Summer 2022 NSERC Research Report

Student: Jane Cohen, Queen's University, B.A.Sc. Engineering Physics (expected 2024)

Supervisor: Rene Plume, University of Calgary, PhD in Astrophysics

Table of Contents

Project Overview

Our galaxy has been studied for millennia in an attempt to understand what exists beyond planet earth. More specifically, the earliest records show a fascination with the stars that light up the dark sky. And the fixation on these celestial bodies has not faded. As technology improves, our ability to describe the formation and evolution of stars continues to progress. Despite the rapid advances in this field of study, many details regarding the environments in which stars form are still unknown.

Where do we come in?

I'm Jane Cohen, a student from Queen's University studying Engineering Physics and Applied Computing. This past summer I had the pleasure of working with Professor R. Plume. Together, we developed the early steps of a procedure that allows us to easily process data from the James Clerk Maxwell Telescope (JCMT) to help us better understand the environments of star formation.
My side of the project was heavily based on developing the procedure and maximizing the efficiency of the software being used. However, I also gained knowledge in numerous topics such as star formation and molecular astrochemistry. Having completed two years of my undergraduate degree, this information may be below your level of study- but I wanted to include it for the sake of completeness. If any topics are not beneficial for you to review, please feel free to skip the background information in Appendix I: Astrochemistry Basics.

This guide is written using my experience and serves simply as a guide for others continuing the project. This is in no way a formal document, and I have not cited external sources. That being said, please note Rene Plume has made significant contributions to the code used to develop this procedure, especially code for the Gaussian Fits and Radex Column Density.

If you have any questions regarding the work, please reach out to me at janecohen17@gmail.com or ask Rene.

Project Software

The procedure we have developed requires three different software to run from start to finish.

- CASA
- Radex
- Jupyter Lab

CASA

CASA is used in the preliminary steps of processing the data. While we can manually perform functions needed to analyze the data using the CASA, doing so by hand takes a lot of time. Therefore, codes can be used to run CASA functions iteratively when we wish to carry out a specific function for many data points.
Throughout our project, CASA served three main purposes: data manipulation, data fitting and data visualization.

For details on downloading CASA, the CASA terminal, "CASAtasks" and the data visualization tools, visit Appendix II: CASA.

Radex

Radex is an externally developed software that allows us to calculate the column density of a molecule at a specific point. You can run Radex through your computer's terminal. However, we have developed a script that runs Radex in Jupyter Lab so that it can be integrated with the other scripts we've produced.

To download Radex, use the following link and follow the instructions.
https://personal.sron.nl/~vdtak/radex/index.shtml

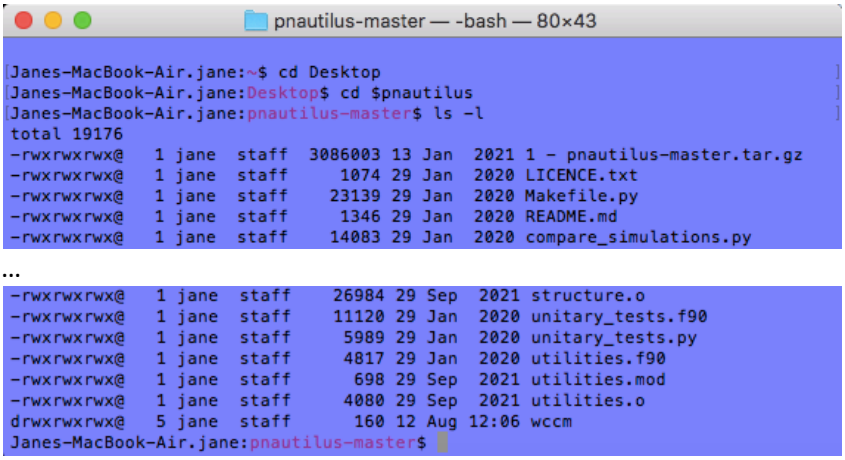
Pnautilus

Similar to Radex, Pnautilus is an open-source software that allows us to carry out complex calculations (beyond the scope of my understanding) to produce data for us to examine. Pnautilus models the evolution of a star and determines the abundance of specific molecules over time. Once again, we've created a script that runs in Jupyter Lab to carry out the functions of Pnautilus.

Setting up Pnautilus

Start by downloading the folder titled "pnautilus-master" in the drive.

You can then navigate to the pnautilus-master directory.



To run pnautilus using your terminal:

```
Janes-MacBook-Air.jane:~$ cd Desktop
Janes-MacBook-Air.jane:Desktop$ cd $pnautilus
Janes-MacBook-Air.jane:pnautilus-master$ cd wccm
Janes-MacBook-Air.jane:wccm$ cd iras
Janes-MacBook-Air.jane:iras$ pnautilus_code     #creates binary code
Janes-MacBook-Air.jane:iras$ pnautilus_outputs    #creates ab file

# ab file columns: year, abundance
```

You can change conditions such as initial gas density, initial gas temperature, initial dust temperature (same as gas) and AV:

```
Janes-MacBook-Air.jane:iras$ more parameters.in
```

For more info on running pnautilus this way, please ask Plume or read the documentation. This project utilizes Jupyter-Lab and does not require running pnautilus using your terminal.

