LSDB-Macauff: The Trials and Tribulations of Catalogue Cross-Matching in the Era of LSST

Tom J Wilson (he/him)
t.j.wilson@exeter.ac.uk

Melissa DeLucchi, Sandro Campos, Sean McGuire, Max West,
Kostya Malanchev, Sam Wyatt, Jeremy Kubica

LINCC TECH TALK - 14/MAR/24

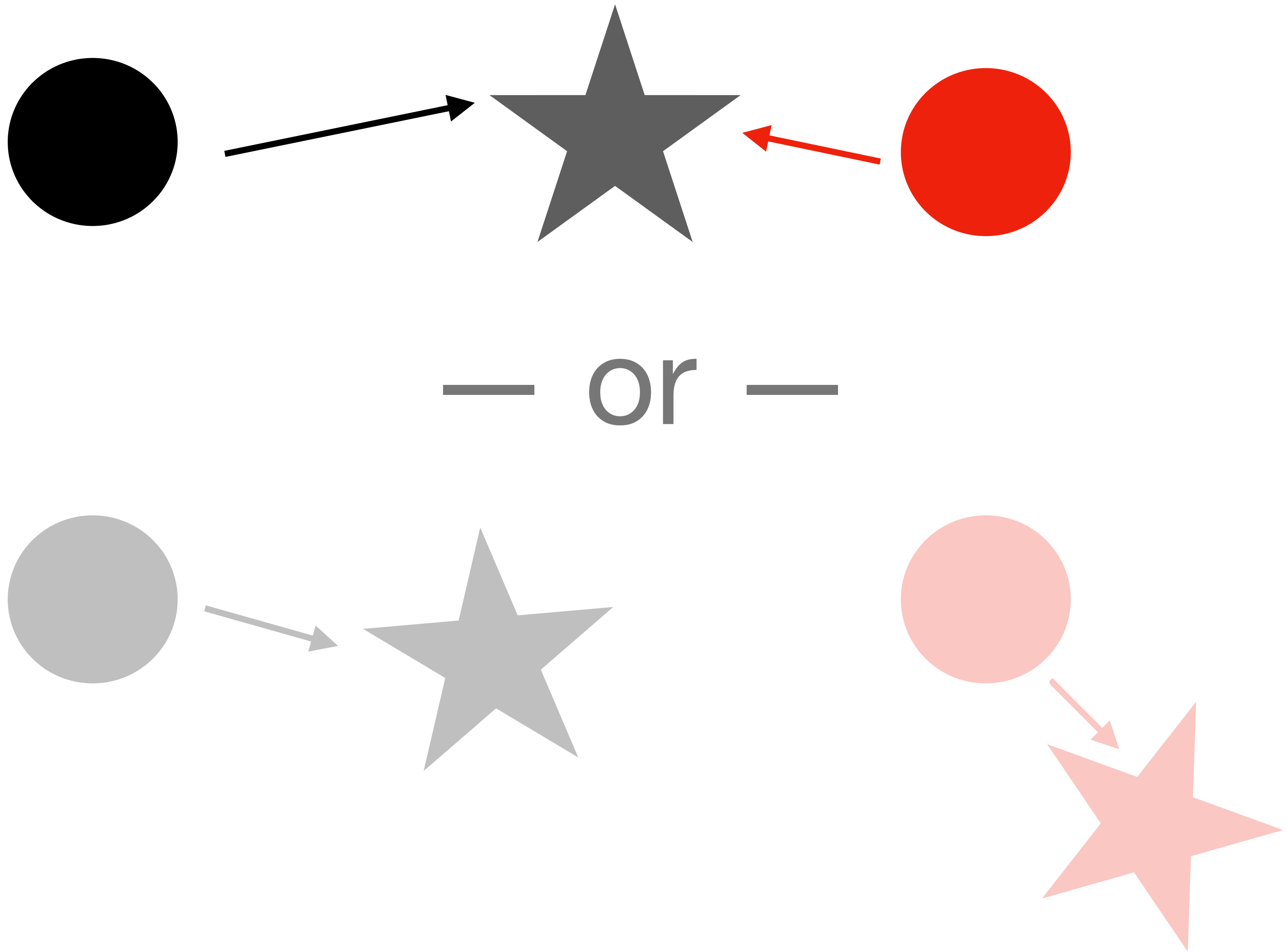
@Onoddil@pm.me
@Onoddil.github.io

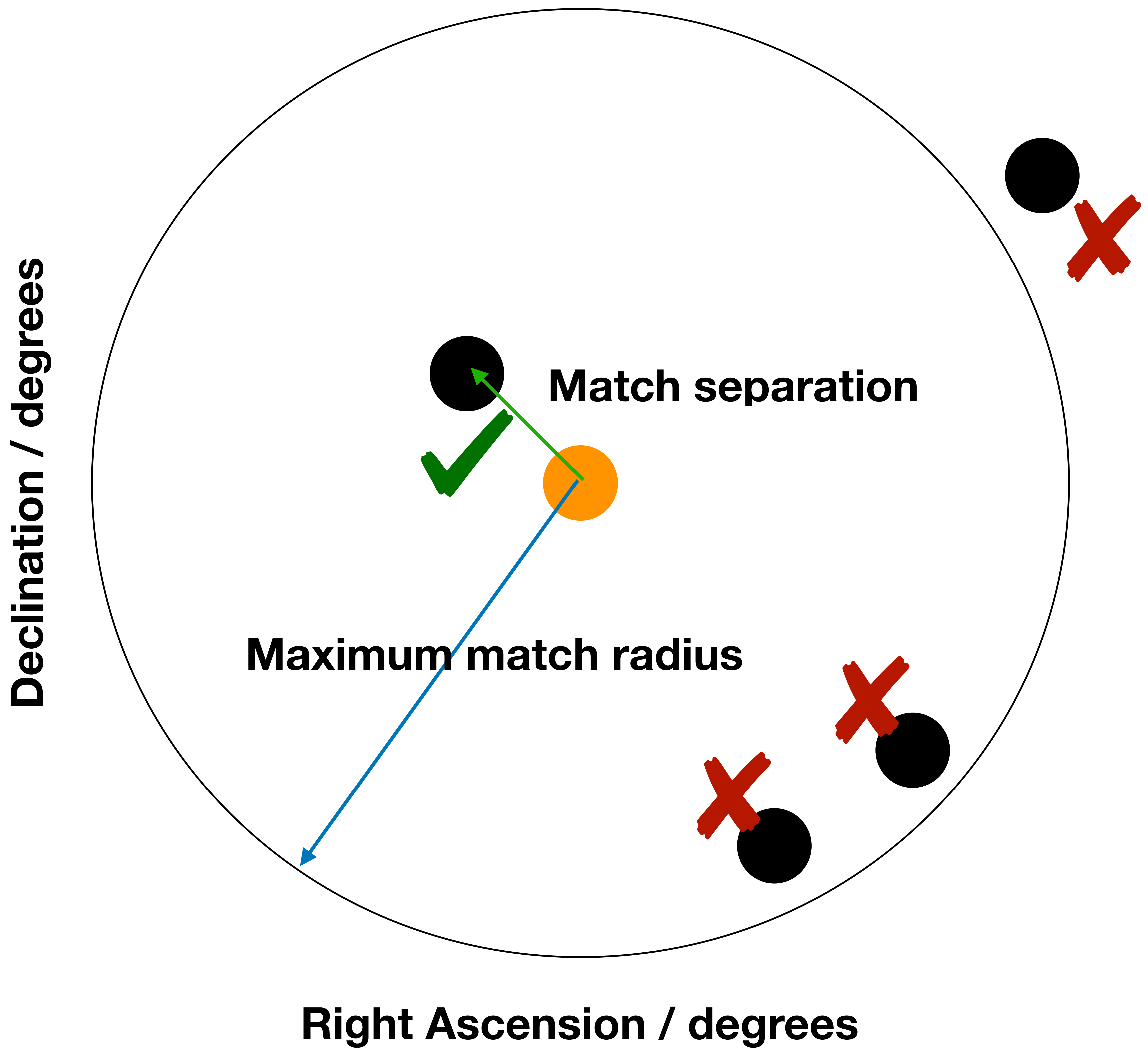
Macauff

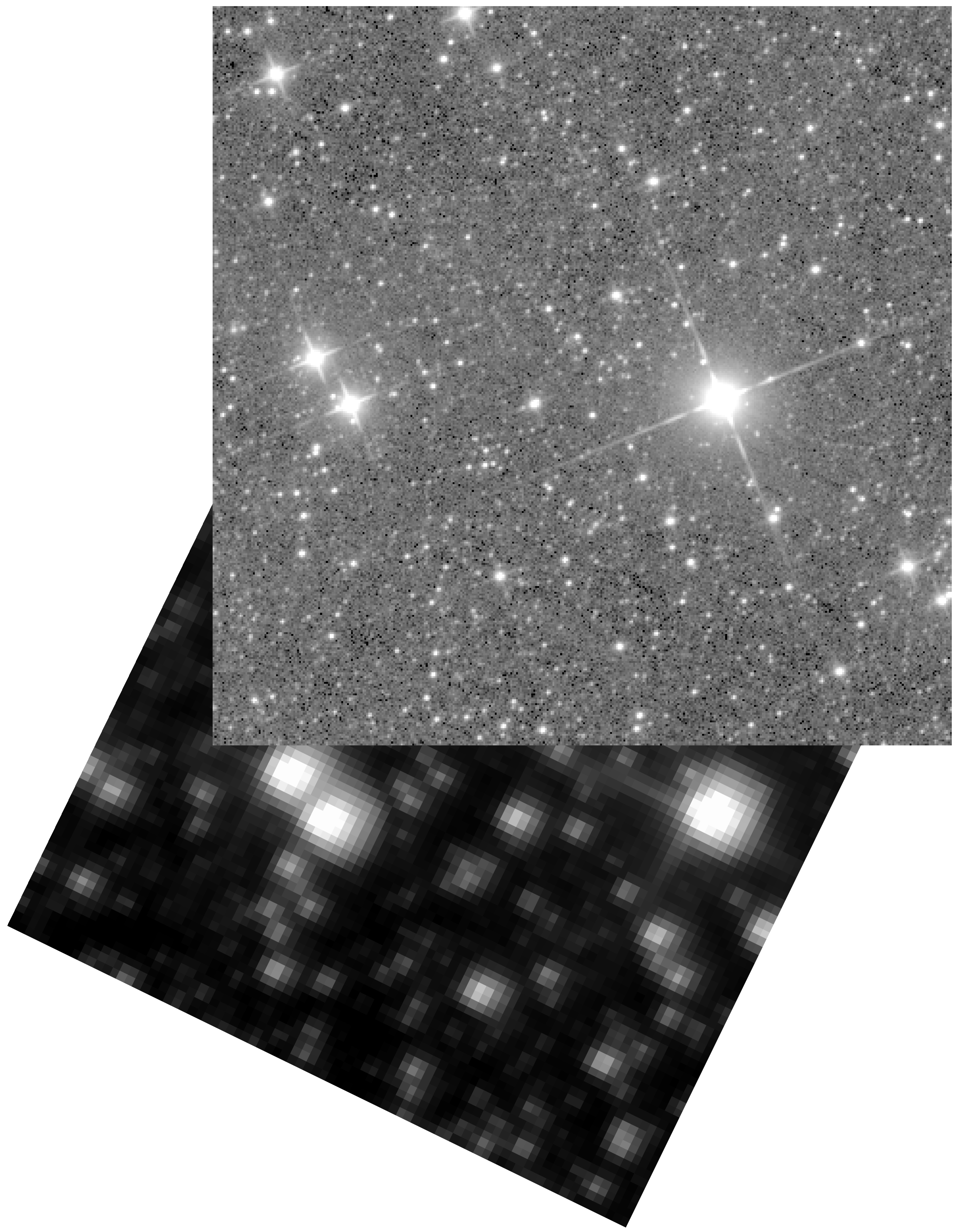
Or, Catalogue Cross-Matching in the Crowded LSST Sky

Tom J Wilson @onoddil

“Simple” Cross-Matching

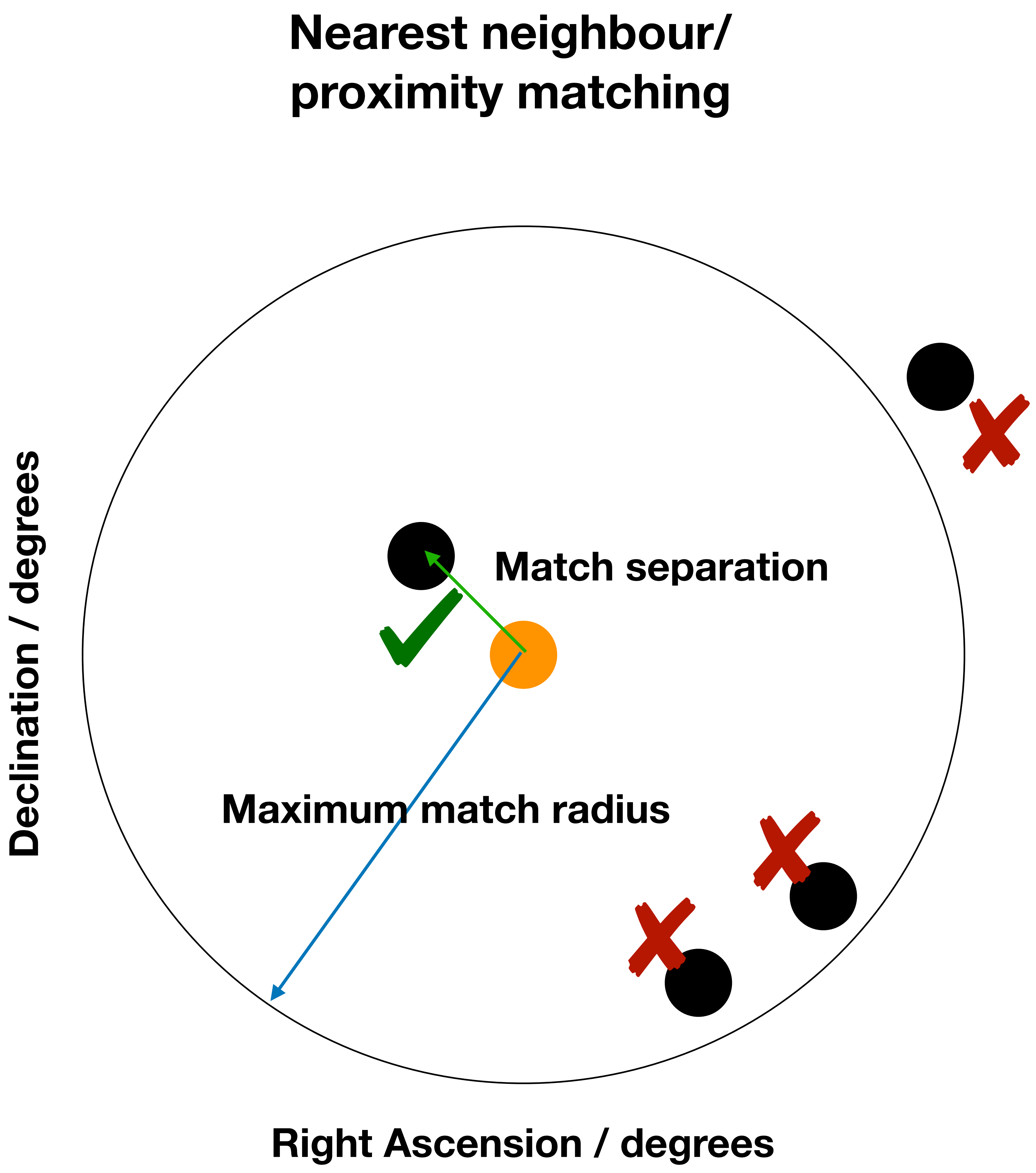


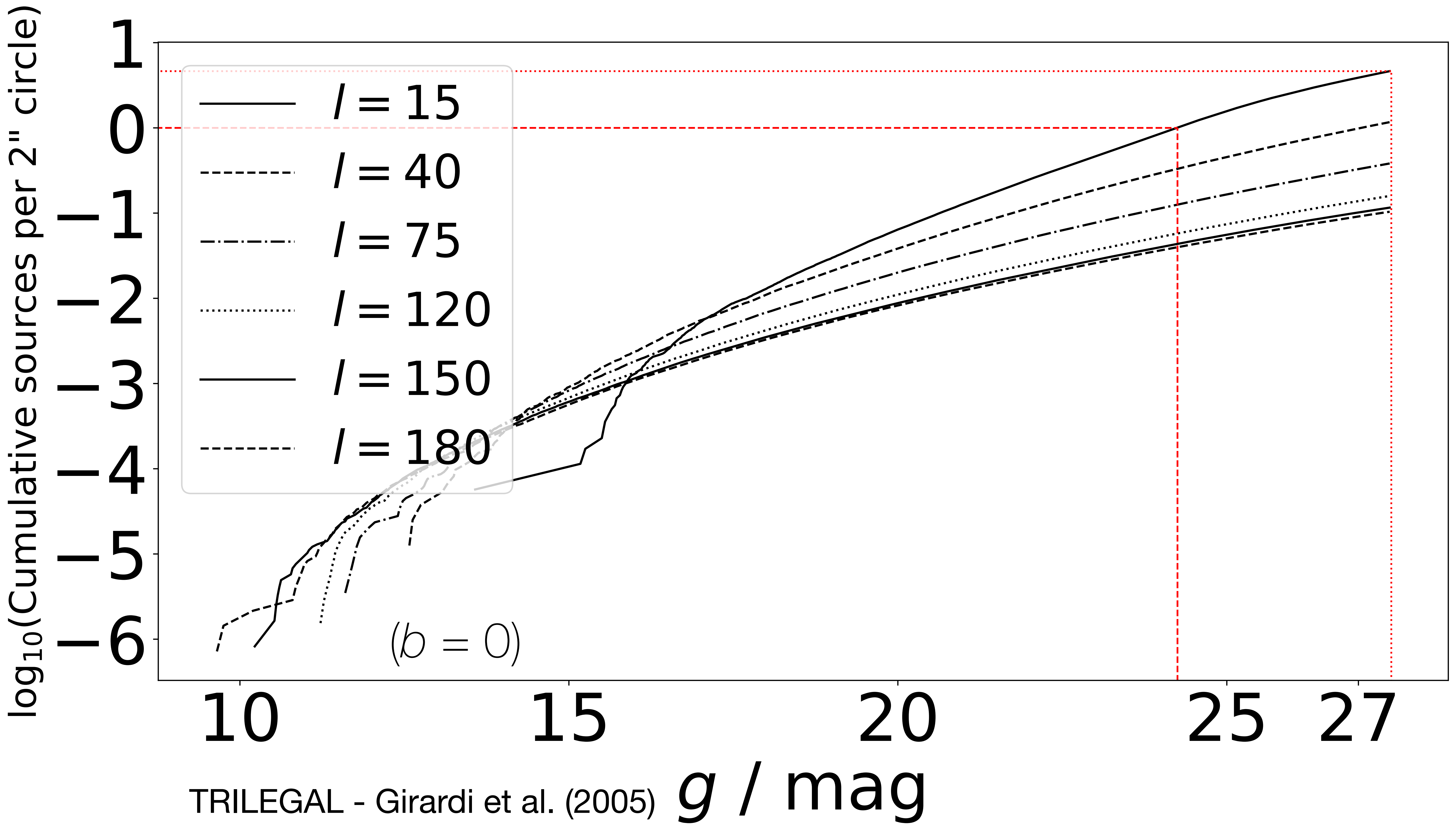




Tom J Wilson @onoddil

The Problem With LSST



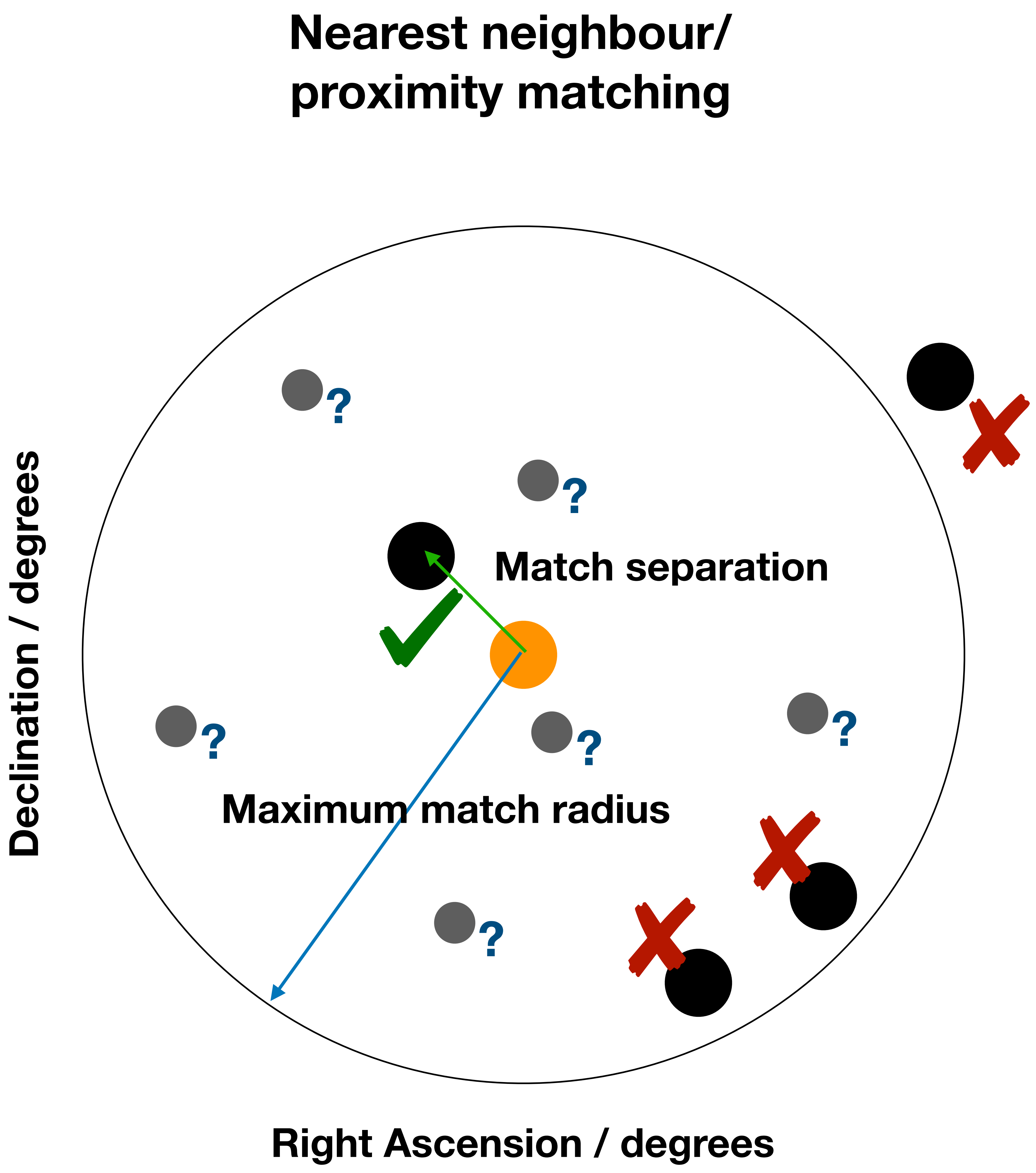


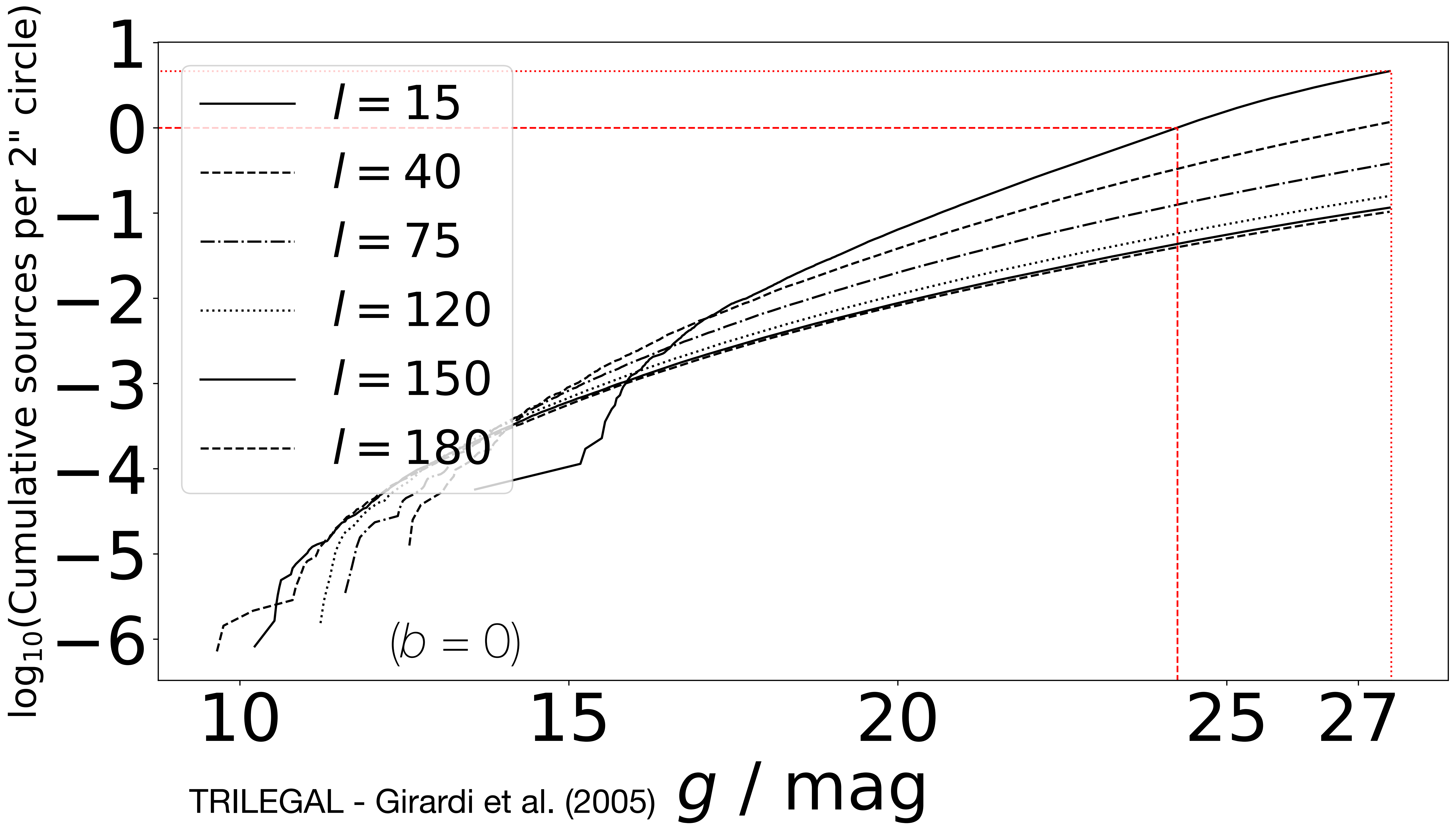
TRILEGAL - Girardi et al. (2005)

Tom J Wilson @onoddil

The Problem With LSST

(It's still a few randomly placed objects in every match radius at high Galactic latitudes)

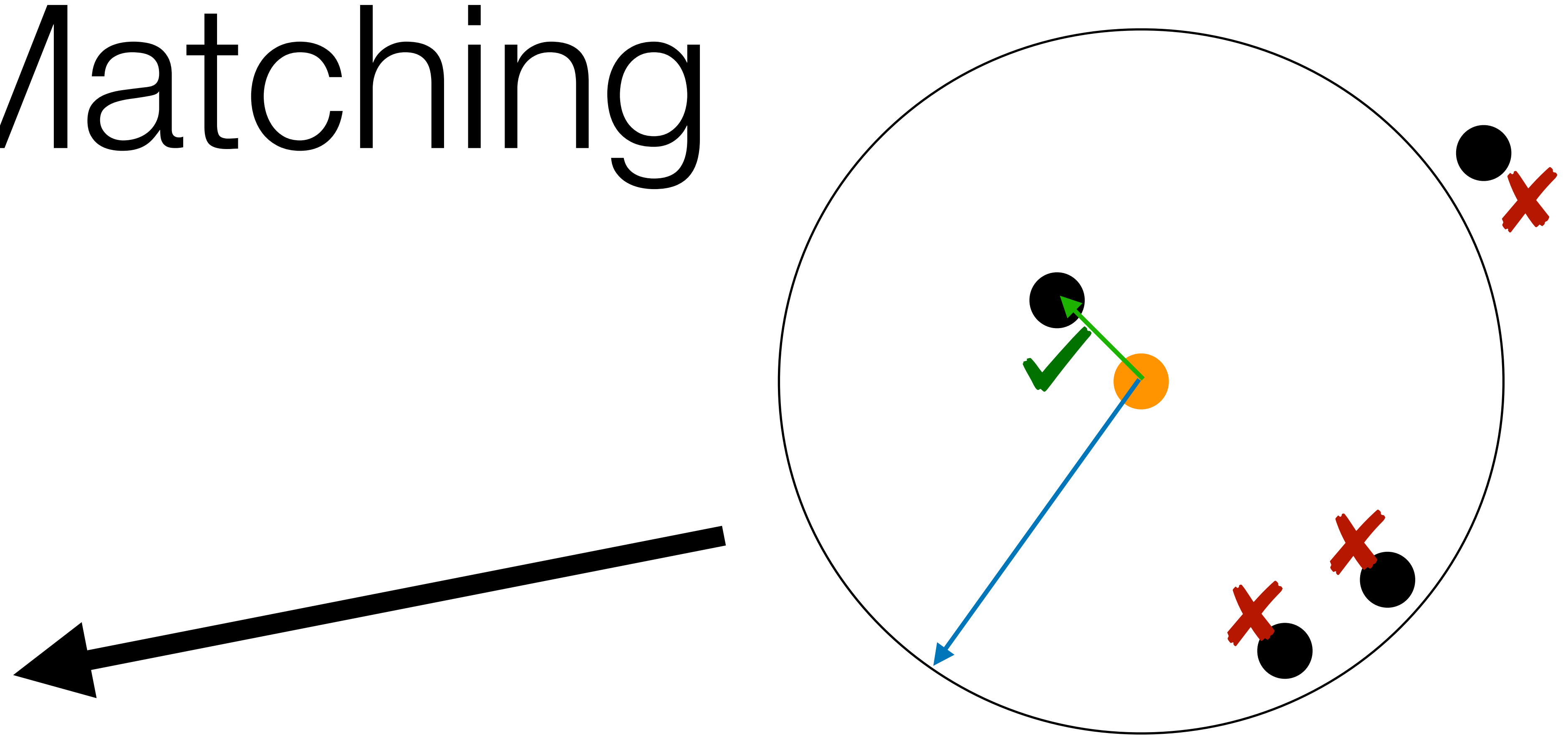


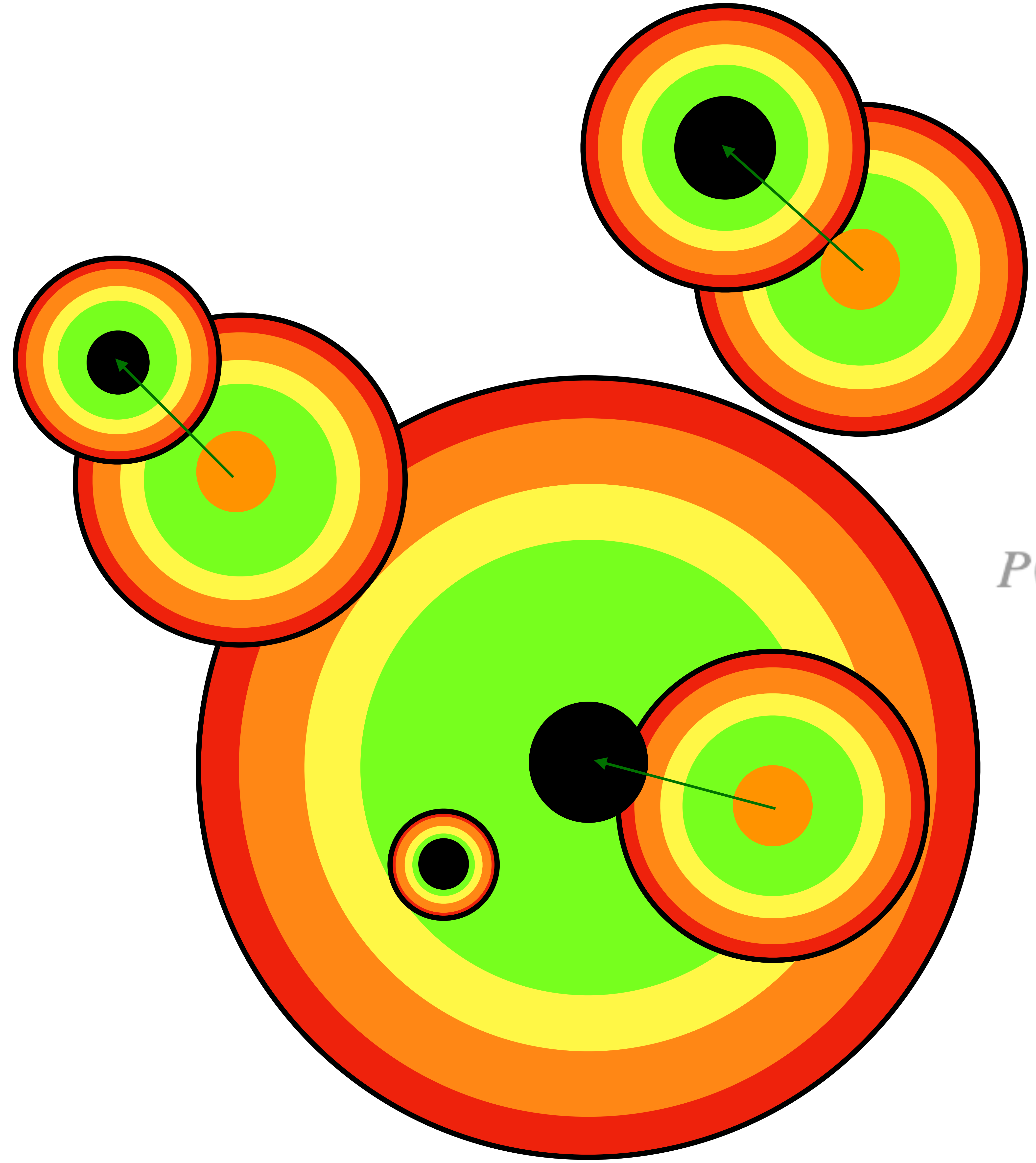


Nearest-neighbour matching *will not* work in the era of Rubin!

