POSSUM: analysis for early science and beyond

Cormac Purcell, Bryan Gaensler and the POSSUM team

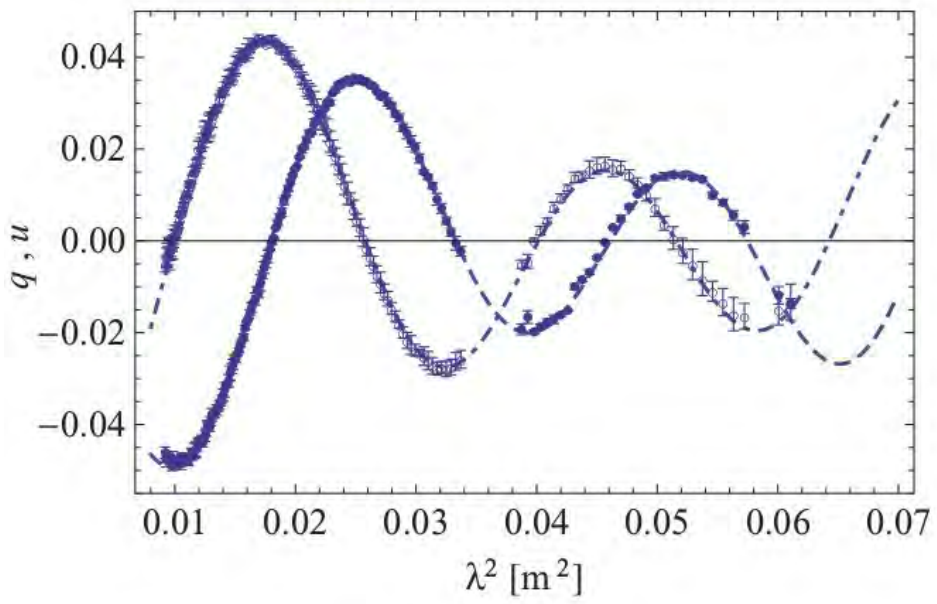
THE UNIVERSITY OF SYDNEY

2016-06-02

ASKAP 2016

askap.org/possum

POSSUM-12 versus POSSUM-36



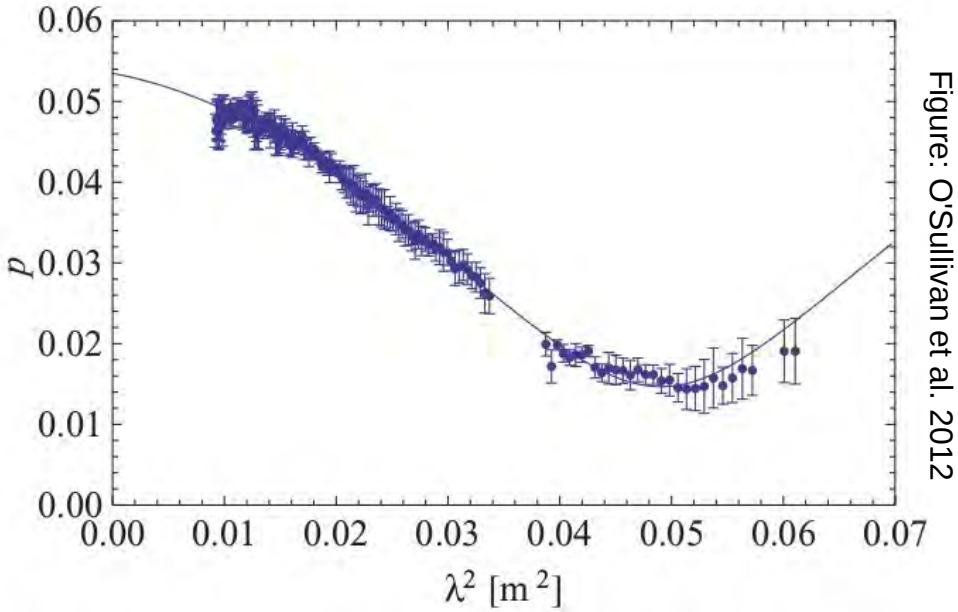


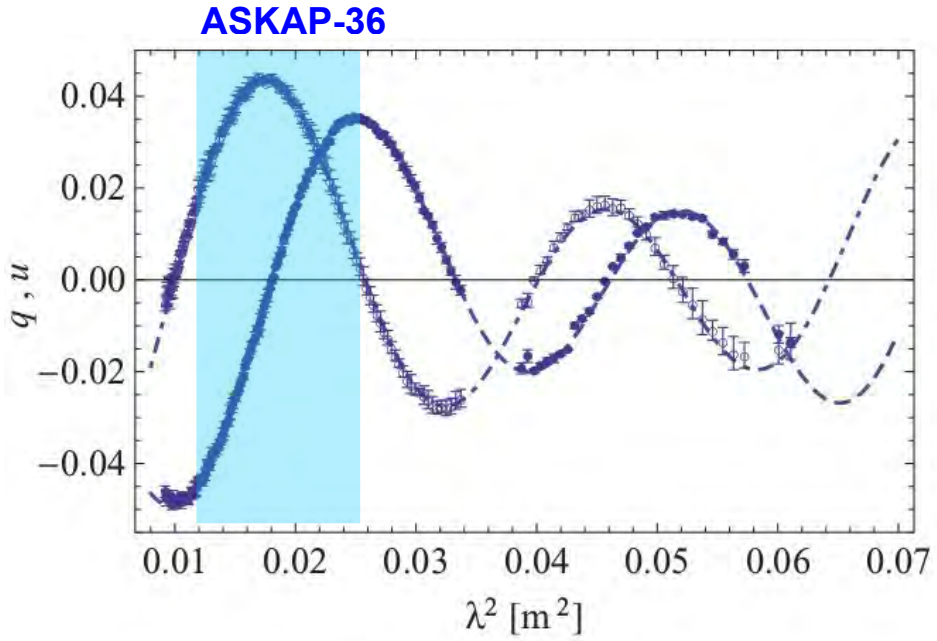
Figure: O'Sullivan et al. 2012

$$P = p_0 e^{2i(\Psi_0 + RM_1 \lambda^2)} + p_0 e^{2i(\Psi_0 + RM_2 \lambda^2)}$$

Rotation 1

Rotation 2

POSSUM-12 versus POSSUM-36



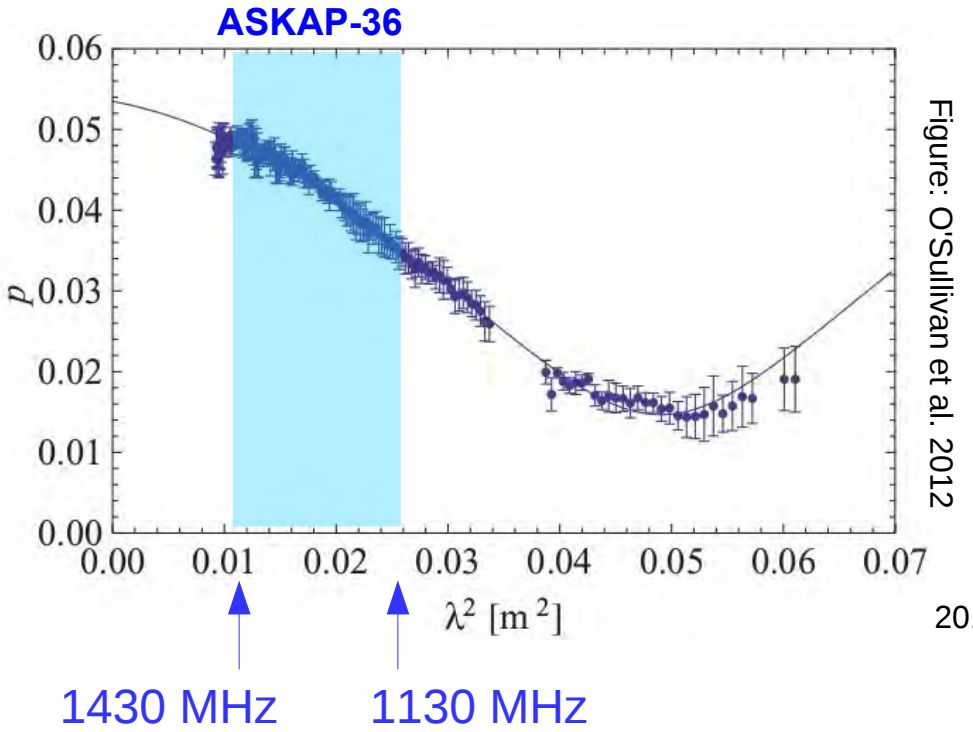
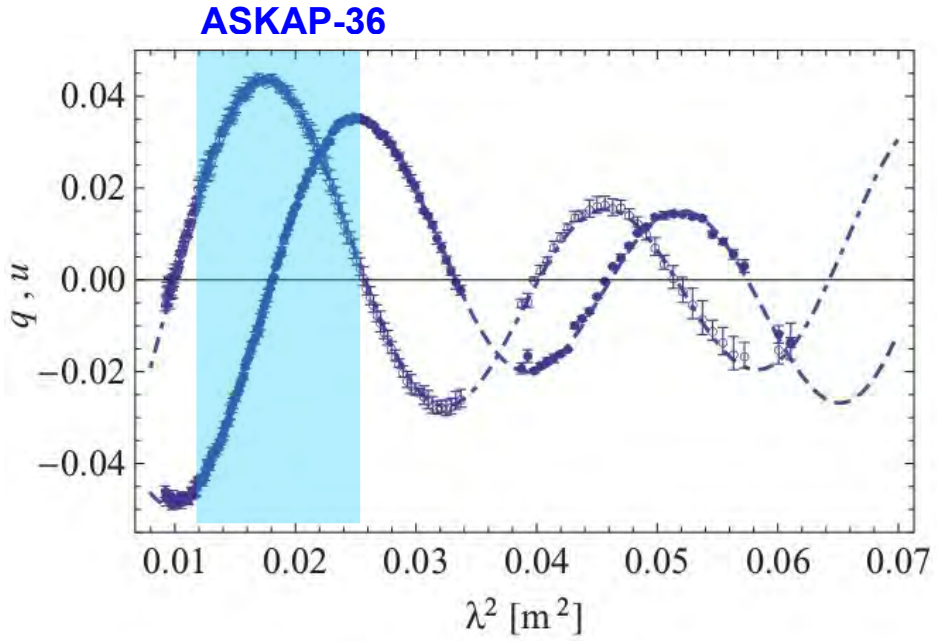


Figure: O'Sullivan et al. 2012

2012

POSSUM-12 versus POSSUM-36



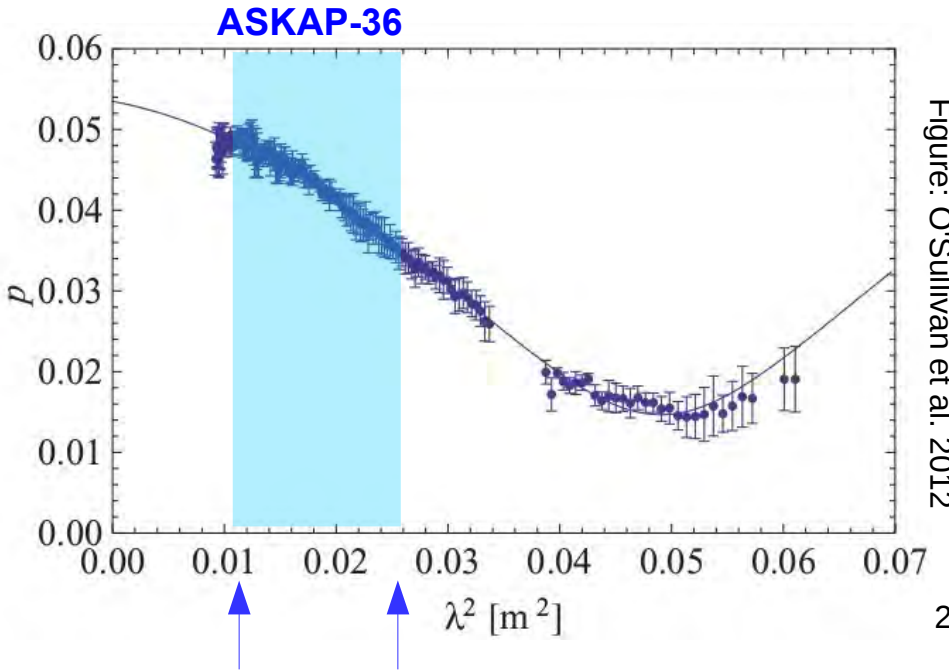
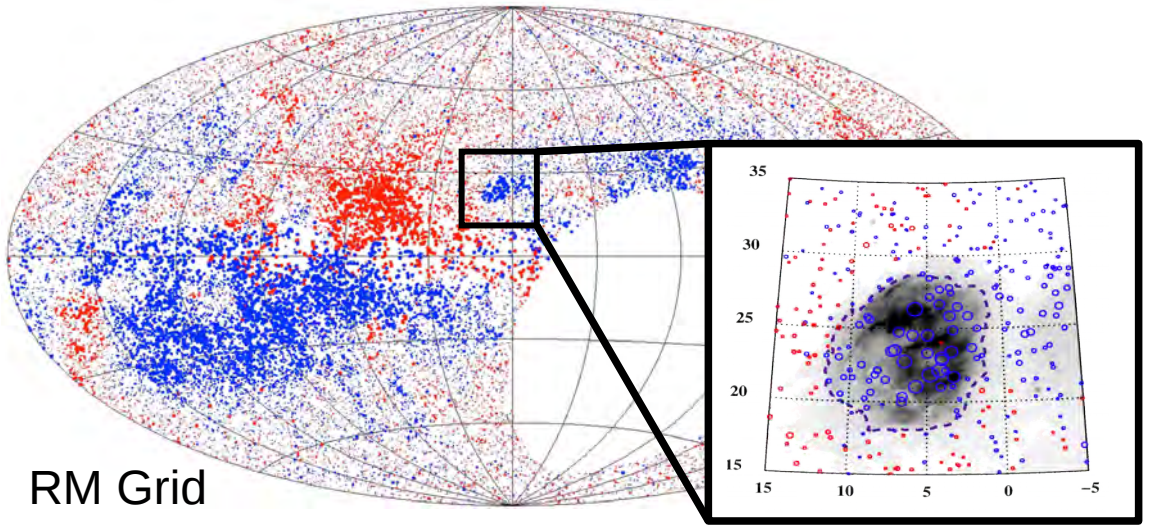


Figure: O'Sullivan et al. 2012

2012

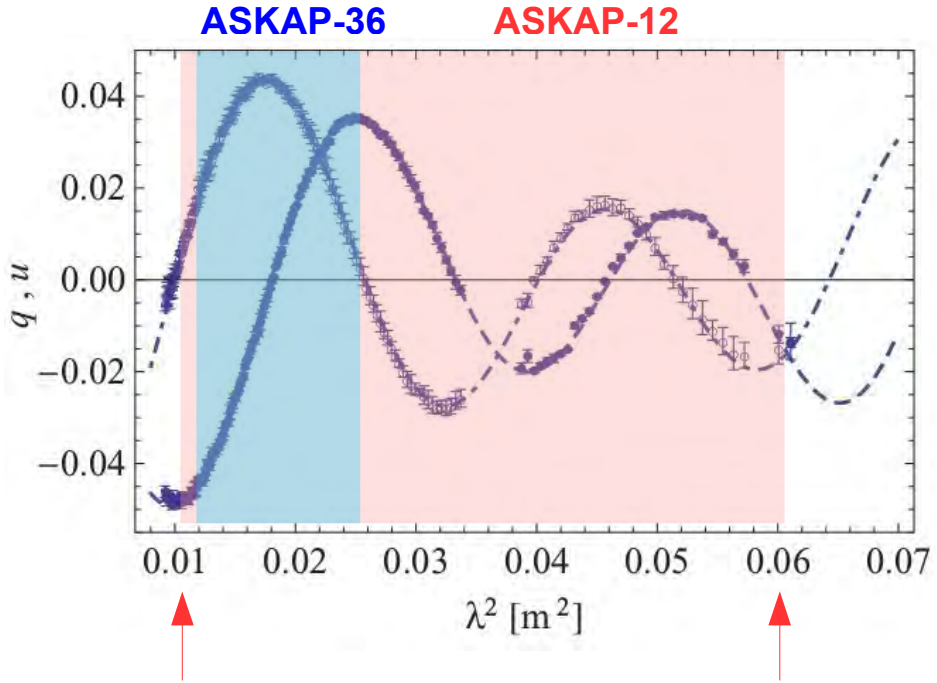
- POSSUM-36: Detect simple Faraday rotating screens



