Virtual Observatories (and other astroinformatics stuff)

Norman Gray
Physics and Astronomy, Glasgow
(& Starlink & AstroGrid & EuroVO & ...)
STFC Astronomy Summer School
Glasgow, 2011 August 29

This is going to be a rather fragmentary hour – lots of things you need to know a bit about.

about me

- I still claim to be an astronomer
- (who just happens to spend 100% of his time with computers)
- I've worked with various astronomy computing projects
- (call it 'astroinformatics')

the plan

- Virtual Observatories
- Software that will help you do your work
- Writing software that will do your work

1. not a lot to say; 2. invaluable when available; 3. necessary but dangerous

virtual observatories

data volumes

- LIGO: 1 PB/year
- LHC: 10 PB/yr
- SKA: 100 GB/s (long distance) = 0.5 EB/yr = 0.05% of worldwide 1 ZB/yr total internet traffic

IEEE 802.3

BS 1363

How many of each are in the room? You don't know? Good.
From our point of view, the detailed design of each of these plugs is less important than that they are a standard, so completely interchangeable

VOs and Grids

Virtual Observatories (VOs) and Grids are different things, but share the same intuition: computing is a distraction, and most astronomers shouldn't (have to) care about the details.

computing infrastructure

- You're not supposed to notice plugs – they're supposed to simply be there, and produce precisely what they ought to.
- VOs and (computing) Grids have the same goal, concerning data and computing power, respectively.
- Do they succeed? Opinions differ.

grids

- Remote computing resources, storage and networking.
- Users/projects have allocations of time (etc) on the resources, after authentication.
- Move the code to the data, not the data to the code.
- Globus is the big name; X.509 is the big pain.
- (yes, this is broadly the same idea as The Cloud)

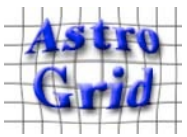
If you're in a project which uses grids, you'll know about it soon enough (so no point talking about it now)

virtual observatory

- Lots of people have data (eg WFAU in Edinburgh, ESO in Europe, CDS in Strasbourg, individual telescopes...), and they all potentially provide it in different forms, and through different interfaces.
- Nightmare.
- If you know what you want, and whom to ask for it, you can get your data, but...

virtual observatory

- ...you want to be able to ask: "I want all of the infrared observations made by *anybody* of *this* patch of the sky between *these* dates"
- ..."and I want my data analysis software to be able to read all of it."
- "And I don't want to think about it."
- Not quite there yet.



Getting there, but people still have to care a bit too much about different standards, and have to be conscious of sockets. Good Thing; just slower than expected

practicalities

So much for background. On to practicalities...

- FITS is still a standard
- VOTable is a standard (cf `www.ivoa.net`)
- UCDs are an imperfect standard
- ADQL and TAP
- TOPCAT and Aladin

fits

- Flexible Image Transport System
- `http://fits.gsfc.nasa.gov`
- Simple data transport file format. Bag-o-bits, or tables, plus key-value metadata
- 30 years old and ~~not dead yet~~ still going strong
- Use cfitsio, or a reliable library; *don't write your own*
- Not a VO format as such

Generally, the applications you use will be able to read FITS, but you MIGHT have to read a FITS file into a program

fits headers

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                  -32 / number of bits per data pixel
NAXIS   =                    3 / number of data axes
NAXIS1  =                   10 / length of data axis 1
NAXIS2  =                   10 / length of data axis 2
NAXIS3  =                    4 / length of data axis 3
EXTEND  =                    T / FITS dataset may contain extensions
BSCALE  =                   1. / REAL = TAPE*BSCALE + BZERO
BZERO   =                   0.
ORIGIN  = 'STScI-STSDAS'       / Fitsio version 21-Feb-1996
FITSDATE= '2004-01-09'         / Date FITS file was created
CRVAL1  =       182.6311886308
CRVAL2  =       39.39633673411
CRPIX1  = '420.    '
CRPIX2  =                424.5
CD1_1   =         -1.06704E-06
CD1_2   =         -1.25958E-05
CD2_1   =         -1.26016E-05
CD2_2   =          1.06655E-06
DATAMIN =           -73.19537 / DATA MIN
DATAMAX =            3777.701 / DATA MAX
MIR_REVR=                    T
ORIENTAT=              -85.16
ERRCNT  =                    0
CTYPE1  = 'RA---TAN'
CTYPE2  = 'DEC--TAN'
```

votable

- XML format, heavily influenced by FITS
- Increasingly supported by VO-aware services and software
- `http://www.ivoa.net/Documents/VOTable/`

votable sample