Tom J Wilson @onoddil

Probabilistic Cross-Matching





Probability of two sources having their <u>on-sky separation</u> given the hypothesis they are counterparts

$$P(\zeta, \lambda, k|\gamma, \phi) = \frac{1}{K} \times \prod_{\delta \notin \zeta \cap \delta \in \gamma} N_\gamma f_\gamma^\delta \prod_{\omega \notin \lambda \cap \omega \in \phi} N_\phi f_\phi^\omega \prod_{i=1}^{k} N_c G_{\gamma\phi}^{\zeta_i \lambda_i} c_{\gamma\phi}^{\zeta_i \lambda_i}$$

Wilson & Naylor (2018a)

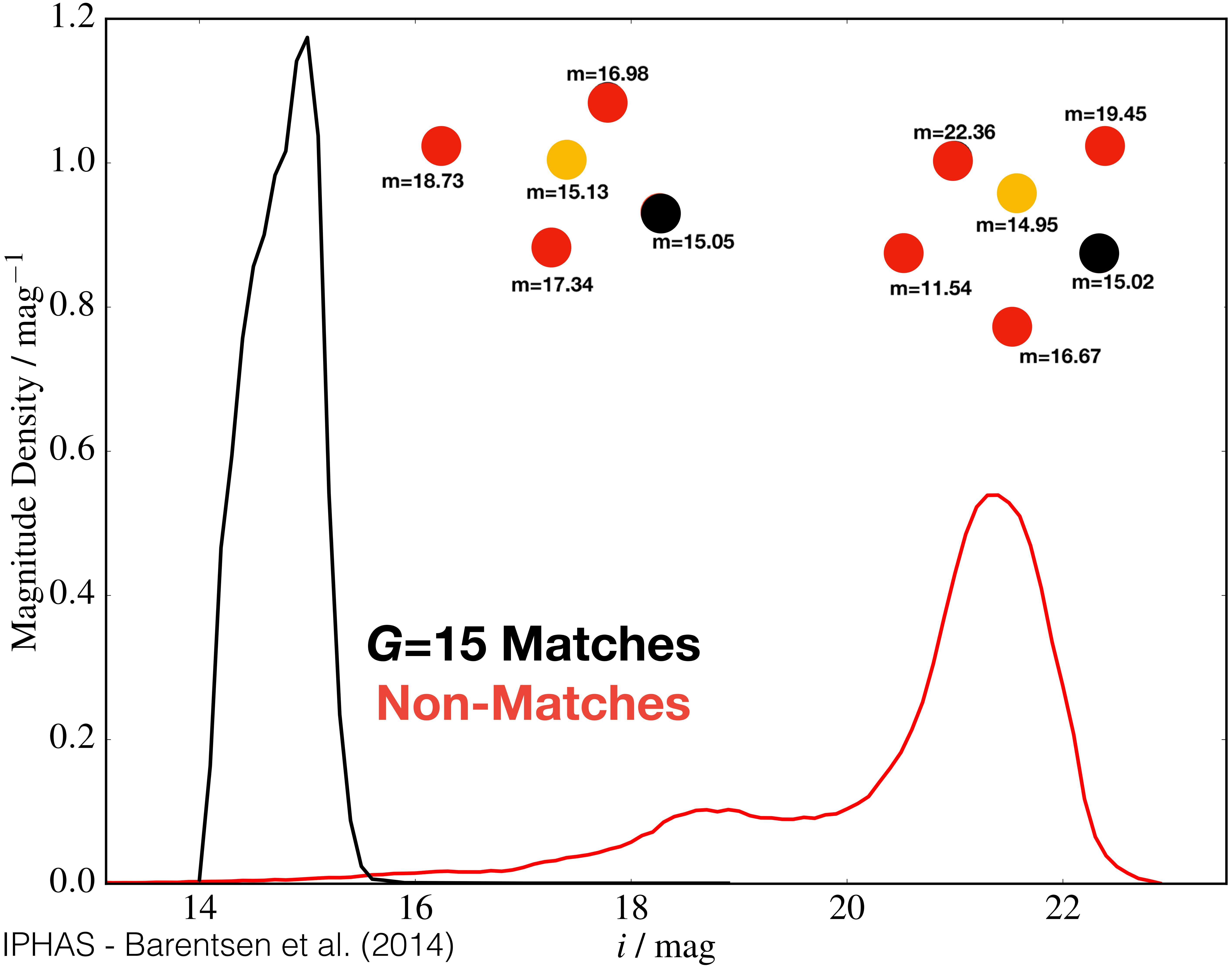
Probability of sources having their <u>brightnesses</u> given they are unrelated to one another ("field stars")

Probability of sources having their <u>brightnesses</u> given they are counterparts

Tom J Wilson @onoddil

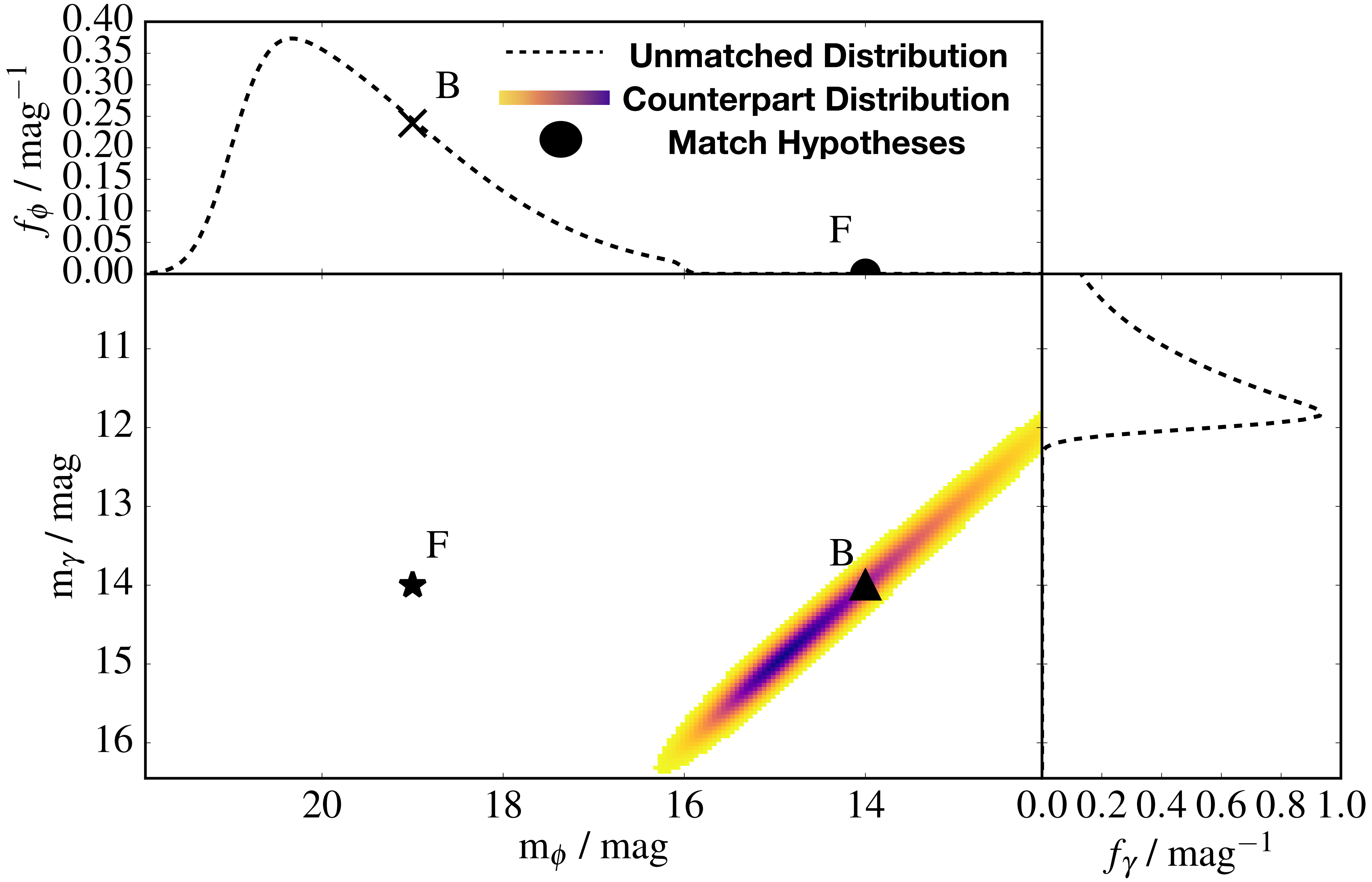
Photometry: Rejecting False Positives

$$P(\zeta, \lambda, k | \gamma, \phi) = \frac{1}{K} \times \prod_{\delta \notin \zeta \cap \delta \in \gamma} N_\gamma \boldsymbol{f_\gamma^\delta} \prod_{\omega \notin \lambda \cap \omega \in \phi} N_\phi \boldsymbol{f_\phi^\omega} \prod_{i=1}^{k} N_\mathrm{c} G_{\gamma\phi}^{\zeta_i \lambda_i} \boldsymbol{c_{\gamma\phi}^{\zeta_i \lambda_i}}$$



IPHAS - Barentsen et al. (2014)

Gaia DR2 - Gaia Collaboration, Brown A. G. A., et al. (2018)

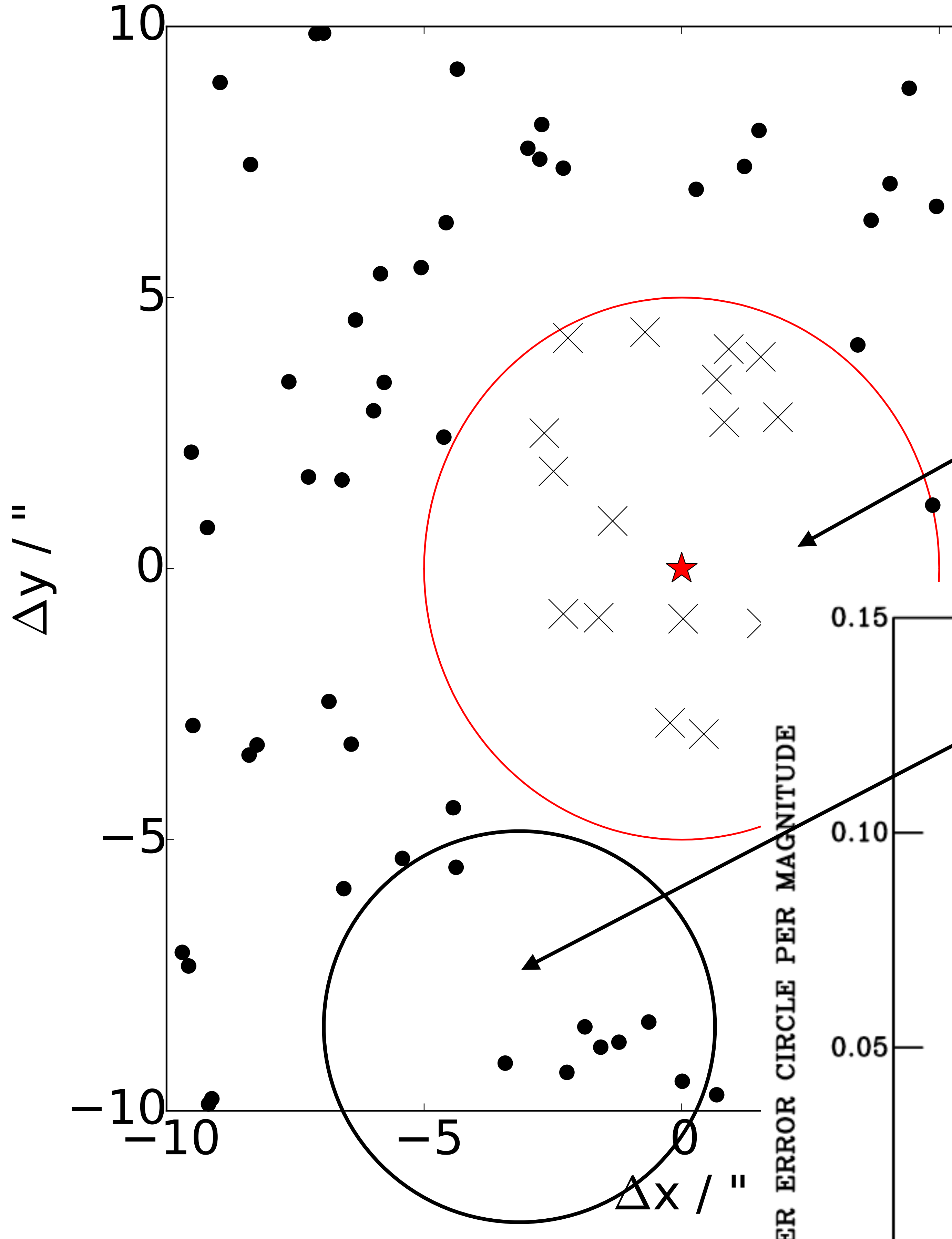


Wilson & Naylor (2018a)

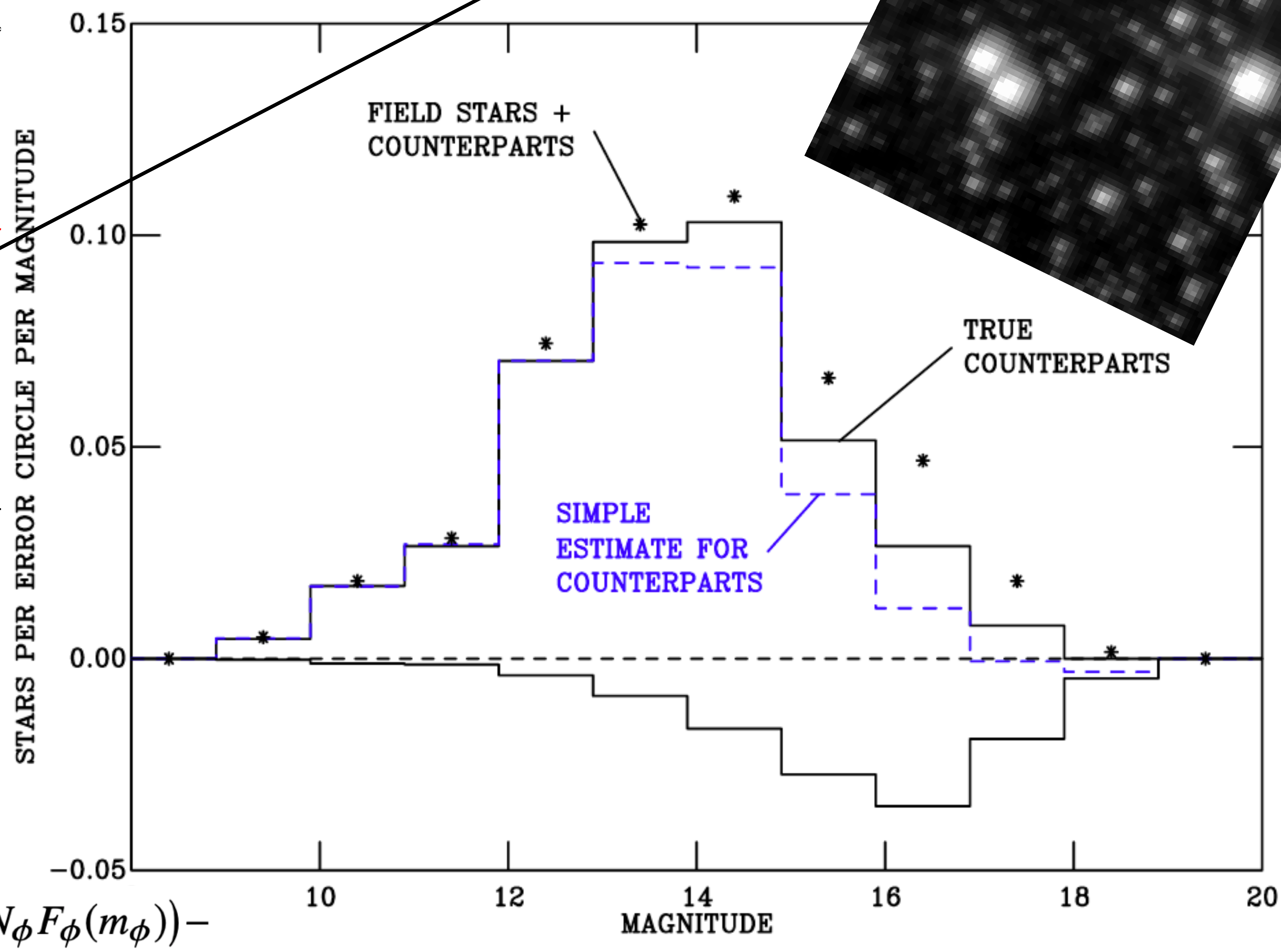
The photometry-based likelihoods (c and f) allow us to mitigate high false positive rate in crowded fields

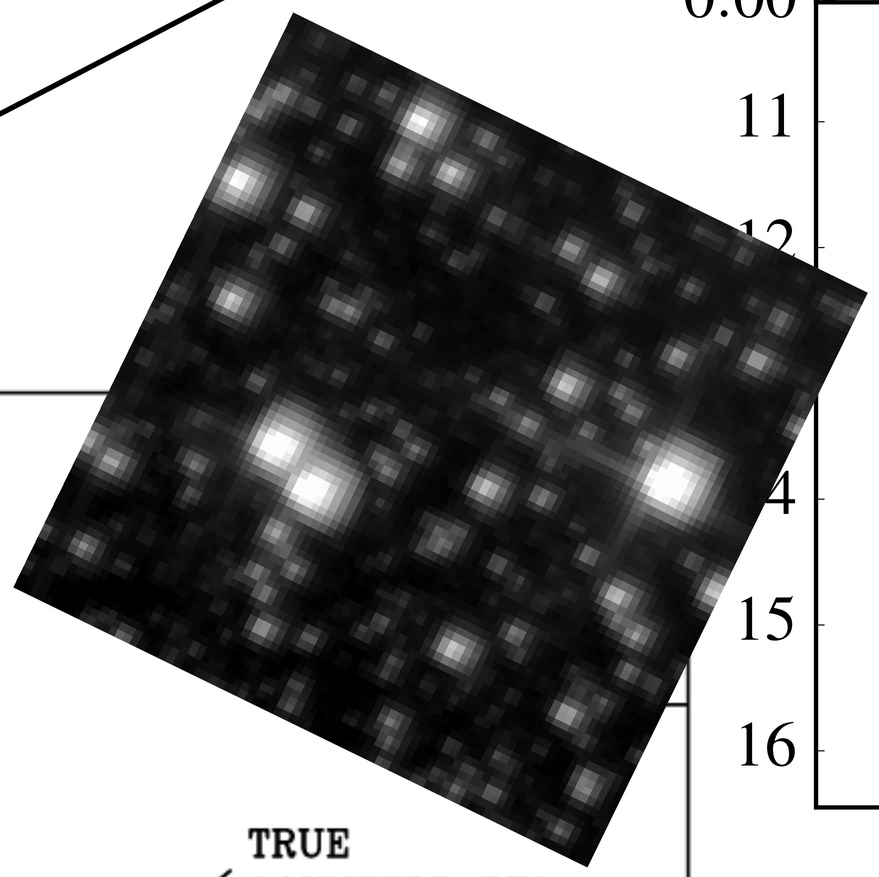
Photometry: Rejecting False Positives

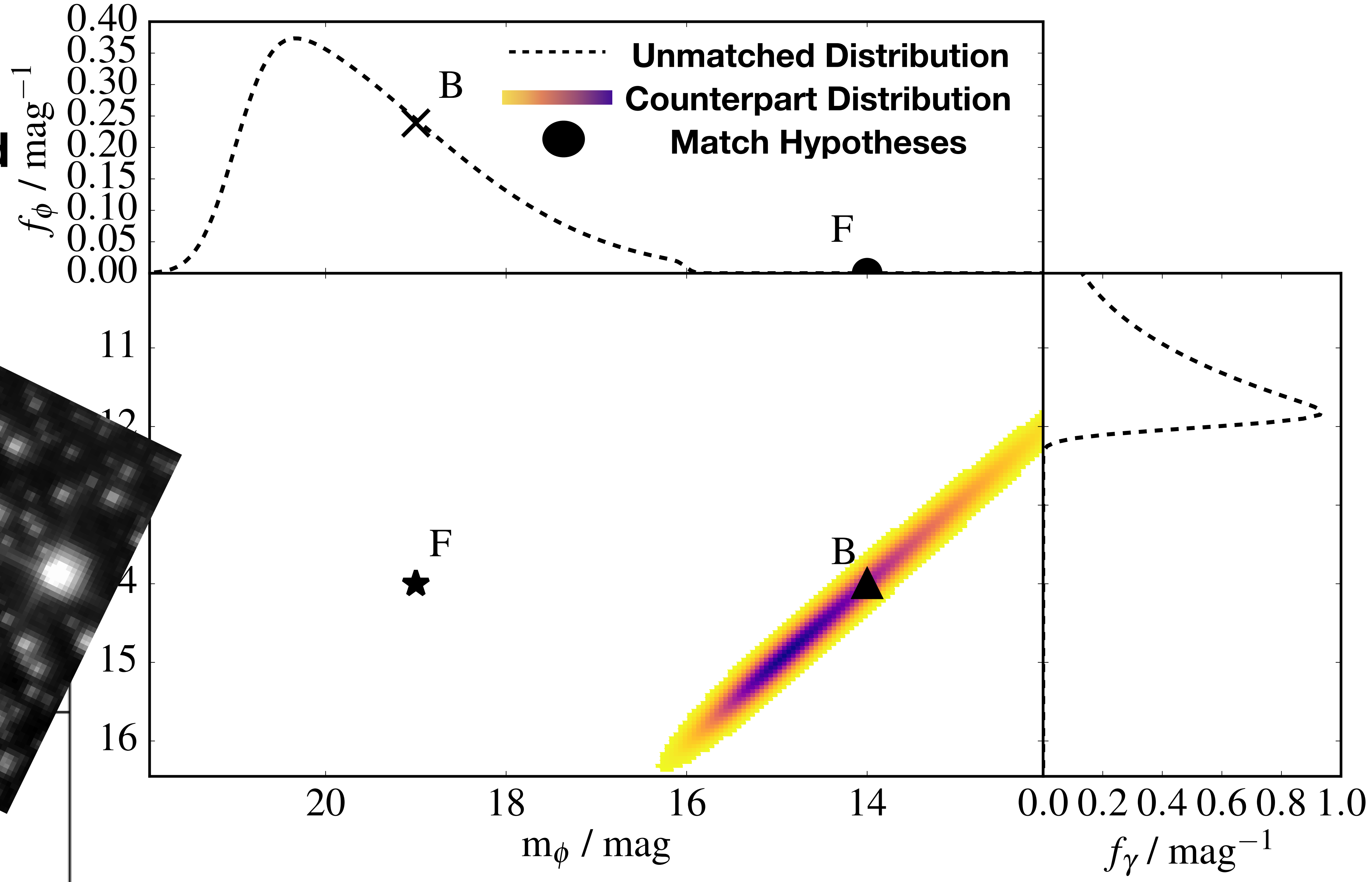
$$P(\zeta,\lambda,k|\gamma,\phi) = \frac{1}{K} \times \prod_{\delta\notin\zeta\cap\delta\in\gamma} N_\gamma f_\gamma^\delta \prod_{\omega\notin\lambda\cap\omega\in\phi} N_\phi f_\phi^\omega \prod_{i=1}^{k} N_c G_{\gamma\phi}^{\zeta_i\lambda_i} c_{\gamma\phi}^{\zeta_i\lambda_i}$$



Counterparts = ("Counterparts" + field) - field







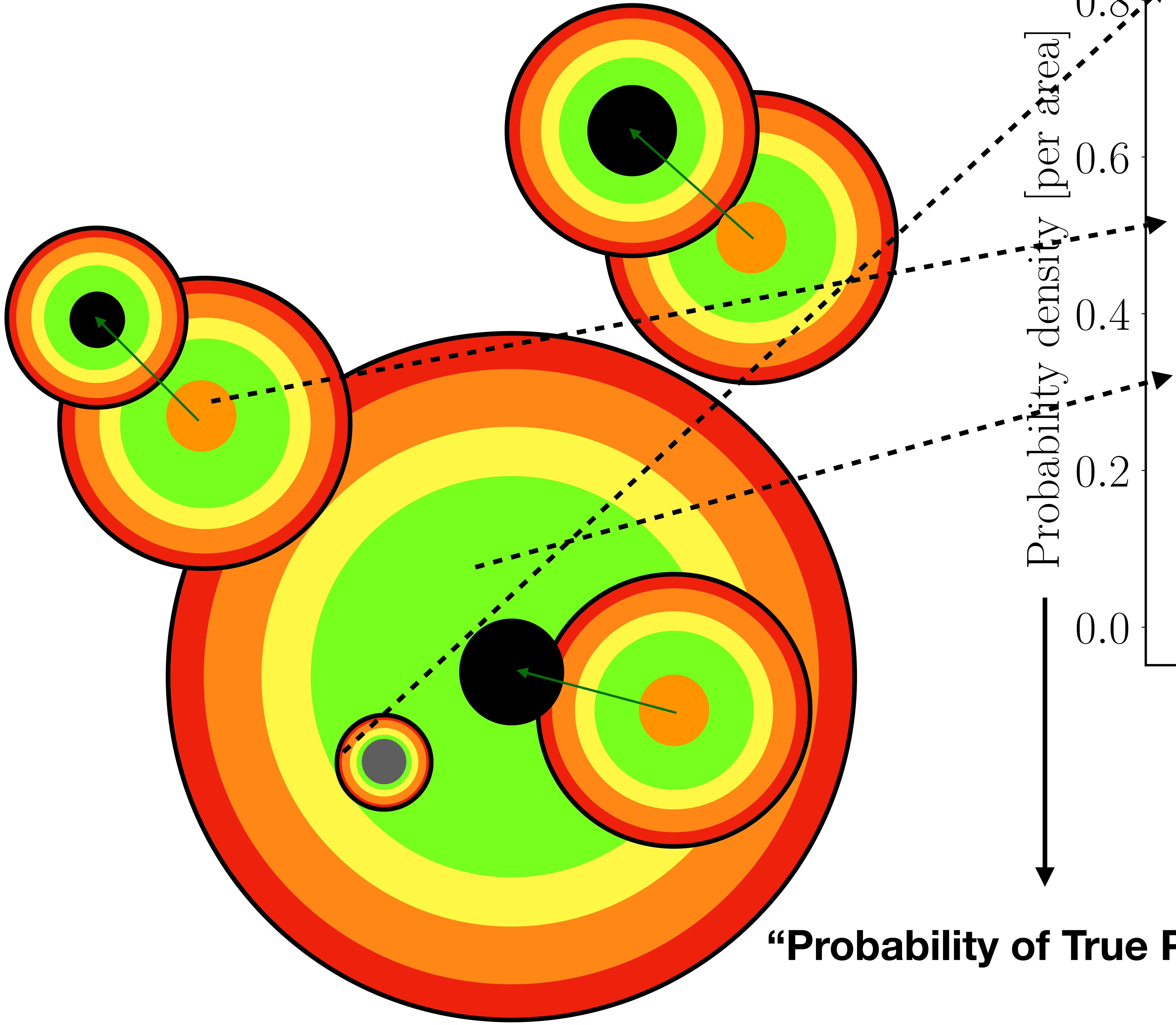
Wilson & Naylor (2018a)

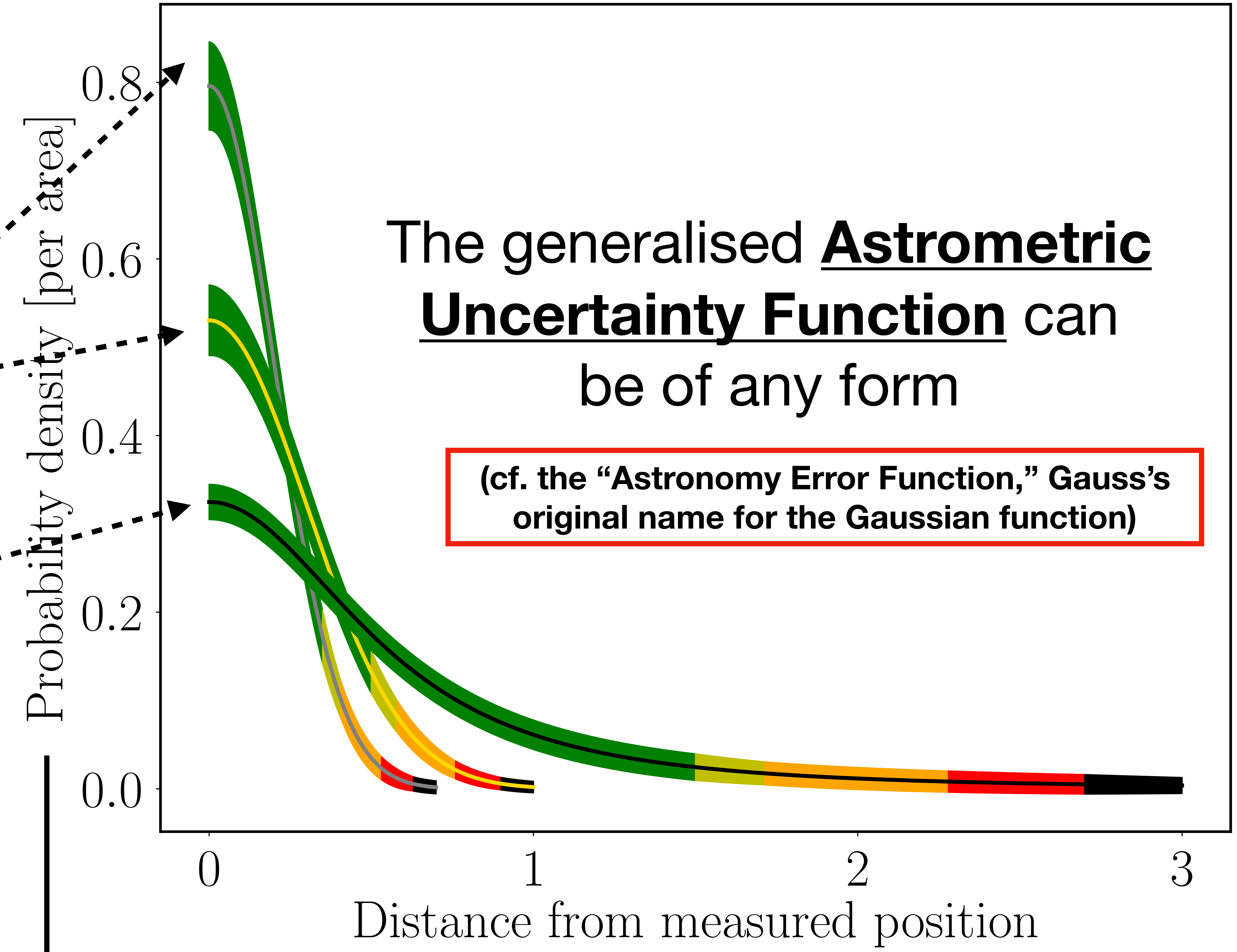
$$Z_{c\gamma}\cdot \boxed{c_\gamma(m_\phi|m_\gamma)} = Z_\gamma b_\gamma(m_\phi|m_\gamma)\exp\left(A_\gamma N_\phi F_\phi(m_\phi)\right) - \left(1 - Z_{c\gamma}C_\gamma(m_\phi|m_\gamma)\right) A_\gamma N_\phi f_\phi(m_\phi).$$

The photometry-based likelihoods (c and f) allow us to mitigate high false positive rate in crowded fields

Naylor, Broos, & Feigelson (2013); Wilson & Naylor (2018a)

Probabilistic Cross-Matching: the AUF

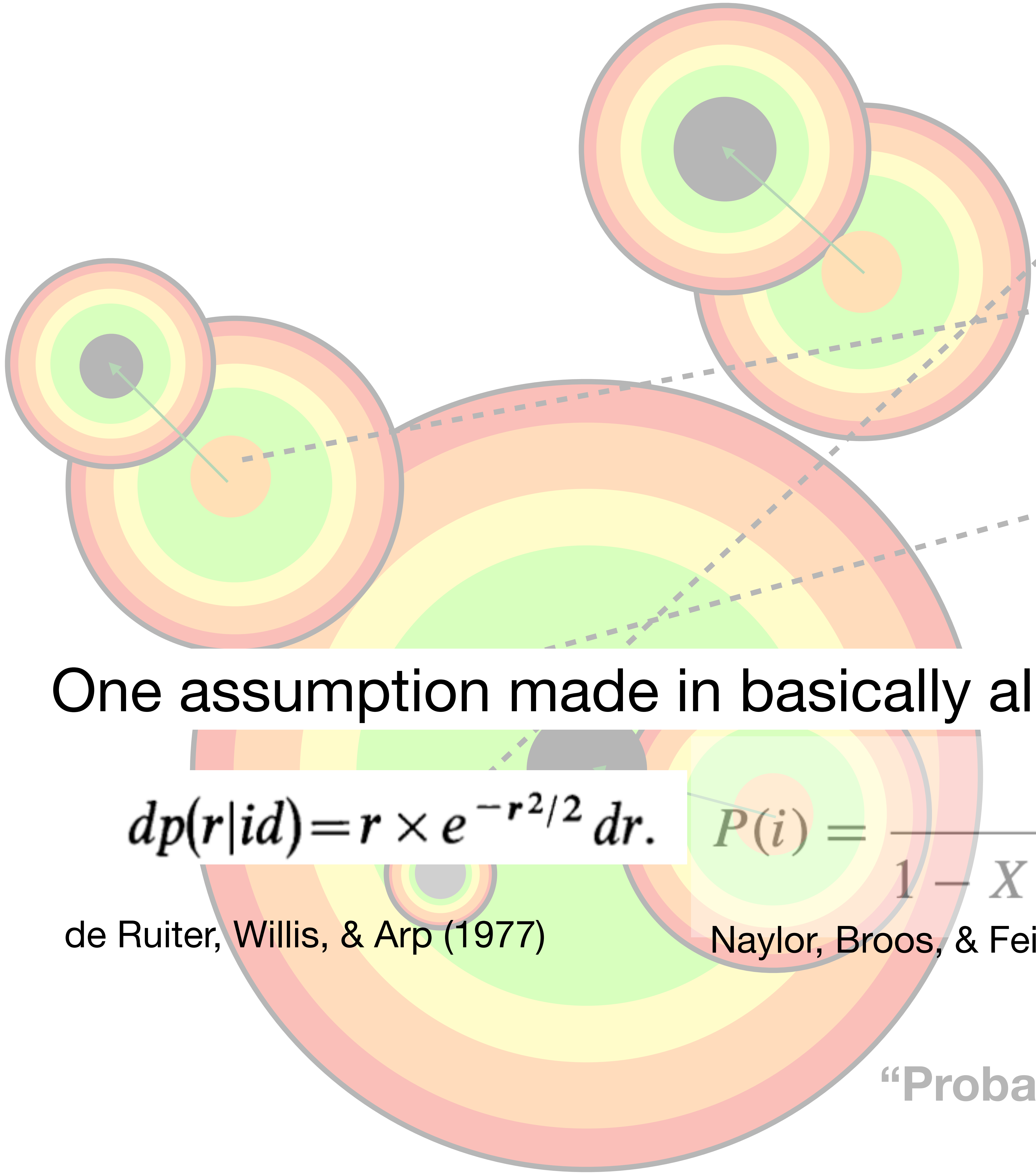


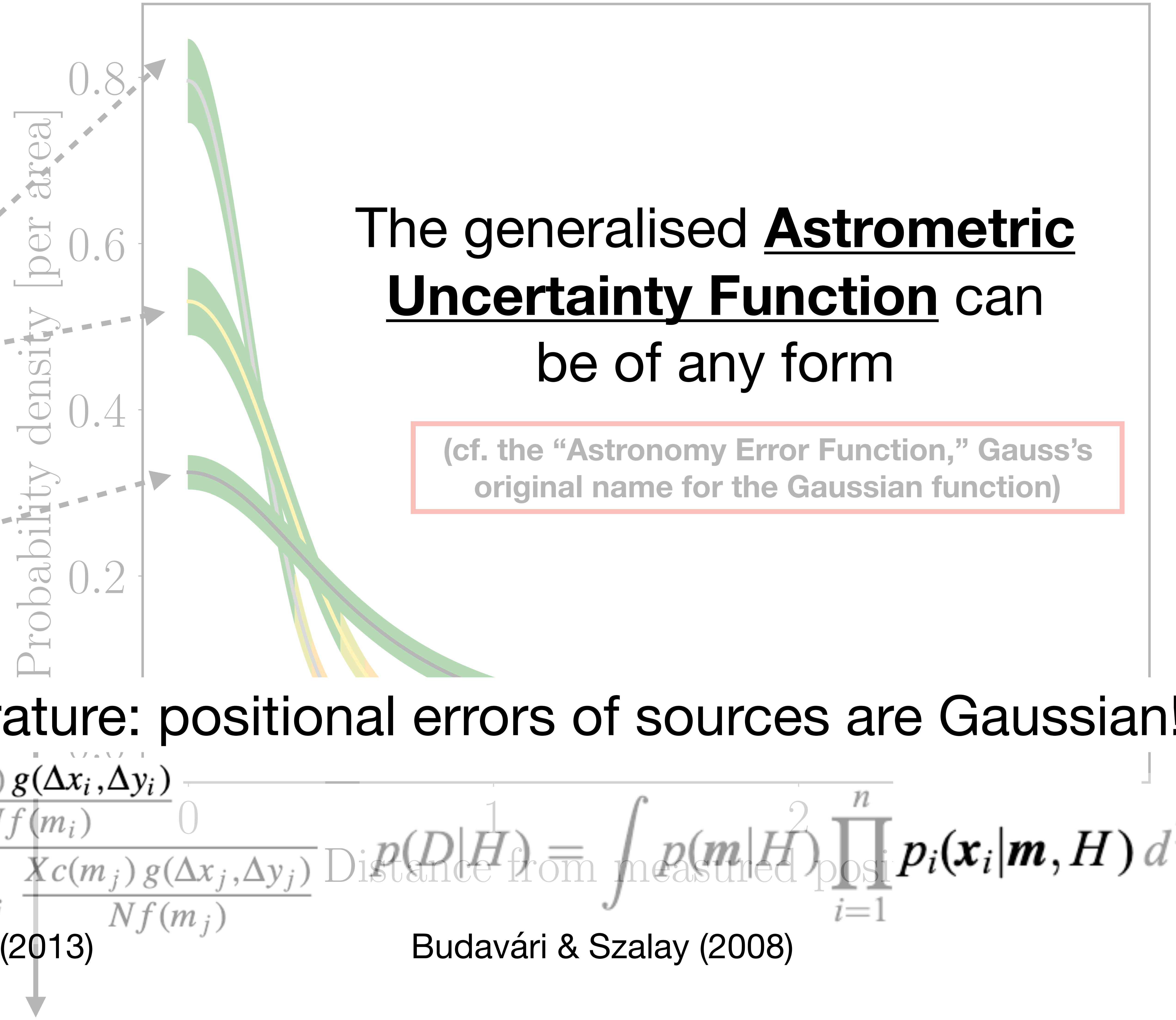


"Probability of True Position being this far from the Measured Position"

Tom J Wilson @onoddil

Probabilistic Cross-Matching: the AUF





The generalised **Astrometric Uncertainty Function** can be of any form

(cf. the "Astronomy Error Function," Gauss's original name for the Gaussian function)

One assumption made in basically all literature: positional errors of sources are Gaussian!

$$dp(r|id) = r \times e^{-r^2/2}\, dr.$$

de Ruiter, Willis, & Arp (1977)

$$P(i) = \frac{\frac{X c(m_i)\, g(\Delta x_i, \Delta y_i)}{N f(m_i)}}{1 - X + \sum_j \frac{X c(m_j)\, g(\Delta x_j, \Delta y_j)}{N f(m_j)}}$$

Naylor, Broos, & Feigelson (2013)

