D3.3.1: LSST/TiDES Metrics Software

WP3.3 Spectroscopic classification of transients

Project Acronym	LUSC-B
Project Title	UK Involvement in the Legacy Survey of Space and Time
Document Number	LUSC-B-17

Submission date	26/OCT/2021
Version	1.0
Status	Approved
Author(s) inc. institutional affiliation	Christopher Frohmaier (Southampton) Mark Sullivan (Southampton)
Reviewer(s)	Stephen Smartt (QUB), Matt Nicholl (Birmingham)

Dissemination level	
Public	

Version History

Version	Date	Comments, Changes, Status	Authors, Contributors, Reviewers
0.1	14/JUL/21	First Draft	
1.0	25/AUG/21	Reviewer comments addressed	
1.0	26/OCT/21	Approved and Final version submitted	

Table of Contents

List of Figures

List of Tables

1 Executive Summary

The Legacy Survey of Space and Time (LSST) and the 4m multi-object spectroscopic telescope (4MOST) commence survey operations in 2023. The Time-Domain Extragalactic Survey (TiDES) will follow-up LSST discovered transients to obtain spectroscopic measurements for tens-of-thousands of supernovae, galaxies, and active-galactic nuclei (AGN). This additional data will allow us to map the astrophysical diversity of cosmic explosions, measure the equation of state parameter for dark energy to unprecedented precision, and perform a comprehensive AGN reverberation mapping experiment. TiDES forms the basis of WP3.3: Spectroscopic classification of transients.

The strategies of both the LSST and 4MOST surveys are yet to be finalised and, with each acting independently, there exists the potential for a disjointed operation. TiDES operates downstream from LSST, spectroscopically following-up any transients it discovers. In D3.3.2 [1] we present our submitted 'Cadence Note' to the LSST Survey Cadence Optimization Committee (SCOC) with recommendations for a strategy suitable for TiDES science goals. In this document, we present an overview of the software we developed to reach our conclusions presented in D3.3.2. Any future simulations produced by LSST and/or 4MOST can also be evaluated following the prescription presented here.

2 Introduction

Work Package 3.3 focuses on the spectroscopic classification of transient events from LSST using the 4MOST/TiDES facility. These transients will be predominantly supernovae (SNe) but, given the industrial scale at which TiDES will operate, more exotic phenomena such as tidal-disruption events and binary neutron star collisions may be observed. The success of both LSST and TiDES will rely on an inter-play of strategies between the two surveys. This software package provides many convenience functions and routines to evaluate the performance of LSST and 4MOST. A flowchart is shown in Figure 1 to schematically detail how we go from LSST and 4MOST simulation files to the end-goal of a TiDES performance metric. In this flow chart, elements above the pink dashed line are from external resources that we are not responsible for maintaining or producing. We do, however, utilise their data products and thus stages below the pink line are included in this deliverable. All the code presented in this document are available from the following GitHub repository: https://github.com/chrisfrohmaier/tides_cadence_note

Our report is presented as follows: in Section 3 we lay out the system requirements and prerequisites for running this software. In Section 4 we detail how SNANA output files are processed into an usable format. In Section 5 we briefly discuss the 4MOST simulation files and how they are integrated into our pipeline. Section 6 presents the software to create a union of LSST and 4MOST simulations. Finally, in Section 7 we present the scripts to reproduce the metrics used in the TiDES Cadence Note.

