Exploring the ALMA Science Archive with

Who we are

Aida Ahmadi

- Fellow at the ALMA Regional Center in The Netherlands (Allegro)
- Postdoctoral researcher at the Leiden Observatory

Alvaro Hacar

- Assistant Professor at the Institute for Astrophysics, University of Vienna
- PI of the ERC Starting Grant EMERGE project

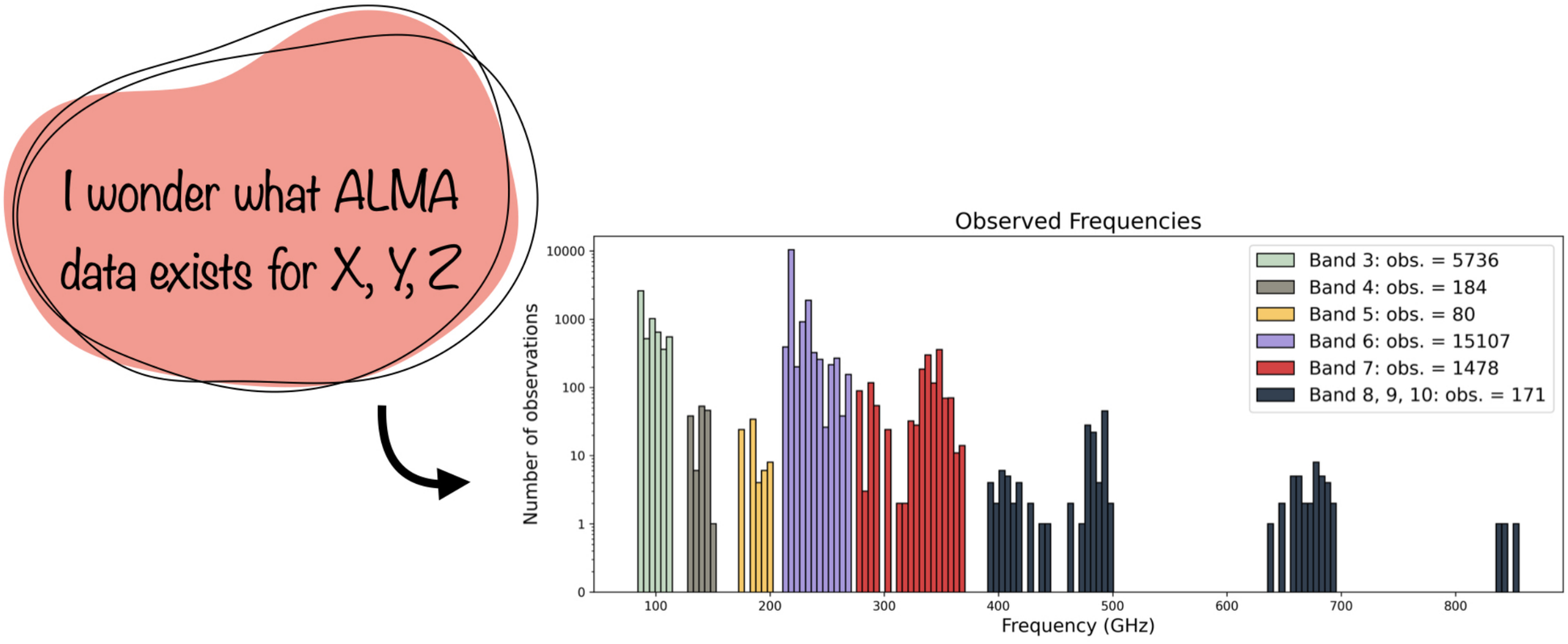
universität wien

European Research Council
Established by the European Commission

ALMA
EUROPEAN ARC
ALMA Regional Centre || Allegro

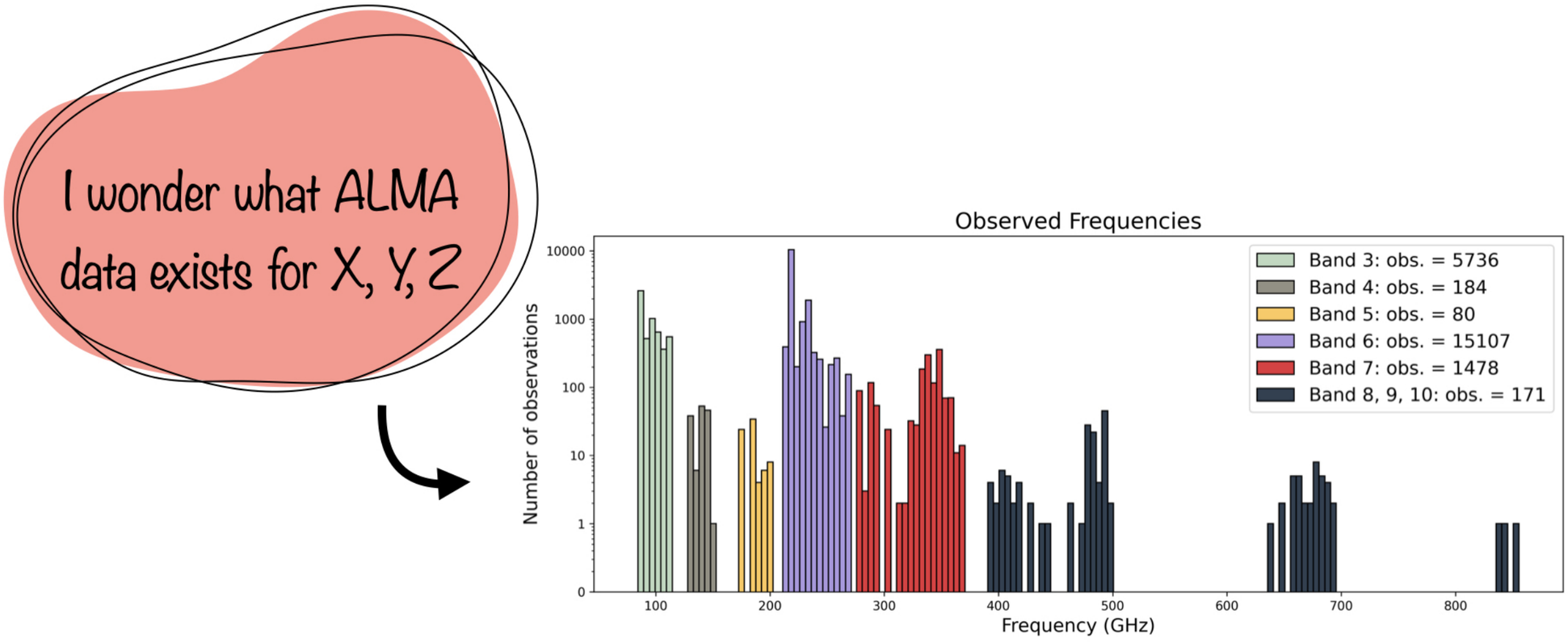
What is ALminer

Python-based code to effectively query, analyse, and visualise the ALMA science archive (ASA).



What is ALminer

Python-based code to effectively **query**, **analyse**, and **visualise** the ALMA science archive (ASA).



Bonus: **download** ALMA data products and/or raw data for further image processing.