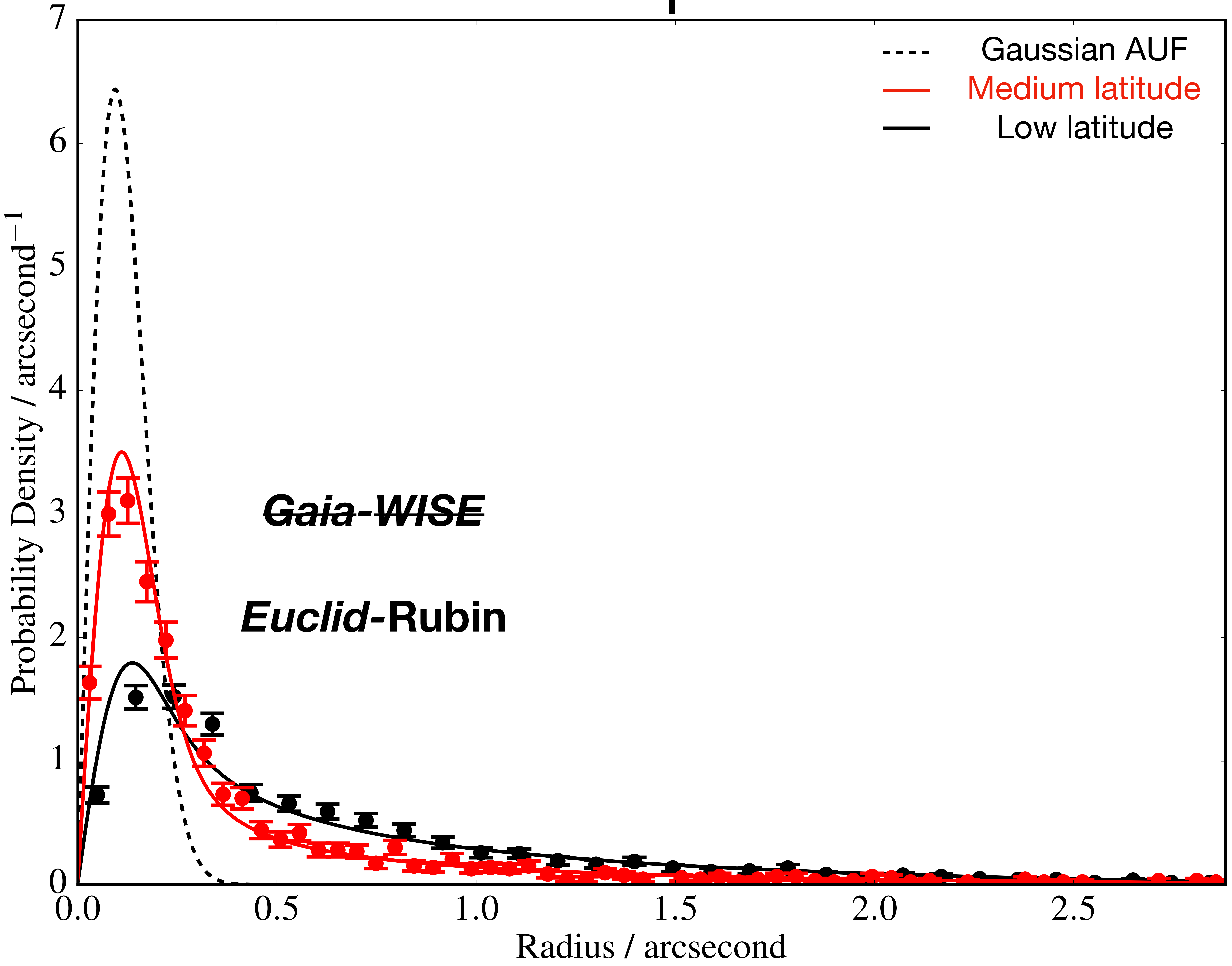
$$p(D|H) = \int p(\boldsymbol{m}|H) \prod_{i=1}^{n} p_i(\boldsymbol{x}_i|\boldsymbol{m}, H)\, d^3m$$

Budavári & Szalay (2008)

"Probability of True Position being this far from the Measured Position"

Tom J Wilson @onoddil

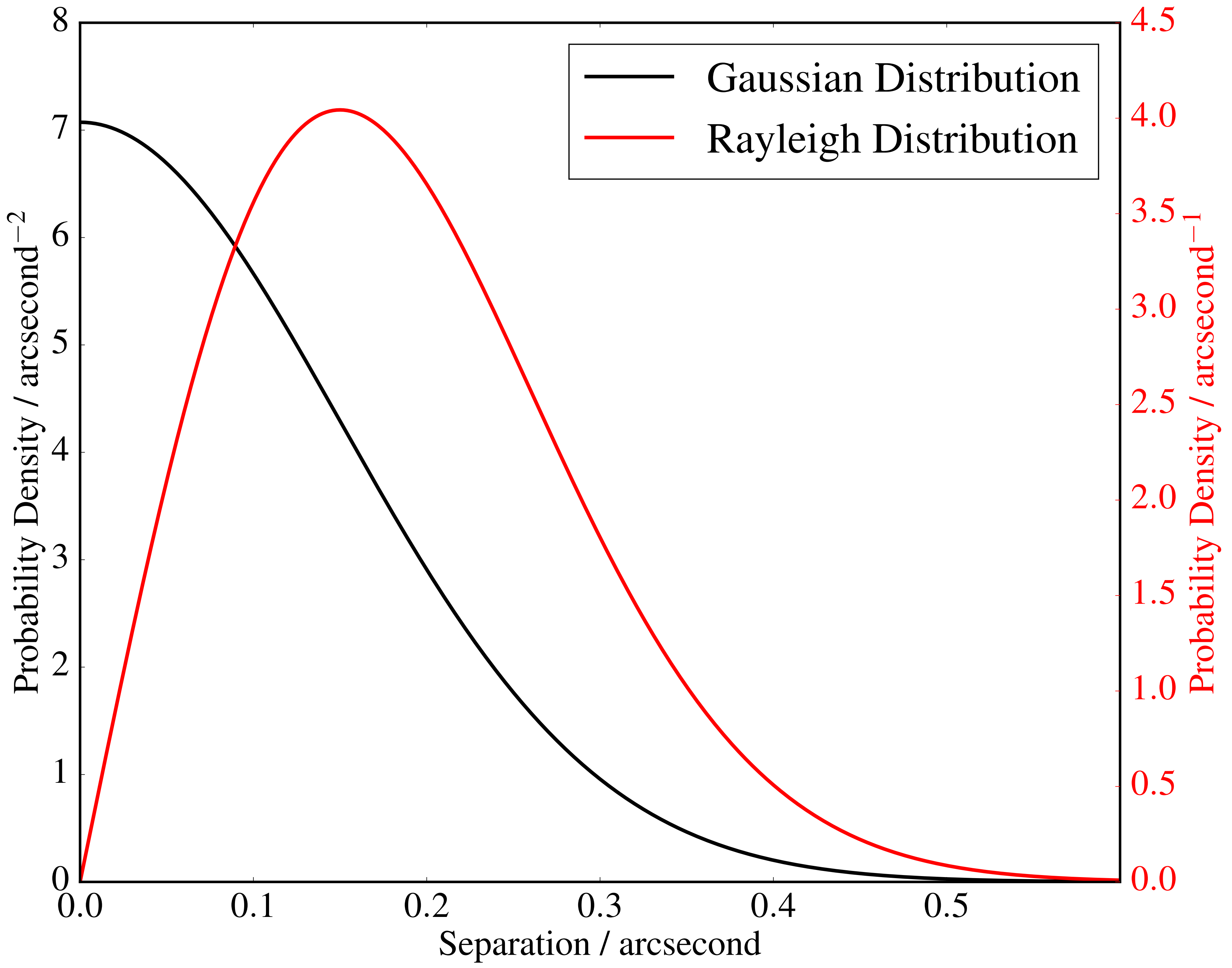
Additional Components of the AUF



$$g(x, y, \sigma) = (2\pi\sigma^2)^{-1} \exp\left(-\frac{1}{2}\frac{x^2 + y^2}{\sigma^2}\right)$$

$$\downarrow$$

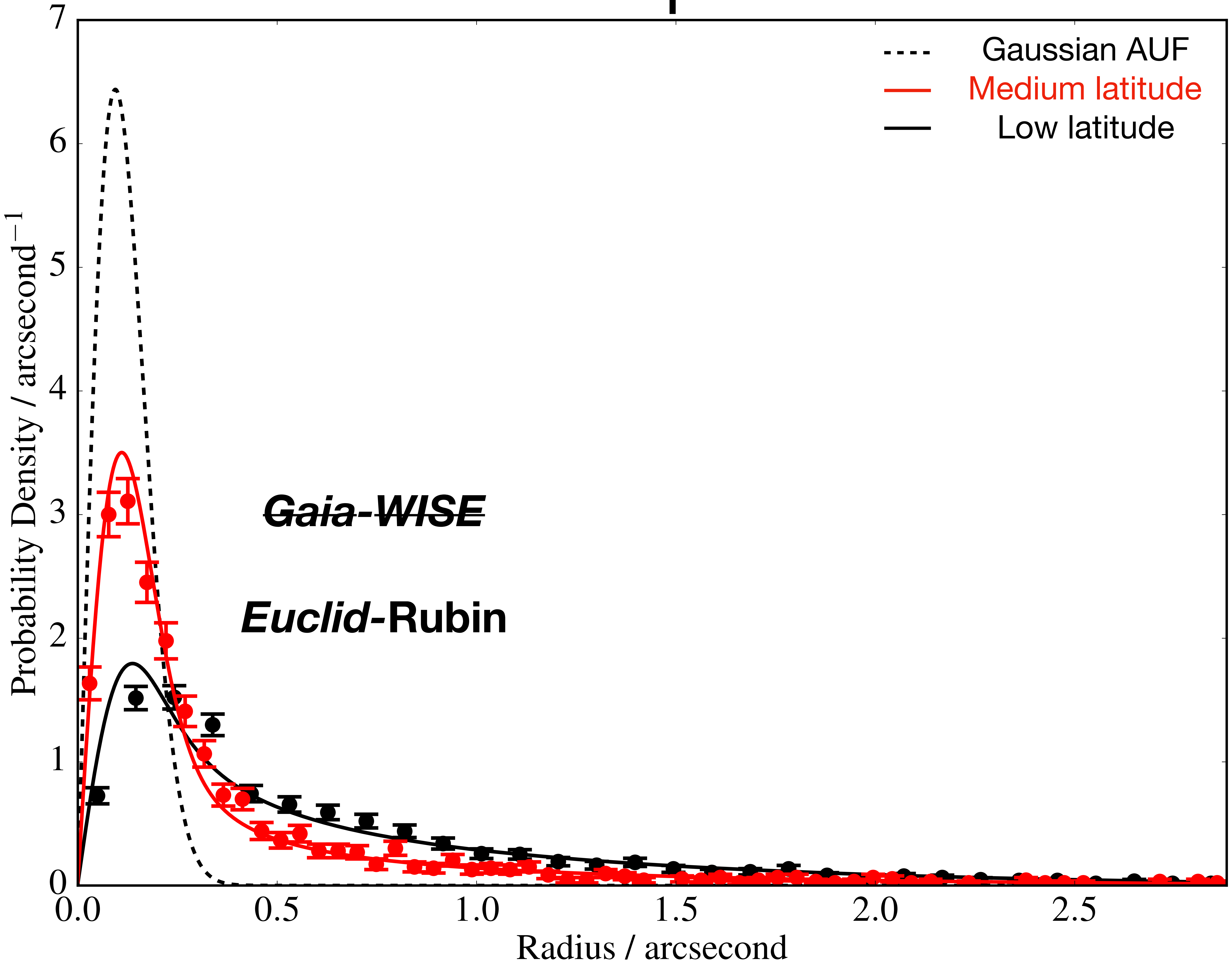
$$g(r, \sigma) = \frac{r}{\sigma^2} \exp\left(-\frac{1}{2}\frac{r^2}{\sigma^2}\right)$$

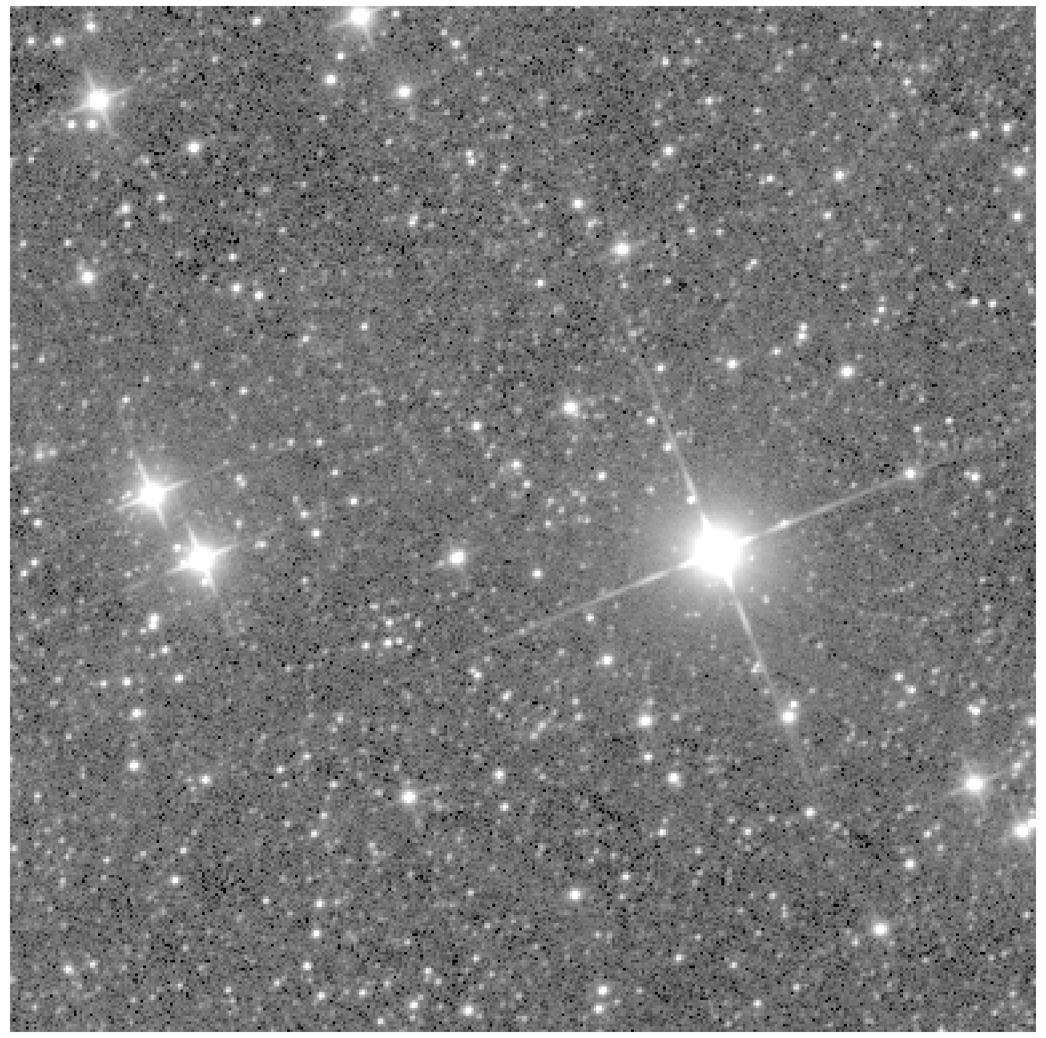


WISE - Wright et al. (2010)
Gaia DR2 - Gaia Collaboration, Brown A. G. A., et al. (2018)

Tom J Wilson @onoddil

Additional Components of the AUF

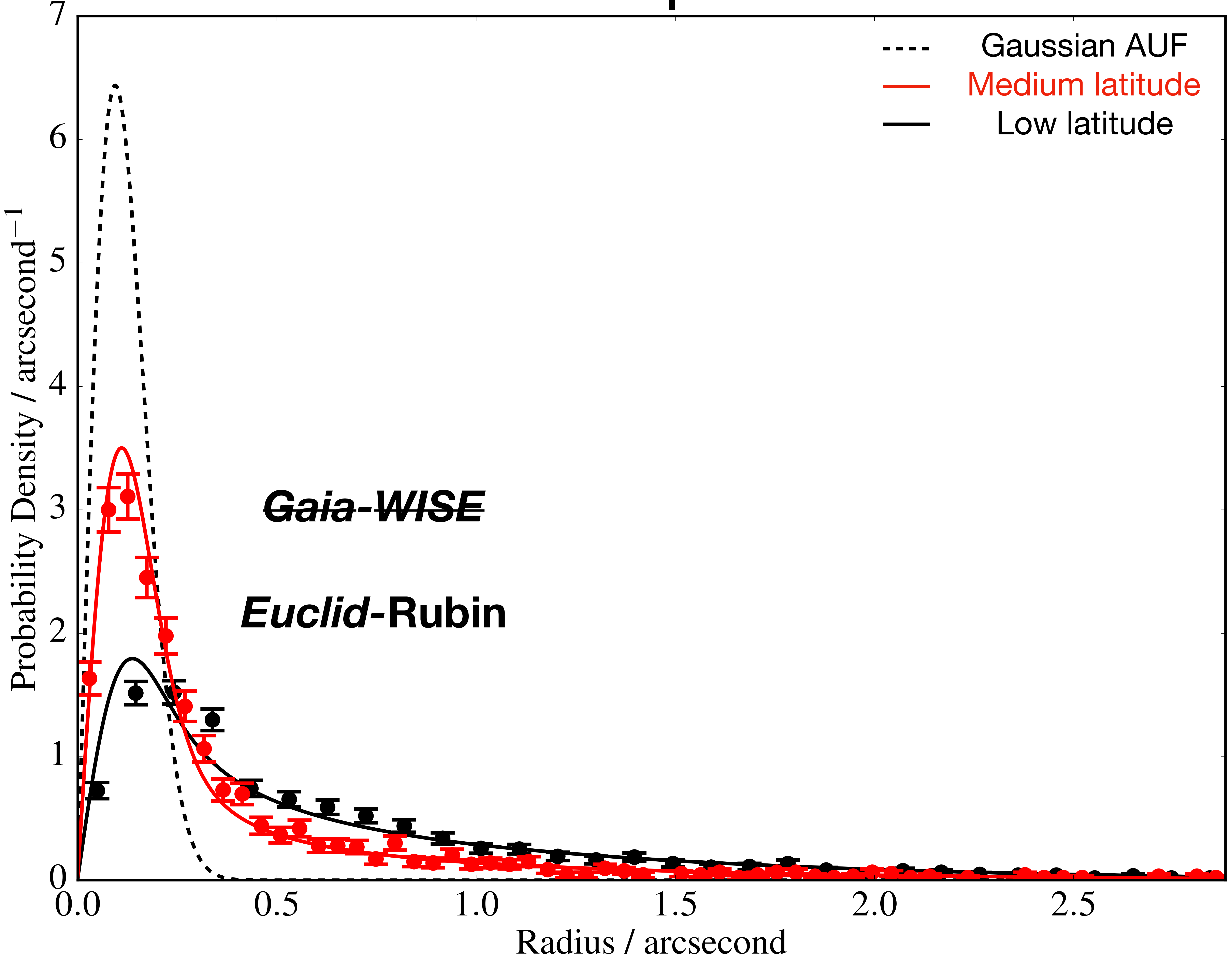


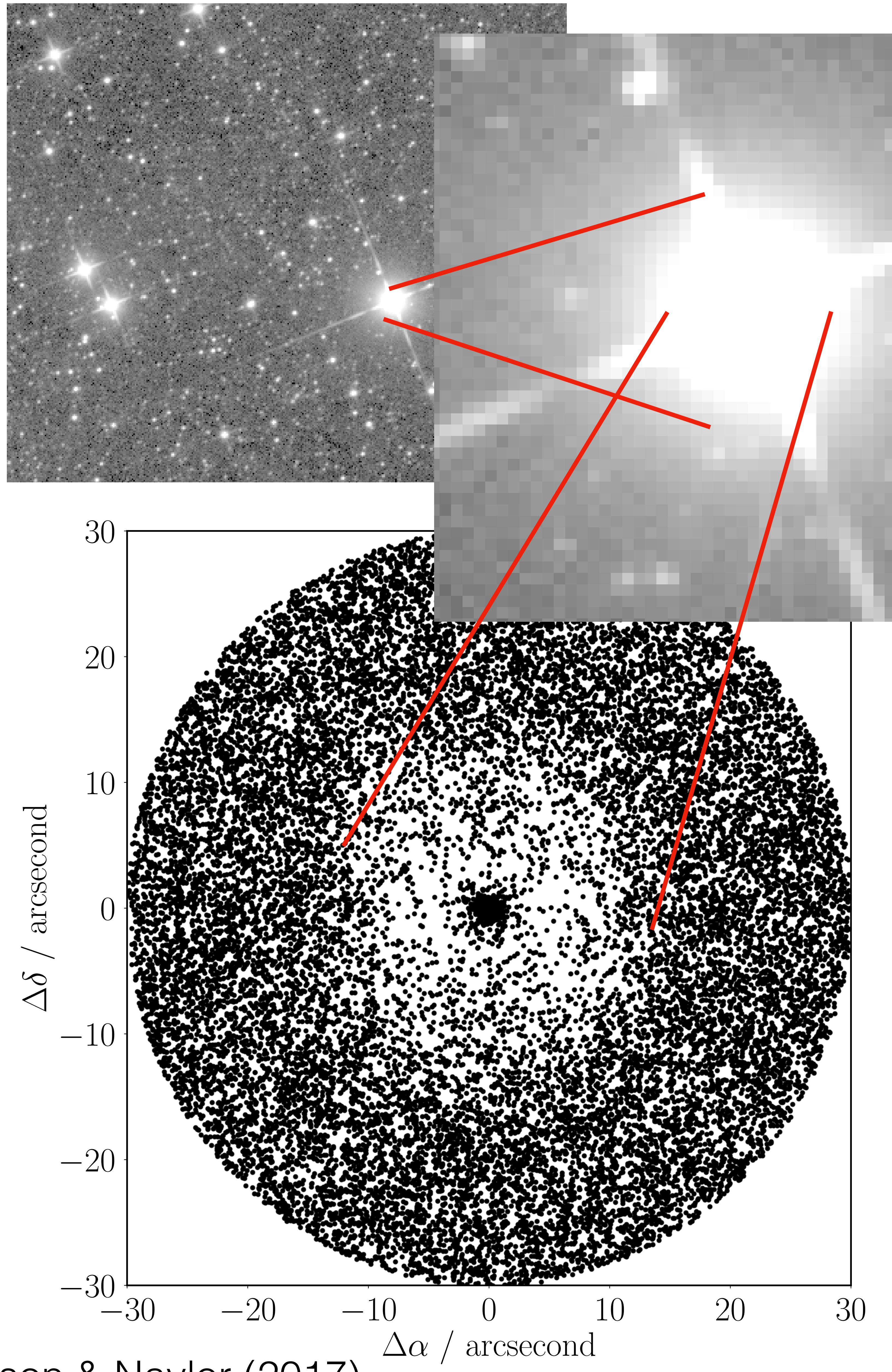


WISE - Wright et al. (2010)
Gaia DR2 - Gaia Collaboration, Brown A. G. A., et al. (2018)

Additional Components of the AUF

(and any other systematic — e.g. proper motions, cf. Wilson 2023, RASTI)





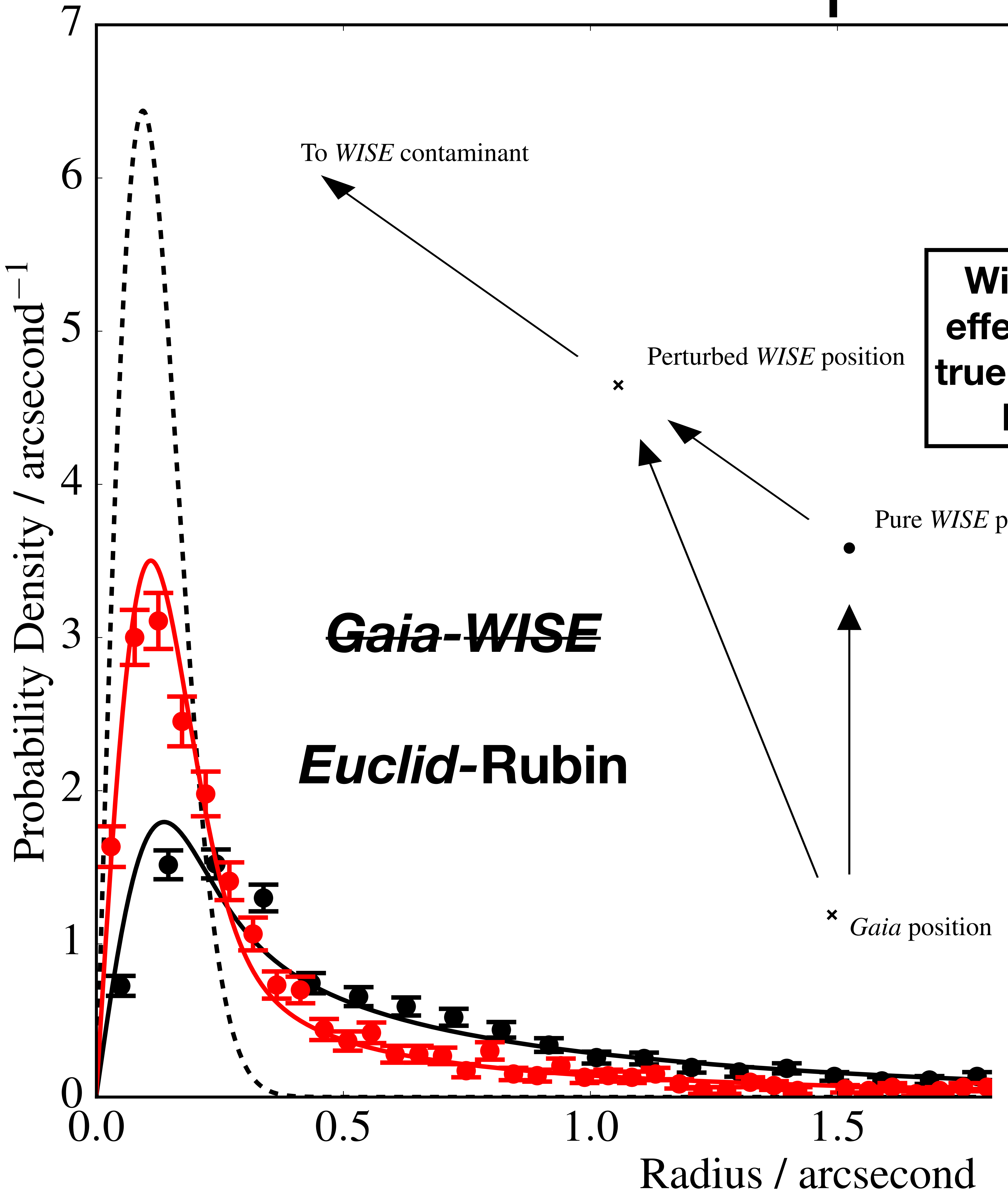
Wilson & Naylor (2017)

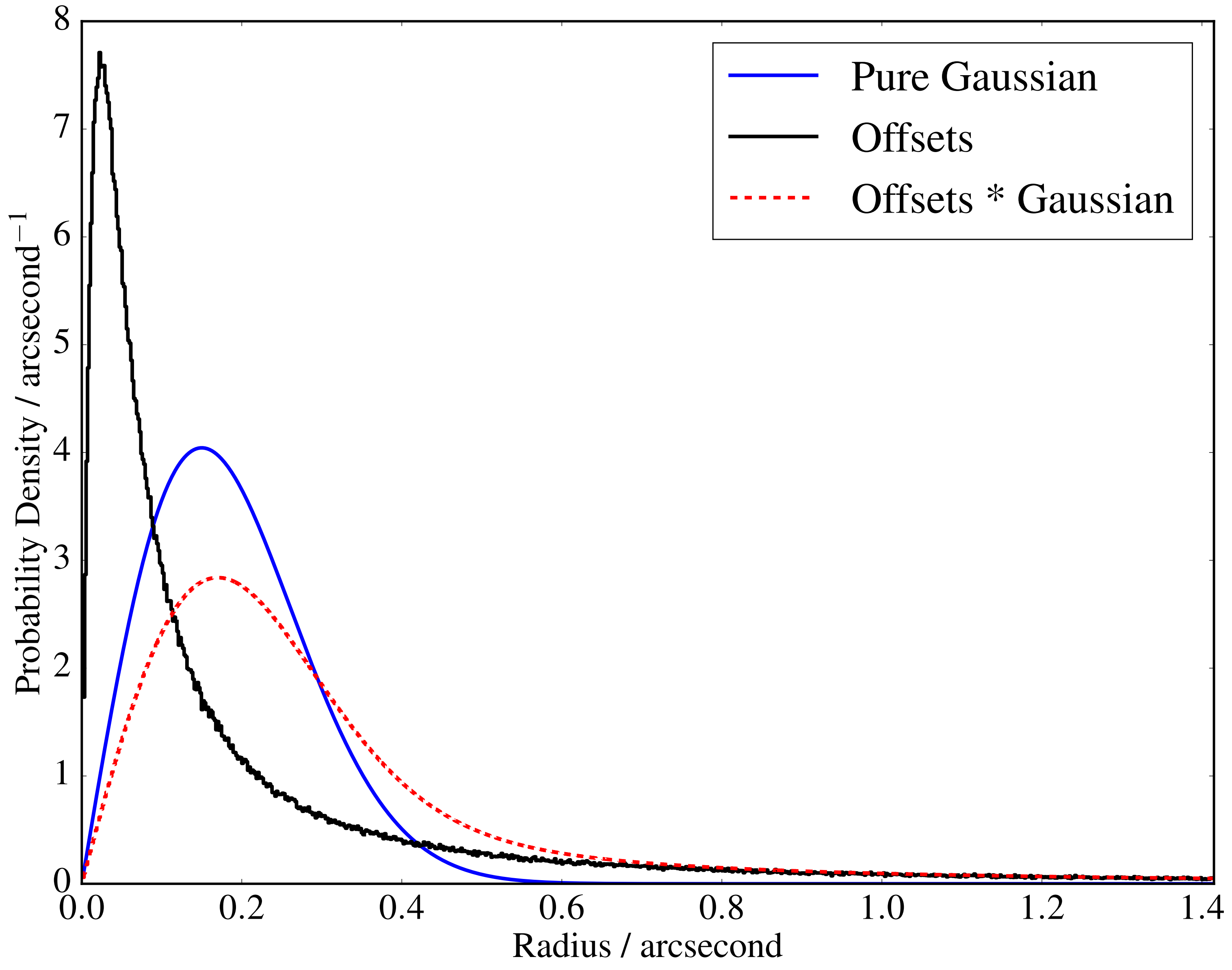
WISE - Wright et al. (2010)
Gaia DR2 - Gaia Collaboration, Brown A. G. A., et al. (2018)

Tom J Wilson @onoddil

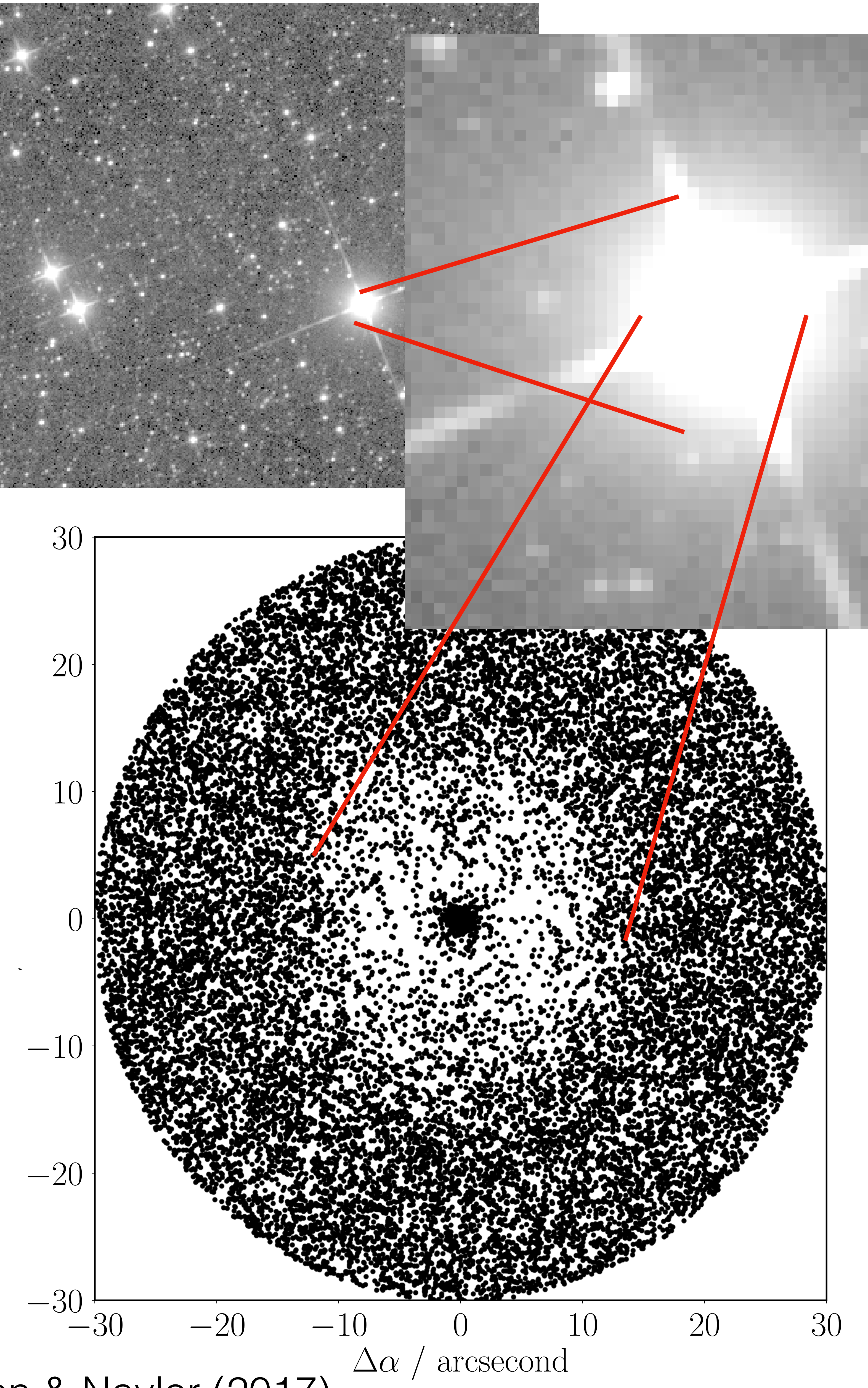
Additional Components of the AUF

(and any other systematic — e.g. proper motions, cf. Wilson 2023, RASTI)





Wilson & Naylor (2018b)

