Practical Pipelining using Python

The University of Sydney

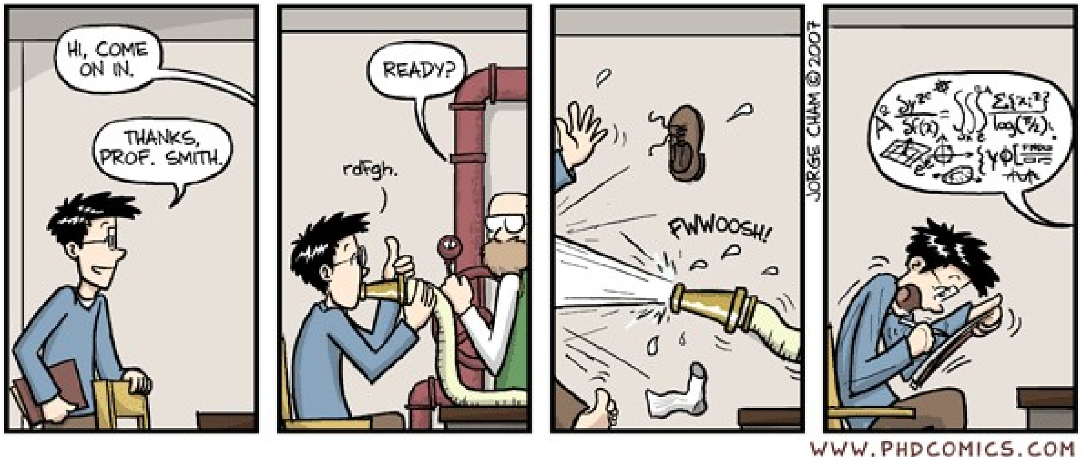
The Sydney Institute for Astrophysics

Cormac Purcell, CASS Radio School, 2nd October 2014

THE UNIVERSITY OF SYDNEY

Data flood

- More projects than ever conducted **survey mode** – even PhD projects
  - Wide area or dispersed: **many pointings**
  - **Multi-epoch**: years of ongoing observations and many observers
  - Multi-wavelength: **linked with sister surveys** on other telescopes



- Huge amount of **meta-data** necessary to manage successful science:
  - Coordinates, sensitivity, outages, coverage, imaging quality, source-finding, cross-matching, quality control, publishing data-products
- Culture of **transparency**, consistency and **reproducibility**
  - Transparent processing from final data-product to raw observations
  - Consistent processing across dataset using well tested software
  - Version control so that results are reproducible after later releases
- **Worth building a software pipeline!**

THE UNIVERSITY OF SYDNEY

Typical pipeline flow

Reduce raw data → Assess quality → Analyse & Interpret → Disseminate results & data

- Data reduction (ingestion, calibration, imaging)
- Quality control (re-observe bad data, weighting, flagging)
- Basic Analysis (source-finding, clump/spectral-fitting, catalogues)
- Advanced analysis (multi-wavelength cross-matching, time-series, object identification)
- Data access and citizen science (web-publication, virtual observatory)
- **Must be a meta-data trail from the start – requires a database**

THE UNIVERSITY OF SYDNEY

Python – a sticky language

Reduce raw data → Assess quality → Analyse & Interpret → **Disseminate results & data**

- Image and table data accessible using **astropy** module (FITS & ASCII)
- Manage **metadata** via simple built-in database or external relational database
- Easily manipulate data in memory to create diagnostic metrics (numpy)
- Flexible plotting ability via **matplotlib** and **APLpy**
- Well-tested suite of **analysis tools**, e.g., source-finding (Aegean, Blobcat), model-fitting (MPFIT, scipy.optomize, EMCEE).
- Python code is **easy to read** and should be published alongside papers
- Web-servers (e.g. Apache2, bottle) understand python. Build dynamic web-interfaces to your scripts and share with your co-authors and community:
    - Catalogue & image servers, 'Zooniverse' style analysis (user-based)
    - Automatically generate plots, tables and reports for quality assessment

Python – astropy and friends

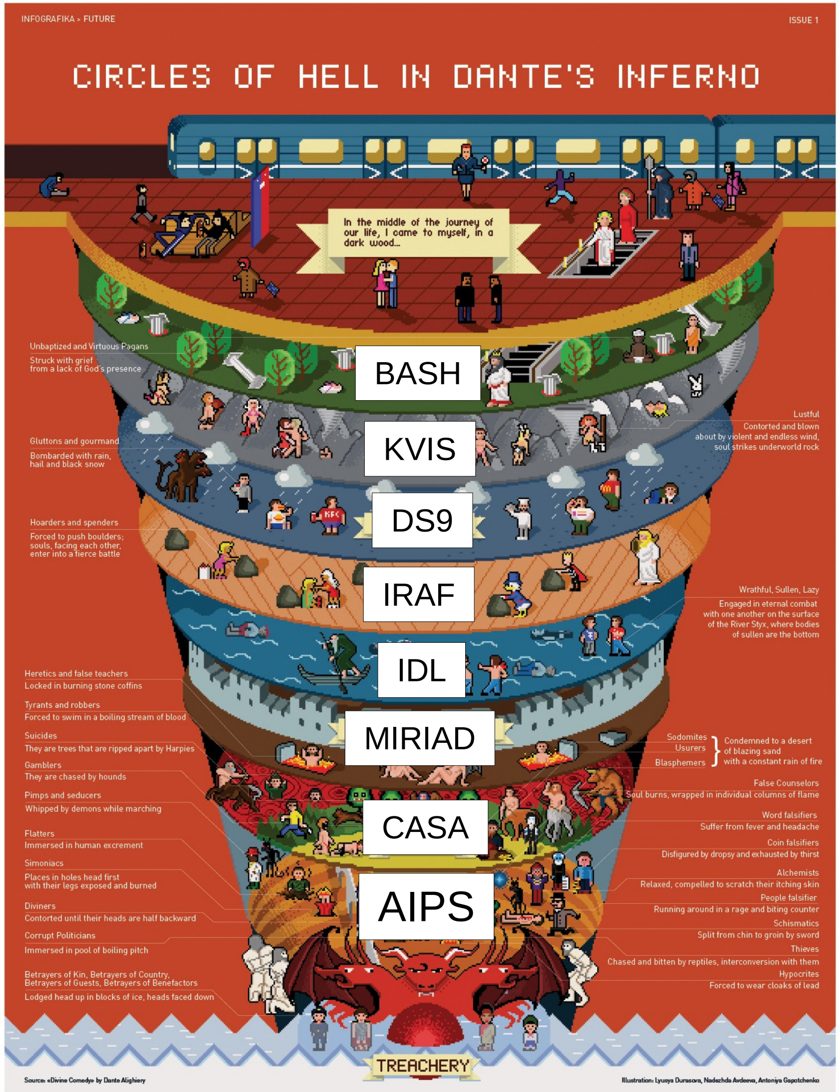
- Until ~2012 python astronomy modules were scattered around the web
- Core modules now unified under **astropy** banner & **affiliated packages**
  - Astropy.io.fits (formerly PyFITS)
  - Astropy.io.ascii
  - Astropy.wcs (formerly PyWCS)
  - Astropy.coordinates
  - Astropy.convolution
- Useful affiliated packages:
  - APLpy - Image plotting
  - PyVO - query the virtual observatory
  - Astroquery – access web services
- Useful non-astronomy packages
  - MySQL DB
  - SQLite3
  - Matplotlib
  - SciPy
  - MPFIT
  - EMCEE

Python – beware the unknown


- **Astropy** is still **under development**
- **Core** packages **mature** & well tested
- **Newer packages** incomplete & **in flux**
- Newer packages often expose less well-known python functionality and add useful features
- Sometimes necessary to peer behind the curtain and **get your hands dirty**
  - Astropy.io.ascii ~ **numpy.loadtxt**
  - Astropy.table ~ **numpy.recarray**
- APLpy easily augmented
  - Matplotlib under the hood
  - Combine with other figures
- No substitute for ***really understanding*** *what is being done to your data*!

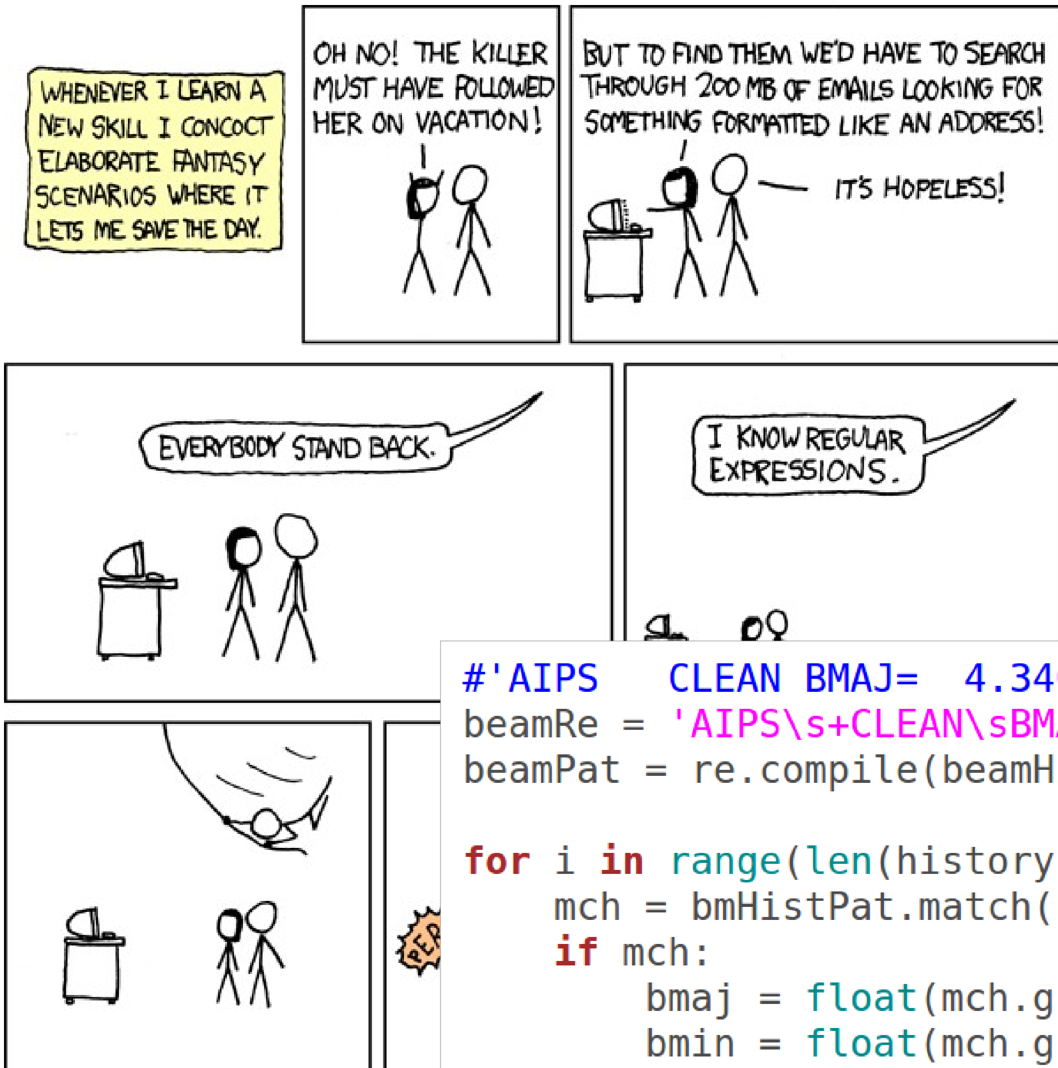
Python – accessing external software

- **Good:** Often easiest to make **system calls**
  - Tell python where the package lives and set up the $ENVIRONMENT
  - Pipe into STDOUT to view output of task in real time
- **Bad(ish):** Some external packages have dedicated python wrappers
  - MIRIAD → **MIRpy**
  - AIPS → **parseltongue** and **obittalk**
  - CASA → **casapy** *
  - Increases *complexity* sometimes (bad)
- **Super-Ugly:** Can interface with *ancient* user-driven software using **pyexpect**
  - Pretends to be the user when dealing with Q&A command line interfaces



* casapy comes with its own version of Python, which means you may need to manage this environment in parallel, i.e., duplicate installed modules

Python – accessing external software



- Regular expressions (**RegEx**)
- Match patterns in any ASCII string
- Incredibly useful for **parsing** the output of programs and tasks, e.g., **MIRIAD uvindex**

