The ALMA Data Mining Toolkit I: Archive Setup and Usage

Peter Teuben[1], Marc Pound[1], Lee Mundy[1], Leslie Looney[2], Douglas Friedel[2]

[1]University of Maryland Illinois [2]University of Illinois

Abstract

We report on an ALMA development study where we employ a novel approach to add data and data descriptors to ALMA archive data and allowing further flexible data mining on retrieved data. We call our toolkit **ADMIT** (the **A**LMA **D**ata **Mi**ning **T**oolkit) that works within the Python based CASA environment. What is described here is a design study, with some exiting toy code to prove the concept.

After ingestion of science ready data cubes, **ADMIT** will compute a number of basic and advanced data products, and their descriptors. Example of such data products are cube statistics, line identification tables, line cubes, moment maps, an integrated spectrum, overlap integrals and feature extraction tables. Together with a descriptive XML file, a small number of visual aids are added to a ZIP file that is deposited into the archive. Large datasets (such as line cubes) will have to be rederived by the user once they have also downloaded the actual ALMA Data Products, or via VO services if available. ADMIT enables the user to rederive all its products with different methods and parameters, and compare archive product with their own.

See the accompanying poster **P72** by *Friedel et al.* about data mining large datasets of ALMA data crossing over projects and sources using **ADMIT**.

Procedure:

The ALMA science data archive (organized in a **Project** / **Source** / **Band** hierarchy) returns science quality data cubes as a result of the ALMA pipeline. ADMIT then harvests meta-data from these cubes and computes a suite of Basic Data Products (BDP's). The smaller of these BDP's are made available in an **admit.zip** file, which also contains a description of the whole project in **admit.xml**. ADMIT is then added to the archive for later user access (the brown box in Figure 1).

A user who then downloads **admit.zip** is able to quickly view most BDP's (see Figure 2, the brown box), though for any detailed analysis ADMIT must be used to recompute or even tune the BDP's. For example, Moment-0 maps may be available as JPG or FITS, but to re-compute them with a different algorithm, the appropriate Line Cube must be first re-created from the Band Cube, and then a Moment map can be re-created.

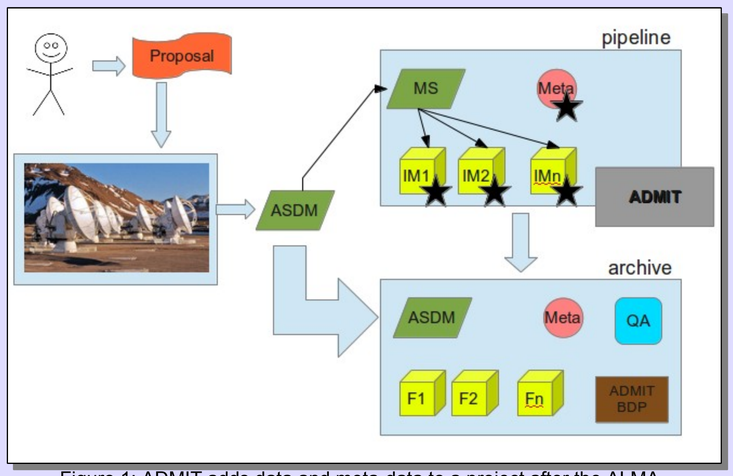

Figure 1: ADMIT adds data and meta-data to a project after the ALMA pipeline has produced the science cubes and makes them available in the data archives for later query and download.

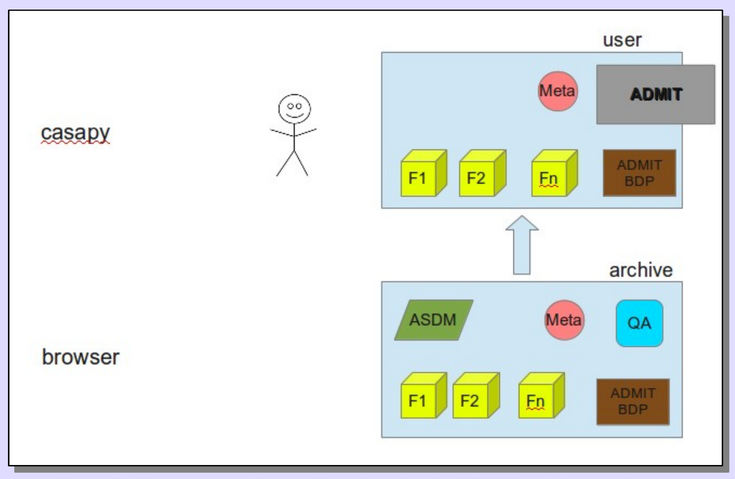

Figure 2: User queries the archive, downloads ADMIT data to preview product and optionally recomputes or adds more components to the data. Note the ADMIT software

