



# Data Challenges for the SKA

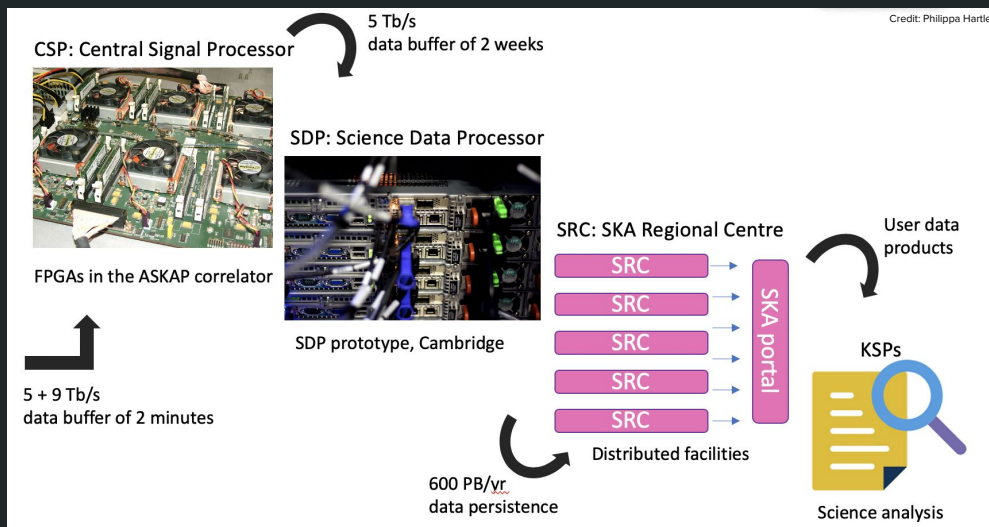
---

Alex Clarke (SKAO)  
a.clarke@skatelescope.org

On behalf of the team: Anna Bonaldi, Phillipa Hartley, Rosie Bolton, James Collinson, Rob Barnsley, Rohini Joshi

# Preparing users for SKA data access

- Users will get data products via regional centres in their part of the world through a single portal
- They will do most of the processing and analysis at these centres, rather than downloading data to their own computers

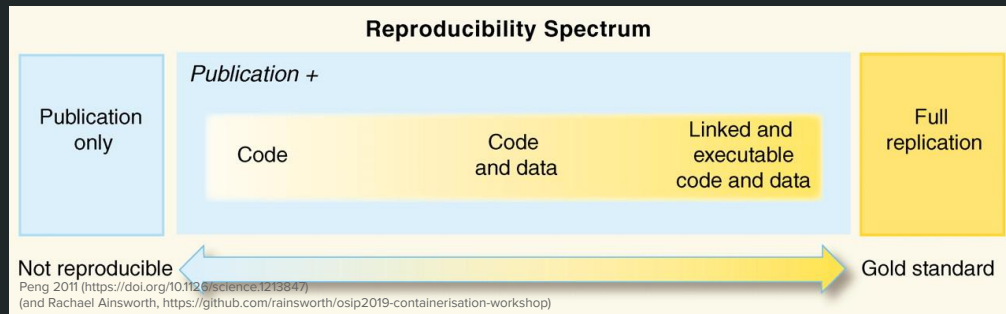


- Data challenges are designed to represent the workflow of users interacting with SKA data products

# Open science and reproducibility

Data challenges are being aimed at motivating users to follow best practices in their workflows

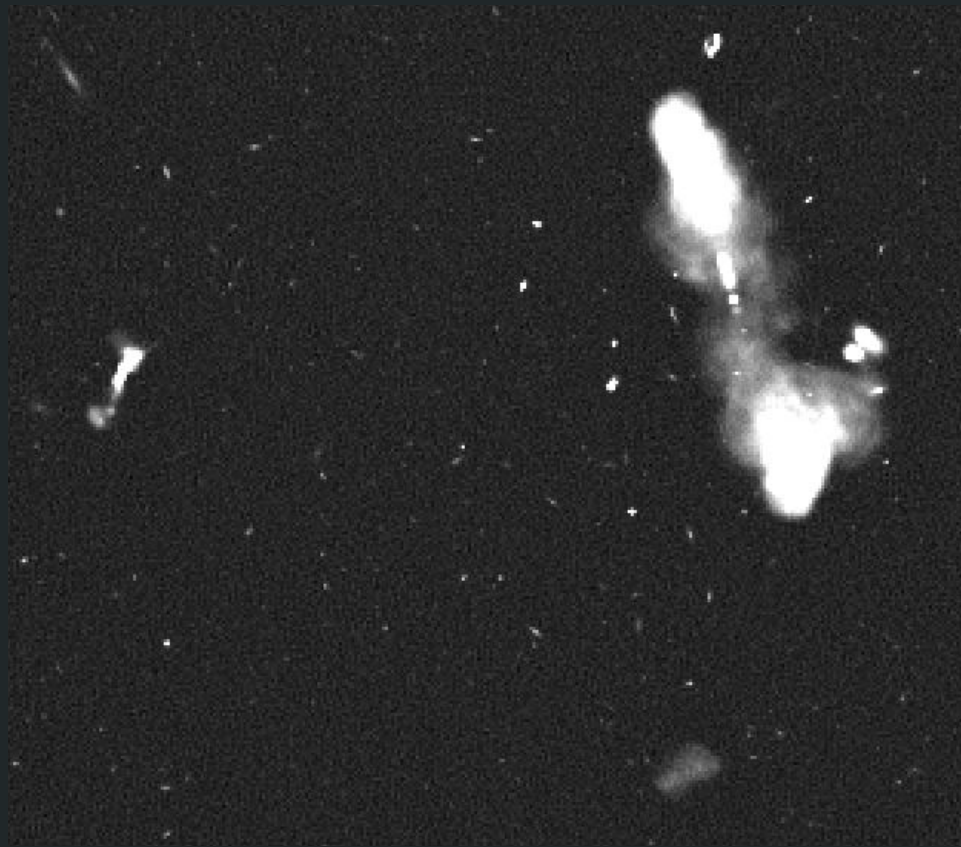
- Coding standards
  - Style guide, documentation, comments, functions, classes, modular, version control
- Publishing their code, data and workflow alongside the results
  - Zenodo/Github
- Ensuring their workflow is reproducible
  - Documenting software dependencies, using containers