\S+ = 1 or more non-space chars

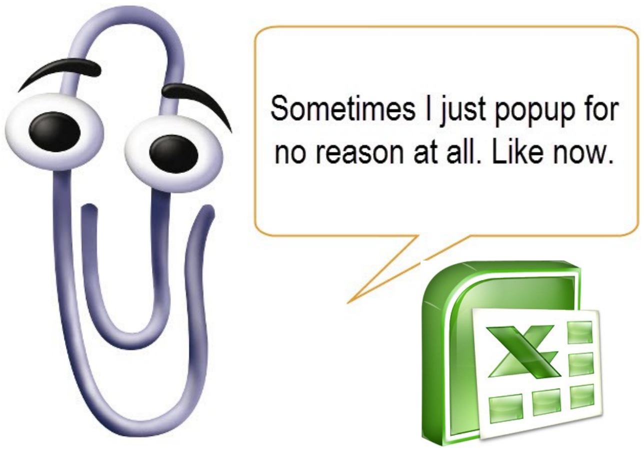
\s* = 0 or more spaces

() = groups of text to extract

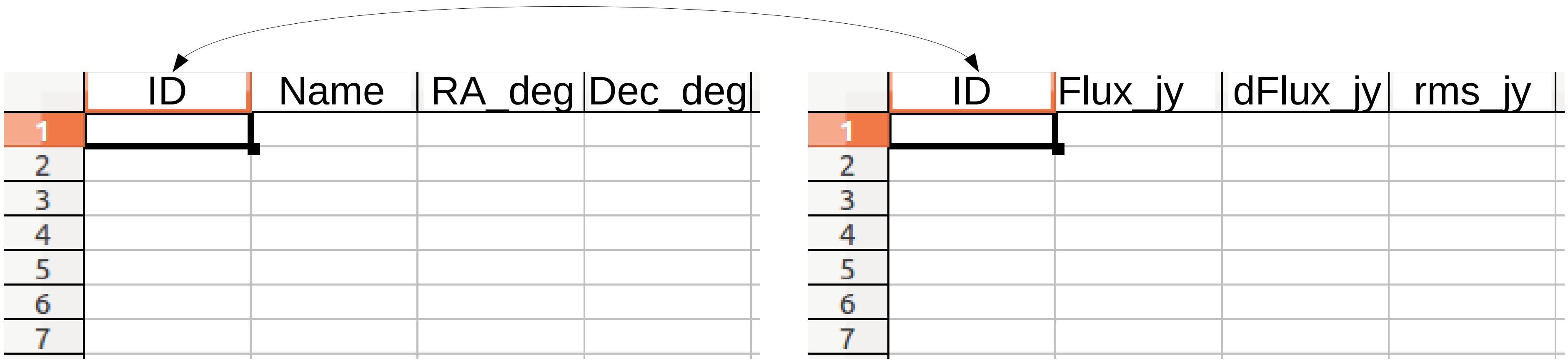
```
#'AIPS   CLEAN BMAJ=  4.3403E-04 BMIN=  3.1039E-04 BPA= -11.55'
beamRe = 'AIPS\s+CLEAN\sBMAJ=\s+(\S+)\s+BMIN=\s+(\S+)\s+BPA=\s+(\S+)'
beamPat = re.compile(beamHistStr)

for i in range(len(history)):
    mch = bmHistPat.match(history[i])
    if mch:
        bmaj = float(mch.group(1))
        bmin = float(mch.group(2))
        bpa  = float(mch.group(3))
        break
```

THE UNIVERSITY OF SYDNEY

Databases – Why?



- Replaces **Excel** (or Topcat, ASCII) in your workflow
- Advantages: **scriptable**, **multi-user** access, *fast*
- Large projects: run on a **dedicated server** (MySQL)
- Small projects: database **built-into python** (SQLite3)
- Server-client model with choice of interface client:
    - **Graphical** clients which mimics **spreadsheets**
    - **Queries** through a script or command-line return a **filtered** or **joined table**
- Under the bonnet:
    - Tables connected by **key** columns with **unique entries**



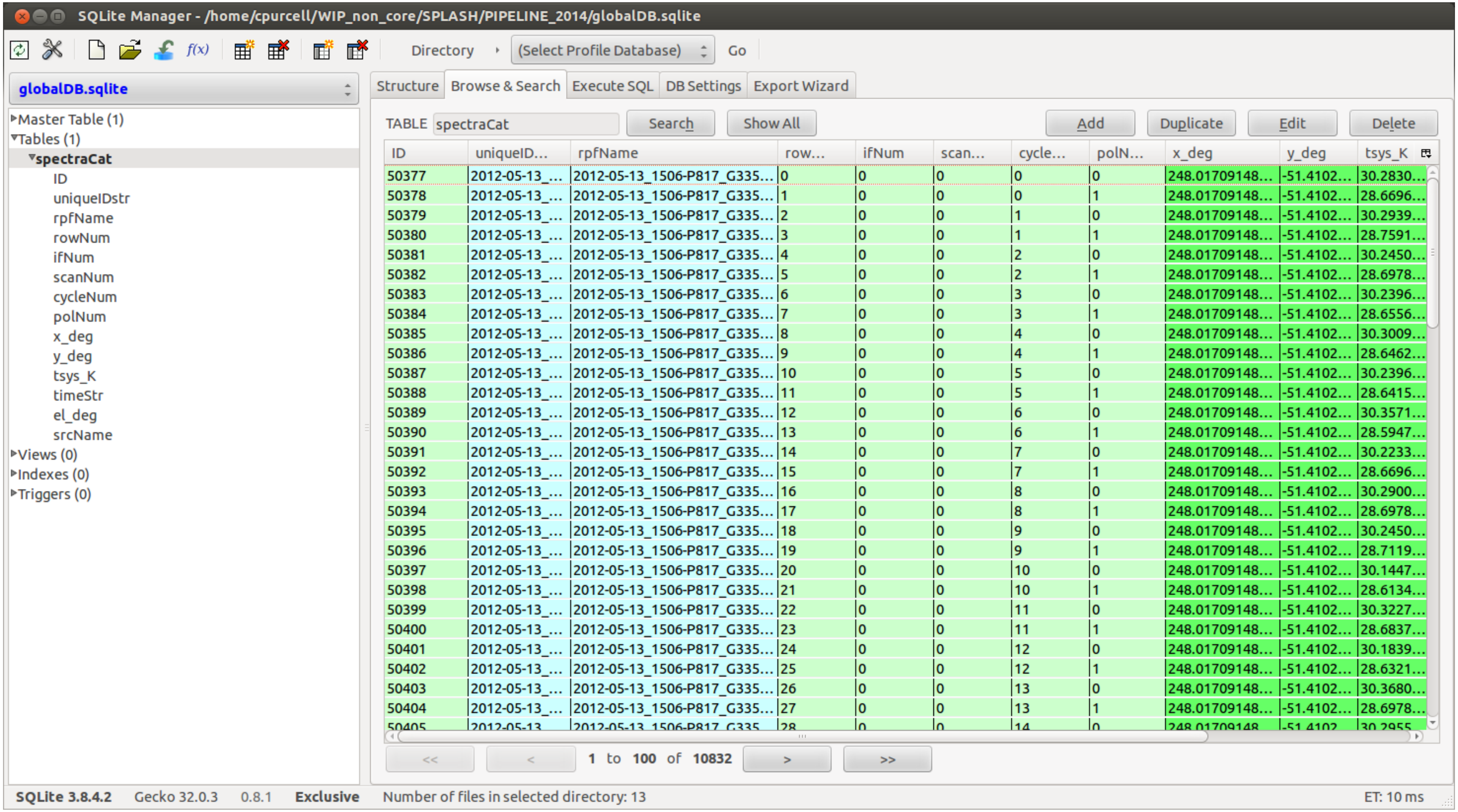
SQLite3 – Python's built-in database

- Distributed with python since V2.5
- **Serverless**: creates a database file
- File can be moved, copied and read on any other system (OS-agnostic)
- Can also operate in memory
- **No** configuration required
- Open source
- Crash safe (prevents data corruption)

http://www.sqlite.org

- Already used by many well known programs to store data on your computer:

SQLite3 – access through Firefox



- SQLite Manager Firefox plug-in: edit, query & export table data

  https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/

THE UNIVERSITY OF SYDNEY

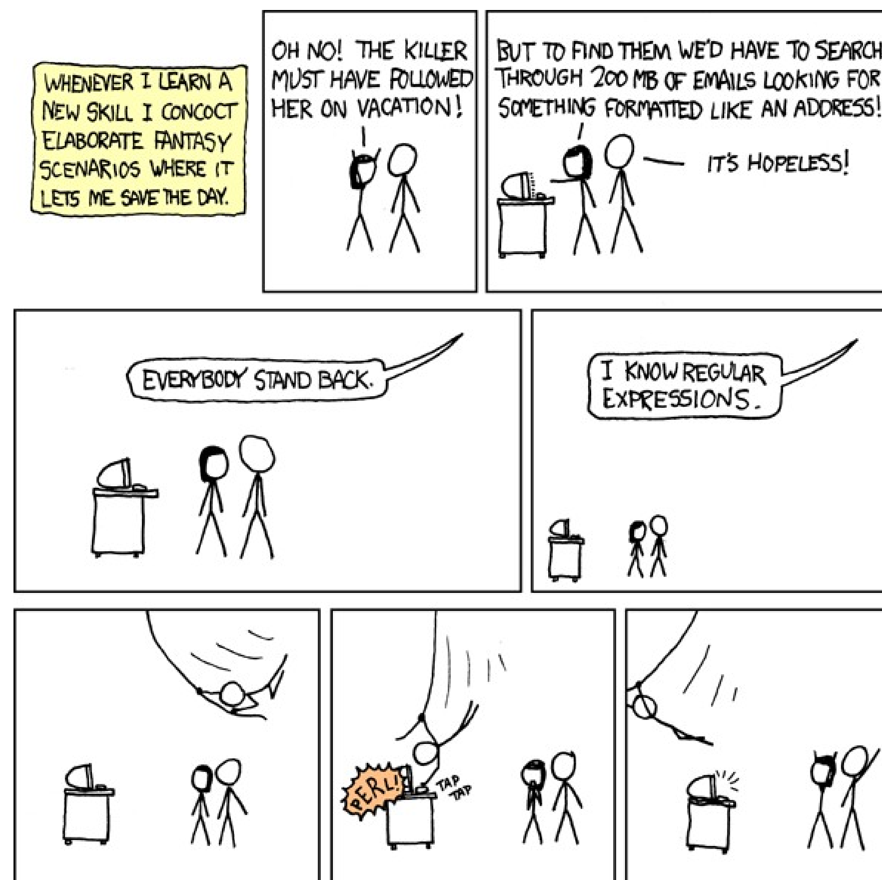
SQLite3 – access through Python

- Python SQLite3 database access takes place via **cursor object**
- Queries are designed in a special statements which are used to **delete, insert** or **select** information from the database tables.
- Interface is almost identical when accessing heavy-duty databases (MySQL, PostgreSQL).

```
# Connect to the database file
dbFile = 'myDatabase.sqlite'
conn = sqlite3.connect(dbFile)
cursor = conn.cursor()

# Fetch all of 'myTable'
statement = "SELECT * FROM myTable;"
cursor.execute(sql)
rows = cursor.fetchall()

# Disconnect
cursor.close()
conn.close()
```

Databases – ~~Sequel~~ SQL

- 'Structured Query Language' (SQL)
  - Close to English in syntax
  - Basic trigonometric and mathematical operations
  - Powerful regular expressions



SQL – creating a table

Raw SQL

```sql
CREATE TABLE spectraCat (
ID INTEGER PRIMARY KEY,
uniqueIDstr VARCHAR(100),
rpfName VARCHAR(50),
rowNum INTEGER,
ifNum INTEGER,
scanNum INTEGER,
cycleNum INTEGER,
polNum INTEGER,
x_deg DOUBLE,
y_deg DOUBLE,
tsys_K DOUBLE,
timeStr VARCHAR(50),
el_deg DOUBLE,
srcName VARCHAR(50));
```

Python

```python
# Create a table
sql = """
CREATE TABLE spectraCat (
ID INTEGER PRIMARY KEY,
uniqueIDstr VARCHAR(100),
rpfName VARCHAR(50),
rowNum INTEGER,
ifNum INTEGER,
scanNum INTEGER,
cycleNum INTEGER,
polNum INTEGER,
x_deg DOUBLE,
y_deg DOUBLE,
tsys_K DOUBLE,
timeStr VARCHAR(50),
el_deg DOUBLE,
srcName VARCHAR(50));
"""
cursor.execute(sql)
```

SQL – populating a database

Raw SQL

```sql
INSERT OR REPLACE
INTO tableName
(column1,
 column2,...
 columnN)
VALUES (value1,
        value2,...
        valueN);
```

Python

```python
# Set some values for columns
colName1 = 'column1'
val1 = 2.0
colName2 = 'column2'
val2 = 3.0

# Form the SQL statement in parts
sql = "INSERT OR REPLACE INTO %s" % tabName
sql += ' (%s, %s)' % (colName1, colName2)
sql += ' VALUES (?, ?)'

# Note: we put '?' in place of values
# and then use the 2nd argument of the
# execute statement to specify the values
cursor.execute(sql, (val1, val2))

# Call to write the changes to disk
conn.commit()
```

SQL – querying a database

Raw SQL

```
SELECT column1, column5
FROM tableName
WHERE
column1 = <condition1>
AND column2 = <condition2>
AND (column3 > value1 OR
     column3 < value2)
ORDER BY column5;
```