2.1 Glossary of Acronyms

4MOST	**4**m **M**ulti-**O**bject **S**pectroscopic **T**elescope
DESC	**D**ark **E**nergy**S**cience **C**ollaboration
LSST	**L**egacy **S**urvey of **S**pace and **T**ime
SCOC	**S**urvey **C**adence **O**ptimization **C**ommittee
SNANA	**S**uper**N**ova **ANA**lysis
TiDES	**T**ime-**D**omain **E**xtragalactic **S**urvey

2.2 LSST Operations Simulator

The LSST Operations Simulator[1] is a software package capable of simulating field selection and image acquisition for the planned 10-year lifetime of the survey. The simulator outputs the detailed properties of each image it acquires based on statistical properties of weather patterns, anticipated engineering downtime, the camera response, and the telescope's performance. The result of this is a detailed file that allows the user to perform a realistic analysis of the impact of the LSST strategy from single-images to a whole 10-year survey key project. Any user can create their own OpSim output file, although the survey strategies are comprehensively covered and publicly available from LSST.[2]

The data products from LSST are in an `SQLite` format which is incompatible with our supernova simulation software SNANA[2]. The LSST OpSim files are converted to an SNANA-compatible file by the `OpSimSummary`[3] package. SNANA is hosted on the Cori supercomputer under the DESC project affiliation and we process the files under their standard prescription described in [4]. SNANA outputs the simulations of supernova properties and light curves into multiple `.fits`. While these files are certainly usable, for a full sky analysis it is ultimately impractical to use this output in its current state. We, therefore, created a script to process SNANA outputs into a SQL database which we could optimise for our analysis. This script is presented in Section 4.

[1] https://github.com/lsst/sims_featureScheduler
[2] http://astro-lsst-01.astro.washington.edu:8081

2.3 4MOST Tiling Files

Independently of LSST, 4MOST perform their own mock survey simulations. There are 10 consortium surveys within 4MOST, each with different science requirements, galactic or extra-galactic targets, and with a varying share of the total available fibre-hours. TiDES is somewhat unique within the consortium surveys; our targets have a time dependence meaning that it is critical to map not just where 4MOST is pointing on the sky, but also *when* those observations are made. This is most pronounced for our supernova targets — which are fleeting in nature — but it also affects our galaxy targets as we select them after their supernova has faded away. As a result, TiDES operates as a 'piggy-back'survey, observing the chance-encounter supernovae and galaxies that fall within the field-of-view of each 4MOST exposure.

At the time-of-writing, the 4MOST simulations do not have the sophistication to handle the time-domain requirement of the TiDES targets. The output of the 4MOST simulation is a file containing the locations and epochs of all observations made during the 5-year survey lifetime. For most other consortium surveys, the total integrated exposure time is the key metric of survey success as it determines the total number of targets obtained. This focuses the algorithmic balance of the tiling solution between total sky coverage and depth given the finite fibre-hour resource available. A detailed breakdown of how the optimisation of the tiling solution performed is presented in [5].

In Section 5 we summarise our code that takes the 4MOST simulation output files and ingests them into a database. This pre-processing stage is crucial in later stages of this deliverable where we perform a time-dependent cross-match between LSST and 4MOST.

3 Requirements

The software products developed as part of this deliverable are written in three languages. The general pipeline execution, interaction with the operating system, and resource management are handled in `bash`. The large data management, storage, and processing tasks are administered through `PostgreSQL`. Finally, jobs requiring more complex processing and interaction with external libraries are undertaken in `Python`.

This entire pipeline was developed in a Linux environment and tested on both Ubuntu 16.04 and macOS 11.4. Given the system interactions, this pipeline will only work on Unix systems and is therefore incompatible with Windows.

3.1 PostgreSQL

A local install of `PostgreSQL` is required. The pipeline was developed on v12.1[3], but is compatible with other versions $\geq$ v9.0. The pipeline utilises a local install due to the large I:O requirements in processing and transferring the tens-of-gigabytes of SNANA simulation results into the database. The data analysis scripts written in Python can be executed remotely if the host address of the PostgreSQL server is known. A database called `tides` must also exist for data to be stored in. The easiest way to create a database is to execute

```
% CREATE DATABASE tides
```

within the `psql` interface. Additional arguments are available to exercise more control over the database creation[4].

[3] https://www.postgresql.org/about/news/postgresql-12-released-1976/
[4] https://www.postgresql.org/docs/9.0/sql-createdatabase.html

Finally, cross-matching between all-sky catalogues requires spherical geometry calculations to me made on the celestial sphere. The extension `Q3C` is required for this task. Please install this package from the github repo: https://github.com/segasai/q3c.