Taylor, Stil & Sunstrum 2009

H II region around ζ Oph (Harvey-Smith, Madsen & BMG 2011)

POSSUM-12 versus POSSUM-36



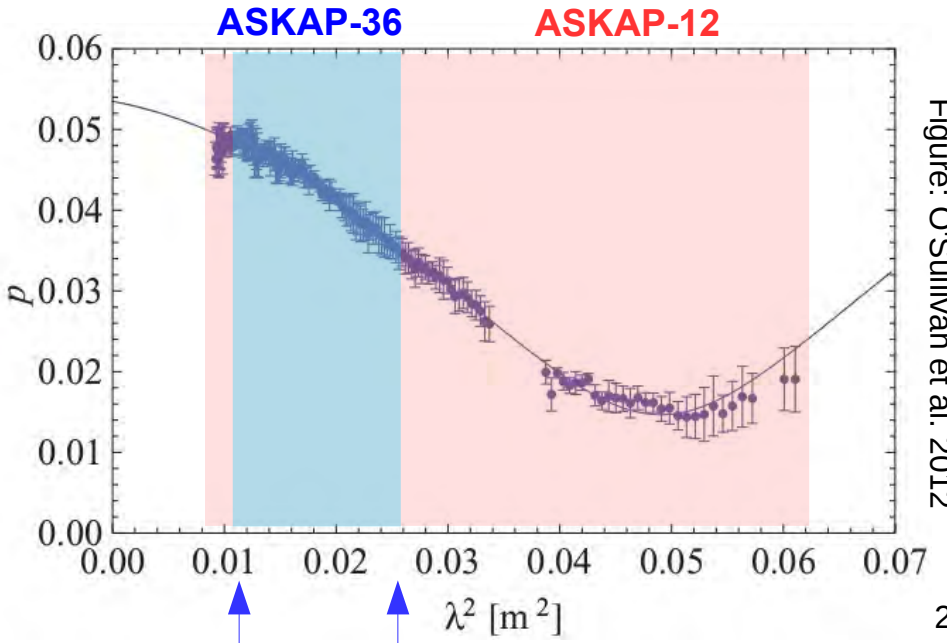


Figure: O'Sullivan et al. 2012

2012

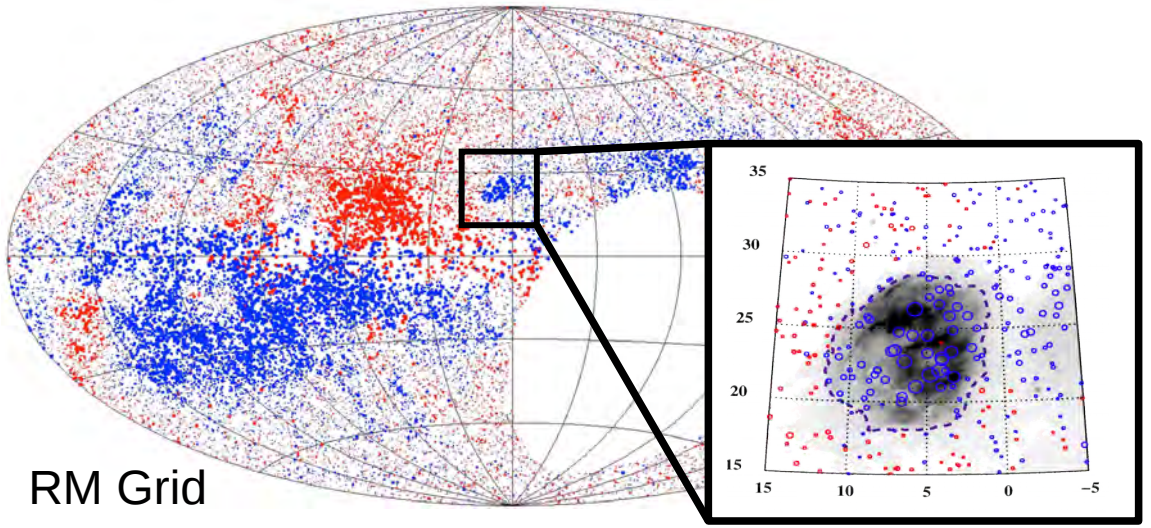
1800 MHz

700 MHz

1430 MHz

1130 MHz

- POSSUM-36: Detect simple Faraday rotating screens

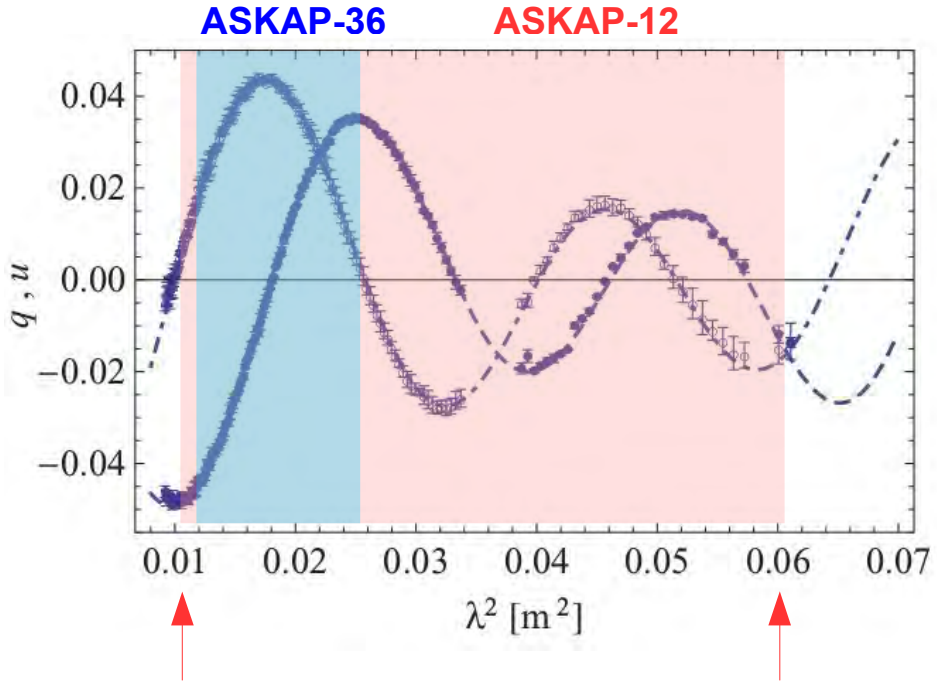


Taylor, Stil & Sunstrum 2009

RM Grid

H II region around ζ Oph (Harvey-Smith, Madsen & BMG 2011)

POSSUM-12 versus POSSUM-36



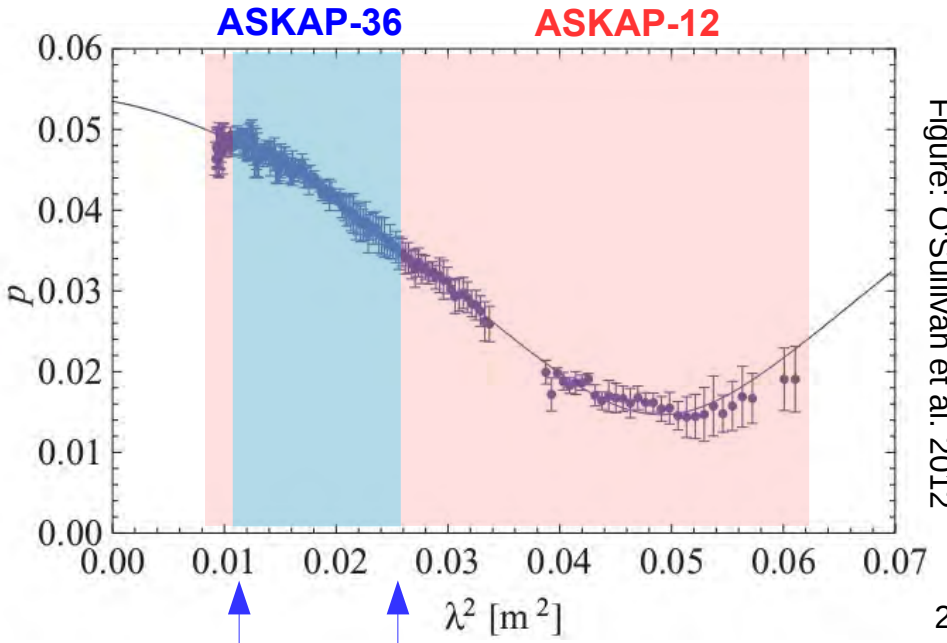


Figure: O'Sullivan et al. 2012

2012

1800 MHz

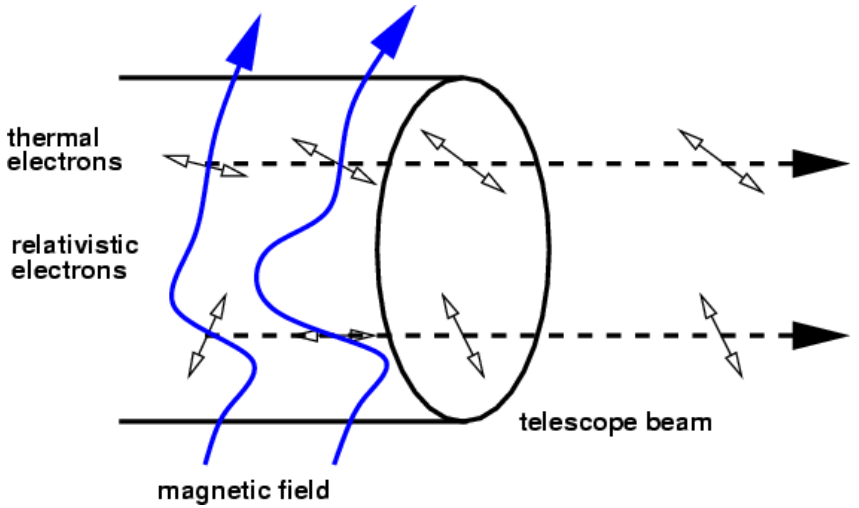
700 MHz

1430 MHz

1130 MHz

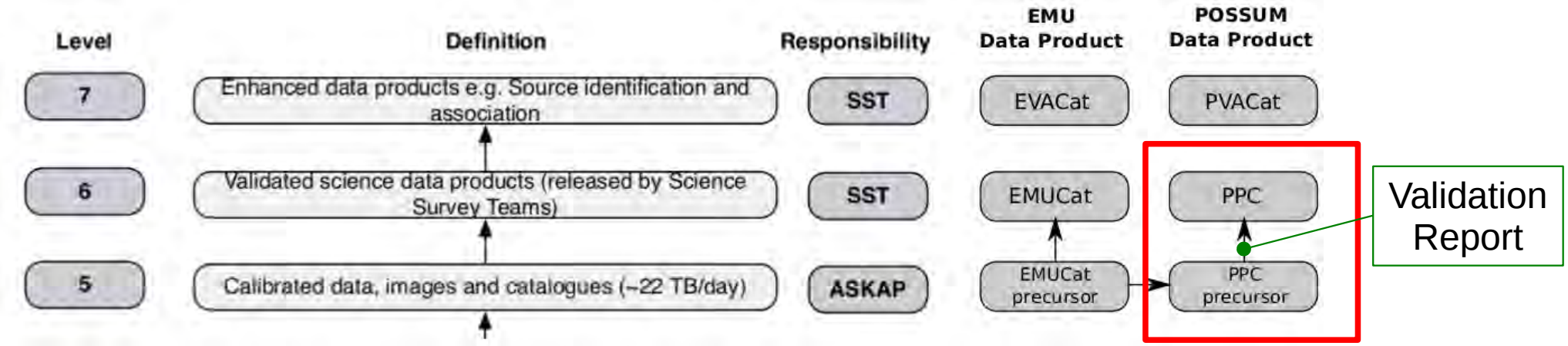
- POSSUM-36: Detect simple Faraday rotating screens

- POSSUM-12: Detect "complex" polarisation mechanisms



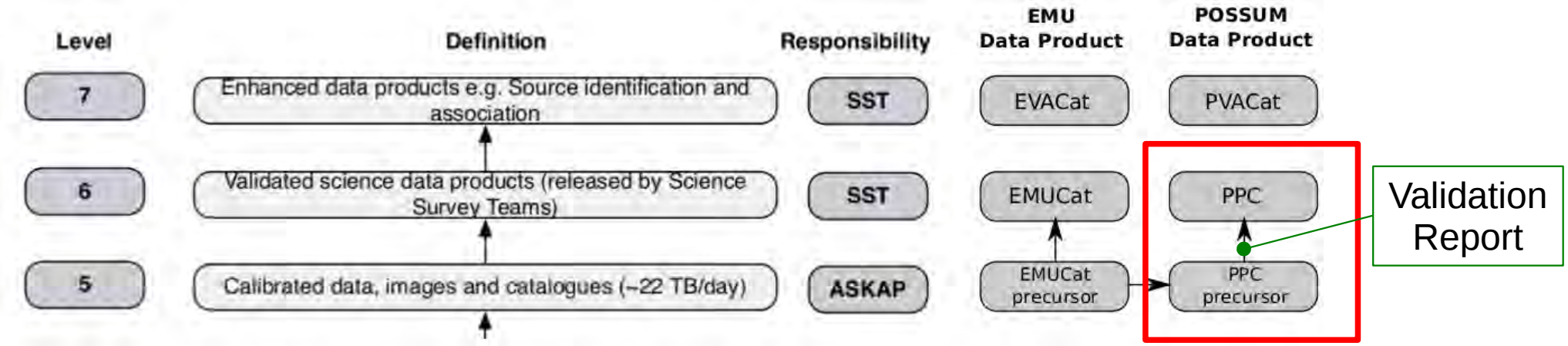
Fletcher et al 2004

POSSUM Polarisation Catalogues



- POSSUM team defines 3 different catalogues:

POSSUM Polarisation Catalogues

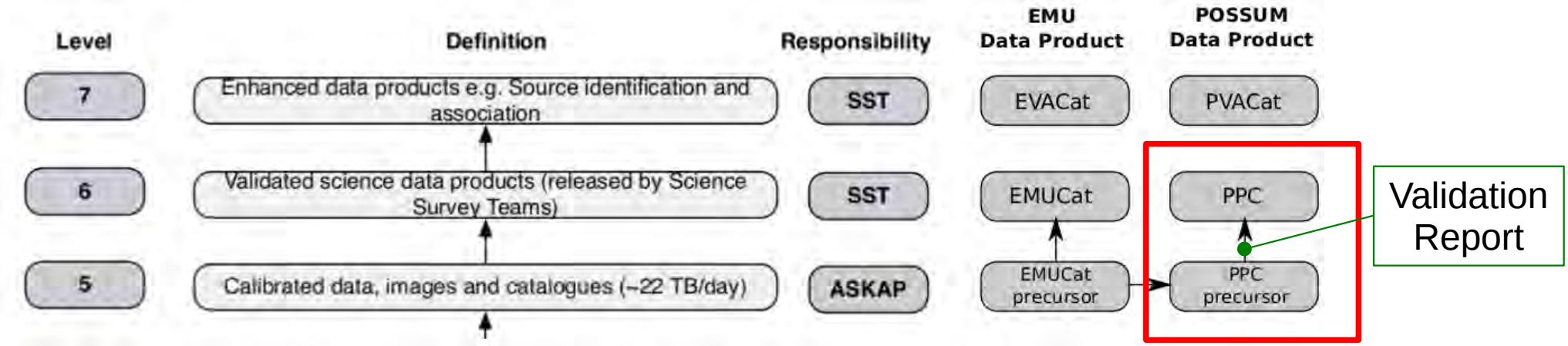


- POSSUM team defines 3 different catalogues:

POSSUM Polarisation Catalogue (PPC)

- **ASKAP-36**
- RM-Synthesis & simple modelling

POSSUM Polarisation Catalogues



- POSSUM team defines 3 different catalogues:

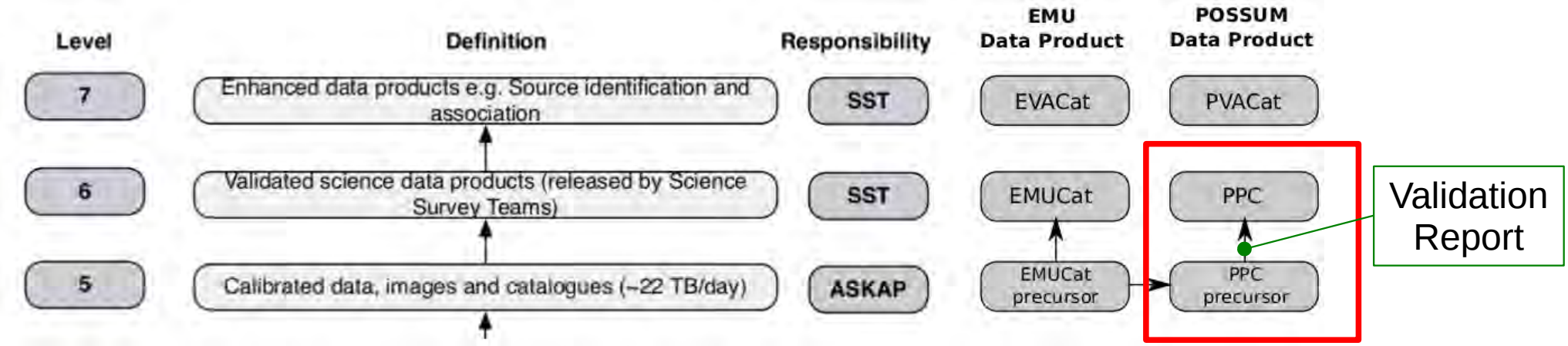
POSSUM Polarisation Catalogue (PPC)

- **ASKAP-36**
- RM-Synthesis & simple modelling

POSSUM Broad-Band Catalogue (PBCat)

- **ASKAP-12**
- Complex Modelling

POSSUM Polarisation Catalogues



- POSSUM team defines 3 different catalogues:

POSSUM Polarisation Catalogue (PPC)

- **ASKAP-36**
- RM-Synthesis & simple modelling

POSSUM Broad-Band Catalogue (PBCat)

- **ASKAP-12**
- Complex Modelling

POSSUM Value Added Catalogue (PBCat)

- **ASKAP-12 & 36**
- Complex Modelling (x, y z, φ) including other sources of information, e.g., synchrotron intensity.