Jupyter Lab

Jupyter Lab is the IDE we use to run all the scripts for this project. It allows us to combine the scripts we've written and work with a flow of variables throughout the entire procedure. It also allows us to only run specific blocks when testing different conditions to minimize the time of computations.

Other IDEs that are suitable may be used instead of Jupyter Lab.

Background Information

Abundant Molecules

In locations of dense star formation, two molecules are abundant: HCN and HCO+. More specifically, the high energy 4-3 transition for these two molecules is readily observed from sites of star formation. This specific transition (energy level 4-3) produces an emission with a unique frequency for each molecule.

Molecule	Level 4-3 emission frequency
HCN	265.9 GHz
HCO+	267.7 GHz

Sources

The JCMT was set to observe specific spots in the Milky Way where star formation is predicted to be. Each source can provide us with data such as temperature, AV, and the abundance and column density of HCN and HCO+. This data allows us to understand the environment around the source and the conditions for star formation and evolution.

Over the summer, we began with two sources that we had pre-existing data for: w40 and s138. In August, we received the first set of data from the JCMT survey: gem_ob1. We've developed scripts that allow us to analyze these data sets with limited manual work. The goal is to be able to apply our code to any source we receive in the future.

While the script automates the majority of the process, parameters for each source must be filled in prior to running the code. Here I've listed these parameters for each source. However, these numbers are not definitive and have been selected by Professor Plume and I based on the intended purpose of our research. They can be changed at your discretion and how to select reasonable values is detailed in Appendix III.

s138

Coordinates (J2000): [22:33:17.225, 22:31:55.810] x [+58.18.25.391, +58.28.55.387]
Coordinates (Galactic): [105°.638049, 105°.499159] x [0°.214705, 0°.345261]

```
vel_guess = -53   # The initial guess for the centroid velocity
peak_guess = 4   # the initial guess for the maximum line strength
dv_min = 1        # set a minimum acceptable line width
dv_max = 10       # set a maximum acceptable line width
snr_min = 3.0     # set a minumum acceptable SNR
min_xpix = 2
max_xpix = 11
min_ypix = 38
max_ypix = 46
```

Note: s138 is also referred to as Sh2-138 or Sharpless 138 in other surveys.

w40

Coordinates (J2000): [18:31:31.146, 18:31:02.327] x [-02.08.31.948, -02.01.19.948]
Coordinates (Galactic): [28°.852079, 28°.694001] x [3°.449430, 3°.607207]

```
vel_guess = 7.5   # The initial guess for the centroid velocity
peak_guess = 8    # the initial guess for the maximum line strength
dv_min = 1        # set a minimum acceptable line width
dv_max = 10       # set a maximum acceptable line width
snr_min = 3.0     # set a minumum acceptable SNR
min_xpix = 29
max_xpix = 56
min_ypix = 15
max_ypix = 50
```

gem_ob1

Coordinates (J2000): [06:08:35.824, 06:09:12.919] x [21.32.03.858, 21.53.52.050]

Coordinates (Galactic): [189°.006911, 188°.758541] x [0°.774844, 1°.076505]

```
vel_guess = 3.5    # The initial guess for the centroid velocity
peak_guess = 10    # the initial guess for the maximum line strength
dv_min = 1         # set a minimum acceptable line width
dv_max = 10        # set a maximum acceptable line width
snr_min = 3.0      # set a minumum acceptable SNR
min_xpix = 27
max_xpix = 45
min_ypix = 58
max_ypix = 80
```

Note: for each source, the coordinates were taken from the HCO+ moment map. If the HCN moment map were used the coordinates could vary by up to 5 decimals, so the difference is negligible.

Scripts

All our codes from the summer have been combined into two files: jc_combine.ipynb and jc_pnautilus.ipynb.

jc_combine.ipynb

This code takes the data cubes of HCO+ and HCN for any source and returns multiple text files, Gaussian fits and column density maps. All this information takes ample time to calculate by hand, so this code serves to extract this information for every pixel in a source automatically.
The only part that changes for each new source is the Global Parameters block. The other blocks of code remain the same.

Inputs

- Spectral data cubes for HCN and HCO+ from the same source
- PPMAP of the temperature for the source
- H2 column density for the source
    - **Note:** ViaLactea PPMAPs give N(H2) of magnitude 20 times too small
        - Use $N(H2) = N(H2)_{map} \times 1e^{20}$
        - W40 H2 map (from Hershel) gives N(H2) so do not adjust
        - S138 and gem_ob1 had to be adjusted

Outputs

- Txt file of the gaussian fit data for each molecule
- PDF of the Gaussian fit curves for each molecule
- Txt file of kinetic temperature for each pixel
- Txt file of all the column densities for each pixel
- PDF of N(HCN)/N(HCO+), N(HCO+) and N(HCN), X(HCO+), X(HCN) and N(H2) plots

