ALminer

Release 0.1.3

Aida Ahmadi

Feb 07, 2023

TUTORIALS

`alminer` is a Python-based code to effectively query, analyse, and visualize the ALMA science archive. It also allows users to directly download ALMA data products and/or raw data for further image processing.

CHAPTER ONE

INSTALLATION

The easiest way to install `alminer` is with `pip`:

```
pip install alminer
```

To obtain the most recent version of the code from GitHub:

```
pip install https://github.com/emerge-erc/ALminer/archive/refs/heads/main.zip
```

Or clone and install from source:

```
# If you have a Github account:
git clone git@github.com:emerge-erc/ALminer.git
# If you do not:
git clone https://github.com/emerge-erc/ALminer.git

# After cloning:
cd ALminer
pip install .
```

Note that depending on your setup, you may need to use pip3.

1.1 Dependencies

The dependencies are `numpy`, `matplotlib`, pandas, pyvo, astropy version 3.1.2 or higher, and astroquery version 0.4.2.dev6649 or higher. We only use the `astroquery` package for downloading data from the ALMA archive. The strict requirement to have its newest version is due to recent changes made to the ALMA archive. `alminer` works in Python 3.

CHAPTER

TWO

GETTING STARTED

We have created an extensive tutorial Jupyter Notebook where all `alminer` features have been highlighted. This is an excellent starting point to get familiar with all the possibilities; a glossary of all functions is provided at the bottom of this notebook. We highly recommend working in a Jupyter notebook environment in order to make use of `alminer`'s visualization tools. We aim to keep adding new notebooks relevant for various sub-fields in the future.

Note that the Jupyter notebooks may be outdated. The most up-to-date information can be found on this documentation page.

Note: To work with the tutorial notebook interactively launch Jupyter Notebook

2.1 1. Query tools

This Section introduces three methods to query the ALMA archive:

- *1.1 - Query by target name (alminer.target)*
- *1.2 - Query a catalog (alminer.catalog)*
- *1.3 - Query by ALMA keywords (alminer.keysearch)*

General notes about the querying functions:

- All querying functions search the ALMA archive for public data by default. To include both public and proprietary data in the search, set *public=None*. Similarly, to search for only propietary data, set *public=False*.
- All querying functions search the ALMA archive for both published and unpublished data. To include only unpublished data, set *published=False*.
- The querying functions will by default print a summary of the observations, including a list of target names. For large queries, it is useful to turn this feature off in order to not have a long list of targets printed to screen. To turn off this feature, simply set *print_targets=False*.
- The queried archive service is by default the Europeran one (ESO). Other services can be specified through the *tap_service* argument. Options are: ESO, NRAO, or NAOJ.
- The queries return all possible observations in PANDAS DataFrame format that can be used to further narrow down your search as demonstrated in *Section 2*.

Load libraries

```
[1]: import alminer
     import pandas
     from astropy.io import ascii
```

2.1.1 1.1 Query by target name

The *alminer.target* function allows one to query objects by name. This function uses the Astropy SESAME resolver which searches multiple databases (Simbad, NED, VizieR) to obtain the coordinates of the object of interest, and then queries the ALMA archive for all observations that contain those coordinates (corresponding to the case of *point=True* which is the default). When *point=False*, the function will return all observations that overlap with a cone extending the position of interest and a search radius around it. The search radius is by default 1.0 arcminute, but can be modified using the *search_radius* keyword (in arcmin units).

Example 1.1.1: query two sources by name

```
[2]: myquery = alminer.target(['Orion KL', "AB Aur"])
```

```
================================
alminer.target results
================================
Target = Orion KL
--------------------------------
Number of projects = 23
Number of observations = 38
Number of unique subbands = 129
Total number of subbands = 160
18 target(s) with ALMA data = ['OMC1_NW', 'Orion-KL', 'orion_kl', 'BN-KL', 'OMC1_SE',
↪'Orion_KL_Field_3_North-west_Clump', 'BN', 'OrionKL', 'Orion', 'Orion_KL', 'orion-
↪IRc2', 'OMC-1', 'ONC', 'Orion1', 'OMC-1_Region2', 'ONC_Mosaic', 'Orion_Source_I',
↪'Orion_BNKL_source_I']
--------------------------------
Target = AB Aur
--------------------------------
Number of projects = 3
Number of observations = 3
Number of unique subbands = 17
Total number of subbands = 17
3 target(s) with ALMA data = ['AB_Auriga', 'AB_Aur', 'ab_aurigae']
--------------------------------
```

Example 1.1.2: query a list of objects by name

First create a catalog or a list of object names. In this example, the catalog `Sample_cat.dat` has the following content:

```
 Name      RA       DEC
------ --------- --------
AB_Aur  73.9412  30.5511
AK_Sco 253.6867 -36.8886
AS_310 278.3383  -4.9683
AS_470 324.0592  57.3586
AS_477 328.1421  47.2289
```

Note that the column that is used is the *Name* column and the coordinates are ignored in this example.

```
[3]: mylist = ascii.read("Sample_cat.dat", header_start=0, data_start=1)
myquery = alminer.target(mylist['Name'])
```

```
================================
alminer.target results
```

(continues on next page)

(continued from previous page)

```
================================
Target = AB_Aur
--------------------------------
Number of projects = 3
Number of observations = 3
Number of unique subbands = 17
Total number of subbands = 17
3 target(s) with ALMA data = ['AB_Auriga', 'AB_Aur', 'ab_aurigae']
--------------------------------
Target = AK_Sco
--------------------------------
Number of projects = 3
Number of observations = 3
Number of unique subbands = 12
Total number of subbands = 12
2 target(s) with ALMA data = ['AK_Sco', 'HIP_82747']
--------------------------------
Target = AS_310
--------------------------------
No observations found.
--------------------------------
Target = AS_470
--------------------------------
No observations found.
--------------------------------
Target = AS_477
--------------------------------
No observations found.
--------------------------------
```

Example 1.1.3: include proprietary data

```
[4]: myquery = alminer.target(mylist['Name'], public=None)
```

```
================================
alminer.target results
================================
Target = AB_Aur
--------------------------------
Number of projects = 6
Number of observations = 10
Number of unique subbands = 56
Total number of subbands = 62
3 target(s) with ALMA data = ['AB_Aur', 'AB_Auriga', 'ab_aurigae']
--------------------------------
Target = AK_Sco
--------------------------------
Number of projects = 4
Number of observations = 4
Number of unique subbands = 16
Total number of subbands = 16
2 target(s) with ALMA data = ['AK_Sco', 'HIP_82747']
--------------------------------
Target = AS_310
--------------------------------
```

(continues on next page)

(continued from previous page)

```
No observations found.
--------------------------------
Target = AS_470
--------------------------------
No observations found.
--------------------------------
Target = AS_477
--------------------------------
No observations found.
--------------------------------
```

Example 1.1.4: account for mosaics

The *alminer.target* function will by default search whether any ALMA observations contain the target of interest's position. To search whether any ALMA observations overlap with a larger region of interest, one can set the argument *point=False* and provide a search radius in arcminutes using the *search_radius* argument. The search radius is by default 1.0 arcminute.

```
[5]: myquery = alminer.target(['Orion KL', "AB Aur"], point=False, search_radius=2.0)
```

```
================================
alminer.target results
================================
Target = Orion KL
--------------------------------
Number of projects = 37
Number of observations = 125
Number of unique subbands = 312
Total number of subbands = 576
57 target(s) with ALMA data = ['OrionKL', 'Orion H2O maser outburst', 'OrionField1-1',
↪ 'OrionField2', 'OrionField1-2', 'f1', 'f7', 'f5', 'f8', 'f4', 'f3', 'orion_kl',
↪'Orion_Source_I', 'BN', 'f13', 'f11', 'f12', 'f10', 'f9', 'f15', 'f14', 'orion-IRc2
↪', 'Orion_KL', 'OMC1_SE', 'BN-KL', 'OMC1_NW', 'f23', 'f16', 'OMC-1S', 'OrionKL-SV',
↪'OMC-1', 'OrionBullets', 'ONC', 'Orion_BNKL_source_I', 'OMC-1_Region5', 'OMC-1_
↪Region1', 'Orion', 'OMC-1_Region2', 'HC602_HC606_HC608', 'GEMS28', 'HC672', 'ONC_
↪Mosaic', 'OMC-1_Region3', 'OMC-1_Region4', 'Orion_KL_Field_1_Orion_Hot_Core',
↪'Orion_KL_Field_2_SMA1', 'Orion_KL_Field_3_North-west_Clump', 'Orion KL', 'Orion1',
↪'101', '32', '104', '107', '71', 'Orion-KL', 'ORS-8', 'ORS-4']
--------------------------------
Target = AB Aur
--------------------------------
Number of projects = 3
Number of observations = 3
Number of unique subbands = 17
Total number of subbands = 17
3 target(s) with ALMA data = ['ab_aurigae', 'AB_Aur', 'AB_Auriga']
--------------------------------
```

2.1.2 1.2 Query by position

The *alminer.conesearch* and *alminer.catalog* functions can be used to directly query the ALMA archive by positions in the sky and a search radius around them. The right ascension and declinations must be given in units of degrees (ICRS). You can use the Astropy coordinates package to convert your desired coordinates to degrees.

Example 1.2.1: query an object by its coordinates (RA, Dec)

```
[6]: myquery = alminer.conesearch(ra=201.365063, dec=-43.019112, point=False, search_
↪radius=10.0)
```

```
--------------------------------
Number of projects = 24
Number of observations = 77
Number of unique subbands = 192
Total number of subbands = 312
9 target(s) with ALMA data = ['J1325-430', 'Centaurus_A', 'J1325-4301', 'CenA',
↪'Centaurus_a', '3FGL_J1325.4-4301', 'Centaurus A', 'NGC_5128', 'Cen_A']
--------------------------------
```

Example 1.2.2: query a catalog of objects by their coordinates (RA, Dec)

Let's first import a catalog, for example the catalog of Spitzer YSOs in Orion from Megeath et al. (2009), and create a PANDAS DataFrame using rows 866 to 869 of this catalog. Then use the *alminer.catalog* function to query the ALMA archive for each target in the DataFrame.

```
[7]: Spitzer = ascii.read("Spitzer_sample.dat", header_start=0, data_start=866, data_
↪end=869)

mycat =  {"Name": Spitzer["Seq"],
          "RAJ2000" : Spitzer["RA2000"],
          "DEJ2000" : Spitzer["DEC2000"]}

mycat = pandas.DataFrame(mycat)

myquery = alminer.catalog(mycat)
```

```
================================
alminer.catalog results
================================
Target = 866
--------------------------------
Number of projects = 1
Number of observations = 1
Number of unique subbands = 4
Total number of subbands = 4
1 target(s) with ALMA data = ['M12_866']
--------------------------------
Target = 867
--------------------------------
Number of projects = 1
Number of observations = 1
Number of unique subbands = 4
Total number of subbands = 4
1 target(s) with ALMA data = ['M12_867']
```

(continues on next page)

(continued from previous page)

```
--------------------------------
Target = 868
--------------------------------
Number of projects = 2
Number of observations = 2
Number of unique subbands = 8
Total number of subbands = 8
1 target(s) with ALMA data = ['HOPS-172']
--------------------------------
```

2.1.3 1.3 Query by ALMA keywords

Query the ALMA archive for any *(string-type) keywords defined in ALMA TAP system* using the *alminer.keysearch* function.

The power of this function is in combining keywords. When multiple keywords are provided, they are queried using 'AND' logic, but when multiple values are provided for a given keyword, they are queried using 'OR' logic. If a given value contains spaces, its constituents are queried using 'AND' logic. Words encapsulated in quotation marks (either ' or ") are queried as phrases. For example,

- `alminer.keysearch({"proposal_abstract": ["high-mass star formation outflow disk"]})` will query the archive for projects with the words "high-mass" AND "star" AND "formation" AND "outflow" AND "disk" in their proposal abstracts.
- `alminer.keysearch({"proposal_abstract": ["high-mass", "star", "formation", "outflow", "disk"]})` will query the archive for projects with the words "high-mass" OR "star" OR "formation" OR "outflow" OR "disk" in their proposal abstracts.
- `alminer.keysearch({"proposal_abstract": ["'high-mass star formation' outflow disk"]})` will query the archive for projects with the phrase "high-mass star formation" AND the words "outflow" AND "disk" in their proposal abstracts.
- `alminer.keysearch({"proposal_abstract": ["star formation"], "scientific_category":['Galaxy evolution']})` will query the archive for projects with the words "star" AND "formation" in their proposal abstracts AND projects that are within the scientific_category of 'Galaxy evolution'.

Note

- Tables of ALMA *scientific categories* and *science keywords* are provided on the sidebar.
- For an overview, see Appendix D of the ALMA Proposer's Guide.