ASKAP & POSSUM

uv-data

Calibration & Imaging

Image Data

Source finding, & measurement

Stokes I Catalogue

Secondary processing

Polarisation Catalogue

Advanced data products

Implement for ASKAP

CSIRO

POSSUM

ASKAP & POSSUM

uv-data

Calibration & Imaging

Image Data

Source finding, & measurement

Stokes I Catalogue

Secondary processing

Polarisation Catalogue

Advanced data products

Implement for ASKAP

CSIRO

POSSUM

Design Phase

CSIRO / Other

POSSUM

ASKAP & POSSUM

uv-data

Calibration & Imaging

Image Data

Source finding, & measurement

Stokes I Catalogue

Secondary processing

Polarisation Catalogue

Advanced data products

Implement for ASKAP

Design Phase

CSIRO

CSIRO / Other

POSSUM

POSSUM

ASKAP & POSSUM

uv-data

Calibration & Imaging

Image Data

Source finding, & measurement

Stokes I Catalogue

Secondary processing

Polarisation Catalogue

Advanced data products

Implement for ASKAP

Design Phase

CSIRO

POSSUM

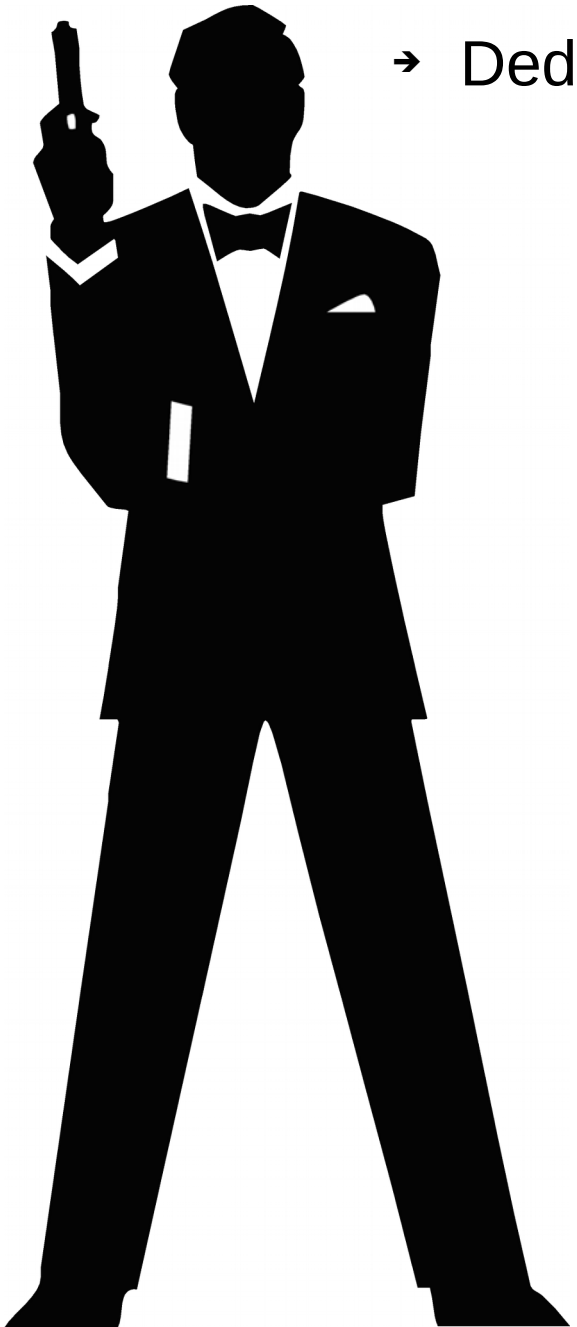
CSIRO / Other

POSSUM

Prototype Pipeline Goals:

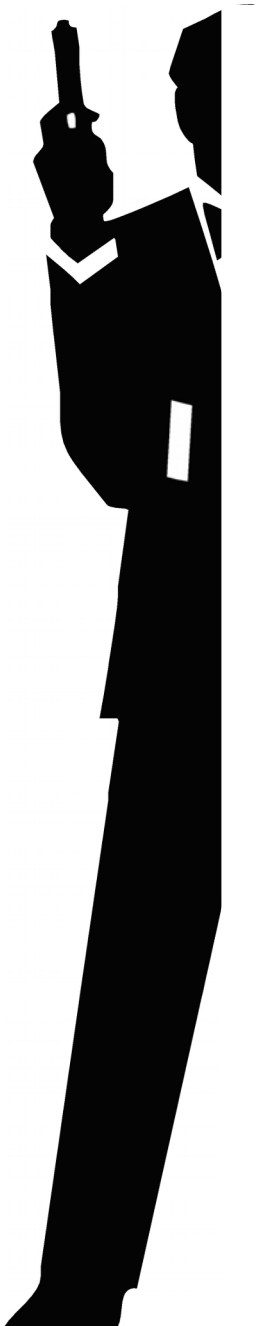
- Develop Algorithms
  - RM-determination.
  - Physical model fitting
  - Degeneracy / model choice
- End-to-end Integration
  - Catch 'edge cases' data
  - Useful catalogue flags
  - Analysis workflow
- Develop Quality Control
  - Pre-release quality report
  - Data quality metrics
  - Granular data selection
- Explore Existing Surveys
  - ATCA test fields, GALFACTS
- **Documentation by code**

Implementing the Prototype



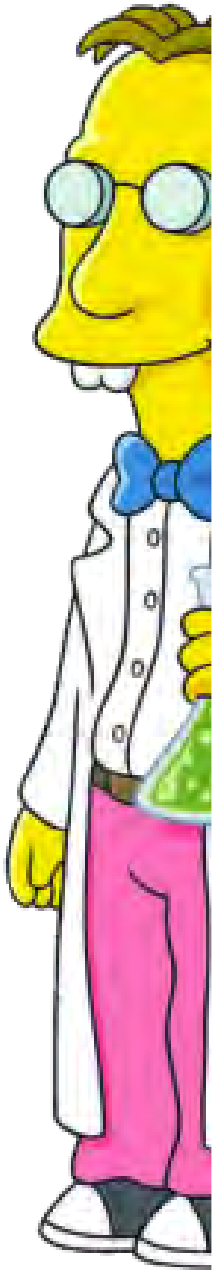
- Dedicated resources @ USyd

Implementing the Prototype



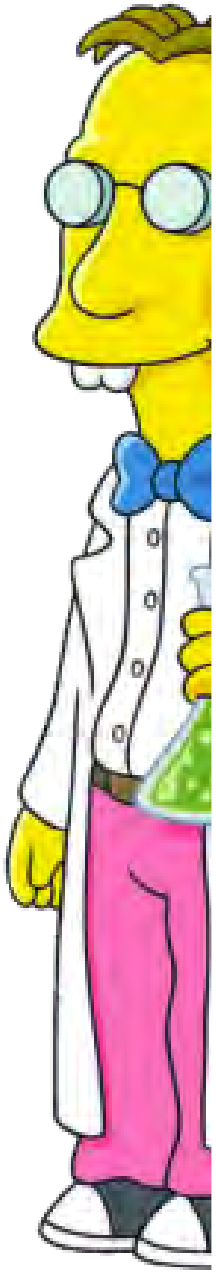
- Dedicated resources @ USyd – half of an astronomer

Implementing the Prototype



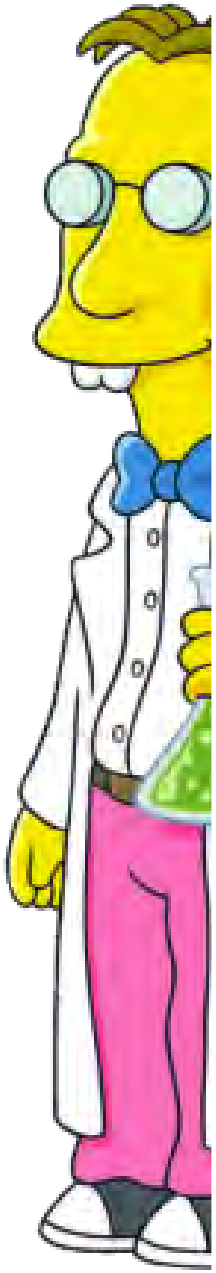
- Dedicated resources @ USyd – half of an ***astronomer***

Implementing the Prototype

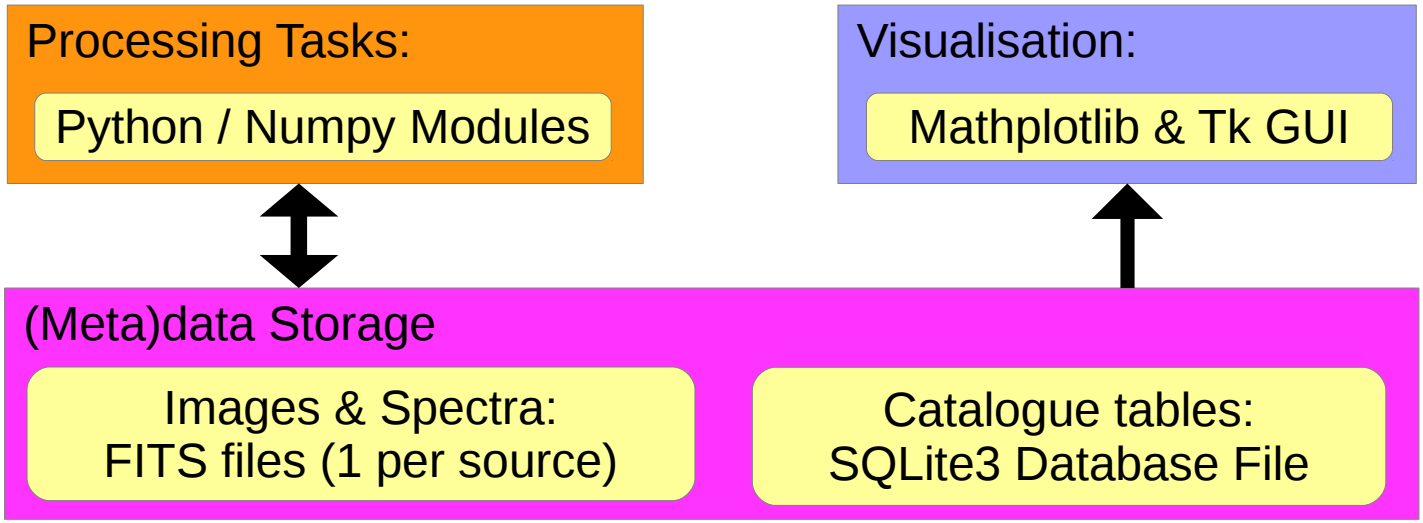


- Dedicated resources @ USyd – half of an ***astronomer***
- Portable, modular, easy to install , easy to run, easy to modify and applicable to current real-world data.

Implementing the Prototype

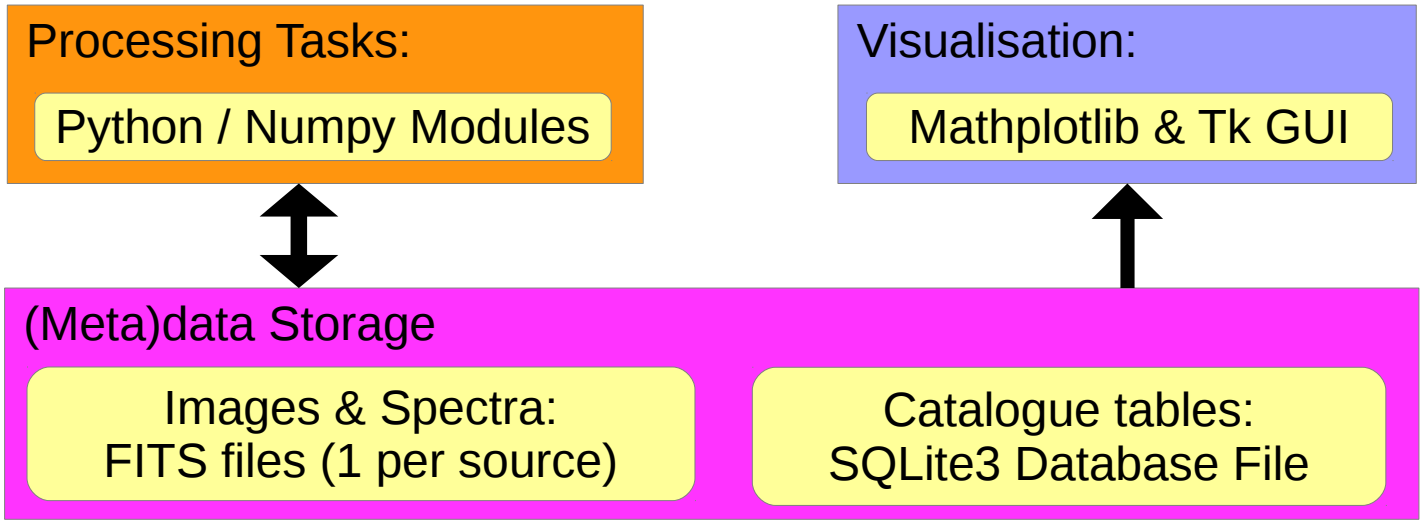


- Dedicated resources @ USyd – half of an ***astronomer***
- Portable, modular, easy to install , easy to run, easy to modify and applicable to current real-world data.



Implementing the Prototype

- Dedicated resources @ USyd – half of an ***astronomer***
- Portable, modular, easy to install , easy to run, easy to modify and applicable to current real-world data.