Python

```
# Coordinate limits
xLimUp = 0.5;  xLimDn = -0.5

# SQL
sql = '''SELECT
uniqueName,
x_deg,
y_deg,
flux_Jybm
FROM spectraTab
WHERE coordTab.x_deg<?
AND coordTab.x_deg>?
ORDER BY x_deg
'''

# Execute using limits
cursor.execute(sql, (xLimUp, xLimDn))
rows = cursor.fetchall()
```

SQL – joining two tables

Raw SQL

```
SELECT
table1.column1,
table1.column2,
table2.column5
FROM table1 INNER JOIN table2
ON table1.column1 = table2.column1
WHERE
table1.column1 = <condition1>
AND table1.column2 = <condition2>
AND (table2.column3 > value1 OR
     table2.column3 < value2)
ORDER BY table2.column5;
```

Python

```
# Coordinate limits
xLimUp = 0.5;  xLimDn = -0.5

# SQL
sql = '''SELECT
coordTab.uniqueName,
coordTab.x_deg,
coordTab.y_deg,
spectraTab.flux_Jybm
FROM coordTab LEFT JOIN spectraTab
ON coordTab.uniqueName =
spectraTab.uniqueName
WHERE coordTab.x_deg<?
AND coordTab.x_deg>?
ORDER BY x_deg
'''

# Execute using limits
cursor.execute(sql, (xLimUp, xLimDn))
rows = cursor.fetchall()
```

SQL – joining two tables

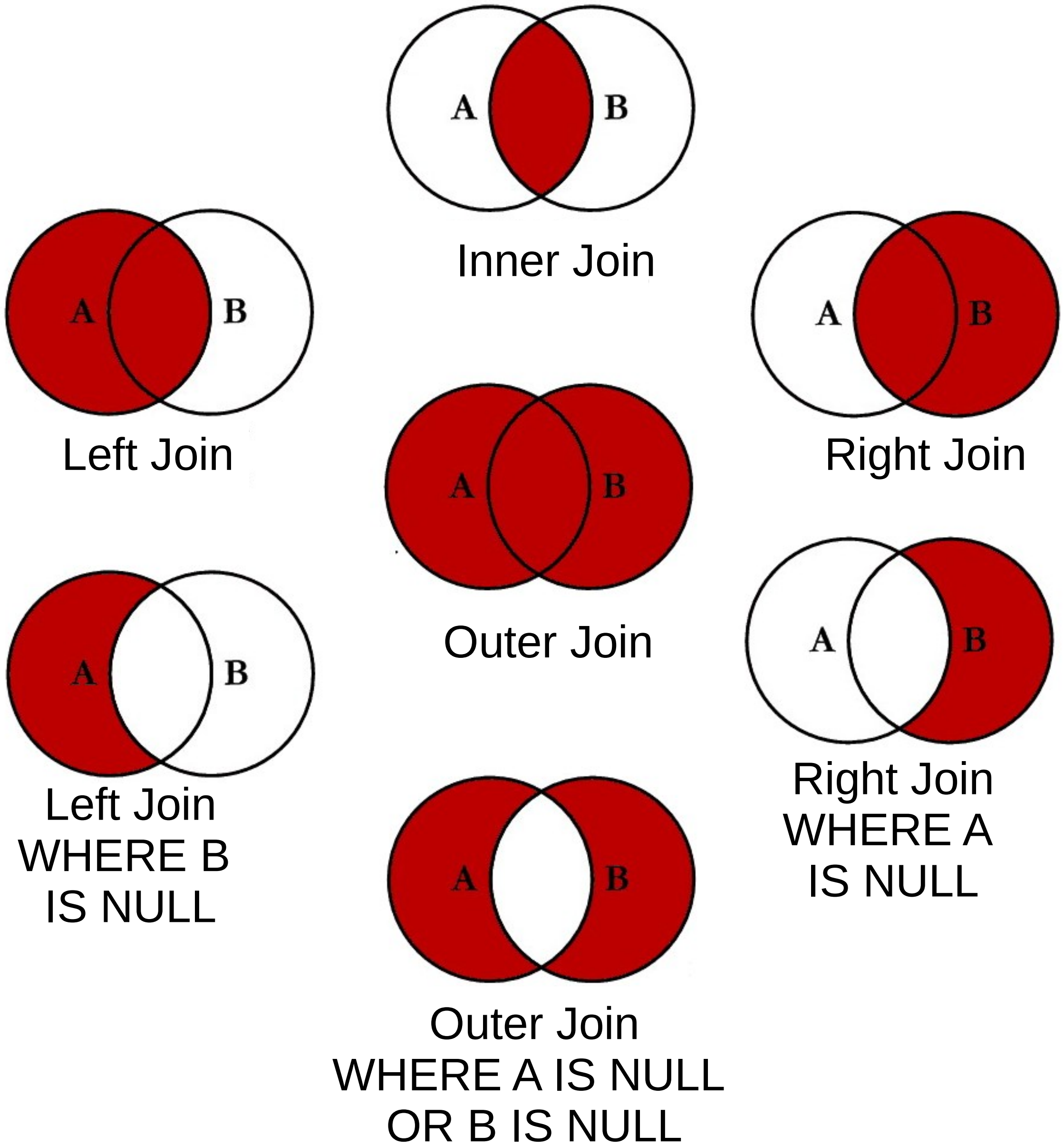


Image credit: stackoverflow

Python – Numpy record arrays

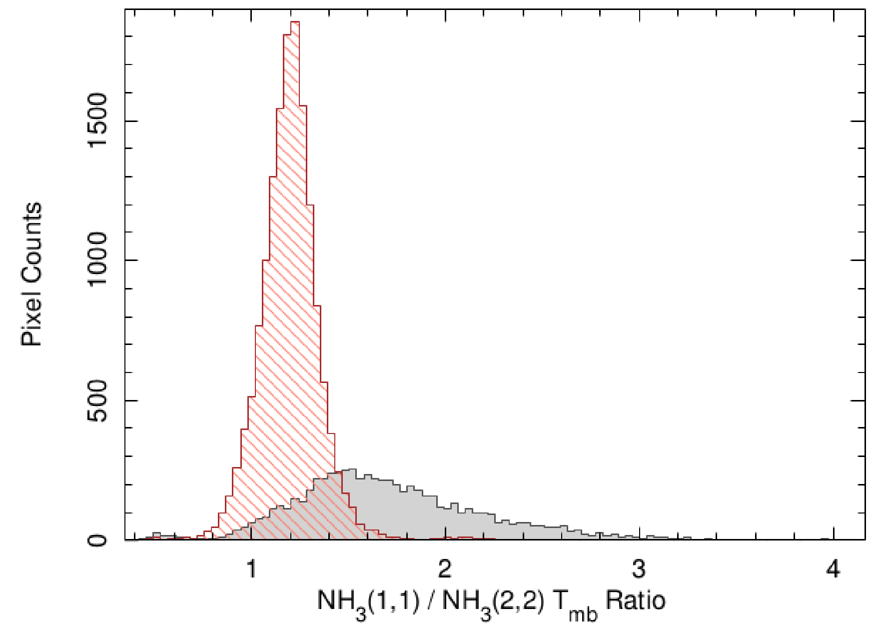
- **Numpy** is the array-processing and vector algebra module for python
- Arrays commonly defined as a **single data-type** (e.g., 32-bit float)
- A more rich definition exists which allows mixing of types - **recarrays**
- Columns can be named and assigned different variable types (dtypes)
- Natural match for tabular data stored in a database
- Some database-like functionality using logical statements and filters
- Augmented version used in the **astropy.table** module

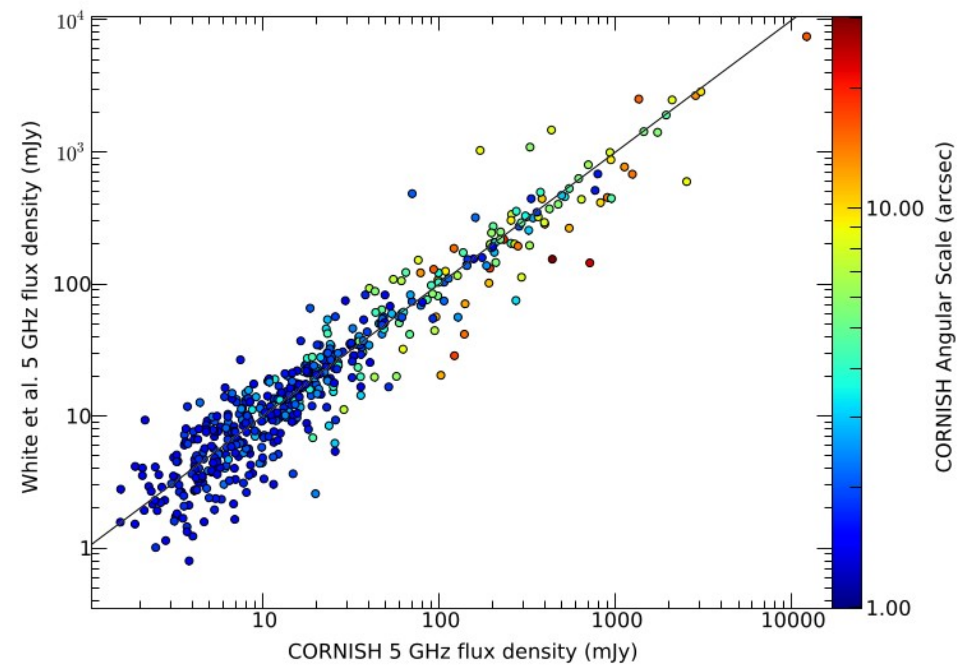
```
# Initialising a numpy array
myArr = np.zeros(shape=(3), dtype='f4')

# Initialising a record array:
dtLst = [('name','a10'),
         ('maxFlux', 'f4'),
         ('RMS', 'f4')]
myRec = np.zeros(shape=(3), dtype=dtLst)

# Accessing a record array:
myRec['name']     # by column
myRec[0]          # by row
```

http://docs.scipy.org/doc/numpy/user/basics.rec.html

Plotting from a database

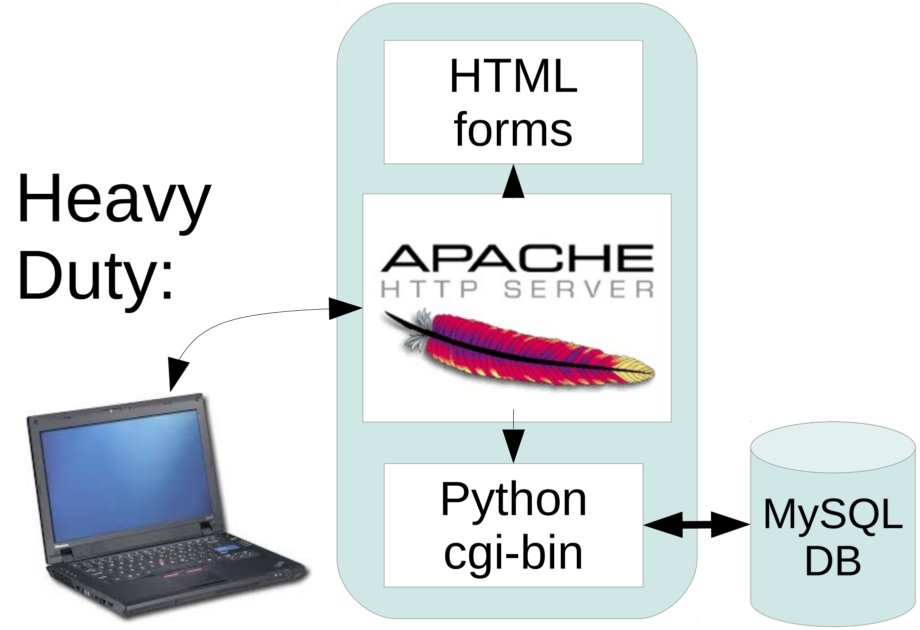
- **Query driven plotting** is a very **powerful** tool for exploring data
- The **query** defines the input **data**
- Two most useful plots in science:
  - Histogram
  - Scatter plot
- Build simple python script to read one or more queries for comparison
- Control flags allow manipulation of plot parameters