Example 1.3.1: query a list of ALMA target names that may not be in SIMBAD/NED/VizieR

```
[8]: myquery = alminer.keysearch({'target_name': ['GRB021004','SPT0319-47', 'G345']})
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 17
Number of observations = 31
```

(continues on next page)

(continued from previous page)

```
Number of unique subbands = 121
Total number of subbands = 191
12 target(s) with ALMA data = ['GRB021004', 'G345.5', 'SPT0319-47', 'G345.5043+00.3480
↪', 'G345.49+1.47', 'G345.50+0.35', 'G345.6487+0.0089', 'G345.01', 'G345.11', 'G345.
↪0029-0.2241', 'G345.5+1.5', 'G345.144-00.216']
--------------------------------
```

Example 1.3.2: query a list of ALMA projects by their proposal IDs

```
[9]: myquery = alminer.keysearch({'proposal_id': ['2015.1.00664.S', '2016.1.00204.S']})
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 2
Number of observations = 16
Number of unique subbands = 16
Total number of subbands = 64
16 target(s) with ALMA data = ['KMOS3DCOS4-24763', 'KMOS3DGS4-25151', 'KMOS3DCOS4-
↪13701', 'KMOS3DCOS4-10347', 'KMOS3DCOS4-13174', 'KMOS3DCOS4-19680', 'KMOS3DCOS4-
↪15813', 'KMOS3DCOS4-15820', 'KMOS3DU4-34138', 'KMOS3DU4-22227', 'KMOS3DU4-32147',
↪'KMOS3DU4-20547', 'KMOS3DGS4-11016', 'KMOS3DGS4-24110', 'KMOS3DGS4-27882', 'AK_Sco']
--------------------------------
```

Example 1.3.3: query by words in the proposal abstract

Query the ALMA archive for proposals that have the phrase 'high-mass star formation' AND the words 'outflow' AND 'disk', OR the phrase 'massive star formation' AND the words 'outflow' AND 'disk' - and do not print the long list of target names in the summary.

```
[10]: myquery = alminer.keysearch({'proposal_abstract': ['"high-mass star formation"␣
↪outflow disk',
                                                     '"massive star formation" outflow␣
↪disk']},
                            print_targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 14
Number of observations = 59
Number of unique subbands = 206
Total number of subbands = 423
Total number of targets with ALMA data = 29
--------------------------------
```

Example 1.3.4: query by combination of keywords

Query the ALMA archive for proposals that have the phrase ‘star formation’ in their abstracts and correspond to the scientific category of ‘Galaxy evolution’.

```
[11]: myquery = alminer.keysearch({'proposal_abstract': ['"star formation"'],
                                   'scientific_category':['Galaxy evolution']}, print_
      ↪targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 245
Number of observations = 2839
Number of unique subbands = 3908
Total number of subbands = 11687
Total number of targets with ALMA data = 1763
--------------------------------
```

Example 1.3.5: query for full polarization data

```
[ ]: myquery = alminer.keysearch({'science_keyword':['"disks around low-mass stars"'],
                                  'pol_states':['XY', 'YX']}, print_targets=False)
```

```
================================
alminer.keysearch results
================================
```

2.2 2. Filter & explore results

The querying functions presented in the previous section return a PANDAS DataFrame that can be used to further narrow down your search. This section presents some examples of how you can further filter and explore the results of your queries:

- *2.1 - Explore results (alminer.explore)*
- *2.2 - Summarize results (alminer.summary)*
- *2.3 - Filter results (alminer.get_info)*
- *2.4 - Line coverage (alminer.line_coverage)*
- *2.5 - CO, 13CO, C18O lines (alminer.CO_lines)*

Load libraries & create a query

To explore these options, we will first query the archive using one of the methods presented in the previous section and use the results in the remainder of this tutorial.

```
[1]: import alminer

     observations = alminer.keysearch({'science_keyword':['Galaxy chemistry']},
                                      print_targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 48
Number of observations = 341
Number of unique subbands = 1166
Total number of subbands = 1368
Total number of targets with ALMA data = 64
--------------------------------
```

2.2.1 2.1 Explore results

You can simply display the DataFrame table returned by the query functions using the name you gave it (in this case *observations*), but often there are limits to how many rows and columns are presented. With the *alminer.explore* function, you can control whether or not you want to display all rows (*allrows=True/False*) and/or all columns (*allcols=True/False*). By default, only the 18 most useful columns are shown and the number of rows is truncated.

Example 2.1.1: View the queried observations as a table (shortened)

```
[2]: alminer.explore(observations)
```

```
[2]:        Obs    project_code  … line_sens_native                 MOUS_id
0         1   2011.0.00268.S  …           112.74  uid://A002/X303d22/X7b
1         2   2011.0.00268.S  …           111.94  uid://A002/X303d22/X7b
2         3   2011.0.00405.S  …           119.14  uid://A002/X36d874/X7a
3         4   2011.0.00405.S  …           120.04  uid://A002/X36d874/X7a
4         5   2011.0.00405.S  …           119.28  uid://A002/X36d874/X7a
5         6   2011.0.00405.S  …            59.69  uid://A002/X36d874/X80
6         7   2011.0.00405.S  …            59.16  uid://A002/X36d874/X80
7         8   2011.0.00405.S  …            61.02  uid://A002/X36d874/X80
8         9   2011.0.00405.S  …            59.71  uid://A002/X36d874/X80
…       …              …  …              …                      …
1359  1360   2019.1.00130.S  …            19.97  uid://A001/X1465/X3890
1360  1361   2019.1.00130.S  …            20.59  uid://A001/X1465/X389c
1361  1362   2019.1.00130.S  …            20.60  uid://A001/X1465/X389c
1362  1363   2019.1.00130.S  …            21.39  uid://A001/X1465/X389c
1363  1364   2019.1.00130.S  …            21.40  uid://A001/X1465/X389c
1364  1365   2019.1.00130.S  …            18.56  uid://A001/X1465/X3898
1365  1366   2019.1.00130.S  …            18.72  uid://A001/X1465/X3898
1366  1367   2019.1.00130.S  …            18.82  uid://A001/X1465/X3898
1367  1368   2019.1.00130.S  …            18.71  uid://A001/X1465/X3898

[1368 rows x 18 columns]
```

Example 2.1.2: View the queried observations as a table and show all columns

```
[3]: alminer.explore(observations, allcols=True, allrows=False)
     # you can also set allrows=True to see all rows in the table
     # but this may be slow for very large queries
```

```
[3]:       Obs    project_code  … scientific_category            lastModified
     0        1  2011.0.00268.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     1        2  2011.0.00268.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     2        3  2011.0.00405.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     3        4  2011.0.00405.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     4        5  2011.0.00405.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     5        6  2011.0.00405.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     6        7  2011.0.00405.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     7        8  2011.0.00405.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     8        9  2011.0.00405.S  …    Galaxy evolution  2022-03-16T15:26:21.801
     …     …               …  …                   …                      …
     1359  1360  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1360  1361  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1361  1362  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1362  1363  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1363  1364  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1364  1365  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1365  1366  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1366  1367  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801
     1367  1368  2019.1.00130.S  …     Active galaxies  2022-03-16T15:26:21.801

     [1368 rows x 81 columns]
```

2.2.2 2.2 Summarize results

The *alminer.summary* function will print a summary of the observations. This is done by default when the query is run, but it's a useful function if the results are filtered further, as shown in the next section.

Example 2.2.1: print the summary of a given query result, including a list of unique ALMA target names

```
[4]: alminer.summary(observations)
```

```
--------------------------------
Number of projects = 48
Number of observations = 341
Number of unique subbands = 1166
Total number of subbands = 1368
64 target(s) with ALMA data = ['LESS J0332-2756', 'PKS1830-211', 'ngc4418', 'NGC7469',
↪ 'NGC_1097', 'Arp220', 'NGC_5253', 'M83', 'NGC253', 'NGC1266', 'vv114', 'Mystery_
↪Object', 'Sgr_A_star', 'SDSS_J080430.99+360718.1', 'NGC_3256', 'ngc_3256', 'NGC_55',
↪ 'ngc3256', 'NGC4418', 'NGC_3627_BE', 'circinus', 'IRAS_F16399-0937', 'ngc6240',
↪'ngc613', 'n613', 'ngc253', 'N159-W_south', 'N159-E', 'N159-W', 'm83', 'IRAS_13120-
↪5453', 'ngc4945', 'HE0433-1028', 'HE0108-4743', 'HE1353-1917', 'HE1108-2813',
↪'HE1029-1831', 'Y050355.87-672045.1', 'Y045622.61-663656.9', 'Y054248.90-694446.3',
↪'Y054826.21-700850.2', 'Y050953.89-685336.7', 'Y052333.40-693712.1', 'Y052343.48-
↪680033.9', 'Y045406.43-664601.4', 'Y054629.32-693514.2', 'ST10', 'Y045358.57-691106.
↪7', 'Y051912.27-690907.2', 'ST6', 'Y044854.41-690948.3', 'Y052423.39-693904.7',
↪'Y052210.08-673459.6', 'Y053952.11-710930.7', 'Y051344.99-693510.6', 'Y051916.87-
```

(continues on next page)

(continued from previous page)

```
↪693757.5', 'Y045100.16-691934.4', 'ngc4526', 'Cloverleaf', 'ngc7465', 'NGC_253',
↪'IRAS_22491-1808', 'NGC4945', 'NGC1068']
--------------------------------
```

Example 2.2.2: print the summary of a given query result WITHOUT the list of target names

For big tables, it is useful to avoid printing the list of target names in the summary. This can be done by setting *print_targets=False*.

[5]:
```
alminer.summary(observations, print_targets=False)
```

```
--------------------------------
Number of projects = 48
Number of observations = 341
Number of unique subbands = 1166
Total number of subbands = 1368
Total number of targets with ALMA data = 64
--------------------------------
```

2.2.3 2.3 Filter results

The search results can be further narrowed down by using PANDAS DataFrame functions. See also this introduction to data structures. For example, you can use `your_query.columns` to get a list of all columns in the DataFrame.

To get the description and units of any column use the function *alminer.get_info('column_name')* where *column_name* is the name of the column.

Example 2.3.1: simple selection - observations with angular resolutions < 0.5"

Let's first check what the description and units of *ang_res_arcsec* column are:

[6]:
```
alminer.get_info('ang_res_arcsec')
```

```
--------------------------------
Column: ang_res_arcsec
--------------------------------
Description: typical spatial resolution
Units: arcsec
--------------------------------
```

Now we can do the some further filtering, say to only keep observations with angular resolutions < 0.5":

[7]:
```
selected = observations[observations['ang_res_arcsec'] < 0.5]
```

and print the summary:

[8]:
```
alminer.summary(selected, print_targets=False)
```

```
--------------------------------
Number of projects = 23
Number of observations = 107
Number of unique subbands = 313
Total number of subbands = 427
```

(continues on next page)

(continued from previous page)

```
Total number of targets with ALMA data = 41
--------------------------------
```

Example 2.3.2: multiple selections - observations with angular resolution < 0.5" & velocity resolution < 1 km/s

[9]:
```
selected = observations[(observations['ang_res_arcsec'] < 0.5) &
                        (observations['vel_res_kms'] < 1.0)]
alminer.summary(selected, print_targets=False)
```

```
--------------------------------
Number of projects = 10
Number of observations = 50
Number of unique subbands = 78
Total number of subbands = 120
Total number of targets with ALMA data = 27
--------------------------------
```

Example 2.3.3: observations containing a given frequency

[10]:
```
freq = 220.5
selected = observations[(observations["min_freq_GHz"] < freq) &
                        (observations["max_freq_GHz"] > freq)]
alminer.summary(selected, print_targets=False)
```

```
--------------------------------
Number of projects = 6
Number of observations = 7
Number of unique subbands = 7
Total number of subbands = 7
Total number of targets with ALMA data = 5
--------------------------------
```

2.2.4 2.4 Line coverage

An alternative to the last example in the previous section is the *alminer.line_coverage* function which determines how many targets were observed at a given frequency with the option to include a redshift for the line of interest to be taken into account. Some notes: * Line frequencies should be given in GHz * Redshift is by default assumed to be 0 * The *line_name* keyword is the user's defined name for the frquency provided

Example 2.4.1: search whether a given frequency is covered in the observations

[11]:
```
myline_obs = alminer.line_coverage(observations,
                                   line_freq=220.5,
                                   z=0,
                                   line_name="My favourite line",
                                   print_targets=True)
```

```
--------------------------------
Summary of 'My favourite line' observations at 220.5 GHz
--------------------------------
Number of projects = 6
Number of observations = 7
Number of unique subbands = 7
Total number of subbands = 7
5 target(s) with ALMA data = ['Arp220', 'ngc_3256', 'IRAS_13120-5453', 'ngc253', 'NGC_
↪253']
--------------------------------
```

Example 2.4.2: search whether a given frequency is observed for a target at a given redshift

[12]:
```
myline_obs = alminer.line_coverage(observations,
                                   line_freq=400.0,
                                   z=0.5,
                                   line_name="My favourite line",
                                   print_targets=True)
```

```
--------------------------------
Summary of 'My favourite line' observations at 400.0 GHz (266.667 GHz at z=0.5)
--------------------------------
Number of projects = 8
Number of observations = 10
Number of unique subbands = 11
Total number of subbands = 11
6 target(s) with ALMA data = ['ngc4418', 'Arp220', 'ngc_3256', 'circinus', 'ngc4945',
↪'ngc253']
--------------------------------
```

2.2.5 2.5 Coverage of CO, 13CO, and C18O lines

The *alminer.CO_lines* function determines how many CO, 13CO, and C18O lines were observed in the provided DataFrame and returns a DataFrame containing all observations of these transitions.