# Science Data Challenge 1

## Source finding in images

- 32000 by 32000 pixel images were provided at 560, 1400, 9200 MHz, at resolutions of 2.4, 1.2 and 0.6 arcseconds, respectively.
- A truth catalogue is provided for a small area of sky, but when the challenge ran in 2019 the full truth catalogue was not provided.
- Participants submitted a catalogue of sources and were scored on how many they found and what they classified them as (star-forming galaxy, or steep/flat spectrum active galactic nuclei).
- There was no requirement for participants to show their workflow, code, or demonstrate it was reproducible.




# Science Data Challenge 1

## Developing an example solution

- We made a containerised example solution after the challenge to demonstrate a variety of best practices in research and software development
- Code is available on Github/Gitlab  
<https://gitlab.com/ska-telescope/sdc/sdc1-solution>
- Data is available on Zenodo  
<https://doi.org/10.5281/zenodo.4328029>
- Modular code structure, rather than one long script, means it's easier to read, document, collaborate on, test, and reuse parts for other projects.

The screenshot shows the 'SDC1 Solution' project page on the SKA Telescope website. The page is titled 'SDC1 Solution' with a project ID of 19657157. It features a 'Star' button with a count of 0. Below the title, it lists project statistics: 181 Commits, 6 Branches, 0 Tags, 5.5 MB Files, and 5.5 MB Storage. A description states it is 'A portable solution to SKA's first Science Data Challenge'. The 'README.md' file is open, showing the 'Science Data Challenge 1 Solution Workflow' section. The text describes the challenge (SDC1) and provides instructions for setting up the containerised environment and running a workflow script using Python helper modules.


ska-telescope > Science Data Challenges > SDC1 Solution > Details

**SDC1 Solution**   
Project ID: 19657157

☆ Star 0

→ 181 Commits ↗ 6 Branches ↗ 0 Tags 📁 5.5 MB Files 📁 5.5 MB Storage

A portable solution to SKA's first Science Data Challenge

 README.md

### Science Data Challenge 1 Solution Workflow

The SKA Science Data Challenge 1 (SDC1, <https://astronomers.skatelescope.org/ska-science-data-challenge-1/>) tasked participants with identifying and classifying sources in synthetic radio images.

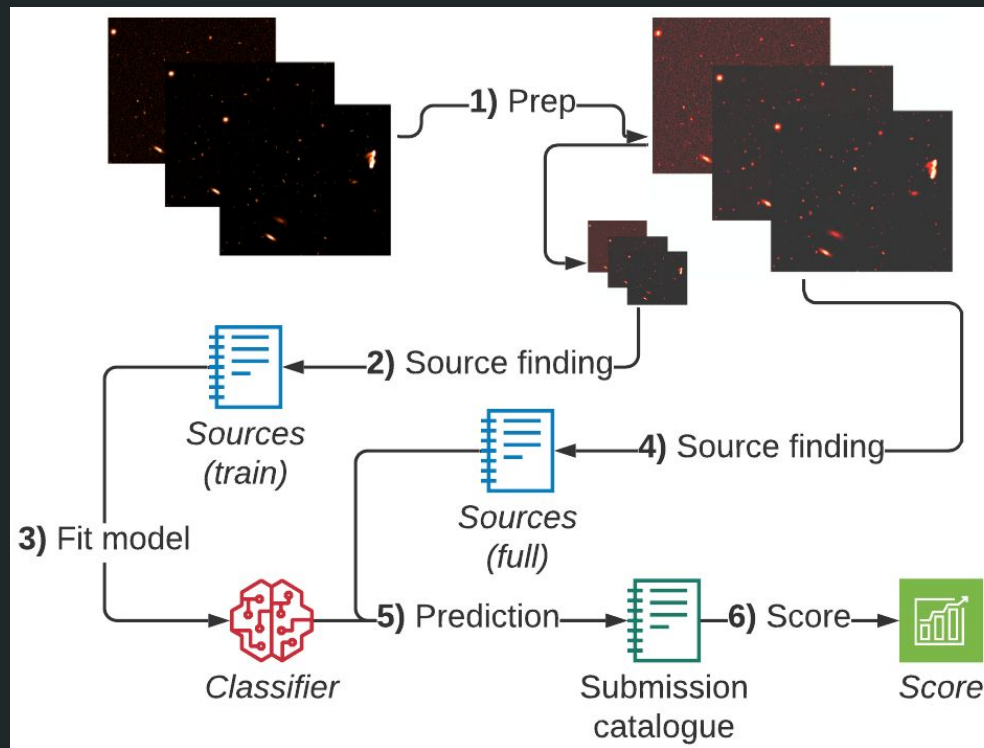
Here we present an environment and workflow for producing a solution to this challenge that can easily be reproduced and developed further.