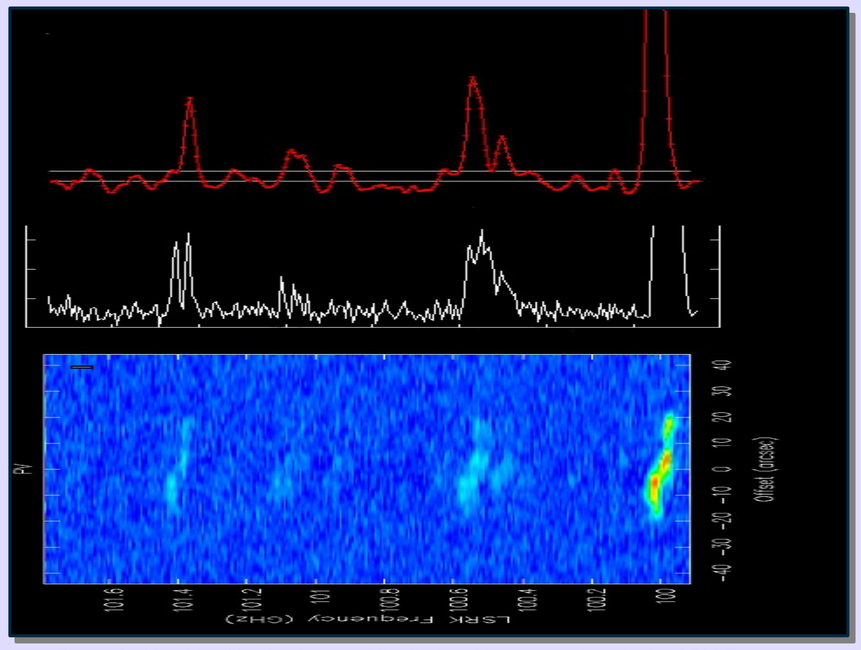

Figure 3: Line identification is one of the first and important steps in the ADMIT procedures where large frequency based Band Cubes are split up in doppler velocity based Line Cubes. This shows a pure robust median based statistics (white) vs. a cross correlation (red) technique.