Example 2.5.1: search whether any CO, 13CO, and C18O lines were observed in the query results

[13]:
```
CO_obs = alminer.CO_lines(observations, print_targets=False)
```

```
--------------------------------
Summary of 'CO (1-0)' observations at 115.271 GHz
--------------------------------
Number of projects = 4
Number of observations = 5
Number of unique subbands = 4
Total number of subbands = 5
Total number of targets with ALMA data = 3
--------------------------------
--------------------------------
Summary of 'CO (2-1)' observations at 230.538 GHz
--------------------------------
Number of projects = 10
Number of observations = 13
```

(continues on next page)

(continued from previous page)

```
Number of unique subbands = 13
Total number of subbands = 13
Total number of targets with ALMA data = 8
--------------------------------
--------------------------------
Summary of 'CO (3-2)' observations at 345.796 GHz
--------------------------------
Number of projects = 3
Number of observations = 5
Number of unique subbands = 5
Total number of subbands = 5
Total number of targets with ALMA data = 4
--------------------------------
--------------------------------
Summary of 'CO (4-3)' observations at 461.041 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'CO (5-4)' observations at 576.268 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'CO (6-5)' observations at 691.473 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'CO (7-6)' observations at 806.652 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'CO (8-7)' observations at 921.8 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of '13CO (1-0)' observations at 110.201 GHz
--------------------------------
Number of projects = 18
Number of observations = 23
Number of unique subbands = 23
Total number of subbands = 25
Total number of targets with ALMA data = 17
--------------------------------
--------------------------------
Summary of '13CO (2-1)' observations at 220.399 GHz
--------------------------------
Number of projects = 6
Number of observations = 7
Number of unique subbands = 7
Total number of subbands = 7
Total number of targets with ALMA data = 5
--------------------------------
--------------------------------
```

(continues on next page)

(continued from previous page)

```
Summary of '13CO (3-2)' observations at 330.588 GHz
--------------------------------
Number of projects = 2
Number of observations = 5
Number of unique subbands = 5
Total number of subbands = 5
Total number of targets with ALMA data = 4
--------------------------------
--------------------------------
Summary of '13CO (4-3)' observations at 440.765 GHz
--------------------------------
Number of projects = 1
Number of observations = 1
Number of unique subbands = 1
Total number of subbands = 1
Total number of targets with ALMA data = 1
--------------------------------
--------------------------------
Summary of '13CO (5-4)' observations at 550.926 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of '13CO (6-5)' observations at 661.067 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of '13CO (7-6)' observations at 771.184 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of '13CO (8-7)' observations at 881.273 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'C18O (1-0)' observations at 109.782 GHz
--------------------------------
Number of projects = 19
Number of observations = 24
Number of unique subbands = 25
Total number of subbands = 27
Total number of targets with ALMA data = 18
--------------------------------
--------------------------------
Summary of 'C18O (2-1)' observations at 219.56 GHz
--------------------------------
Number of projects = 7
Number of observations = 9
Number of unique subbands = 12
Total number of subbands = 12
Total number of targets with ALMA data = 6
--------------------------------
--------------------------------
Summary of 'C18O (3-2)' observations at 329.331 GHz
```

(continues on next page)

(continued from previous page)

```
--------------------------------
Number of projects = 2
Number of observations = 3
Number of unique subbands = 3
Total number of subbands = 3
Total number of targets with ALMA data = 2
--------------------------------
--------------------------------
Summary of 'C18O (4-3)' observations at 439.089 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'C18O (5-4)' observations at 548.831 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'C18O (6-5)' observations at 658.553 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'C18O (7-6)' observations at 768.252 GHz
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'C18O (8-7)' observations at 877.922 GHz
--------------------------------
No observations found.
--------------------------------
```

Example 2.5.2: search whether any redshifted CO, 13CO, and C18O lines were observed in the query results

```
[14]: CO_obs = alminer.CO_lines(observations, z=1, print_targets=False)
```

```
--------------------------------
Summary of 'CO (1-0)' observations at 115.2712018 GHz (57.636 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'CO (2-1)' observations at 230.538 GHz (115.269 GHz at z=1)
--------------------------------
Number of projects = 4
Number of observations = 5
Number of unique subbands = 4
Total number of subbands = 5
Total number of targets with ALMA data = 3
--------------------------------
--------------------------------
Summary of 'CO (3-2)' observations at 345.7959899 GHz (172.898 GHz at z=1)
--------------------------------
```

(continues on next page)

(continued from previous page)

```
Number of projects = 2
Number of observations = 3
Number of unique subbands = 3
Total number of subbands = 3
Total number of targets with ALMA data = 2
--------------------------------
--------------------------------
Summary of 'CO (4-3)' observations at 461.0407682 GHz (230.52 GHz at z=1)
--------------------------------
Number of projects = 10
Number of observations = 13
Number of unique subbands = 13
Total number of subbands = 13
Total number of targets with ALMA data = 8
--------------------------------
--------------------------------
Summary of 'CO (5-4)' observations at 576.2679305 GHz (288.134 GHz at z=1)
--------------------------------
Number of projects = 3
Number of observations = 6
Number of unique subbands = 6
Total number of subbands = 6
Total number of targets with ALMA data = 3
--------------------------------
--------------------------------
Summary of 'CO (6-5)' observations at 691.4730763 GHz (345.737 GHz at z=1)
--------------------------------
Number of projects = 3
Number of observations = 5
Number of unique subbands = 5
Total number of subbands = 5
Total number of targets with ALMA data = 4
--------------------------------
--------------------------------
Summary of 'CO (7-6)' observations at 806.651806 GHz (403.326 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'CO (8-7)' observations at 921.7997 GHz (460.9 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of '13CO (1-0)' observations at 110.2013218 GHz (55.101 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of '13CO (2-1)' observations at 220.3986195 GHz (110.199 GHz at z=1)
--------------------------------
Number of projects = 18
Number of observations = 23
Number of unique subbands = 23
Total number of subbands = 25
Total number of targets with ALMA data = 17
--------------------------------
```

(continues on next page)

(continued from previous page)

```
--------------------------------
Summary of '13CO (3-2)' observations at 330.5878671 GHz (165.294 GHz at z=1)
--------------------------------
Number of projects = 1
Number of observations = 2
Number of unique subbands = 2
Total number of subbands = 2
Total number of targets with ALMA data = 1
--------------------------------
--------------------------------
Summary of '13CO (4-3)' observations at 440.7651735 GHz (220.383 GHz at z=1)
--------------------------------
Number of projects = 6
Number of observations = 7
Number of unique subbands = 7
Total number of subbands = 7
Total number of targets with ALMA data = 5
--------------------------------
--------------------------------
Summary of '13CO (5-4)' observations at 550.9262851 GHz (275.463 GHz at z=1)
--------------------------------
Number of projects = 3
Number of observations = 4
Number of unique subbands = 4
Total number of subbands = 4
Total number of targets with ALMA data = 3
--------------------------------
--------------------------------
Summary of '13CO (6-5)' observations at 661.0672766 GHz (330.534 GHz at z=1)
--------------------------------
Number of projects = 2
Number of observations = 5
Number of unique subbands = 5
Total number of subbands = 5
Total number of targets with ALMA data = 4
--------------------------------
--------------------------------
Summary of '13CO (7-6)' observations at 771.184125 GHz (385.592 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of '13CO (8-7)' observations at 881.272808 GHz (440.636 GHz at z=1)
--------------------------------
Number of projects = 1
Number of observations = 1
Number of unique subbands = 1
Total number of subbands = 1
Total number of targets with ALMA data = 1
--------------------------------
--------------------------------
Summary of 'C18O (1-0)' observations at 109.7821734 GHz (54.891 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'C18O (2-1)' observations at 219.5603541 GHz (109.78 GHz at z=1)
```

(continues on next page)

(continued from previous page)

```
--------------------------------
Number of projects = 19
Number of observations = 24
Number of unique subbands = 25
Total number of subbands = 27
Total number of targets with ALMA data = 18
--------------------------------
--------------------------------
Summary of 'C18O (3-2)' observations at 329.3305525 GHz (164.665 GHz at z=1)
--------------------------------
Number of projects = 1
Number of observations = 2
Number of unique subbands = 2
Total number of subbands = 2
Total number of targets with ALMA data = 1
--------------------------------
--------------------------------
Summary of 'C18O (4-3)' observations at 439.0887658 GHz (219.544 GHz at z=1)
--------------------------------
Number of projects = 7
Number of observations = 9
Number of unique subbands = 12
Total number of subbands = 12
Total number of targets with ALMA data = 6
--------------------------------
--------------------------------
Summary of 'C18O (5-4)' observations at 548.8310055 GHz (274.416 GHz at z=1)
--------------------------------
Number of projects = 2
Number of observations = 3
Number of unique subbands = 3
Total number of subbands = 3
Total number of targets with ALMA data = 2
--------------------------------
--------------------------------
Summary of 'C18O (6-5)' observations at 658.5532782 GHz (329.277 GHz at z=1)
--------------------------------
Number of projects = 2
Number of observations = 3
Number of unique subbands = 3
Total number of subbands = 3
Total number of targets with ALMA data = 2
--------------------------------
--------------------------------
Summary of 'C18O (7-6)' observations at 768.2515933 GHz (384.126 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
--------------------------------
Summary of 'C18O (8-7)' observations at 877.9219553 GHz (438.961 GHz at z=1)
--------------------------------
No observations found.
--------------------------------
```

2.3 3. Plot results

This section introduces all the plotting functions that help visualise the queried observations:

- *3.1 - Plot an overview of the observations (alminer.plot_overview)*
- *3.2 - Plot an overview of a given line in the observations (alminer.plot_line_overview)*
- *3.3 - Plot observed frequencies in each band (alminer.plot_bands)*
- *3.4 - Plot observations in each band (alminer.plot_observations)*
- *3.5 - Sky distribution (alminer.plot_sky)*

General notes about the plotting functions:

- Most of the plotting functions have the option to mark frequencies of CO, 13CO, and C18O lines by toggling *mark_CO=True*. A redshift can be provided by setting the *z* parameter that will shift the marked frequencies accordingly.
- Most of the plotting functions have the option to mark a list of frequencies. A redshift can be provided by setting the *z* parameter that will shift the marked frequencies accordingly.
- Plots can be saved in PDF format by setting the *savefig* parameter to the desired filename. Figures are saved in a subdirectory called 'reports' within the current working directory. If the directory doesn't exist, it will be created.
- To avoid displaying the plot (for example to create and save plots in a loop), set *showfig=False*.

Load libraries & create a query

To explore these options, we will first query the archive using one of the methods presented in the previous section and use the results in the remainder of this tutorial.

[1]:
```
import alminer

observations = alminer.keysearch({'science_keyword':['Galaxy chemistry']},
                                 print_targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 54
Number of observations = 376
Number of unique subbands = 1288
Total number of subbands = 1501
Total number of targets with ALMA data = 79
--------------------------------
```

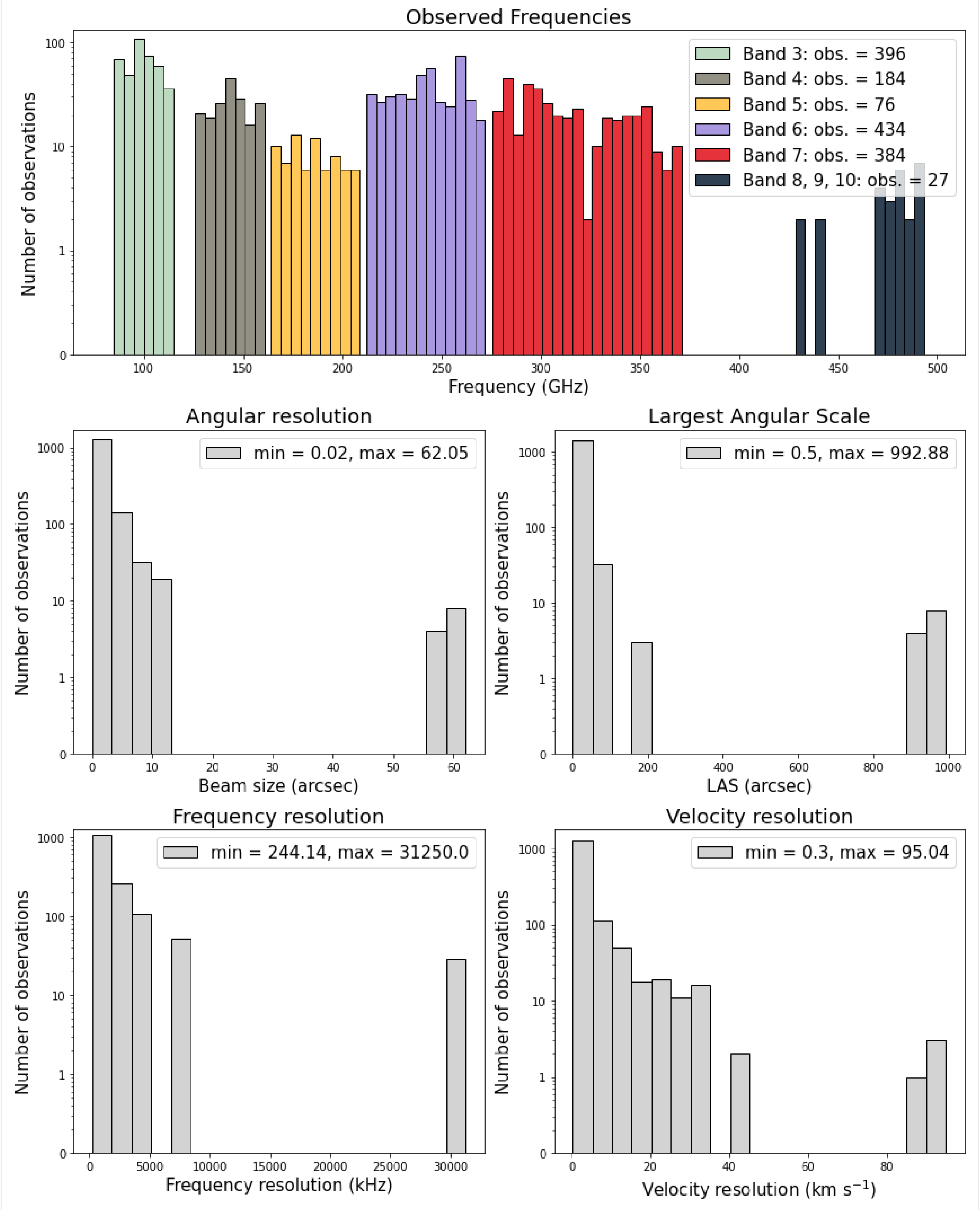
2.3.1 3.1 Plot an overview of the observations

The *alminer.plot_overview* function plots a summary of the observed frequencies, angular resolution, largest angular scales (LAS), and frequency and velocity resolutions.