3.2 Python

All `python` code developed in this deliverable was written in v3.8, although it should be compatible with any version ≥ 3.0. It utilises several inbuilt modules, but also makes extensive use of external packages. The following list are presented with the version used in development:

- astropy = 4.1
- numpy = 1.19.5
- pandas = 0.2
- sqlalchemy = 1.1
- psycopg2 = 2.7
- yaml = 3.12

3.2.1 Password protected database access

The python scripts to access the SQL databases may require login credentials. A simple login file in the `yaml` format is used and stored away from the public-facing directories e.g. GitHub. An example login file, with dummy credentials, is shown below.

```
tidesdb:
    username: RFederer
    password: 20GrandSlams
    host: cannon.phys.soton.ac.uk
    port: 5432
    db_name: tides
```

It is necessary to create one of these files to store your `PostgreSQL` login credentials. The appropriate python scripts must then be edited to point to the location of this file.

4 Software to process SNANA results

This section details the ingestion of the SNANA data files in the PostgreSQL database. It is represented by the stage 'S1 - Upload SNANA data files' in Figure 1. The code also performs general database maintenance and optimisation for our later applications. Directories in this section will be referenced from the GitHub base directory `tides_cadence_note`.

The directory is populated as follows:

```
./catUploadScripts/
|-- dropTablesTiDES_DB.sh
|-- findSNANAFiles.sh
|-- makeCSVfromSNANA.py
|-- makeTables.sql
|-- makeTiDESdb.sh
|-- postUploadDBMaintence.sql
|-- spatialIndexCluster.sql
`-- upload2DB.sql
```

The pipeline is run from the `makeTiDESdb.sh` bash script and here we breakdown the use of each file in order of execution.

- `makeTiDESdb.sh`: A bash shell script to interact with the operating system. It takes as input the absolute path of the SNANA data files and the number of processors the system should allocate to the whole pipeline. When the script is first launched, the user is asked to input a 'prefix name' for their tables. For example, `baseline_1pt7` would be appropriate for the baseline v1.7 family of LSST simulations. The total execution time of the pipeline can total many hours due to the large file I:O operations. The following list details the scripts executed in this routine:.
  - `makeTable.sql`: An SQL script to create 6 different tables to store the SNANA `HEAD`, `PHOT`, and summary data. It creates 3 tables for the Wide-Fast-Deep survey and 3 for the Deep-Drilling Surveys.
  - `makeCSVfromSNANA.py`: This python script pre-processes the SNANA `HEAD` and `PHOT` files using the `pandas` package. It strips out unnecessary data columns for our analysis and performs checks to ensure the data is compatible with the database. It creates CSV output files that are stored in the same directories as the SNANA simulation files. As each SNANA data file is independent, multiple instances of this script are launched dictated by the `-n` flag in the master bash script.
  - The CSV files are uploaded to the PostgreSQL database from within the main script.
  - `postUploadDBMaintence.sql`: This SQL script is executed after all data is uploaded to the database. It includes the creation of several indexes.
  - `spatialIndexCluster.sql`: Finally, this SQL script creates a spatial index on columns with sky coordinates. This uses the `Q3C` library.
- `findSNANAfiles.sh` and `upload2DB.sql` are utility scripts that replicate some of the internal processes in `makeTable.sql`. They are useful if individual files need to be processed without running the entire pipeline.
- `dropTablesTiDES_DB.sh`: This script enables the user to delete every table with a common `<prefix>` in the database. Several confirmation commands are required to ensure the user deletes only the requested tables.