Breakdown

[1]: **Python Libraries and Packages**

→ run each time you open Jupyter or the script

Global Parameters

In [2]:

```
##################################################
# Path to directories. Change for each new computer used.

# directory where the data are located and the files are to be written
datadir = '/Users/jane/Desktop/MAJORS/analysis/w40/'

# Path to the radex molecular data files
radexpath = '/Users/jane/Desktop/Radex/data/'

# Extension to the molecular data files
extension = '.dat'

# Path to the radex executable program (how executable command will be called)
radexexec = '/Users/jane/Desktop/Radex/bin/radex'

# How executable command will be called: os.system(radexexec+ '<' +radexpath+'/'+inputFile+ '> /dev/null')
```

↳ change for new source

set up once for your computer

```
##################################################
# Input files. Already exist. Refer to external isntructions for creating files.

# Name of the spectral data cubes on which to perform gaussian fitting
hco_spec_filename='w40_hco+_smooth_gal.fits'   # input fits file name
hcn_spec_filename='w40_hcn_smooth_gal.fits'    # input fits file name

# Name of the input temperature map/FITS file
# If temperature file DNE for source, use 'none' and set constant temp. in "Tkin Data Retrieval" block.
temp_fitsfilename = 'w40_temp_regrid.fits'

# N(H2) file
NH2regrid = 'w40_NH2_regrid.fits'
```

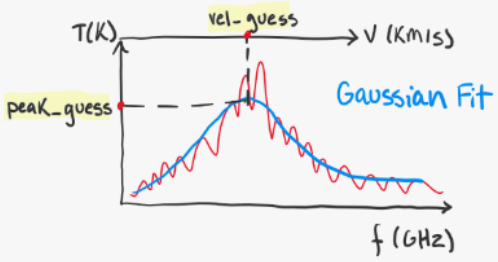
given by project head

must find yourself (see Appendix III)

← you MUST format these files correctly before using them (see Appendix III)

change for each source

```
##################################################
# Parameters for the Gaussian fitting.  CHANGE FOR EACH NEW SOURCE

vel_guess = 7.5   # The initial guess for the centroid velocity of the gaussian fit
peak_guess = 8    # the initial guess for the maximum line strength (sets the yscaling of the plots)
dv_min = 1        # set a minimum acceptable line width
dv_max = 10       # set a maximum acceptable line width
snr_min = 3.0     # set a minumum acceptable SNR

# set the range of pixels in the x direction over which to loop
min_xpix = 29
max_xpix = 56

# set the range of pixels in the y direction over which to loop
min_ypix = 15
max_ypix = 50
```




```
##################################################
# density for the RADEX models
nh2 = 1.0e5      # nH2 cm^-3
##################################################
# Names of various output files. Files will be created or overwritten if they exist.
# YOU ONLY NEED TO CHANGE THESE IF YOU WANT DIFFERENT NAMES THAN THE DEFAULT ONES
```

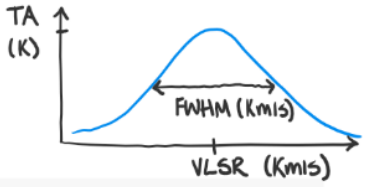
↳ Volume density of H2 - we assume a value (1×10^4, 1×10^5 or 1×10^6)

→ output files don't need to be changed if you have seperate directories for each source

Gaussian fits

In [3]:
```
### CAN BE OMITTED IF GAUSSIAN FITS FILES ARE ALREADY CREATED ###
#####################################################################
```

This whole block of code performs a Gaussian fit for EACH pixel from the HCO+ & HCN data cubes



we use galactic coordinates & pixel numbers to identify data

naming convention is very important to not mix up coordinates from different data sets

Variables:

peak temperature

```
# initialize the arrays to write to an ascii table file at the end
x, y, glat, glong, vlsr, TA, sigma, fwhm, Tmb, figs = [], [], [], [], [], [], [], [], [], []
```

pixel — galactic coordinates — vel. of peak — TA/0.6

Outputs: x, y, glat, glong, vlsr, TA, fwhm, Tmb

Tkin Data Retrieval

In [4]:
```
# Run once for all molecules- same temperature map for HCO+ and HCN per source
```

Outputs: xPixT, yPixT, glatT, glongT, Tk

RADEX Column Density

In [5]:
```
### CAN BE OMITTED IF COLUMN DENSITY FILES ARE ALREADY CREATED ###
#####################################################################
```

```
# The column density will be found using Radex for the specified molecules.

#### HCO+ (molecule A)
filename = hco_gauss_outfile # input fits file name
#outfile = radexhcoinp # file name of the radex file
X = pd.read_csv(datadir+filename, sep="\t", header=None)
df = pd.DataFrame(data=X)

# put pixel x, y, glat, glong, Tmb, FWHM and VLSR in lists and remove headers
xPixA = np.array(df[0].tolist())
xPixA = xPixA[1:]
yPixA = np.array(df[1].tolist())
yPixA = yPixA[1:]
glatA = np.array(df[2].tolist())
glatA = glatA[1:]
glongA = np.array(df[3].tolist())
glongA = glongA[1:]
tmbA = (df[5].tolist())
tmbA = tmbA[1:]
fwhmA = np.array(df[8].tolist())
fwhmA = fwhmA[1:]
vlsrA = np.array(df[6].tolist())
vlsrA = vlsrA[1:]
```