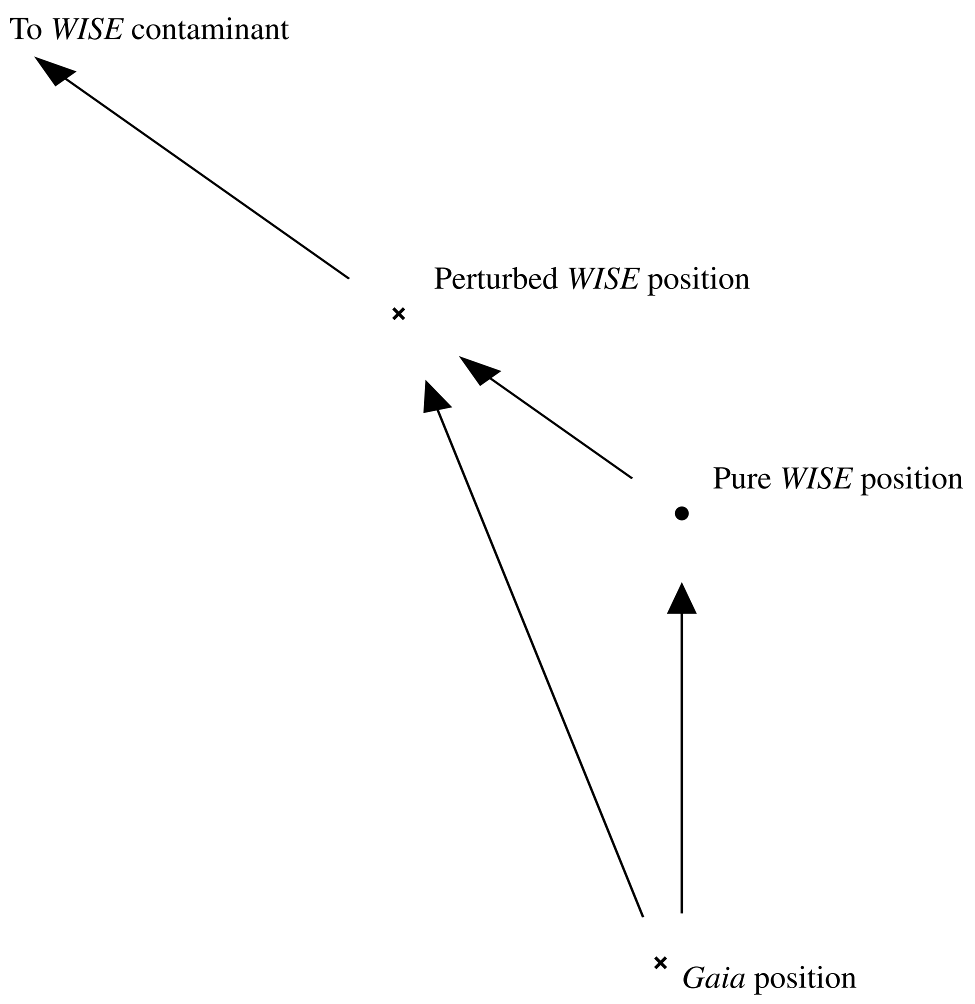


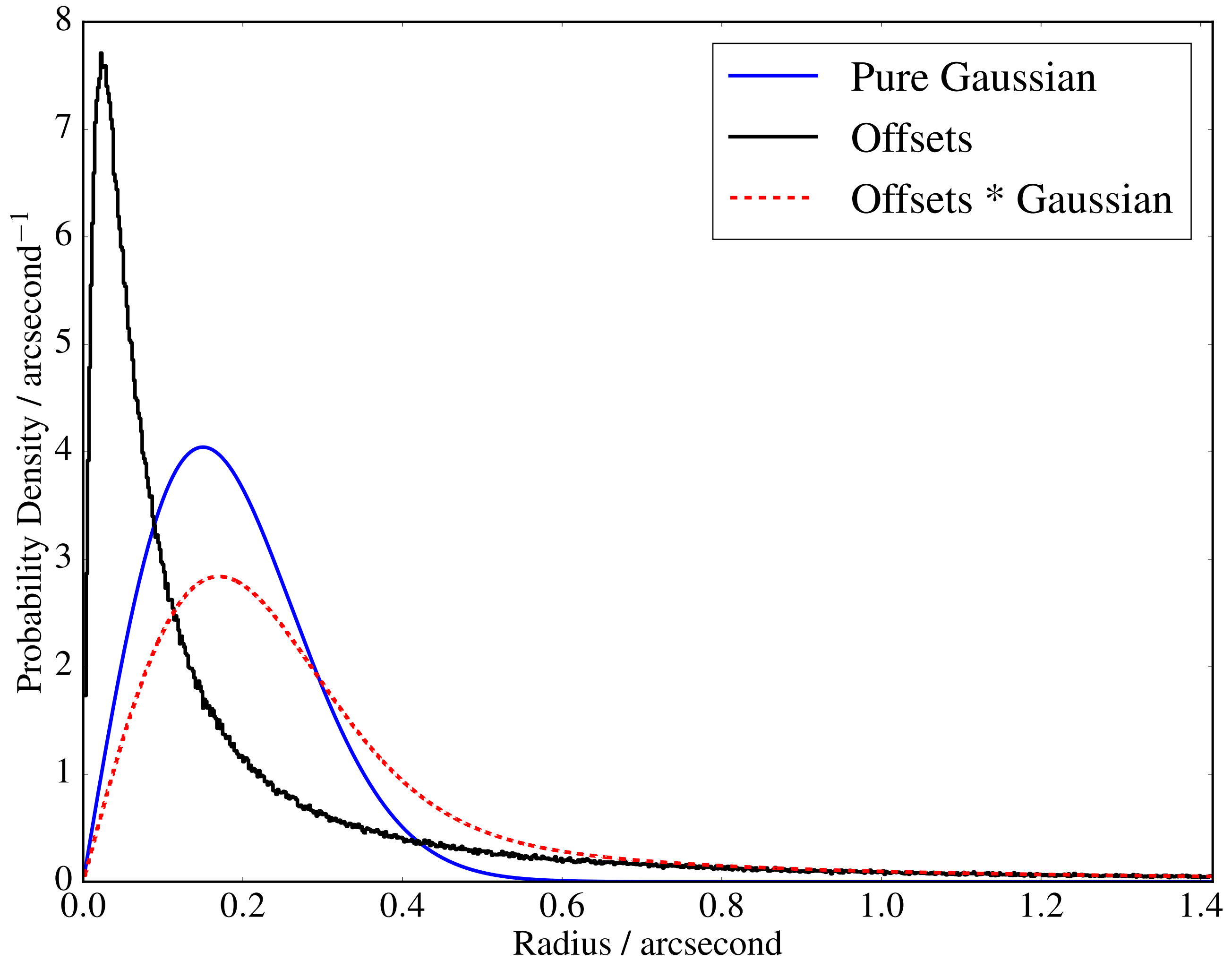
Wilson & Naylor (2017)

WISE - Wright et al. (2010)
Gaia DR2 - Gaia Collaboration, Brown A. G. A., et al. (2018)

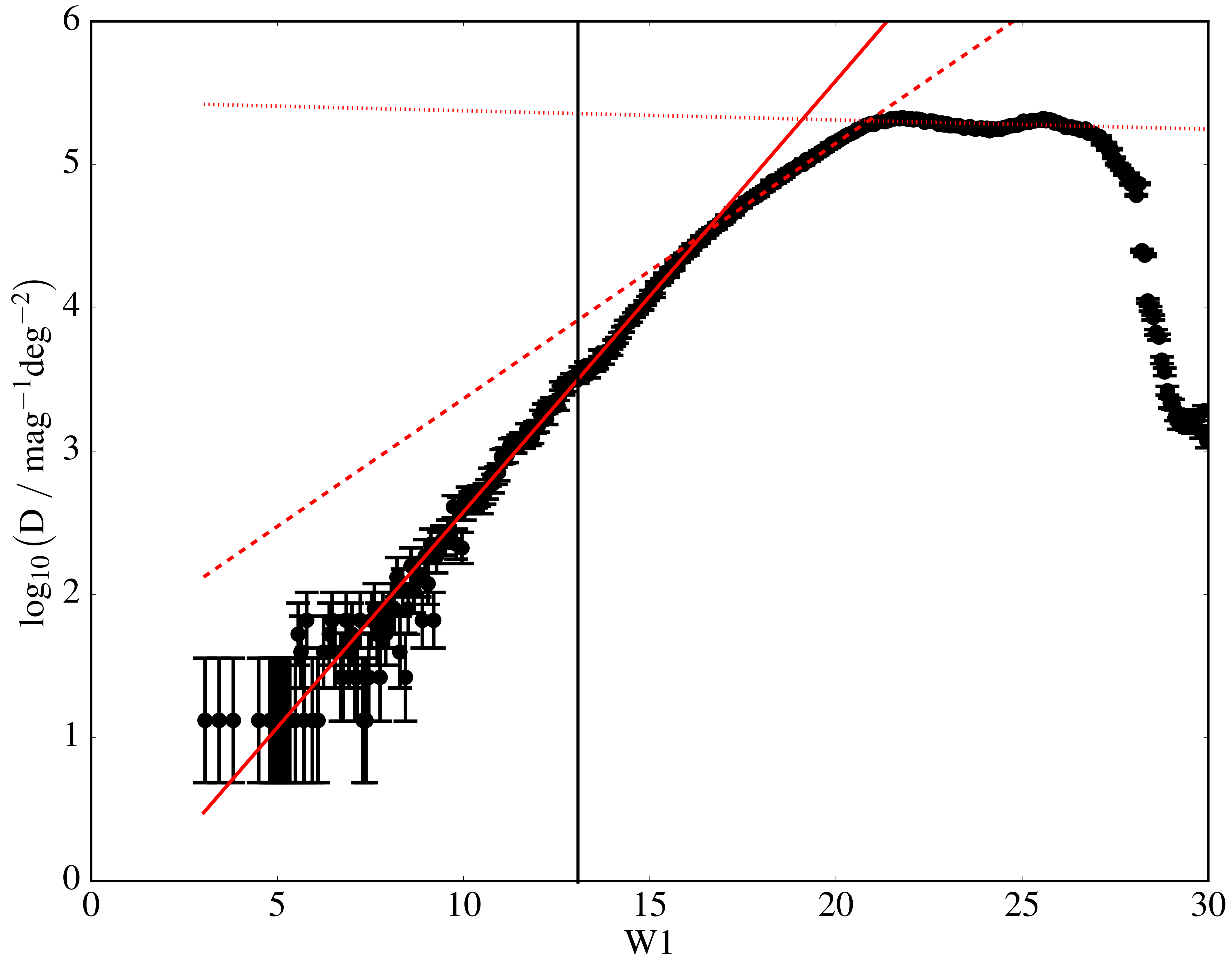
Tom J Wilson @onoddil

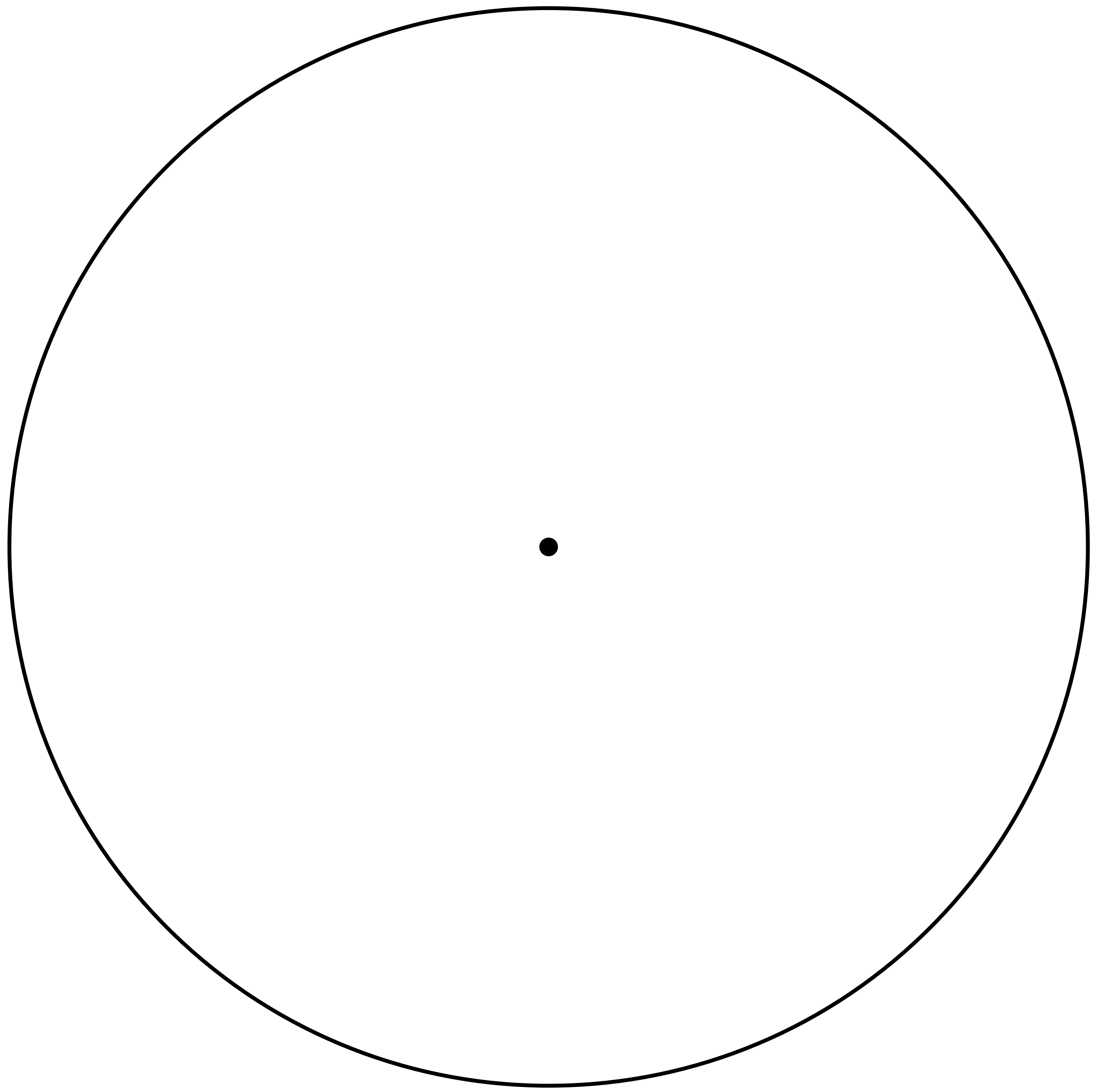
Building Empirical AUFs





PSF radius ~ 1.2 FWHM (Rayleigh criterion)



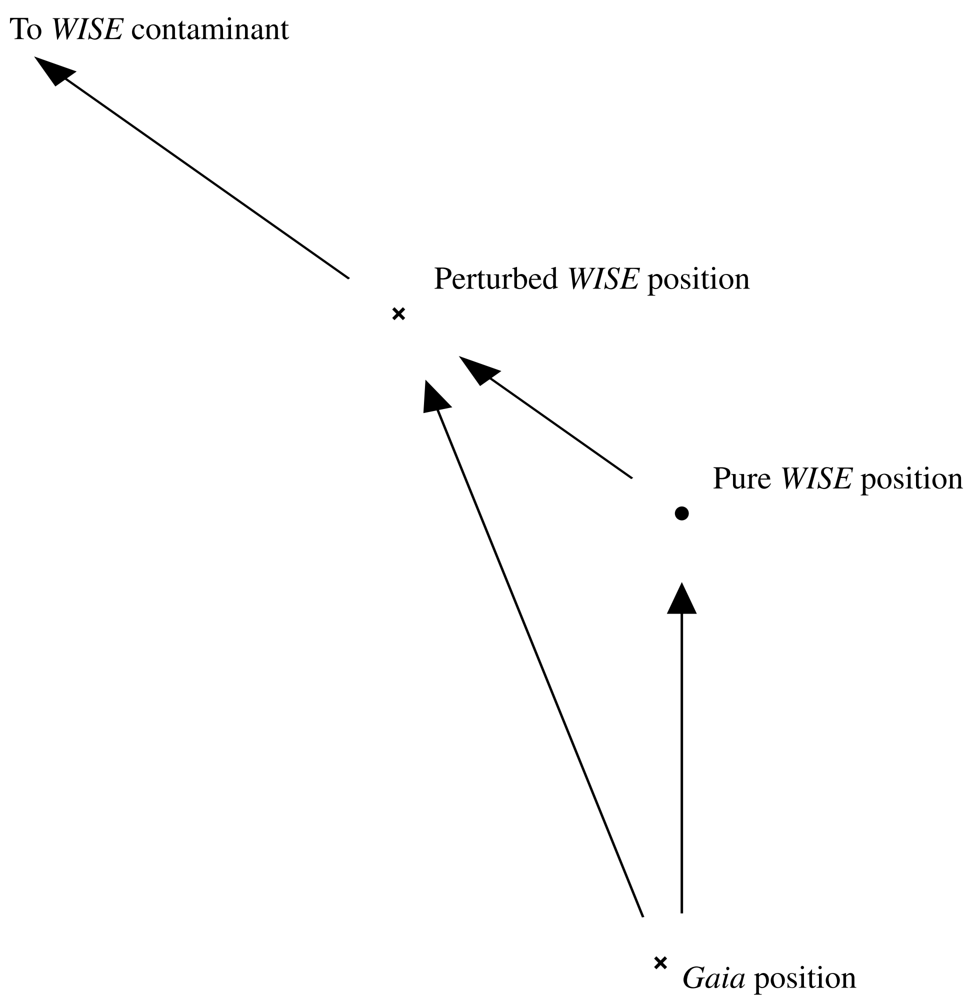


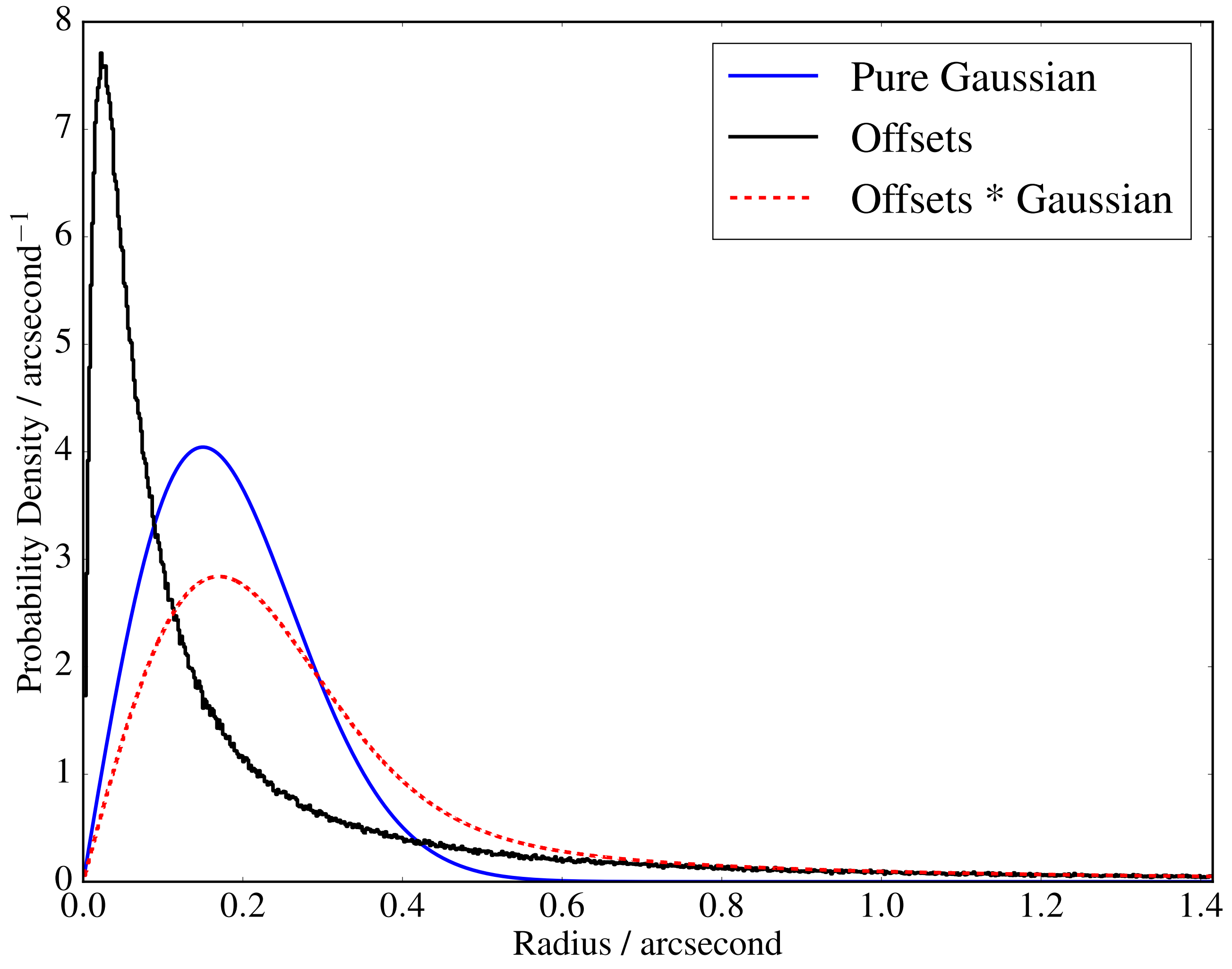
(sources per PSF circle ~ 10^-6 sources per mag per sq deg)

Wilson & Naylor (2018b)
TRILEGAL - Girardi et al. (2005)

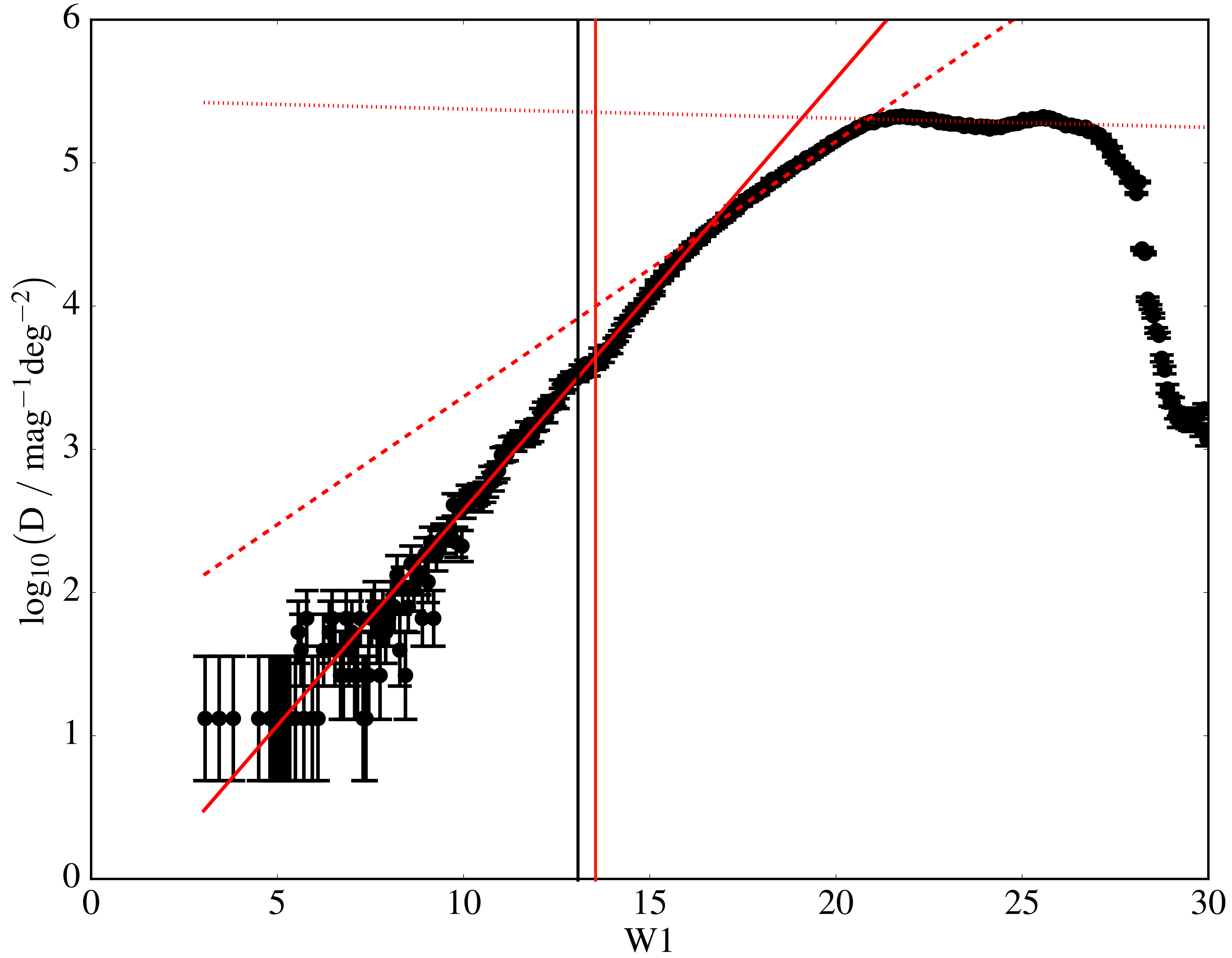
Tom J Wilson @onoddil

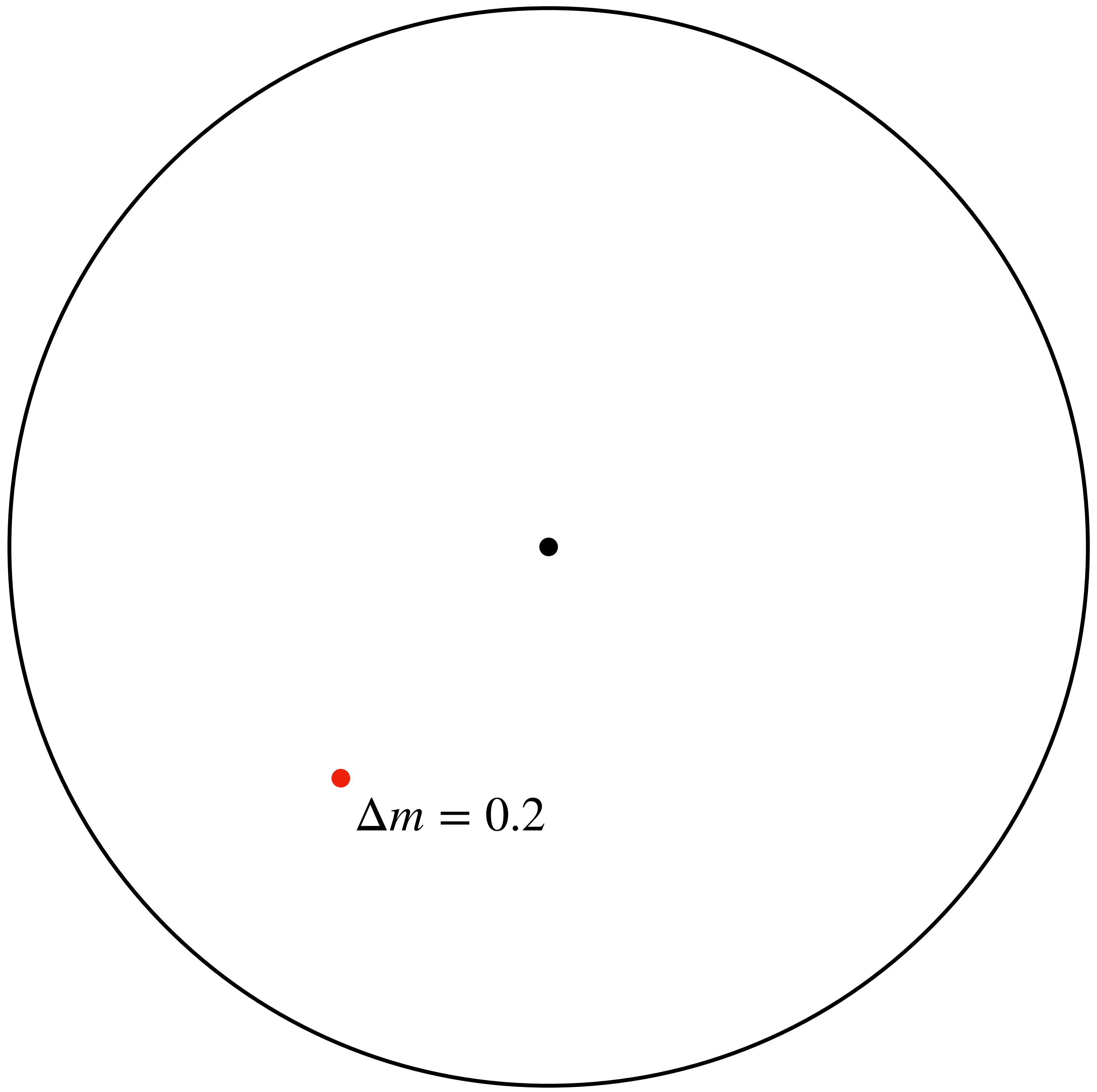
Building Empirical AUFs





PSF radius ~ 1.2 FWHM (Rayleigh criterion)



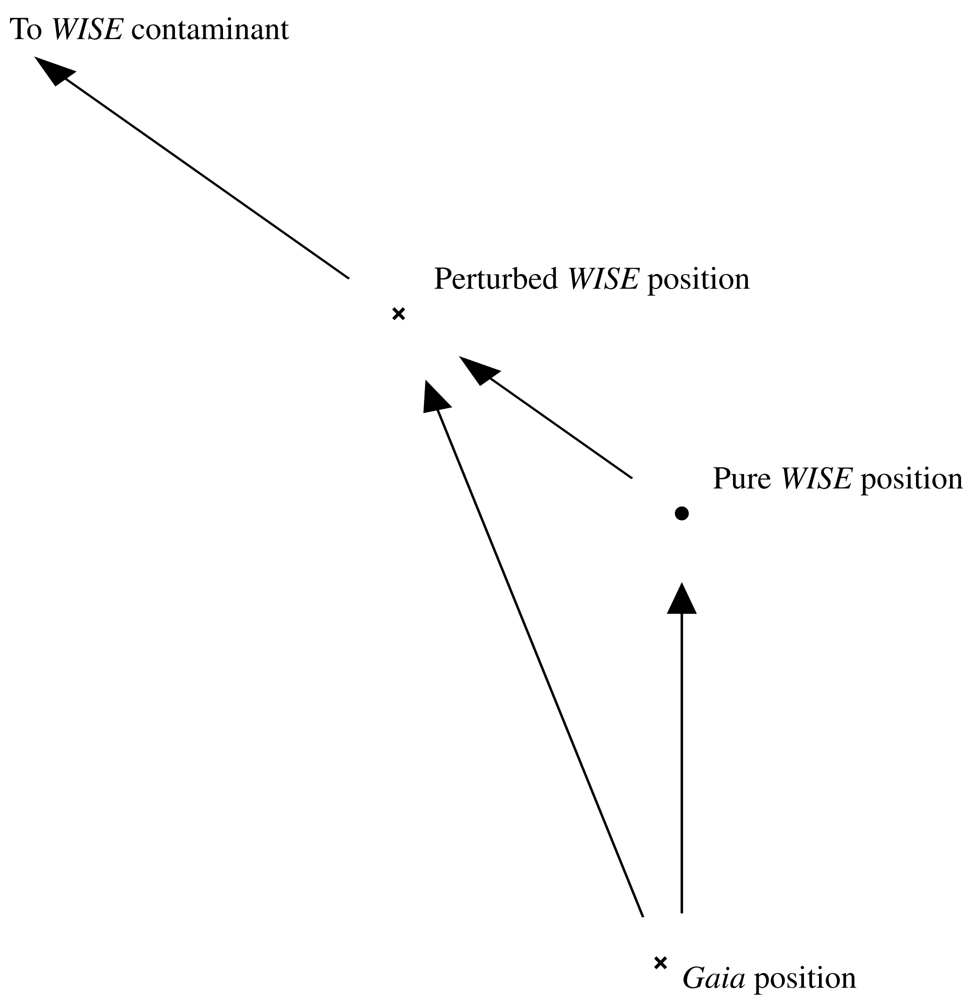


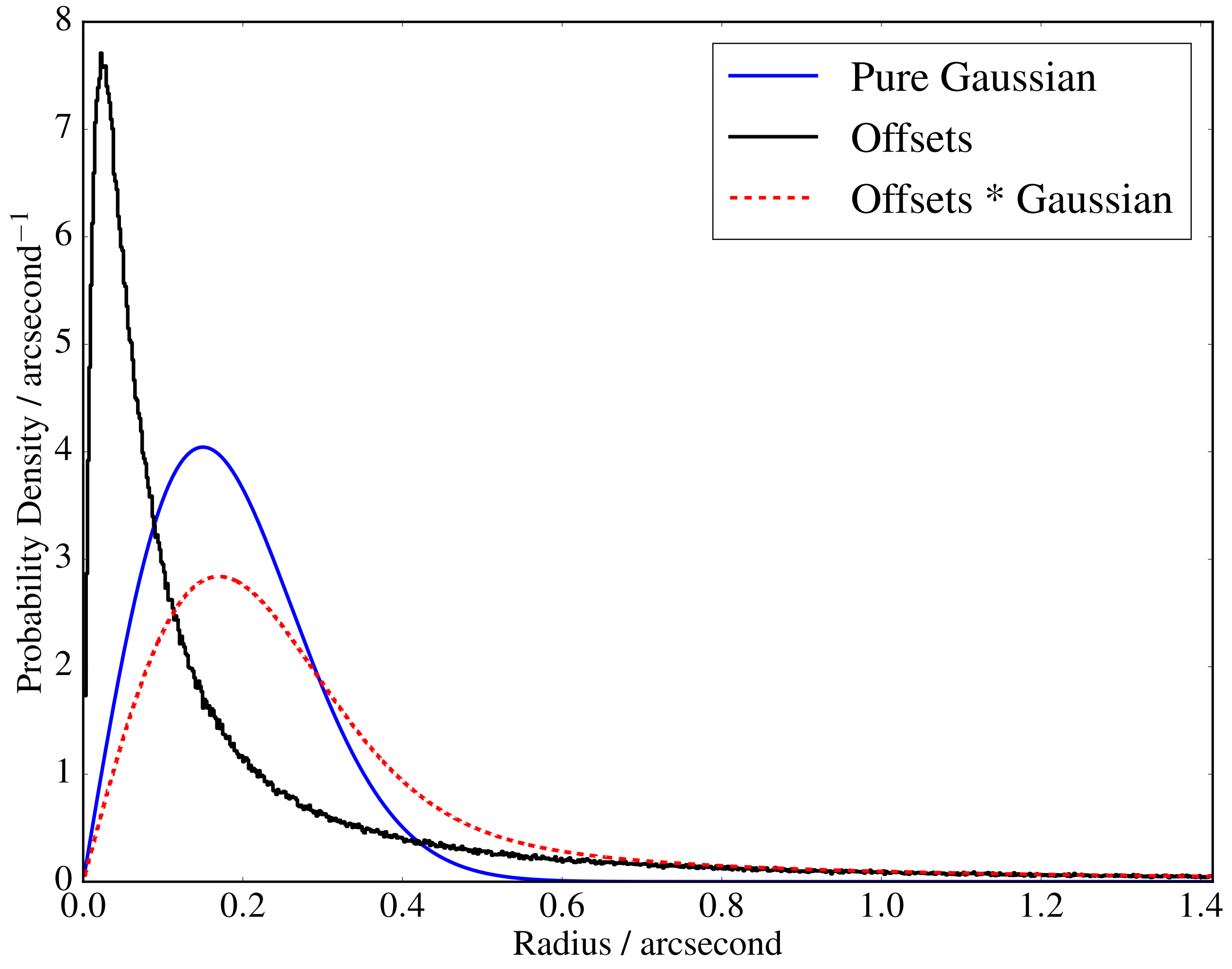
(sources per PSF circle ~ 10^-6 sources per mag per sq deg)

Wilson & Naylor (2018b)
TRILEGAL - Girardi et al. (2005)

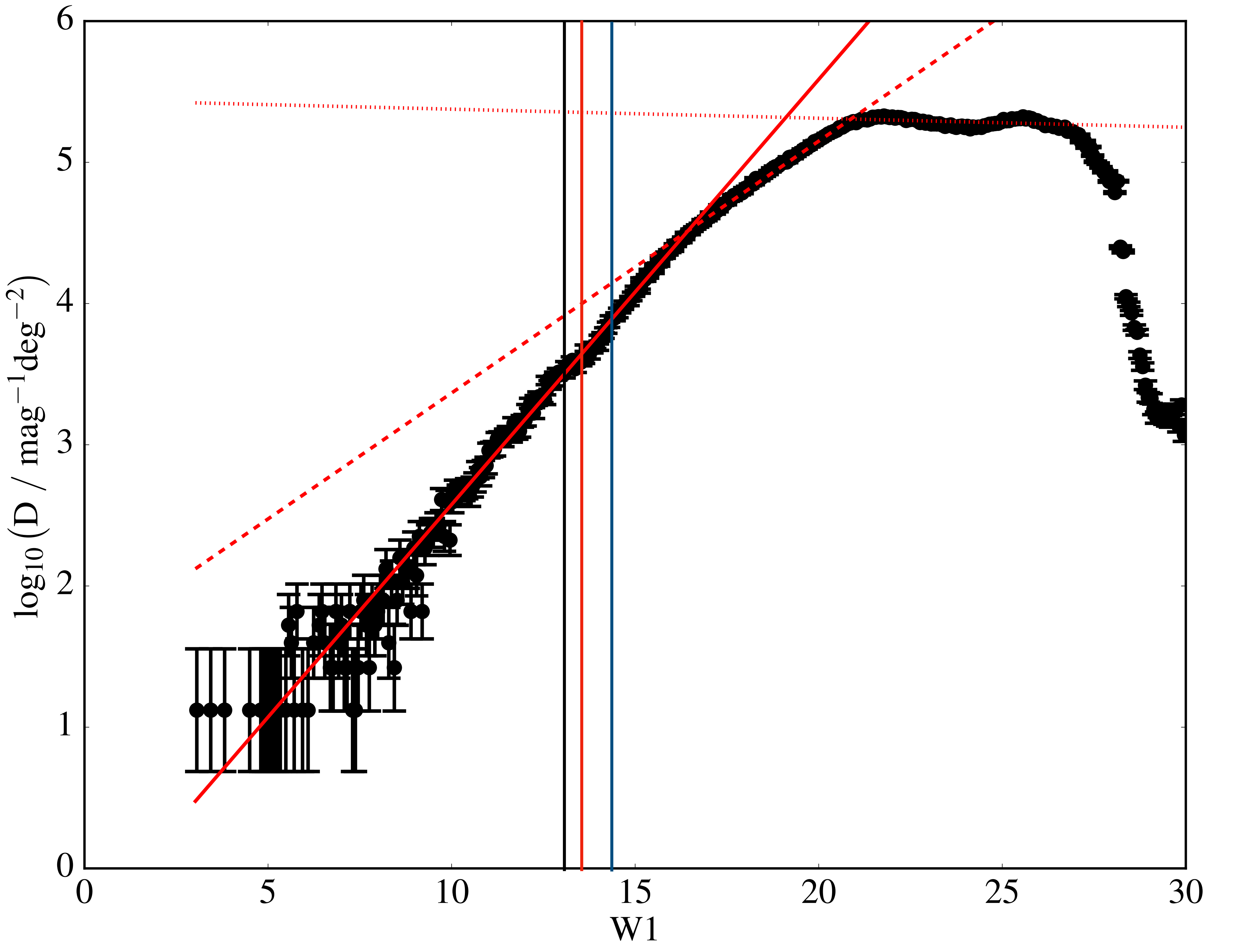
Tom J Wilson @onoddil

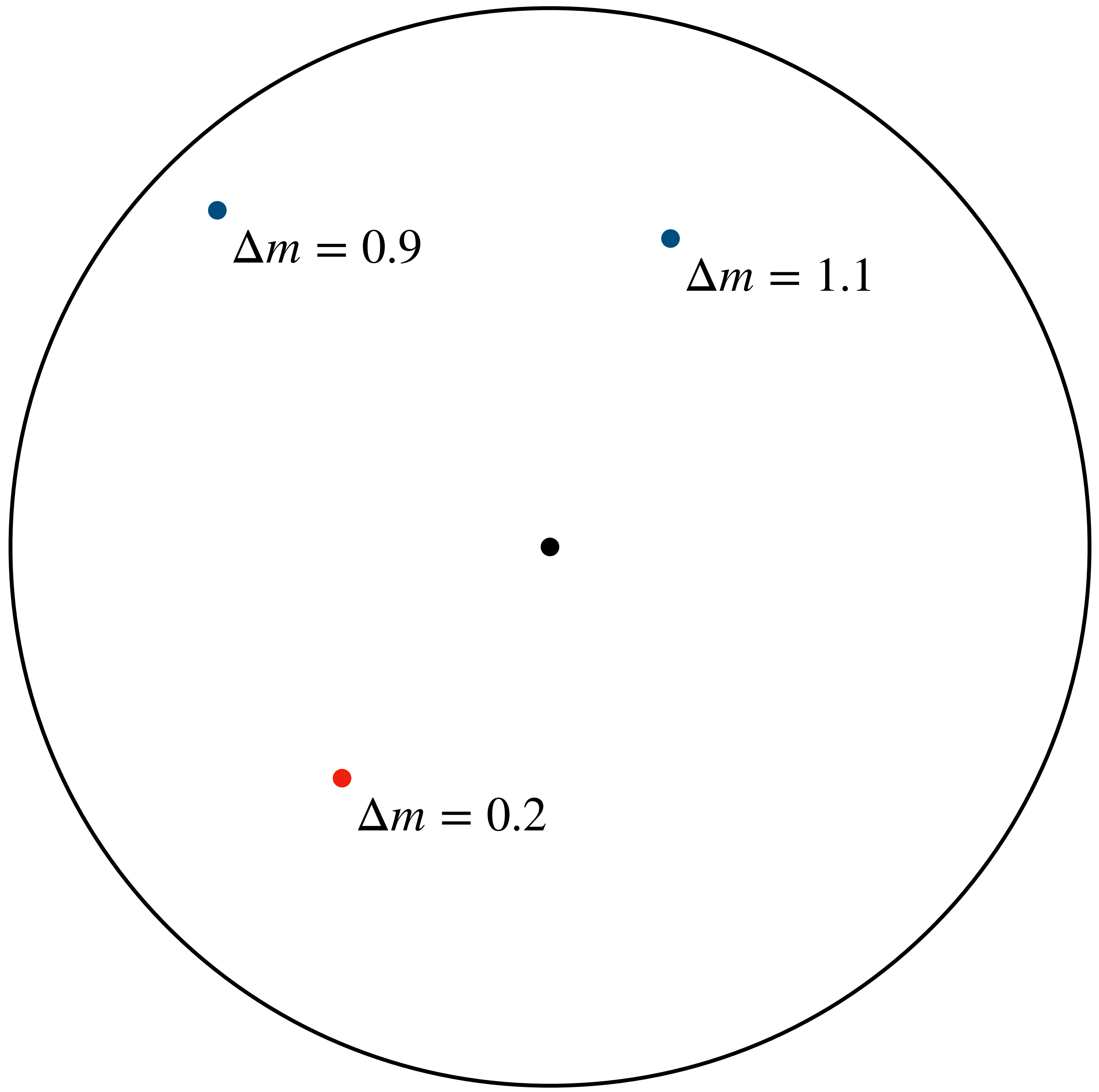
Building Empirical AUFs





PSF radius ~ 1.2 FWHM (Rayleigh criterion)