After the execution of this pipeline, the user will have 6 new tables in their PostgreSQL database. These tables are split by Wide-Fast-Deep (wfd) and Deep-Drilling Fields (ddf). An example if presented below.

```
 Schema |                  Name                  |   Type   | Owner
--------+----------------------------------------+----------+--------
 public | prefix_1pt7_ddf_sn_head                | table    | cf5g09
 public | prefix_1pt7_ddf_sn_phot                | table    | cf5g09
 public | prefix_1pt7_ddf_sn_summary             | table    | cf5g09
 public | prefix_1pt7_wfd_sn_head                | table    | cf5g09
 public | prefix_1pt7_wfd_sn_phot                | table    | cf5g09
 public | prefix_1pt7_wfd_sn_summary             | table    | cf5g09
```

4.1 Example Execution

Here we present an example of how to execute the `makeTiDESdb.sh` script. We assume SNANA data directories are in the `/data/cf5g09/tides/SNANA_Data/` path which is the argument for the `-p` flag. We also assign 10 to the `-n` flag to state our maximum allocation of 10 cores to the processing. Finally, the prefix name is taken as an interactive user input, in this case we assign the dummy name `d331_example`.

```
$ ./makeTiDESdb.sh -p /data/cf5g09/tides/SNANA_Data/ -n 10


Search path: /data/cf5g09/tides/SNANA_Data/
Found files, saved in files2convertFITStoCSV_{TRANSIENT_TYPES}.txt
-------
Please enter the prefix name for the database tables:
(e.g. s10april2020)
--- Note: table prefix will be converted to lowercase ---
--- Note2: Please indicate if this is a 4MOST candence upload in
the prefix
e.g. s10april2020_4most
d331_example
```

Several messages are printed to the terminal during the subsequent execution, but no further user interaction is required.

5 Software to Process 4MOST Tiles

This sections details the scripts that upload the 4MOST Tiles to a `PostgreSQL` database. It is represented by the stage 'S2 - Upload 4MOST tile files' in Figure 1. The 4MOST tiling files are privately hosted and available to 4MOST consortium members. Specifically, these files are available from this address: https://4most.mpe.mpg.de/IWG2/, under the 'Simulation Data' column. The available data includes not only the 4MOST tiles, but also the other survey targets, and the allocation of fibre-hours in each simulation run. 4MOST break-down their simulation data into sub-directories based on the date of the simulation ({`MONTH`}{`YEAR`}) and the numerical ID of the simulation run. Since all simulations are uploaded into the same table, this date and ID (`monyear` and `vid` in the database) are used to distinguish different simulations.

The code directory structure is as follows:

```
./qmostUploadScripts/
|-- createTileTable.sql
|-- uploadQMOSTInputCatalogue.py
|-- uploadQMOSTtargets.py
`-- uploadQMOSTtiles.py
```

- `createTileTable.sql`: This script is run to create the table in the tides database that will host all subsequent 4MOST tile uploads. **Note:** it only needs to be executed once to create the table and not every time new data is ingested. The file is executed from the commandline as follows `psql tides -f createTileTable.sql`
- `uploadQMOSTtiles.py`: This python script reads the input simulations, organises the data in a `pandas` table, and then uploads the result into the tides database.

Other scripts in this directory are able process the additional data from 4MOST. However, the data are not used nor are they relevant for this deliverable.

5.1 Example execution of Python script

Here we show an example of how to run the Python script to upload the 4MOST tiling data.

```
$ python uploadQMOSTtiles.py -p /path/to/tile/file/qmost_out_tiles_sim00.fits
-my may21 -vr 2 -c /path/to/credentials/dbLogin.yml
```

The flags are explained as follows:

- `-p`: The full path to the 4MOST simulation files.
- `-my`: The month-year of the 4MOST simulation, e.g. `may2021`.
- `-vr`: The version of the simulation. This number is an integer that increments for each new simulation release in the month-year directory.
- `-c`: The full path to the database login credentials file explained in Section 3.2.1.

After the successful execution of this script, the `qmost_tiles` database will be populated with the 4MOST tiling pattern for the 5-year survey.

