LAnDmARk - Lofar Automated Data Access & pRocessing

1. Introduction

1.1. Acknowledgements

This software was written by Janis Steinbergs, Kristaps Veitners, under the supervision of Marco Iacobelli and Vladislav Bezrukovs. If you make use of this software to get results that appear in a publication or presentation please include this acknowledgement: “We have made use of LAnDmARk, a tool developed by Janis Steinbergs, Kristaps Veitners.”

1.2. Capabilities of LAnDmARk

LAnDmARk has been thought to make LOFAR data access (and processing) more automated and user friendly. Currently it allows users to perform:

- Data selection from LOFAR Long Term Archive (LTA https://lta.lofar.eu).
- Data retrieval from LOFAR LTA
- Direction-independent processing of LOFAR LBA/HBA data via prefactor (v3) pipelines (https://github.com/lofar-astron/prefactor)
    - Setup working environment and run prefactor pipelines

2. Installation and usage

2.1. Installation

LAnDmARk can be downloaded at https://github.com/sklandrausis/LAnDmARk.git

2.2. Dependencies

- python 3
    - Coloredlogs 10.0
    - awlofar (necessary for staging and retrieval data) 2.7.1
    - matplotlib 3.0.3
    - Numpy 1.16.3
    - Seaborn 0.9.0
- Prefactor v3
- LOFAR software 3.2.0

2.3. Running LAnDmARk

To run LAnDmARk a configuration file is needed and a template version is provided (see `config.cfg`). The configuration file contains the following sections: *Data, Operation, Paths, Cluster*. The *Data* section collects needed info for data querying to LTA database and the *Operation* section specifies the action(s) required. Enabled operations are: **selection**, **stage**, **retrieve** and **process**. Note that in order to query, to stage and to retrieve data, users must have registered to LOFAR LTA (https://lta.lofar.eu). The *Paths* section contains pointers to the needed software (i.e. where the software is installed) as well as the working directory (i.e. where archived data will be collected and processed). The *Cluster* section allows users to set computing resources needed for data processing (maximum processes per single computing node, process data on a single computing node or on multiple nodes). In the *Data* section *ProductType* parameter specifies the if raw data must be downloaded or pipeline data products. In the *Operations* section the optional parameter *which_obj* specifies which type of products - *calibrators*, *targets* or *all* - should be processed.

When the configuration file is ready users can execute the selected operations with the command:

```
python3 main.py
```

The `main.py` script can be run with additional options:
`-v` or `--version` to display current version
`-h` or `--help` for help
`-c` or `--config` to point to configuration file. Default path is: `LAnDmARk/config.cfg`
`-d` or `--printlogs` to display additional information

Currently LAnDmARk supports only simple queries. First it checks if the user is part of the selected project (or if data under the selected project are public). Next LAnDmARk will use the specified calibrator ids, target ids and target name to find data products. Note however that if the selected project is "MSSS_HBA_2013", the user does not need to specify SAS id for the calibrator source. In this case the SAS ID of calibrator source(s) are obtained via the following logic:
Calirator <sas id> = target source SASid-1.

Note that staging requests cannot exceed 5 TB in volume and 5000 files at any point of time (see https://www.astron.nl/lofarwiki/doku.php?id=public:lta_howto). LAnDmARk will check if such constraints are satisfied before any staging of data products and if necessary abort the run and report to the user.

Finally, the **process** option currently only supports prefactor (v3) pipelines (https://github.com/lofar-astron/prefactor). In case only the target products are processed, the user should add the proper path … the calibrator solution h5parm file.