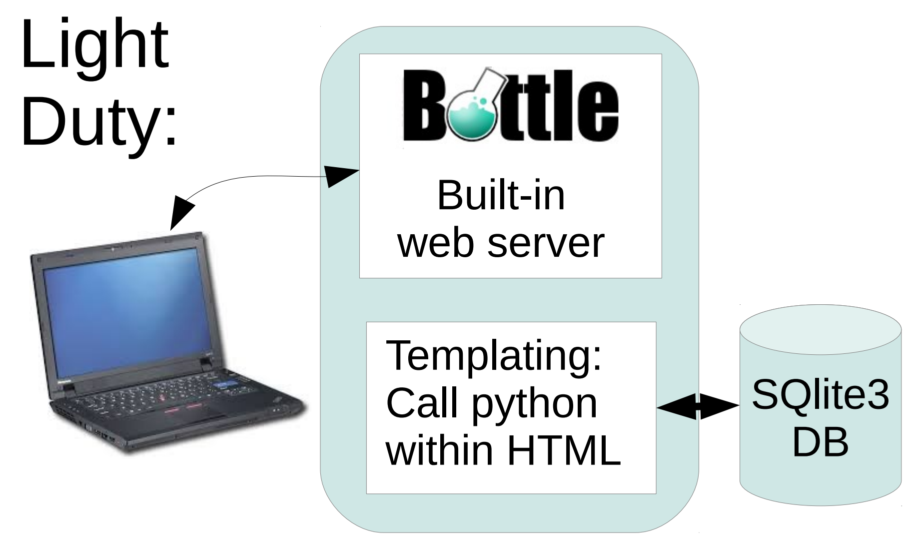




Sharing your science – web access

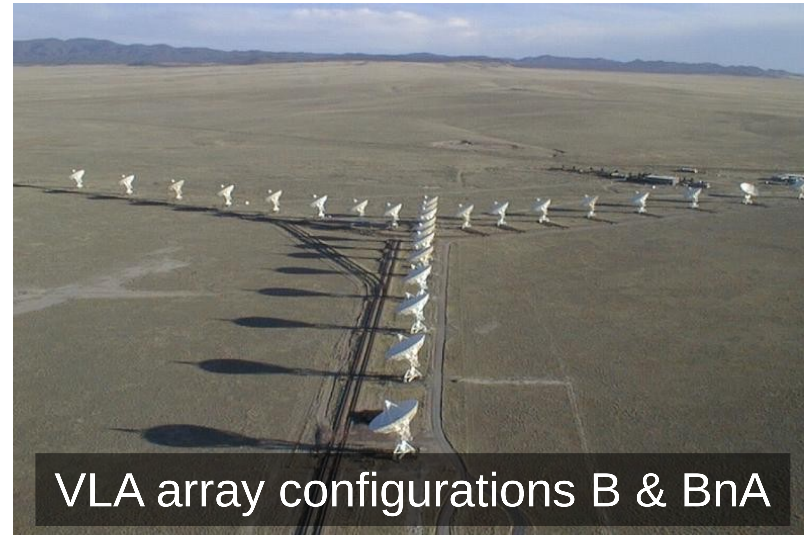
- Interface to your project at **all stages**
  - Managing observations (e.g., MALT90)
  - Raw data quality control (daily reports)
  - Data-reduction control (selection)
  - User driven tasks: **eyeball carnage**
- Two main choices for web
  - Heavy duty server e.g., **Apache2**
  - Light web 'framework' e.g., **Bottle**
- **Apache**: Comon Gateway Interface
  - Python scripts in the cgi-bin directory
  - Matching forms in separate HTML
- **Bottle**: Templating language
  - HTML template defines forms and calls python in one place
  - **Stand-alone** or use within Apache

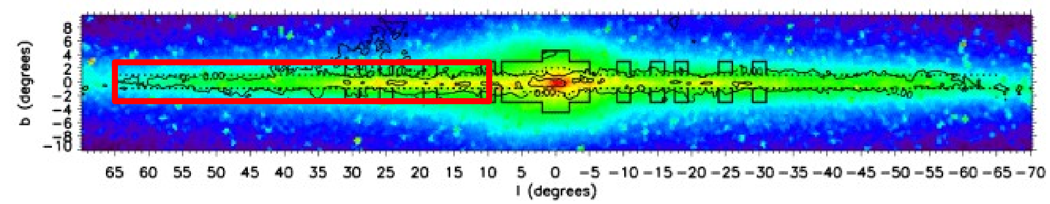


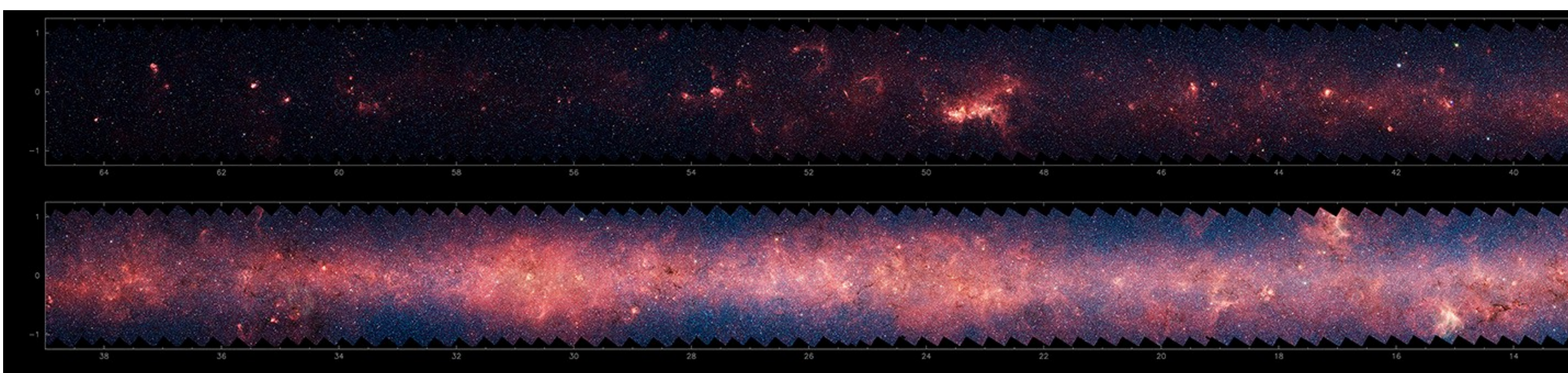


Pipeline case study – CORNISH

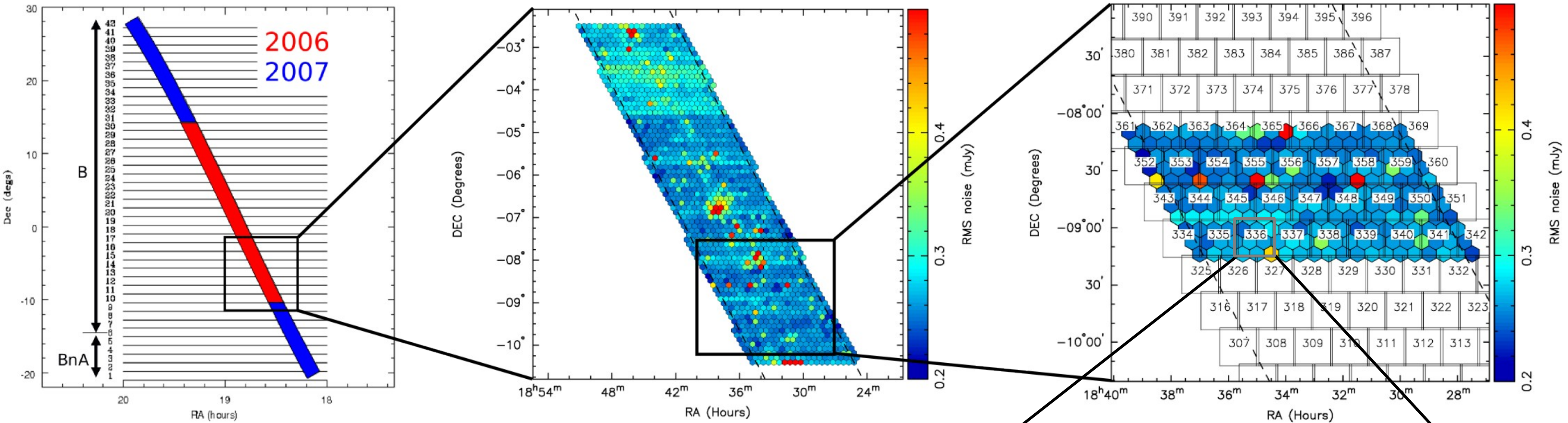
- Northern GLIMPSE I region
- $10^{0} < l < 65^{0}$ $|b| < 1^{0}$
- 100 square degrees
- 8.5' primary beam, 1.5'' resolution
- < 0.4 mJy/bm rms noise level
- **9351 fields**, hexagonal pattern
  - 2x 40s cuts → **18702** data



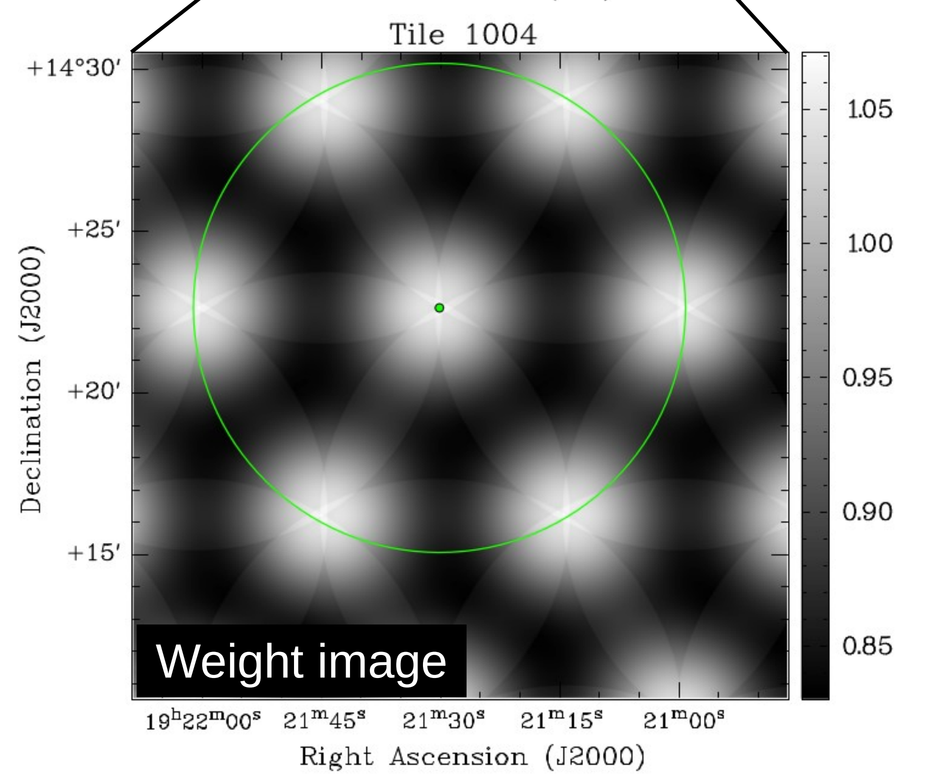




Pipeline case study – CORNISH

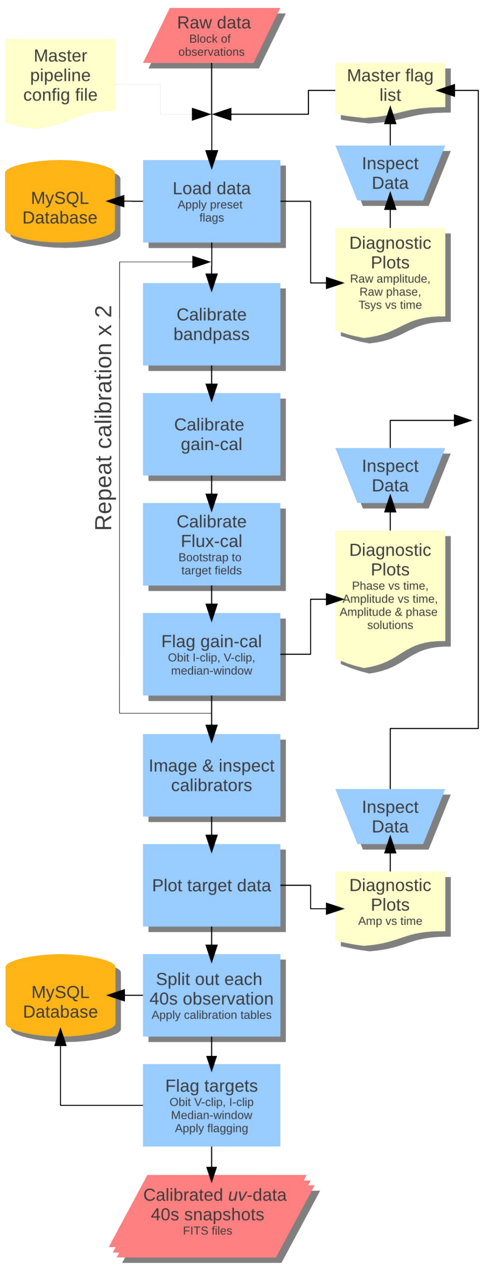


- Primary data product is a mosaiced image 'tile'
- Observations conducted over 3 years
- Important to quickly reduce data and track data-quality so bad fields could be re-observed