Note: The histogram of observed frequencies displays the distribution of central frequencies and also depends on the choice of binning. To get an overview plot for a specific frequency, use *alminer.plot_observations* and *alminer.plot_line_overview* (see examples below).

Example 3.1.1: plot an overview of the observations and save the figure

```
[2]: alminer.plot_overview(observations, savefig='alma_galaxy_chemistry')
```

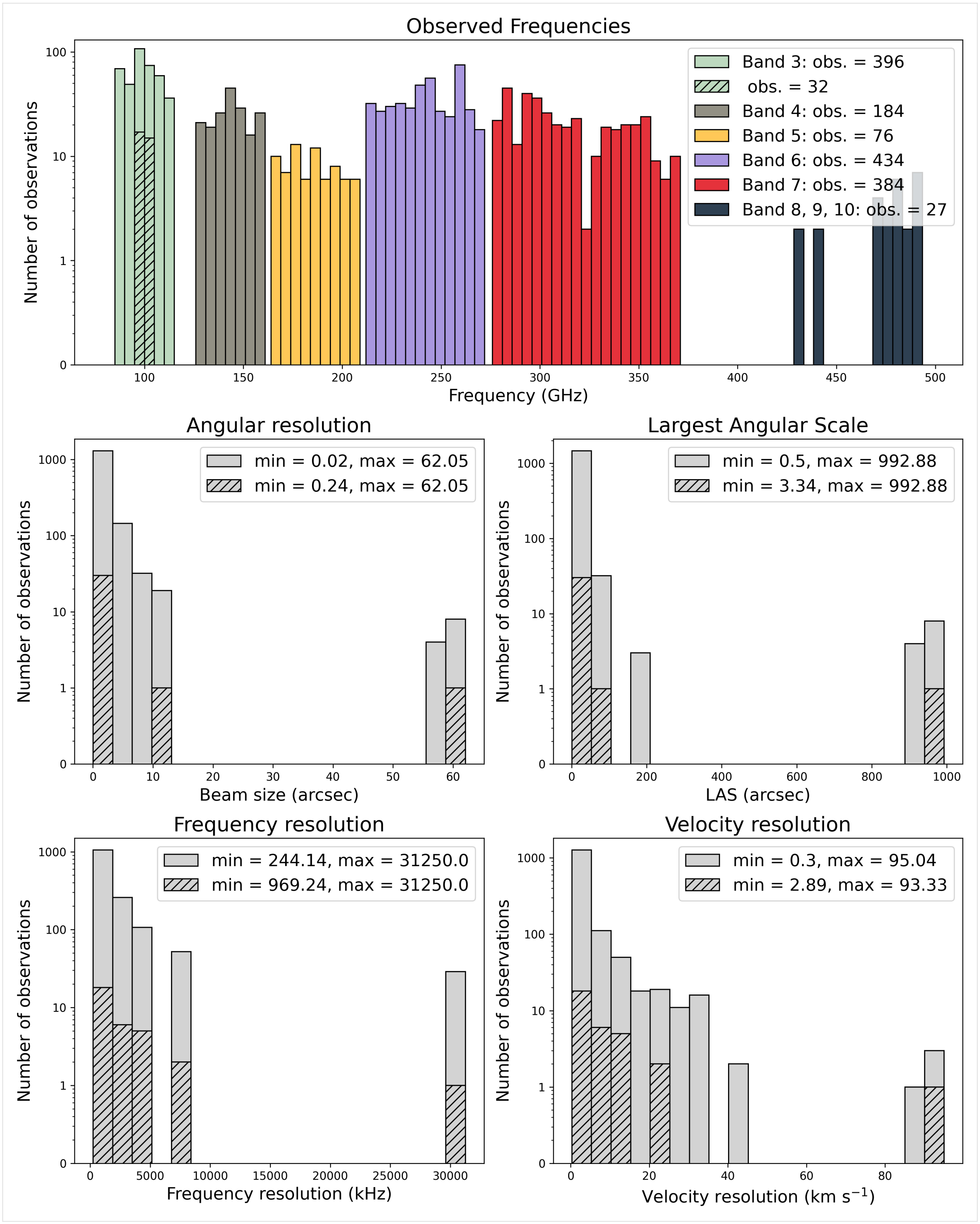


2.3.2 3.2 Plot an overview of a given line in the observations

The *alminer.plot_line_overview* function creates overview plots of observed frequencies, angular resolution, LAS, frequency and velocity resolutions for the input DataFrame, and highlights the observations of a give frequency (redshift if the *z* parameter is set) specified by the user.

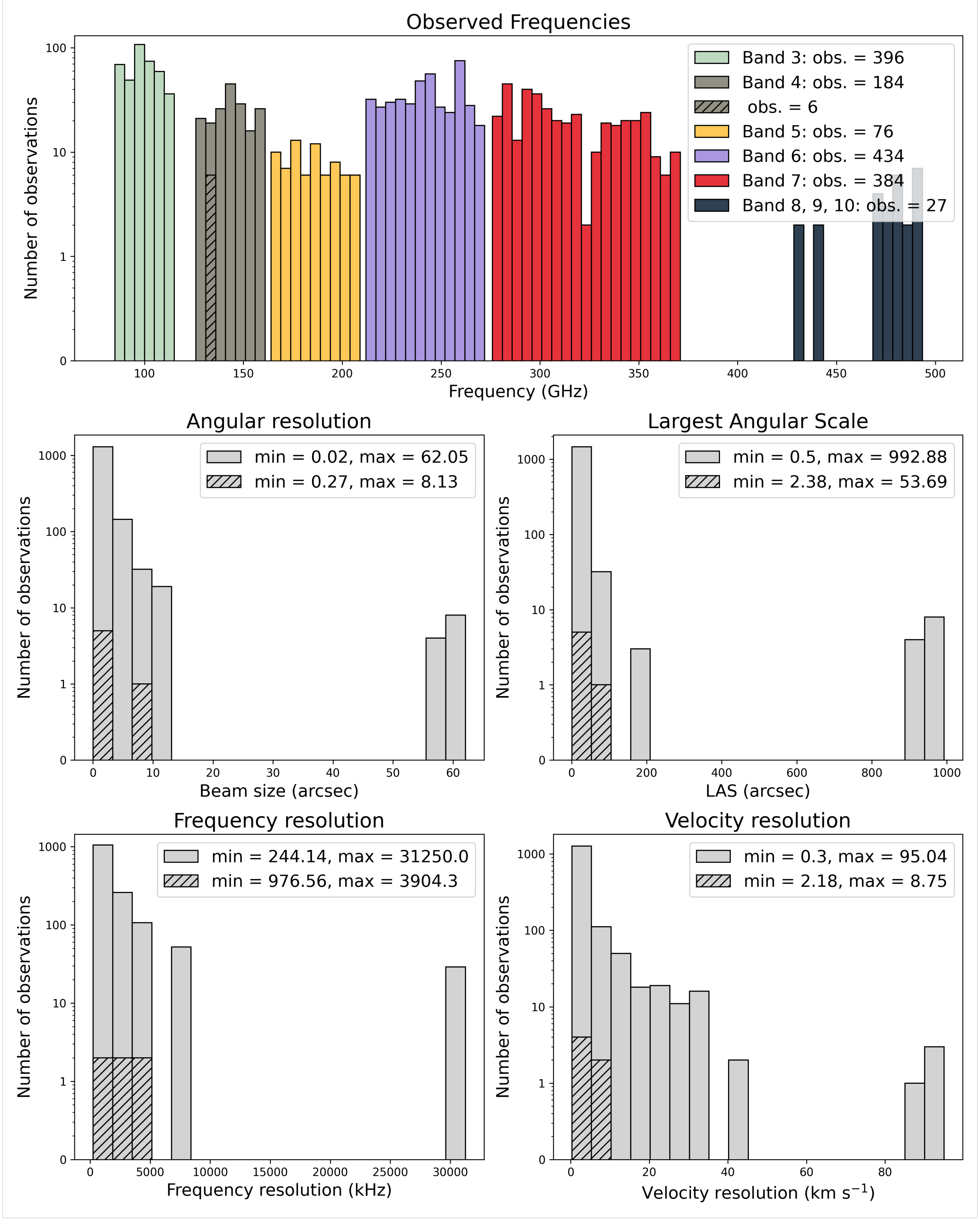
Example 3.2.1: plot an overview of the observations and highlight observations at a particular frequency

```
[3]: alminer.plot_line_overview(observations, line_freq=100.0)
```


Observed Frequencies
Band 3: obs. = 396
obs. = 32
Band 4: obs. = 184
Band 5: obs. = 76
Band 6: obs. = 434
Band 7: obs. = 384
Band 8, 9, 10: obs. = 27
Number of observations
Frequency (GHz)
Angular resolution
min = 0.02, max = 62.05
min = 0.24, max = 62.05
Beam size (arcsec)
Largest Angular Scale
min = 0.5, max = 992.88
min = 3.34, max = 992.88
LAS (arcsec)
Frequency resolution
min = 244.14, max = 31250.0
min = 969.24, max = 31250.0
Frequency resolution (kHz)
Velocity resolution
min = 0.3, max = 95.04
min = 2.89, max = 93.33
Velocity resolution (km s^{-1})

Example 3.2.2: plot an overview of the observations and highlight observations at a redshifted frequency

```
[4]: alminer.plot_line_overview(observations, line_freq=400.0, z=2)
```

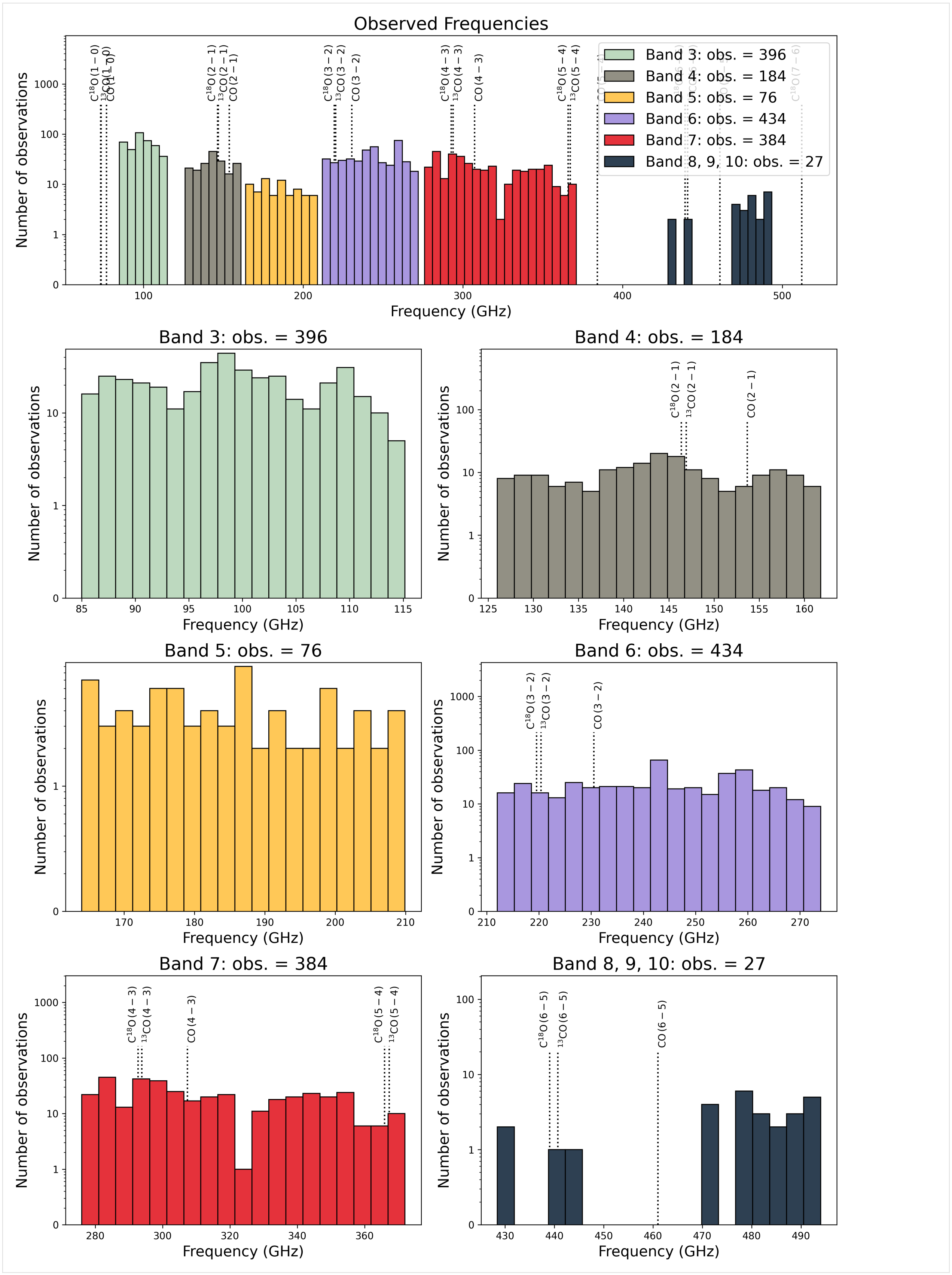


2.3.3 3.3 Plot observed frequencies in each band

The *alminer.plot_bands* function creates detailed plots of observed frequencies in each band.