Instructions for setting up the (containerised) environment and running a simple workflow script using some Python helper modules are provided in this document.

# Science Data Challenge 1

## Running the example solution

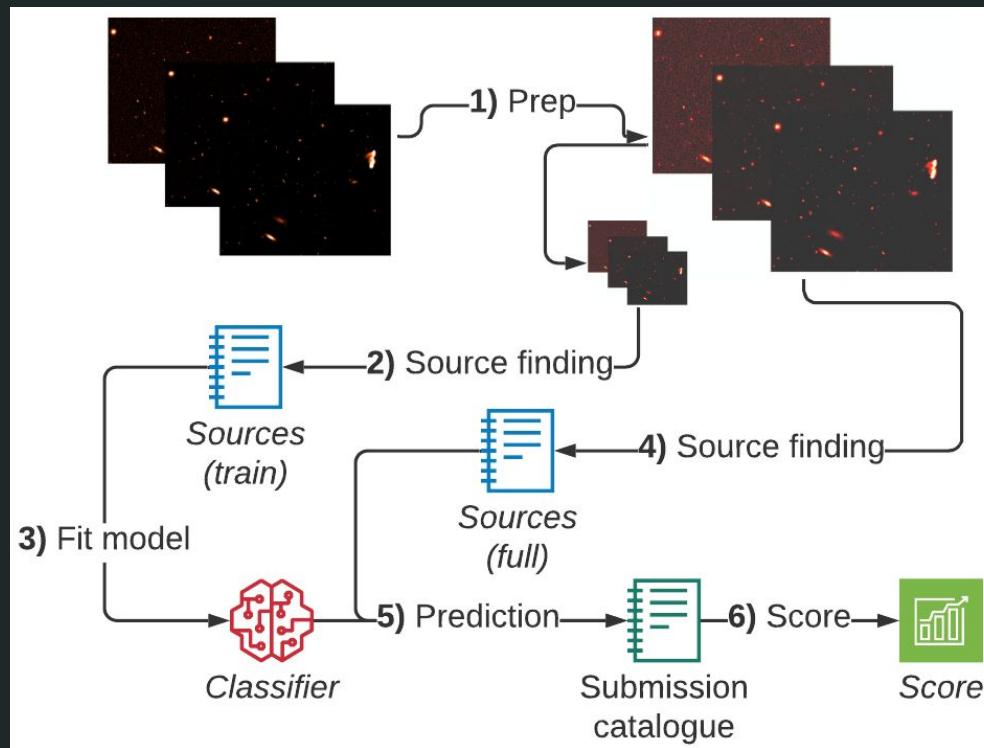
- Get Docker
- Environment variable and aliases
- Download data
- Make the Docker image
- Start the Container
- In the Container, run `sdc1_solution.py`, which reproduces the entire workflow, producing output catalogues and scores, taking 3 hours



# Science Data Challenge 1

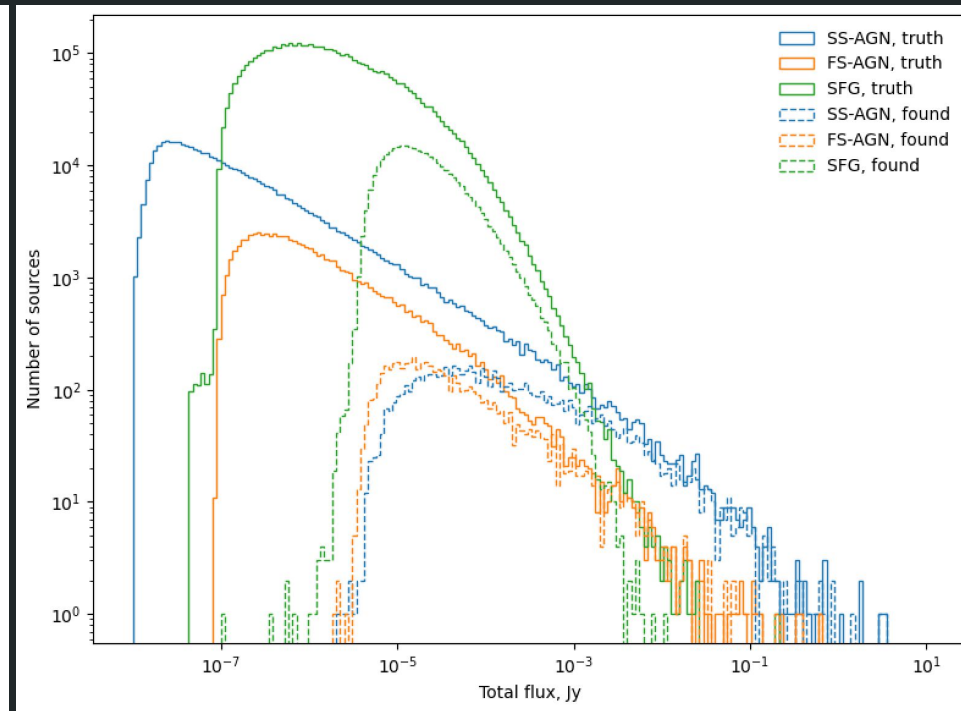
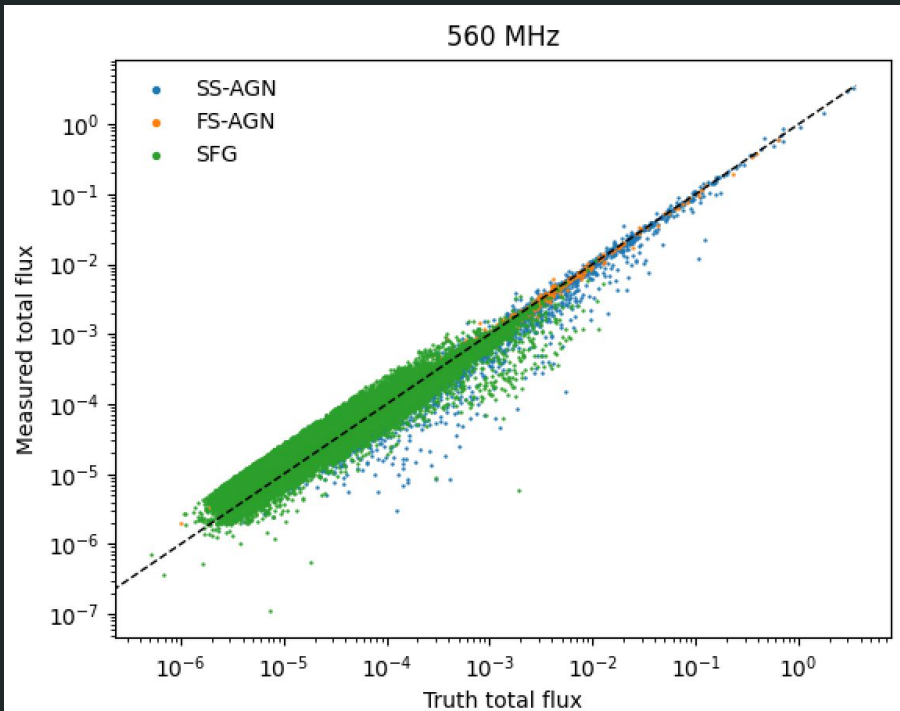
## Running the example solution