Where to find ALminer

- **Documentation:** https://alminer.readthedocs.io/
- **GitHub:** https://github.com/emerge-erc/ALminer

Installation

It's as simple as

```
> pip install alminer
```

Installation

It's as simple as

```
> pip install alminer
```

Dependencies

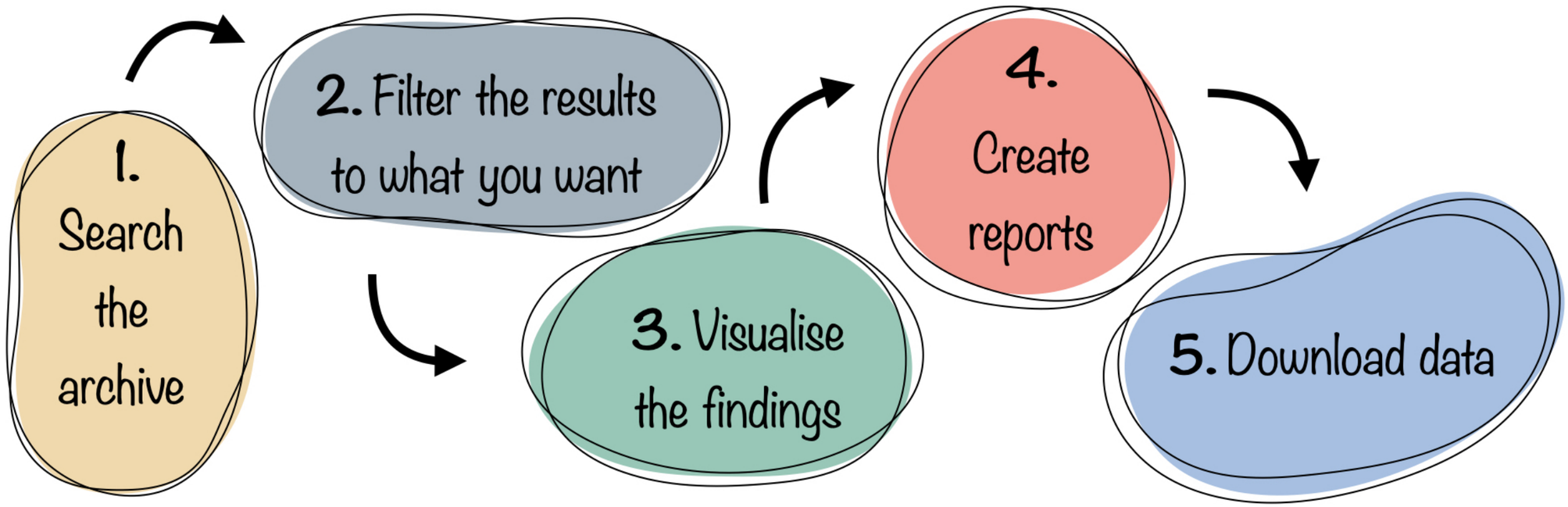
- ALminer works in Python>=3.6
- We depend on `numpy`, `matplotlib`, `pandas`, `pyvo`, & `astropy`
- For the data download, we rely on `astroquery`

Let's get started...

- This presentation is a live Jupyter Notebook so you can follow along with me!
- For the **live presentation**, go to https://bit.ly/ALminer_I-TRAIN to launch the Jupyter Notebook
- Each cell can be executed by pressing `Shift`+`Enter`

- For the **static version**, follow along at https://bit.ly/ALminer_I-TRAIN_static

Tutorial outline



We first need to import all the relevant packages

We first need to import all the relevant packages

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

We first need to import all the relevant packages

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

If you were successful in running the notebook so far, we are ready to begin!

1. Query tools

Three different methods for querying the ALMA archive:

- 1.1 - Target name
- 1.2 - Target position
- 1.3 - ALMA keywords

1. Query tools

Three different methods for querying the ALMA archive:

Common features:

- The queries return all possible observations in PANDAS DataFrame format
- Default mode searches only for **public data**
    - to only search for propietary data -> set *public=False*
    - to search for all data -> set *public=None*
- By default, a summary of the observations, including a list of **target names** is printed
    - to turn this off for large queries -> set *print_targets=False*

Function: `target`

1.1 Query by target name

The target name must be resolved in one of SIMBAD, NED, or VizieR.
The query will:

1. retrieve the coordinates from one of the databases
2. search for ALMA data within a radius of 1'
    - default search radius can be changed using the *search_radius* parameter in arcmin

Example 1.1.1: query two sources by name

Example 1.1.1: query two sources by name

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

```
================================
alminer.target results
================================
Target = Orion KL
--------------------------------
Number of projects = 32
Number of observations = 86
Number of unique subbands = 269
Total number of subbands = 382
36 target(s) with ALMA data = ['Orion KL', 'Orion H2O maser outburst', 'OrionKL', 'orion-IRc2', 'f3',
'f1', 'f11', 'f13', 'f10', 'f14', 'f15', 'f12', 'Orion_Source_I', 'OMC1_SE', 'orion_kl', 'BN', 'OMC1_N
W', 'BN-KL', 'Orion_KL', 'f23', 'OrionKL-SV', 'ONC', 'Orion_BNKL_source_I', 'Orion', 'OMC-1_Region5',
'OMC-1_Region2', 'OMC-1_Region4', '104', 'HC602_HC606_HC608', 'Orion_KL_Field_2_SMA1', 'Orion_KL_Field_
1_Orion_Hot_Core', 'Orion_KL_Field_3_North-west_Clump', 'ONC_Mosaic', 'f16', 'Orion-KL', 'ORS-8']
--------------------------------
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']
--------------------------------
```

Now let's increase the search radius:

Now let's increase the search radius:

In [3]:
```
myquery = alminer.target(['Orion KL', "AB Aur"], search_radius=5.0)
```

```
================================
alminer.target results
================================
Target = Orion KL
--------------------------------
Number of projects = 41
Number of observations = 166
Number of unique subbands = 348
Total number of subbands = 780
81 target(s) with ALMA data = ['Orion KL', 'Orion H2O maser outburst', 'OrionField2', 'OrionField1-1',
'OrionField1-2', 'OrionField3', 'OrionKL', 'Orion_Bar', 'orion-IRc2', 'f5', 'f4', 'f3', 'f7', 'f1', 'f
8', 'f11', 'f13', 'f10', 'f9', 'f14', 'f15', 'f12', 'Orion_Source_I', 'OMC1_SE', 'orion_kl', 'BN', 'OMC
1_NW', 'BN-KL', 'Orion_KL', 'OMC-1S', 'f23', 'f17', 'f16', 'f21', 'f20', 'f22', 'f19', 'f18', 'OrionKL-
SV', 'OMC-1', 'Orion_Bar_PDR', 'OrionBullets', 'ONC', 'Orion_BNKL_source_I', 'd203-506', 'Orion', 'OMC-
1_Region5', 'OMC-1_Region1', 'OMC-1_Region2', 'OMC-1_Region3', 'OMC-1_Region4', '71', '115', '111', '11
0', '95', '19', '32', '55', '29', '20', '107', '63', '104', '101', '61', '141-1952_136-1955', 'HC602_HC
606_HC608', 'HC672', 'GEMS28', 'Orion_KL_Field_2_SMA1', 'Orion_KL_Field_1_Orion_Hot_Core', 'Orion_KL_Fi
eld_3_North-west_Clump', 'ONC_Mosaic', 'Orion1', 'Orion-KL', 'ORS-5', 'ORS-7', 'ORS-9', 'ORS-8', 'ORS-
4']
--------------------------------
Target = AB Aur
--------------------------------
Number of projects = 7
Number of observations = 8
Number of unique subbands = 38
Total number of subbands = 38
7 target(s) with ALMA data = ['2MASS_J04555605+3036209', 'ab_aurigae', 'SU_Aur', 'J04554801+3028050',
'AB_Aur', '2MASS_J04554801+3028050', 'AB_Auriga']
--------------------------------
```

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: the column that is used is the *Name* column and the coordinates provided by the user are ignored.