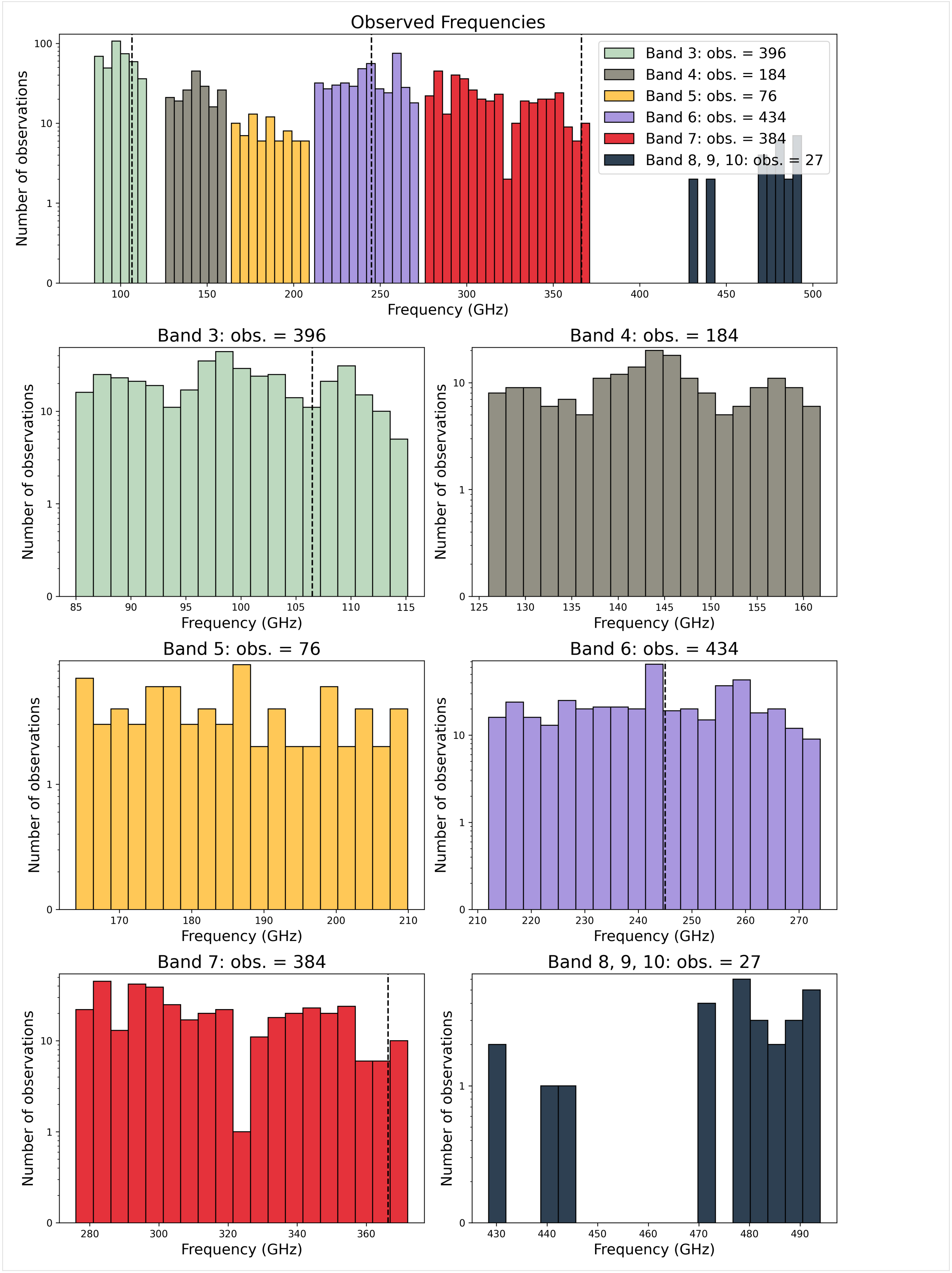
Example 3.3.1: plot observed frequencies in each band and mark redshifted CO lines

```
[5]: alminer.plot_bands(observations, mark_CO=True, z=0.5)
```


Observed Frequencies
Band 3: obs. = 396
Band 4: obs. = 184
Band 5: obs. = 76
Band 6: obs. = 434
Band 7: obs. = 384
Band 8, 9, 10: obs. = 27
Number of observations
Frequency (GHz)
$C^{18}O\,(1-0)$
$^{13}CO\,(1-0)$
$CO\,(1-0)$
$C^{18}O\,(2-1)$
$^{13}CO\,(2-1)$
$CO\,(2-1)$
$C^{18}O\,(3-2)$
$^{13}CO\,(3-2)$
$CO\,(3-2)$
$C^{18}O\,(4-3)$
$^{13}CO\,(4-3)$
$CO\,(4-3)$
$C^{18}O\,(5-4)$
$^{13}CO\,(5-4)$
$C^{18}O\,(6-5)$
$^{13}CO\,(6-5)$
$CO\,(6-5)$
$C^{18}O\,(7-6)$

Example 3.3.2: plot observed frequencies in each band and mark frequencies of choice

```
[6]: alminer.plot_bands(observations, mark_freq=[106.5, 245.0, 366.3])
```

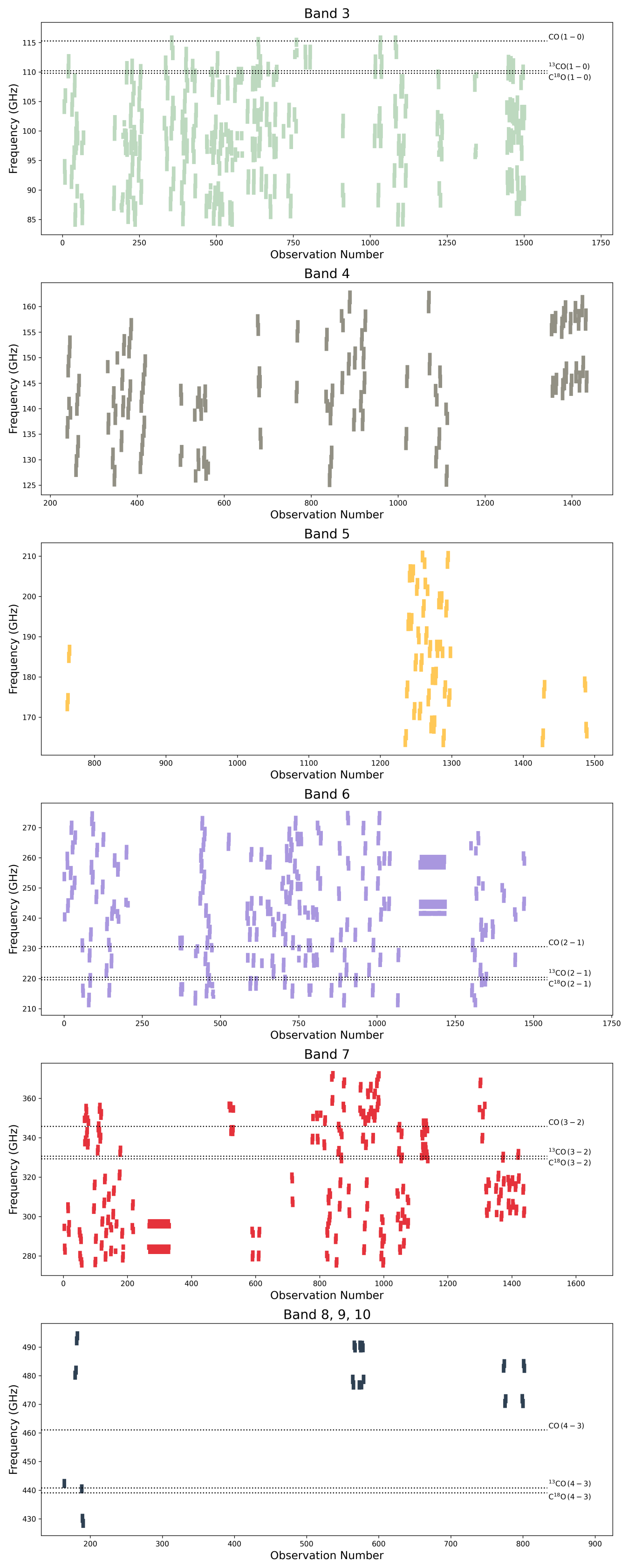

Observed Frequencies
Band 3: obs. = 396
Band 4: obs. = 184
Band 5: obs. = 76
Band 6: obs. = 434
Band 7: obs. = 384
Band 8, 9, 10: obs. = 27
Number of observations
Frequency (GHz)
Band 3: obs. = 396
Band 4: obs. = 184
Band 5: obs. = 76
Band 6: obs. = 434
Band 7: obs. = 384
Band 8, 9, 10: obs. = 27

2.3.4 3.4 Plot observed frequencies in each band

The *alminer.plot_observations* function creates a detailed plot of observations in each band showing the exact observed frequency ranges. Observation numbers are the input DataFrame's index values.

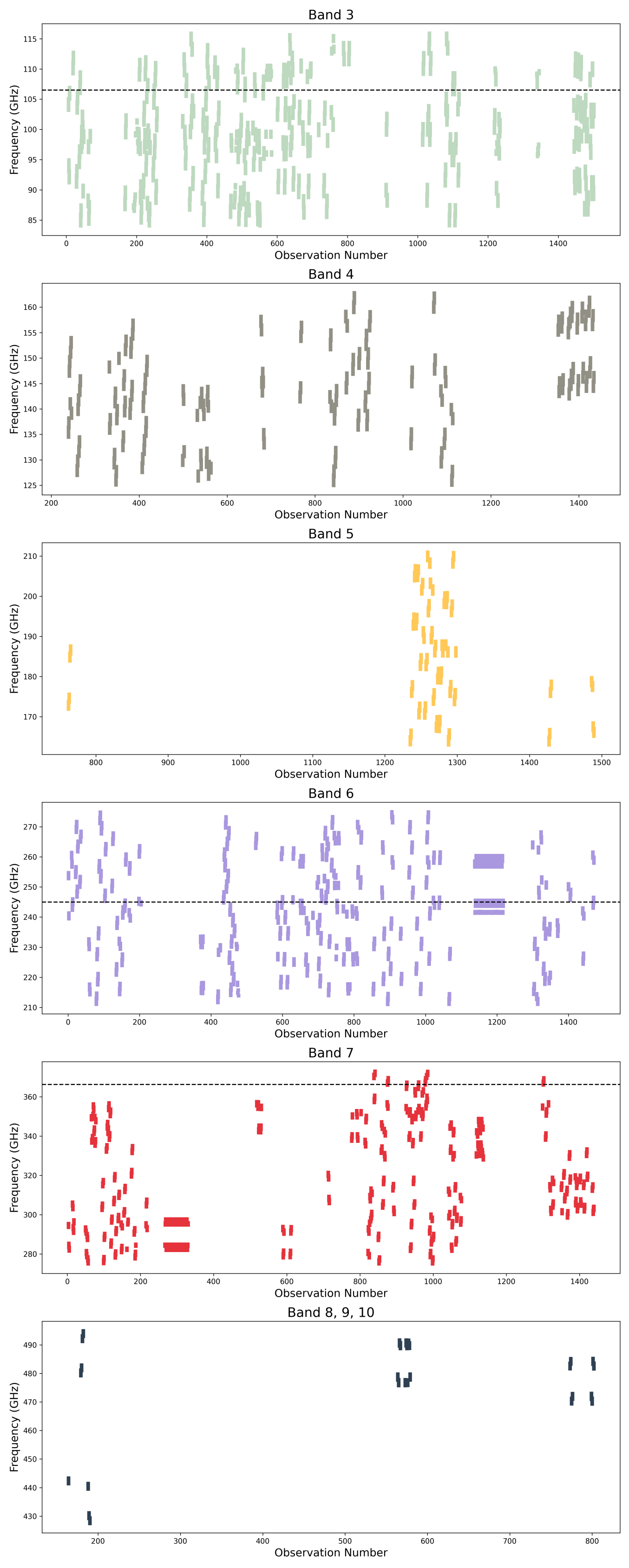
Example 3.4.1: plot observed frequencies in each band and mark CO lines

```
[7]: alminer.plot_observations(observations, mark_CO=True)
```


Band 3
Frequency (GHz)
Observation Number
CO (1 − 0)
13CO(1 − 0)
C18O (1 − 0)
Band 4
Band 5
Band 6
CO (2 − 1)
13CO (2 − 1)
C18O (2 − 1)
Band 7
CO (3 − 2)
13CO (3 − 2)
C18O (3 − 2)
Band 8, 9, 10
CO (4 − 3)
13CO (4 − 3)
C18O (4 − 3)

Example 3.4.2: plot observed frequencies and mark frequencies of choice

```
[8]: alminer.plot_observations(observations, mark_freq=[106.5, 245.0, 366.3])
```

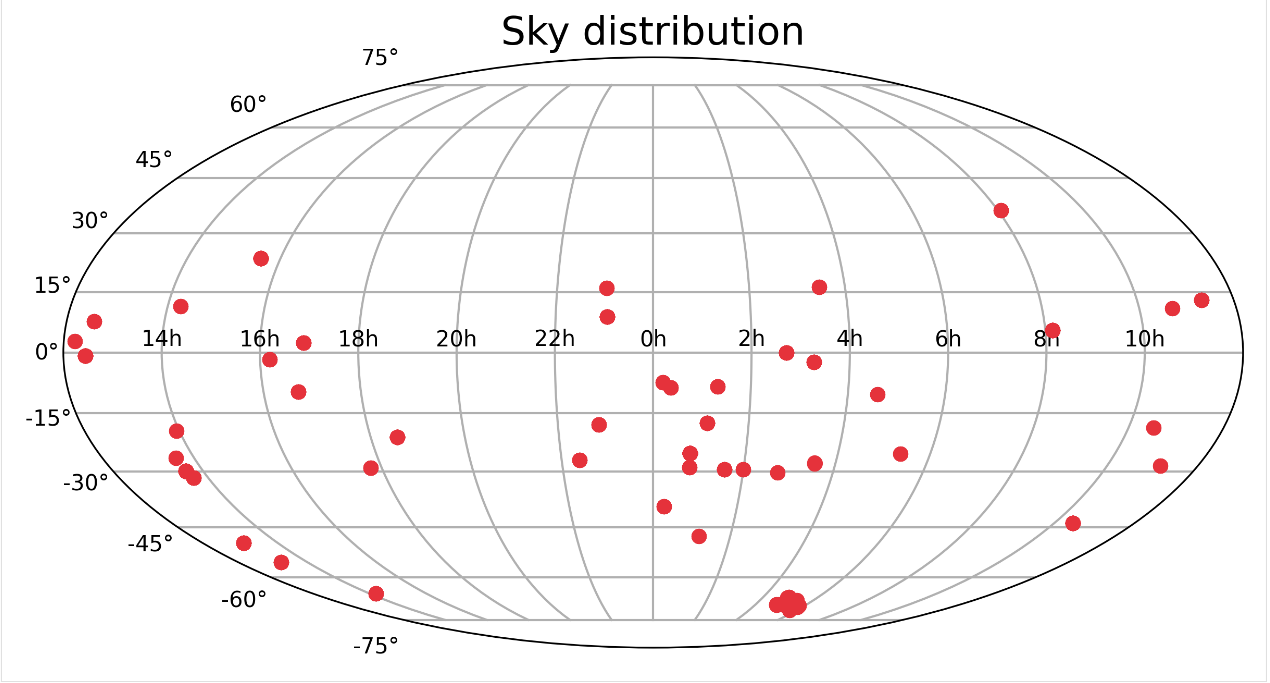

Band 3
Frequency (GHz)
Observation Number
Band 4
Frequency (GHz)
Observation Number
Band 5
Frequency (GHz)
Observation Number
Band 6
Frequency (GHz)
Observation Number
Band 7
Frequency (GHz)
Observation Number
Band 8, 9, 10
Frequency (GHz)
Observation Number

2.3.5 3.5 Plot sky distribution

The *alminer.plot_sky* function creates a plot of the distribution of targets on the sky.