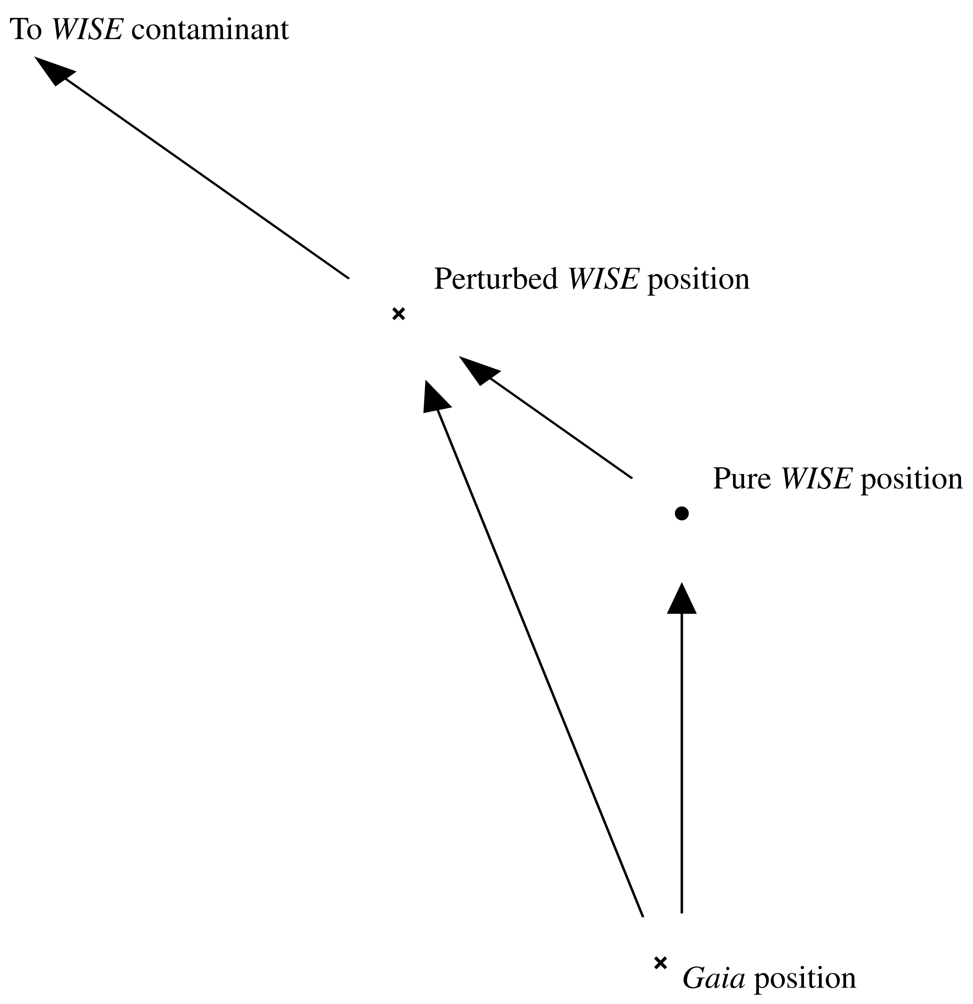


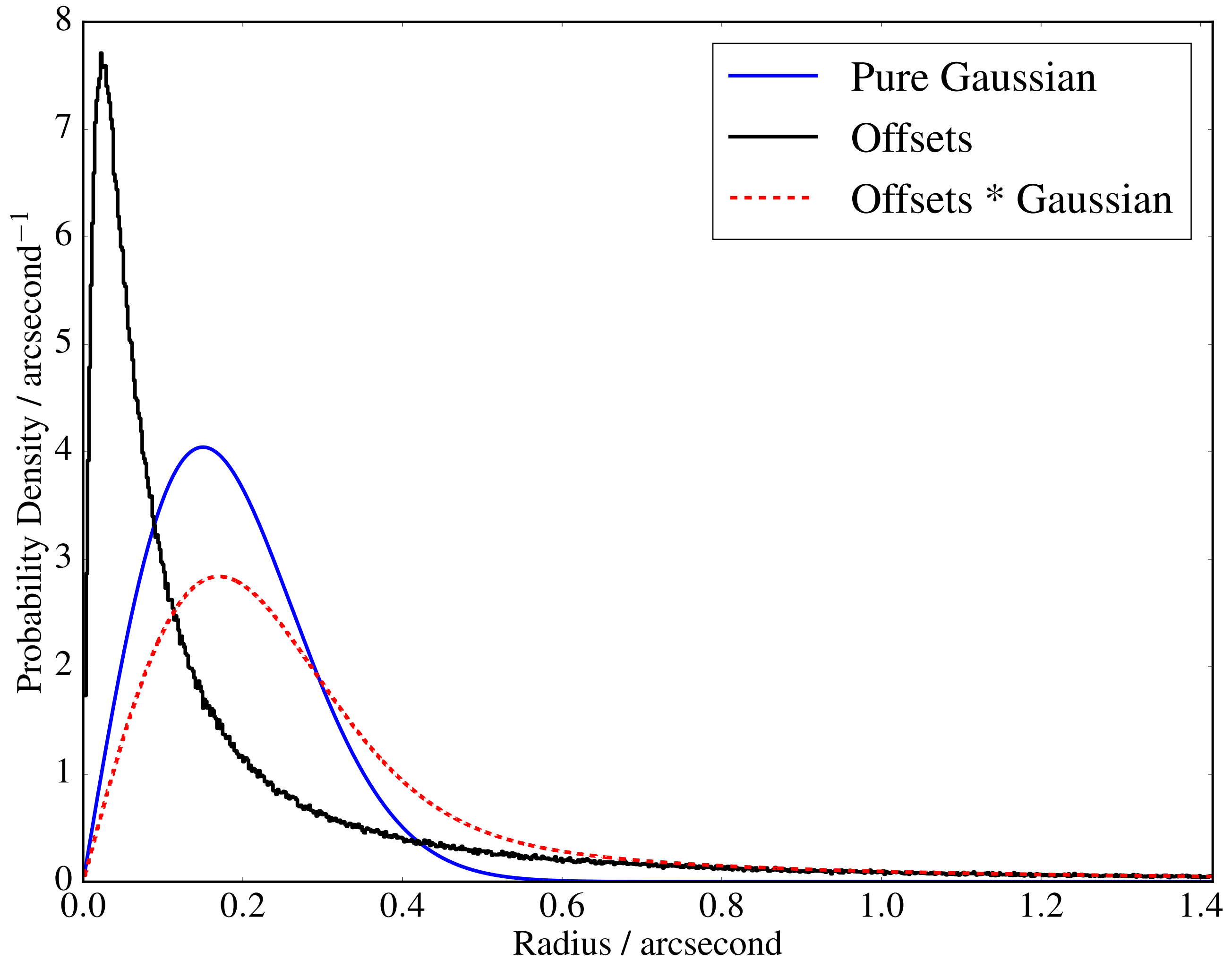
(sources per PSF circle ~ 10^-6 sources per mag per sq deg)

Wilson & Naylor (2018b)
TRILEGAL - Girardi et al. (2005)

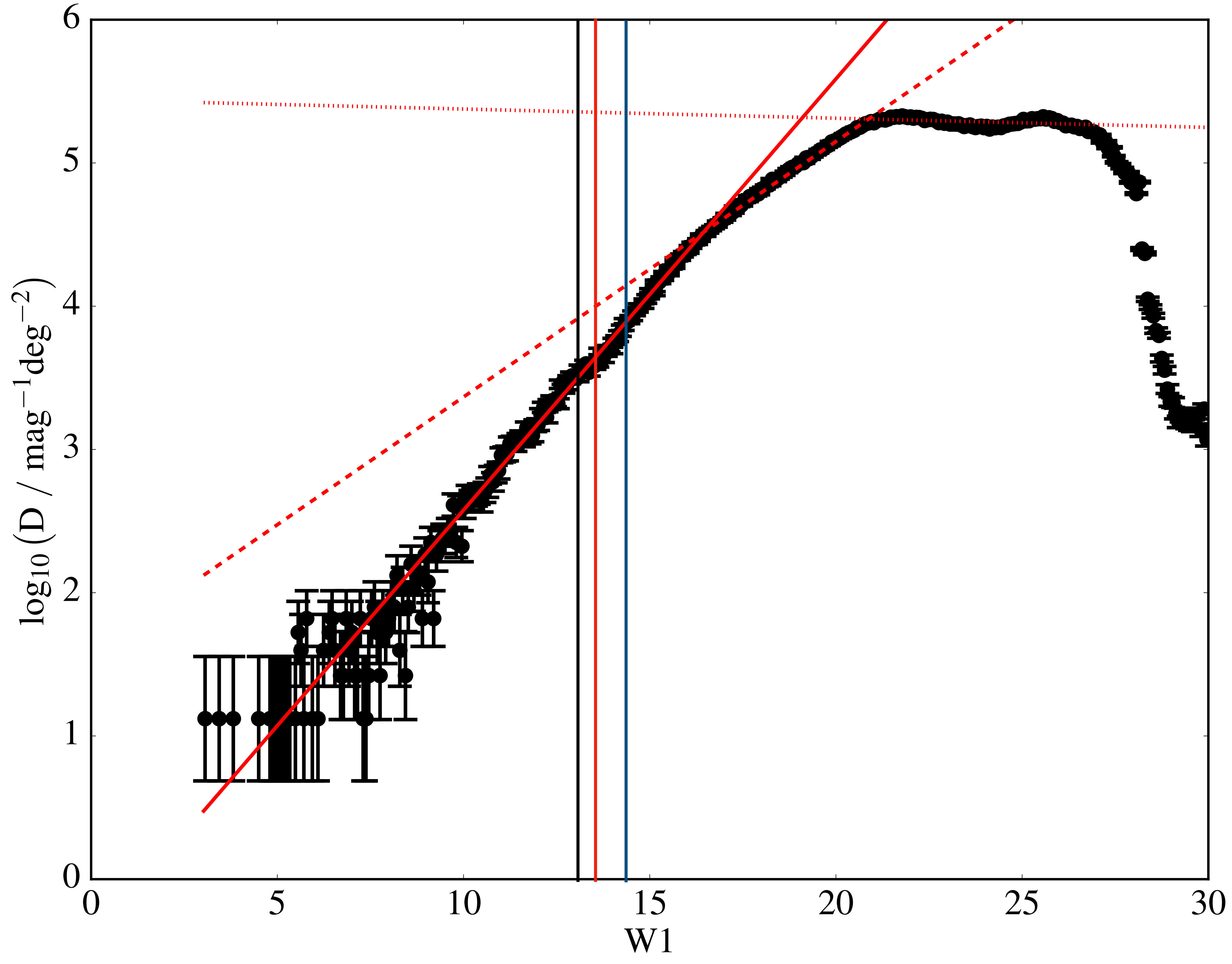
Tom J Wilson @onoddil

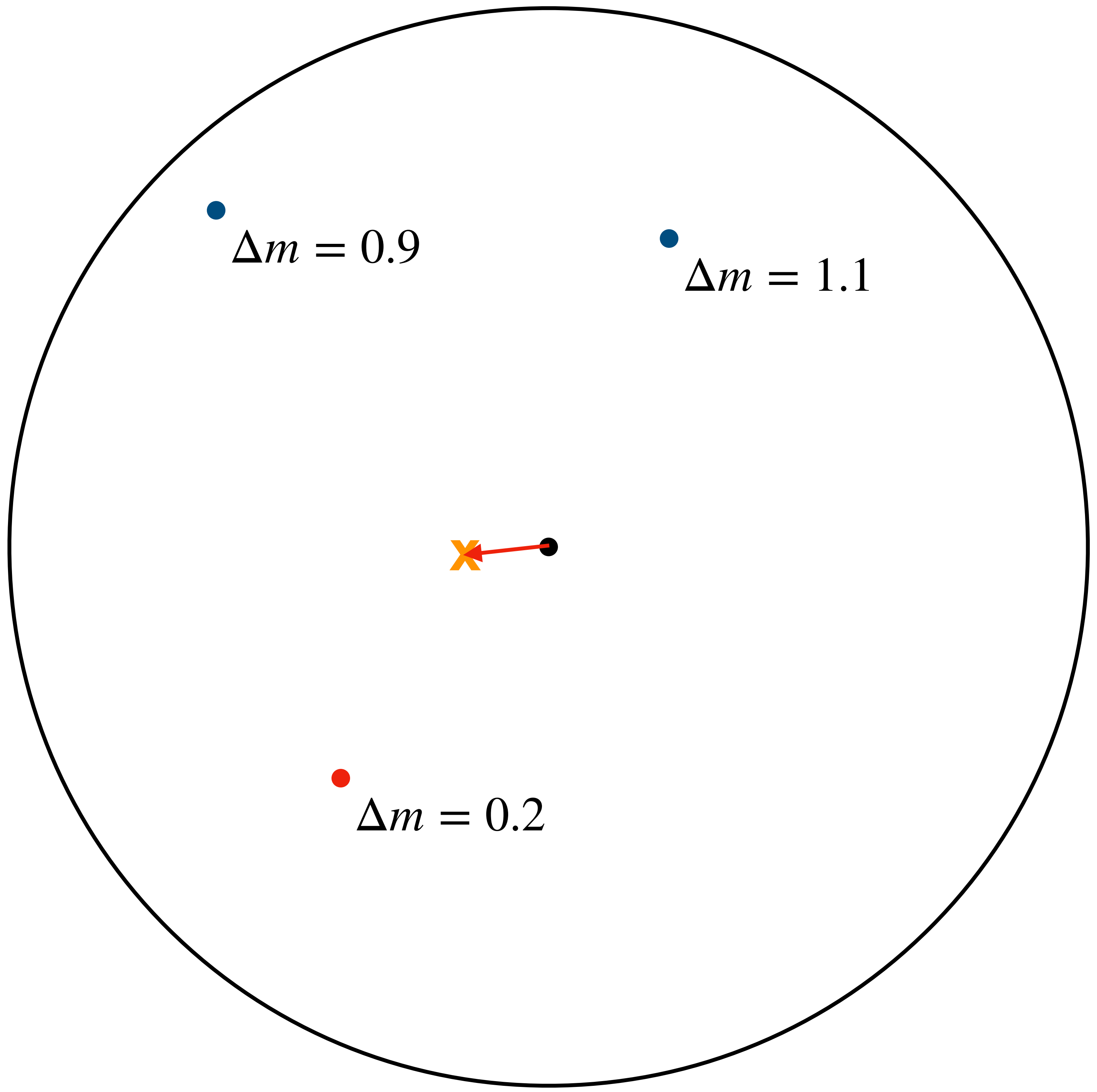
Building Empirical AUFs





PSF radius ~ 1.2 FWHM (Rayleigh criterion)





(sources per PSF circle ~ 10^-6 sources per mag per sq deg)

Wilson & Naylor (2018b)
TRILEGAL - Girardi et al. (2005)

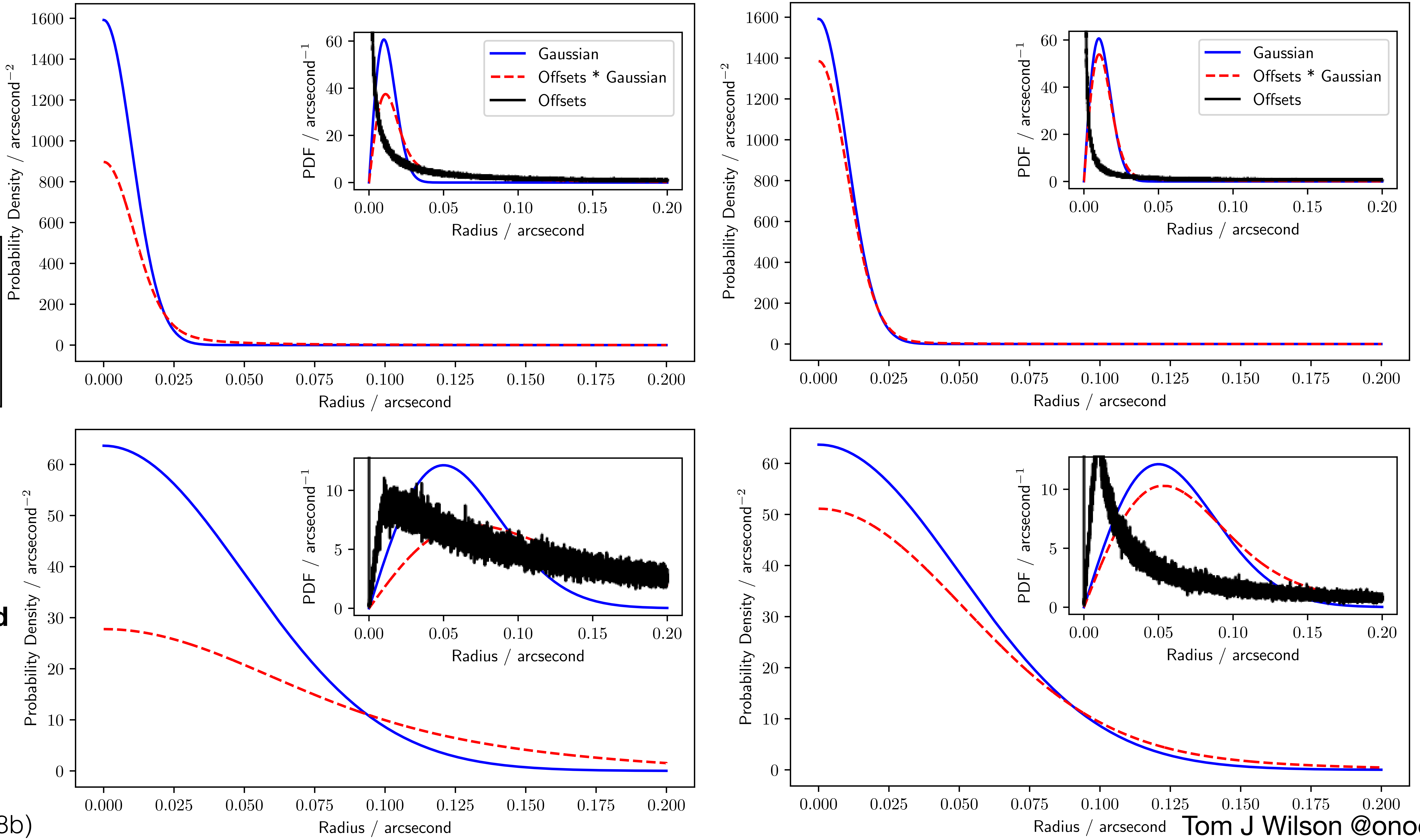
Tom J Wilson @onoddil

The Rubin AUF: Galactic Plane

Galactic Centre | **Not the Galactic Centre**

Single-visit

Co-add

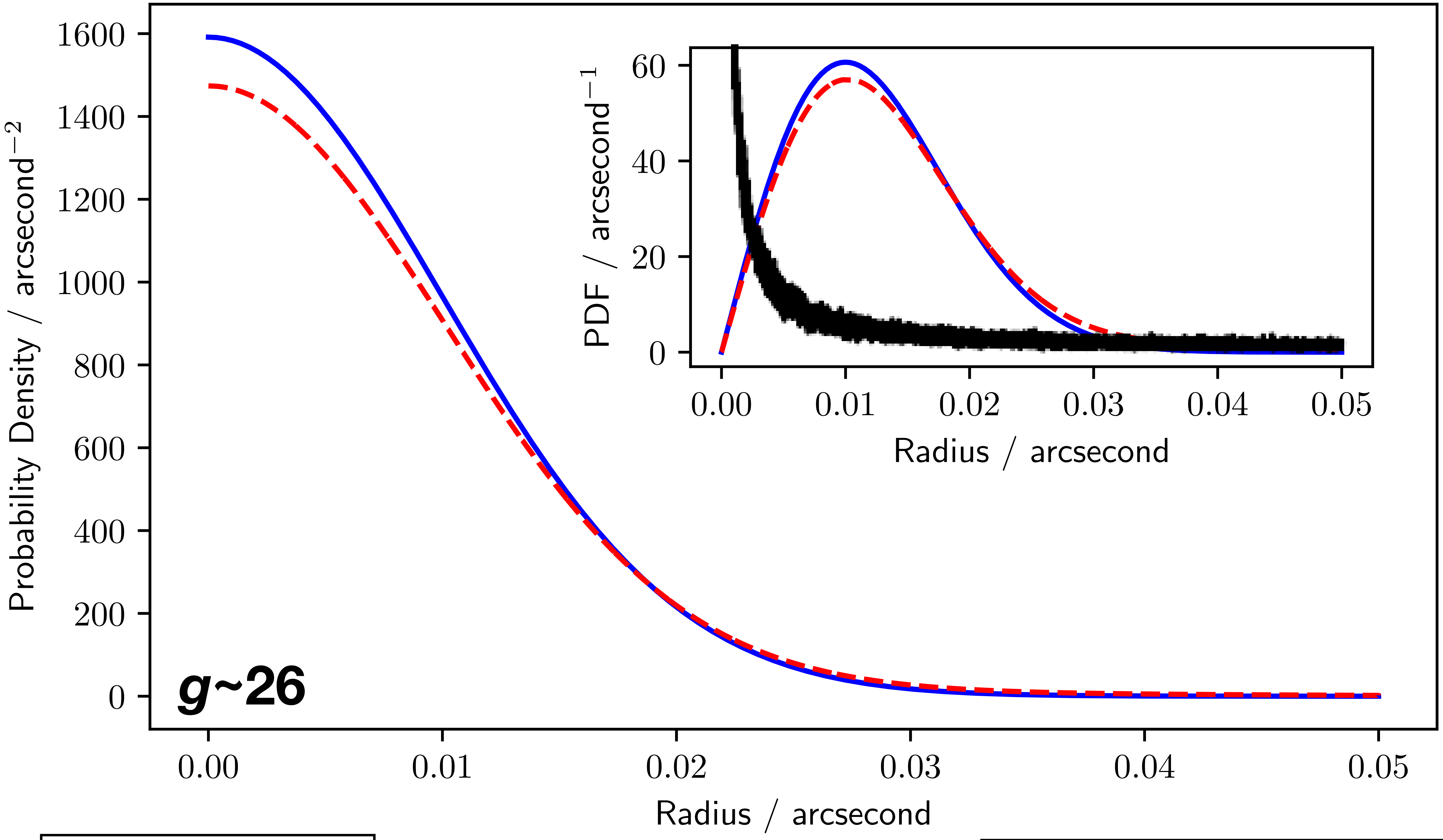


Without modelling this extra effect, we fail to recover many true pairings, with an artificially high false negative rate!

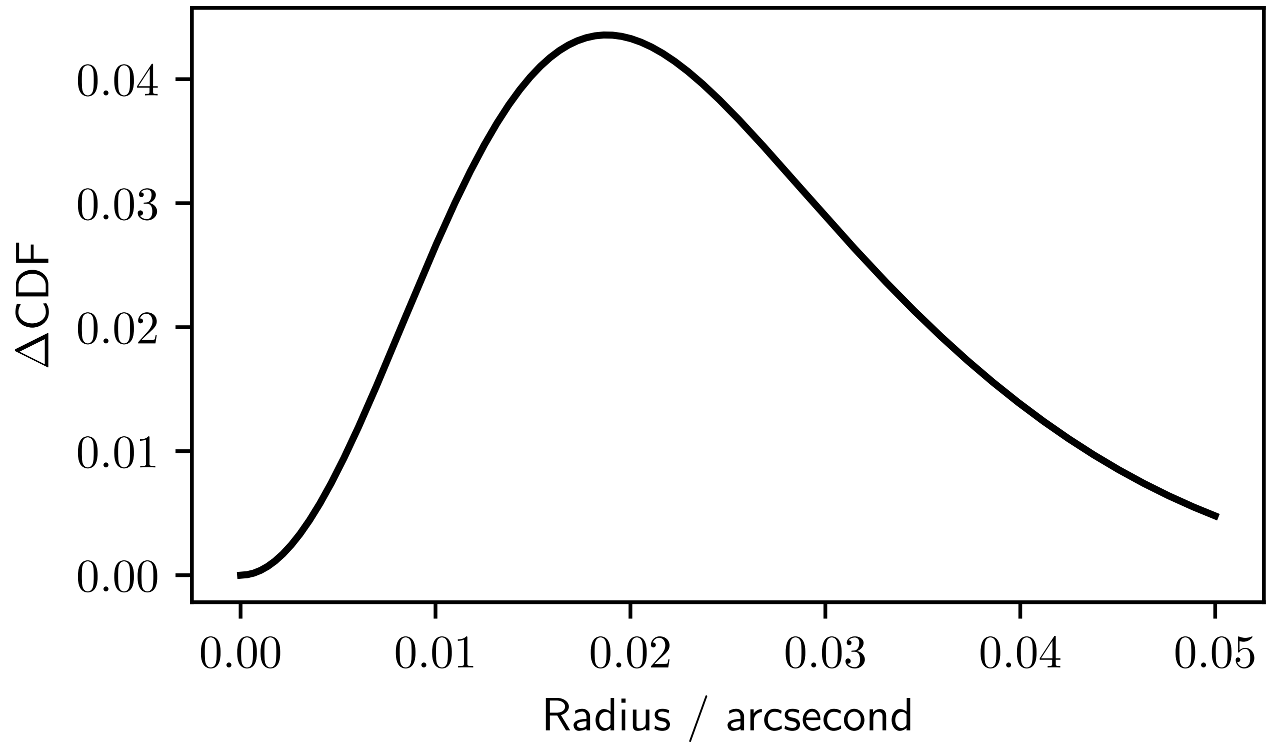
Wilson & Naylor (2018b)

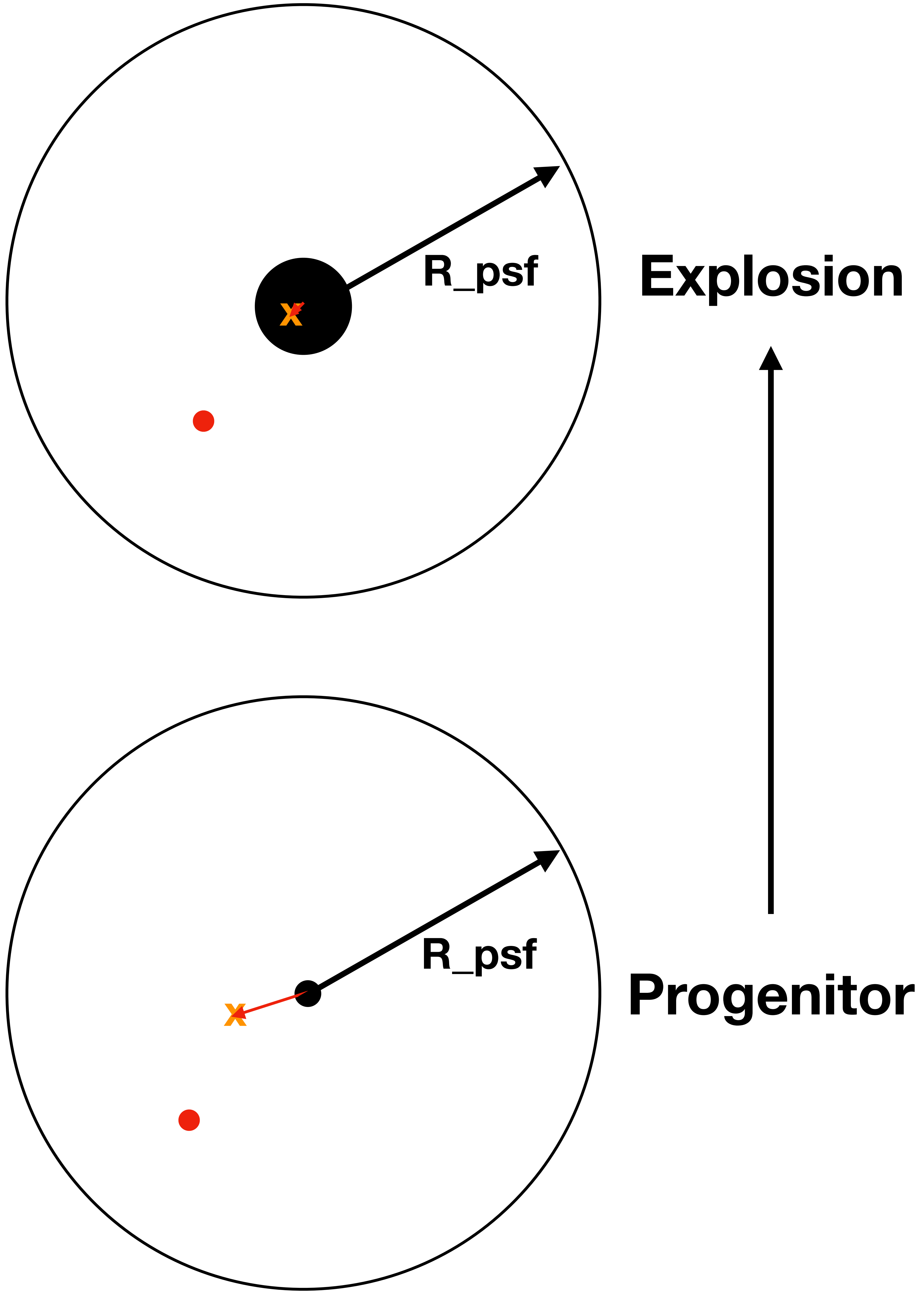
Tom J Wilson @onoddil

The Rubin AUF: Extra-Galactic, Transients



Without modelling this extra effect, we fail to recover many true pairings, with an artificially high false negative rate!