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: the column that is used is the *Name* column and the coordinates provided by the user are ignored.

In [4]:
```
mylist = ascii.read("Sample_cat.dat", header_start=0, data_start=1)
myquery = alminer.target(mylist['Name'])
```

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

Example 1.1.3: include proprietary data

Example 1.1.3: include proprietary data

In [5]:
```
myquery = alminer.target(mylist['Name'], public=None)
```

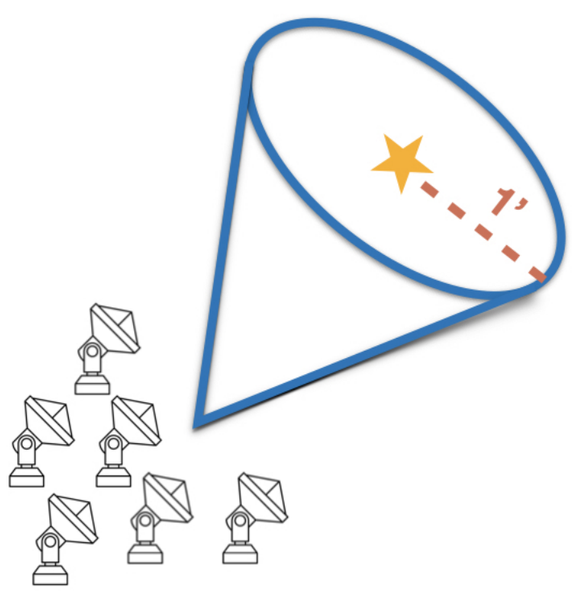
```
================================
alminer.target results
================================
Target = AB_Aur
--------------------------------
Number of projects = 3
Number of observations = 5
Number of unique subbands = 25
Total number of subbands = 25
3 target(s) with ALMA data = ['ab_aurigae', 'AB_Aur', 'AB_Auriga']
--------------------------------
Target = AK_Sco
--------------------------------
Number of projects = 5
Number of observations = 5
Number of unique subbands = 17
Total number of subbands = 20
2 target(s) with ALMA data = ['HIP_82747', 'AK_Sco']
--------------------------------
Target = AS_310
--------------------------------
No observations found.
--------------------------------
Target = AS_470
--------------------------------
No observations found.
--------------------------------
Target = AS_477
--------------------------------
No observations found.
--------------------------------
```

1.2 Query by position

Functions: `conesearch` `catalog`

Query the archive by **positions in the sky** and a **search radius** around them:

- right ascension (*ra*) and declinations (*dec*) must be given in units of **degrees** (ICRS)
    - use the Astropy coordinates package to convert your desired coordinates to degrees
- default search radius is 1 arcmin, but can be changed using the *search_radius* parameter



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

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

In [6]:
```
myquery = alminer.conesearch(ra=201.365063, dec=-43.019112, search_radius=10.0)
```

```
--------------------------------
Number of projects = 23
Number of observations = 75
Number of unique subbands = 190
Total number of subbands = 305
9 target(s) with ALMA data = ['Centaurus A', 'CenA', 'J1325-430', 'Centaurus_A', 'J1325-4301', 'Centaur
us_a', '3FGL_J1325.4-4301', 'Cen_A', 'NGC_5128']
--------------------------------
```

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

Let's first import a few rows of a catalog, e.g. the catalog of Spitzer YSOs in Orion from Megeath et al. (2009) (see the file here):

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

Let's first import a few rows of a catalog, e.g. the catalog of Spitzer YSOs in Orion from Megeath et al. (2009) (see the file here):

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

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

Let's first import a few rows of a catalog, e.g. the catalog of Spitzer YSOs in Orion from Megeath et al. (2009) (see the file here):

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

and create a PANDAS DataFrame with the columns **Name**, **RAJ2000**, **DEJ2000**:

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

Let's first import a few rows of a catalog, e.g. the catalog of Spitzer YSOs in Orion from Megeath et al. (2009) (see the file here):

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

and create a PANDAS DataFrame with the columns Name, RAJ2000, DEJ2000:

In [8]:
```
mycat = pandas.DataFrame({"Name": Spitzer['Seq'],
                          "RAJ2000": Spitzer["RA2000"],
                          "DEJ2000": Spitzer["DEC2000"]})
```

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

Let's first import a few rows of a catalog, e.g. the catalog of Spitzer YSOs in Orion from Megeath et al. (2009) (see the file here):

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

and create a PANDAS DataFrame with the columns Name, RAJ2000, DEJ2000:

In [8]:
```
mycat = pandas.DataFrame({"Name": Spitzer['Seq'],
                          "RAJ2000": Spitzer["RA2000"],
                          "DEJ2000": Spitzer["DEC2000"]})
```

Run the query:

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

Let's first import a few rows of a catalog, e.g. the catalog of Spitzer YSOs in Orion from Megeath et al. (2009) (see the file here):

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

and create a PANDAS DataFrame with the columns Name, RAJ2000, DEJ2000:

In [8]:
```
mycat = pandas.DataFrame({"Name": Spitzer['Seq'],
                          "RAJ2000": Spitzer["RA2000"],
                          "DEJ2000": Spitzer["DEC2000"]})
```

Run the query:

In [9]:
```
myquery = alminer.catalog(mycat, search_radius=1.0)
```

```
================================
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
```

Function: `keysearch`

1.3 Query by ALMA keywords

Query the ALMA archive for **any (string-type) keywords** defined in ALMA TAP system (see columns in the ASA and this table here).

Very powerful tool for querying topics of interest, especially when keywords are combined!

General rules:

- spaces are interpreted with 'AND' logic

```
almaner.keysearch({"proposal_abstract": ["high-mass star formation outflow disk"]})
                                         AND  AND    AND     AND
```

- when multiple values are provided for a given keyword, they are queried using 'OR' logic

```
almaner.keysearch({"proposal_abstract": ["high-mass", "star", "formation", "outflow", "disk"]})
                                          OR      OR        OR         OR
```

- when multiple keywords are provided, they are queried using 'AND' logic

```
almaner.keysearch({"proposal_abstract": ["star formation"], "scientific_category":['Galaxy evolution']})
                                         AND                AND
```

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

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

In [10]:
```
myquery = alminer.keysearch({'target_name': ['GRB021004','SPT0319-47', 'G345']})
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 16
Number of observations = 28
Number of unique subbands = 121
Total number of subbands = 179
10 target(s) with ALMA data = ['GRB021004', 'G345.5', 'SPT0319-47', 'G345.5043+00.3480', 'G345.49+1.4
7', 'G345.50+0.35', 'G345.6487+0.0089', 'G345.01', 'G345.11', 'G345.0029-0.2241']
--------------------------------
```

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

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

In [11]:
```
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 = ['KMOS3DGS4-11016', 'KMOS3DCOS4-13174', 'KMOS3DU4-32147', 'KMOS3DGS4-2411
0', 'KMOS3DCOS4-19680', 'KMOS3DU4-22227', 'KMOS3DU4-20547', 'KMOS3DCOS4-15813', 'KMOS3DCOS4-13701', 'KM
OS3DGS4-25151', 'KMOS3DU4-34138', 'KMOS3DGS4-27882', 'KMOS3DCOS4-10347', 'KMOS3DCOS4-24763', 'KMOS3DCOS
4-15820', 'AK_Sco']
--------------------------------
```

Example 1.3.3: query by words in the proposal abstract

Let's do a big query, but avoid printing the long list of target names in the summary by setting the *print_targets* parameter to *False*:

Example 1.3.3: query by words in the proposal abstract

Let's do a big query, but avoid printing the long list of target names in the summary by setting the *print_targets* parameter to *False*:

In [12]:
```
myquery = alminer.keysearch({'proposal_abstract': ['high-mass star formation outflow']}, print_targets=Fals
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 35
Number of observations = 459
Number of unique subbands = 443
Total number of subbands = 3039
Total number of targets with ALMA data = 259
--------------------------------
```

Example 1.3.4: query by combination of keywords

Query the ALMA archive for proposals that have the words ('high-mass' AND 'star' AND 'formation' AND 'outflow') *OR* ('massive' AND 'star' AND 'formation') corresponding to the scientific category of 'Galaxy evolution'.