```
<?xml version="1.0"?>
<VOTABLE version="1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://www.ivoa.net/xml/VOTable/v1.2"
 xmlns:stc="http://www.ivoa.net/xml/STC/v1.30" >
  <RESOURCE name="myFavouriteGalaxies">
    <TABLE name="results">
      <DESCRIPTION>Velocities and Distance estimations</DESCRIPTION>
      <GROUP ID="J2000" utype="stc:AstroCoords">
        <PARAM datatype="char" arraysize="*" ucd="pos.frame" name="cooframe"
             utype="stc:AstroCoords.coord_system_id" value="UTC-ICRS-TOPO" />
        <FIELDref ref="col1"/>
        <FIELDref ref="col2"/>
      </GROUP>
      <PARAM name="Telescope" datatype="float" ucd="phys.size;instr.tel"
             unit="m" value="3.6"/>
      <FIELD name="RA"   ID="col1" ucd="pos.eq.ra;meta.main" ref="J2000"
             utype="stc:AstroCoords.Position2D.Value2.C1"
             datatype="float" width="6" precision="2" unit="deg"/>
      <FIELD name="Dec"  ID="col2" ucd="pos.eq.dec;meta.main" ref="J2000"
             utype="stc:AstroCoords.Position2D.Value2.C2"
             datatype="float" width="6" precision="2" unit="deg"/>
```


Note UCDs embedded here – hints to software

adql and tap

- Astronomical Data Query Language (astronomical extensions to SQL)
- Table Access Protocol (standardised protocol for retrieving data)
- Both quietly rolling out; use this interface if it's available

vo software

- Mostly server software (ie inter-archive)
- But TOPCAT (`http://www.star.bristol.ac.uk/~mbt/`) & STILTS
- ...and Aladin (`http://aladin.u-strasbg.fr/`)
- are user tools, for table manipulation and catalogue viewing respectively.

existing software

Use existing software

You are not here to be software developers

Existing software...

- ...is reliable
- ...is believed
- ...has helpful people around

- Funded from 1980–2005 by SRC/SERC/PPARC, and ~~not dead yet~~ still going as part of JACH operations
- Data reduction and analysis software
- `http://www.starlink.ac.uk`
- Multiple tools, including
- GAIA, ORAC-DR, KAPPA, CCDPACK, SLALIB, ...
- Support is now 'open-sourceish'

iraf

- Image Reduction and Analysis Facility
- Funded by NOAO, and ~~not dead yet~~ still going, on a shoestring, principally for NOAO projects
- `http://iraf.noao.edu/`
- Data reduction and analysis; strong support for particular instruments

scipy

- Scientific Python, unequivocally going strong (not even slightly dead)
- `http://www.scipy.org/`
- An open-source project, with all the good and bad things that implies. Focused on enabling your scripts rather than providing a full working environment.

idl

- More-or-less compulsory if you're in Solar Physics (because 'solarsoft' is written in IDL)
- Interactive
- Otherwise, not nearly dead enough
- Proprietary and expensive
- GDL exists, but is still beta

Possibly some complicated zombie joke here. GDL addresses the proprietary issue (imperfectly), but it's still an awkward language

sunpy

- 'Python for solar physics'
- `http://www.sunpy.org/`
- It's 'not ready yet to be used in day-to-day software development'

matlab

- Proprietary and expensive, but a well thought-out and reasonably modern language
- Interactive
- Fairly easily extensible (including for example FITS readers)
- Matlab is really a scripting environment

...scripting

which brings us to...

developing software

you are not software developers

- It is not your job to be software developers
- But you'll probably spend at least some of your time writing code
- There are good ways and bad ways to do that

Depends on the type of PhD, the work you're doing, and the group you're in

Not good lab practice

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main(){F_OO();printf("%1.3f\n",
4.*-F/OO/OO);}F_OO()
{
          _-_-_-_
       _-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_
       _-_-_-_-_-_-_-_
          _-_-_-_-_
}
```