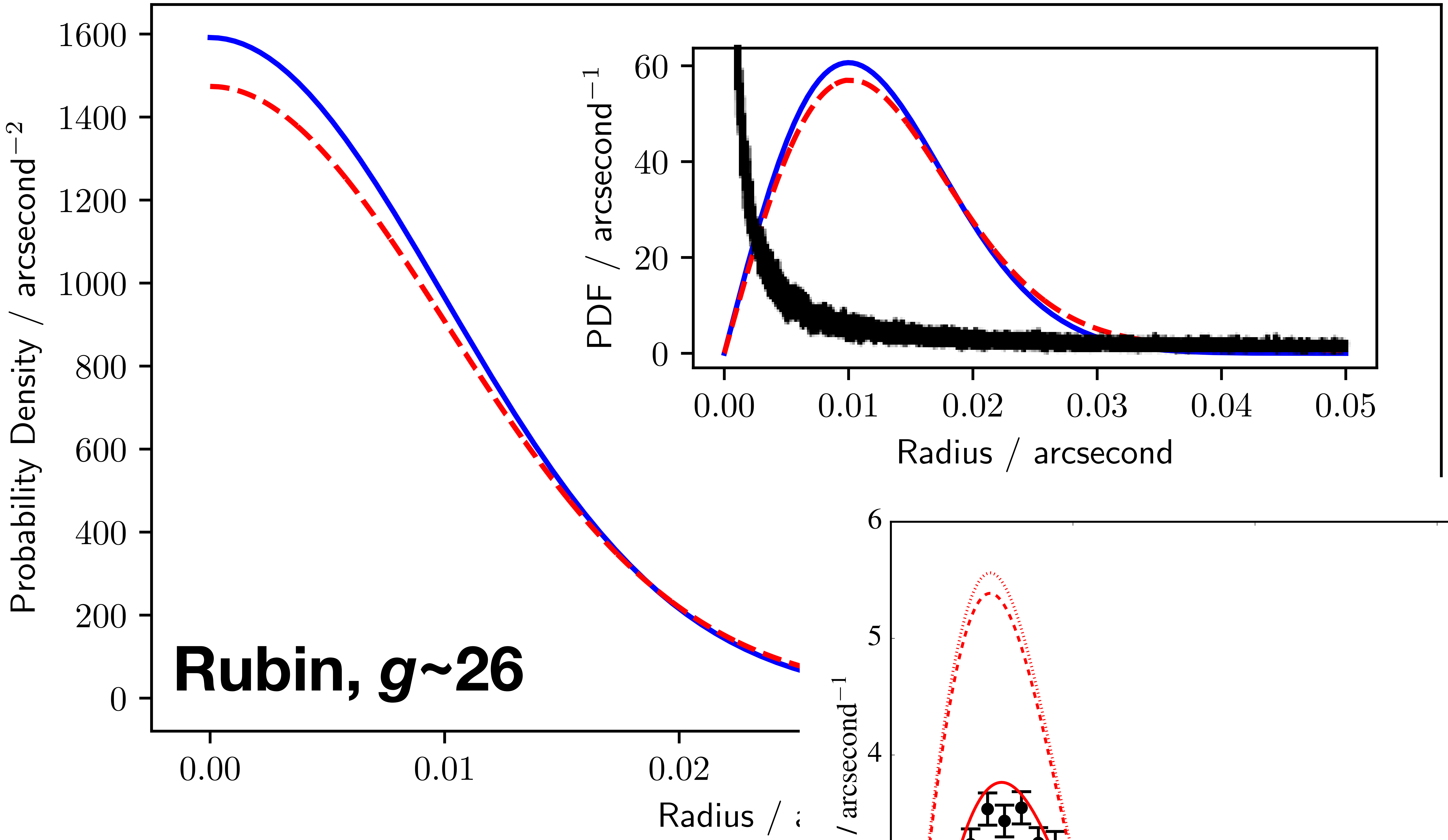


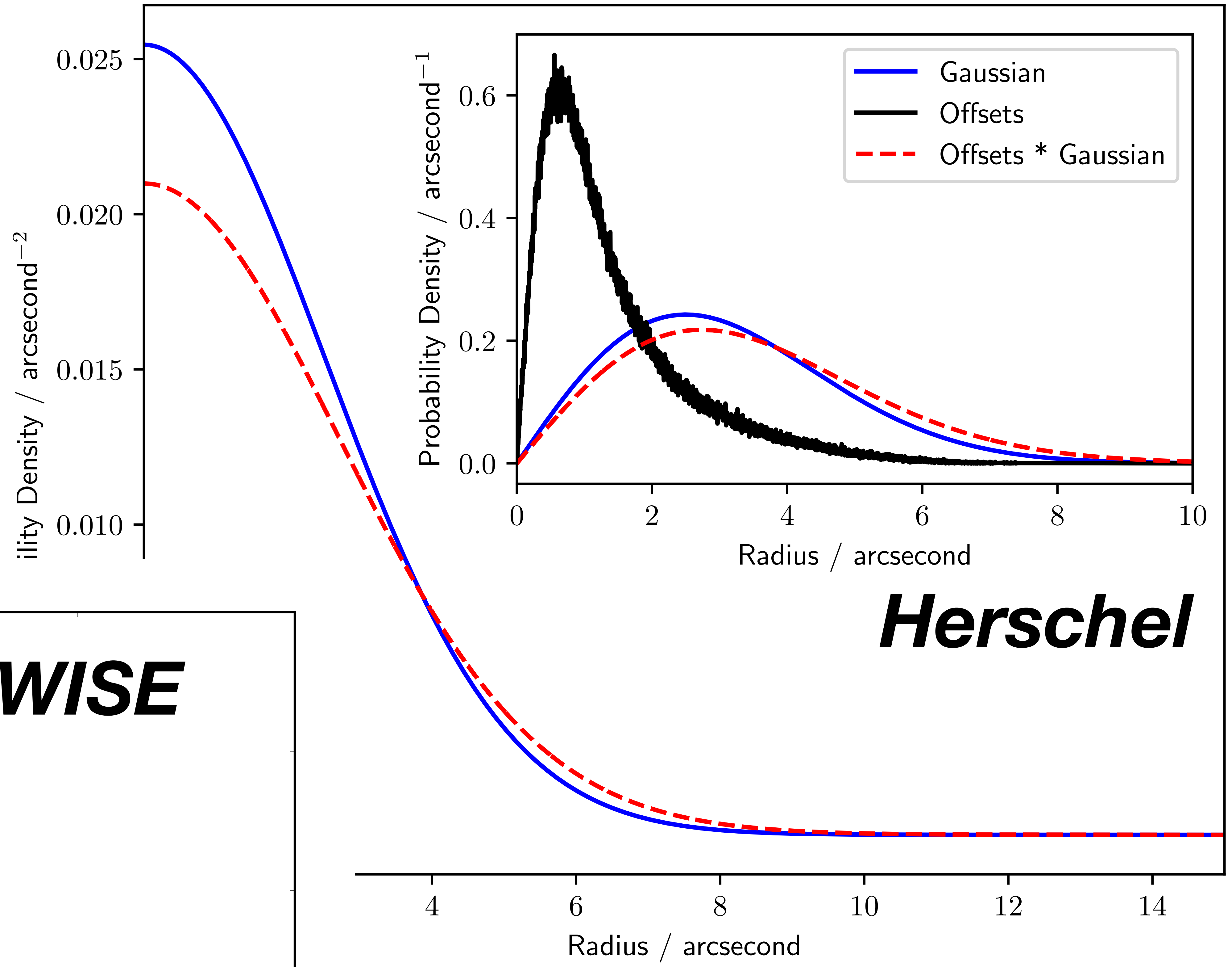


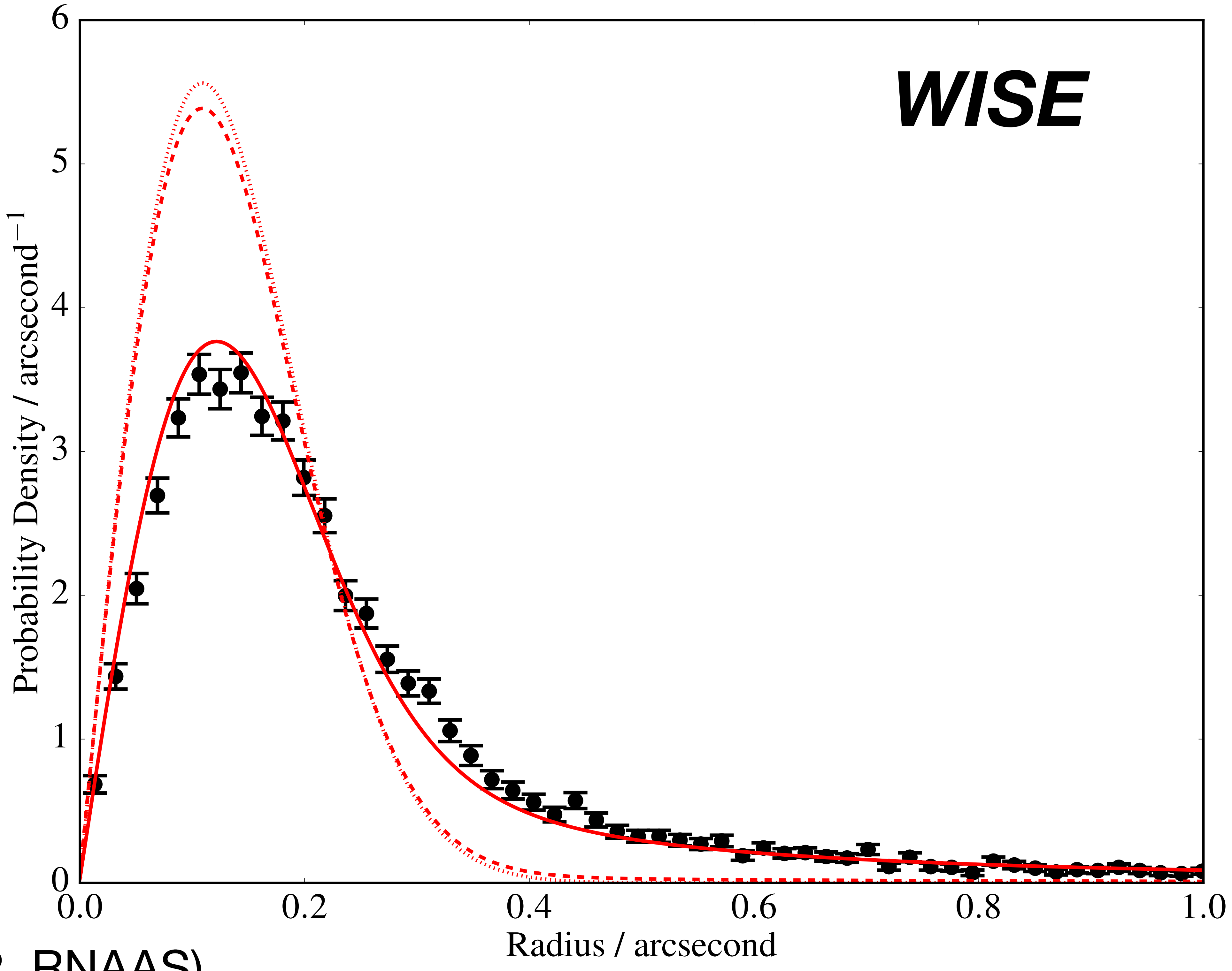
Wilson & Naylor (2018b); also see Wilson (2022, RNAAS)

Tom J Wilson @onoddil

The Rubin AUF: Extra-Galactic, Transients







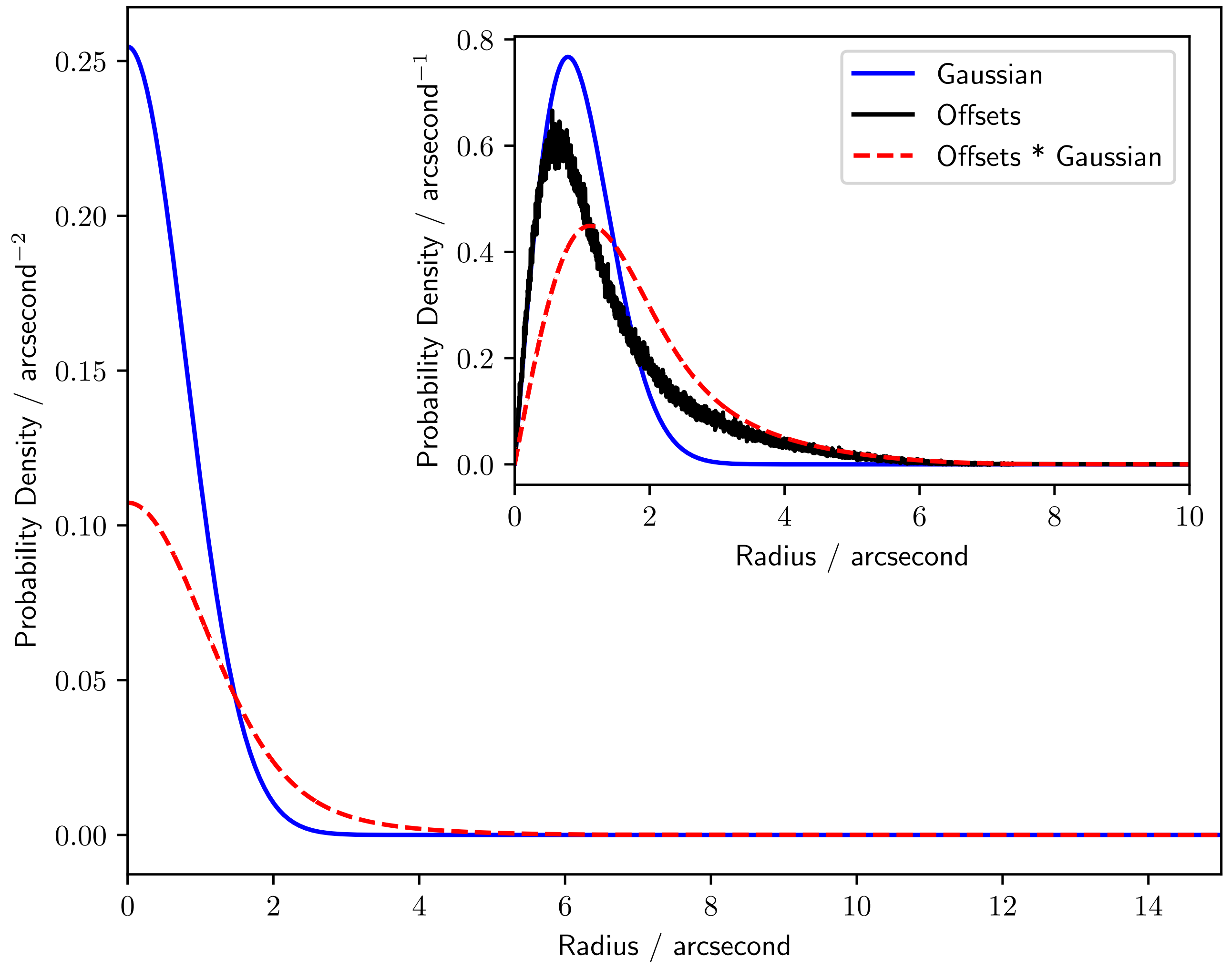
Without modelling this extra effect, we fail to recover many true pairings, with an artificially high false negative rate!

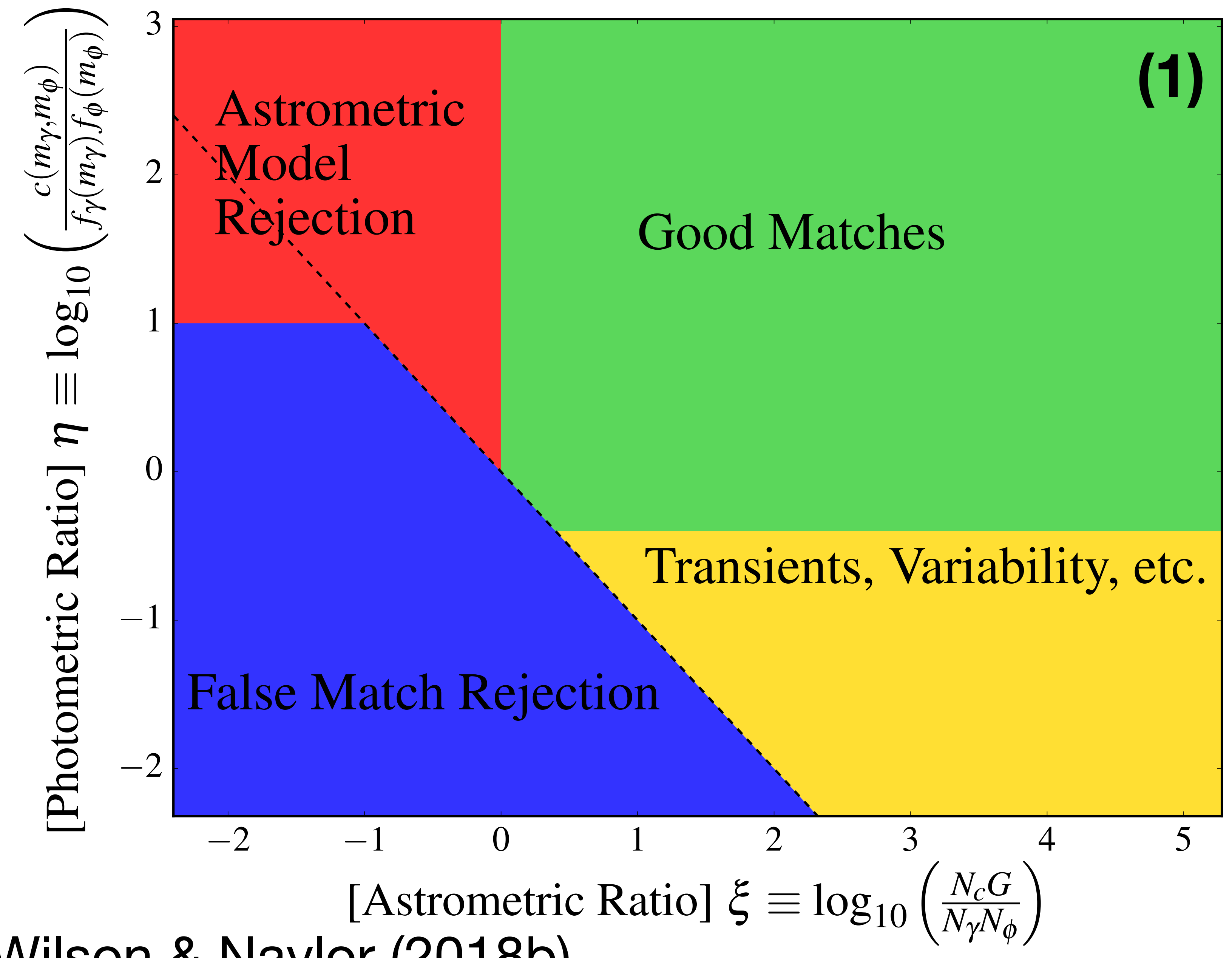
Wilson & Naylor (2018b); also see Wilson (2022, RNAAS)

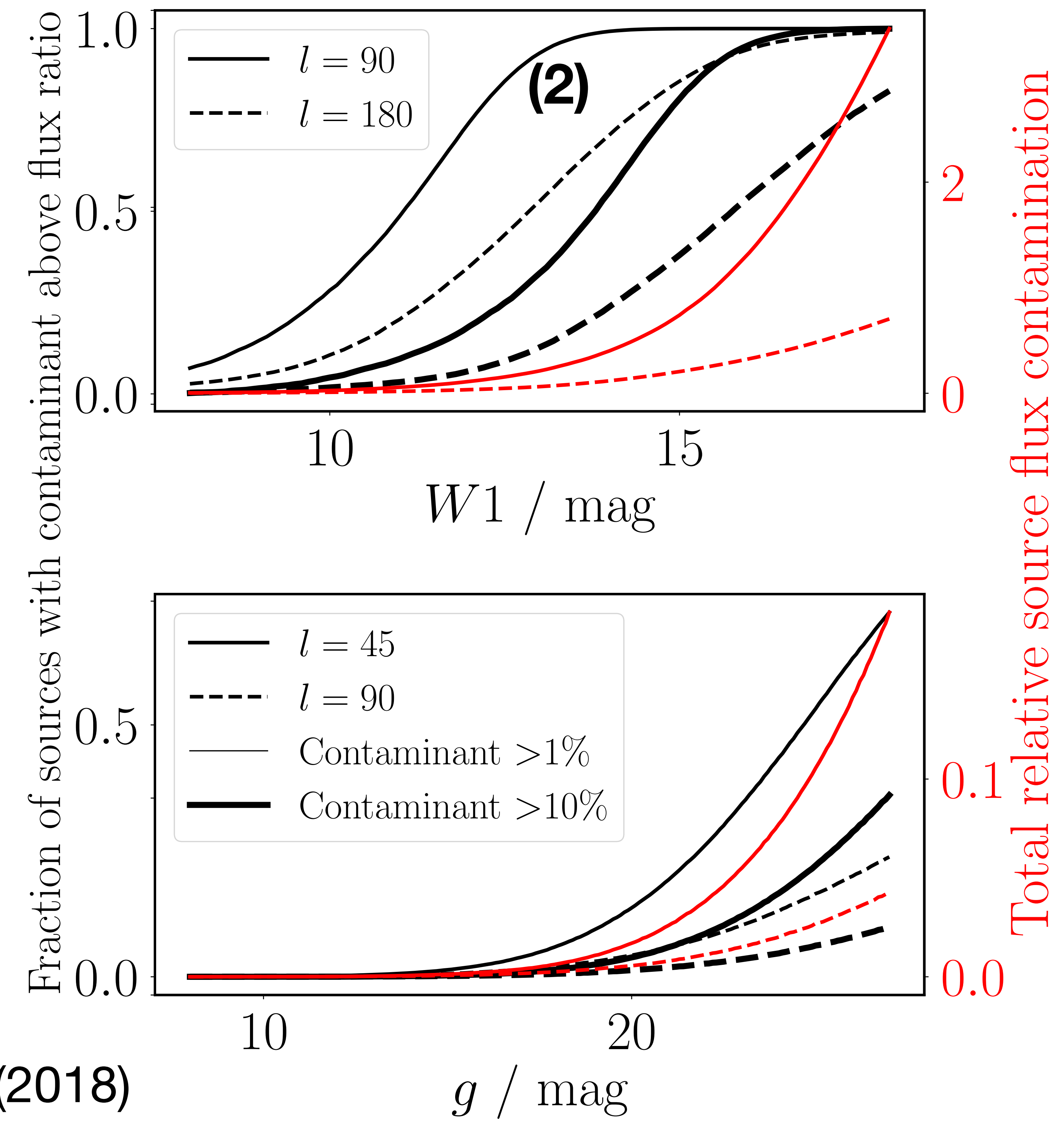
Tom J Wilson @onoddil

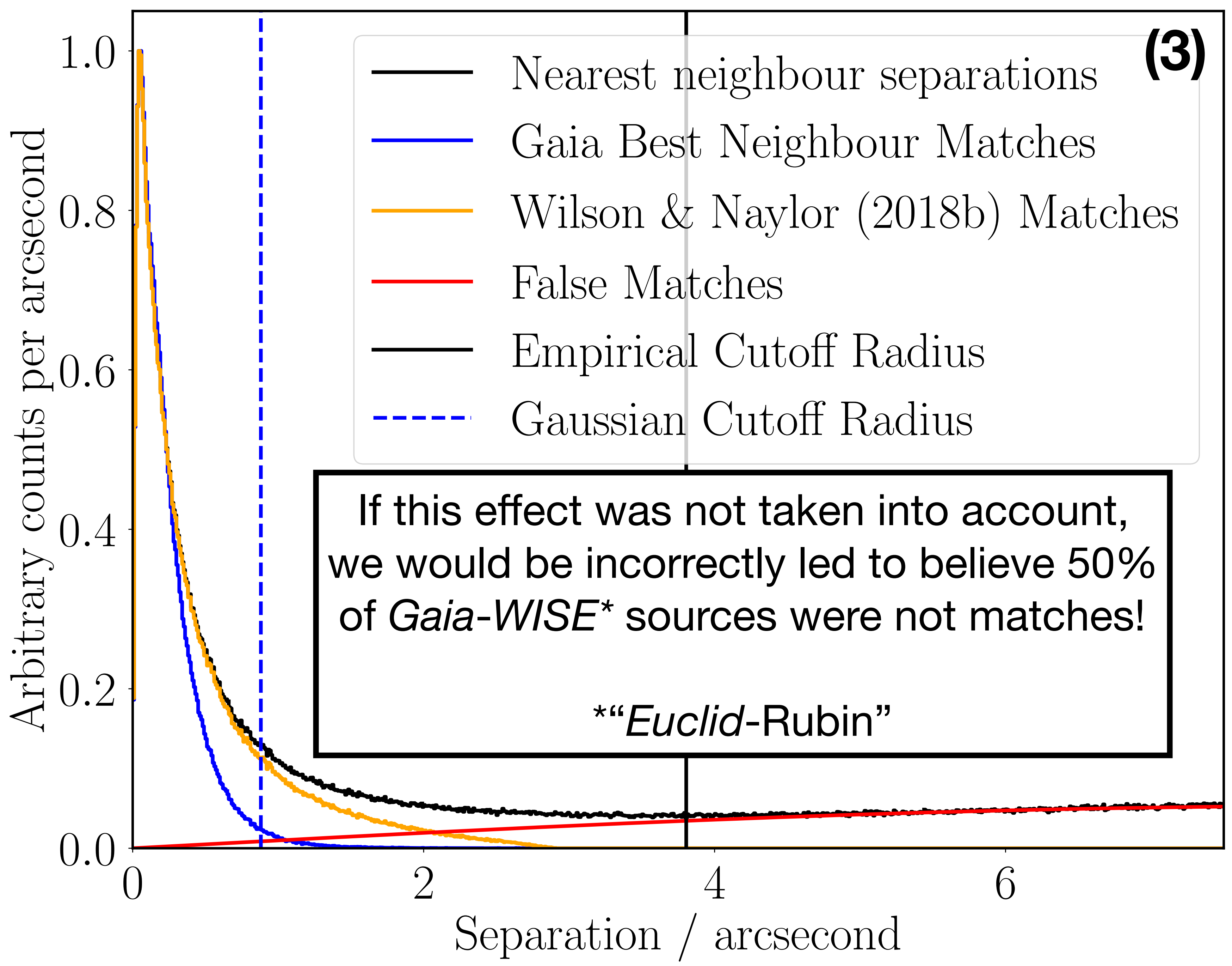
Why Use Macauff's Cross-Matches?

0) Getting cross-matches, even for "well behaved" fields
1) Finding "odd" objects, either using the inclusion vs non-inclusion of the photometry in the two match runs, or via the likelihood ratio space — separately-planned "real time" matching service for transient objects
2) Removing e.g. IR excess or correcting for extinction-like crowding brightening, through Average Contamination; crucial for "1% photometry" in both precision *and* accuracy
3) Recovering additional sources missed by other match services — either in crowded fields (we recover up to twice as many *Gaia-WISE* matches than the *Gaia* best neighbour matches), or with our extension to unknown proper motion modelling as an extra systematic









Wilson & Naylor (2018b)
WISE - Wright et al. (2010)
Gaia matches - Marrese et al. (2019)
Gaia DR2 - Gaia Collaboration, Brown A. G. A., et al. (2018)

Tom J Wilson @onoddil

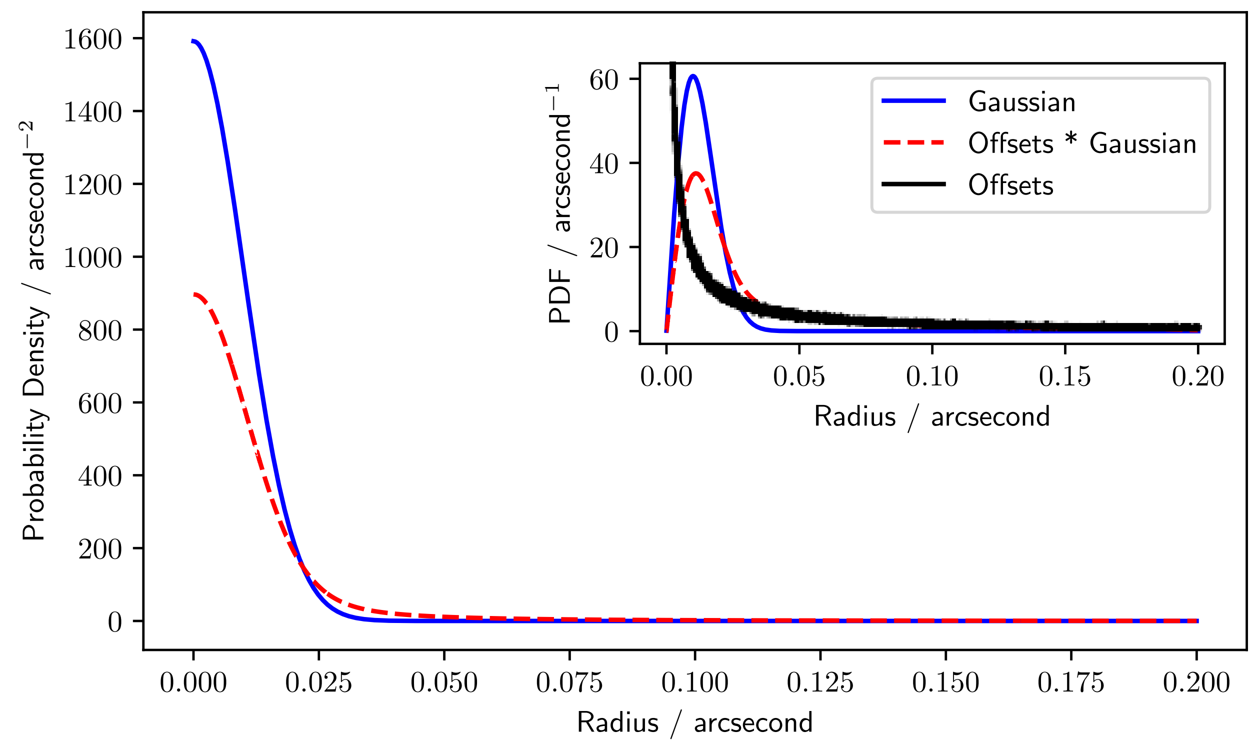
macauff: Cross-Matching in the Crowded LSST Sky

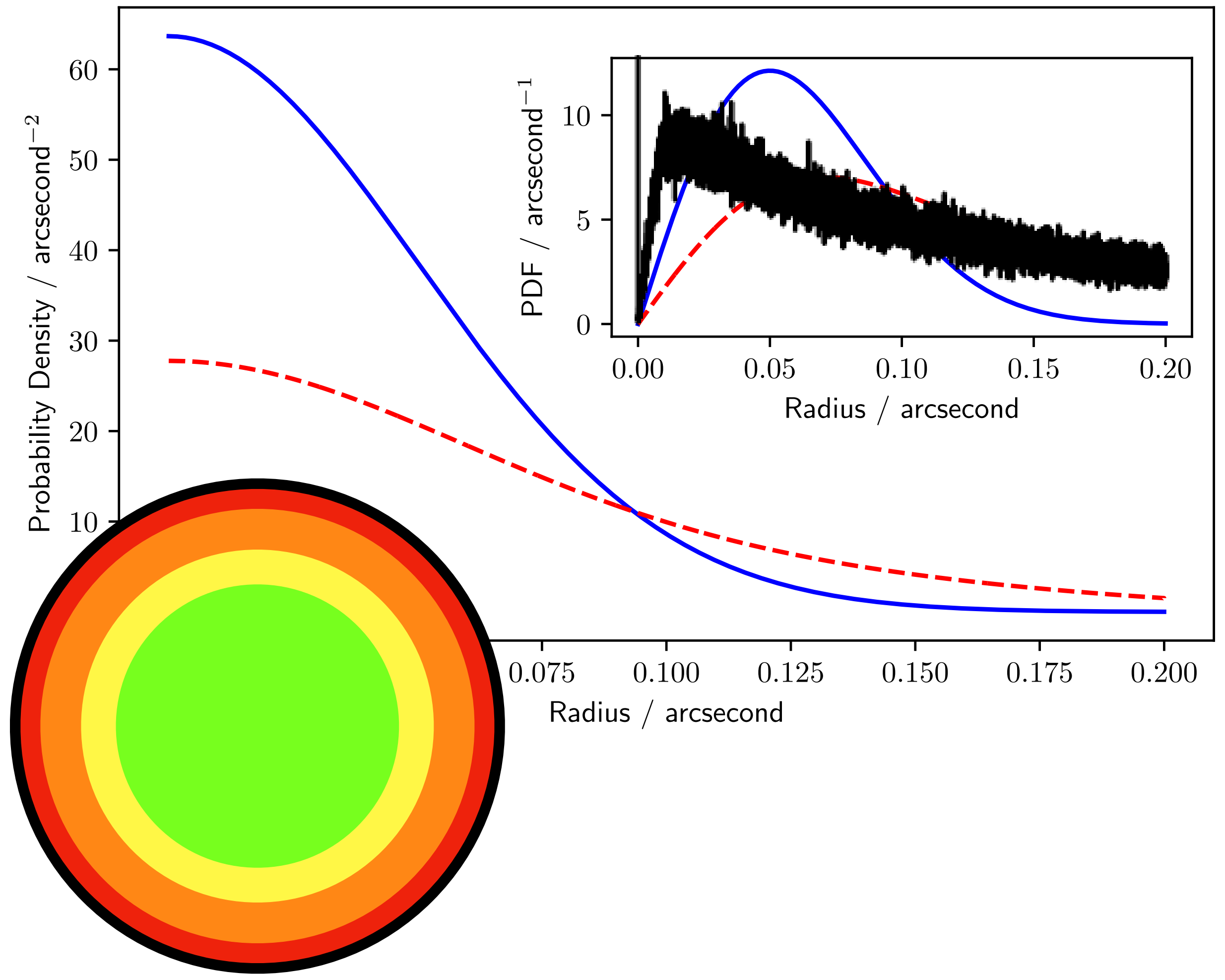
- **Our cross-match algorithms include two key elements to avoid issues with crowded & confused data**
  - **A generalised approach to the Astrometric Uncertainty Function allows for the full inclusion of the effects of perturbation due to blended sources — reduce false -ves!**
  - **Use of (two-sided) photometry to sort out multiple competing matches— reduce false +ves!**

- **Software package macauff developed to cross-match catalogues, including the effect of unresolved contaminant sources (and rejection of interloper objects using photometry in the static sky)**
  - **Developed through an IKC to Rubin/LSST:UK, matches planned to *Gaia*, *WISE*, VISTA, SDSS, …**
  - **We have compute time to cross-match datasets — let me know your favourite combo, and what you need matched (to LSST or otherwise)!**

- **Incorporating this extension of position uncertainty into real-time matches allows for more robust counterpart identification in the alert stream and a more accurate and precise transient SED**

- **Furthermore, we can provide *statistical* information on the level of photometric contamination unresolved contaminant sources cause, which can be subtracted in a probabilistic framework!**





Nearest-neighbour matching *will not* work in the era of Rubin!

The AUF does not need to, and in fact quite often should *not*, be Gaussian!

VERA C. RUBIN OBSERVATORY

University of Exeter

@Onoddil

@pm.me

.github.io

https://github.com/macauff/macauff

Wilson & Naylor, 2017, MNRAS, 468, 2517
Wilson & Naylor, 2018a, MNRAS, 473, 5570
Wilson & Naylor, 2018b, MNRAS, 481, 2148
Wilson, 2022, RNAAS, 6, 60
Wilson, 2023, RASTI, 2, 1

LSST:UK Consortium

UKRI Science and Technology Facilities Council

LINCC

Macauff

Tom J Wilson @onoddil

Or, Algorithms Aren’t The Whole Problem Anymore

Tom J Wilson @onoddil

lsdb, hipscat, and the problem of 30-billion row tables

https://github.com/astronomy-commons

hipscat Public
Hierarchical Progressive Survey Catalog
Python · 7 · BSD-3-Clause · 1 · 10 (1 issue needs help) · 1 · Updated 16 minutes ago

lsdb Public
Large Survey DataBase
Python · 3 · BSD-3-Clause · 2 · 48 (1 issue needs help) · 1 · Updated 39 minutes ago

hipscat-import Public
HiPSCat import - generate HiPSCat-partitioned catalogs
Python · 4 · BSD-3-Clause · 1 · 20 (3 issues need help) · 2 · Updated 3 days ago

LINCC

HiPSCat

Template | LINCC Frameworks Python Project Template

pypi v0.2.7 · build passing · codecov 100% · docs passing · benchmarks passing

Hierarchical Progressive Survey Catalog

A HiPSCat catalog is a partitioning of objects on a sphere. Its purpose is for storing data from large astronomy surveys, but could probably be used for other use cases where you have large data with some spherical properties.

Check out our ReadTheDocs site for more information on partitioning, installation, and contributing.

See related projects:

- HiPSCat Import (on GitHub) (on ReadTheDocs)
- LSDB (on GitHub) (on ReadTheDocs)

LSDB

Template | LINCC Frameworks Python Project Template

pypi v0.1.4 · build passing · codecov 99% · docs passing · benchmarks passing

LSDB - Large Survey DataBase

A framework to facilitate and enable spatial analysis for extremely large astronomical databases (i.e. querying and crossmatching O(1B) sources). This package uses dask to parallelize operations across multiple HiPSCat partitioned surveys.

Check out our ReadTheDocs site for more information on partitioning, installation, and contributing.

See related projects:

- HiPSCat (on GitHub) (on ReadTheDocs)
- HiPSCat Import (on GitHub) (on ReadTheDocs)

hipscat-import

Template | LINCC Frameworks Python Project Template

pypi v0.2.5 · build passing · codecov · docs passing

HiPSCat import - Utility for ingesting large survey data into HiPSCat structure.

Check out our ReadTheDocs site for more information on partitioning, installation, and contributing.