Example 1.3.4: query by combination of keywords

Query the ALMA archive for proposals that have the words ('high-mass' AND 'star' AND 'formation' AND 'outflow') *OR* ('massive' AND 'star' AND 'formation') corresponding to the scientific category of 'Galaxy evolution'.

In [13]:
```
myquery = alminer.keysearch({'proposal_abstract': ['high-mass star formation', 'massive star formation'],
                             'scientific_category':['Galaxy evolution']}, print_targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 119
Number of observations = 2056
Number of unique subbands = 2481
Total number of subbands = 8247
Total number of targets with ALMA data = 1376
--------------------------------
```

Example 1.3.5: query for full polarization data

Example 1.3.5: query for full polarization data

In [14]:
```
myquery = alminer.keysearch({'science_keyword':['disks around low-mass stars'],
                             'pol_states':['XY', 'YX']}, print_targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 27
Number of observations = 50
Number of unique subbands = 94
Total number of subbands = 200
Total number of targets with ALMA data = 36
--------------------------------
```

2. Filter & explore results

The querying functions 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 the DataFrame
- 2.2 - Filter & summarise results
- 2.3 - Line coverage
- 2.4 - CO, 13CO, C18O lines

Let's create a query that we can use for the rest of the tutorial

Let's create a query that we can use for the rest of the tutorial

In [15]:
```
observations = alminer.keysearch({'science_keyword':['Galaxy chemistry']}, print_targets=False)
```

```
================================
alminer.keysearch results
================================
--------------------------------
Number of projects = 47
Number of observations = 339
Number of unique subbands = 1162
Total number of subbands = 1361
Total number of targets with ALMA data = 63
--------------------------------
```

Function: `explore`

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*
- display all columns -> *allcols=True/False*

Note: 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)

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

In [16]:
```
alminer.explore(observations)
```

Out[16]:

	Obs	project_code	ALMA_source_name	RAJ2000	DEJ2000	ang_res_arcsec	min_freq_GHz	max_freq_GHz	central_freq_GHz	bandwidth_G
0	1	2011.0.00268.S	LESS J0332-2756	53.122042	-27.938694	1.167	252.85	254.73	253.79	1.875
1	2	2011.0.00268.S	LESS J0332-2756	53.122042	-27.938694	1.167	239.50	241.37	240.43	1.875
2	3	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.442	290.68	292.56	291.62	1.880
3	4	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	93.07	94.95	94.01	1.878
4	5	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	105.07	106.95	106.01	1.878
5	6	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	91.20	93.08	92.14	1.878
6	7	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	103.20	105.08	104.14	1.878
7	8	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.455	283.51	285.39	284.45	1.874
8	9	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.455	281.64	283.51	282.57	1.874
9	10	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.563	259.58	261.47	260.52	1.894
10	11	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.563	256.63	258.53	257.58	1.894
11	12	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.563	242.31	244.20	243.25	1.894
12	13	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.563	244.13	246.02	245.08	1.893
...	...	...	...	...	...	...	...	...	...	...
1348	1349	2019.1.00130.S	NGC1068	40.669879	-0.013289	2.247	96.60	98.47	97.53	1.873
1349	1350	2019.1.00130.S	NGC4945	196.363663	-49.467900	2.309	96.80	98.67	97.73	1.873
1350	1351	2019.1.00130.S	NGC4945	196.363663	-49.467900	2.309	108.87	110.74	109.80	1.873
1351	1352	2019.1.00130.S	NGC4945	196.363663	-49.467900	2.309	98.49	100.36	99.42	1.873
1352	1353	2019.1.00130.S	NGC4945	196.363663	-49.467900	2.309	110.76	112.63	111.69	1.873
1353	1354	2019.1.00130.S	NGC4418	186.727617	-0.877642	2.537	89.43	91.30	90.36	1.873
1354	1355	2019.1.00130.S	NGC4418	186.727617	-0.877642	2.537	101.24	103.11	102.18	1.873
1355	1356	2019.1.00130.S	NGC4418	186.727617	-0.877642	2.537	91.20	93.07	92.13	1.873
1356	1357	2019.1.00130.S	NGC4418	186.727617	-0.877642	2.537	103.05	104.92	103.99	1.873
1357	1358	2019.1.00130.S	NGC4945	196.363663	-49.467900	2.523	103.59	105.46	104.52	1.873
1358	1359	2019.1.00130.S	NGC4945	196.363663	-49.467900	2.523	101.78	103.65	102.72	1.873
1359	1360	2019.1.00130.S	NGC4945	196.363663	-49.467900	2.523	89.90	91.77	90.84	1.873
1360	1361	2019.1.00130.S	NGC4418	186.727617	-0.877642	1.357	97.97	99.84	98.90	1.873

1361 rows × 18 columns

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

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

In [17]:
```
alminer.explore(observations, allcols=True, allrows=False)
```

Out[17]:

	Obs	project_code	ALMA_source_name	RAJ2000	DEJ2000	ang_res_arcsec	min_freq_GHz	max_freq_GHz	central_freq_GHz	bandwidth_G
0	1	2011.0.00268.S	LESS J0332-2756	53.122042	-27.938694	1.167	252.85	254.73	253.79	1.875
1	2	2011.0.00268.S	LESS J0332-2756	53.122042	-27.938694	1.167	239.50	241.37	240.43	1.875
2	3	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.442	290.68	292.56	291.62	1.880
3	4	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	93.07	94.95	94.01	1.878
4	5	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	105.07	106.95	106.01	1.878
5	6	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	91.20	93.08	92.14	1.878
6	7	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	103.20	105.08	104.14	1.878
7	8	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.455	283.51	285.39	284.45	1.874

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

In [17]:
```
alminer.explore(observations, allcols=True, allrows=False)
```

Out[17]:

	Obs	project_code	ALMA_source_name	RAJ2000	DEJ2000	ang_res_arcsec	min_freq_GHz	max_freq_GHz	central_freq_GHz	bandwidth_G
0	1	2011.0.00268.S	LESS J0332-2756	53.122042	-27.938694	1.167	252.85	254.73	253.79	1.875
1	2	2011.0.00268.S	LESS J0332-2756	53.122042	-27.938694	1.167	239.50	241.37	240.43	1.875
2	3	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.442	290.68	292.56	291.62	1.880
3	4	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	93.07	94.95	94.01	1.878
4	5	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	105.07	106.95	106.01	1.878
5	6	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	91.20	93.08	92.14	1.878
6	7	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	1.331	103.20	105.08	104.14	1.878
7	8	2011.0.00405.S	PKS1830-211	278.416330	-21.061080	0.455	283.51	285.39	284.45	1.874