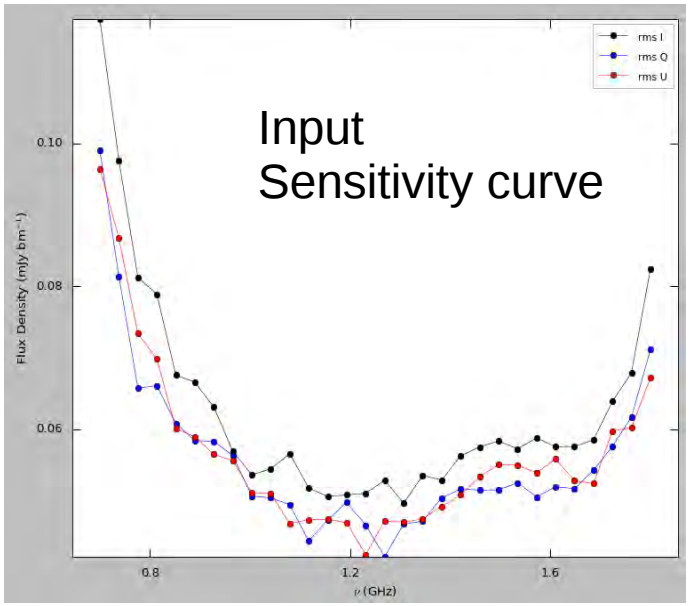
numpy … calculations
tkinter … visualisation
sqlite3 ... local database
mpfit & emcee … for model fitting
astropy.io.fits … FITS file I/O
astropy.io.wcs … WCS processing

Standard Python Library
Standard Astronomy Library
Distributed with Pipeline

Workflow

Pipeline Tasks:

0_gen_model_images.py

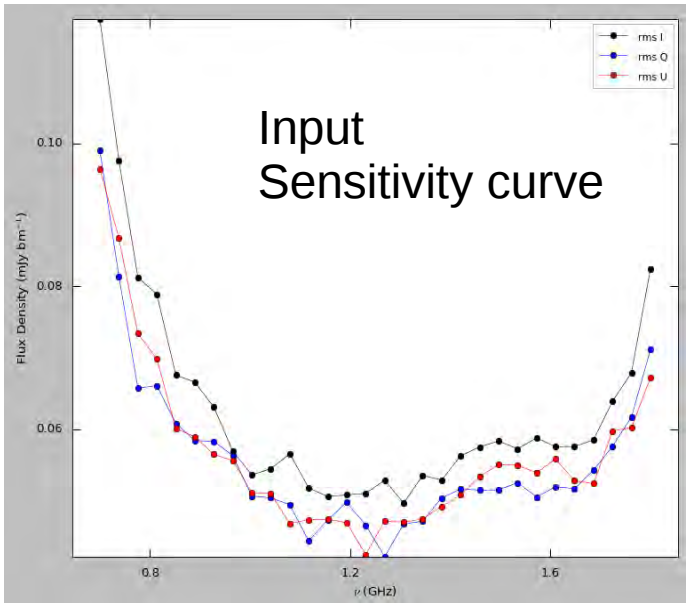



```
#-----------------------------------------------------------------------#
#                                                                       #
# Input catalogue file for use with the POSSUM pipeline.                #
# Used to generate artificial data for testing purposes.                #
#                                                                       #
# Note: spatial information (x,y,maj,min,pa) is ignored when generating 1D   #
#       ASCII data. If generating image data these are the world-coordinates  #
#       and parameters of the injected Gaussians.                       #
#                                                                       #
# C. Purcell 06-April-2016                                              #
#                                                                       #
#-----------------------------------------------------------------------#

# MODEL TYPE 1: External depolarised components within same beam (not LOS).
# Type, x_deg, y_deg, maj_asec, min_asec, pa_deg, fluxI_mJy, SI, [polFrac, evpa_deg, faradayDepth_radm2, far
#
# MODEL TYPE 2: Emitting & rotating regions stacked along the line-of-sight.
# Type, x_deg, y_deg, maj_asec, min_asec, pa_deg, fluxI_mJy, SI, [polFrac, evpa_deg, faradayDepth_radm2]xN
#

1, 89.9981, +0.0001, 0.0, 0.0, 0.0, 10.0, -0.0,     0.6, 180.0, 30.0, 0.0
1, 90.0016, +0.0023, 0.0, 0.0, 0.0, 0.5, -0.7,      0.4, 30.0, 100.0, 0.0,      0.3, 70.0, 80.0, 0.0
1, 90.0016, +0.0023, 0.0, 0.0, 0.0, 0.5, -0.7,      0.4, 30.0, 100.0, 30.0,      0.3, 70.0, 80.0, 30.0
2, 90.0026, +0.000, 0.0, 0.0, 0.0, 10.0, 0.0,       0.5, 45.0, -20.0
2, 90.0, +0.000, 0.0, 0.0, 0.0, 10.0, 0.0,          0.5, 45.0, -20.0,   0.25, 10, 50
2, 90.0, +0.000, 0.0, 0.0, 0.0, 10.0, 0.0,          0.4, 45.0, -30.0,   0.25, 10, 50,  0.25, 10, 90
```

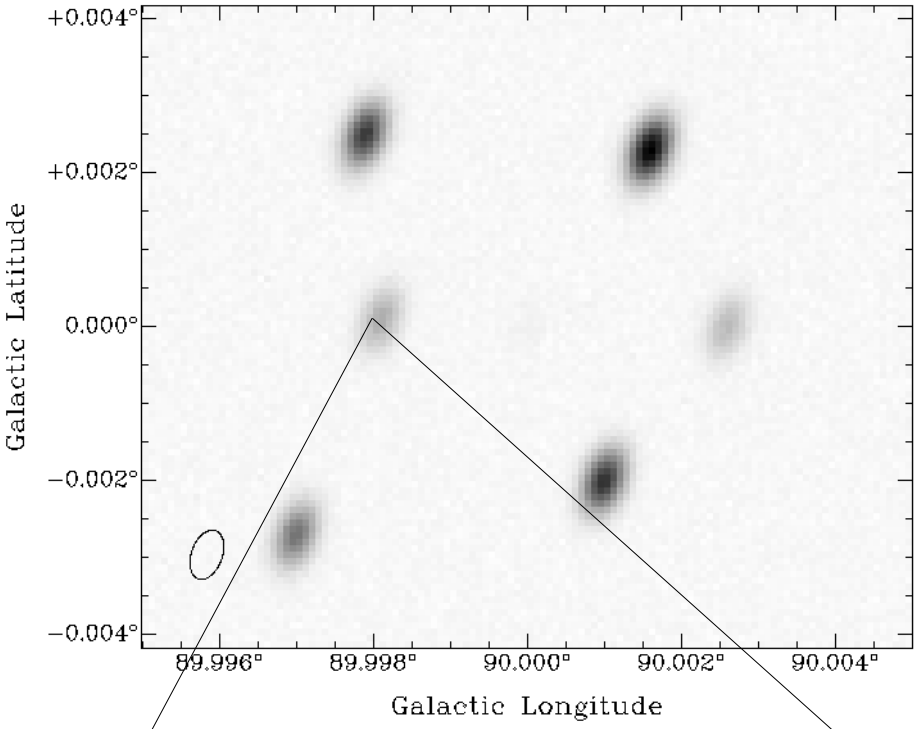
Workflow

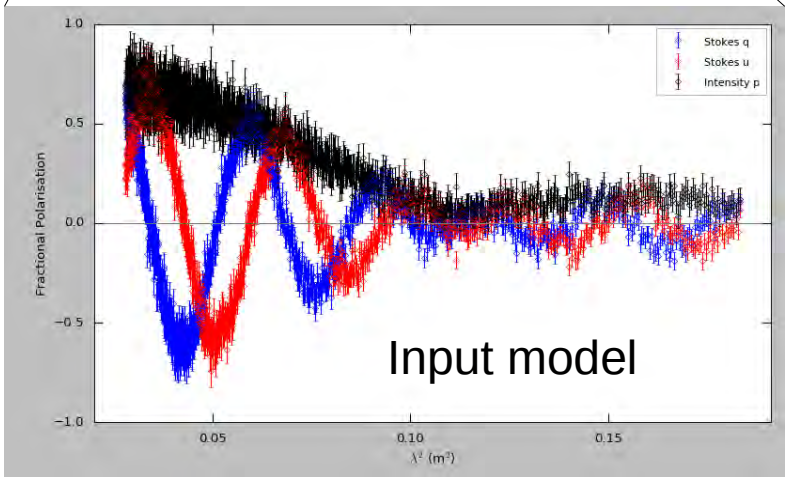
Pipeline Tasks:

0_gen_model_images.py




```
#-------------------------------------------------------------------------#
#                                                                         #
# Input catalogue file for use with the POSSUM pipeline.                  #
# Used to generate artificial data for testing purposes.                  #
#                                                                         #
# Note: spatial information (x,y,maj,min,pa) is ignored when generating 1D #
#       ASCII data. If generating image data these are the world-coordinates #
#       and parameters of the injected Gaussians.                         #
#                                                                         #
# C. Purcell 06-April-2016                                                #
#                                                                         #
#-------------------------------------------------------------------------#

# MODEL TYPE 1: External depolarised components within same beam (not LOS).
# Type, x_deg, y_deg, maj_asec, min_asec, pa_deg, fluxI_mJy, SI, [polFrac, evpa_deg, faradayDepth_radm2, far
#
# MODEL TYPE 2: Emitting & rotating regions stacked along the line-of-sight.
# Type, x_deg, y_deg, maj_asec, min_asec, pa_deg, fluxI_mJy, SI, [polFrac, evpa_deg, faradayDepth_radm2]xN
#

1, 89.9981, +0.0001, 0.0, 0.0, 0.0, 10.0, -0.0,      0.6, 180.0, 30.0, 0.0
1, 90.0016, +0.0023, 0.0, 0.0, 0.0, 0.5, -0.7,       0.4, 30.0, 100.0, 0.0,      0.3, 70.0, 80.0, 0.0
1, 90.0016, +0.0023, 0.0, 0.0, 0.0, 0.5, -0.7,       0.4, 30.0, 100.0, 30.0,      0.3, 70.0, 80.0, 30.0
2, 90.0026, +0.000, 0.0, 0.0, 0.0, 10.0, 0.0,        0.5, 45.0, -20.0
2, 90.0, +0.000, 0.0, 0.0, 0.0, 10.0, 0.0,           0.5, 45.0, -20.0,   0.25, 10, 50
2, 90.0, +0.000, 0.0, 0.0, 0.0, 10.0, 0.0,           0.4, 45.0, -30.0,   0.25, 10, 50,  0.25, 10, 90
```





Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

Pipeline supports **'sessions'** to facilitate different processing pathways on large datasets

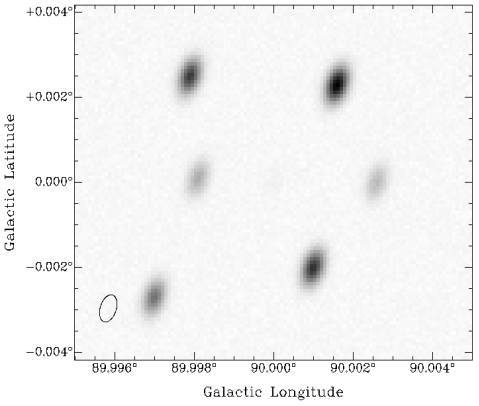
Source finding an external step: input can be catalogue or a FITS mask.

Session Directory

```
# ATLAS DR3 ELAIS Component Catalogue
#
# Sky coverage area : 2.697 sq. degrees (8.217 x 10^-4 sr)
#
# Columns:
#    1   ID          Component identification number
#    2   Name        Full catalogue name
#    3   RA          Right Ascension (degrees, J2000)
#    4   DEC         Declination (degrees, J2000)
#    5   RA_ERR      Error in right ascension (arcsec)
#    6   DEC_ERR     Error in declination (arcsec)
#    7   SNR         signal-to-noise ratio of raw dectection
#    8   RMS         local rms noise level (mJy/beam)
#    9   BWS         local bandwidth smearing value
#    10  Sp          Fitted source peak (mJy/beam)
#    11  Sp_ERR      Error in fitted source peak (mJy/beam)
#    12  S           Integrated flux density (mJy)
#    13  S_ERR       Error in integrated flux density (mJy)
#    14  V           Visibility area
#    15  OBS_FREQ    Frequency at which the peak and integrated flux was measured (MHz)
#    16  SINDEX      Spectral index of source between 1400 and 1710 MHz
#    17  INDEX_ERR   Error on spectral index
#
#     ID |                                Name |        RA |       DEC |    RA_ERR |   DEC_ERR |       SNR |       RMS |
#        |                                     |   degrees |   degrees |    arcsec |    arcsec |           | mJy/bm    |
#      1 |                                   2 |         3 |         4 |         5 |         6 |         7 |         8 |
   EI0001  ATLASDR3_J003616.720-430934.734_I   9.069665 -43.159648   0.018000   0.025000   8125.00      0.019
 EI0002C1  ATLASDR3_J003128.847-434120.170_I   7.870197 -43.688936   0.003350   0.001960   6089.09      0.022
 EI0002C2  ATLASDR3_J003128.646-434122.322_I   7.869357 -43.689534   0.015020   0.012620   1784.09      0.022
 EI0002C3  ATLASDR3_J003128.493-434136.107_I   7.868720 -43.693363   0.019700   0.015800    754.32      0.022
   EI0003  ATLASDR3_J003309.271-435926.845_I   8.288628 -43.990790   0.018000   0.025000   3862.00      0.017
 EI0004C1  ATLASDR3_J003442.890-433034.495_I   8.678708 -43.509582   0.012600   0.012080   3192.67      0.015
 EI0004C2  ATLASDR3_J003442.526-433030.010_I   8.677192 -43.508336   0.202000   0.150000    176.00      0.015
 EI0004C3  ATLASDR3_J003441.426-433043.236_I   8.672608 -43.512010   0.200000   0.309000     25.27      0.015
   EI0005  ATLASDR3_J003249.319-442150.581_I   8.205496 -44.364050   0.018000   0.026000   1869.00      0.020
   EI0006  ATLASDR3_J003723.328-442245.055_I   9.347198 -44.379182   0.018000   0.026000   1807.00      0.024
```

Source finder catalogue or FITS mask

Flexible catalogue ingestion via SQL CREATE



Link

Workflow

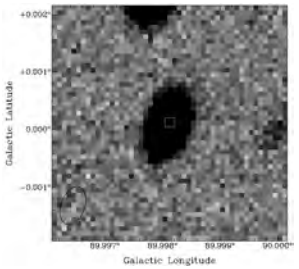
Pipeline Tasks:

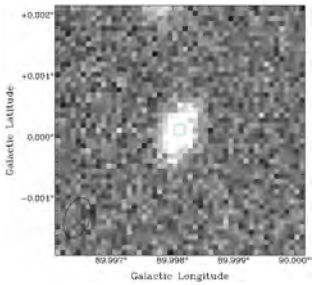
0_gen_model_images.py

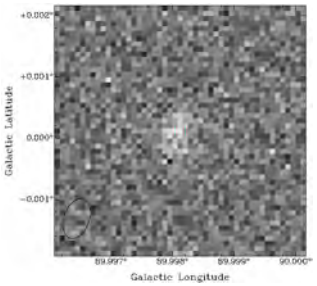
1_verify_image_data.py

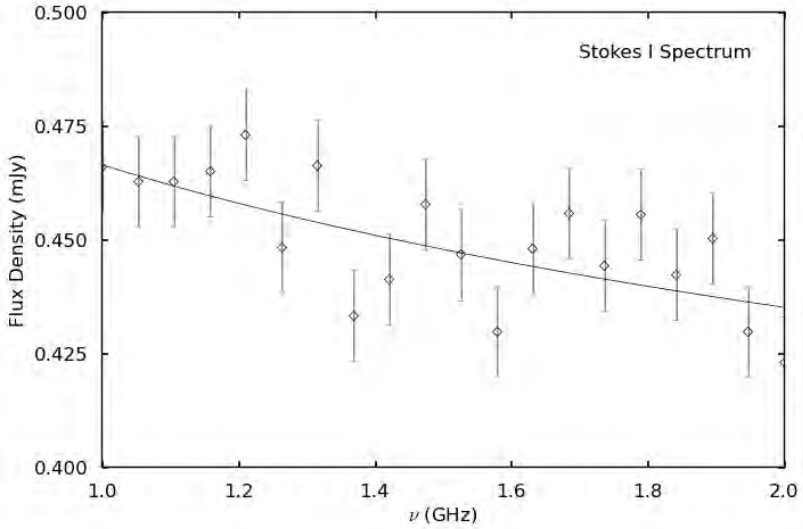
2_create_image_session.py

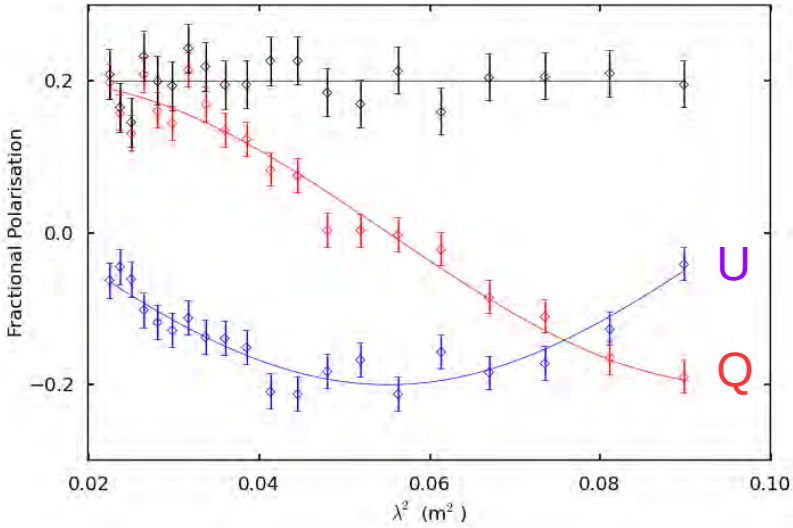
3_extract_spectra.py











Robust spectral extraction, noise estimation & postage-stamp 'cublet' creation

1D & 3D versions

Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

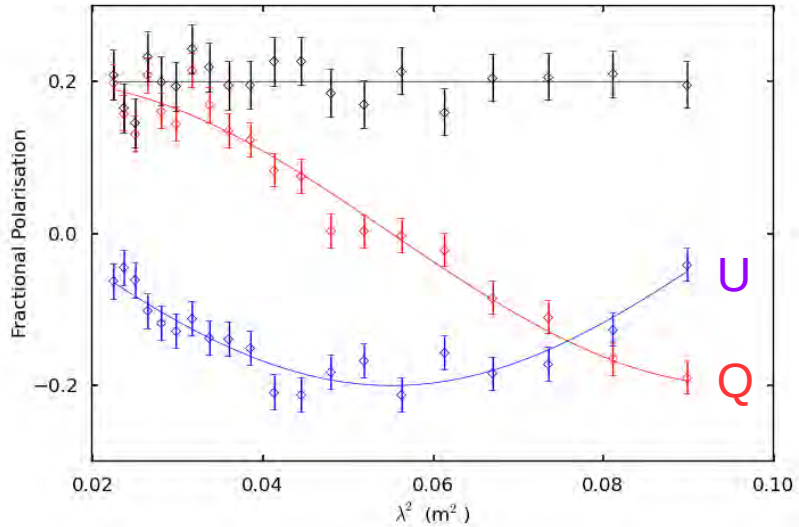
2_create_image_session.py

3_extract_spectra.py

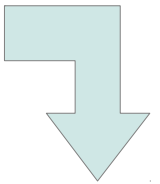
4_do_RM-synthesis.py

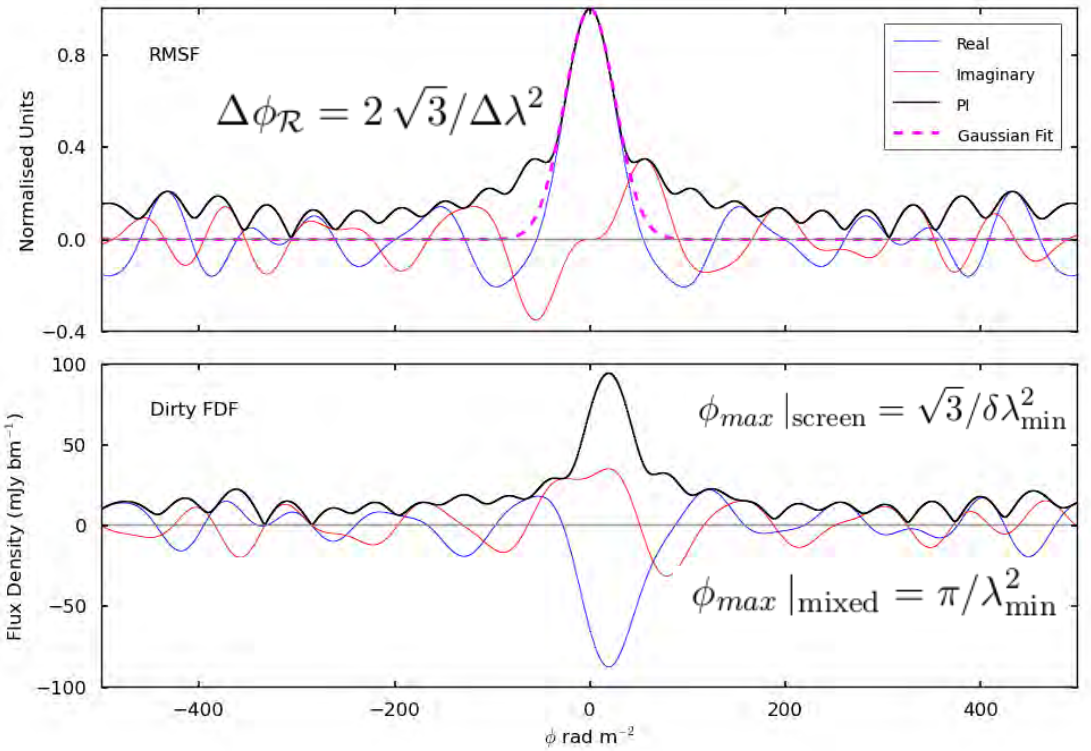
Operate on Fourier transformed data

Wavelength to Faraday Depth



Fourier Transform





Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

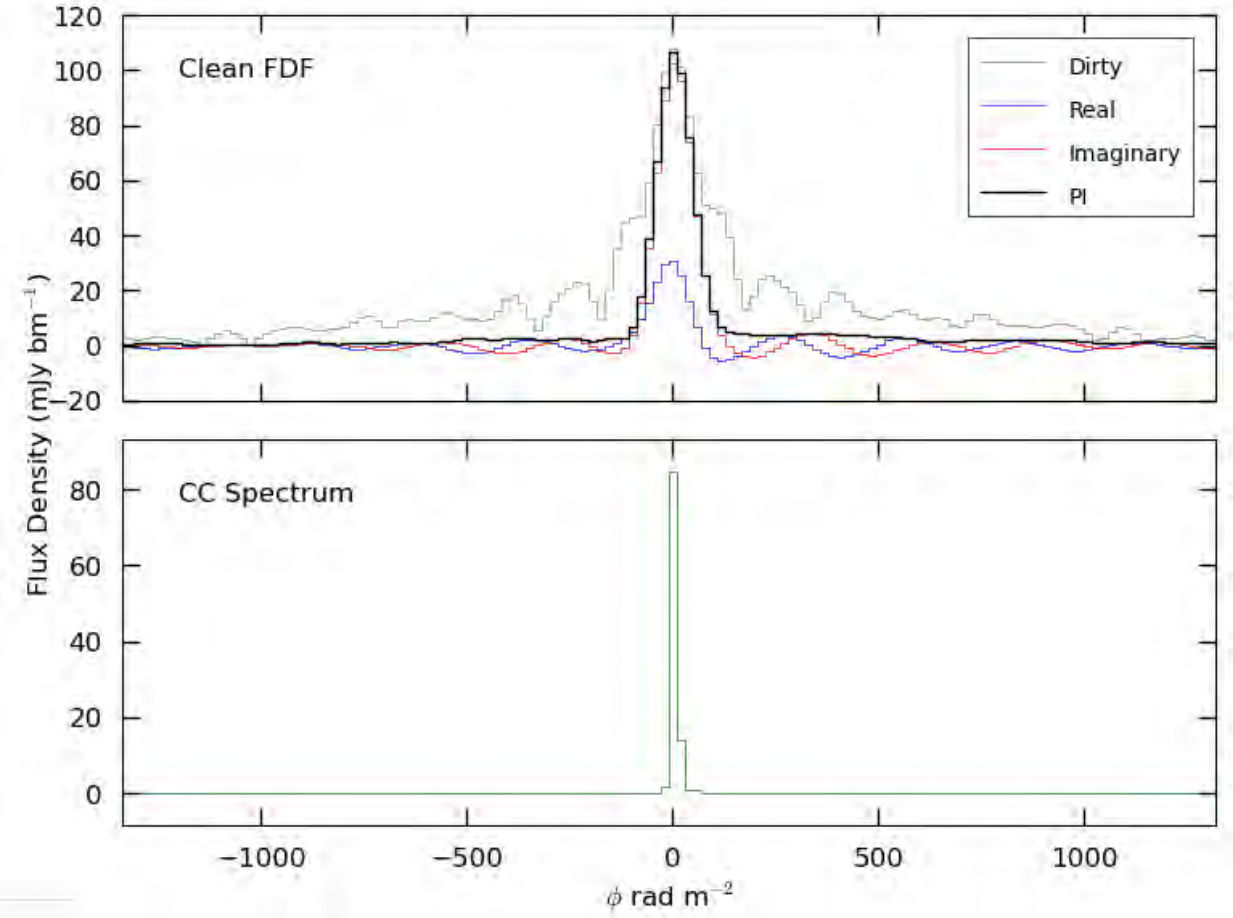
2_create_image_session.py

3_extract_spectra.py

4_do_RM-synthesis.py

5_do_RM-clean.py

Deconvolve FDF



Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

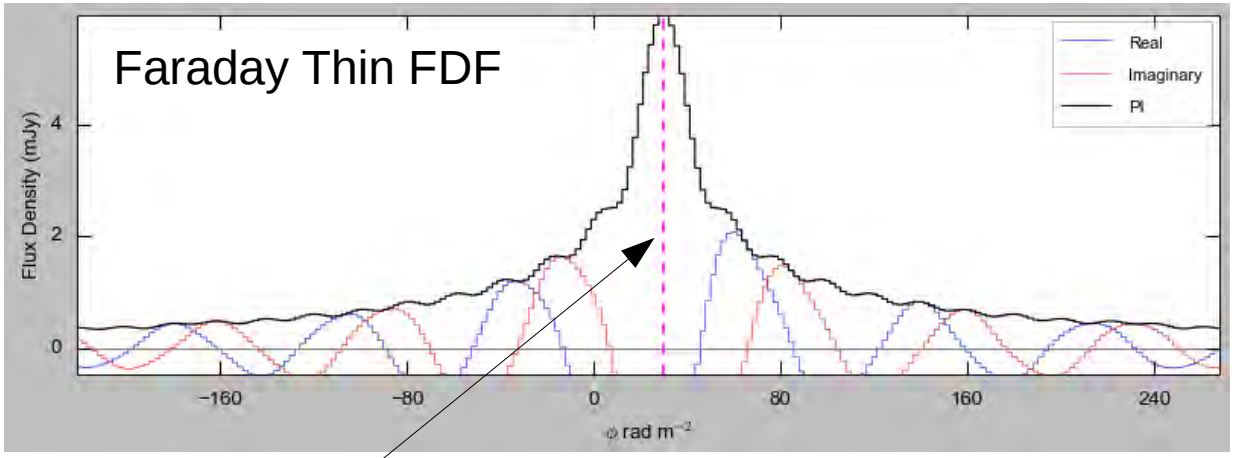
2_create_image_session.py

3_extract_spectra.py

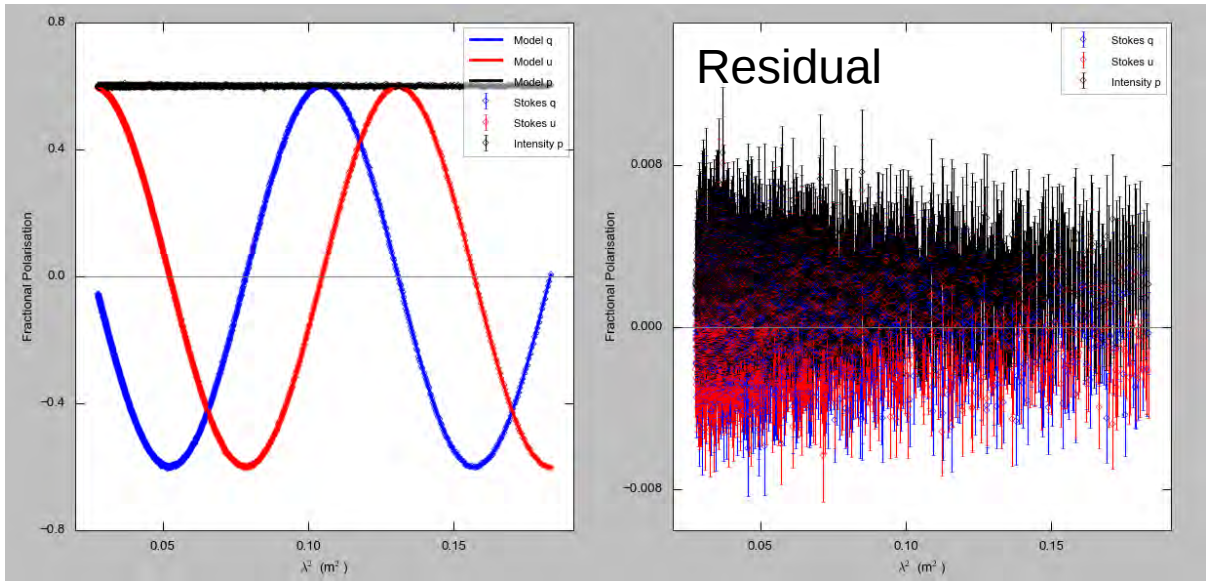
4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py



Detect peak, subtract thin model & measure residual



Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

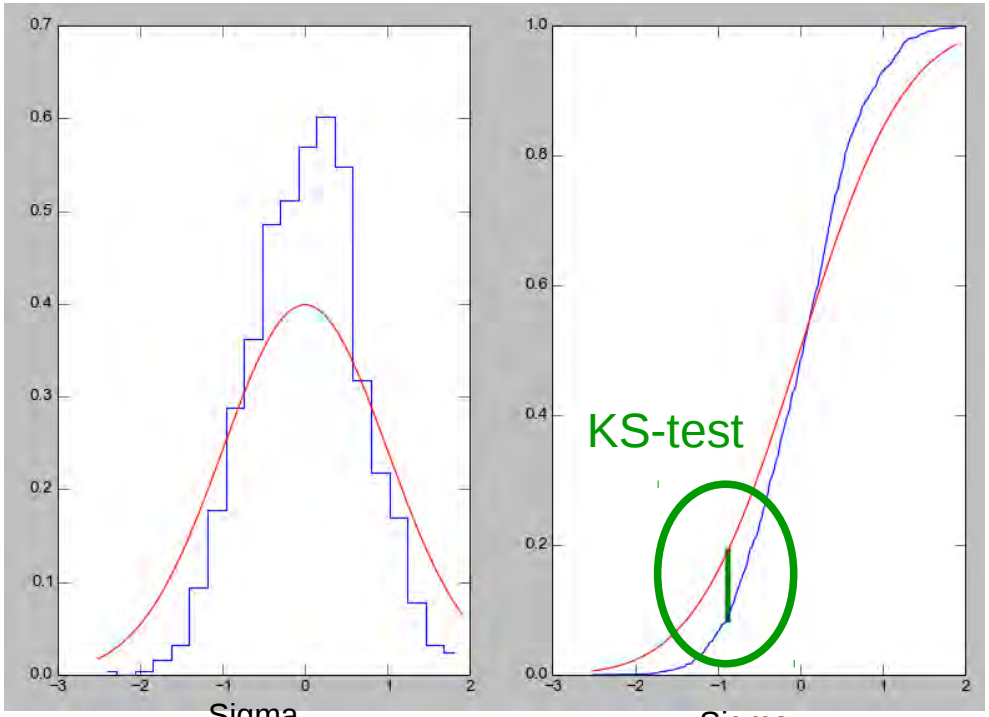
3_extract_spectra.py

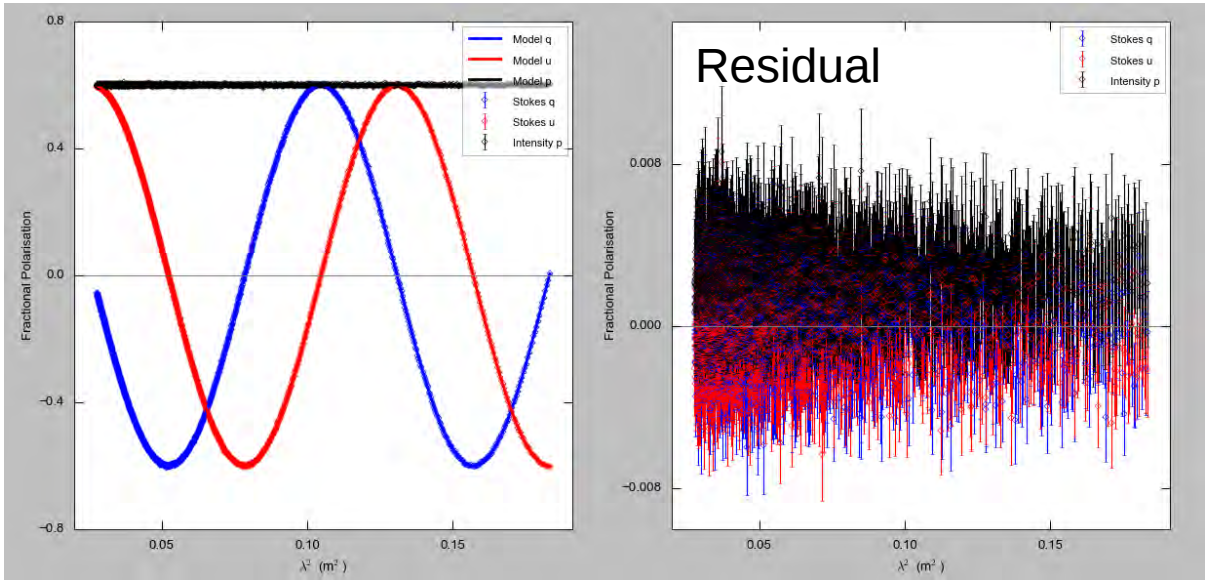
4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py