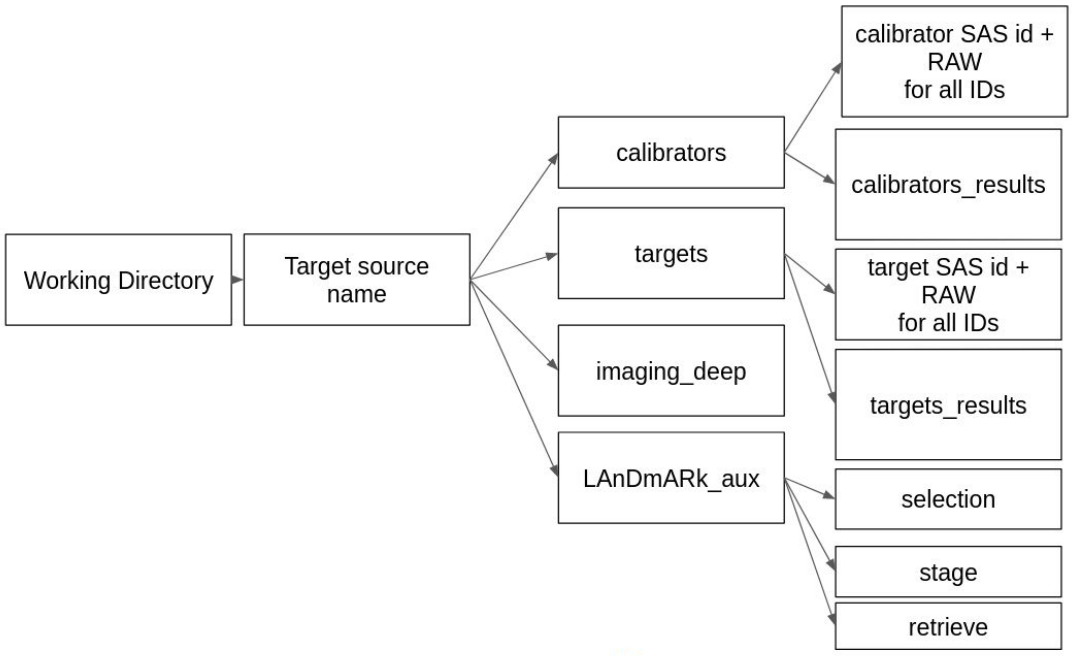
Scripts	**Description**
main.py	Run all scripts based on the config file
setup.py	Create working environment

selectionStaging.py	Perform **Selection** and **Stage** operations
retrieveDataproducts.py	Retrieve data products (**Retrieve** operation), display staging progress.
runPipelines.py	Process the data (**Process** operation)

3. LAnDmARk Output products

In the working directory specified via the configuration file LAnDmARk creates an output directory named as specified in the *TargetName* field of the *Data* section. Such a directory contains several subdirectories (as shown in the figure below):

- the `LAnDmARk_aux` directory, which stores auxiliary files. Also, diagnostic plots for each selected operation in are collected in dedicated subfolders.
- the `calibrators` directory, where calibrator data products are stored and inspection plots for prefactor will be placed.
- the `target` directory, where target data products are stored and inspection plots for prefactor will be placed.
- the `Image_deep` directory, where imaging pipeline inspection plots and output products will be placed.



If **selection** operation is set both a *calibrator_<SAS ID>_SURIs.txt.* and a *target_<SAS ID>_SURIs.txt* files will be created, which will contain the selected URIs. If in addition data products are staged, they can be retrieved using command *wget -i calibrator_<SAS ID>_SURIs.txt* and *wget -i target_<SAS ID>_SURIs.txt*

3.1. Inspection plots

For each enabled operation LAnDmARk will create inspection plots. For each selected SAS ID static plots will report on the number of valid dataproducts, and the count of array stations. Also a dynamic summary plot tracking the number and percentage of staged files.

Plot name	**explanation**
valid_data_per_sas_id.png	Valid data per SAS ID
station_count_per_sas_id.png	Station count per SAS ID
staging_progress.png	Percentage and number of files staged as a function of time

4. Changelog

Version 1.0

First release. Changes since version pre1.0 are:

- Enabled data selection from the LOFAR LTA
- Enabled data retrieval from the LOFAR LTA; added verification of size and number of data products at staging step
- Enabled direction-independent processing of LOFAR LBA/HBA data via prefactor (v3) pipelines (https://github.com/lofar-astron/prefactor)
- Enabled to specify the output for data selection

5. Getting Help

Bug reports, feature requests and make contributions (e.g. code patches) can be reported by opening a “new issue” ticket on GitHub. Please give as much information (e.g. the software component, version) as you can in the ticket. For bugs, it is extremely useful if a small self-contained code snippet that reproduces the problem is provided.