Example 3.5.1: plot sky distribution

[9]:
```
alminer.plot_sky(observations)
```



2.4 4. Create reports

This section introduces different ways to save query results:

- *4.1 - Export results as a table (alminer.save_table)*
- *4.2 - Save overview plots for each target (alminer.save_source_reports)*

Load libraries & create a query

To explore these options, we will first query the archive using one of the methods presented in the previous section and use the results in the remainder of this tutorial.

[1]:
```
import alminer

observations = alminer.keysearch({'science_keyword':['"Galaxy chemistry"']},
                                 print_targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 48
Number of observations = 341
```

(continues on next page)

(continued from previous page)

```
Number of unique subbands = 1166
Total number of subbands = 1368
Total number of targets with ALMA data = 64
--------------------------------
```

2.4.1 4.1 Export results as a table

The *alminer.save_table* function writes the provided DataFrame to a table in CSV format in the 'tables' folder within the current working directory. If the 'tables' folder does not exist, it will be created.

Example 4.1.1: save query results as a table

```
[2]: alminer.save_table(observations, filename="galaxy_chemistry")
```

2.4.2 4.2 Save overview plots

The *alminer.save_source_reports* function creates overview plots of observed frequencies, angular resolution, LAS, frequency and velocity resolutions for each source in the provided DataFrame and saves them in PDF format in the 'reports' folder in the current working directory. If the 'reports' folder does not exist, it will be created. The reports are named after the target names.

Note: Currently, the grouping is done based on ALMA target names, so the same source with a slighly different naming schemes will be treated as separate targets.

Example 4.2.1: save overview plots of each target with CO lines marked

Let's first narrow down our large query to a smaller subset to only a range of frequencies (Band 3) and angular resolutions < 0.5":

```
[3]: selected = observations[(observations["min_freq_GHz"] > 80.0) &
                             (observations["max_freq_GHz"] < 115.0) &
                             (observations["ang_res_arcsec"] < 0.5)]
     alminer.summary(selected)
```

```
--------------------------------
Number of projects = 7
Number of observations = 16
Number of unique subbands = 61
Total number of subbands = 61
7 target(s) with ALMA data = ['NGC1266', 'Arp220', 'ngc6240', 'n613', 'NGC4418',
↪'NGC7469', 'Cloverleaf']
--------------------------------
```

Now we can create and save plots for each source, with CO and its isotopologues marked:

```
[4]: alminer.save_source_reports(selected, mark_CO=True)
```

2.5 5. Download data

The *alminer.download_data* function allows the user to download the data from the archive directly to a location on the local disk.

General notes about the download function:

- The default download location is the 'data' subdirectory in the current working directory. The desired location can be changed by setting the *location* parameter to the desired path.
- The archive mirror used for downloading can be specified through the *archive_mirror* parameter. ESO is the default, and other options are NRAO and NAOJ.
- To check the amount of disk space needed, the *dryrun* parameter can be toggled to *True* which will only stage the data and write to the terminal how much space is required.
- By default, tar files (including both raw and FITS data products) associated with uids in the provided DataFrame will be downloaded.
- To download only the FITS data products, the *fitsonly* parameter can be toggled to *True*.
- It is possible to provide a list of strings (to the *filename_must_include* parameter) that the user wants to be included in the filenames that are downloaded. This is useful to restrict the download further, for example, to data that have been primary beam corrected ('.pbcor') or that have the science target ('_sci' or the ALMA target name). The choice is largely dependent on the cycle and type of reduction that was performed, and data products that exist on the archive as a result.
- A list of URLs (files) to be downloaded from the archive can be printed to the terminal by setting *print_urls*=*True*.

Load libraries & create a query

To explore these options, we will first query the archive using one of the methods presented in the previous section and use the results in the remainder of this tutorial.

[1]:
```
import alminer

observations = alminer.keysearch({'target_name':['G31.41'], 'proposal_id': ['2018']})
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 2
Number of observations = 3
Number of unique subbands = 9
Total number of subbands = 12
1 target(s) with ALMA data = ['G31.41+0.31']
--------------------------------
```

2.5.1 Example 5.1: download all data products (raw + products)

[2]:
```
alminer.download_data(observations, fitsonly=False, dryrun=True,
                      location='./data', print_urls=False)
```

```
================================
This is a dryrun. To begin download, set dryrun=False.
================================
Download location = ./data
```

(continues on next page)

(continued from previous page)

```
Total number of Member OUSs to download = 3
Selected Member OUSs: ['uid://A001/X133d/X325', 'uid://A001/X133d/X327', 'uid://A001/
↪X133d/X21b4']
Number of files to download = 13
Needed disk space = 450.5 GB
--------------------------------
```

2.5.2 Example 5.2: download only continuum FITS images for the science target

[3]:
```
alminer.download_data(observations, fitsonly=True, dryrun=True, location='./data',
                      filename_must_include=['_sci', '.pbcor', 'cont', 'G31.41'],
                      print_urls=True)
```

```
================================
This is a dryrun. To begin download, set dryrun=False.
================================
Download location = ./data
Total number of Member OUSs to download = 3
Selected Member OUSs: ['uid://A001/X133d/X325', 'uid://A001/X133d/X327', 'uid://A001/
↪X133d/X21b4']
Number of files to download = 4
Needed disk space = 48.9 MB
File URLs to download = https://almascience.eso.org/dataPortal/member.uid___A001_
↪X133d_X325._G31.41p0.31__sci.spw25_27_29_31.cont.I.tt0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X133d_X325._G31.41p0.31__sci.
↪spw25_27_29_31.cont.I.tt1.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X133d_X327._G31.41p0.31__sci.
↪spw25_27_29_31.cont.I.tt0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X133d_X327._G31.41p0.31__sci.
↪spw25_27_29_31.cont.I.tt1.pbcor.fits
--------------------------------
```

2.6 6. Advanced query features

This Section introduces:

- *6.1 - Create and run your own TAP query (alminer.run_query)*
- *6.2 - Convert results to ALminer format (alminer.filter_results)*

Load alminer

[2]:
```
import alminer
```

2.6.1 6.1 Create and run your own TAP query

You can use Astronomical Data Query Language (ADQL) to create more complex queries relevant for your work.

The ALminer querying functions provide an option to print the query string that was used to search the ALMA archive for the user by setting *print_query* parameter to *True*.

Once you have created the query string of interest, you can run it using the *alminer.run_query* function.

Example 6.1.1: Retrieve the ADQL query string used in ALminer query functions

```
[7]: obs = alminer.keysearch({'proposal_abstract': ['"planet-forming disk"']},
                             print_targets=False, print_query=True)
```

```
================================
alminer.keysearch results
================================
Your query is: SELECT * FROM ivoa.obscore WHERE ((LOWER(proposal_abstract) LIKE '
↪%planet-forming disk%')) AND (LOWER(data_rights) LIKE '%public%') AND (LOWER(scan_
↪intent) LIKE '%target%') ORDER BY proposal_id
--------------------------------
Number of projects = 15
Number of observations = 130
Number of unique subbands = 115
Total number of subbands = 665
Total number of targets with ALMA data = 113
--------------------------------
```

Example 6.1.2: Modify the query string and run the query

In the previous example, we searched the ALMA archive for projects with the phrase 'planet forming disk' in their abstracts. But let's say you want to include the filtering options directly through the TAP query, for example to only query observations with angular resolutions less than 0.5".

You can modify the query string accordingly:

```
[8]: query_str = "SELECT * FROM ivoa.obscore WHERE ((LOWER(proposal_abstract) LIKE '
     ↪%planet-forming disk%')) AND (spatial_resolution < 0.5) AND (LOWER(data_rights)␣
     ↪LIKE '%public%') AND (LOWER(scan_intent) LIKE '%target%') ORDER BY proposal_id"
```

And run the query

```
[9]: myquery = alminer.run_query(query_str)
```

2.6.2 6.2 Convert your own query results into ALminer format

To make use of other `alminer` functions, the resulting DataFrame returned from running your query has to be converted to `alminer` format where a few useful columns are added to the DataFrame. This can be done through the *alminer.filter_results* function.

Example 6.2.1: Convert query results to ALminer format

```
[10]: myquery_obs = alminer.filter_results(myquery)
```

```
--------------------------------
Number of projects = 10
Number of observations = 102
Number of unique subbands = 61
Total number of subbands = 509
100 target(s) with ALMA data = ['HD_163296', 'TW_Hya', 'LupusIII_80', 'LupusIII_115',
↪'LupusIII_74', 'LupusI_10', 'LupusIII_137', 'LupusIII_73', 'LupusIV_142', 'LupusIII_
↪121', 'LupusIII_72', 'LupusIII_133', 'LupusIII_1004', 'LupusIII_1013', 'LupusIII_70
↪', 'LupusIII_132', 'LupusIII_38', 'LupusIII_103', 'LupusIII_106', 'LupusIII_57',
↪'LupusIII_1007', 'LupusIV_153', 'LupusIII_18', 'LupusIII_34', 'LupusIII_60',
↪'LupusIII_1008', 'LupusIII_85', 'LupusI_11', 'LupusIII_43', 'LupusIII_141',
↪'LupusIII_79', 'LupusIII_1009', 'LupusIII_44', 'LupusIII_42', 'LupusIII_37',
↪'LupusIII_109', 'LupusI_5', 'LupusIV_144', 'LupusIII_99', 'LupusIII_120', 'LupusIII_
↪89', 'LupusIII_76', 'LupusIII_82', 'LupusIII_28', 'LupusI_14', 'LupusIII_51',
↪'LupusIII_88', 'LupusIII_75', 'LupusIII_1010', 'LupusIII_68', 'LupusIII_67',
↪'LupusIII_71', 'LupusIII_26', 'LupusIII_66', 'LupusIII_130', 'LupusIII_53',
↪'LupusIII_33', 'LupusIII_94', 'LupusIII_40', 'LupusIII_91', 'LupusIV_159',
↪'LupusIII_65', 'LupusIII_116', 'LupusIII_111', 'LupusIII_19', 'LupusIII_21',
↪'LupusI_13', 'LupusI_15', 'LupusIII_1005', 'LupusIII_1006', 'LupusIII_1003',
↪'LupusIII_1002', 'LupusIII_49', 'LupusI_12', 'LupusI_16', 'LupusIII_1001',
↪'LupusIII_1015', 'LupusIII_50', 'LupusI_4', 'LupusIII_114', 'LupusIV_145',
↪'LupusIII_30', 'LupusIII_52', 'LupusIV_151', 'LupusIV_150', 'LupusIII_87', 'LupusIV_
↪148', 'LupusIV_147', 'LupusIII_113', 'LupusIII_102', 'LupusIII_1014', 'LupusIII_41',
↪ 'Serpens-FIRS1', 'V4046_Sgr', 'DG_Tau', 'AS205A', 'RY_Tau', 'ex_lup', 'EM_star_SR_
↪24_N', 'EM_star_SR_20']
--------------------------------
```

Now you can use all the analysis and plotting routines presented before on these observations.