6 Cross-Matching LSST and 4MOST

At this stage, we have kept the LSST and 4MOST tables separate from each other. In this section, we detail the SQL script that performs the cross-match to create a sample of SNe and galaxies that 4MOST would have observed. This stage forms 'S3 - Cross-match LSST and TiDES' of Figure 1. This SQL script is a complex, nested-query that performs several key sub-tasks that are mapped out in the flowchart presented in Figure 2. The codes available for these tasks are executed to create two data outputs. One for the live-SN sample and a second for the host-galaxy sample. The TiDES metrics software in Section 7 is then run independently on each. These scripts in this section can be found in the `./catalogues4FS` directory.

While the SQL may appear complex in the available code, the principles behind it are simple. To construct the live SN sample, we first select all SNe in the LSST simulations with ≥ 2 nights of a 5σ detections in at least 3 different filters. We then filter that sample to only include those with ≥ 2 epochs brighter than the single-OB 4MOST magnitude limit (22.5 mag), this is a minimum criterion to ensure we realistically could obtain a good quality spectrum. The date of the second detection is nominally designated as the 'Trigger date' on which 4MOST becomes aware of the target. The next stage introduces the 4MOST tiles by querying which of the remaining SN targets fall within a field-of-view *after* the object's trigger date. All transients have a nominal lifetime of 5 days in the 4MOST queue, after each subsequent LSST detection (brighter than 22.5 in any band) the target is refreshed for a further 5 days. This ensures that old and faded targets do not have fibres placed on them, thus minimising wasted fibre hours. We assert that any SN targets within the 4MOST field-of-view with a recently refreshed and 'alive' status were successfully observed.

The query for the host-galaxy sample has a similar algorithmic background. However, we exclude the need for the SN to be above the 4MOST detection limit and instead require this of the host-galaxy. Additionally, the trigger date is now defined as the epoch when the SN finally fades below the 4MOST limit. This was chosen to limit the impact of SN flux contaminating the galaxy spectrum.

A natural consequence of these queries is that the live-SN catalogue is a subset of the host-galaxy catalogue. The additional targets in the host-galaxy catalogue come from SN that remained faint, never reaching 4MOST detection limits, but were observed to high confidence by LSST e.g. in the Deep-Drilling Fields. In the following section, we demonstrate how to run the SQL scripts.

6.1 Running the cross-matching queries

An example execution of the script is shown below. It should be noted that when Postgres is called from the command line, variables are set using the `-v` flag followed by the assignment.

```
$ psql tides -v p=/path/to/save/output/ -v prefix=<prefix> -v field=<field>
 -v suffix=<suffix> -v qvid=<vr> -v monyear=<my> -f script.sql
```

The flags explained as follows:

- `p`: The output destination path for the resulting csv files from the query.
- `prefix`: The prefix of the LSST table you wish to query. This is the same prefix that was previously defined from the interactive input in Section 4.1.
- `field`: either `wfd` or `ddf` for Wide-Fast-Deep or Deep-Drilling Fields respectively.
- `suffix`: This is a user-defined string that is appended to the output's filename to help distinguish it from other executions of the script. A small description like `'cadenceNoteMay2020'` is a useful example.
- `qvid`: This is the 4MOST tiling solution version ID used for the cross-matching. It takes a value defined in the catalogue ingestion shown in Section 5.1.
- `monyear`: The 4MOST tiling solution month-year identifier explained in Section 5.1.
- `-f`: This flag is used to point to the SQL script to execute. In the current directory, the `indexedCrossMatchedIndexedLSSTand4MOST.sql` will perform the live-SN analysis and `hostgalaxy_indexedCrossMatchedIndexedLSSTand4MOST.sql` for the host-galaxy analysis.

After the script has successful completed, the output directory will be populated by the csv files containing the information we need to finally evaluate the performance of TiDES in the next section.