Functions: `get_info` `summary`

2.2 Filter & summrise 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
- get the description and units of any column in the DataFrame using the function `alminer.get_info('column_name')`
- filter the query results further
- use `alminer.summary` function to print a summary of the observations after filtering

Functions: `get_info` `summary`

2.2 Filter & summrise 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
- get the description and units of any column in the DataFrame using the function `alminer.get_info('column_name')`
- filter the query results further
- use `alminer.summary` function to print a summary of the observations after filtering

In [18]:
```
observations.columns
```

Out[18]:
```
Index(['Obs', 'project_code', 'ALMA_source_name', 'RAJ2000', 'DEJ2000',
       'ang_res_arcsec', 'min_freq_GHz', 'max_freq_GHz', 'central_freq_GHz',
       'bandwidth_GHz', 'freq_res_kHz', 'vel_res_kms', 'LAS_arcsec',
       'FoV_arcsec', 'cont_sens_bandwidth', 'line_sens_10kms',
       'line_sens_native', 'MOUS_id', 'access_url', 'access_format',
       'proposal_id', 'data_rights', 'gal_longitude', 'gal_latitude',
       'obs_publisher_did', 'obs_collection', 'facility_name',
       'instrument_name', 'obs_id', 'dataproduct_type', 'calib_level',
       'target_name', 's_ra', 's_dec', 's_fov', 's_region', 's_resolution',
       't_min', 't_max', 't_exptime', 't_resolution', 'em_min', 'em_max',
       'em_res_power', 'pol_states', 'o_ucd', 'band_list', 'em_resolution',
       'authors', 'pub_abstract', 'publication_year', 'proposal_abstract',
       'schedblock_name', 'proposal_authors', 'sensitivity_10kms',
```

Now let's check what the description and units of *ang_res_arcsec* column are:

Now let's check what the description and units of *ang_res_arcsec* column are:

In [19]:
```
alminer.get_info('ang_res_arcsec')
```

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

Example 2.2.1: simple selection - observations with angular resolutions < 0.5”

Example 2.2.1: simple selection - observations with angular resolutions < 0.5”

In [20]:
```
selected = observations[observations['ang_res_arcsec'] < 0.5]
```

Example 2.2.1: simple selection - observations with angular resolutions < 0.5”

In [20]:
```
selected = observations[observations['ang_res_arcsec'] < 0.5]
```

and print the summary:

Example 2.2.1: simple selection - observations with angular resolutions < 0.5”

In [20]:
```
selected = observations[observations['ang_res_arcsec'] < 0.5]
```

and print the summary:

In [21]:
```
almine.summary(selected, print_targets=False)
```

```
--------------------------------
Number of projects = 23
Number of observations = 121
Number of unique subbands = 367
Total number of subbands = 484
Total number of targets with ALMA data = 41
--------------------------------
```

Example 2.2.2: multiple selections - observations with angular resolution < 0.5” & velocity resolution < 1 km/s

Example 2.2.2: multiple selections - observations with angular resolution < 0.5” & velocity resolution < 1 km/s

In [22]:
```
selected = observations[(observations['ang_res_arcsec'] < 0.5) &
                        (observations['vel_res_kms'] < 1.0)]
almaiminer.summary(selected, print_targets=False)
```

```
--------------------------------
Number of projects = 9
Number of observations = 48
Number of unique subbands = 65
Total number of subbands = 109
Total number of targets with ALMA data = 26
--------------------------------
```

Example 2.2.3: observations containing a given frequency

Example 2.2.3: observations containing a given frequency

In [23]:
```
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
--------------------------------
```

Function: `line_coverage`

2.3 Line coverage

The `alminer.line_coverage` function 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.

Important to note:

- line frequencies should be given in GHz -> *line_freq* parameter
- redshift is by default assumed to be 0 -> *z* parameter
- the *line_name* parameter is the user's defined name for the frquency provided