- Skewness
- KS-test
- Anderson-Darling test





Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

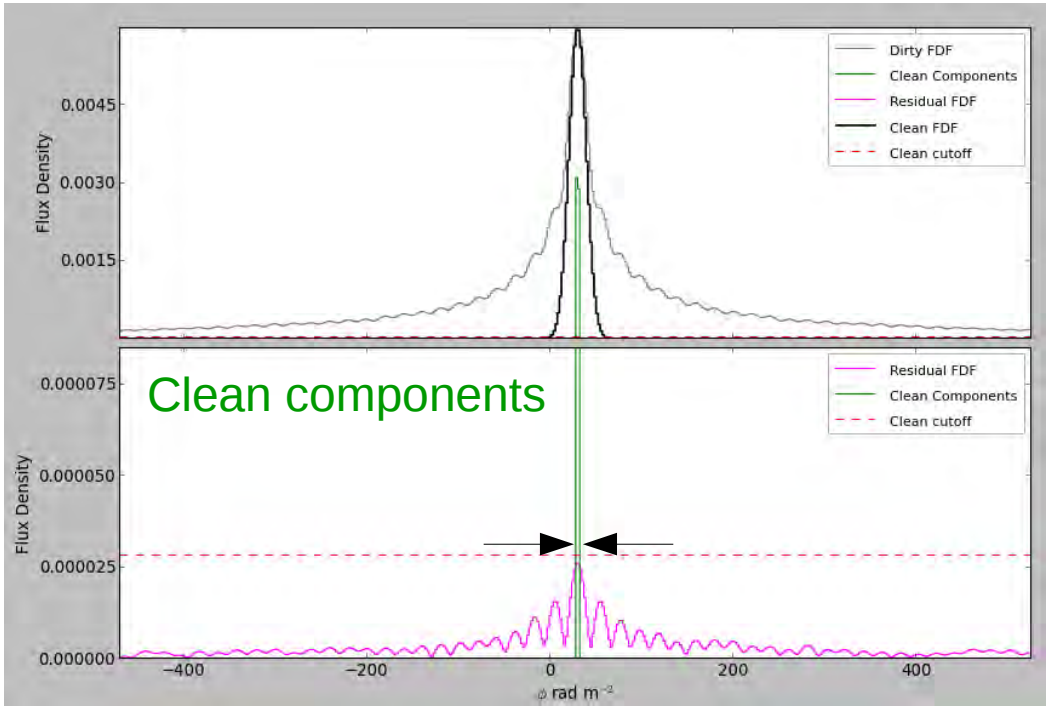
3_extract_spectra.py

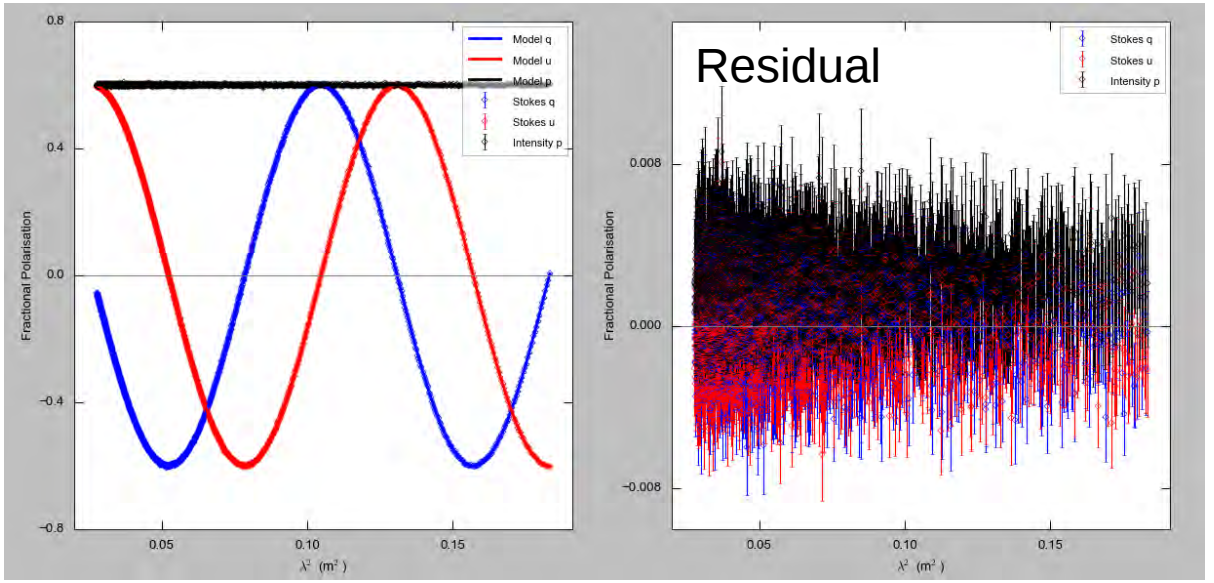
4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py

- 2nd moment of Clean component spectrum





Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

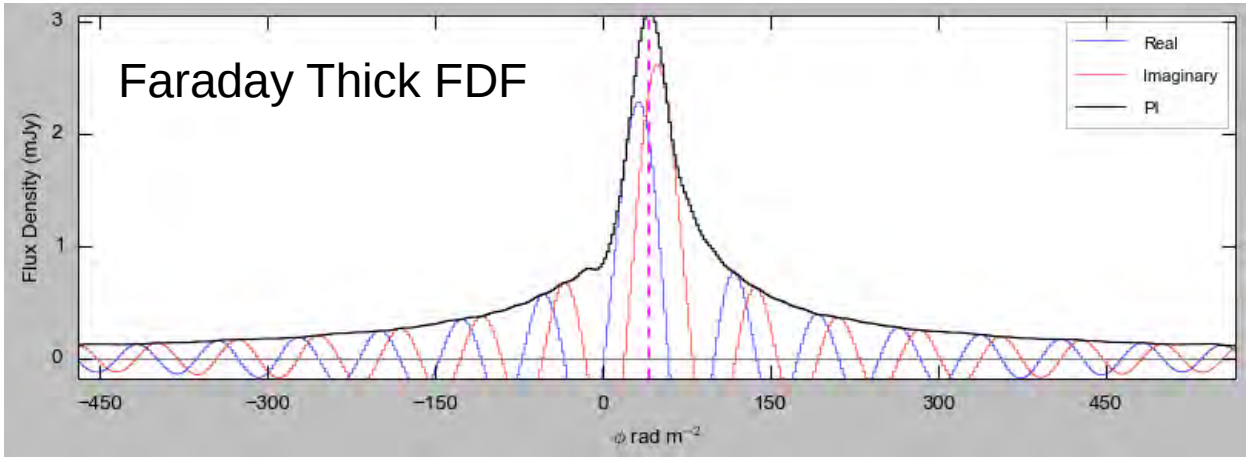
2_create_image_session.py

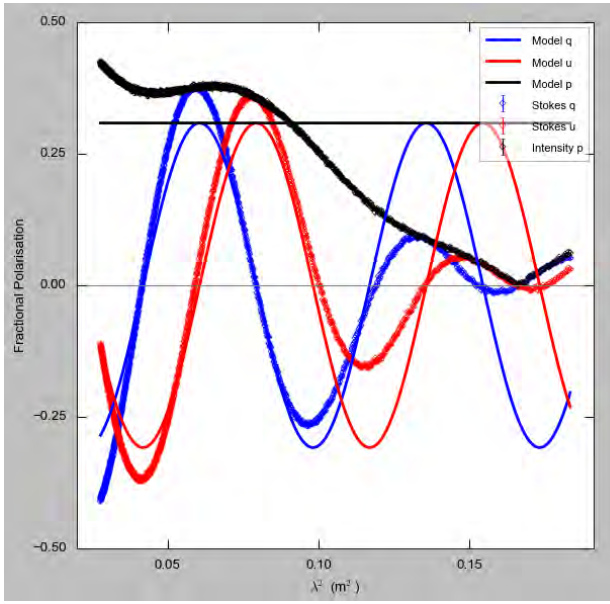
3_extract_spectra.py

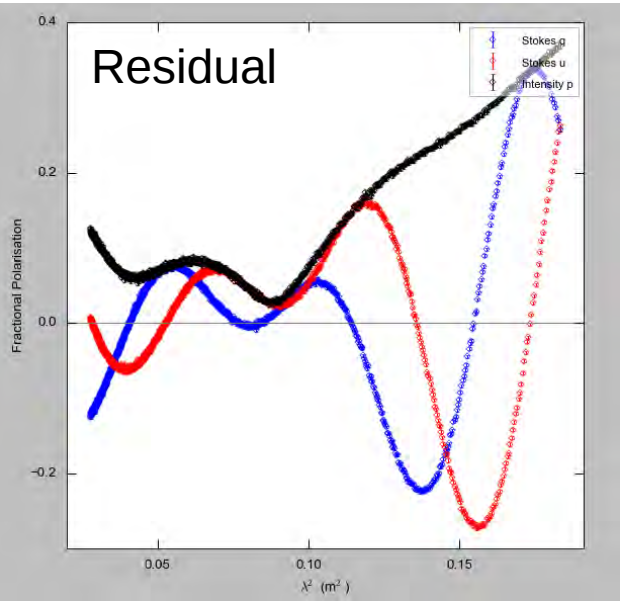
4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py







Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

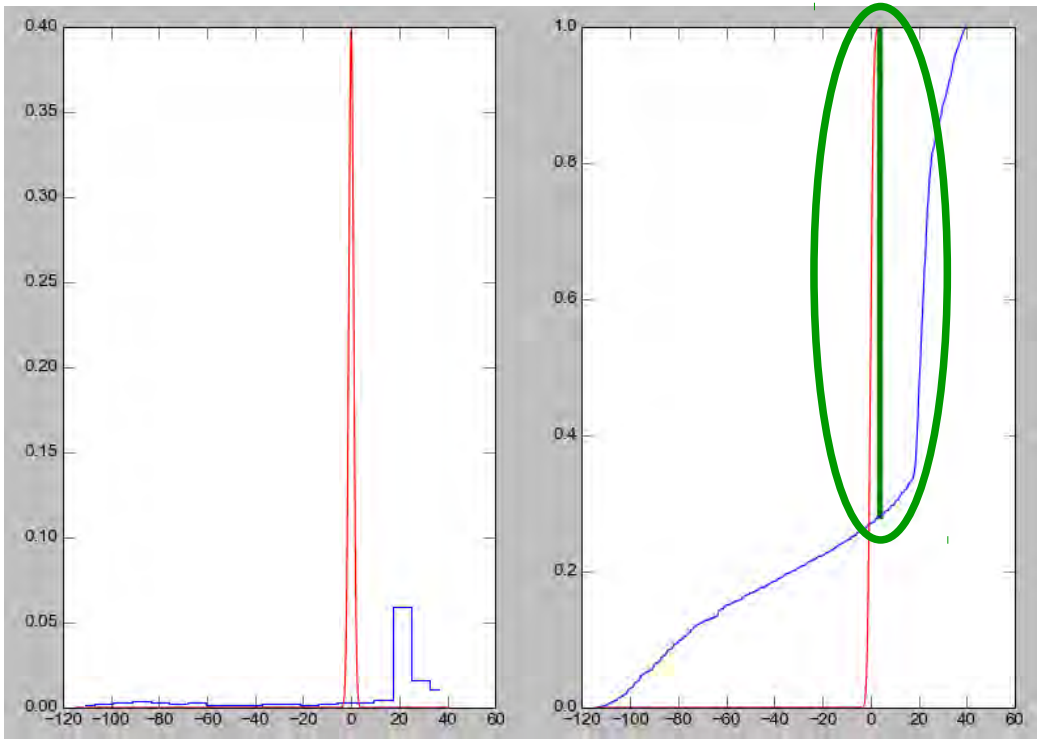
2_create_image_session.py

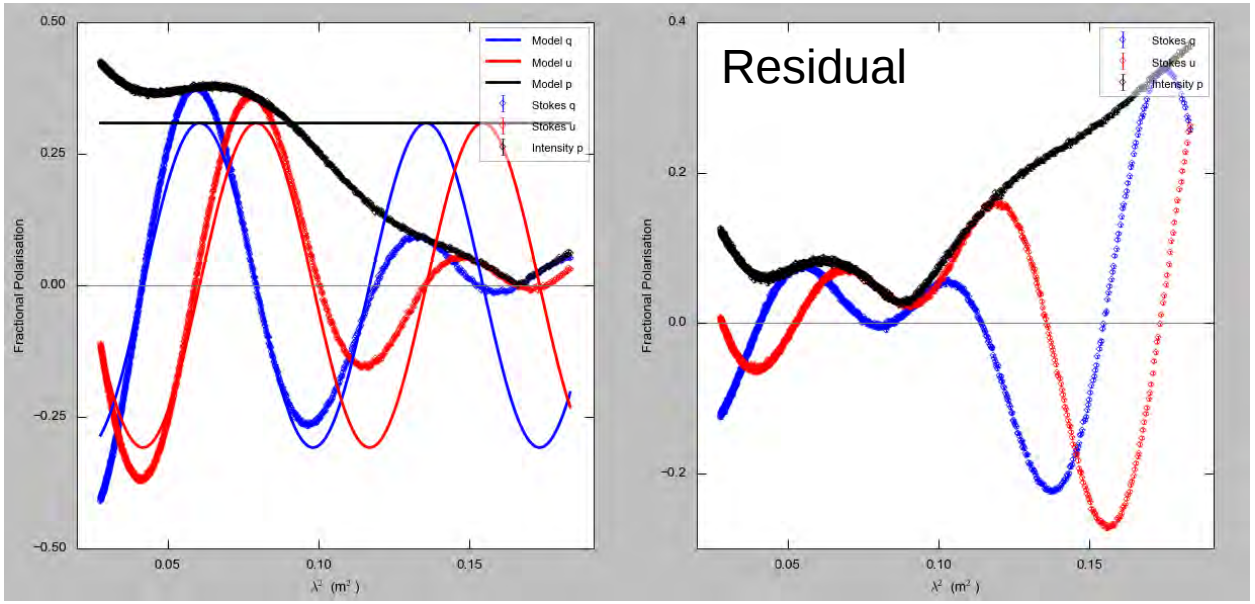
3_extract_spectra.py

4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py





Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

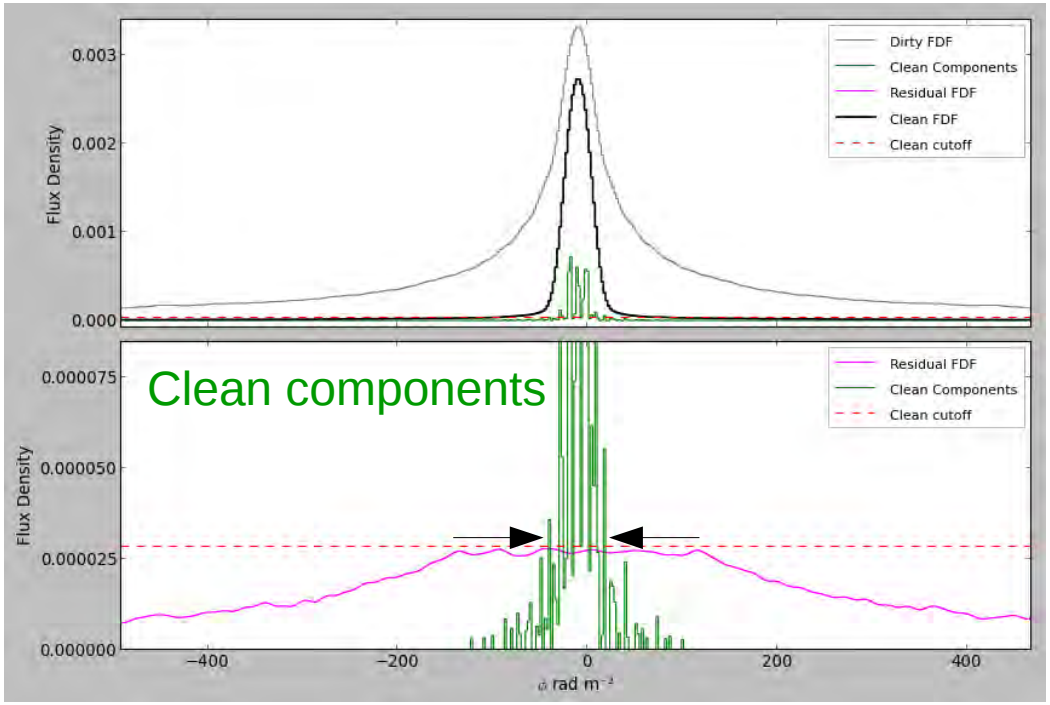
2_create_image_session.py

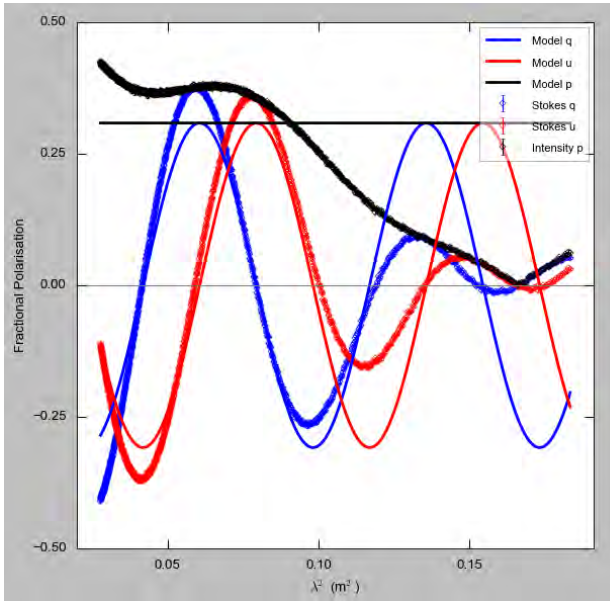
3_extract_spectra.py

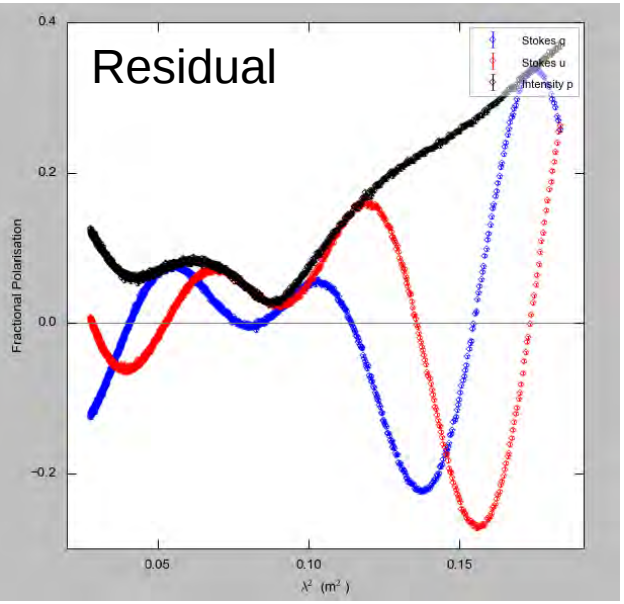
4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py







Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

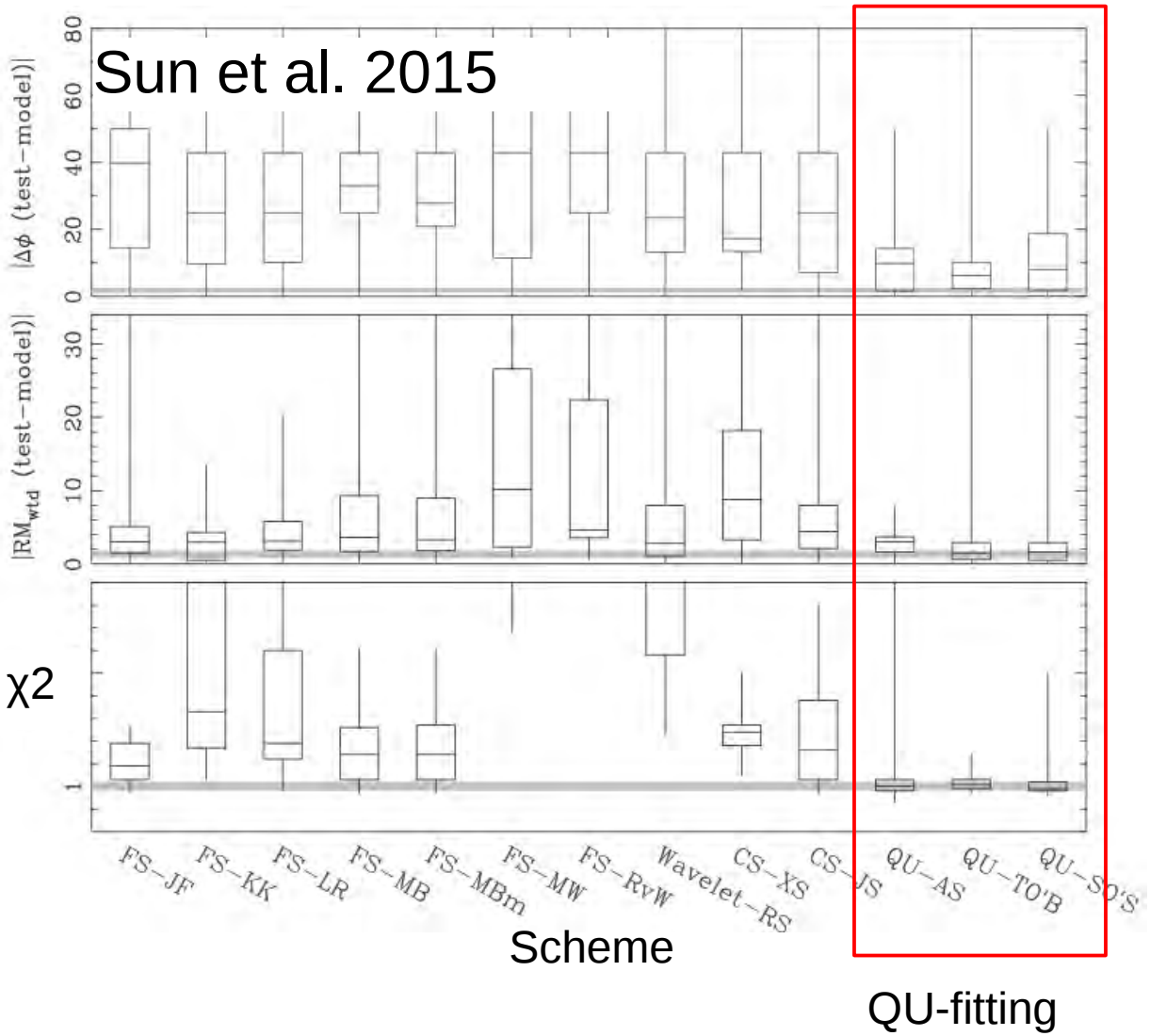
3_extract_spectra.py

4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py

7_do_QUfit_MCMC.py



Workflow

- One rotating screen + foreground depolarisation:

$$P = p_0 e^{-2\sigma_{RM}^2 \lambda^4} e^{2i(\Psi_0 + RM\lambda^2)}$$

Depolarisation Rotation

- Two rotating screens within the beam:

$$P = p_0 e^{2i(\Psi_0 + RM_1 \lambda^2)} + p_0 e^{2i(\Psi_0 + RM_2 \lambda^2)}$$

Rotation 1 Rotation 2

- *N* screens with RM-gradients:

$$\mathcal{P}(\lambda^2) = p_0 \sum_{i=1}^{N} \frac{I_i}{I} \frac{\sin(\phi_i \lambda^2)}{\phi_i \lambda^2} \exp\left(2i(\psi_{o,i} + \frac{1}{2}\phi_i \lambda^2 + \sum_{j=i+1}^{N} \phi_j \lambda^2)\right)$$

Sokoloff et al. 1998

Workflow

- Current work focused on testing robust code to fit models to fractional polarisation data.
- Convergence can be a problem:

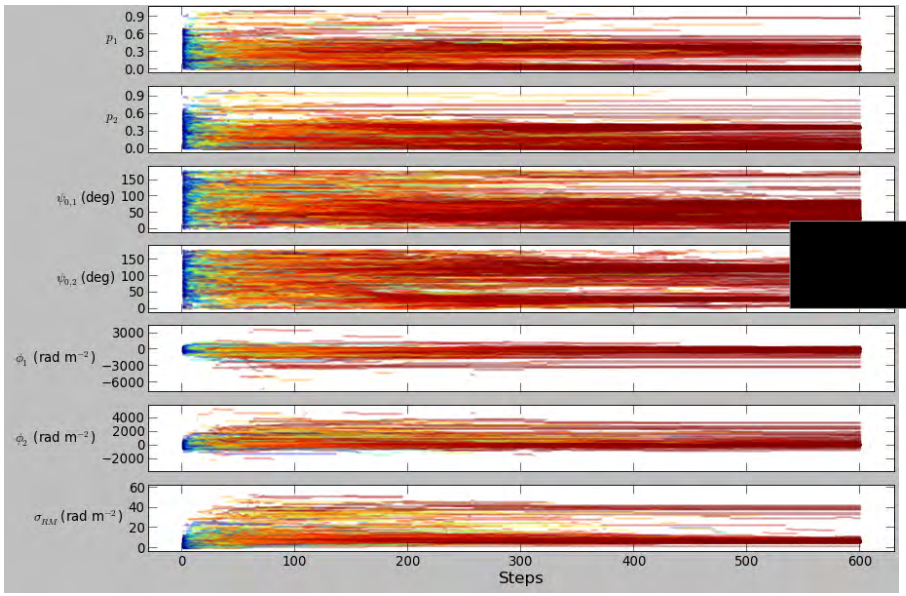
Foreman-Mackey et al. 2013

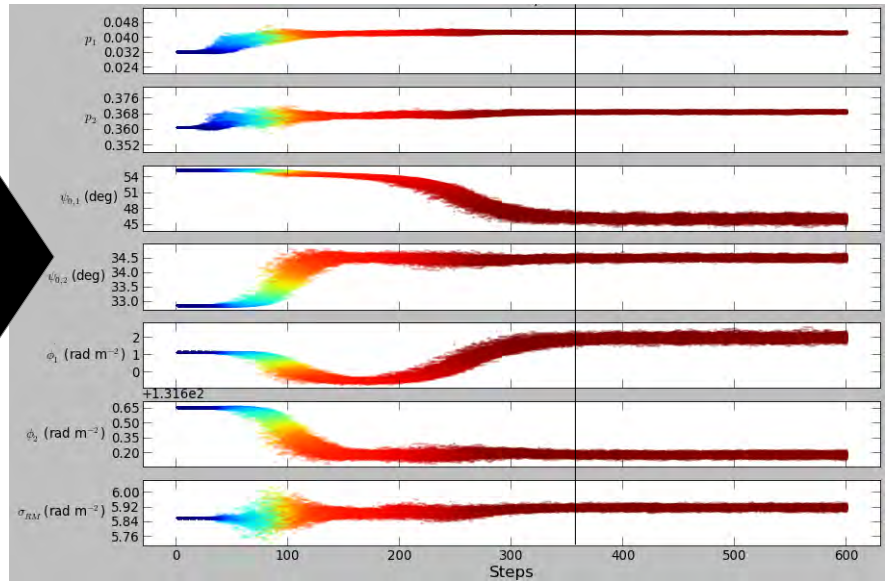
Robust MCMC fitting code (Purcell et al. in prep)

Exploration

Burn-in

Convergence



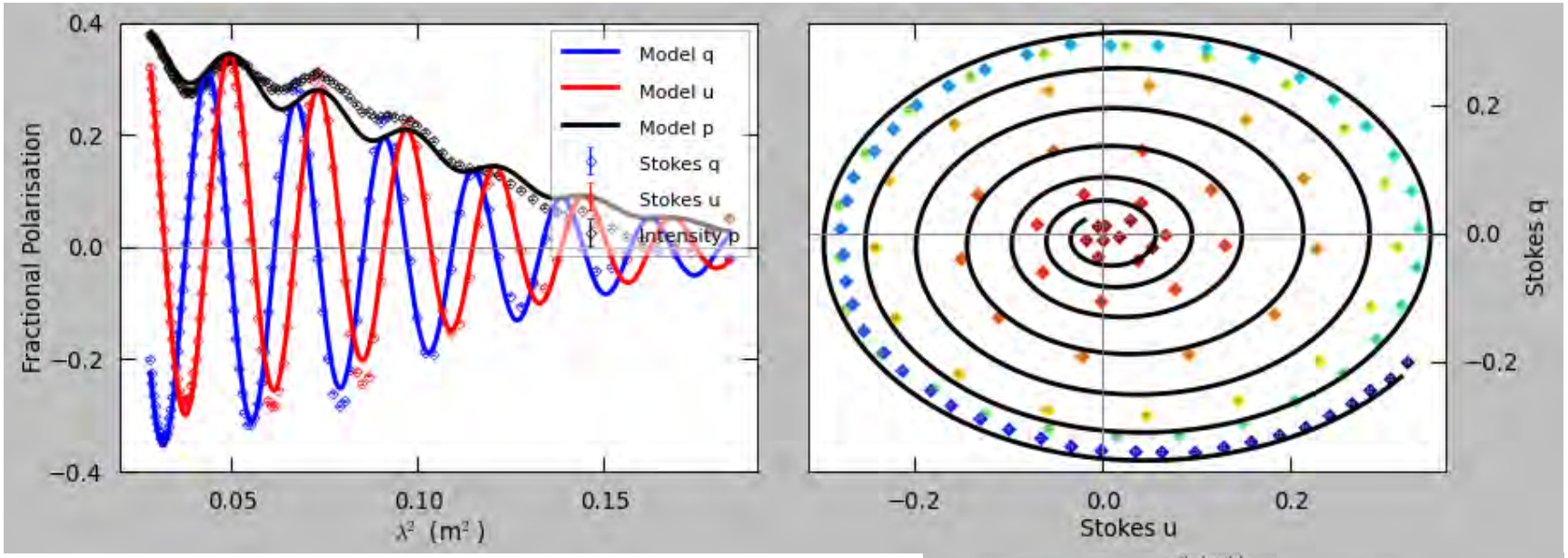


Workflow

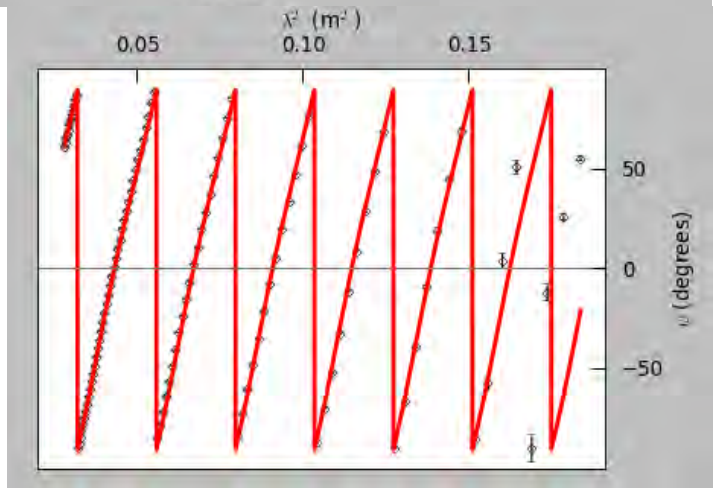
$$\mathcal{P}(\lambda^2) = p_0 \sum_{i=1}^{N} \frac{I_i}{I} \frac{\sin(\phi_i \lambda^2)}{\phi_i \lambda^2} \exp\left(2i(\psi_{o,i} + \frac{1}{2}\phi_i\lambda^2 + \sum_{j=i+1}^{N} \phi_j \lambda^2)\right)$$

N RM-gradient screens

Sokoloff et al. 1998



- Scary: 2-3 component models can fit almost any observed spectrum!