from Gauss. Fits

process: • match xPixA, yPixA to xPixT, yPixT

• use Tmb, fwhm, TK for matching pixels as RADEX inputs

• RADEX calculates HCO+ (A) column density

```
b = np.asarray(tmb[i], dtype=float)
c = np.asarray(fwhm[i], dtype=float)
d = np.asarray(tk[i], dtype=float)

tkin = d        # Tkin (K)
tbg = 2.73      # Tbg (K)
obs = b         # Observed line intensity (K)- Tmb
dv = c          # FWHM line width km/s
bw = 0.01       # Bandwidth (GHz)
tol = 0.01      # tolerance
```

TK → tkin; Tmb → obs; fwhm → dv

outputs: xPixFull, yPixFull, glatFull, glongFull, tKFull, tmbFull, fwhmFull, colDen

Repeats for HCN (B)

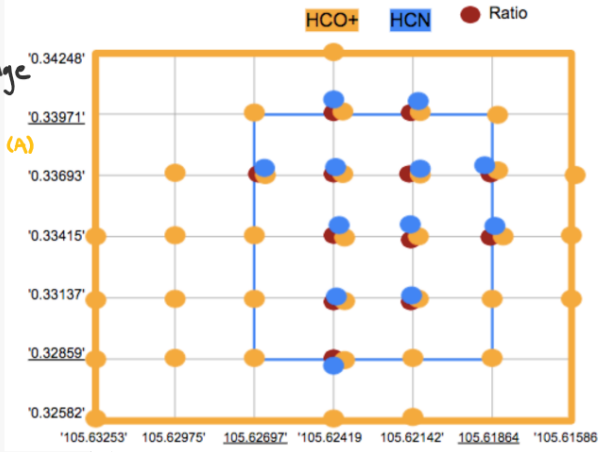
Importing data for plots

In [7]:
```
### Run before any plots ###
```

```
# Finding size of matrix to fill and plot
# <= for smaller array range, >= for larger array range
# regardless of range choice, density ratio will only populate areas both ranges cover

if len(glatAshort) >= len(glatBshort):
    lenglat = len(glatAshort)
    savedglat = glatAshort # store non-rounded glat value
    savedyPix = list(x[0][1]for x in coordAzip)
else:
    lenglat = len(glatBshort)
    savedglat = glatBshort # store non-rounded glat value
    savedyPix = list(x[0][1]for x in coordBzip)

if len(glongAshort) >= len(glongAshort):
    lenglong = len(glongAshort)
    savedglong = glongAshort # store non-rounded glong value
    savedxPix = list(x[0][0]for x in coordAzip)
else:
    lenglong = len(glongBshort)
    savedglong = glongBshort # store non-rounded glong value
    savedxPix = list(x[0][0]for x in coordBzip)

mainArray = np.zeros([lenglat,lenglong]) # array of desired size of zeros for chosen span of coordinates
hcoArray = np.zeros([lenglat,lenglong])
hcnArray = np.zeros([lenglat,lenglong])

# Assigning pixel numbers to each spot in matrix
xPixelsRev = list(shorten(savedxPix))
xPixels = list(reversed(xPixelsRev))
yPixels = list(shorten(savedyPix))

# Grid of pixel number
gridx, gridy = np.meshgrid(xPixels,yPixels)

# populating arrays for ratio, hco+, hcn
# ratio of column density will be found for pixels that contain data for both molecules

values = []
for i in range(len(coordAzip)):
    for j in range(len(coordBzip)):
        if (coordAzip[i][0] == coordBzip[j][0]):
            ratio = (coordBzip[j][3]/coordAzip[i][3])
            temp = (coordAzip[i][0], ratio)
            values.append(temp)

for x in values:
    for i in range(lenglat):
        for j in range(lenglong):
            if (x[0][0] == gridx[i][j] and x[0][1] == gridy[i][j]):
                mainArray[i][j] = x[1] # put ratio into large array

for x in coordAzip:
    for i in range(lenglat):
        for j in range(lenglong):
            if (x[0][0] == gridx[i][j] and x[0][1] == gridy[i][j]):
                hcoArray[i][j] = x[3] # put nA into array

for x in coordBzip:
    for i in range(lenglat):
        for j in range(lenglong):
            if (x[0][0] == gridx[i][j] and x[0][1] == gridy[i][j]):
                hcnArray[i][j] = x[3] # put nB into array
```

setting size of arrays

ex) for S138 the domain & range of the plots is dictated by HCO+ (A)

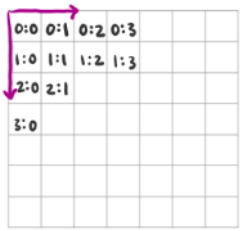


see below

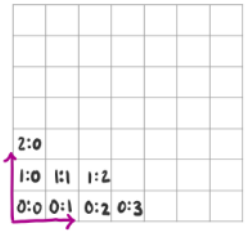
The ratio can only be found for the points where N(HCN) & N(HCO+) exist.