Example 2.3.1: search whether a given frequency is covered in the observations

Example 2.3.1: search whether a given frequency is covered in the observations

In [24]:
```
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 2.3.2: search whether a given frequency is observed for a target at a given redshift

Example 2.3.2: search whether a given frequency is observed for a target at a given redshift

In [25]:
```
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']
--------------------------------
```

Function: `CO_lines`

2.4 Coverage of CO, ^{13}CO, and C^{18}O lines

The `alminer.CO_lines` function determines **how many CO, ^{13}CO, and C^{18}O lines were observed** in the provided DataFrame and returns a DataFrame containing all observations of these transitions.

Example 2.4.1: search whether any CO, ^{13}CO, and C^{18}O lines were observed in the query results

Example 2.4.1: search whether any CO, ^{13}CO, and C^{18}O lines were observed in the query results

In [26]:
```
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
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
```

Example 2.4.2: search whether any redshifted CO, ^{13}CO, and C^{18}O lines were observed in the query results

Example 2.4.2: search whether any redshifted CO, ^{13}CO, and C^{18}O lines were observed in the query results

In [27]:
```
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)
--------------------------------
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
```

3. Plot results

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

Common features:

Most of the plotting functions have the option to:

- mark frequencies of CO, ^{13}CO, and C^{18}O lines by toggling *mark_CO=True*
- mark a list of frequencies -> *mark_freq* parameter
- provide a redshift by which by the marked frequencies can be shifted -> *z* parameter
- save the plot in PDF format -> set the *savefig* parameter to the desired filename
- avoid displaying the plot (for example to create and save plots in a loop) -> set *showfig=False*

Function: `plot_overview`

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 & velocity resolutions**.

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

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

In [28]:
```
alminer.plot_overview(observations, savefig='alma_galaxy_chemistry')
```

Function: `plot_line_overview`

3.2 Plot an overview of a given line in the observations

The `alminer.plot_line_overview` creates the same plot as the `alminer.plot_line_overview` function but it **highlights observations of a give frequency** specified by the user (redshifted if the *z* parameter is set).

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

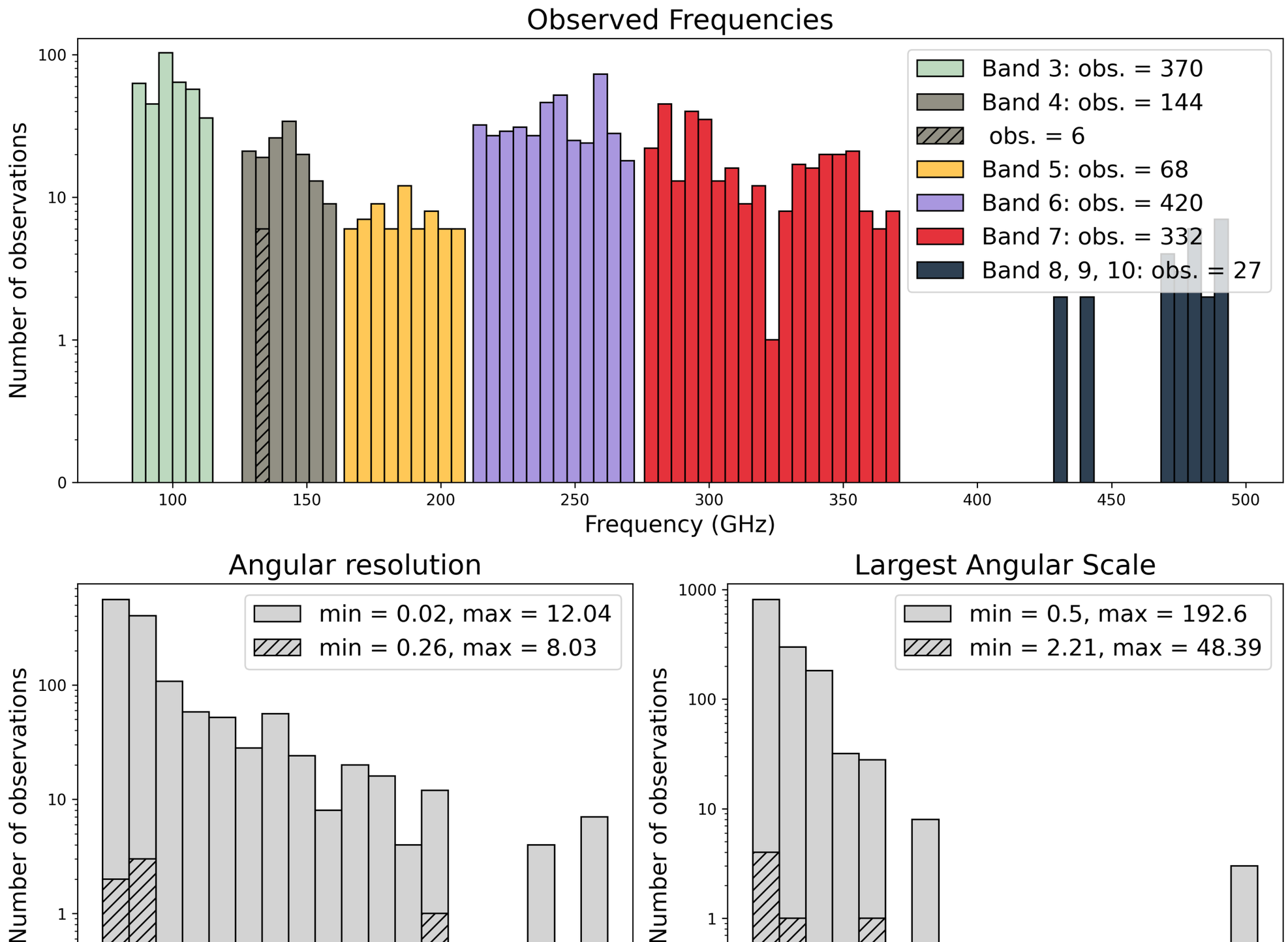
Example 3.2.1: plot an overview of the observations and highlight observations at a particular frequency

In [29]:
```
alminer.plot_line_overview(observations, line_freq=100.0)
```

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

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

In [30]:
```
alminer.plot_line_overview(observations, line_freq=400.0, z=2)
```



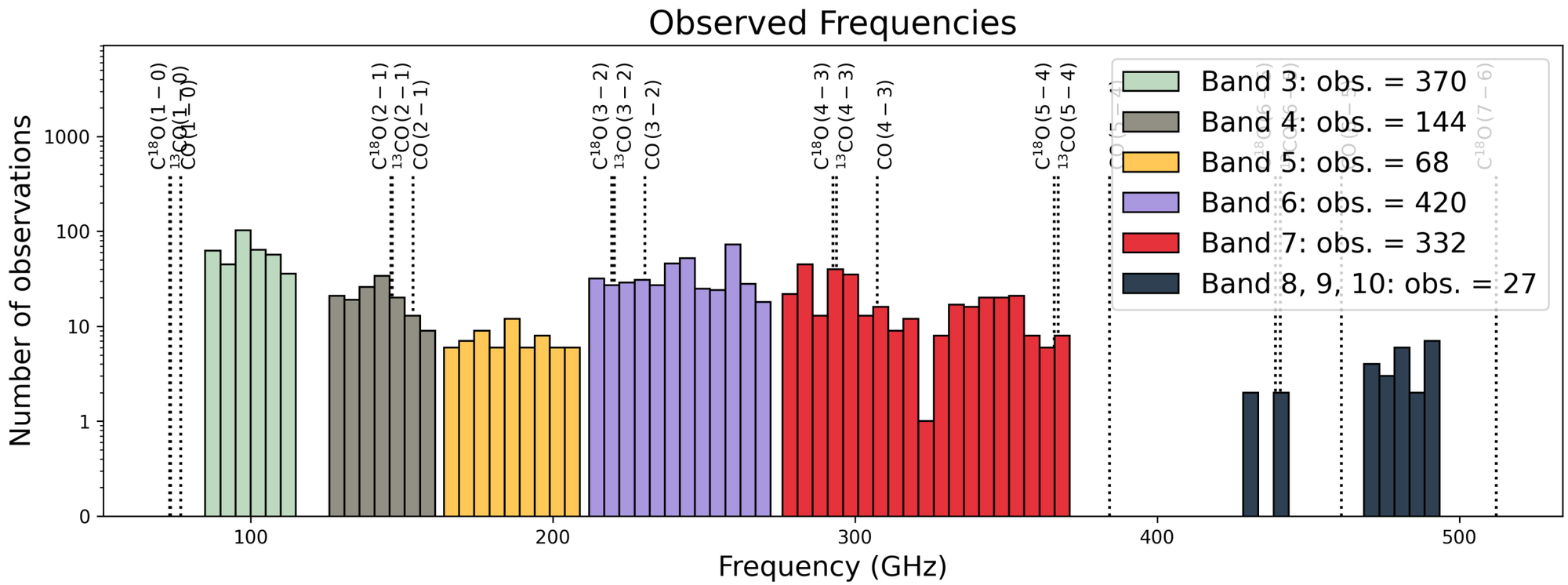
3.3 Plot observed frequencies in each band

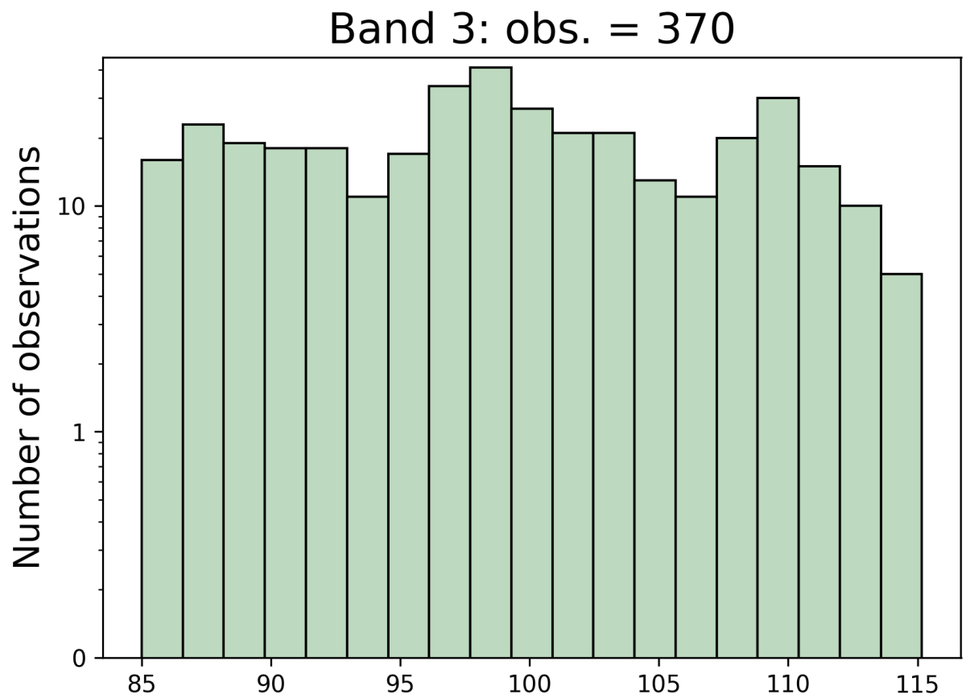
The `alminer.plot_bands` function creates detailed plots of **observed frequencies in each band**.

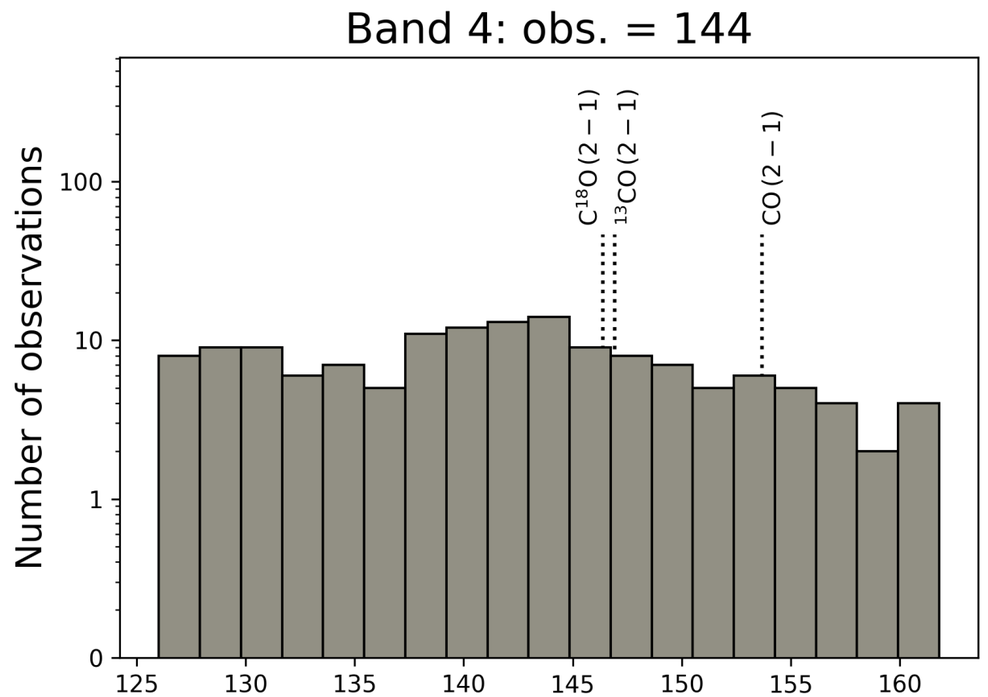
Example 3.3.1: plot observed frequencies in each band and mark redshifted CO lines

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

In [31]:
```
alminer.plot_bands(observations, mark_CO=True, z=0.5)
```







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

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

In [32]:
```
alminer.plot_bands(observations, mark_freq=[106.5, 245.0, 366.3])
```

Function: `plot_observations`

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**.

Note: Observation numbers are the input DataFrame's index values.

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

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

In [33]:
```
alminer.plot_observations(observations, mark_CO=True)
```

Example 3.4.2: plot observed frequencies and mark frequencies of choice

Example 3.4.2: plot observed frequencies and mark frequencies of choice

In [34]:
```
alminer.plot_observations(observations, mark_freq=[106.5, 245.0, 366.3])
```

Function: `plot_sky`

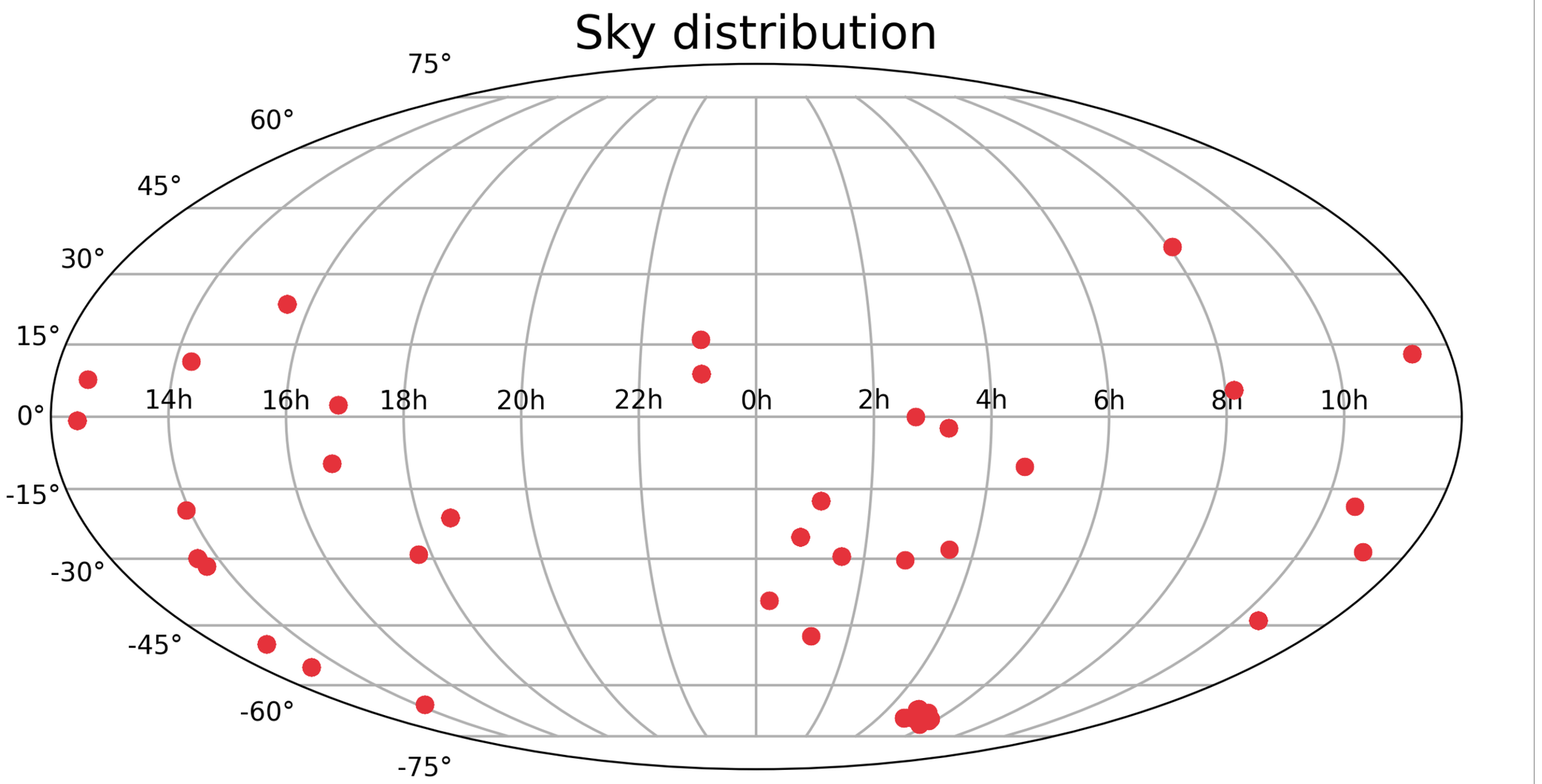
3.5 Plot sky distribution

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

Example 3.5.1: plot sky distribution

Example 3.5.1: plot sky distribution

In [35]:
```
alminer.plot_sky(observations)
```



4. Create reports

This section introduces different ways to save query results:

- 4.1 - Export results as a table
- 4.2 - Save overview plots for each target

Function: `save_table`

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.

Note: If the 'tables' folder does not exist, it will be created.

Example 4.1.1: save query results as a table

Example 4.1.1: save query results as a table

In [36]:
```
alminer.save_table(observations, filename="galaxy_chemistry")
```

Function: `save_source_reports`

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.

Notes:

- If the 'reports' folder does not exist, it will be created.
- The reports are named after the target names.
- Currently, the grouping is done based on ALMA target names, so the same source with a slightly different naming schemes will be treated as separate targets.

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

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":

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":

In [37]:
```
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 = 8
Number of observations = 21
Number of unique subbands = 81
Total number of subbands = 81
8 target(s) with ALMA data = ['NGC7469', 'NGC1266', 'Arp220', 'ngc6240', 'IRAS_F16399-0937', 'n613', 'N
GC4418', 'Cloverleaf']
--------------------------------
```

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":

In [37]:
```
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 = 8
Number of observations = 21
Number of unique subbands = 81
Total number of subbands = 81
8 target(s) with ALMA data = ['NGC7469', 'NGC1266', 'Arp220', 'ngc6240', 'IRAS_F16399-0937', 'n613', 'N
GC4418', 'Cloverleaf']
--------------------------------
```

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

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":

In [37]:
```
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 = 8
Number of observations = 21
Number of unique subbands = 81
Total number of subbands = 81
8 target(s) with ALMA data = ['NGC7469', 'NGC1266', 'Arp220', 'ngc6240', 'IRAS_F16399-0937', 'n613', 'N
GC4418', 'Cloverleaf']
--------------------------------
```

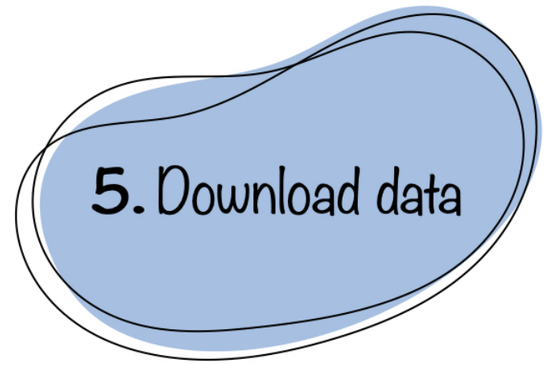
Now we can create and save plots for each source, with CO and its isotopologues marked:

In [38]:
```
alminer.save_source_reports(selected, mark_CO=True)
```

Function: `download_data`

5. Download data

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



General notes about the download function:

- The desired location can be changed -> set the *location* parameter to the desired path
    - default location: 'data' subdirectory in the current working directory
- To check the amount of disk space needed first before downloading -> set *dryrun=True*
- By default, tar files (*raw* + *FITS*) associated with uids in the provided DataFrame will be downloaded
    - To download only the FITS data products -> set *fitsonly=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
    - 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 (e.g. primary beam corrected: '.pbcor', the science target: '_sci' or the ALMA target name)
- A list of URLs (files) to be downloaded from the archive can be printed to the terminal by setting *print_urls=True*

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

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

In [40]:
```
almaner.download_data(selected, fitsonly=False, dryrun=True, location='./data', print_urls=False)
```