7 Software to evaluate TiDES metrics

Two scripts are provided in the `./cadenceNoteMetrics` directory and represent 'S4 - TiDES Metrics' in the flowchart of Figure 1. One to analyse the live SN sample and the other to analyse the host-galaxy sample. The goal of this final data processing stage is to create an output capable of reproducing the results presented in the TiDES Cadence Note in D3.3.2. The software presented here will process just one combination of an LSST and TiDES survey strategy, however, our Cadence Note presented several and performed a relative analysis. In order to replicate this, the entire software package must be executed on the many desired LSST and TiDES input files.

7.1 Metric Outputs

In this final step, we will use the csv files generated in Section 6.1 as the inputs. An example execution of the live-SN script is shown below.

```
$ python liveSN_Metrics.py -in /path/to/input/file.csv -outdir
/path/to/save/output/
```

The analysis of the host-galaxy data is performed identically, except the `liveSN_Metrics.py` file is swapped for `hostGalaxyMetrics.py`. The inputs are as follows:

- `-in`: The path to the csv file for the analysis. This is the csv output from Section 6.1.
- `outdir`: The path where the analysis output files will be stored. It is strongly recommended that when you analyse different simulation strategies, you output to different directories to prevent overwriting of previous results.

7.1.1 Output for live SNe

The live supernova analysis will produce the following output files:

- `liveSNmetrics.txt`: This file states, for each simulated SN sub-type, the total number of 4MOST spectra obtained.
- `numberRedshiftBinsLive.pdf`: A redshift-histogram, broken-down by SN sub-type.
- `nzHistogramLiveSNe.csv`: The raw data used to create the histogram.

7.1.2 Output for Host-Galaxies

The host-galaxy analysis presents several additional metrics over the live-SNe as we analyse some light curve parameters too. Furthermore, we also analyse a sample of cosmologically useful SN Ia and present some statistics on their discovery. The metrics calculated in this section were used to generate Figure 2 in D3.3.2:

- `hostGalaxymetrics.txt`: This file states, for each simulated SN sub-type, the total number of 4MOST spectra obtained.
- `hostGalaxy_Ia_stats.txt`: The columns in this file are:'Metric' , 'Mean uncertainty value' , 'Standard Deviation' . The metrics are derived from SALT2 fitting to the SN Ia light curves and describe the uncertainty on peak date estimate, uncertainty on the distance modulus, SALT2 color and x1 uncertainty. Additionally, the mean number of pre- and post-peak 5σ detections are provided.
- `numberRedshiftBinsHostGalaxy.pdf`: A redshift-histogram, broken-down by SN sub-type.
- `nzHistogramHostgalaxySNe.csv`: The raw data used to create the histogram.

From all the processing steps, output files, and survey metrics presented in this document, we have provide a complete pipeline to analyse any combination of LSST and 4MOST survey strategy. This software stack is compatible with future releases of survey simulations and can be robustly applied to reproduce and expand on the results presented in the TiDES cadence note[1].

References

[1] TiDES Cadence Note, Project Deliverable D3.3.2

[2] Richard Kessler et al. 2009, "SNANA: A Public Software Package for Supernova Analysis" DOI 10.1086/605984.

[3] Rahul Biswas et al. 2020,"Enabling Catalog Simulations of Transient and Variable Sources Based on LSST Cadence Strategies" DOI 10.3847/1538-4365/ab72f2.

[4] Richard Kessler et al. 2019, "Models and Simulations for the Photometric LSST Astronomical Time Series Classification Challenge (PLAsTiCC)" DOI 10.1088/1538-3873/ab26f1.

[5] Elmo Tempel et al. 2020, "An optimized tiling pattern for multiobject spectroscopic surveys: application to the 4MOST survey" DOI 10.1093/mnras/staa2285.