Filling the arrays

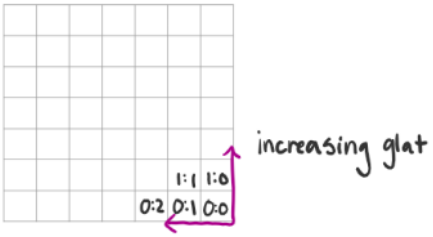
how an array is populated:



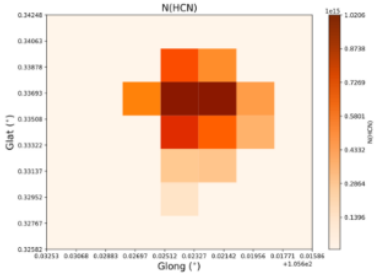
how values are plotted:



how we want our coordinates:



∴ we reverse our list of x pixels & the x-axis when we plot to get this:



Outputs: column density ratio

- A. Filled contour plot
- B. Pixel contour plot

} two kinds of plots
- note: filled contour rounds up when interpolating between layers

Outputs: N(HCO+), N(HCN), N[HCO+/HCN] plots

N(H2) and HCO+, HCN abundances

In [11]:
```
### MUST run "importing data for plots" first ###
```

```
# Temporary lists
xPixTemp, yPixTemp, glatTemp, glongTemp, nhcoTemp, nhcnTemp, nh2Temp, xhcoTemp, xhcnTemp= [], [], [], [], [], [], [], [], []
```

Outputs: pixel, coordinates, N(HCO+), N(HCN), N(H2), X(HCO+), X(HCN)

N(H2) plots

- A. Filled plots
- B. Pixel contour plots

outputs: X(HCO+), X(HCN), N(H2) plots

jc_pnautilus.ipynb

The pnautilus code is a newer script that is still being developed. It uses data from the previous code and therefore must be used after successfully running jc_combined.

MAKE SURE: your script is in the same folder as the ab files/

- Desktop>pnautilus-master>wccm>iras

Before running this code, create a folder within the folder for the respective source labelled "pnautilus_results". For example: Desktop>MAJORS>analysis>s138>pnautilus_results.

The goal of this script is to take the results from the previous code (column density ratio) and find the conditions that create these values. Ideally, we will find initial conditions that result in all pixels from the source achieving their predicted column density ratio at the same time.

Inputs

- Column density file (for HCO+, HCN and their ratio)
- Temperature

Outputs

- Text file with results from pnautilus
- Plots with results from pnautilus

Breakdown

[1]: **Python Libraries and Packages**

→ run each time you open Jupyter or the script

Global Parameters

In [2]:
```
#################################################
# Path to directories. Change for each new computer used.

# Path to the pnautilius executables
pnaut_path = '/Users/jane/Desktop/pnautilus-master/'

# Directory containing the working models
model_dir = '/Users/jane/Desktop/pnautilus-master/wccm/iras'

# Directory where the data are located
datadir = '/Users/jane/Desktop/MAJORS/analysis/s138/'

# Directory where the output files are to be written
resultdir = '/Users/jane/Desktop/MAJORS/analysis/s138/pnautilus_results/'

#################################################
# Input files. Already exist. Refer to external isntructions for creating files.

# Name of the file for N(HCN)/N(HCO+)
coldenratio = 'ColDenRatio.txt'

# Name of the input temperature data (chose hcn column density file because it has less entires than the entire temp.txt file- s
peeds things up)
temp_filename = 'HCN_colden.txt'

# Name of the N(H2) file
NH2data = 'abundance_data.txt'
```

only change source name

from jc_combined.ipynb

```
################################################
# density used in the RADEX models !! must use the same nh2
nh2 = 1.0e5      # nH2 cm^-3

outputsNum = 100 # number of pnautilus outputs (Log)

UVflux = 10.0 # UV flux for pnautilus code

startTime = 1.0
endTime = 1.000E+06

# Errors- use 0 if you don't want to consider error
ratioError = 0.1 # 10% → tests ratio (Radex) ±10%
avError = 0.1 # 10% → test AV (from NH2) ± 10%
```

we can test diff. conditions (nh2, UV)

```
# Pixels to run pnautilus for- temperature, NH2 and column density ratio MUST exist for specified pixels. If not, choose all.
# pixels = [['xPix', 'yPix']], pixels = [['xPix1', 'yPix1'], ['xPix2', 'yPix2'], ..., ['xPixN', 'yPixN']] or pixels = 'all'
# Example: pixels = [['3', '41'], ['2', '42'], ['3', '45']]
pixels = 'all'
```

it runs slow for 'all': when making changes, I use 1 pixel

```
################################################
# Names of various output files. Files will be created or overwritten if they exist.
# YOU ONLY NEED TO CHANGE THESE IF YOU WANT DIFFERENT NAMES THAN THE DEFAULT ONES
```