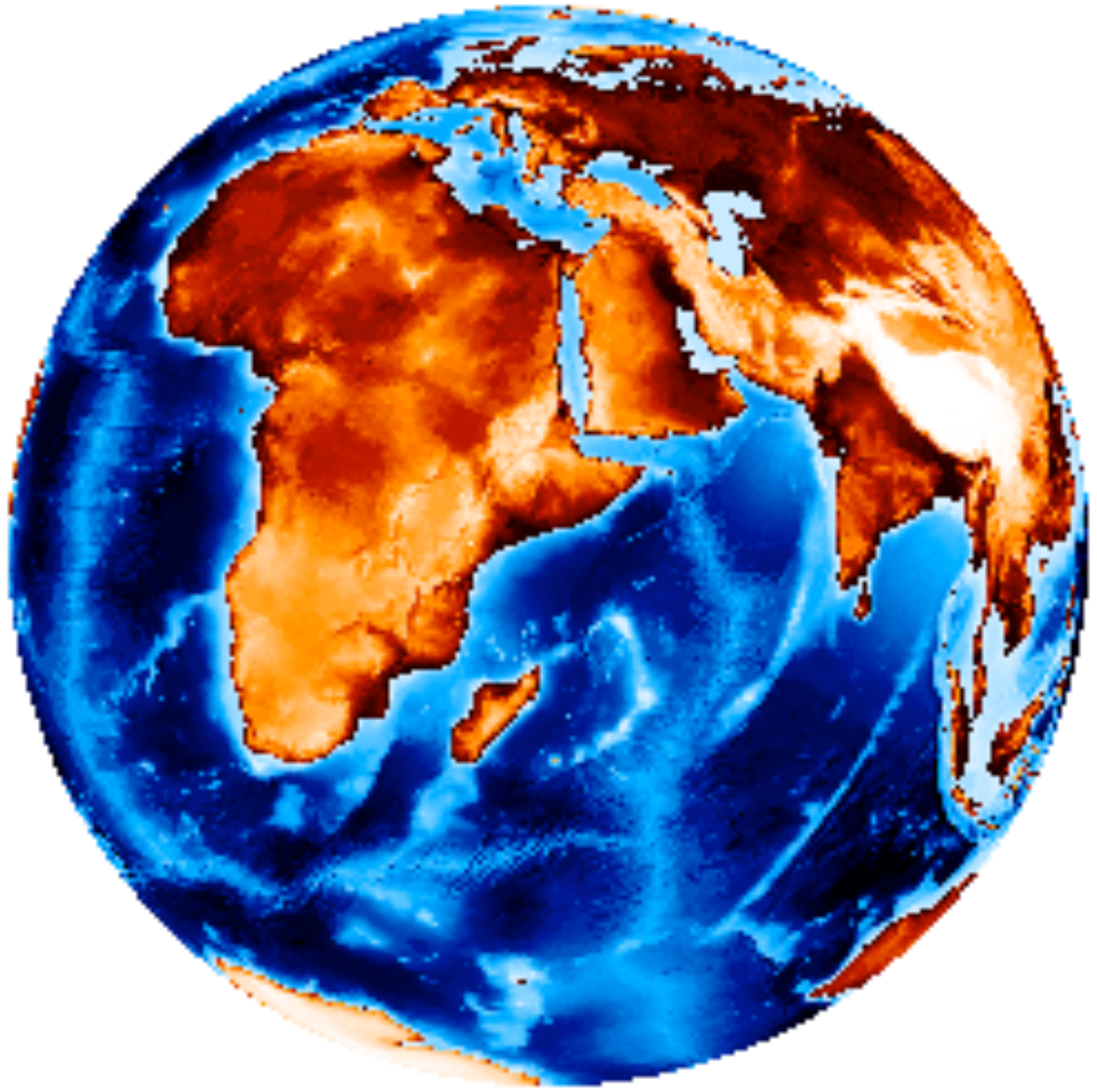
See related projects:

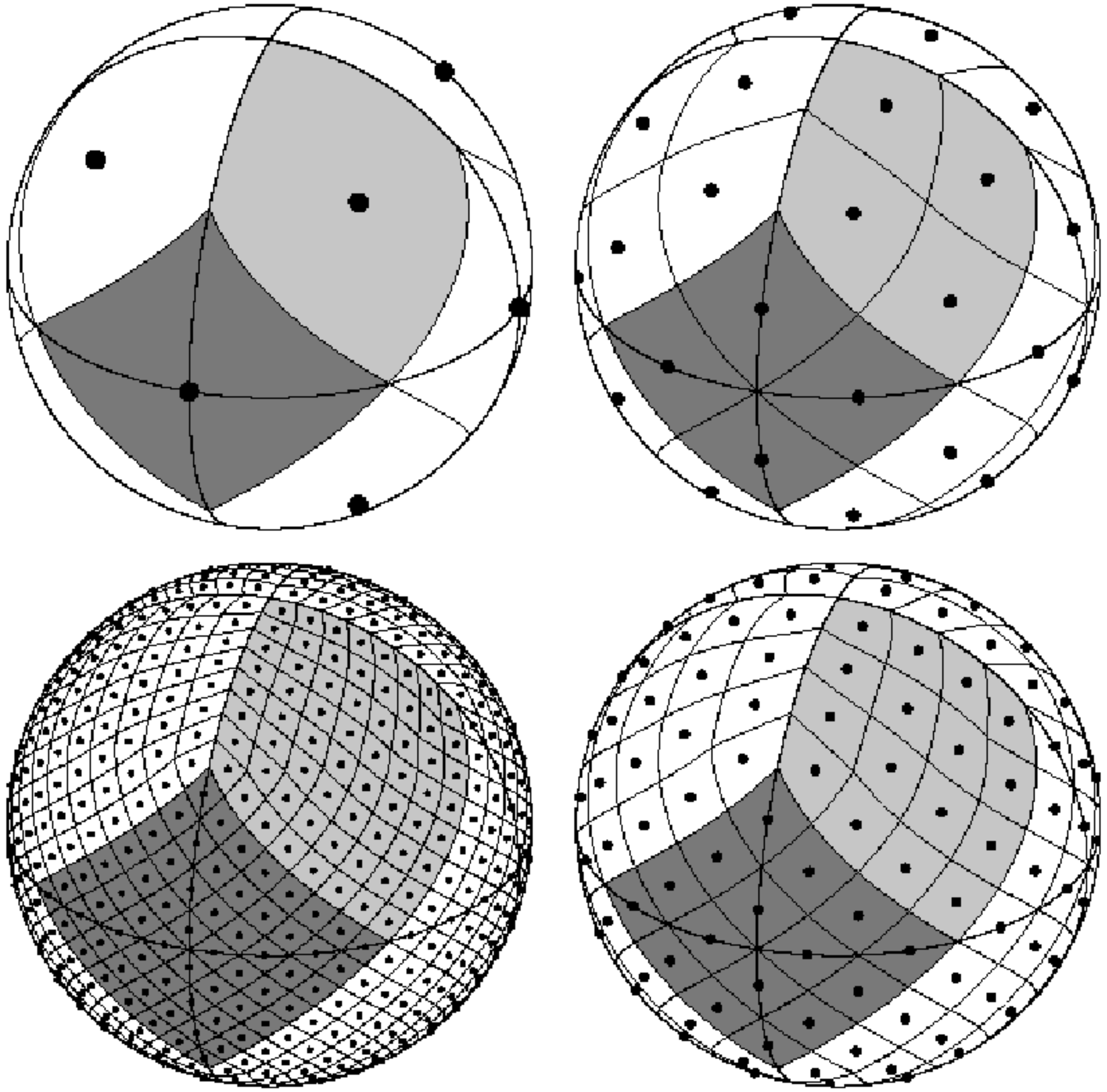
- HiPSCat (on GitHub) (on ReadTheDocs)
- LSDB (on GitHub) (on ReadTheDocs)

Tom J Wilson @onoddil

High-Speed, Robust Spatial Analysis

lsdb does astronomy on hipscat-formatted surveys, with hipscat built on healpix



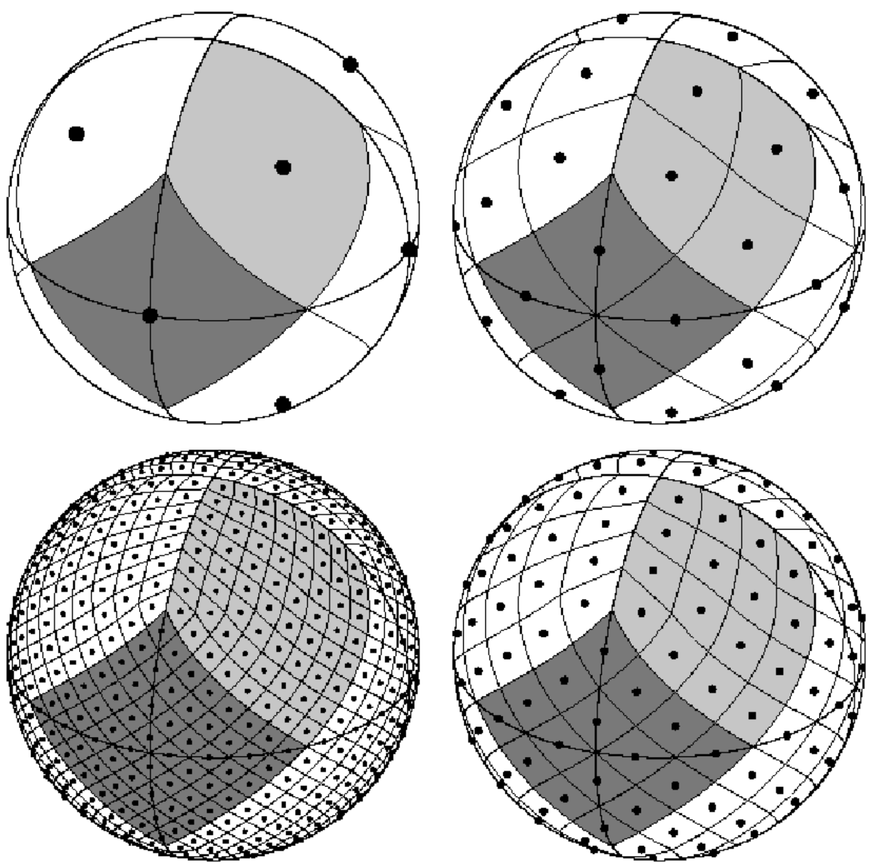


https://healpix.jpl.nasa.gov/

Tom J Wilson @onoddil

Importing catalogues, doing science

1. (Optional) Use hipscat-import to generate some valid catalogues



```
__ /path/to/catalogs/<catalog_name>/
    |__ catalog_info.json
    |__ partition_info.csv
    |__ Norder=1/
    |    |__ Dir=0/
    |        |__ Npix=0.parquet
    |        |__ Npix=1.parquet
    |__ Norder=J/
         |__ Dir=10000/
             |__ Npix=K.parquet
             |__ Npix=M.parquet
```

```python
import pandas as pd

import hipscat_import.pipeline as runner
from hipscat_import.catalog.arguments import ImportArguments
from hipscat_import.catalog.file_readers import CsvReader

# Load the column names and types from a side file.
type_frame = pd.read_csv("neowise_types.csv")
type_map = dict(zip(type_frame["name"], type_frame["type"]))

args = ImportArguments(
    output_artifact_name="neowise_1",
    input_path="/path/to/neowiser_year8/",
    file_reader=CsvReader(
        header=None,
        separator="|",
        column_names=type_frame["name"].values.tolist(),
        type_map=type_map,
        chunksize=250_000,
    ).read,
    ra_column="RA",
    dec_column="DEC",
    pixel_threshold=2_000_000,
    highest_healpix_order=9,
    use_schema_file="neowise_schema.parquet",
    sort_columns="SOURCE_ID",
    output_path="/path/to/catalogs/",
)
runner.run(args)
```

https://hipscat.readthedocs.io/en/latest

https://hipscat-import.readthedocs.io/en/latest

Tom J Wilson @onoddil

Importing catalogues, doing science

2. Having imported your datasets, do some spatial analysis

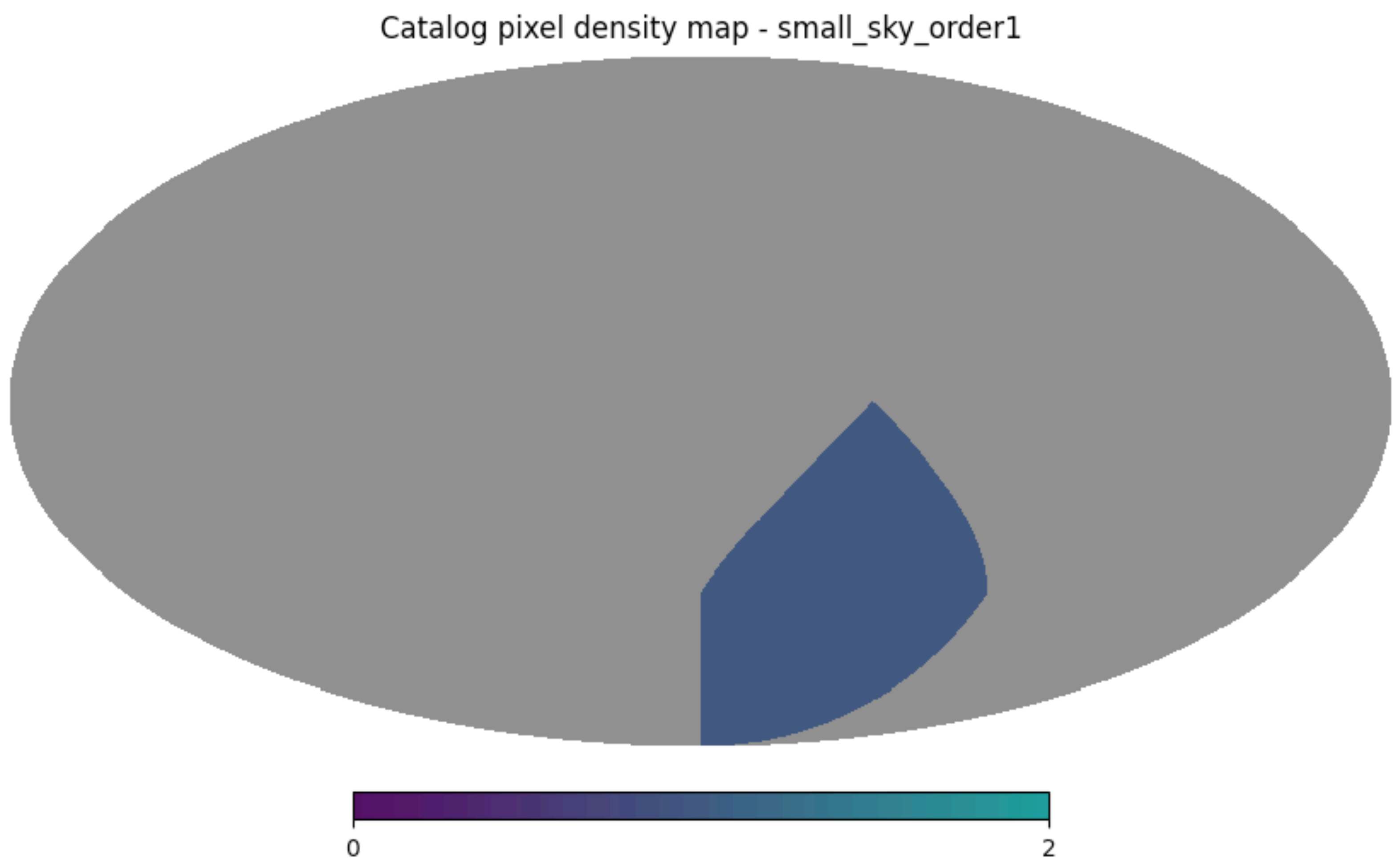
```
[1]: import hipscat
     import healpy as hp
     import numpy as np

     ## Fill in these variables with what's relevant in your use case:

     ### Change this path!!!
     catalog_path = "../../tests/data/small_sky_order1"

     ra = 0  # degrees
     dec = -80  # degrees
     radius_degrees = 10  # degrees
```

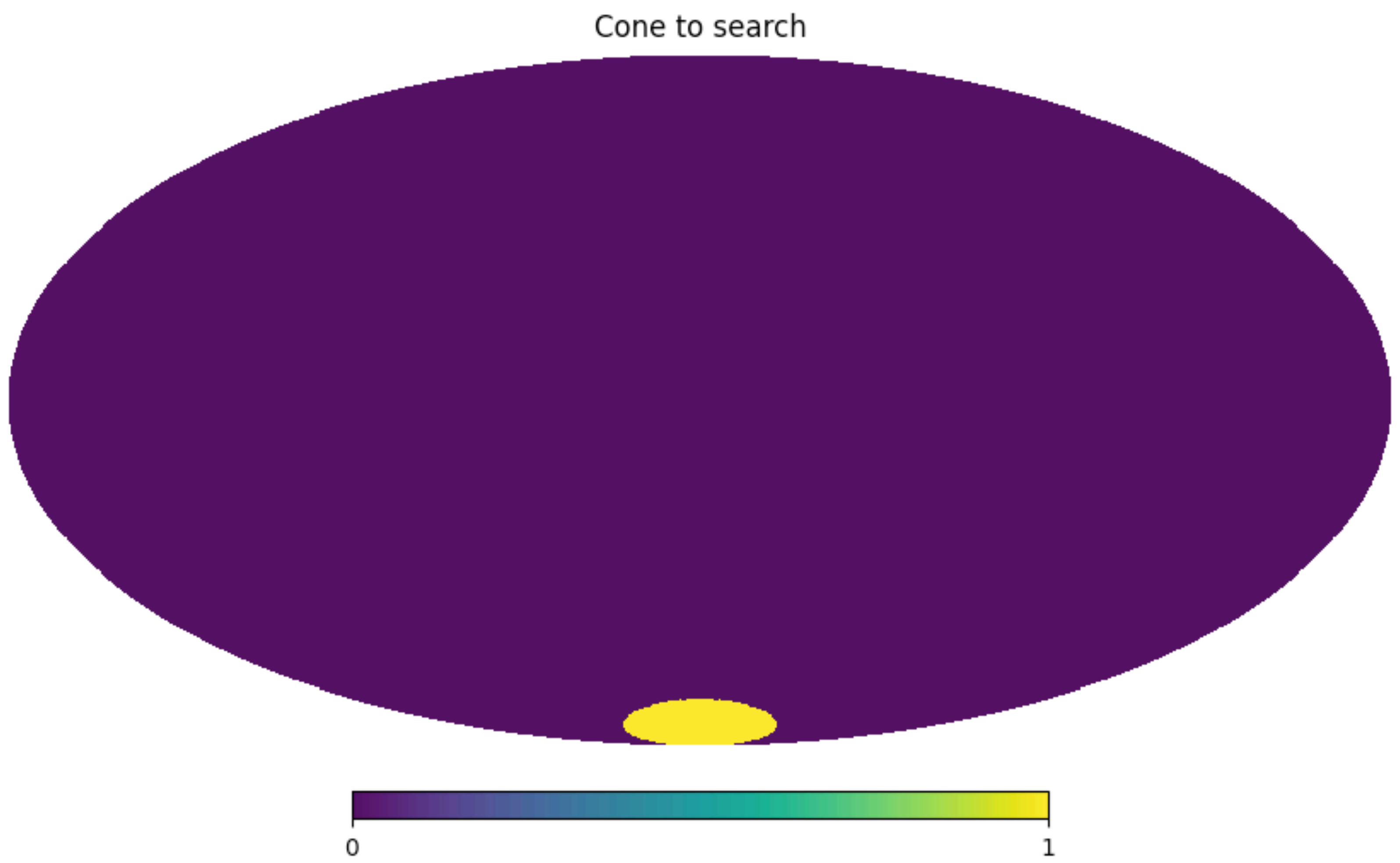
```
[2]: ## Load catalog

     catalog = hipscat.read_from_hipscat(catalog_path)
```

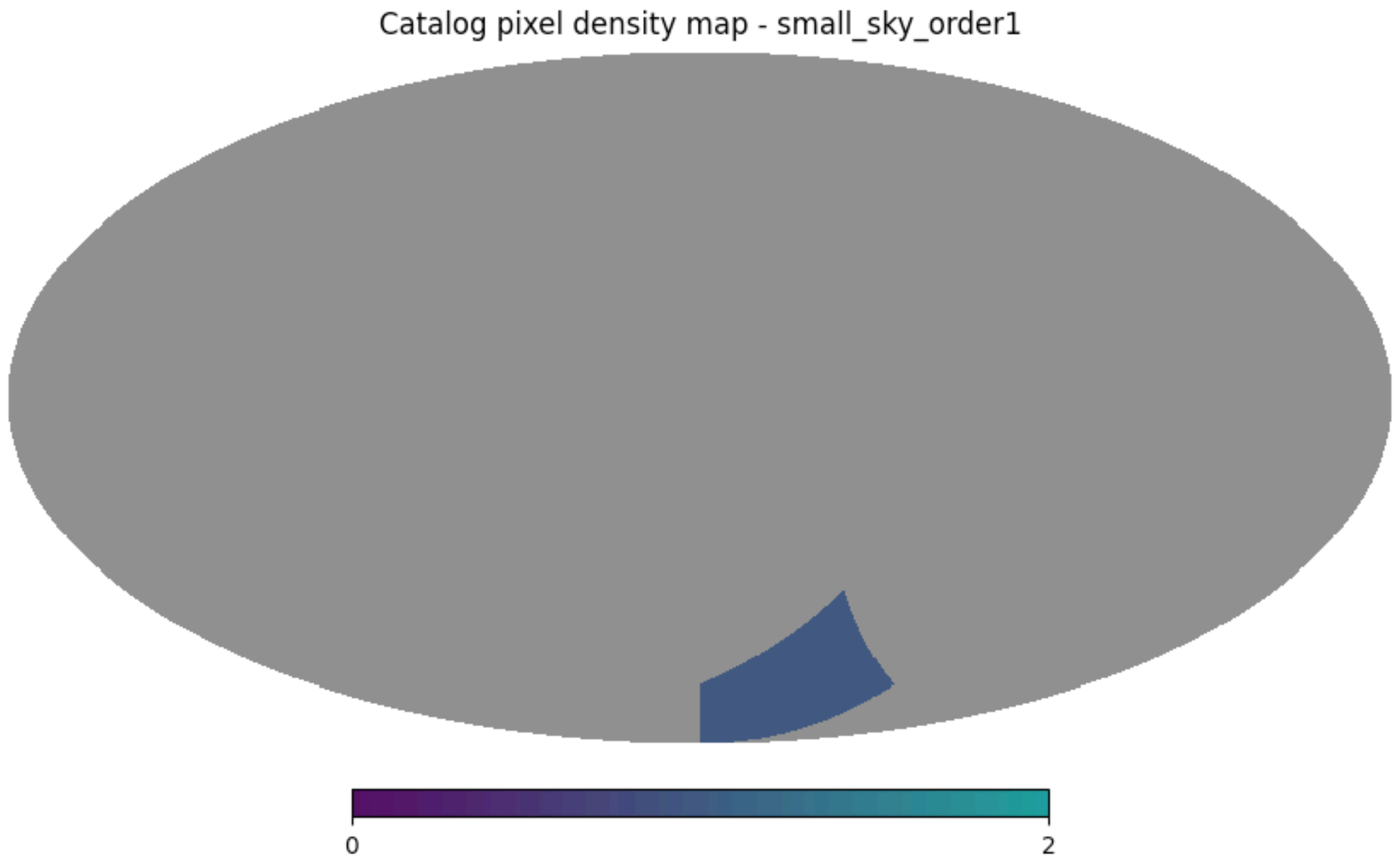
```
[3]: ## Plot catalog pixels

     hipscat.inspection.plot_pixels(catalog)
```

```
[4]: ## Plot the cone using healpy for demonstration

     NSIDE = 256
     NPIX = hp.nside2npix(NSIDE)
     m = np.zeros(NPIX)
     center_vec = hp.ang2vec(ra, dec, lonlat=True)
     radius_radians = np.radians(radius_degrees)
     cone_pixels = hp.query_disc(NSIDE, center_vec, radius_radians)
     m[cone_pixels] = 1
     hp.mollview(m, title="Cone to search")
```

```
[5]: ## Filter catalog and plot filtered pixels

     radius_arcseconds = radius_degrees * 3600
     filtered_catalog = catalog.filter_by_cone(ra, dec, radius_arcseconds)

     hipscat.inspection.plot_pixels(filtered_catalog)
```







https://hipscat.readthedocs.io/en/latest

Tom J Wilson @onoddil

Importing catalogues, doing science

3. Finally, do some science with lsdb, building off hipscat

Example use-case: cross-match ZTF BTS and NGC

Here we demonstrate how to cross-match Zwicky Transient Facility (ZTF) Bright Transient Survey (BTS) and New General Catalogue (NGC) using LSDB.

[5]:
```
%%time

ztf_bts = lsdb.from_dataframe(df_ztf_bts, ra_column="ra_deg", dec_column="dec_deg")
ngc = lsdb.from_dataframe(df_ngc, ra_column="ra_deg", dec_column="dec_deg", margin_thresho

ztf_bts = ztf_bts.query("redshift < 0.01")

matched = ztf_bts.crossmatch(ngc, radius_arcsec=1200, suffixes=("_ztf", "_ngc"))
matched
```

[6]:
```
%%time

# Create default local cluster
with Client():
    matched_df = matched.compute()

# Let's output transient name, NGC name and angular distance between them
matched_df = matched_df[["IAUID_ztf", "Name_ngc", "_dist_arcsec", "RA_ztf", "Dec_ztf"]].so
    by=["_dist_arcsec"]
)
matched_df
```

[6]:

_hipscat_index	IAUID_ztf	Name_ngc	_dist_arcsec	RA_ztf	Dec_ztf
7828968065004994560	SN2022xxf	3705	2.985601	11:30:05.93	+09:16:57.2
7200132046067335168	SN2020vg	I 738	8.931140	11:48:54.43	-04:40:53.8
3231460713012658176	SN2022pgf	5894	12.223285	15:11:41.90	+59:49:12.2

[8]:
```
c = SkyCoord(matched_df["RA_ztf"].values[0], matched_df["Dec_ztf"].values[0], unit=("hourar
ra = c.ra.degree
dec = c.dec.degree
oid = matched_df["IAUID_ztf"].values[0]
table = getimages(ra, dec, size=1200, filter
url = (
    "https://ps1images.stsci.edu/cgi-bin/fit
    "ra={}&dec={}&size=1200&format=jpg&red={
).format(ra, dec, table["filename"][0], tabl
im = get_ps1_image(url)
fig, ax = plt.subplots(figsize=(7, 3))
if im is not None:
    ax.imshow(im)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.scatter(np.average(plt.xlim()), np.av
    ax.set_title(oid)
    plt.show()
```

https://lsdb.readthedocs.io/en/latest

Tom J Wilson @onoddil

Importing catalogues, doing science

3. Finally, do some science with lsdb, building off hipscat

Example use-case: cross-match ZTF BTS and NGC

Here we demonstrate how
(BTS) and New General Cat

cross-match ZTF BTS and NGC

```
[5]: %%time

ztf_bts = lsdb.from_dataframe(df_ztf_bts, ra_column="ra_deg", dec_column="dec_deg")
ngc = lsdb.from_dataframe(df_ngc, ra_column="ra_deg", dec_column="dec_deg", margin_thresho

ztf_bts = ztf_bts.query("redshift < 0.01")

matched = ztf_bts.crossmatch(ngc, radius_arcsec=1200, suffixes=("_ztf", "_ngc"))
matched
```

```
[6]: %%time

# Create default local
with Client():
    matched_df = matche

# Let's output transien
matched_df = matched_df
    by=["_dist_arcsec"]
)
matched_df
```

[6]:

	IAUID_ztf	Name_ngc	_dist_arcsec	RA_ztf	Dec_ztf
					57.2
					53.8
3231460713012658176	SN2022pgf	5894	12.223285	15:11:41.90	+59:49:12.2

```
[8]: c = SkyCoord(matched_df["RA_ztf"].values[0], matched_df["Dec_ztf"].values[0], unit=("hourar
ra = c.ra.degree
dec = c.dec.degree
```

```
Args:
    algorithm (BuiltInCrossmatchAlgorithm | Type[AbstractCrossmatchAlgorithm]): The
        algorithm to use to perform the crossmatch. Can be either a string to specify one of
        the built-in cross-matching methods, or a custom method defined by subclassing
        AbstractCrossmatchAlgorithm.

        Built-in methods:
            -`kd_tree`: find the k-nearest neighbors using a kd_tree
```

Nearest-neighbour matching _will not_ work in the era of Rubin!

Tom J Wilson @o

https://lsdb.readthedocs.io/en/latest

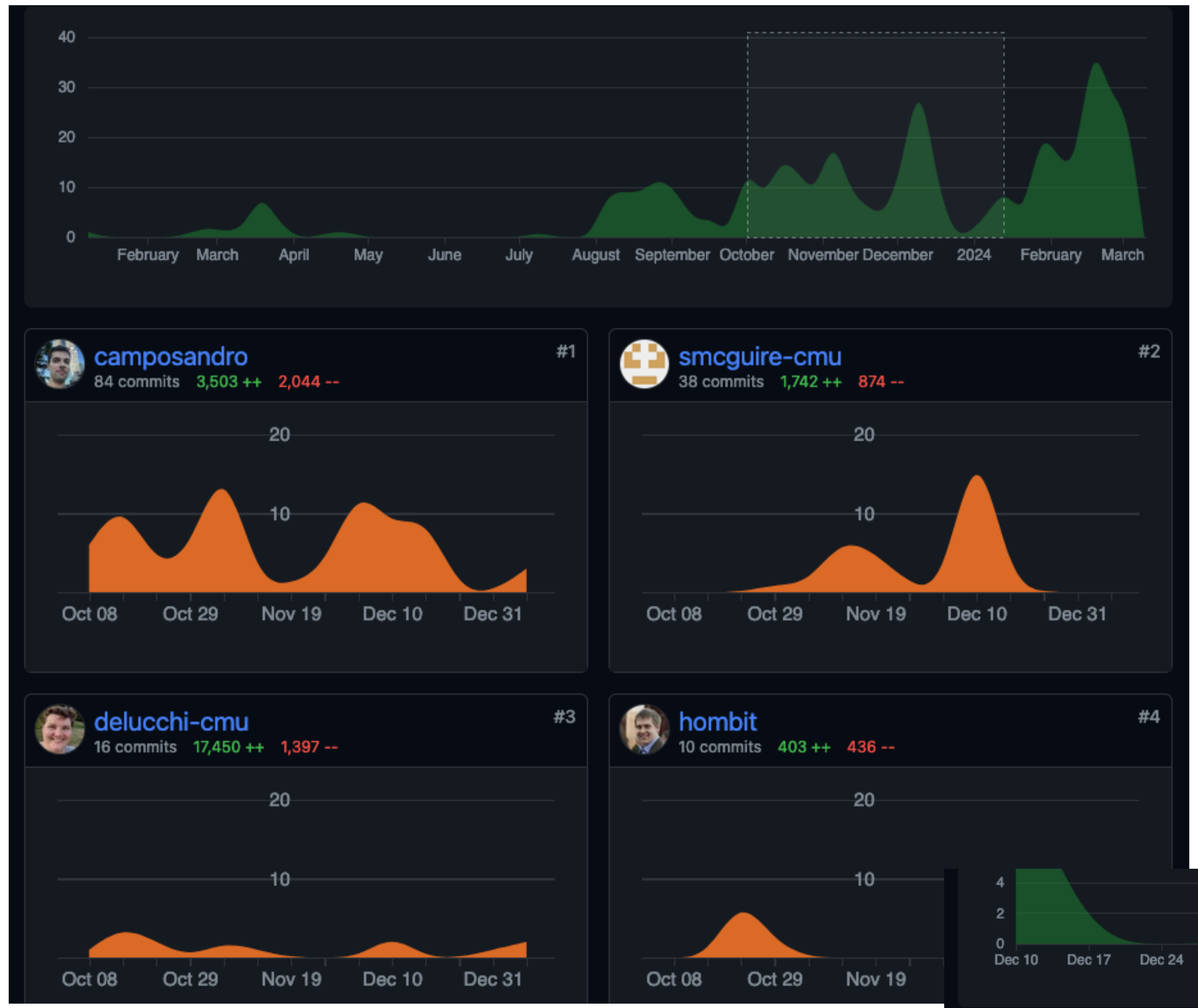
LSDB - Macauff

Or, The Actual LINCC Incubator!

lsdb-macauff: The Best of Both Worlds

The LINCC Incubator's goal was simple: combine lsdb and macauff.
In practice, this was quite involved, with lots of codebase activity!

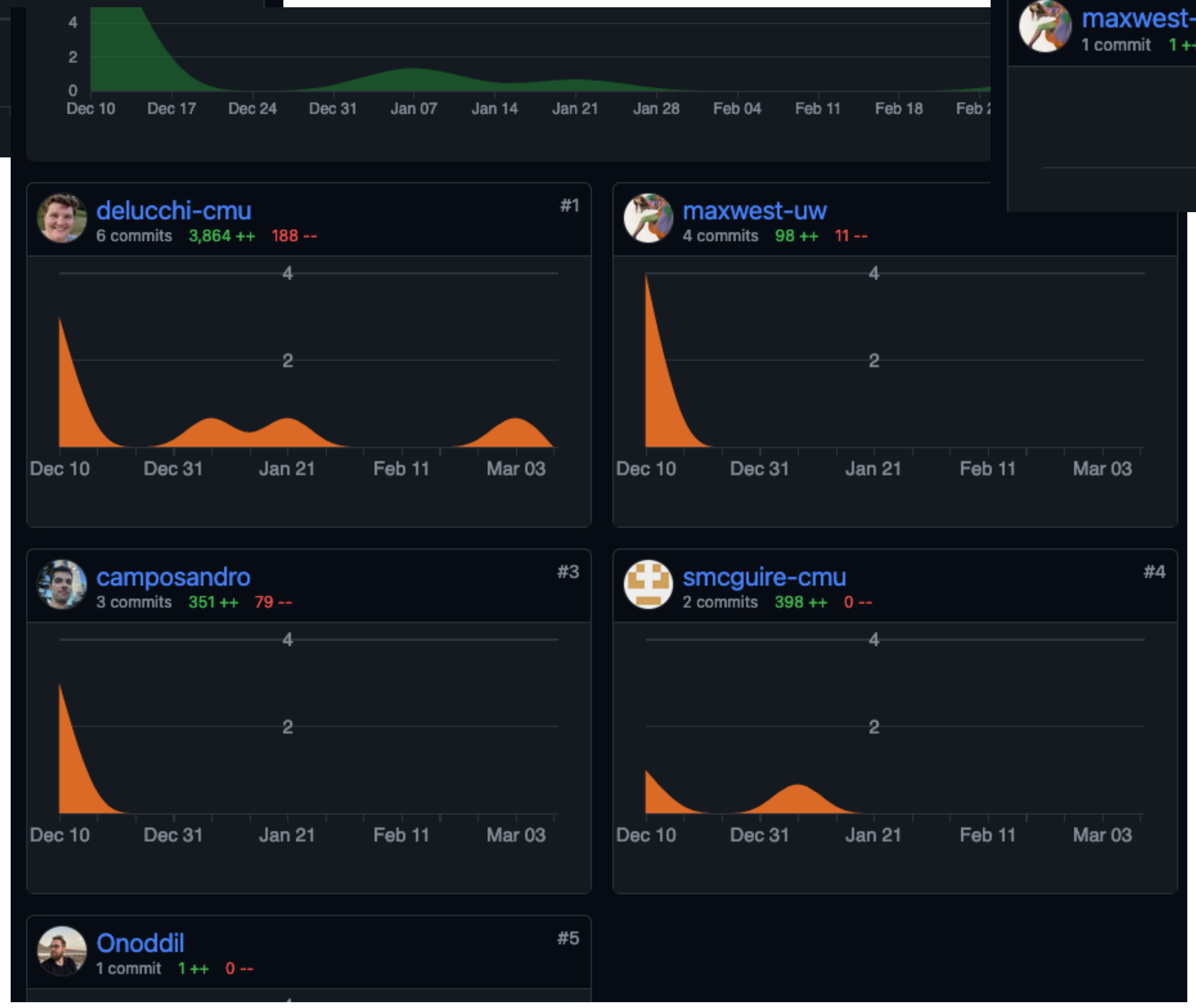
lsdb*



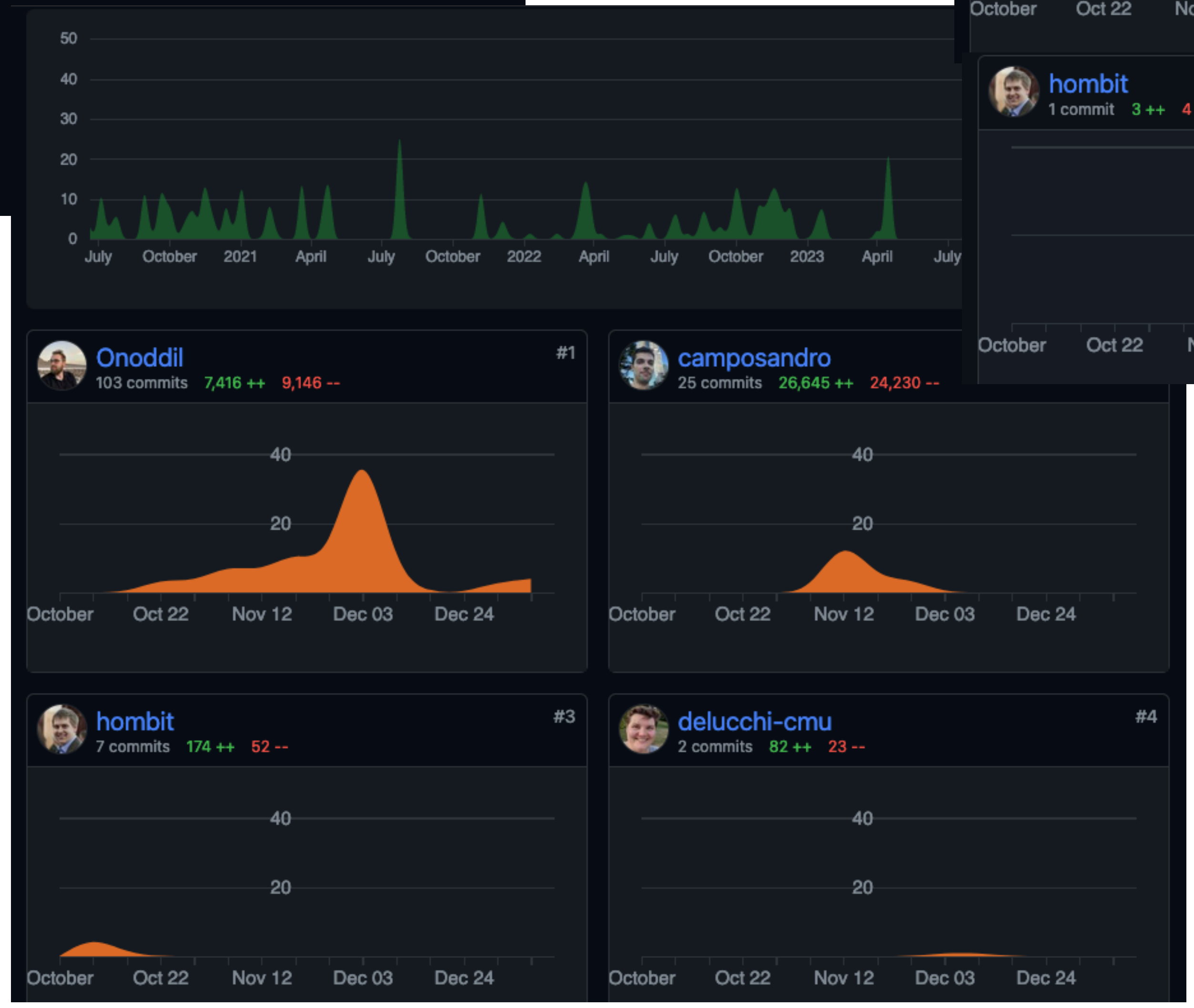
hipscat*

hipscat-import*

lsdb-macauff



macauff



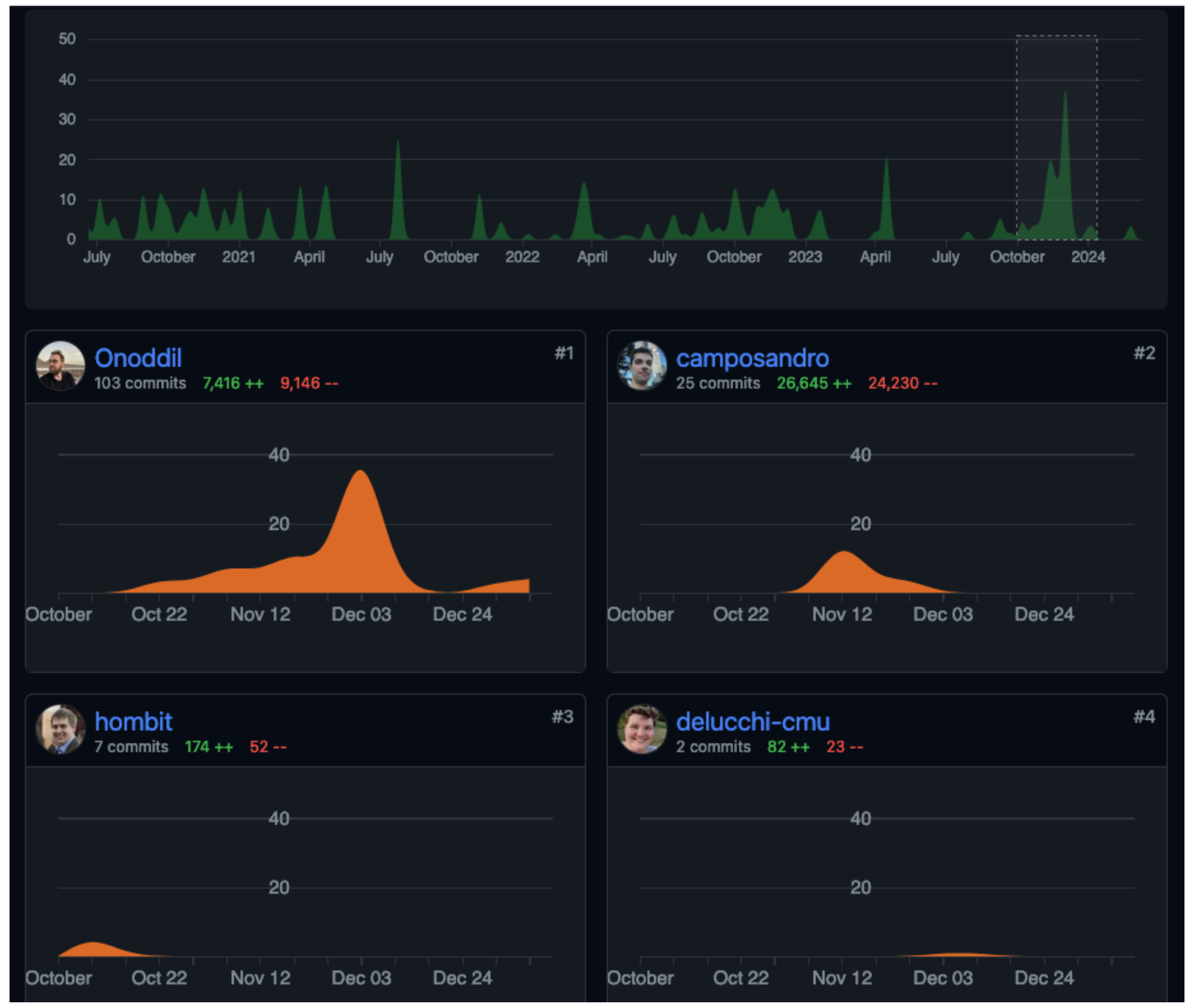
* some of this was non-incubator work, of course

