another paper:

> Finish.py O5-3 Gabriel

Processing paper O5-3 by Gabriel

Commenting and uncommenting O5-3.tex as required

Commenting out: \documentclass[11pt,twoside]{article}

Commenting out: \usepackage{asp2014}

Commenting out: \aspSuppressVolSlug

Commenting out: \begin{document}

Uncommenting: %\aindex{Gabriel,~C.}

Uncommenting: %\aindex{Rosen,~S.}

Uncommenting: %\aindex{Webb,~N.}

Uncommenting: %\aindex{Rodriguez,~P.}

Uncommenting: %\aindex{Ojero-Pascual,~E.}

Uncommenting: %\aindex{Perea,~J.~P.}

Uncommenting: %\ssindex{observatories!space-based!XMM-Newton}

Uncommenting: %\ssindex{techniques!source detection}

Uncommenting: %\ssindex{applications!SAS}

Uncommenting: %\ssindex{data!pipelines!processing}

Uncommenting: %\ssindex{computer languages!Perl}

Uncommenting: %\ssindex{databases!query language!SQL}

Uncommenting: %\ssindex{Centre de Donn\'ees astronomiques de Strasbourg (CDS)}

Uncommenting: %\ssindex{astronomy!point spread function (PSF)}

Uncommenting: %\ssindex{catalogues!individual!2MASS}

Uncommenting: %\ssindex{catalogues!individual!SDSS}

Uncommenting: %\ssindex{catalogues!individual!USNO-B Catalog}

Uncommenting: %\ssindex{catalogues!individual!3XMM}

Uncommenting: %\ssindex{astronomy!astrometry}

Commenting out: \end{document}

Saving original O5-3.tex as O5-3.tex\_old

Checking subject index entries

Reading master keyword list from /Users/ks/Proceedings/Work/subjectKeywords.txt

Adding any new keywords to /Users/ks/Proceedings/Work/newKeywords.txt

Entries for this paper already in /Users/ks/Proceedings/Work/newKeywords.txt will be replaced

Adding the following entries to Work/newKeywords.txt:

applications

SAS

catalogues

individual

3XMM

Previous new keyword file saved as /Users/ks/Proceedings/Work/newKeywords.txt\_old

Keywords for this paper added to /Users/ks/Proceedings/Work/newKeywords.txt

Will append local .bib file to master .bib file

Master .bib file already has entries for this paper.

These will be replaced

Adding the following references from the local .bib file:

jansen2001

gabriel2004

read2011

rosen2015

watson2009

michel2009

watson2001

webb2014

T1\_adassxxv

Previous master .bib file saved as /Users/ks/Proceedings/Work/adassXXVreferences.bib\_old

References for this paper added to /Users/ks/Proceedings/Work/adassXXVreferences.bib

The lists of keywords and references produced by Finish.py is quite a useful check on the names used by the references, and on the subject keywords you’ve used. If Finish.py has trouble parsing anything in the .tex file, it will say so and will ask you to fix it and try again. In this case it makes no changes to the file. (A common problem is an \ssindex entry without a closing ‘}’ – LaTeX won’t pick this up, as up until now the line has been commented out.)

Note that in the above example, Finish.py says it’s replacing the references and subject index entries in the master files. This is because I re-ran it to get some sample output for this document. However, it often needs re-running, because you may go back to the file and make further changes. For this reason the new entries in the master files are identified by comment lines that show which paper they came from, and Finish.py can replace the ones from the current paper rather than keep adding them to the file along with what may be earlier erroneous entries.

Of course, to re-typeset the file once Finish.py has commented out things like the \begin{document} line isn’t going to work unless the changes made by Finish.py are wound back. That’s done by yet another script, called Unfinish.py, which takes just the same arguments, and whose output is so obvious it isn’t worth showing here.

And finally, all you should need to upload for the final processing for any individual paper are its .tex file and any .eps files it uses.

# Using for future ADASS proceedings

I’d like to think that these could be used for subsequent ADASS Proceedings. The only ADASS XXV-specific thing is the name of the overall .bib file, adassXXVreferences.bib. This only appears in one place in the code, at the very start of AdassChecks.py, which contains the line:

\_\_AdassConference\_\_ = "XXV"

Changing this should be the only code change required. Changing this to “XXVI” will make the reference file adassXXVIreferences.bib. If you want a completely different style of name, you will have to change the one line in the code for GetBibFileName() in the same AdassChecks.py file, which at the moment is:

return ("adass" + \_\_AdassConference\_\_ + "references.bib")

Which doesn’t mean, of course, that another conference won’t want to do some things in a completely different way, and that may need more substantial changes. But for that, you’re on your own.