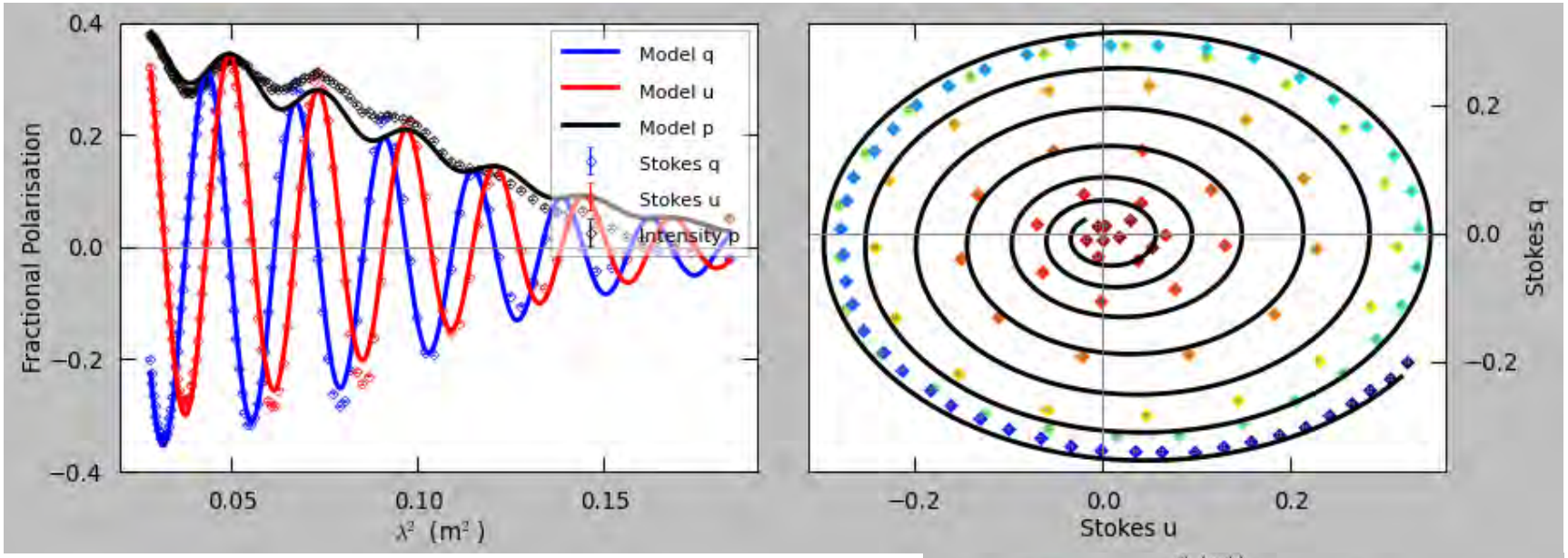


Workflow

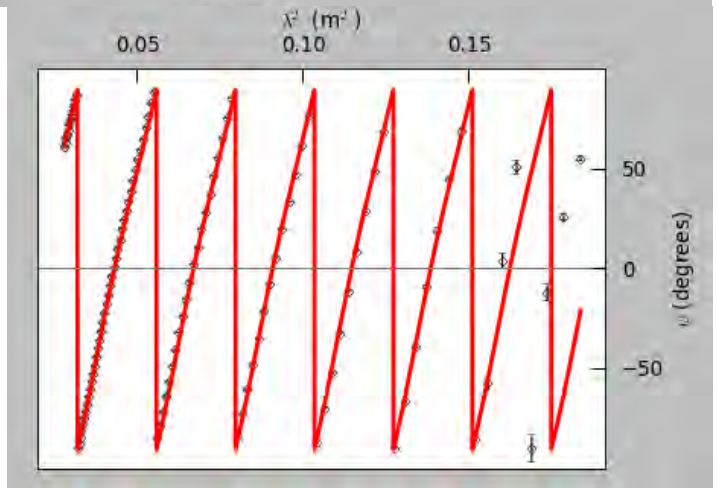
$$\mathcal{P}(\lambda^2) = p_0 \sum_{i=1}^{N} \frac{I_i}{I} \frac{\sin(\phi_i \lambda^2)}{\phi_i \lambda^2} \exp\left(2i(\psi_{o,i} + \frac{1}{2}\phi_i\lambda^2 + \sum_{j=i+1}^{N} \phi_j \lambda^2)\right)$$

N RM-gradient screens

Sokoloff et al. 1998



- Scary: 2-3 component models can fit almost any observed spectrum!
- **Very high S/N needed across broad band to dissentangle**



Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

3_extract_spectra.py

4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py

7_do_QUfit_MCMC.py

rmPipeViewer.py

Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

3_extract_spectra.py

4_do_RM-synthesis.py

5_do_RM-clean.py

6_assess_complexity.py

7_do_QUfit_MCMC.py

rmPipeViewer.py

Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

3_extract_spectra.py

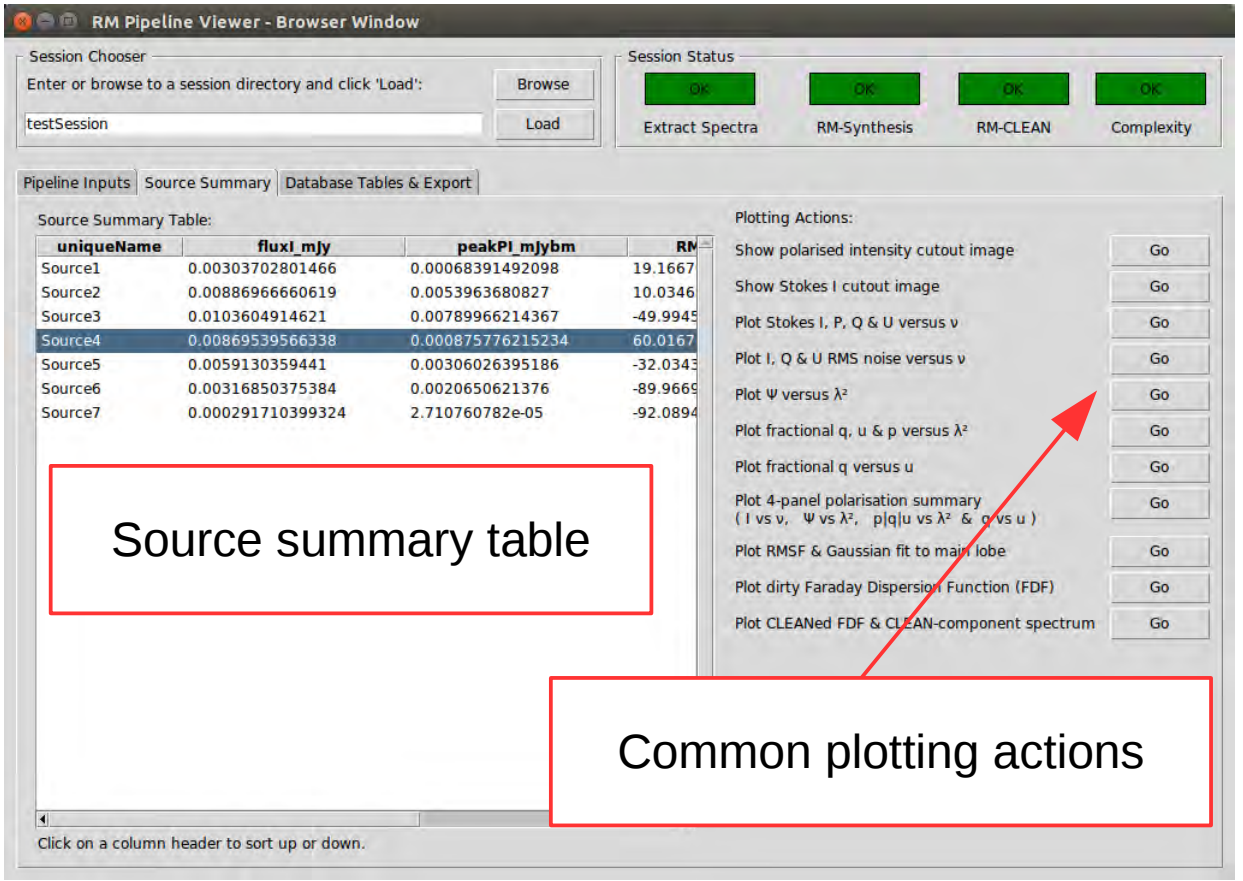
4_do_RM-synthesis.py

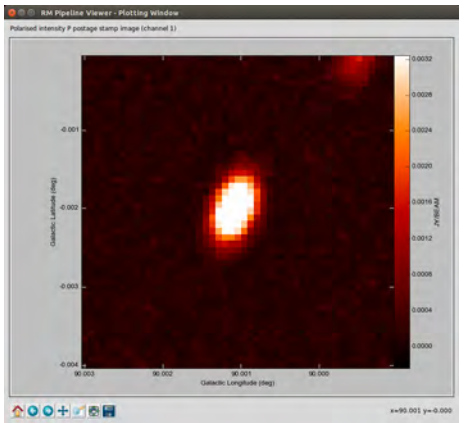
5_do_RM-clean.py

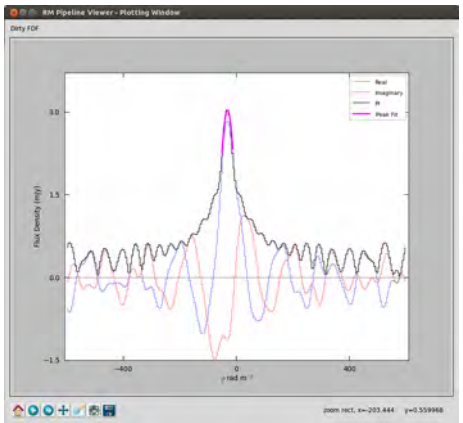
6_assess_complexity.py

7_do_QUfit_MCMC.py

rmPipeViewer.py







Workflow

Pipeline Tasks:

0_gen_model_images.py

1_verify_image_data.py

2_create_image_session.py

3_extract_spectra.py

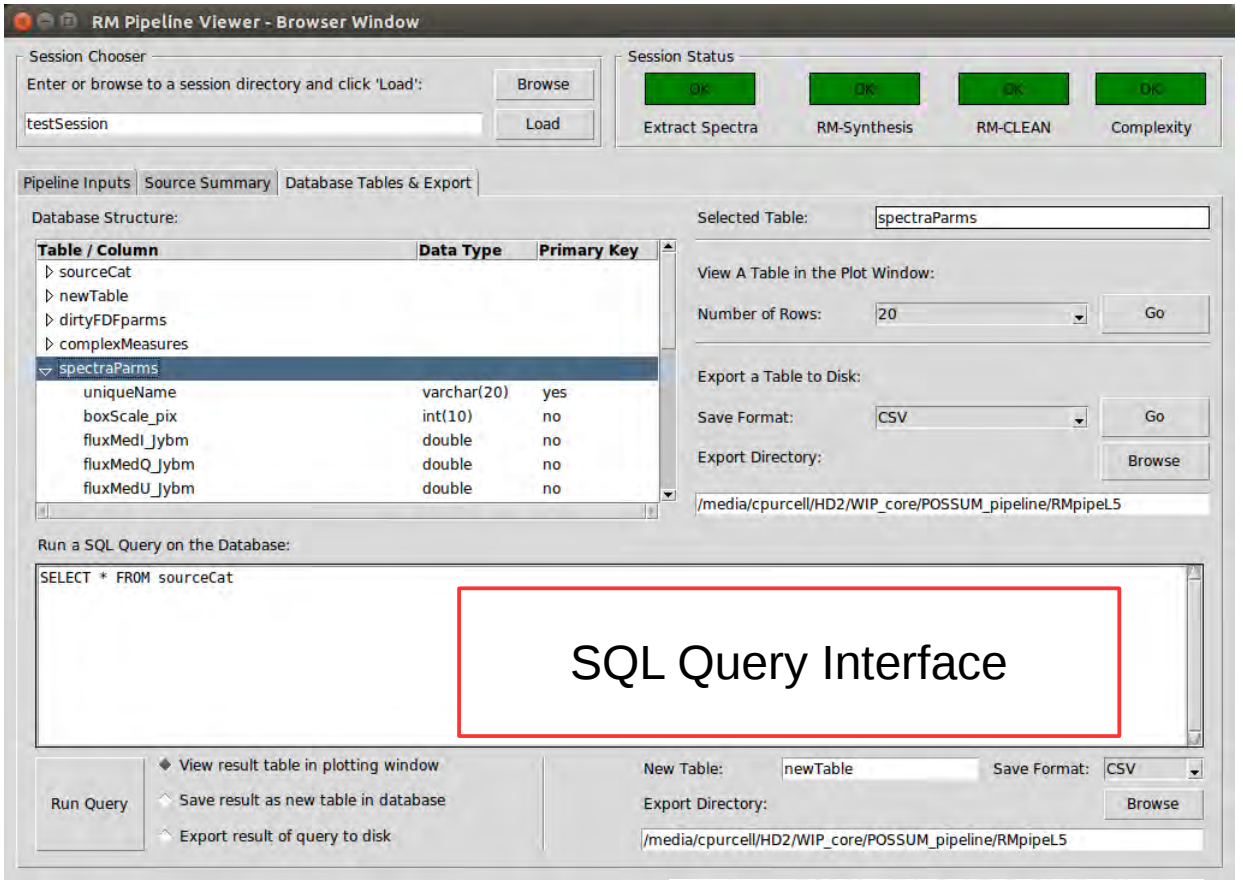
4_do_RM-synthesis.py

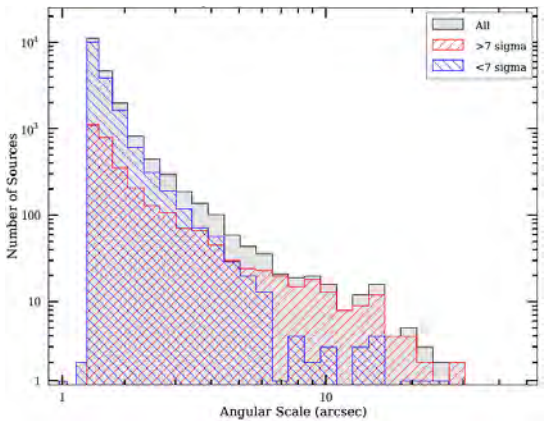
5_do_RM-clean.py

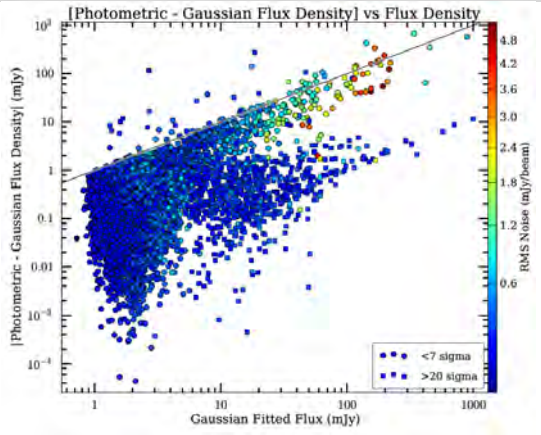
6_assess_complexity.py

7_do_QUfit_MCMC.py

rmPipeViewer.py







Summary

- Goal: Build a prototype pipeline for the POSSUM project
  - Easy to deploy and use
  - Well commented & documented for CSIRO
  - Framework for testing and integrating algorithms
  - Robust enough to use on existing datasets.

Summary

- Goal: Build a prototype pipeline for the POSSUM project
  - Easy to deploy and use ✓
  - Well commented & documented for CSIRO
  - Framework for testing and integrating algorithms
  - Robust enough to use on existing datasets.

Summary

- Goal: Build a prototype pipeline for the POSSUM project
  - Easy to deploy and use ✓
  - Well commented & documented for CSIRO ✓
  - Framework for testing and integrating algorithms
  - Robust enough to use on existing datasets.

Summary

- Goal: Build a prototype pipeline for the POSSUM project
  - Easy to deploy and use ✓
  - Well commented & documented for CSIRO ✓
  - Framework for testing and integrating algorithms ✓
  - Robust enough to use on existing datasets.

Summary

- Goal: Build a prototype pipeline for the POSSUM project
  - Easy to deploy and use ✓
  - Well commented & documented for CSIRO ✓
  - Framework for testing and integrating algorithms ✓
  - Robust enough to use on existing datasets. ✓

Summary

- Goal: Build a prototype pipeline for the POSSUM project
  - Easy to deploy and use ✓
  - Well commented & documented for CSIRO ✓
  - Framework for testing and integrating algorithms ✓
  - Robust enough to use on existing datasets. ✓

- **Code release in two tranches:**
- 1D & 3D RM-synthesis, RM-clean & QU-fitting modules + demonstration code
  - Will be released on GitHub soon (announcement to POSSUM list):
  - **https://github.com/crpurcell**
- POSSUM pipeline code to make specific PPC and PBCat catalogues
  - Contact **cormac.purcell@sydney.edu.au**