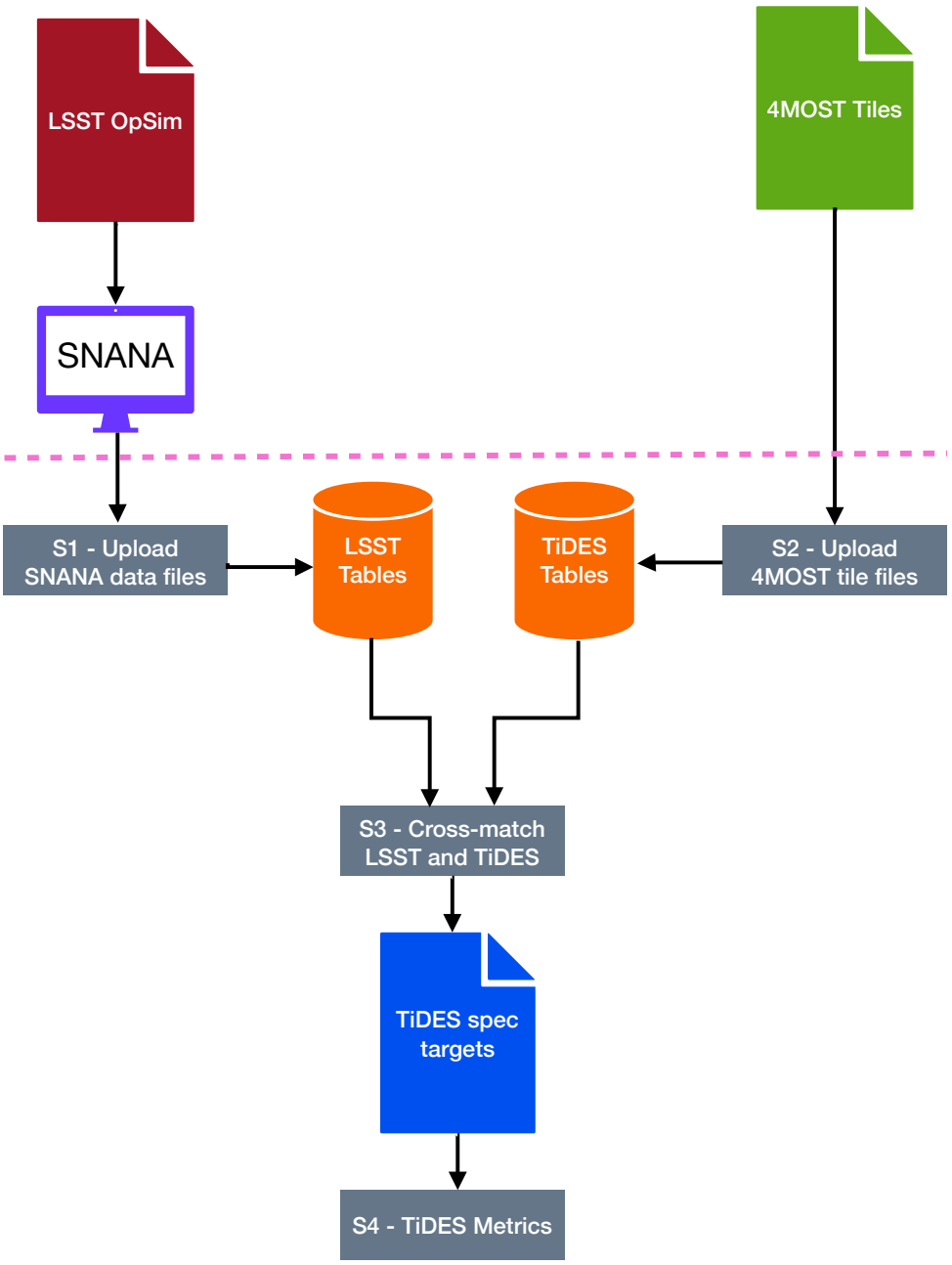

Figure 1: Flow chart of the data analysis pipeline for the TiDES Cadence Note. The inputs to the pipeline start with the LSST OpSim cadence files and the 4MOST Sky tiling packages. Files and software in the flowchart above the pink dashed line are not included in this deliverable as they are developed externally and independently maintained.