Tom J Wilson @onoddil

lsdb-macauff: The Best of Both Worlds

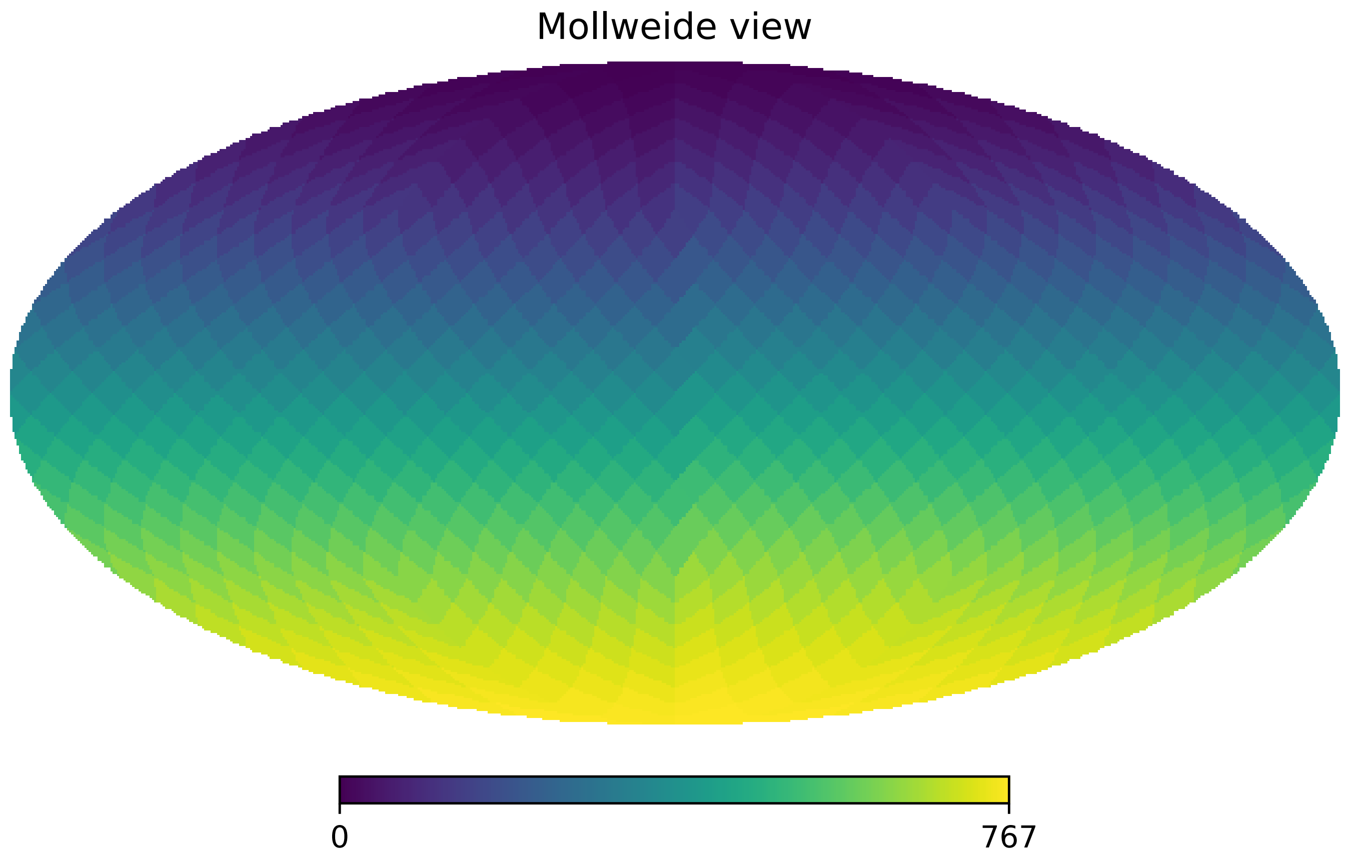
The LINCC Incubator's goal was simple: combine lsdb and macauff.
In practice, this was quite involved!

macauff



Improvements to macauff's algorithms:

- Better algorithms that handle more complex match arrangements, like healpix
  - Current IKC work built on rectilinear grid; relaxing this assumption improves use cases
- Split out inner loop from I/O
  - macauff-internal algorithms (i.e., code-versions of the first half of this talk) are independent of how you load input catalogues and output counterparts
  - Pass various arrays and variables around in memory, necessary for lsdb's dask functionality
- Allow for certain variables to be pre-calculated ahead of time, and others to be loaded from memory ahead of time, required for lsdb workflows



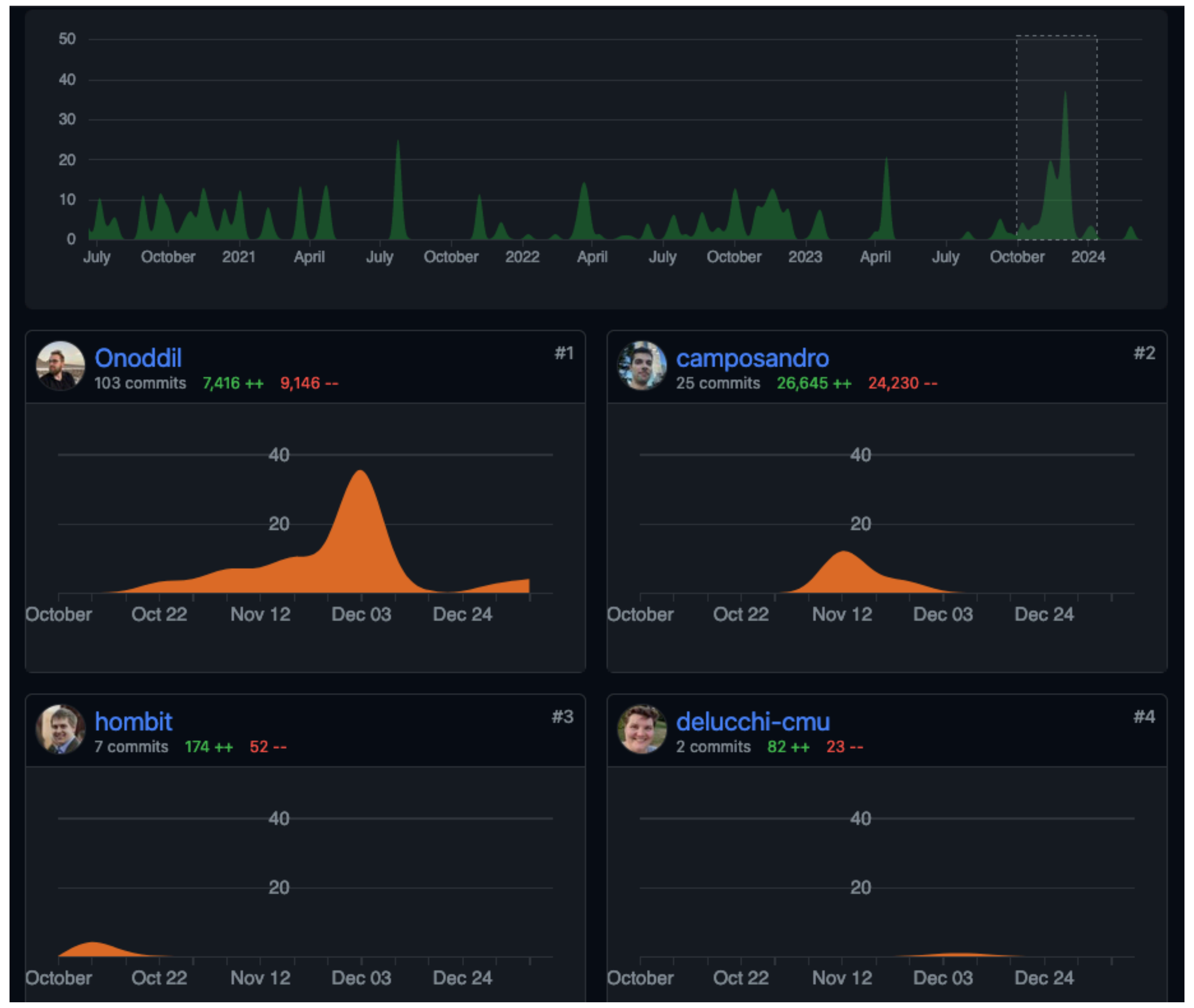

```
class Macauff():
    '''
    Class to perform the catalogue-catalogue association determination, on two
    datasets which are already pre-processed and set up for cross-matching.

    Parameters
    ----------
    cm : Class
        Input "IO" class, with the necessary parameters and datasets configured
        for cross-matching.
    '''
```

Tom J Wilson @onoddil

lsdb-macauff: The Best of Both Worlds

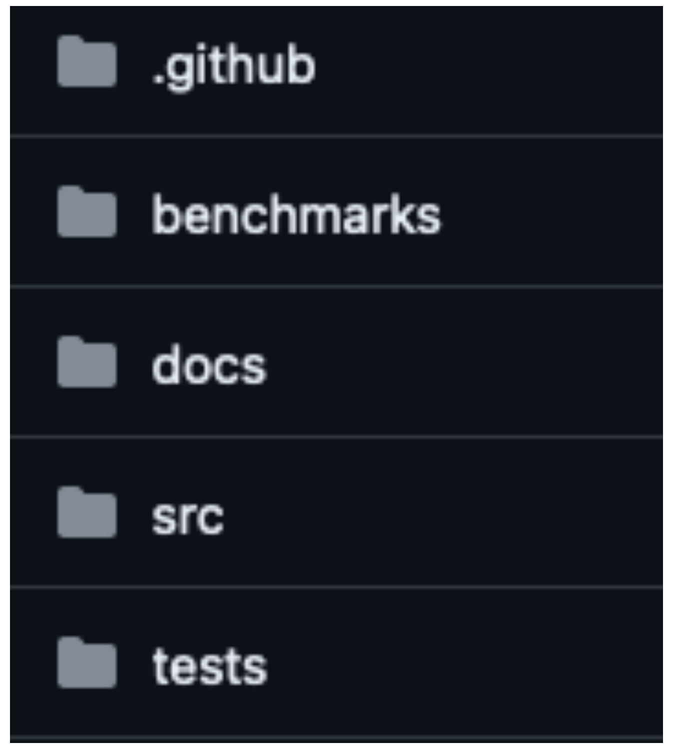
The LINCC Incubator's goal was simple: combine lsdb and macauff.
In practice, this was quite involved!

macauff



Improvements to macauff's codebase:

- **New code layout, for better security and test reproduction**
- **New build system**
  - **As with almost every software package of recent years, macauff was compiled using setup.py, which is deprecated. Moved to a future-proofed CMake build, allowing for "modern" python packaging with Fortran code.**
- **Apply the LINCC Python Project Template, a very easy way to get good packaging**
  - **Linting, package builds, benchmarking, pip installing, automated version tags, etc. for your project.**
  - **A good stress-test of the PPT for an already-mature codebase as well!**
- **Helped set up a readthedocs account for documentation, which I had been putting off getting done…**



Template LINCC Frameworks Python Project Template

https://macauff.readthedocs.io/en/latest/

Tom J Wilson @onoddil

lsdb-macauff: The Best of Both Worlds

The LINCC Incubator's goal was simple: combine lsdb and macauff.
In practice, this was quite involved!

hipscat* **hipscat-import***

- **Various extensions/changes to hipscat, hipscat-import that kicking the tyres in a new way revealed**
- **Added new functionality to allow for the loading of pre-computed macauff associations as a hipscat files**

```
import_pipeline
    arguments.py
    convert_metadata.py
    map_reduce.py
    resume_plan.py
    run_import.py
```

```python
def run(args, client):
    """run macauff cross-match import pipeline"""
```

```python
def from_yaml(input_file, output_directory):
    """Read YAML file with column metadata for the various cross-match files from macauff.
```

```python
single_path = '                    /macauff_association/Norder=2/Dir=0/Npix=0.parquet'
pq.read_metadata(single_path).schema
```

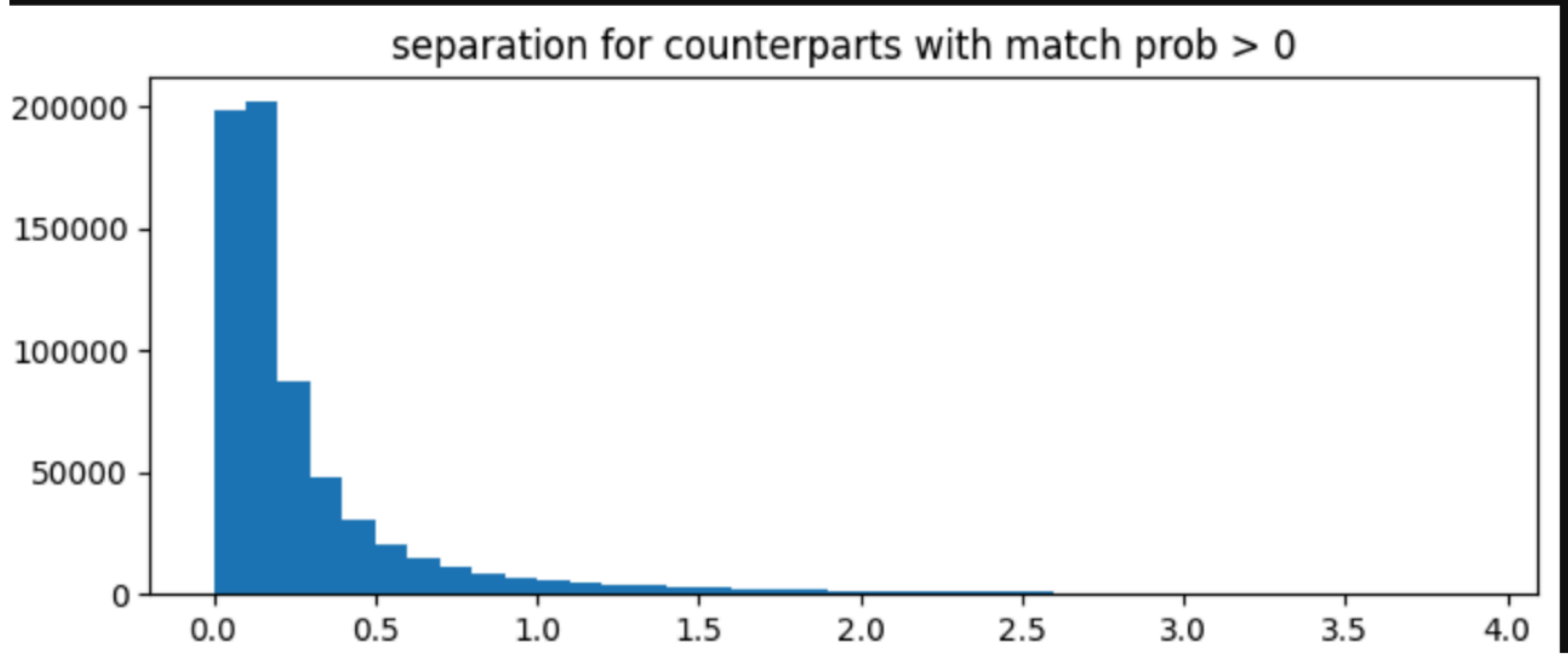
```
<pyarrow._parquet.ParquetSchema object at 0x7f52db1542c0>
required group field_id=-1 schema {
  optional int64 field_id=-1 gaia_source_id;
  optional double field_id=-1 gaia_ra;
```

```python
assoc_frame = pd.read_parquet(single_path)
```

```python
assoc_frame.head(5)[["gaia_source_id", "catwise_name", "match_p", "separation"]]
```

	gaia_source_id	catwise_name	match_p	separation
0	1488120269370752	J025039.37+021935.2	0.999997	0.117720

```python
sep_bins = np.arange(0, 4, .1)

for threshold in [0, 0.5, 0.85, 0.95, 0.99]:
    hist, bins = np.histogram(assoc_frame["separation"][assoc_frame["match_p"] > threshold], bins=sep_bins)

    width = np.diff(bins)
    center = (bins[:-1] + bins[1:]) / 2

    fig, ax = plt.subplots(figsize=(8,3))
    ax.bar(center, hist, align='center', width=width)
    plt.title(f"separation for counterparts with match prob > {threshold}")
    plt.show()
```



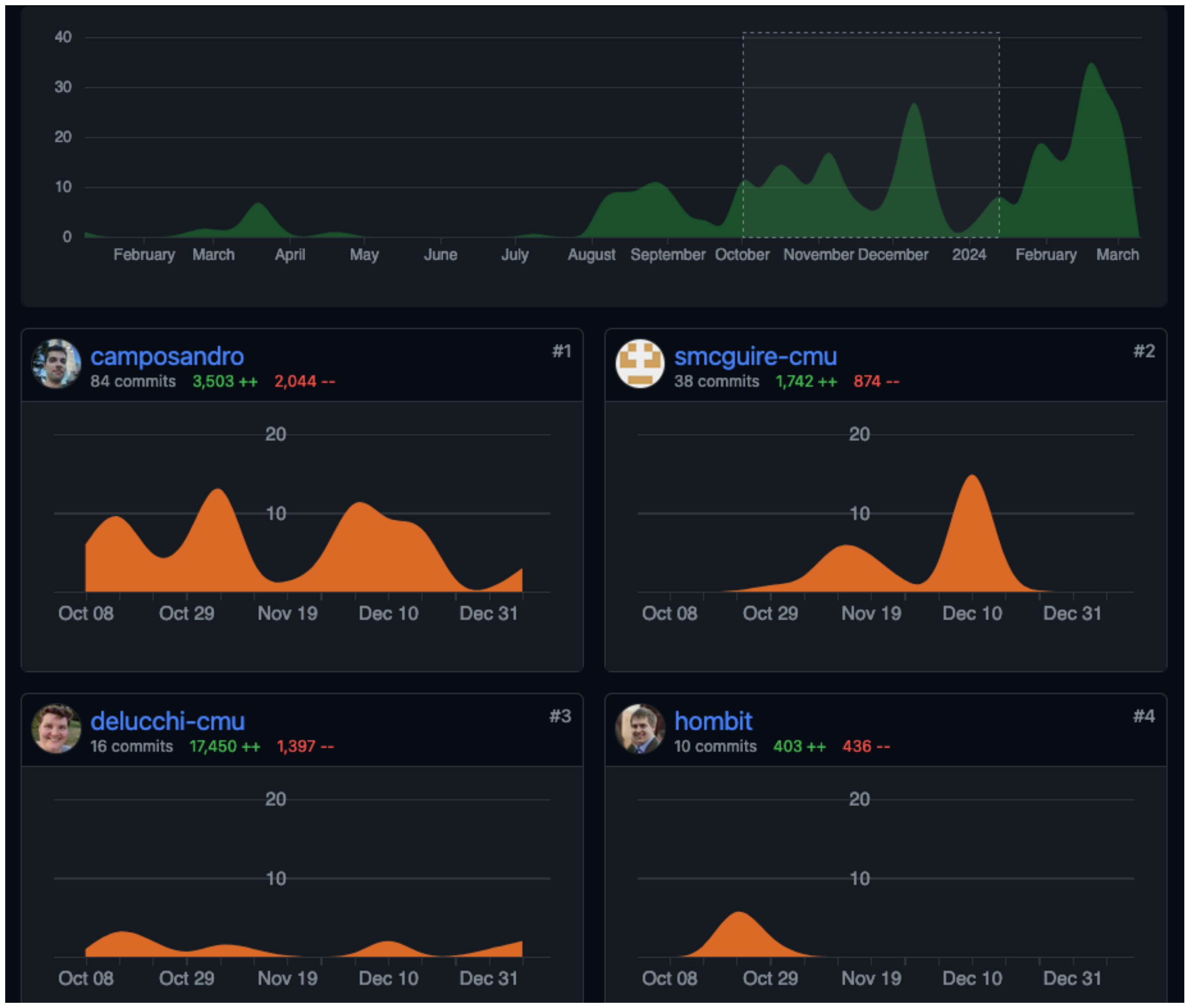
* some of this was non-incubator work, of course

Tom J Wilson @onoddil

lsdb-macauff: The Best of Both Worlds

The LINCC Incubator's goal was simple: combine lsdb and macauff.
In practice, this was quite involved!

lsdb*



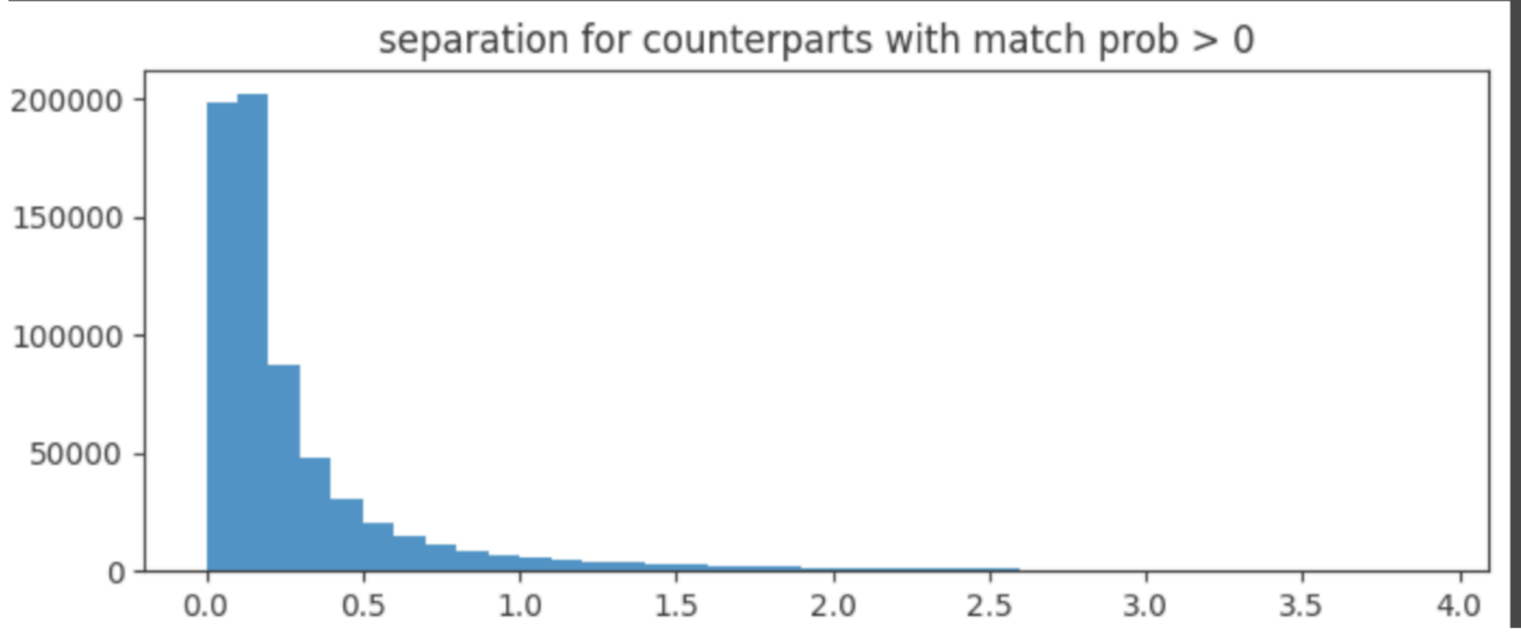
- When results from macauff are imported into lsdb (through hipscat!), you will now be able to do much more flexible cross-match refining than with nearest neighbour matching, using all of the "extras" macauff will produce, basically for free!

```
import lsdb

from dask.distributed import Client

client = Client(n_workers=6, memory_limit='20GB', local_directory="[redacted]",
```

Each catalog is loaded using the `lsdb.read_hipscat` method. First the macauff association table, and then the gaia and wise object catalogs. The loading is done lazily, so at this stage the results show just the column names and types.

```
macauff_association = lsdb.read_hipscat("[redacted]/macauff_association/")
macauff_association
```

```
center = (bins[:-1] + bins[1:]) / 2

fig, ax = plt.subplots(figsize=(8,3))
ax.bar(center, hist, align='center', width=width)
plt.title(f"separation for counterparts with match prob > {threshold}")
plt.show()
```




```
joined = gaia.join(catwise, through=macauff_association)
joined
```

```
joined.query("match_p > 0.999")._ddf.partitions[0].compute()
```

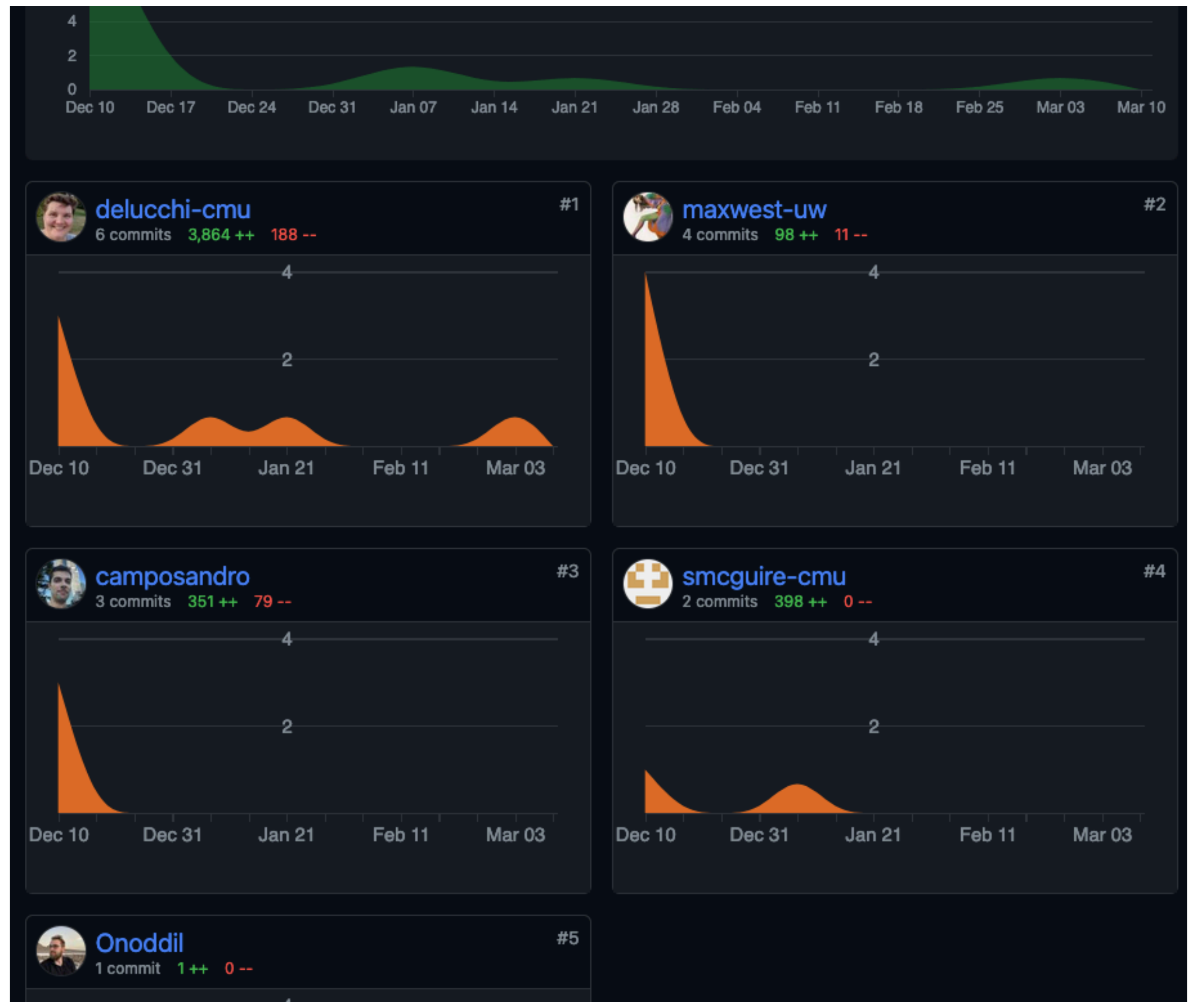
* some of this was non-incubator work, of course

Tom J Wilson @onoddil

lsdb-macauff: The Best of Both Worlds

The LINCC Incubator's goal was simple: combine lsdb and macauff.
In practice, this was quite involved!

lsdb-macauff



- **New shell that talks to both macauff and lsdb — hence lsdb-macauff!**
- **Allows macauff to be called within lsdb, for smaller-scale, individual cross-match runs**
- **Will provide all of the same advantages that using pre-join tables does, but for other science cases**

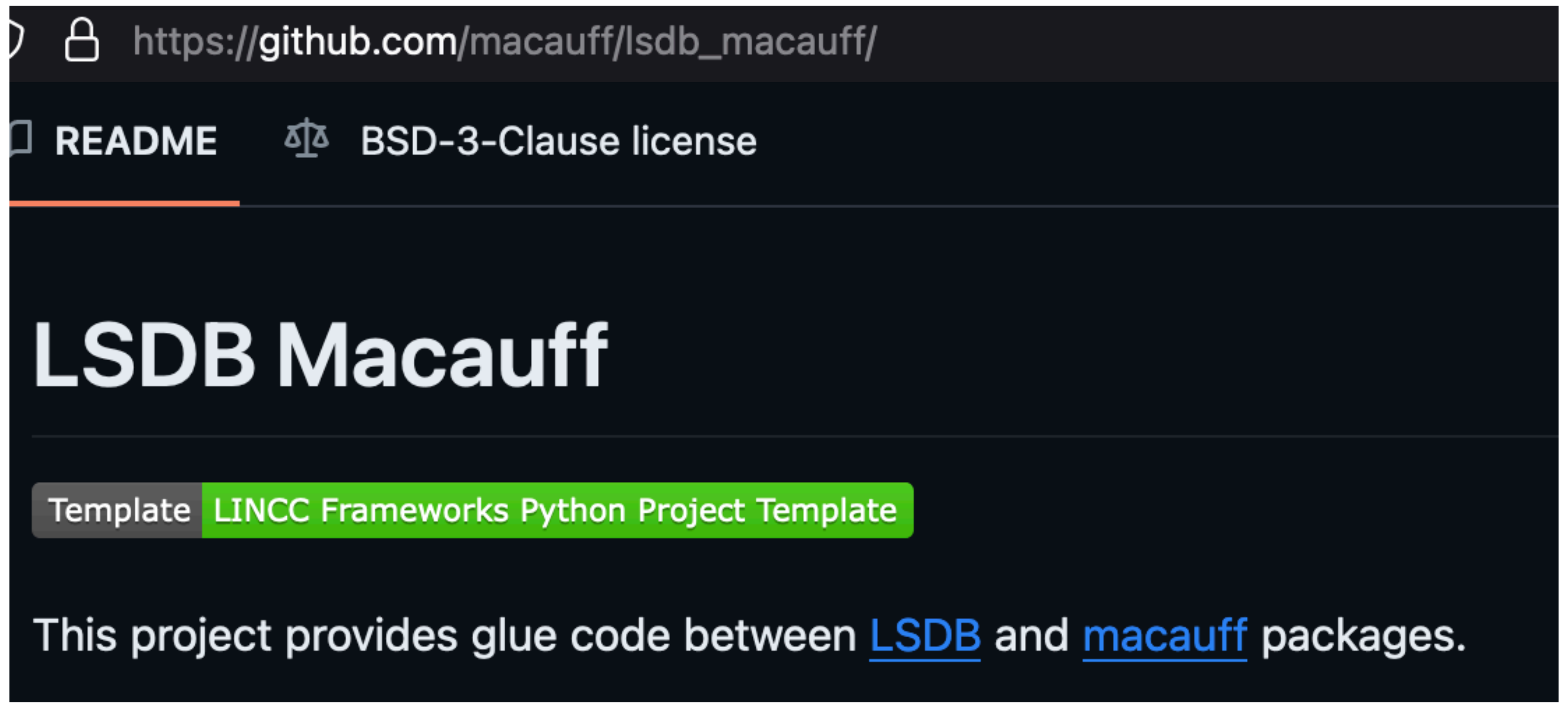
```
def test_macauff_crossmatch(
    catalog_a_csv, catalog_b_csv, all_sky_params, gaia_all_sky_params, wise_all_sky_params
):

    catalog_a = Catalog(CatalogInfo({"catalog_name": "catalog_a"}), [HealpixPixel(0, 0)])
    catalog_b = Catalog(CatalogInfo({"catalog_name": "catalog_b"}), [HealpixPixel(0, 0)])

    algo = MacauffCrossmatch(
        left=pd.read_csv(catalog_a_csv),
        right=pd.read_csv(catalog_b_csv),
        left_order=0,
        left_pixel=0,
        right_order=0,
        right_pixel=0,
        left_metadata=catalog_a,
        right_metadata=catalog_b,
        suffixes=("a", "b"),
        right_margin_hc_structure=None,
    )
    algo.crossmatch(all_sky_params, gaia_all_sky_params, wise_all_sky_params, None, None)
```

```
class MacauffCrossmatch(AbstractCrossmatchAlgorithm):
    """Class that runs the Macauff crossmatch"""

    def crossmatch(
        self,
        joint_all_sky_params,
        left_all_sky_params,
        right_all_sky_params,
        left_tri_map_histogram,
        right_tri_map_histogram,
    ) -> pd.DataFrame:
        # Calculate macauff pixel params using self.left_order
        # self.left_pixel, self.right_order, self.right_pixel
        macauff_pixel_params = None
        left_pixel_params = PixelParams(self.left_order, self.
        right_pixel_params = PixelParams(self.right_order, sel
        all_macauff_attrs = AllMacauffAttrs(
            joint_all_sky_params,
            macauff_pixel_params,
            left_all_sky_params,
            right_all_sky_params,
            left_pixel_params,
            right_pixel_params,
```

```
Args:
    algorithm (BuiltInCrossmatchAlgorithm | Type[AbstractCrossmatchAlgori
        algorithm to use to perform the crossmatch. Can be either a stri
        the built-in cross-matching methods, or a custom method defined b
        AbstractCrossmatchAlgorithm.

        Built-in methods:
            -`kd_tree`: find the k-nearest neighbors using a kd_tree
```

LSDB - Macauff

- Macauff is great, lsdb is too.
- Now they can talk to each other! Either in using pre-generated all-sky association tables (such as those we are making through LSST:UK!), or through individual use-cases with the lsdb-macauff wrapper.
  - Still some work left to do — there's always more to be done in software development!
- Extends lsdb-scale catalogue analysis to probabilistic cross-matching, much-needed in the era of LSST!



@Onoddil@pm.me
@Onoddil.github.io

VERA C. RUBIN OBSERVATORY

University of Exeter

LSST:UK Consortium

UKRI Science and Technology Facilities Council

LINCC