1. Image preprocessing
  - Primary beam correction
  - Split out training sub-image
  - (Chunk images)
2. Source finding (training)
3. Model fitting (training)
4. Source finding (full)
5. Model prediction (full)
6. Score results



# Science Data Challenge 1

Running the example solution





# Precision, recall, F1 score for the Random Forest (RF) - 560 MHz

Fit RF to half of the sources in the training area, test on other half:

	precision	recall	f1-score	support
FSAGN	0.8250	0.2426	0.3750	136
SFG	0.9801	0.9984	0.9892	10424
SSAGN	0.5926	0.1280	0.2105	125

Apply this model to classify the rest of the sources in the field:

	precision	recall	f1-score	support
FSAGN	0.8453	0.2383	0.3718	5342
SFG	0.9714	0.9964	0.9838	328561
SSAGN	0.4993	0.1575	0.2395	6736

Run the scoring code with these class labels:

```
Score was 231607.00169231306
Number of detections 343272
Number of matches 340639
Number of matches too far from truth 1433
Number of false detections 2633
Score for all matches 234240.00169231306
Accuracy percentage 68.7648806191637
```

The F1 score is the harmonic mean of precision and recall, interpreting true positives (TP), false positives (FP) and false negatives (FN):

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

# Precision, recall, F1 score for the Random Forest (RF) - 560 MHz

Fit RF to half of the sources in the training area, test on other half:

	precision	recall	f1-score	support
FSAGN	0.8250	0.2426	0.3750	136
SFG	0.9801	0.9984	0.9892	10424
SSAGN	0.5926	0.1280	0.2105	125

Apply this model to classify the rest of the sources in the field:

	precision	recall	f1-score	support
FSAGN	0.8453	0.2383	0.3718	5342
SFG	0.9714	0.9964	0.9838	328561
SSAGN	0.4993	0.1575	0.2395	6736

Run the scoring code with these class labels:

```
Score was 231607.00169231306
Number of detections 343272
Number of matches 340639
Number of matches too far from truth 1433
Number of false detections 2633
Score for all matches 234240.00169231306
Accuracy percentage 68.7648806191637
```

This is a hugely imbalanced classification problem. Let's compare scores for guessing against the RF model:

Model	SDC Score	Average F1-score
RF:	231 607	0.53
Guess all SFG:	231 475	0.33
Guess all SS-AGN:	188 275	0.01

Guessing they are all SFG will result in much worse precision, recall and F1 scores:

	precision	recall	f1-score	support
FSAGN	0.0000	0.0000	0.0000	5342
SFG	0.9645	1.0000	0.9820	328561
SSAGN	0.0000	0.0000	0.0000	6736

The inclusion of the class label in the scoring package gives a score biased towards finding star-forming galaxies. Thus the score from the scoring package is not a fair representation of overall performance, it is only fair at assessing the identification of sources, not their class.

F1 scores are a fair representation of the performance of a multiclass, imbalanced classification scheme.