```
================================
This is a dryrun. To begin download, set dryrun=False.
================================
Download location = ./data
Total number of Member OUSs to download = 21
Selected Member OUSs: ['uid://A002/X6444ba/X17d', 'uid://A001/X121/X3d2', 'uid://A001/X121/X3d8', 'ui
d://A001/X121/X3d5', 'uid://A001/X121/X3cf', 'uid://A001/X2fe/Xcce', 'uid://A001/X2fe/Xcd2', 'uid://A00
1/X2fe/Xcde', 'uid://A001/X2fe/Xcd6', 'uid://A001/X2fe/Xcda', 'uid://A001/X2fe/X728', 'uid://A001/X2fe/
X724', 'uid://A001/X2fe/X734', 'uid://A001/X2fe/X738', 'uid://A001/X5a4/X155', 'uid://A001/X87d/X1fb',
'uid://A001/X87d/X1f5', 'uid://A001/X87d/X207', 'uid://A001/X1290/X17', 'uid://A001/X1288/X6c2', 'ui
d://A001/X1288/X6be']
Number of files to download = 115
Needed disk space = 4.1 TB
--------------------------------
```

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

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

In [41]:
```
almaner.download_data(selected, fitsonly=True, dryrun=True, location='./data',
                      filename_must_include=['_sci', '.pbcor', 'cont'], print_urls=True)
```