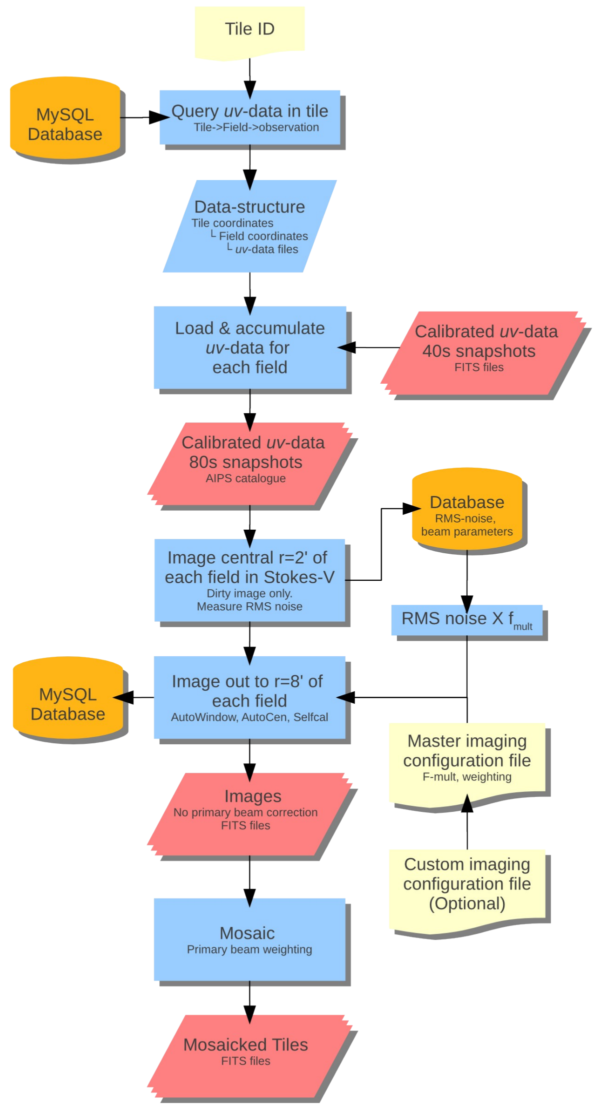


Pipeline case study – CORNISH

Calibration:



Imaging:

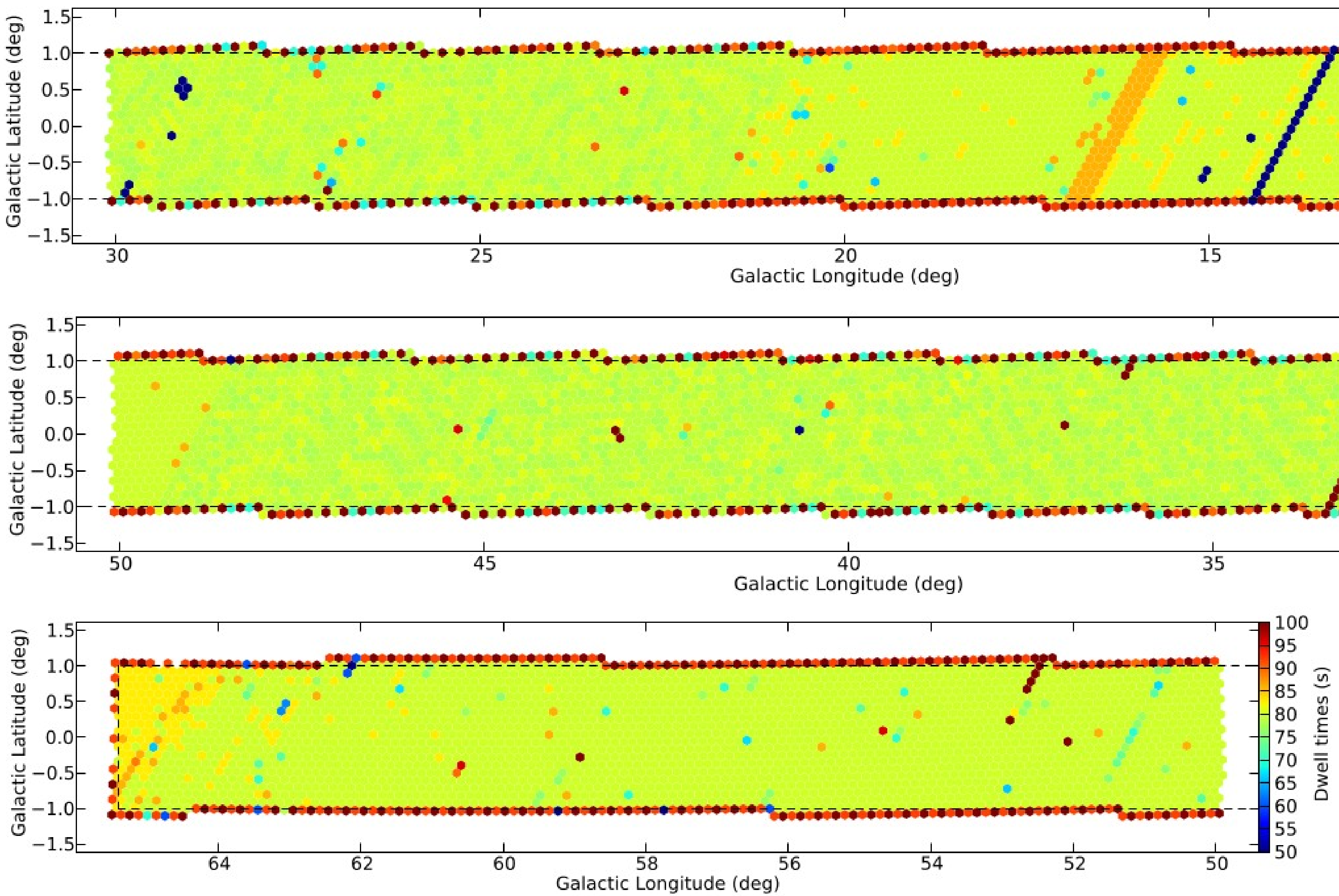


- Fairly standard data-reduction
- However, observations took place during a **rolling antenna upgrade,** control **software upgrade** and peak **electrical storm season**
- Python based pipeline was driven by ASCII configuration files and stored all metadata in a MySQL database
- Raw and intermediate data were stored on disk as *uv*- or image-format FITS files
- Database was instrumental for managing data producing **quality control diagnostic plots**

THE UNIVERSITY OF SYDNEY

Pipeline case study – CORNISH

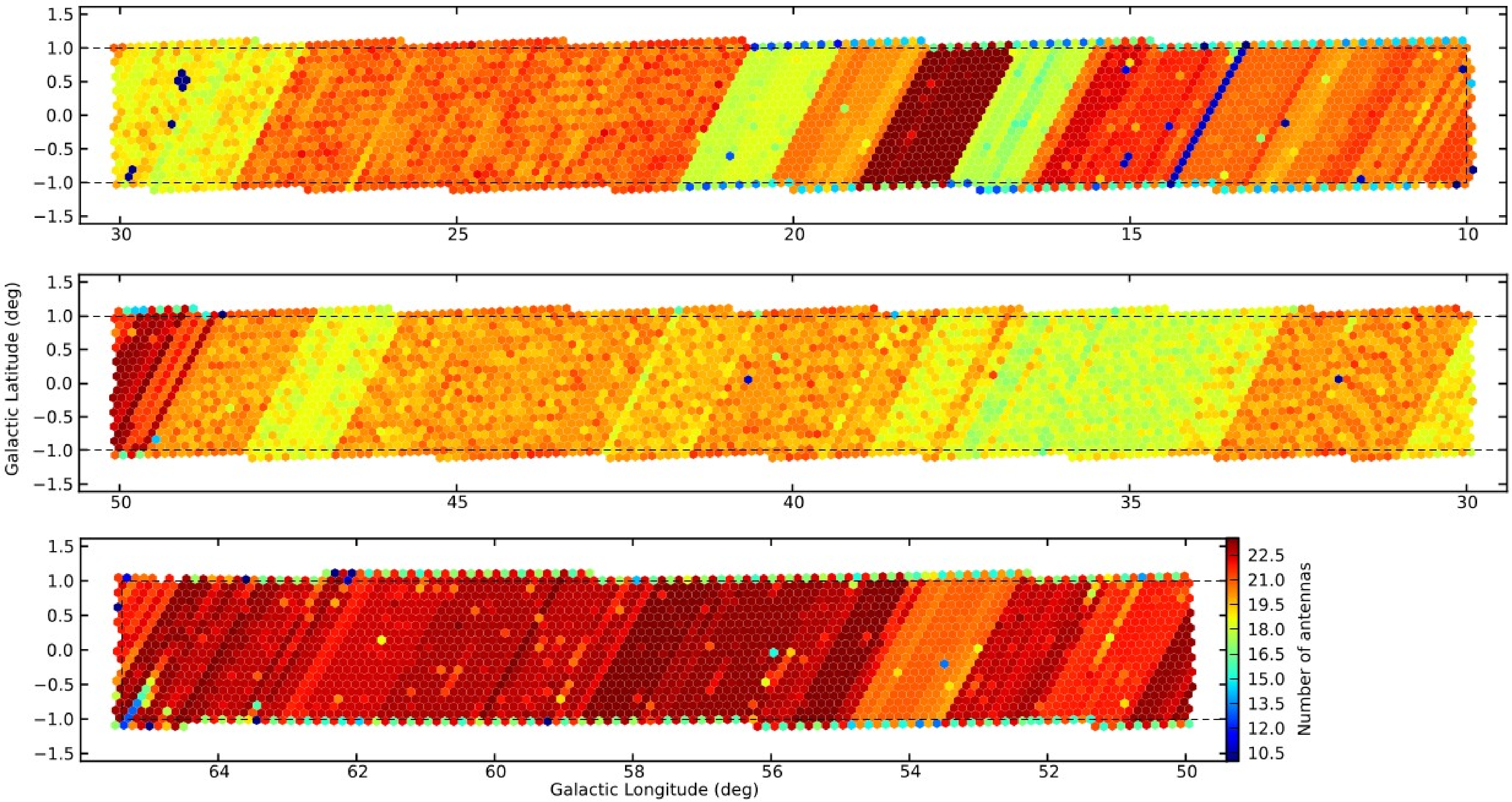
Dwell Times:



- Actual dwell times 77 – 80 seconds for most fields

Pipeline case study – CORNISH

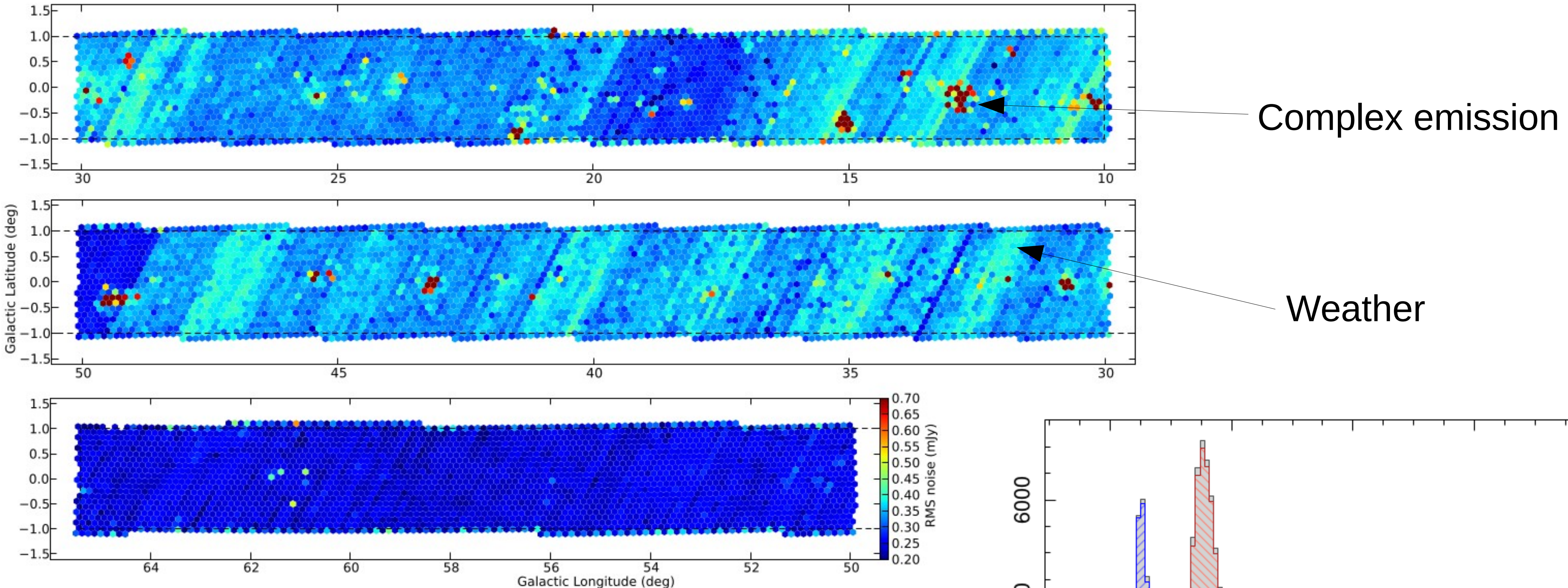
Effective number of antennas:



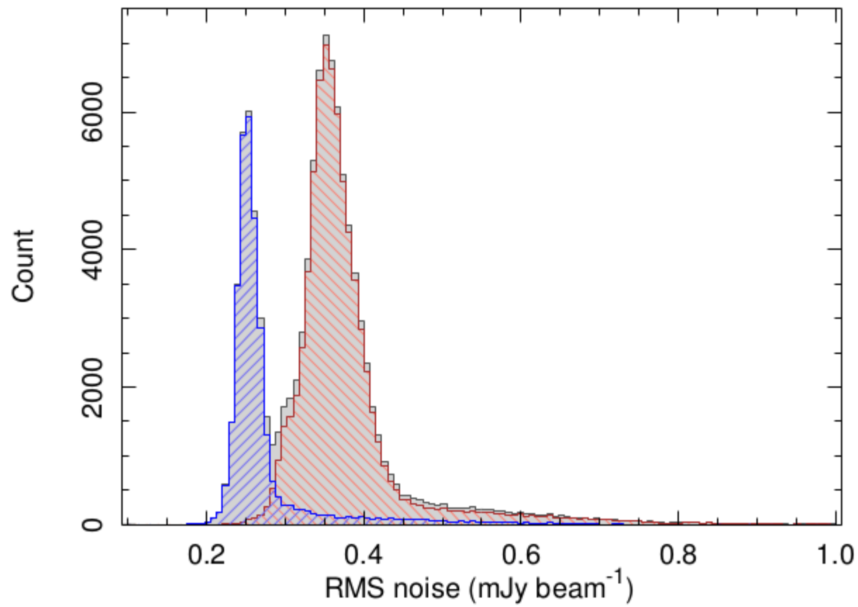
- After flagging > 18 effective antennas for most fields.