Import Data reads 3 files:

In [6]:
```
# Import data (column density ratio, temperature, N(H2))
```

matches pixels & returns list:

```
# List with elements ([xPix, yPix], ratio, I, NH2, NHCO, NHCN)
list = checkCoord(coordR, coordT, coordN)
```

Run pnautilus A function to run pnautilus when called

In [8]:
```
# Run the pnautilus code

def runpnautilus(item): # item is ([xPix, yPix], ratio, temperature, n(H2))
```

```
    output = []
    for i in range(len(ratioPnaut)):
        temp = (hco[i][1].astype(float)*2e21, hcn[i][1].astype(float)*2e21, ratioPnaut[i], hco[i][0]) # N(HCO+), N(HCN), ratio,
time
        output.append(temp)

    return output
```

output: for each pixel - N(hco+), N(hcn), N(hco+/hcn), time
all from pnautilus calculations

Create results

run pnautilus & collect results

In [10]:
```
plots, xPixFinal, yPixFinal, timeFinal, AVFinal, tempFinal, ratioFinal, radexNHCO, radexNHCN, pnautNHCO, pnautNHCN = [], [], [], [], [], [], [], [], [], [], []

for x in list: # "List" has elements ([xPix, yPix], ratioRadex, temperature, N(H2), N(HCO), N(HCN)) from Radex
    h2colden = x[3].astype(float)

    if avError == 0:
        AVnum = 1
    else:
        AVnum = 3 # change for number of av values used within error #DO NOT CHANGE THIS NUMBER- just don't

    timeAxis = []
    ratioAxis = []
    for n in (np.linspace((h2colden-h2colden*avError), (h2colden+h2colden*avError), AVnum)):

        pnautInput = (x[0], x[1], x[2], n, x[4], x[5])
        bigList = runpnautilus(pnautInput)
        ratioRequired = x[1].astype(float)

        if ratioError == 0:
            Rnum = 1
        else:
            Rnum = 5 # change for number of ratios used within error

        for r in (np.linspace((ratioRequired-ratioRequired*ratioError), (ratioRequired+ratioRequired*ratioError), Rnum)):
            diffReq = 10 # reset min difference for each new r
            # finding ratio closest to Radex ratio
            for i in range(len(bigList)): # "bigList" has elements of (N(HCO+), N(HCN), ratioPnaut, time) from pnautilus
                diffTemp = np.abs(bigList[i][2]-r)
                if diffTemp < diffReq:
                    diffReq = diffTemp
                    time = bigList[i][3]

            # Add results to lists
            xPixFinal.append(x[0][0])
            yPixFinal.append(x[0][1])
            timeFinal.append(round(time.astype(float),3)) # Holds values for all pixels
            timeAxis.append(round(time.astype(float),3)) # Holds values only for current pixel
            AVFinal.append(round(n.astype(float)/(2e21), 2))
            tempFinal.append(round(x[2].astype(float),2))
            ratioFinal.append(round(r.astype(float),4)) # Holds values for all pixels
            ratioAxis.append(round(r.astype(float),4))# Holds values only for current pixel
            radexNHCO.append('{:0.4e}'.format(x[4].astype(float)))
            radexNHCN.append('{:0.4e}'.format(x[5].astype(float)))
            pnautNHCO.append('{:0.4e}'.format(bigList[i][0].astype(float)))
            pnautNHCN.append('{:0.4e}'.format(bigList[i][1].astype(float)))
```

loops through pixels

→ all from previous code, NOT pnautilus

loops through AV ±10%.

loops through ratio ±10%.

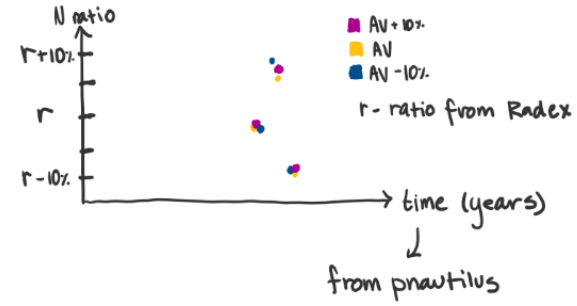
will be used in text file

previous data

pnautilus data

→ will be used in plot of individual pixel

The result:



Writing to xlsx

To write an data from a file to an xlsx file, use the following block of code:

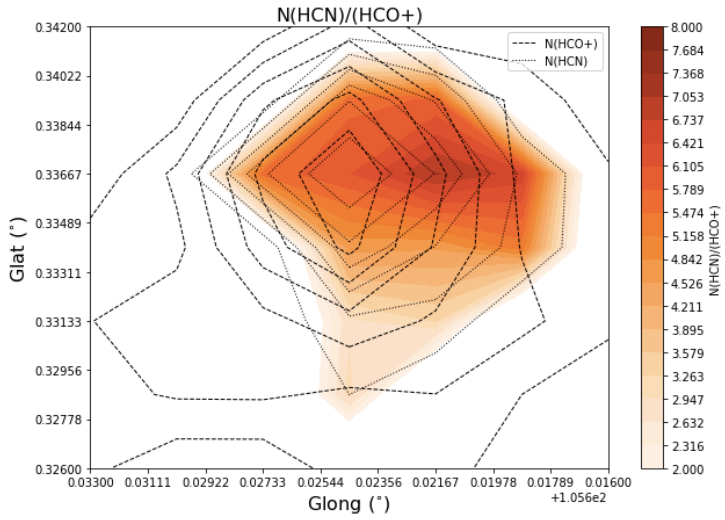
```
import pandas as pd
D = pd.read_csv(resultdir+pnautilus_outfile, sep="\t", header=None)

df = pd.DataFrame(data=D)

df.to_excel(resultdir+'pnautilus_data.xlsx')
```

To import to a usable sheet open google sheets or excel.
If using sheets: file>import>upload>select a file from your device and open the xlsx file.

Extra plot script

To plot N(HCO+)< N(HCN) and their ratio on one plot.

```
breaksHCO = np.linspace(np.floor(min(nAfloat)),np.ceil(max(nAfloat)),7) # can modify
number of fill levels
breaksHCN = np.linspace(np.floor(min(nBfloat)),np.ceil(max(nBfloat)),7) # can modify
number of fill levels
breaksRatio = np.linspace(np.floor(min(nRatioTemp)),np.ceil(max(nRatioTemp)),20) # can
modify number of fill levels

# filled contour
fig = plt.figure(figsize=(10,7), facecolor='white')
CS1 = plt.contourf(v, u, mainArray, breaksRatio, cmap='Oranges')

# colour bar
cbar = plt.colorbar(ticks=breaksRatio, orientation='vertical')
cbar.set_label('N(HCN)/(HCO+)')

# contour lines
CS2 = plt.contour(v, u, hcoArray, breaksHCO, colors='k', linestyles='dashed',
linewidths=1)
CS3 = plt.contour(v, u, hcnArray, breaksHCN, colors='k', linestyles='dotted',
linewidths=1)

# legend
h1,_ = CS2.legend_elements()
h2,_ = CS3.legend_elements()
plt.legend([h1[0], h2[0]], ['N(HCO+)', 'N(HCN)'])

# level labels
# plt.clabel(CS2, colors='k', inline=False)

# labels
plt.xlabel("Glong ($^{\circ}$)", fontsize=16)
plt.ylabel("Glat ($^{\circ}$)", fontsize=16)
plt.title('N(HCN)/(HCO+) Combination Plot', size=16)

# axes
plt.yticks(breaksy)
plt.xticks(breaksx)
plt.gca().invert_xaxis()

#plt.grid(color='white', ls='dotted', lw=1)

# saving the figure
plt.savefig(datadir+'ratioPlot_combined.png')
print('Plot saved to', datadir+'ratioPlot_combined.png')
```



Other Resources

Here I have listed a few resources I used to expand my knowledge or Rene recommended to me.

- "Introduction to CASA" by Bjorn Edmots, NRAO astronomer.
  - https://youtu.be/0QdqAibJH1w
- "The chemical effects of cosmic ray collisions with interstellar dust grain ice mantles "
  - UCF AVS Astrochemistry Webinar: Dr. Christopher Shingledecker
  - https://www.youtube.com/watch?v=pSTWSV_2_bY&t=169s
- Astrophysics Data System (ADS)
  - https://ui.adsabs.harvard.edu/classic-form
  - Search using 'Author last name, first initial, year of publication, journal, volume #, starting page number'
- Journals:
  - MNRAS - Monthly Notices of the Royal Astronomical Society
  - ApJ - Astrophysics Journal
  - A&A - Astronomy & Astrophysics
  - AJ - Astronomical journal
- W40 complex information
  - https://ui.adsabs.harvard.edu/abs/2016MNRAS.460.4150R/abstract
  - ADS search: Rumble et al. 2016, MNRAS, 460, 4150
- S138 information
  - https://ui.adsabs.harvard.edu/abs/2015MNRAS.454.4335B/abstract
  - ADS search: Baug et. al 2015, MNRAS, 454, 4335

Appendix I: Astrochemistry Basics

Glossary

Accretion: The process by where dust and gas accumulated into larger bodies such as stars and planets.

Accretion luminosity: As material moves in through the accretion disc, more luminosity is generated over a smaller region that results in emission at a higher temperature.

Column density (n): (or area density, surface density) a measure of the density through one dimension of a volume.

H_{II} region (diffuse nebula or emission nebula): a region of interstellar atomic hydrogen that is ionized.

- **UCH_{II} region:** Ultra Compact H_{II} region

Interstellar radiation field: Beside the diffuse matter, the interstellar space is filled with the electromagnetic radiation field produced by stars and interstellar matter.

JCMT Gould Belt Survey (GBL): using three JCMT survey instruments (SCUBA-2, HARP and POL-2) to survey nearby star-forming regions

Kiloparsec (kpc): is a measurement of distance equal to 1,000 parsecs or 3,260 light years.