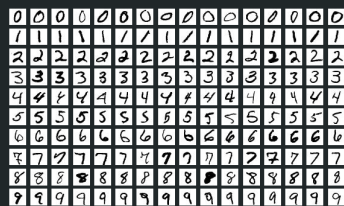
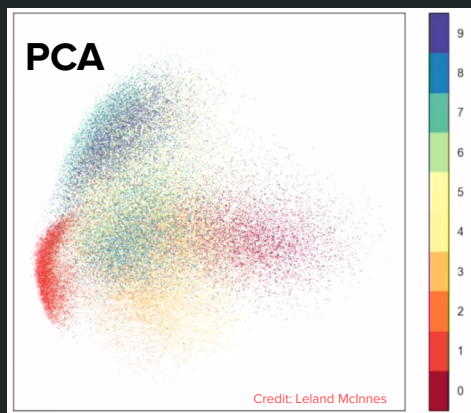
# Dimension reduction

- High dimensional data can often be represented extremely well in a lower dimensional space
- High dimensional data opens the door for complex models to fail or have undetected inconsistencies/biases
- Our existence is based in three dimensions. It is unnatural to expect us to have any significant intuition beyond this
- Visualisation is everything. You will reap the rewards of being able explain your data and models to other people intuitively

# Dimension reduction

## Matrix Factorisation

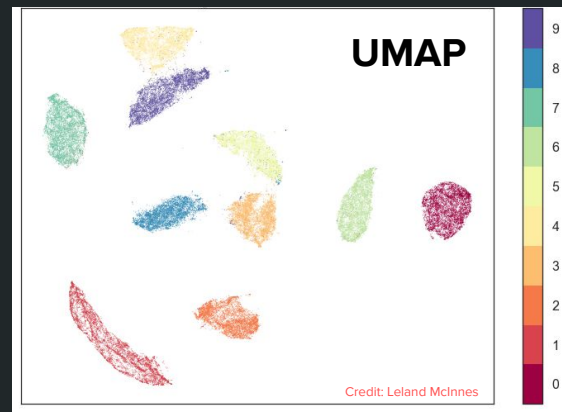
- Principal Component Analysis
- Latent Dirichlet Allocation
- Linear auto-encoder



Hand drawn digits dataset (MNIST dataset)

## Neighbour Graphs

- Isomap
- t-SNE
- UMAP



I would highly recommend Leland McInnes' talk which goes over the basic mathematical concepts behind matrix factorisation and neighbour graphs: <https://www.youtube.com/watch?v=9iol3Lk6kyU>

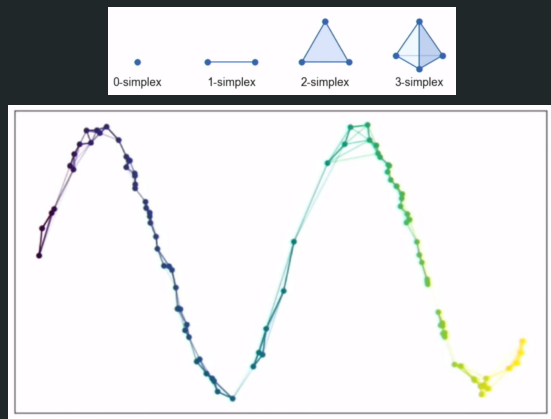
# Dimension reduction

## Neighbour Graphs: Uniform Manifold Approximation and Projection

UMAP: McInnes et al

<https://arxiv.org/abs/1802.03426>

- 1) Compute a graphical representation of a dataset (Topology: fuzzy simplicial complex)
- 2) Through stochastic gradient descent, optimise a low-dimensional embedding of the graph.



Parametric UMAP: replace step 2 with a neural network that learns a parametric embedding

### PARAMETRIC UMAP: LEARNING EMBEDDINGS WITH DEEP NEURAL NETWORKS FOR REPRESENTATION AND SEMI-SUPERVISED LEARNING

Tim Sainburg  
UC San Diego  
tsainbur@ucsd.edu

Leland McInnes  
Tutte Institute for Mathematics  
and Computing  
leland.mcinnnes@gmail.com

Timothy Q Gentner  
UC San Diego  
tgentner@ucsd.edu

#### ABSTRACT

We propose Parametric UMAP, a parametric variation of the UMAP (Uniform Manifold Approximation and Projection) algorithm. UMAP is a non-parametric graph-based dimensionality reduction algorithm using applied Riemannian geometry and algebraic topology to find low-dimensional embeddings of structured data. The UMAP algorithm consists of two steps: (1) Compute a graphical representation of a dataset (fuzzy simplicial complex), and (2) Through stochastic gradient descent, optimize a low-dimensional embedding of the graph. Here, we replace the second step of UMAP with a deep neural network that learns a parametric relationship between data and embedding. We demonstrate that our method performs similarly to its non-parametric counterpart while conferring the benefit of a learned parametric mapping (e.g. fast online embeddings for new data). We then show that UMAP loss can be extended to arbitrary deep learning applications, for example constraining the latent distribution of autoencoders, and improving classifier accuracy for semi-supervised learning by capturing structure in unlabeled data.