Pipeline case study – CORNISH

RMS noise per field (scale: 0.2 -0.7 mJy/beam)

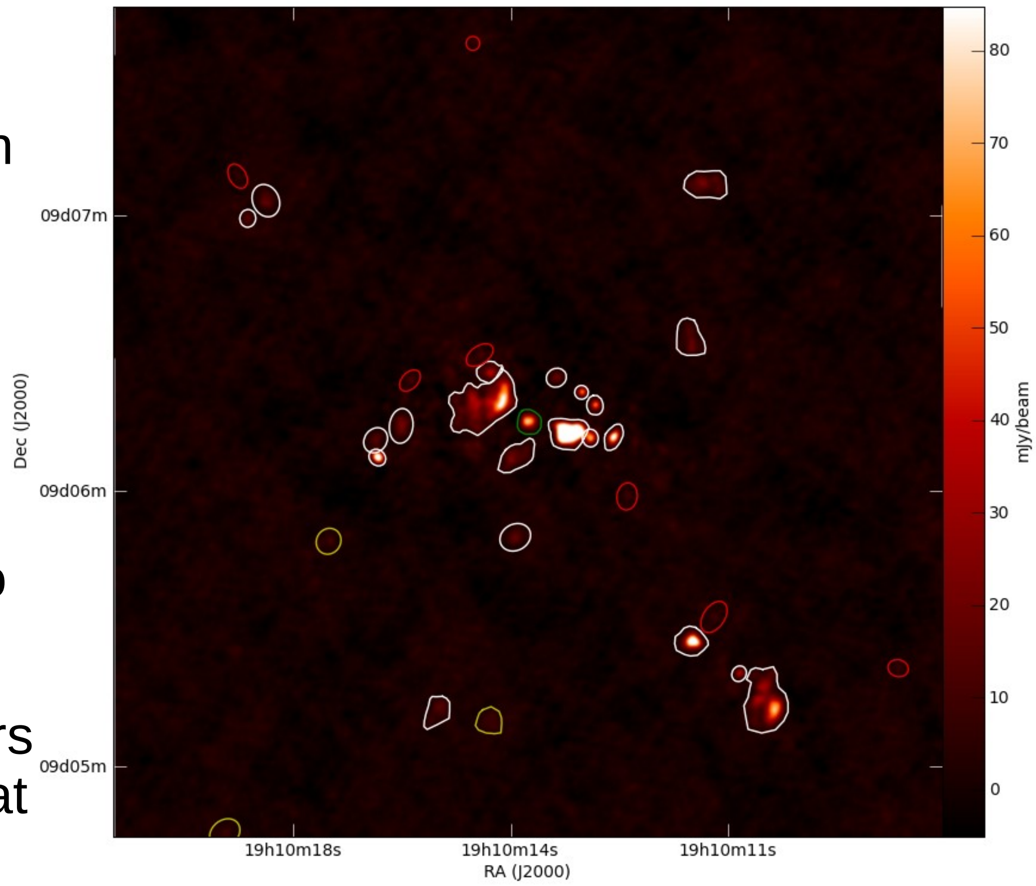


- Sensitivity split between two Epochs
  - 0.25 mJy/beam (EVLA, high-dec)
  - 0.35 mJy/beam (mixed VLA/EVLA)



Pipeline case study – CORNISH

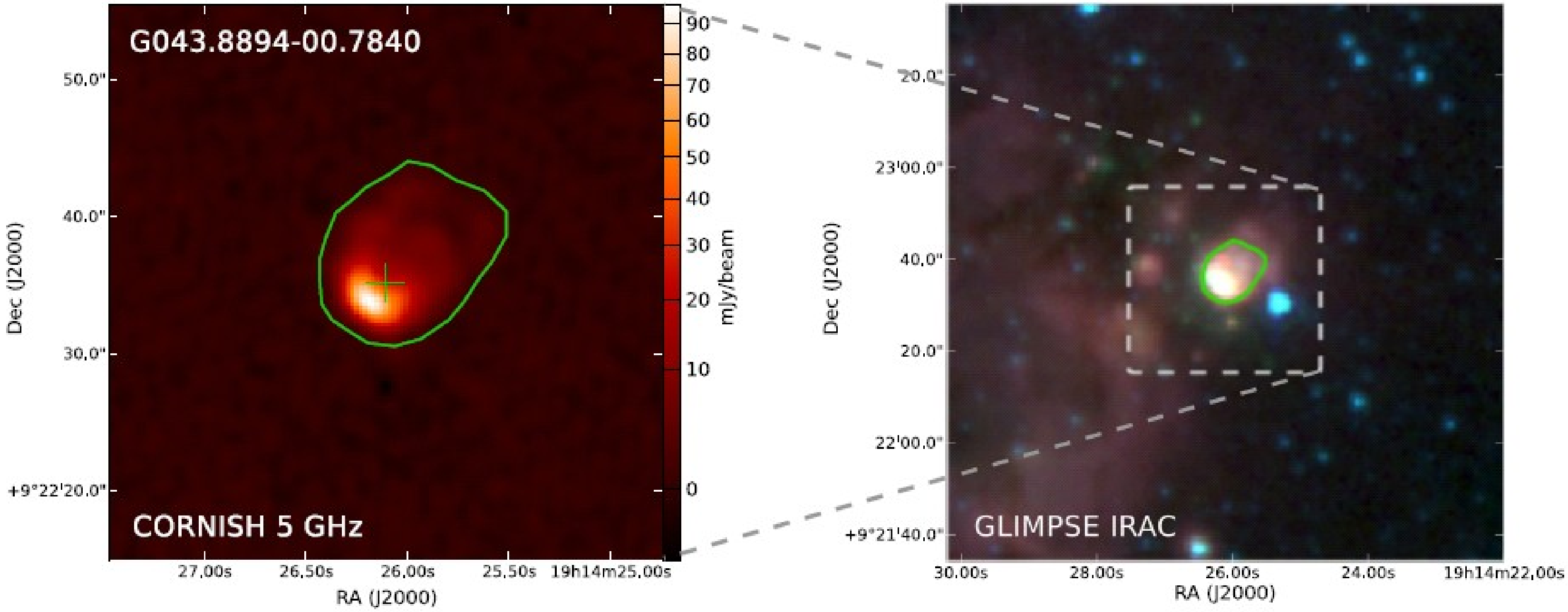
- Once the data was imaged progressed onto source-finding to create a catalogue
- Characterised unresolved emission using Gaussians (Obit *Soufnd*)
- Measured extended sources using polygonal aperture photometry
- Source finders wrote directly into the database and external python scripts were used to merge the two types of catalogue
- Note 1: More modern source finders available now - Aegean and Blobcat
- Note 2: Python Shapely module very useful for manipulating polygons http://toblerity.org/shapely/



https://github.com/PaulHancock/Aegean

http://blobcat.sourceforge.net/

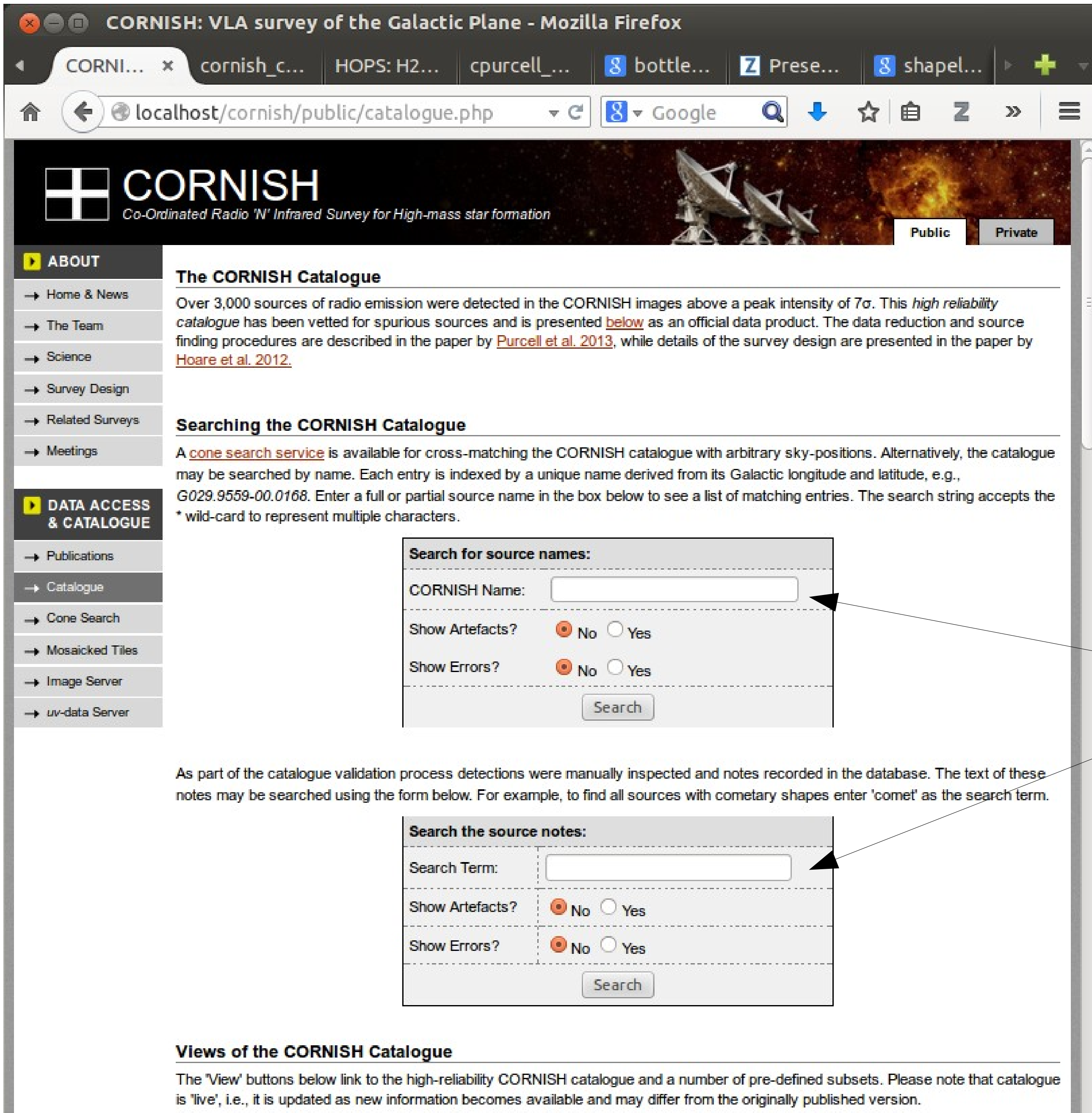
Pipeline case study – CORNISH



- **C**o-**O**rdinated **R**adio '**N**' **I**nfrared **S**urvey for **H**igh-mass star formation
- Key science requires reference to the mid-IR GLIMPSE data
- However, most objects of interest are **extended** – difficult to automate
- Solution: **Eyeball carnage**!