2.7 Scientific categories

Below is a table of possible ALMA science categories (as of July 2021) that can be provided to the 'scientific_category' keyword in the *alminer.keysearch* function:

ALMA science category
Active galaxies
Cosmology
Disks and planet formation
Galaxy evolution
ISM and star formation
Local Universe
Solar system
Stars and stellar evolution
Sun

2.8 Science keywords

Below is a table of possible ALMA science categories (as of July 2021) that can be provided to the 'science_keyword' keyword in the *alminer.keysearch* function:

ALMA science keyword
Active Galactic Nuclei (AGN)/Quasars (QSO)
Astrochemistry
Asymptotic Giant Branch (AGB) stars
Black holes
Brown dwarfs
Cataclysmic stars
Cosmic Microwave Background (CMB)/Sunyaev-Zel'dovich Effect (SZE)
Damped Lyman Alpha (DLA) systems
Debris disks
Disks around high-mass stars
Disks around low-mass stars
Dwarf/metal-poor galaxies
Early-type galaxies
Evolved stars - Chemistry
Evolved stars - Shaping/physical structure
Exo-planets
Galactic centres/nuclei
Galaxy Clusters
Galaxy chemistry
Galaxy groups and clusters
Galaxy structure & evolution
Gamma Ray Bursts (GRB)
Giant Molecular Clouds (GMC) properties
Gravitational lenses
HII regions
High-mass star formation
High-z Active Galactic Nuclei (AGN)
Hypergiants
Infra-Red Dark Clouds (IRDC)
Inter-Stellar Medium (ISM)/Molecular clouds
Intermediate-mass star formation
Low-mass star formation
Luminous Blue Variables (LBV)
Luminous and Ultra-Luminous Infra-Red Galaxies (LIRG & ULIRG)
Lyman Alpha Emitters/Blobs (LAE/LAB)
Lyman Break Galaxies (LBG)
Magellanic Clouds
Main sequence stars
Merging and interacting galaxies
Outflows, jets, feedback
Outflows, jets and ionized winds
Photon-Dominated Regions (PDR)/X-Ray Dominated Regions (XDR)
Post-AGB stars
Pre-stellar cores
Pulsars and neutron stars

continues on next page

Table 1 – continued from previous page

ALMA science keyword
Solar system - Asteroids
Solar system - Comets
Solar system - Planetary atmospheres
Solar system - Planetary surfaces
Solar system - Trans-Neptunian Objects (TNOs)
Spiral galaxies
Starburst galaxies
Starbursts, star formation
Sub-mm Galaxies (SMG)
Supernovae (SN) ejecta
Surveys of galaxies
The Sun
Transients
White dwarfs

2.9 Query keywords

Below is a table of possible keywords that can be used to query the ALMA Science Archive using the *alminer.keysearch* function:

ALMA query keyword	Type	Description
access_format	char(9)	Content format of the data
access_url	char(72*)	URL to download the data
antenna_arrays	char(660*)	Blank-separated list of Pad:Antenna pairs, i.e., A109:DV09 J504:DV02 J505:DV05 for antennas DV09, DV02 and DV05 sitting on pads A109, J504, and J505, respectively.
asdm_uid	char(32*)	UID of the ASDM containing this Field.
authors	char(4000*)	Full list of first author and all co-authors
band_list	char(30*)	Space delimited list of bands
bib_reference	char(30*)	Bibliography code
data_rights	char(11)	Access to data.
dataproduct_type	char(5*)	type of product
facility_name	char(3)	telescope name
first_author	char(256*)	The first author as provided by telbib.eso.org.
frequency_support	char(4000*)	All frequency ranges used by the field
group_ous_uid	char(64*)	Group OUS ID
instrument_name	char(4)	instrument name
is_mosaic	char(1)	Flag to indicate if this ASDM represents a mosaic or not.
lastModified	char(*)	Time stamp of last modification of the metadata
member_ous_uid	char(64*) |	Member OUS ID
o_ucd	char(35) |	UCD describing the observable axis (pixel values)
obs_collection	char(4)	short name for the data collection
obs_creator_name |	char(256*) |	case-insensitive partial match over the full PI name. Wildcards can be used
obs_id	char(64*) |	internal dataset identifier
obs_publisher_did	char(33*)	publisher dataset identifier

continues on next page

Table 2 – continued from previous page

ALMA query keyword	Type	Description
obs_release_date	char(*)	timestamp of date the data becomes publicly available
obs_title	char(256*)	Case-insensitive search over the project title
pol_states	char(64*)	polarization states present in the data
proposal_abstract	char(4000*)	Text search on the proposal abstract. Only abstracts will be returned which contain the given text. The search is case-insensitive.
proposal_authors	char(2000*)	Full name of CoIs.
proposal_id	char(64*)	Identifier of proposal to which NO observation belongs.
pub_abstract	char(4000*)	Case insensitive text search through the abstract of the publication.
pub_title	char(256*)	Case insensitive search through the title of the publication.
qa2_passed	char(1)	Quality Assessment 2 status: does the Member / Group OUS fulfil the PI's requirements?
s_region	char(*)	region bounded by observation
scan_intent	char(256*)	Scan intent list for the observed field.
schedblock_name	char(128*)	Name of the Scheduling Block used as a template for executing the ASDM containing this Field.
science_keyword	char(200*)	Chosen by the PI in the observing tool at the time of proposal submission. For an overview, see Appendix D of the ALMA Proposer's Guide. For a precise list, see a *this table of science keywords*.
science_observation	char(1)	Flag to indicate whether this is a science observation.
scientific_category	char(200*)	Chosen by the PI in the observing tool at the time of proposal submission. For an overview, see Appendix D of the ALMA Proposer's Guide. For a precise list, see *this table of scientific categories*.
target_name	char(256*)	name of intended target
type	char(16*)	Type flags.

2.10 API

Note: This documentation is automatically generated from docstrings in the code. Please open an issue on GitHub if you find some missing documentation or if you have suggestions for improvements.

2.10.1 ALminer: ALMA archive mining and visualization toolkit

A package for mining the Atacama Large Millimeter/submillimeter Array (ALMA) data archive and visualizing the queried observations.

`alminer.`**`catalog`**(*target_df*, *search_radius=1.0*, *tap_service='ESO'*, *point=False*, *public=True*, *published=None*, *print_query=False*, *print_targets=True*)

Query the ALMA archive for a list of coordinates or a catalog of sources based on their coordinates.

Parameters

- **`target_df`** (*`pandas.DataFrame`*) – Source names and coordinates.

Index:
RangeIndex

Columns:
Name: Name, dtype: str, description: target name (can be numbers or dummy names) Name: RAJ2000, dtype: float64, description: right ascension in degrees (ICRS) Name: DEJ2000, dtype: float64, description: declination in degrees (ICRS)

- **search_radius** (*float, optional*) – (Default value = 1. arcmin) Search radius (in arcmin) around the source coordinates.
- **tap_service** (*str, optional*) – (Default value = 'ESO') The TAP service to use. Options are: 'ESO' for Europe (https://almascience.eso.org/tap), 'NRAO' for North America (https://almascience.nrao.edu/tap), or 'NAOJ' for East Asia (https://almascience.nao.ac.jp/tap)
- **point** (*bool, optional*) – (Default value = True) Search whether the specified position (ra, dec) is contained within any ALMA observations (point=True) or query all ALMA observations that overlap with a cone centred at the specified position (ra, dec) and extending the search_radius (point=False). In the case of point=True, the search_radius parameter is ignored.
- **public** (*bool, optional*) – (Default value = True) Search for public data (public=True), proprietary data (public=False), or both public and proprietary data (public=None).
- **published** (*bool, optional*) – (Default value = None) Search for published data only (published=True), unpublished data only (published=False), or both published and unpublished data (published=None).
- **print_query** (*bool, optional*) – (Default value = True) Print the ADQL TAP query to the terminal.
- **print_targets** (*bool, optional*) – (Default value = False) Print a list of targets with ALMA data (ALMA source names) to the terminal.

Return type
pandas.DataFrame containing the query results.

`alminer.`**`CO_lines`**(*observations*, *z=0.0*, *print_summary=True*, *print_targets=True*)

Determine how many CO, 13CO, and C18O lines were observed in the provided query DataFrame.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **z** (*float64, optional*) – (Default value = 0.) Redshift by which the frequencies should be shifted.
- **print_summary** (*bool, optional*) – (Default value = True) Print a summary of the observations for each (redshifted) CO, 13CO, and C18O line to the terminal.
- **print_targets** (*bool, optional*) – (Default value = True) Print the target names (ALMA source names) with ALMA data for each (redshifted) CO, 13CO, and C18O line to the terminal.

Return type
pandas.DataFrame containing all observations of (redshifted) CO, 13CO, and C18O lines.

`alminer.`**`conesearch`**(*ra*, *dec*, *search_radius=1.0*, *tap_service='ESO'*, *point=False*, *public=True*, *published=None*, *print_targets=True*, *print_query=False*)

Query the ALMA archive for a given position and radius around it.

Parameters

- **`ra`** (*`float`*) – Right ascension in degrees (ICRS).
- **`dec`** (*`float`*) – Declination in degrees (ICRS).
- **`search_radius`** (*`float, optional`*) – (Default value = 1. arcmin) Search radius (in arcmin) around the source coordinates.
- **`tap_service`** (*`str, optional`*) – (Default value = 'ESO') The TAP service to use. Options are: 'ESO' for Europe (https://almascience.eso.org/tap), 'NRAO' for North America (https://almascience.nrao.edu/tap), or 'NAOJ' for East Asia (https://almascience.nao.ac.jp/tap)
- **`point`** (*`bool, optional`*) – (Default value = True) Search whether the specified position (ra, dec) is contained within any ALMA observations (point=True) or query all ALMA observations that overlap with a cone centred at the specified position (ra, dec) and extending the search_radius (point=False). In the case of point=True, the search_radius parameter is ignored.
- **`public`** (*`bool, optional`*) – (Default value = True) Search for public data (public=True), proprietary data (public=False), or both public and proprietary data (public=None).
- **`published`** (*`bool, optional`*) – (Default value = None) Search for published data only (published=True), unpublished data only (published=False), or both published and unpublished data (published=None).
- **`print_query`** (*`bool, optional`*) – (Default value = True) Print the ADQL TAP query to the terminal.
- **`print_targets`** (*`bool, optional`*) – (Default value = False) Print a list of targets with ALMA data (ALMA source names) to the terminal.

Return type

pandas.DataFrame containing the query results

`alminer.`**`download_data`**(*observations*, *fitsonly=False*, *dryrun=False*, *print_urls=False*, *filename_must_include=''*, *location='./data'*, *archive_mirror='ESO'*)

Download ALMA data from the archive to a location on the local machine.

Parameters

- **`observations`** (*`pandas.DataFrame`*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **`fitsonly`** (*`bool, optional`*) – (Default value = False) Download individual fits files only (fitsonly=True). This option will not download the raw data (e.g. 'asdm' files), weblogs, or README files.
- **`dryrun`** (*`bool, optional`*) – (Default value = False) Allow the user to do a test run to check the size and number of files to download without actually downloading the data (dryrun=True). To download the data, set dryrun=False.
- **`print_urls`** (*`bool, optional`*) – (Default value = False) Write the list of urls to be downloaded from the archive to the terminal.
- **`filename_must_include`** (*`list of str, optional`*) – (Default value = '') A list of strings the user wants to be contained in the url filename. This is useful to restrict the download further, for example, to data that have been primary beam corrected ('.pbcor') or that have the science target or calibrators (by including their names). The choice is largely dependent on the cycle and type of reduction that was performed and data products that exist

on the archive as a result. In most recent cycles, the science target can be filtered out with the flag '_sci' or its ALMA target name.

- **`location`** (*`str, optional`*) – (Default value = ./data) directory where the downloaded data should be placed.
- **`archive_mirror`** (*`str, optional`*) – (Default value = 'ESO') The archive service to use. Options are: 'ESO' for Europe (https://almascience.eso.org), 'NRAO' for North America (https://almascience.nrao.edu), or 'NAOJ' for East Asia (https://almascience.nao.ac.jp)

`alminer.`**`explore`**(*observations*, *allcols=False*, *allrows=False*)

Control how much of the pandas.DataFrame with the query results is presented in the displayed table.

Parameters

- **`observations`** (*`pandas.DataFrame`*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **`allcols`** (*`bool, optional`*) – (Default value = False) Show all 81 columns (allcols=True), or the first 18 columns (allcols=False).
- **`allrows`** (*`bool, optional`*) – (Default value = False) Show all rows in the DataFrame (allrows=True), or just a summary (allrows=False).

Return type

pandas.DataFrame containing the query results displayed to the user interface as specified by the user.

`alminer.`**`filter_results`**(*TAP_df*, *print_targets=True*)

Add a few new useful columns to the pandas.DataFrame with the query results from the PyVO TAP service and return the full query DataFrame and optionally a summary of the results.

Parameters

- **`TAP_df`** (*`pandas.DataFrame`*) – This is likely the output of 'run_query' function.
- **`print_targets`** (*`bool, optional`*) – (Default value = True) Print a list of targets with ALMA data (ALMA source names) to the terminal.

Return type

pandas.DataFrame containing the query results.

`alminer.`**`get_description`**(*column*)

Print the description of a given column in the query results DataFrame.

`alminer.`**`get_units`**(*column*)

Print the units for a given column in the query results DataFrame.

`alminer.`**`get_info`**(*column*)

Print the description and units of a given column in the query results DataFrame.

Parameters

`column` (*`str`*) – A column in the pandas.DataFrame query table.

`alminer.`**`keysearch`**(*search_dict*, *tap_service='ESO'*, *public=True*, *published=None*, *print_query=False*, *print_targets=True*)

Query the ALMA archive for any (string-type) keywords defined in ALMA TAP system.

Parameters

- **`search_dict`** (*`dict[str, list of str]`*) – Dictionary of keywords in the ALMA archive and their values. Values must be formatted as a list. A list of valid keywords are stored in VALID_KEYWORDS_STR variable.

- **`tap_service`** (*`str, optional`*) – (Default value = 'ESO') The TAP service to use. Options are: 'ESO' for Europe (https://almascience.eso.org/tap), 'NRAO' for North America (https://almascience.nrao.edu/tap), or 'NAOJ' for East Asia (https://almascience.nao.ac.jp/tap)
- **`public`** (*`bool, optional`*) – (Default value = True) Search for public data (public=True), proprietary data (public=False), or both public and proprietary data (public=None).
- **`published`** (*`bool, optional`*) – (Default value = None) Search for published data only (published=True), unpublished data only (published=False), or both published and unpublished data (published=None).
- **`print_query`** (*`bool, optional`*) – (Default value = True) Print the ADQL TAP query to the terminal.
- **`print_targets`** (*`bool, optional`*) – (Default value = False) Print a list of targets with ALMA data (ALMA source names) to the terminal.