See this talk for an overview:

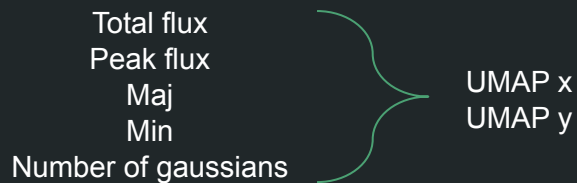
<https://jbca-machinelearning.github.io/journalclub.html#6>

I would highly recommend Leland McInnes' talk which goes over the basic mathematical concepts behind matrix factorisation and neighbour graphs: <https://www.youtube.com/watch?v=9iol3Lk6kyU>

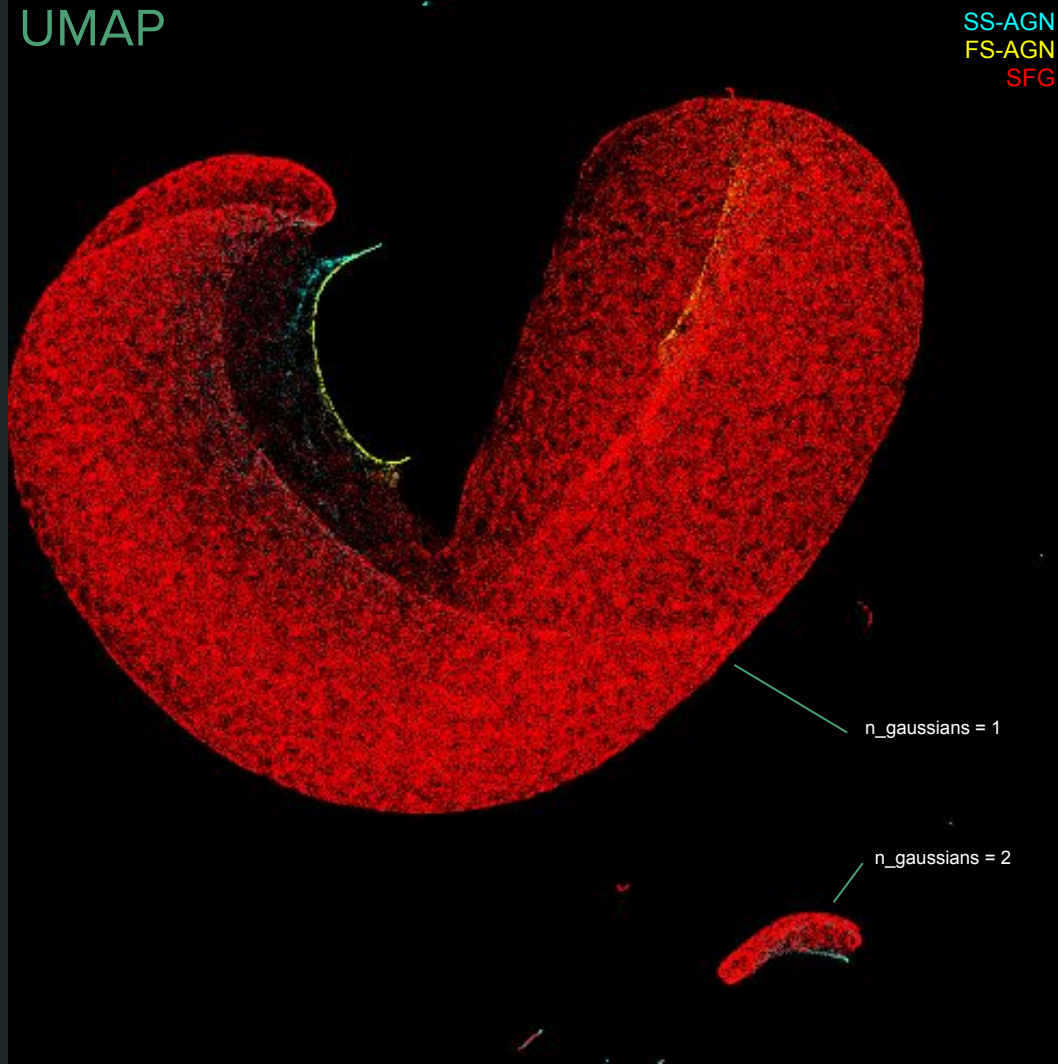
# Visualising the classifications with UMAP

Uniform Manifold Approximation and Learning (UMAP) is a non-linear dimension reduction algorithm.

We use it to reduce the number of features from 5 to 2



Colour code this plot by truth class label to see if the spatial separation is correlated with class





# Visualising the classifications with

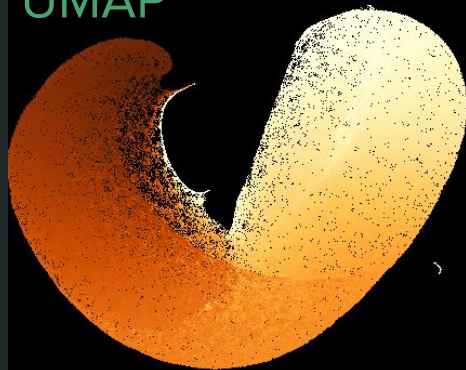
Can colour the plot by features or other parameters to investigate their distribution or correlations with class label or beyond

Akin to looking at histograms over all parameters

## UMAP

Source Area

Aspect Ratio



Small values

Large values

SS-AGN  
FS-AGN  
SFG



Total Flux



| Total Flux - Peak Flux |



Thanks to the datashader package for visualising so many sources - the pixel brightness is proportional to number of sources counted in that pixel (log scale)

# Science Data Challenge 1

## Learning from the example solution

- Hopefully this helps share knowledge about technologies and techniques that enable more reproducible results, and people can use this example for future data challenges
- Future science data challenges will test the reproducibility of results
- Technologies like containers will become essential for future SKA data challenges, where you bring your software pipelines to the data at regional centres