THE UNIVERSITY OF SYDNEY

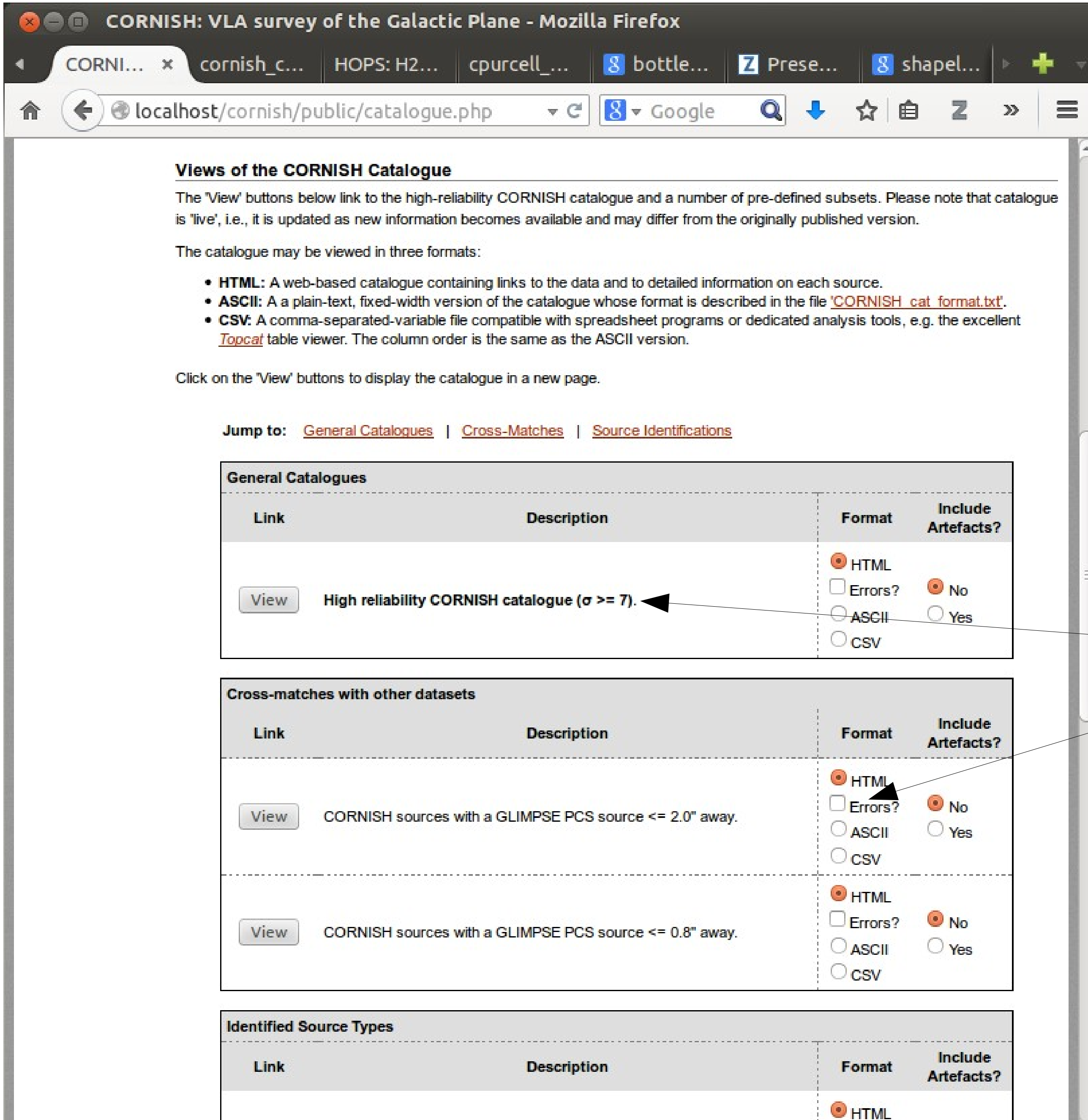
Pipeline case study – CORNISH



- Web interface to the image and catalogue data using Apache and Python CGI

Search tools:
- By partial name
- By user notes

Pipeline case study – CORNISH

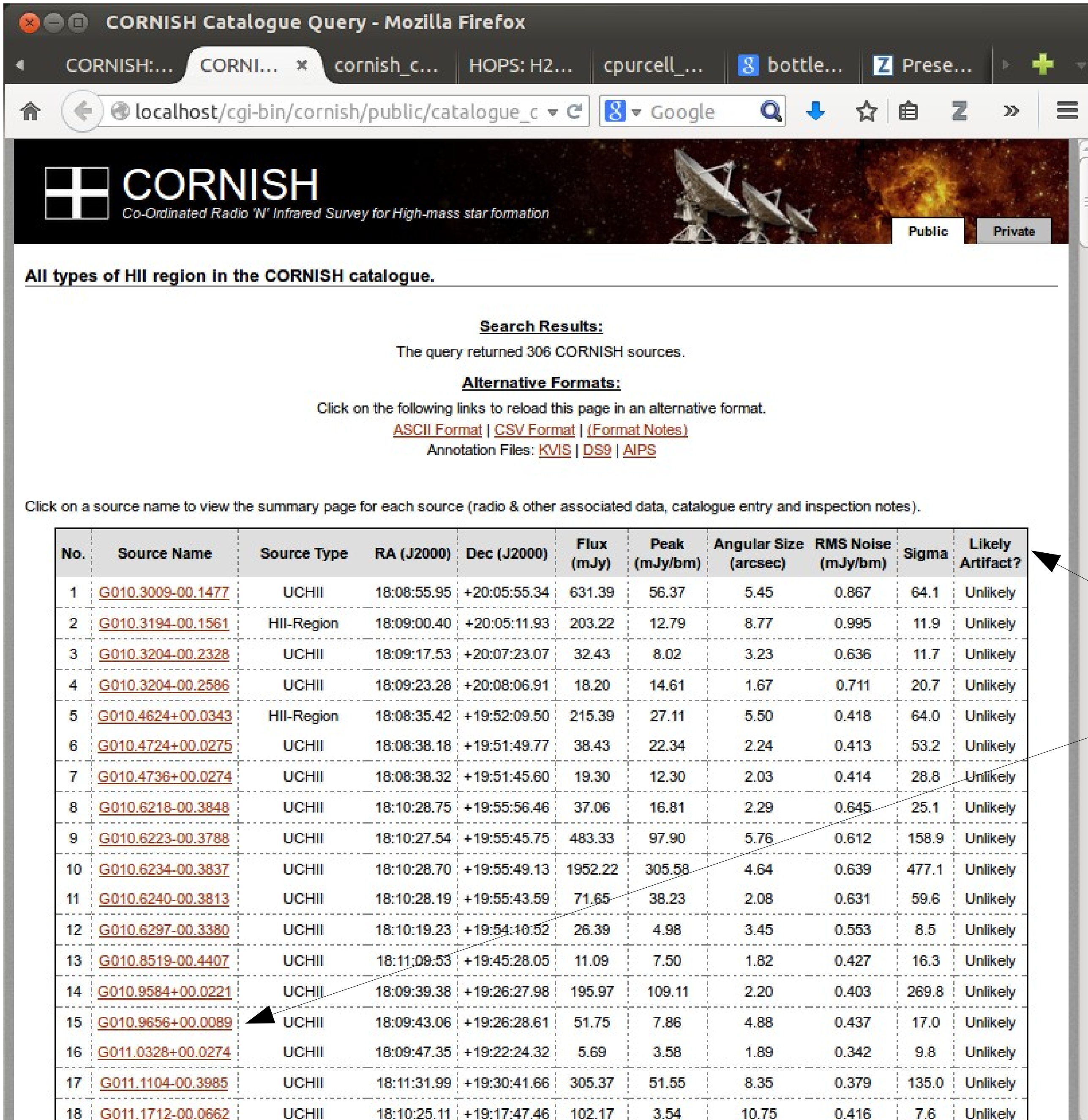


- Web interface to the image and catalogue data using Apache and Python CGI

Catalogue access

- Pre-defined SQL queries
- Links to catalogue and image server

Pipeline case study – CORNISH



CORNISH Catalogue Query - Mozilla Firefox

localhost/cgi-bin/cornish/public/catalogue_c

CORNISH
Co-Ordinated Radio 'N' Infrared Survey for High-mass star formation

Public | Private

All types of HII region in the CORNISH catalogue.

Search Results:

The query returned 306 CORNISH sources.

Alternative Formats:

Click on the following links to reload this page in an alternative format.

ASCII Format | CSV Format | (Format Notes)

Annotation Files: KVIS | DS9 | AIPS

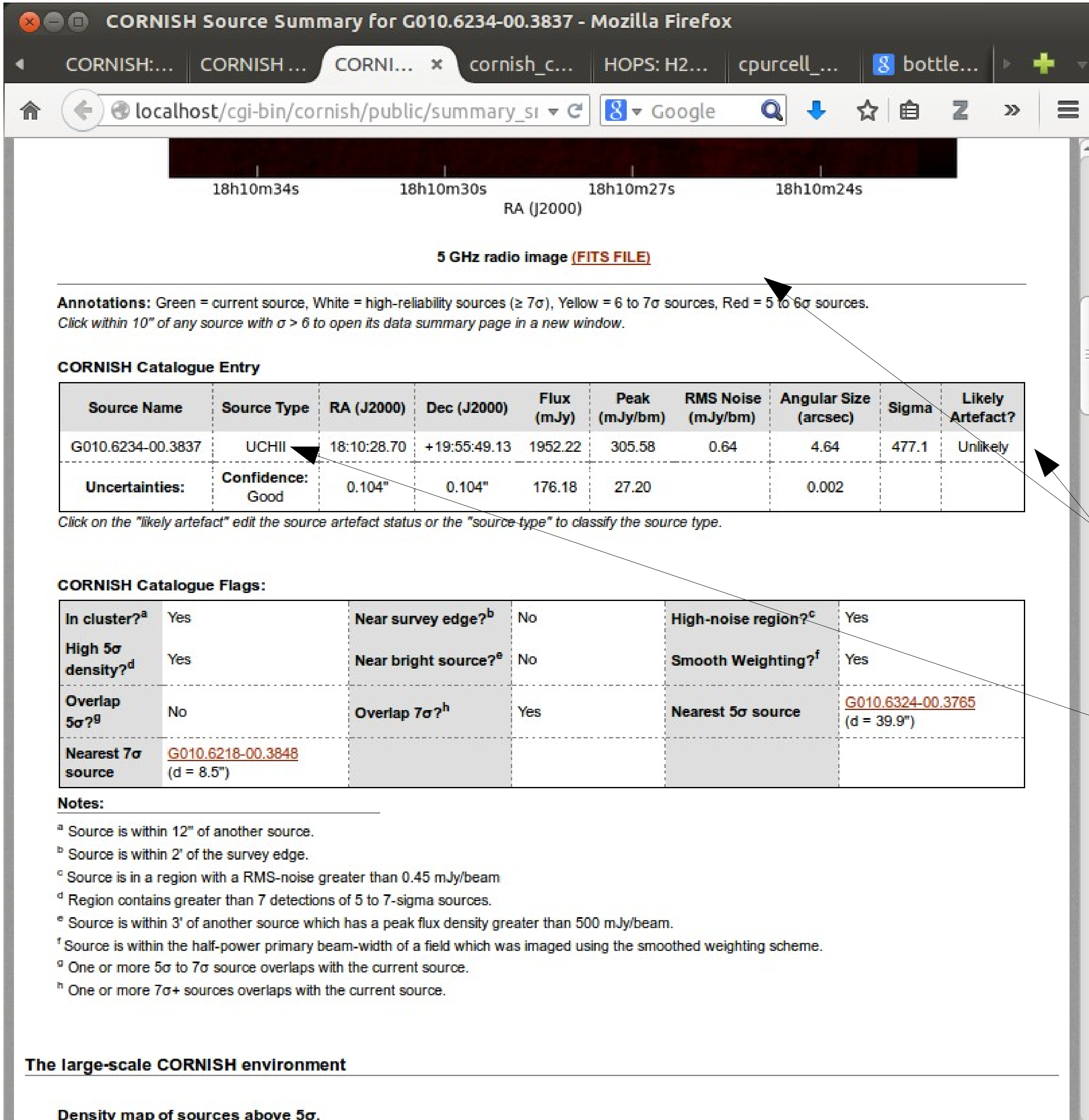
Click on a source name to view the summary page for each source (radio & other associated data, catalogue entry and inspection notes).