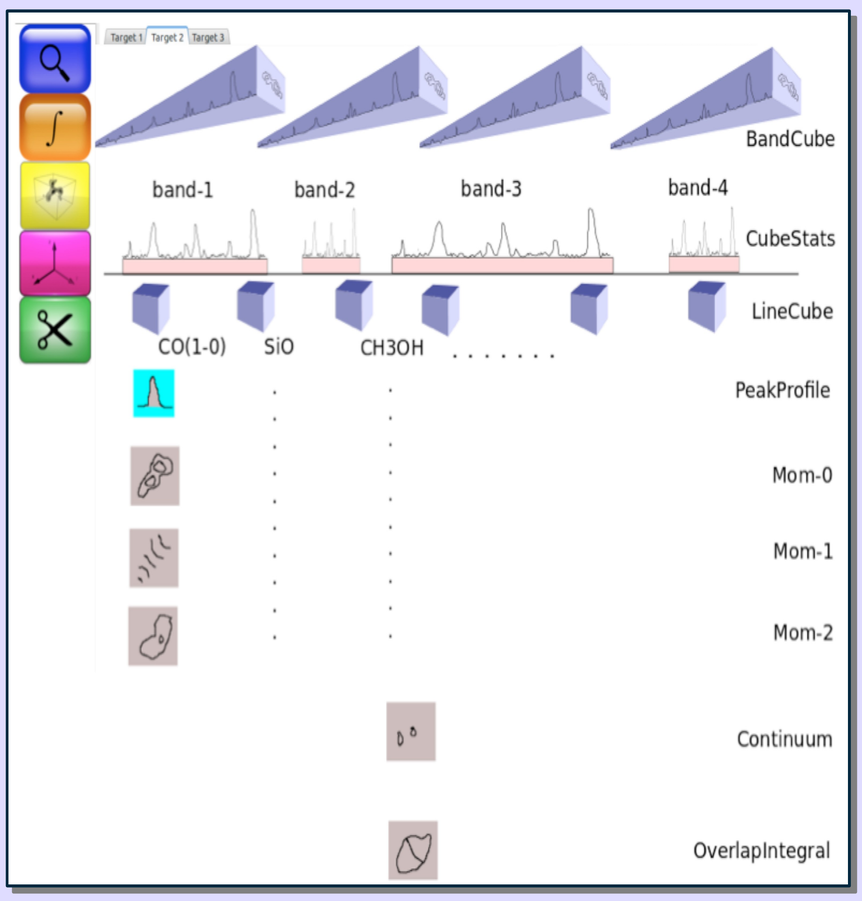

Figure 4: Overview of some of the ADMIT data products: starting with Band Cubes and line identification, Line Cubes can be cut, with derived products such as Peak Profile, Moment Maps, Feature Extraction, Overlap Integrals, and a suite of Description Vectors.

admit.xml

Central to the ADMIT procedures is an **admit.xml (**sketched to the left) description of the data and meta-data that ADMIT had harvested from the Band Cubes. It is also this database from which any further data mining will take place.
Note the **Project** – **Source** – **Band/Line** hierarchy.

For a more proper XML view, see **P72.**

```
Project(name)[NP]
  Summary
    <atask name=at_summary>
  Source(name)[NS]
    Summary
      ra,dec,vlsr,...
      <atask name=at_summary>
    Band(number)[NB]
      URI:im
      Summary
        FreqMin,FreqMax,FreqStep
      CubeStats
        VoTable:tab
        <atask name=at_cubestats>
        <dep>
          URI:im
      PosVelSlice
        URI:im
        file:jpg
        <atask name=at_pv>
        <dep>
          URI:im
      LineList
        VoTable:tab
        <atask name=at_band2line>
        <dep>
          CubeStats
    LineList
      votable:tab
      <atask name=at_linemerge>
      <dep>
        band[NB].LineList
    Continuum(name)
      URI:im
      file.jpg
      <atask name=at_continuum>
      <dep>
        Band(this)
    Line(name)[NL]
      LineCube
        URI:im
        <atask name=at_reframe>
        <dep>
          LineList
      RMS (since cubestats can differ per channel)
      Mom0
        URI:im
        file:jpg
        <atask name=at_moment>
        <dep>
          LineCube
      Mom1
      Mom2
      PeakSpectrum
        VoTable:tab
        <summary>
          Peak, RMS, V0, FWHM, SdV
        <atask name=at_spectrum>
        <dep>
          LineCube
      IntegratedSpectrum
        VoTable:tab
        <summary>
          Peak, RMS, V0, FWHM, SdV
        <atask name=at_spectrum>
        <dep>
          LineCube
      FeatureList
        VoTable:tab
        <atask name=at_feature>
        <dep>
          LineCube
      DescriptionVector
        VoTable:tab
  DescriptionVector
    VoTable:tab
DescriptionVector
  VoTable:tab
```

BDP (Basic Data Product)

Name
Data Element 1
Data Element 2
…
Data Element N
Task
Dependencies

ADMIT procedures

Any ADMIT procedure results in a BDP, but as ADMIT re-computes a BDP, its dependencies need to be tracked down and this may require re-computing its ancestors as well. Figure 4 shows some of the obvious dependencies in the BDP's.

The initial stages of harvesting data is very traditional: Line Cubes and Moment Maps derived from the Line Identification process. Features can be extracted from Line Cubes.

One of the more advanced BDP's we propose are based on **Overlap Integrals** and **Descriptor Vectors** (DV). They show in either a Data Cube or Image Map where and which detected lines are present, and once correlated across many lines and objects, give insight into the distribution and formation of molecules. A DV is constructed from a small number of parameters of a given object. For example, from a feature extraction program, these could be the moments of inertia, or across lines these could be the different intensity levels in addition to the moments of inertia. As in a Principle Component Analysis, this allows one in some defined parameter space to find correlations and clumps, define distances between “objects”, etc.

Take Home Points

- This is a design study, with a toy implementation (code available via CVS upon request as **ASTUTE**)
- ADMIT can also work offline on “any” science data cubes (ATCA, CARMA, SMA, VLA, WSRT etc.)
- ADMIT executes basic CASA tasks, but can also execute foreign packages, such as MIRIAD.
- ADMIT is a extensible toolkit, and works in Python as well as can be directed from a GUI (see also **P72)**
- ADMIT can do effective data mining (see also **P72**) via its Python interfaces, and integrates well with community software.