# Science Data Challenge 2

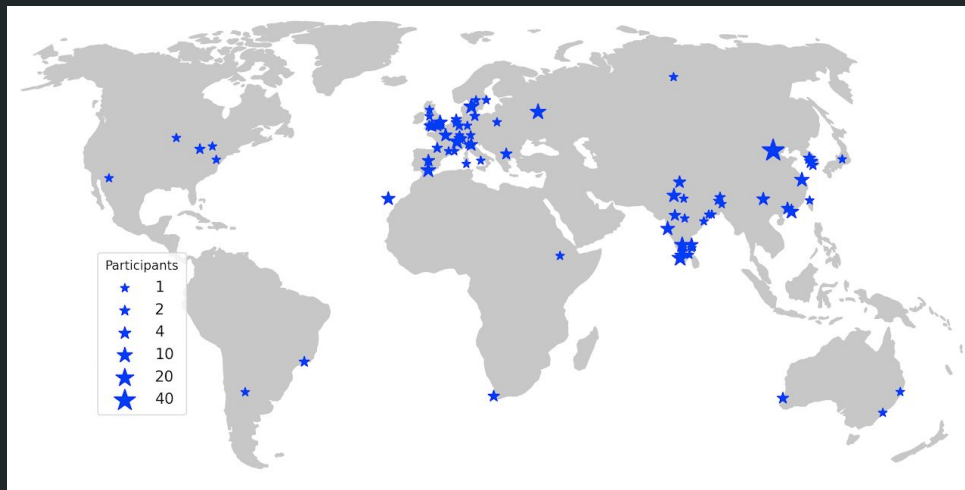
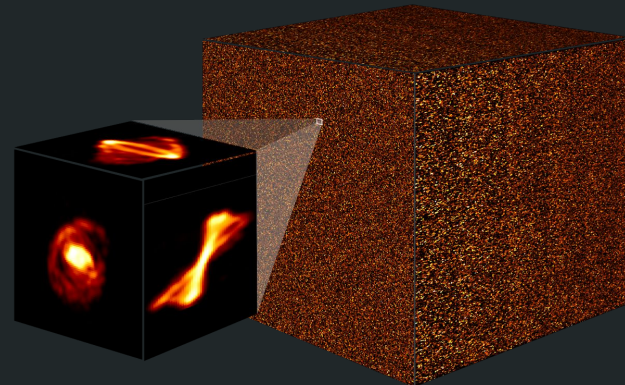
## Source finding in a 1TB image cube

- A simulated 3D image cube of the neutral hydrogen line, where participants must find sources.
- Each team is given access to a computational resource provider acting as a regional centre

(IRIS, INAF ICT, IAA-CSIC proto-SRC, GENCI-IDRIS, EngageSKA-UCLCA, CSCS-Piz Daint, China SKA proto-SRC, AusSRC-Pawsey)

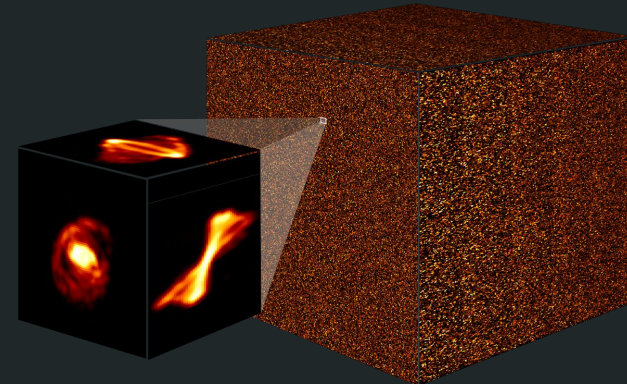
- The cube is hosted at these facilities and teams bring their pipelines to the data.
- 40 teams, 276 participants, representing 80 institutes and 23 countries
- Awards will be given out for demonstrating reproducible results:

<https://sdc2.astronomers.skatelescope.org/sdc2-challenge/reproducibility-awards>



# Science Data Challenge 2

## Source finding in a 1TB image cube



- A simulated 3D image cube of the neutral hydrogen line, where participants must find sources.
- Each team is given access to a computational resource provider acting as a regional centre

(IRIS, INAF ICT, IAA-CSIC proto-SRC, GENCI-IDRIS, EngageSKA-UCLCA, CSCS-Piz Daint, China SKA proto-SRC, AusSRC-Pawsey)

- The cube is hosted at these facilities and teams bring their pipelines to the data.
- 40 teams, 276 participants, representing 80 institutes and 23 countries
- Awards will be given out for demonstrating reproducible results:

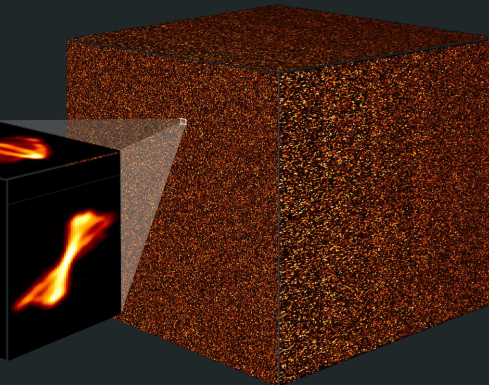
<https://sdc2.astronomers.skatelescope.org/sdc2-challenge/reproducibility-awards>