```
================================
This is a dryrun. To begin download, set dryrun=False.
================================
Download location = ./data
Total number of Member OUSs to download = 21
Selected Member OUSs: ['uid://A002/X6444ba/X17d', 'uid://A001/X121/X3d2', 'uid://A001/X121/X3d8', 'ui
d://A001/X121/X3d5', 'uid://A001/X121/X3cf', 'uid://A001/X2fe/Xcce', 'uid://A001/X2fe/Xcd2', 'uid://A00
1/X2fe/Xcde', 'uid://A001/X2fe/Xcd6', 'uid://A001/X2fe/Xcda', 'uid://A001/X2fe/X728', 'uid://A001/X2fe/
X724', 'uid://A001/X2fe/X734', 'uid://A001/X2fe/X738', 'uid://A001/X5a4/X155', 'uid://A001/X87d/X1fb',
'uid://A001/X87d/X1f5', 'uid://A001/X87d/X207', 'uid://A001/X1290/X17', 'uid://A001/X1288/X6c2', 'ui
d://A001/X1288/X6be']
Number of files to download = 19
Needed disk space = 303.7 MB
File URLs to download = https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_Xcda.Arp220_sci.s
pw25_27_29_31.cont.I.tt0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_Xcda.Arp220_sci.spw25_27_29_31.cont.I.tt
1.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_X728.ngc6240_sci.spw19_25_27_29.cont.I.tt
0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_X728.ngc6240_sci.spw19_25_27_29.cont.I.tt
1.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_X724.ngc6240_sci.spw25_27_29_31.cont.I.tt
0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_X724.ngc6240_sci.spw25_27_29_31.cont.I.tt
1.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_X738.IRAS_F16399-0937_sci.spw25_27_29_31.
cont.I.tt0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X2fe_X738.IRAS_F16399-0937_sci.spw25_27_29_31.
cont.I.tt1.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X5a4_X155.n613_sci.spw25_27_29_31_33.cont.I.tt
0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X5a4_X155.n613_sci.spw25_27_29_31_33.cont.I.tt
1.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X87d_X1fb.NGC4418_sci.spw25_27_29_31.cont.I.tt
0.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X87d_X1fb.NGC4418_sci.spw25_27_29_31.cont.I.tt
1.pbcor.fits
https://almascience.eso.org/dataPortal/member.uid___A001_X87d_X1f5.NGC4418_sci.spw25_27_29_31.cont.I.tt
0.pbcor.fits
```

ALminer resources

- **Documentation:** https://alminer.readthedocs.io/ (especially the API)
- **For updates, check out our GitHub page:** https://github.com/emerge-erc/ALminer

Acknowledgements

If you use ALminer as part of your research, please consider citing us (ADS reference will be added to the Github page when available).

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

Contact us!

If you encounter any issues or have questions, you can e-mail us or better yet, use GitHub's issue tracker.

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

Contact us!

If you encounter any issues or have questions, you can e-mail us or better yet, use GitHub's issue tracker.

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

Thanks for using ALminer and happy mining =)