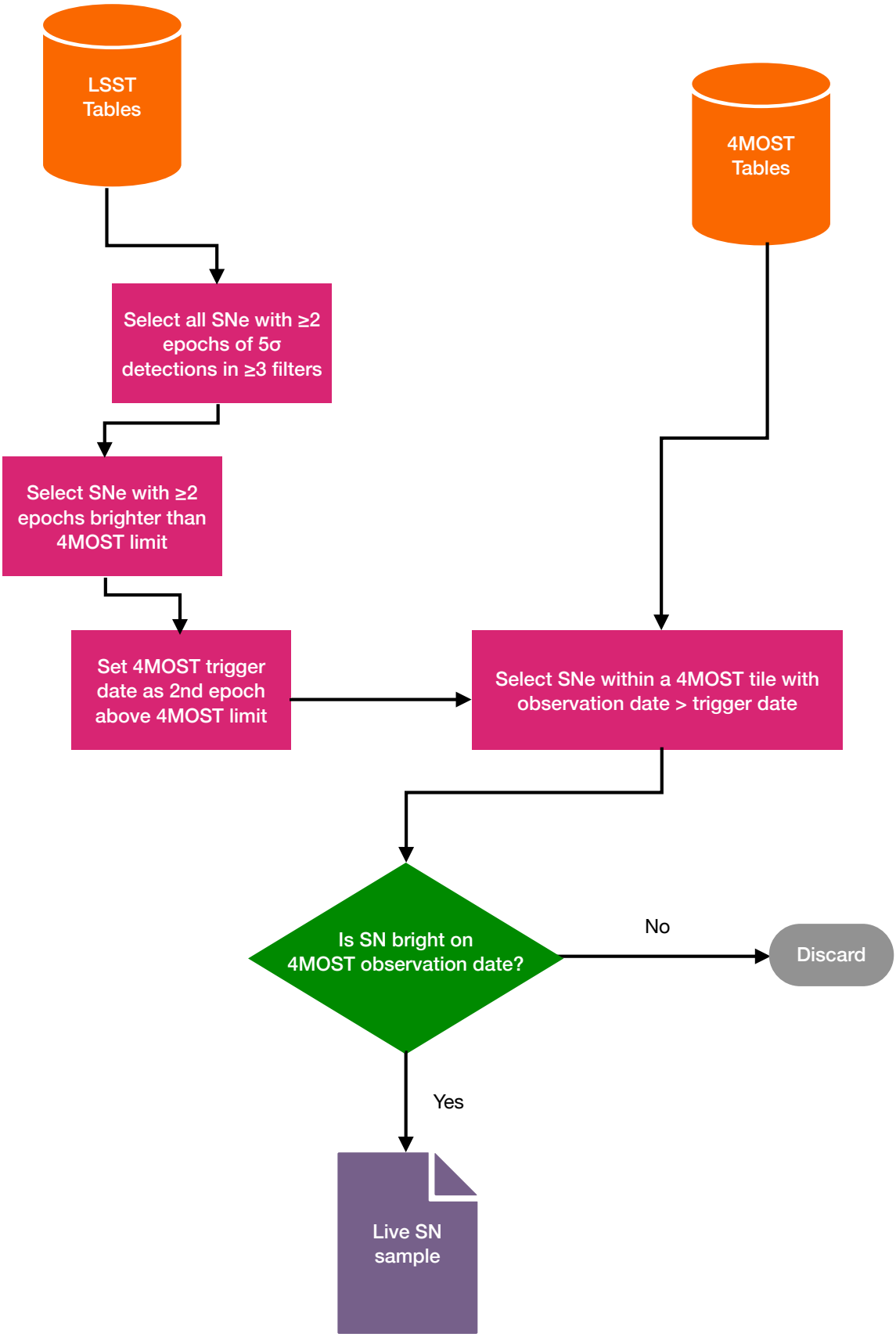

Figure 2: A flowchart of the live supernova cross-matching between the LSST simulations and the 4MOST tiling solution.