Reproducibility of the solution	
Can the software pipeline be re-run easily to produce the same results? Is it:	
<ul style="list-style-type: none"> <li>Well-documented <a href="#">Research software documentation best practice</a></li> <li>Easy to install <a href="#">Top tips for packaging software</a></li> <li>Easy to use <a href="#">Top tips for documentation</a></li> </ul>	
Well-documented	High-level description of what/who the software is for is available
	High-level description of what the software does is available
	High-level description of how the software works is available
	Documentation consists of clear, step-by-step instructions
	Documentation gives examples of what the user can see at each step e.g. screenshots or command-line excerpt
Easy to install	Documentation uses monospace fonts for command-line inputs and outputs, source code fragments, function names, class names etc
	Documentation is held under version control alongside the code
	Full instructions provided for building and installing any software
	All dependencies are listed, along with web addresses, suitable versions, licences and whether they are mandatory or optional
	All dependencies are available
Easy to use	Tests are provided to verify that the installation has succeeded
	A containerised package is available, containing the code together with all of the related configuration files, libraries, and dependencies required. Using <i>e.g. Docker/Singularity</i>
	A getting started guide is provided outlining a basic example of using the software <i>e.g. a README file</i>
	Instructions are provided for many basic use cases
	Reference guides are provided for all command-line, GUI and configuration options

Reusability of the pipeline	
Can the code be reused easily by other people to develop new projects? Does it:	
<ul style="list-style-type: none"> <li>Have an open licence <a href="#">Choosing an open source licence</a></li> <li>Have easily accessible source code <a href="#">Choosing a repository for your project</a></li> <li>Adhere to coding standards <a href="#">Writing reproducible source code</a></li> <li>Utilise tests <a href="#">Testing your software</a></li> </ul>	
Open licence	Software has an open source licence <i>e.g. GNU General Public License (GPL), BSD 3-Clause</i>
	Licence is stated in source code repository
Accessible code	Each source code file has a licence header
	Access to source code repository is available online
Code standards	Repository is hosted externally in a sustainable third-party repository <i>e.g. SourceForge, LaunchPad, GitHub: <a href="#">Introduction to GitHub</a></i>
	Documentation is provided for developers
Testing	Source code is laid out and indented well
	Source code is commented
	There is no commented out code
	Source code is structured into modules or packages
	Source code uses sensible class, package and variable names
	Source code structure relates clearly to the architecture or design
	Source code has unit tests
	Software recommends tools to check conformance to coding standards <i>e.g. A 'linter' such as PyLint for Python</i>

# Science Data Challenge 2

## Reproducibility Awards



	Reproducibility of the solution Can the software pipeline be re-run easily to produce the same results? Is it: <ul style="list-style-type: none"> <li>Well-documented <a href="#">Research software documentation best practice</a></li> <li>Easy to install <a href="#">Top tips for packaging software</a></li> <li>Easy to use <a href="#">Top tips for documentation</a></li> </ul>	
	Well-documented	High-level description of what/who the software is for is available High-level description of what the software does is available High-level description of how the software works is available Documentation consists of clear, step-by-step instructions Documentation gives examples of what the user can see at each step e.g. screenshots or command-line excerpt Documentation uses monospace fonts for command-line inputs and outputs, source code fragments, function names, class names etc Documentation is held under version control alongside the code
	Easy to install	Full instructions provided for building and installing any software All dependencies are listed, along with web addresses, suitable versions, licences and whether they are mandatory or optional All dependencies are available Tests are provided to verify that the installation has succeeded A containerised package is available, containing the code together with all of the related configuration files, libraries, and dependencies required. Using .e.g. Docker/Singularity
	Easy to use	A getting started guide is provided outlining a basic example of using the software e.g. a README file Instructions are provided for many basic use cases Reference guides are provided for all command-line, GUI and configuration options

	Reusability of the pipeline Can the code be reused easily by other people to develop new projects? Does it: <ul style="list-style-type: none"> <li>Have an open licence <a href="#">Choosing an open source licence</a></li> <li>Have easily accessible source code <a href="#">Choosing a repository for your project</a></li> <li>Adhere to coding standards <a href="#">Writing readable source code</a></li> <li>Utilise tests <a href="#">Testing your software</a></li> </ul>	
	Open licence	Software has an open source licence e.g. GNU General Public License (GPL), BSD 3-Clause Licence is stated in source code repository Each source code file has a licence header
	Accessible code	Access to source code repository is available online Repository is hosted externally in a sustainable third-party repository e.g. SourceForge, LaunchPad, GitHub: <a href="#">Introduction to GitHub</a> Documentation is provided for developers
	Code standards	Source code is laid out and indented well Source code is commented There is no commented out code Source code is structured into modules or packages Source code uses sensible class, package and variable names Source code structure relates clearly to the architecture or design
	Testing	Source code has unit tests Software recommends tools to check conformance to coding standards e.g. A 'linter' such as PyLint for Python

# Conclusions

- Data challenge 1 was a successful starting point, 9 teams competed, we developed an example solution afterwards to help guide the community
- Data challenge 2 is running now (Feb-July) with 40 teams, stepping up data size and workflow expectations
- We hope these data challenges will help prepare people for the kind of workflows that will be necessary with SKA data (but also relevant for any scientific data analysis)
- A great platform for spreading the word amongst the community about relevant technologies, efficient workflows, best practices in coding, publishing data and code, reproducibility
- Also helping a lot with our work on prototyping and developing technologies for the SKA regional centres

Alex Clarke (SKAO)

[a.clarke@skatelescope.org](mailto:a.clarke@skatelescope.org)

On behalf of the team: Anna Bonaldi, Phillipa Hartley, Rosie Bolton, James Collinson, Rob Barnsley, Rohini Joshi

Thanks for listening