Return type
pandas.DataFrame containing the query results.

Notes

The power of this function is in combining keywords. When multiple keywords are provided, they are queried using 'AND' logic, but when multiple values are provided for a given keyword, they are queried using 'OR' logic. If a given value contains spaces, its constituents are queried using 'AND' logic. Words encapsulated

in quotation marks (either ' or ") are queried as phrases. Values for the 'target_name' keyword are queried with 'OR' logic.

Examples

keysearch({"proposal_abstract": ["high-mass star formation outflow disk"]})
will query the archive for projects with the words "high-mass" AND "star" AND "formation" AND "outflow" AND "disk" in their proposal abstracts.

keysearch({"proposal_abstract": ["high-mass", "star", "formation", "outflow", "disk"]})
will query the archive for projects with the words "high-mass" OR "star" OR "formation" OR "outflow" OR "disk" in their proposal abstracts.

keysearch({"proposal_abstract": ["'high-mass star formation' outflow disk"]})
will query the archive for projects with the phrase "high-mass star formation" AND the words "outflow" AND "disk" in their proposal abstracts.

keysearch({"proposal_abstract": ["'star formation'"], "scientific_category":['Galaxies']})
will query the archive for projects with the phrase "star formation" in their proposal abstracts AND projects that are within the scientific_category of 'Galaxies'.

`alminer.`**`line_coverage`**(*observations*, *line_freq*, *z=0.0*, *line_name=''*, *print_summary=True*, *print_targets=True*)

Determine how many observations were observed at a given frequency (+redshift).

Parameters

- **`observations`** (*`pandas.DataFrame`*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **`line_freq`** (*`float64`*) – Frequency of the line of interest in GHz.

- **z** (*float64, optional*) – (Default value = 0.) Redshift by which the frequency given in 'line_freq' parameter should be shifted.
- **line_name** (*str, optional*) – (Default value = '') Name of the line specified in 'line_freq'.
- **print_summary** (*bool, optional*) – (Default value = True) Print a summary of the observations to the terminal.
- **print_targets** (*bool, optional*) – (Default value = True) Print a list of targets with ALMA data (ALMA source names) to the terminal.

Return type

pandas.DataFrame containing all observations of line of interest.

alminer.**plot_line_overview**(*observations*, *line_freq*, *z=0.0*, *line_name=''*, *showfig=True*, *savefig=None*)

Create overview plots of observed frequencies, angular resolution, LAS, frequency and velocity resolutions, highlighting the observations of a give (redshifted) frequency with hatches on the bar plots.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **line_freq** (*float64*) – Frequency of the line of interest in GHz.
- **z** (*float64, optional*) – (Default value = 0.) Redshift by which the frequency given in 'line_freq' parameter should be shifted.
- **line_name** (*str, optional*) – (Default value = '') Name of the line specified in 'line_freq'.
- **showfig** (*bool, optional*) – (Default value = True) Display the plot (showfig=True) or not (showfig=False).
- **savefig** (*str, optional*) – (Default value = None) Filename (without an extension) for the plot to be saved as. Default file extension is PDF. Figure is saved in a subdirectory called 'reports' within the current working directory. If the directory doesn't exist, it will be created. Default quality is dpi=300.

alminer.**plot_overview**(*observations*, *mark_freq=''*, *z=0.0*, *mark_CO=False*, *showfig=True*, *savefig=None*)

Create overview plots of observed frequencies, angular resolution, LAS, frequency and velocity resolutions.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **mark_freq** (*list of float64, optional*) – (Default value = '') A list of frequencies to mark on the plot with dashed lines.
- **z** (*float64, optional*) – (Default value = 0.) Redshift by which the frequencies given in 'mark_freq' and 'mark_CO' parameters should be shifted. Currently only one redshift can be given for all targets.
- **mark_CO** (*bool, optional*) – (Default value = False) Mark CO, 13CO, and C18O frequencies on the plot with dashed lines.
- **showfig** (*bool, optional*) – (Default value = True) Display the plot (showfig=True) or not (showfig=False).
- **savefig** (*str, optional*) – (Default value = None) Filename (without an extension) for the plot to be saved as. Default file extension is PDF. Figure is saved in a subdirectory

called 'reports' within the current working directory. If the directory doesn't exist, it will be created. Default quality is dpi=300.

`alminer.`**`plot_observations`**(*observations*, *mark_freq=''*, *z=0.0*, *mark_CO=False*, *showfig=True*, *savefig=None*)

Create detailed plots of observations in each band. The x-axis displays the observation number 'Obs' column in the input DataFrame.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **mark_freq** (*list of float64, optional*) – (Default value = '') A list of frequencies to mark on the plot with dashed lines.
- **z** (*float64, optional*) – (Default value = 0.) Redshift by which the frequencies given in 'mark_freq' and 'mark_CO' parameters should be shifted. Currently only one redshift can be given for all targets.
- **mark_CO** (*bool, optional*) – (Default value = False) Mark CO, 13CO, and C18O frequencies on the plot with dashed lines.
- **showfig** (*bool, optional*) – (Default value = True) Display the plot (showfig=True) or not (showfig=False).
- **savefig** (*str, optional*) – (Default value = None) Filename (without an extension) for the plot to be saved as. Default file extension is PDF. Figure is saved in a subdirectory called 'reports' within the current working directory. If the directory doesn't exist, it will be created. Default quality is dpi=300.

`alminer.`**`plot_bands`**(*observations*, *mark_freq=''*, *z=0.0*, *mark_CO=False*, *showfig=True*, *savefig=None*)

Create overview and detailed plots of observed frequencies in each band.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **mark_freq** (*list of float64, optional*) – (Default value = '') A list of frequencies to mark on the plot with dashed lines.
- **z** (*float64, optional*) – (Default value = 0.) Redshift by which the frequencies given in 'mark_freq' and 'mark_CO' parameters should be shifted. Currently only one redshift can be given for all targets.
- **mark_CO** (*bool, optional*) – (Default value = False) Mark CO, 13CO, and C18O frequencies on the plot with dashed lines.
- **showfig** (*bool, optional*) – (Default value = True) Display the plot (showfig=True) or not (showfig=False).
- **savefig** (*str, optional*) – (Default value = None) Filename (without an extension) for the plot to be saved as. Default file extension is PDF. Figure is saved in a subdirectory called 'reports' within the current working directory. If the directory doesn't exist, it will be created. Default quality is dpi=300.

`alminer.`**`plot_sky`**(*observations*, *showfig=True*, *savefig=None*)

Plot the distribution of the targets on the sky.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **showfig** (*bool, optional*) – (Default value = True) Display the plot (showfig=True) or not (showfig=False).
- **savefig** (*str, optional*) – (Default value = None) Filename (without an extension) for the plot to be saved as. Default file extension is PDF. Figure is saved in a subdirectory called 'reports' within the current working directory. If the directory doesn't exist, it will be created. Default quality is dpi=300.

alminer.**run_query**(*query_str*, *tap_service='ESO'*)

Run the TAP query through PyVO service.

Parameters

- **query_str** (*str*) – ADQL query to send to the PyVO TAP service
- **tap_service** (*str, optional*) – (Default value = 'ESO') The TAP service to use. Options are: 'ESO' for Europe (https://almascience.eso.org/tap), 'NRAO' for North America (https://almascience.nrao.edu/tap), or 'NAOJ' for East Asia (https://almascience.nao.ac.jp/tap)

Return type

pandas.DataFrame containing the query results

alminer.**summary**(*observations*, *print_targets=True*)

Print a summary of the observations.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **print_targets** (*bool, optional*) – (Default value = True) Print a list of targets with ALMA data (ALMA source names) to the terminal.

alminer.**target**(*sources*, *search_radius=1.0*, *tap_service='ESO'*, *point=False*, *public=True*, *published=None*, *print_query=False*, *print_targets=True*)

Query targets by name.

This is done by using the astropy SESAME resolver to get the target's coordinates and then the ALMA archive is queried for those coordinates and a search_radius around them. The SESAME resolver searches multiple databases (Simbad, NED, VizieR) to parse names commonly found throughout literature and returns their coordinates. If the target is not resolved in any of these databases, consider using the 'keysearch' function and query the archive using the 'target_name' keyword (e.g. keysearch({'target_name': sources})).

Parameters

- **sources** (*str or list of str*) – list of sources by name. (IMPORTANT: source names must be identified by at least one of Simbad, NED, or Vizier)
- **search_radius** (*float, optional*) – (Default value = 1. arcmin) Search radius (in arcmin) around the source coordinates.
- **tap_service** (*str, optional*) – (Default value = 'ESO') The TAP service to use. Options are: 'ESO' for Europe (https://almascience.eso.org/tap), 'NRAO' for North America (https://almascience.nrao.edu/tap), or 'NAOJ' for East Asia (https://almascience.nao.ac.jp/tap)
- **point** (*bool, optional*) – (Default value = True) Search whether the specified position (ra, dec) is contained within any ALMA observations (point=True) or query all ALMA

observations that overlap with a cone centred at the specified position (ra, dec) and extending the search_radius (point=False). In the case of point=True, the search_radius parameter is ignored.

- **public** (*bool, optional*) – (Default value = True) Search for public data (public=True), proprietary data (public=False), or both public and proprietary data (public=None).
- **published** (*bool, optional*) – (Default value = None) Search for published data only (published=True), unpublished data only (published=False), or both published and unpublished data (published=None).
- **print_query** (*bool, optional*) – (Default value = True) Print the ADQL TAP query to the terminal.
- **print_targets** (*bool, optional*) – (Default value = False) Print a list of targets with ALMA data (ALMA source names) to the terminal.

Return type

pandas.DataFrame containing the query results.

See also:

keysearch

Query the ALMA archive for any (string-type) keywords defined in ALMA TAP system.

`alminer.`**`save_source_reports`**(*observations*, *mark_freq=''*, *z=0.0*, *mark_CO=False*)

Create overview plots of observed frequencies, angular resolution, LAS, frequency and velocity resolutions for each source in the provided DataFrame and save them in PDF format in the 'reports' subdirectory. If the directory doesn't exist, it will be created.

Parameters

- **observations** (*pandas.DataFrame*) – This is likely the output of e.g. 'conesearch', 'target', 'catalog', & 'keysearch' functions.
- **mark_freq** (*list of float64, optional*) – (Default value = '') A list of frequencies to mark on the plot with dashed lines.
- **z** (*float64, optional*) – (Default value = 0.) Redshift by which the frequencies given in 'mark_freq' and 'mark_CO' parameters should be shifted. Currently only one redshift can be given for all targets.
- **mark_CO** (*bool, optional*) – (Default value = False) Mark CO, 13CO, and C18O frequencies on the plot with dashed lines.

Notes

Reports will be grouped by ALMA target names, therefore the same source with many different ALMA names will be treated as individual unique targets (e.g. TW_Hya, TW Hya, twhya).

`alminer.`**`save_table`**(*observations*, *filename='mytable'*)

Write the DataFrame with the query results to a table in CSV format.

The table will be saved in the 'tables' subdirectory within the current working directory. If the directory doesn't exist, it will be created.

Parameters

- **observations** (*pandas.DataFrame*) –

- **`filename`** (*`str`*) – (Default value = "mytable") Name of the table to be saved in the 'tables' subdirectory.

CHAPTER

THREE

WHAT'S NEW

- You can now specify which archive mirror to download data from: [ESO](https://almascience.eso.org/aq) is the default, and other options are [NRAO](https://almascience.nrao.edu/aq) and [NAOJ](https://almascience.nao.ac.jp/aq). This option can be given through the *'archive_mirror'* parameter in the *download_data* function.
- You can now specify which archive service to query: [ESO](https://almascience.eso.org/tap) is the default, and other options are [NRAO](https://almascience.nrao.edu/tap) and [NAOJ](https://almascience.nrao.edu/tap). This option can be given through the *'tap_service'* parameter to all functions that do the query (e.g. keysearch, target, catalog). For example:
  - `alminer.target(["TW Hya", "HL Tau"], tap_service='NRAO')`
  - Note that currently the ESO service is not returning all results, hence it is advisable to test your queries with multiple services until further notice.
- It is now possible to query entire phrases with the *keysearch* function. For example:
  - `alminer.keysearch({'proposal_abstract': ['"high-mass star formation" outflow disk']})` will query the proposal abstracts for the phrase *high-mass star formation* AND the words *outflow* AND *disk*.
  - `alminer.keysearch({'proposal_abstract': ['"high-mass star formation" outflow disk', '"massive star formation" outflow disk']})` will query the the proposal abstracts for the phrase *high-mass star formation* AND the words *outflow* AND *disk* OR the phrase *massive star formation* AND the words *outflow* AND *disk*.

CHAPTER

FOUR

ACKNOWLEDGEMENTS

`alminer` has been developed through a collaboration between Allegro, the ALMA Regional Centre in The Netherlands, and the University of Vienna as part of the EMERGE-StG project. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 851435).

If you use `alminer` as part of your research, please consider citing this ASCL article (ADS reference will be added to the GitHub page when available).

`alminer` makes use of different routines in Astropy and Astroquery. Please also consider citing the following papers: - Astropy: Astropy Collaboration et al. 2013 - Astroquery: Ginsburg et al. 2019

We also acknowledge the work of Leiden University M.Sc. students, Robin Mentel and David van Dop, who contributed to early versions of this work.

CHAPTER

FIVE

CONTACT US

If you encounter issues, please open an issue.

If you have suggestions for improvement or would like to collaborate with us on this project, please e-mail Aida Ahmadi and Alvaro Hacar.

PYTHON MODULE INDEX

INDEX