No.	Source Name	Source Type	RA (J2000)	Dec (J2000)	Flux (mJy)	Peak (mJy/bm)	Angular Size (arcsec)	RMS Noise (mJy/bm)	Sigma	Likely Artifact?
1	G010.3009-00.1477	UCHII	18:08:55.95	+20:05:55.34	631.39	56.37	5.45	0.867	64.1	Unlikely
2	G010.3194-00.1561	HII-Region	18:09:00.40	+20:05:11.93	203.22	12.79	8.77	0.995	11.9	Unlikely
3	G010.3204-00.2328	UCHII	18:09:17.53	+20:07:23.07	32.43	8.02	3.23	0.636	11.7	Unlikely
4	G010.3204-00.2586	UCHII	18:09:23.28	+20:08:06.91	18.20	14.61	1.67	0.711	20.7	Unlikely
5	G010.4624+00.0343	HII-Region	18:08:35.42	+19:52:09.50	215.39	27.11	5.50	0.418	64.0	Unlikely
6	G010.4724+00.0275	UCHII	18:08:38.18	+19:51:49.77	38.43	22.34	2.24	0.413	53.2	Unlikely
7	G010.4736+00.0274	UCHII	18:08:38.32	+19:51:45.60	19.30	12.30	2.03	0.414	28.8	Unlikely
8	G010.6218-00.3848	UCHII	18:10:28.75	+19:55:56.46	37.06	16.81	2.29	0.645	25.1	Unlikely
9	G010.6223-00.3788	UCHII	18:10:27.54	+19:55:45.75	483.33	97.90	5.76	0.612	158.9	Unlikely
10	G010.6234-00.3837	UCHII	18:10:28.70	+19:55:49.13	1952.22	305.58	4.64	0.639	477.1	Unlikely
11	G010.6240-00.3813	UCHII	18:10:28.19	+19:55:43.59	71.65	38.23	2.08	0.631	59.6	Unlikely
12	G010.6297-00.3380	UCHII	18:10:19.23	+19:54:10.52	26.39	4.98	3.45	0.553	8.5	Unlikely
13	G010.8519-00.4407	UCHII	18:11:09.53	+19:45:28.05	11.09	7.50	1.82	0.427	16.3	Unlikely
14	G010.9584+00.0221	UCHII	18:09:39.38	+19:26:27.98	195.97	109.11	2.20	0.403	269.8	Unlikely
15	G010.9656+00.0089	UCHII	18:09:43.06	+19:26:28.61	51.75	7.86	4.88	0.437	17.0	Unlikely
16	G011.0328+00.0274	UCHII	18:09:47.35	+19:22:24.32	5.69	3.58	1.89	0.342	9.8	Unlikely
17	G011.1104-00.3985	UCHII	18:11:31.99	+19:30:41.66	305.37	51.55	8.35	0.379	135.0	Unlikely
18	G011.1712-00.0662	UCHII	18:10:25.11	+19:17:47.46	102.17	3.54	10.75	0.416	7.6	Unlikely

- Web interface to the image and catalogue data using Apache and Python CGI

- Catalogue table filtered by SourceType HII region
- Links to source-summary and cross-match page

Pipeline case study – CORNISH


CORNISH Source Summary for G010.6234-00.3837 - Mozilla Firefox

localhost/cgi-bin/cornish/public/summary_s

18h10m34s 18h10m30s 18h10m27s 18h10m24s

RA (J2000)

5 GHz radio image (FITS FILE)

Annotations: Green = current source, White = high-reliability sources (≥ 7σ), Yellow = 6 to 7σ sources, Red = 5 to 6σ sources.
Click within 10" of any source with σ > 6 to open its data summary page in a new window.

CORNISH Catalogue Entry

Source Name	Source Type	RA (J2000)	Dec (J2000)	Flux (mJy)	Peak (mJy/bm)	RMS Noise (mJy/bm)	Angular Size (arcsec)	Sigma	Likely Artefact?
G010.6234-00.3837	UCHII	18:10:28.70	+19:55:49.13	1952.22	305.58	0.64	4.64	477.1	Unlikely
Uncertainties:	**Confidence:** Good	0.104"	0.104"	176.18	27.20		0.002		

Click on the "likely artefact" edit the source artefact status or the "source type" to classify the source type.

CORNISH Catalogue Flags:

In cluster?[a]	Yes	**Near survey edge?**[b]	No	**High-noise region?**[c]	Yes
High 5σ density?[d]	Yes	**Near bright source?**[e]	No	**Smooth Weighting?**[f]	Yes
Overlap 5σ?[g]	No	**Overlap 7σ?**[h]	Yes	**Nearest 5σ source**	G010.6324-00.3765 (d = 39.9")
Nearest 7σ source	G010.6218-00.3848 (d = 8.5")				

Notes:

[a] Source is within 12" of another source.
[b] Source is within 2' of the survey edge.
[c] Source is in a region with a RMS-noise greater than 0.45 mJy/beam
[d] Region contains greater than 7 detections of 5 to 7-sigma sources.
[e] Source is within 3' of another source which has a peak flux density greater than 500 mJy/beam.
[f] Source is within the half-power primary beam-width of a field which was imaged using the smoothed weighting scheme.
[g] One or more 5σ to 7σ source overlaps with the current source.
[h] One or more 7σ+ sources overlaps with the current source.

The large-scale CORNISH environment

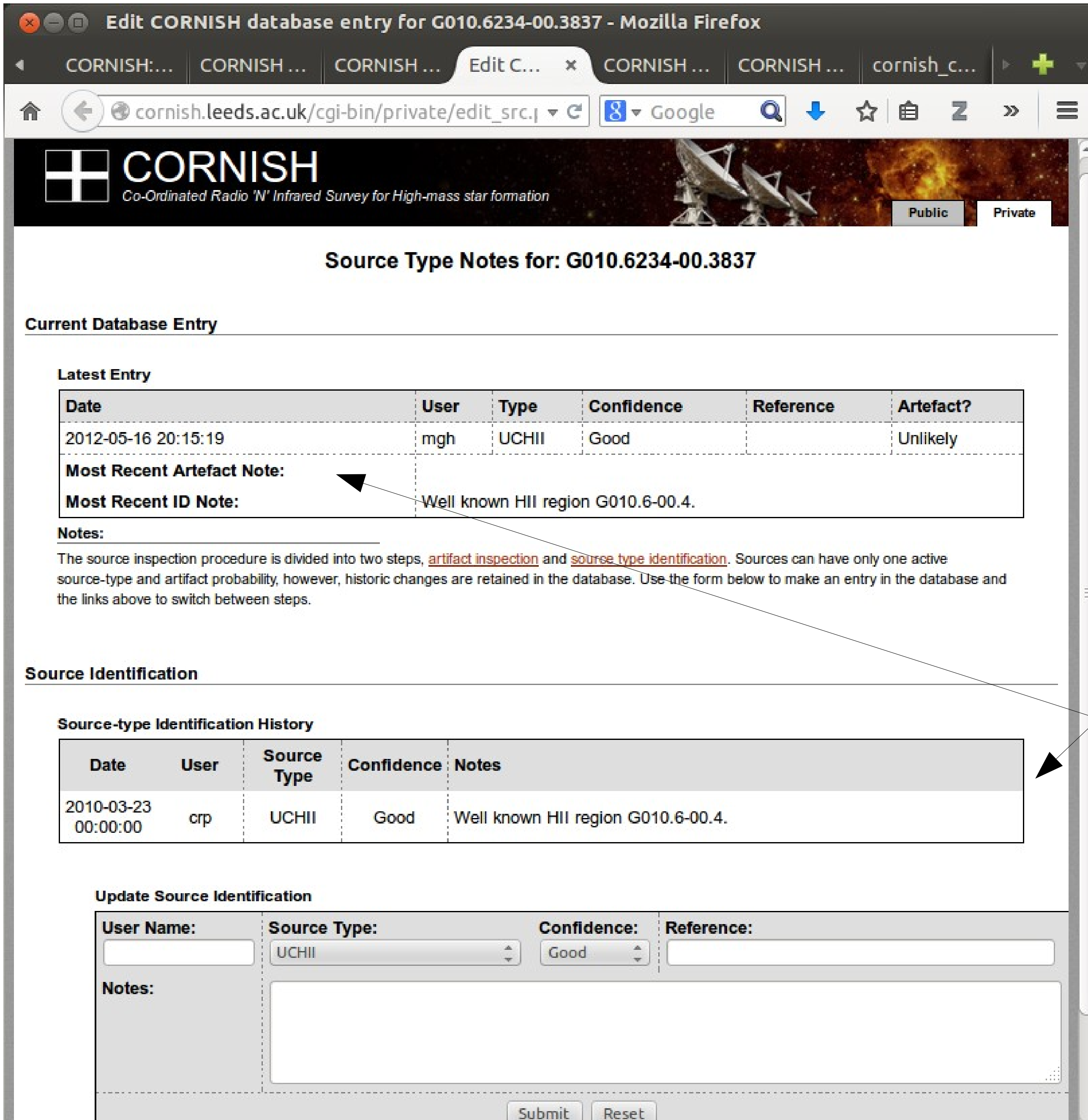
Density map of sources above 5σ.

- Web interface to the image and catalogue data using Apache and Python CGI

- Source summary page
- Details of source pulled from the database
- Links to edit pages

Made using APLpy

Pipeline case study – CORNISH

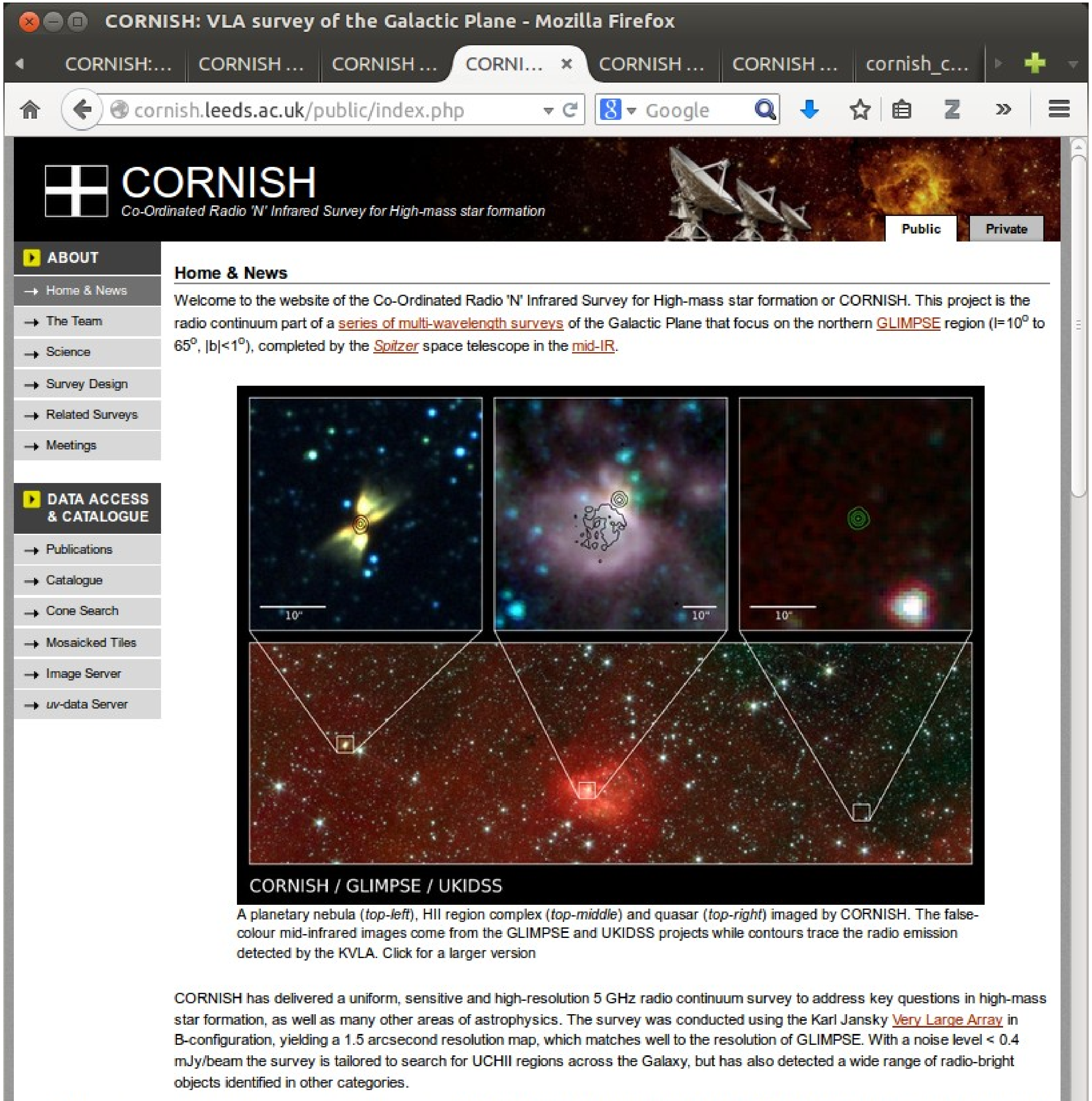


- Web interface to the image and catalogue data using Apache and Python CGI

- Form allowing editing of identified source type and notes in the DB.
- Tracks history and user

THE UNIVERSITY OF SYDNEY

Pipeline case study – CORNISH



- Live web page at:
  http://cornish.leeds.ac.uk
- Catalogue server
- uv-date server
- Image cutout server and clickable maps
- Batch Image server using uploaded catalogue
- Cross-matching service
- All done in Python

Summary and notes on the future

THE UNIVERSITY OF SYDNEY

- Take-away message:
  - Python can stitch practically *everything* together and **make your life easier**
  - Huge range of well-tested libraries (although don't treat as a black-box)
  - Lends itself well collaborating on the web and publishing data
  - If you are short on time and just want to learn **one tool – learn Python**
- Caveats:
  - Not always the best tool for the job
  - For very large web projects javascript browser based plotting will be superior

Thanks for listening!

Links on **next slide**

Resources

- NumPy RecArrays:
  - http://docs.scipy.org/doc/numpy-1.8.1/reference/generated/numpy.recarray.html
  - http://docs.scipy.org/doc/numpy/user/basics.rec.html
- SQL & databases:
  - https://docs.python.org/2/library/sqlite3.html
  - http://zetcode.com/db/sqlitepythontutorial/
  - https://www.sqlite.org/lang.html
  - http://mysql-python.sourceforge.net/MySQLdb.html
  - http://zetcode.com/db/mysqlpython/
- Reqular expressions:
  - https://docs.python.org/2/howto/regex.html
  - http://www.tutorialspoint.com/python/python_reg_expressions.htm
- Plotting:
  - http://aplpy.github.io/
  - http://matplotlib.org/ And also http://plplot.sourceforge.net/ (for PGPLOT users)