There are also bad programming practices

things you should know about

- `http://software-carpentry.org/` ← a syllabus
- Languages (including scripting)
- ‘make’
- Testing
- Version control

key message

Efficiency matters!

Your efficiency matters

not program efficiency

- 1st rule of optimisation: don't do it
- 2nd rule of optimisation (experts only): don't do it yet

Michael A Jackson

Don't optimise until you KNOW you have a problem. A correct slow program is better than a fast incorrect one (or unknown)
It's easier to optimise a working program, than fix an optimised one

languages

It's OK to know more than one language!

- Python
- Matlab
- Java
- (Shell)
- C
- Fortran

python

- Very flexible
- Popular
- Reasonably easy to pick up
- Readable/maintainable
- Large set of libraries
- Extensible

'popular' is a Good Thing – other people to ask questions of

matlab

- Good for numerical/array work
- Easy to use interactively
- Decent set of libraries
- ...but expensive

java

- Very well known
- Very maintainable
- Huge set of libraries
- ...but not much fun (this is a good thing)

shell scripting

- Glue
- 'shell scripting' really means 'using all the little unix tools'
- Don't get carried away
- Google for 'bash' or 'bourne shell' (csh is a bit clumsy)
- Automation breeds consistency – executable notes

c

- Good for bit-twiddling and talking to hardware
- Well-known
- Fun in small doses (this is a *bad* thing)
- Hard to write good large-scale code

There are very few questions to which the correct answer is 'C'.
It's not _impossible_ to write good C, but it needs discipline and experience, and is HARD

fortran

- Good for very large-scale numerical problems
- Wide range of well-respected numerical libraries

mixed-language programming

- For example, calling Fortran functions from Python
- Less scary than it sounds
- Benefits of existing libraries with the program organisation of a decent language

efficiency

Your time is more important than computer time

efficiency

Exhortations about efficiency notwithstanding, if you're doing numerical programming, there are some libraries you should know about and use.

- Numerical Recipies
- NAG
- Netlib

make

```
prog: mod1.o mod2.o
    gcc -o prog -Lextralib/ mod1.o mod2.o

mod1.o: mod1.c
    gcc -o mod1.o -Iinc -DBUF=20 mod1.c

mod2.o: mod2.f
    f77 -o mod2.o --funky-build-option mod2.f
```

Then just type '`make prog`'

See `http://www.gnu.org/software/make/manual/make.html` (or just google 'gnu make manual')

This is another type of 'executable note'. The Makefile knows how to build your program, so you don't have to remember

testing

- See the ‘software carpentry’ slides
- Initially feels like an annoying waste of time
- But if a change you made today breaks a function you wrote last month, you want to know that now
- Collect tests, and regularly run them all together
- Comprehensive tests mean you can change things with confidence
- Make mistakes at most once!

version control

- Not just for sharing code
- Roll back time
- ‘This was working last week!’
- ‘What version did I use for that paper?’
- Mercurial: `http://mercurial.selenic.com/guide/`
- Git: `http://git-scm.com/`
- Subversion

Automated testing is a Good Thing
...though no-one ever believes you

Be lazy – use libraries

Reduce entropy – automate

Make mistakes only once – test

`http://software-carpentry.org`

Norman Gray : `http://nxg.me.uk`