Molecular outflow: An energetic mass-ejection phenomenon associated with very early stage of stellar evolution.

Photosphere: A star's outer shell from which light is radiated.

Radiative feedback (RFB) or stellar feedback: The process whereby large quantities of energy, momentum and mass are released by massive stars into the interstellar medium (ISM) surrounding star formation regions in galaxies.

SCUBA: Submillimetre Common-User Bolometer Array

Spectral index: a measure of the dependence of radiative flux density (that is, radiative flux per unit of frequency) on frequency. Given frequency v and radiative flux density S_v, the spectral index α is given approximately by: $S_v = v^{\alpha}$.

Star classes

- **B type:**
  - Young, hot, luminous, blue stars with lots of UV radiation.
  - Surface temperatures between 10,000 and 30,000 K.
  - Their spectra have neutral helium, and moderate hydrogen lines.
- **O type:**
  - Rare, hot, bright, blue-white star of spectral type O.
  - Due to their high mass, O-type stars end their lives rather quickly in violent supernova explosions, resulting in black holes or neutron stars after roughly one to fifteen million years.
  - Temperatures in excess of 30,000 kelvin (K).
  - Strong absorption lines of ionised helium, strong lines of other ionised elements, and hydrogen and neutral helium lines weaker than spectral type B.

- Most of these stars are young massive main sequence, giant, or supergiant stars, but the central stars of planetary nebulae, old low-mass stars near the end of their lives, also usually have O spectra.
- Typically located in regions of active star formation, such as the spiral arms of a spiral galaxy or a pair of galaxies undergoing collision and
- These stars illuminate any surrounding material and are largely responsible for the distinct coloration of a galaxy's arms.

Stellar association: a very loose cluster of stars that share a common origin but have become gravitationally unbound and are still moving together through space. Associations are identified by their common age and origin, chemical composition and amplitude and direction in their vector of velocity.

- **OB associations:** Contain 10–100 massive stars of spectral class O and B and are known as OB associations. OB associations are generally only a few million years in age or less and the O-B stars in the association will have burned all their fuel within 10 million years.
- **T associations:** Sparse populations of up to a thousand T Tauri stars are known as T associations. T associations are often found in the vicinity of the molecular cloud from which they formed. Some, but not all, include O-B class stars.
- **R associations:** Associations of stars that illuminate reflection nebulae are called R associations. These young stellar groupings contain main sequence stars that are not sufficiently massive to disperse the interstellar clouds in which they formed. This allows the properties of the surrounding dark cloud to be examined by astronomers.

Variable stars: a star that changes brightness/magnitude. Periodic or random.

- **T-Tauri:** newly-formed (< 10 million years old) low to intermediate mass stars (< 3 solar masses) with central temperatures too low for nuclear fusion to have started.
- **Herbig AeBe star:** another pre-main sequence star, that is not yet fusing hydrogen to helium. It is the intermediate mass equivalent of T-Tauri Stars with masses of between 2 and 8 solar masses. Once in main sequence, it will be either an A or B spectral type stars.

Visual extinction (AV): the dimming of distant objects due to the presence of dust in the interstellar medium along the line of sight. I.e., how opaque the cloud is.

Common equations

Abundance

$$X(x) = \frac{N(x)}{N(H2)}$$

Where x the molecule whose abundance is found.

Visual extinction

$$N(H2) = AV \times (2 \times 10^{21})$$

Concepts

Heating via feedback:

a. Internal mechanisms:

- Radiative heating by the stellar photosphere and accretion luminosity of young stellar objects (YSOs).
- Molecular outflows and shocks may also radiatively heat a cloud to a lesser extent.

Internal radiative feedback can suppress cloud fragmentation, leading to higher mass star formation.

b. External sources:
- Photons produced by stars, which can drive strong stellar winds and H_{II} regions.
- Interstellar radiation field.

External heating can influence the evolution of star-forming clouds.

Free-free emission: produced by free electrons scattering off ions without being captured—the electrons are free before the interaction and remain free afterward.

- Typically observed from large-scale, optically thin, diffuse H_{II} regions with an approximately flat spectral index, α_{ff}.
- Detected on smaller scales comparable to a protostellar core. On these smaller scales, free–free emission is produced by an ionized stellar wind where emission is considered partially optically thick.

Detecting sources:
- Ionized hydrogen (H+) means there's a UV source
- Areas with infrared but not a lot of visible: early/embedded protostar formation
- Molecules in "dark" clouds:
    - Don't emit photons by excited electrons jumping between orbits
    - Radiate weak radio waves whenever they change their rotation speeds
    - Can only spin in a few specific speeds: changing between these speeds produce unique energy emitted

Reaction rates

Take molecules A, B, C and D that undergo the following reactions:

A + B = C

A + C = B + D

D + A = B + C

Each reaction has its own reaction rate k_{AB}, k_{AC} and k_{DA}.

The rate of change of the density of a molecule is given by

$$\frac{dn_X}{dt} = create - destroy$$

For example,

$$\frac{dn_C}{dt} = n_A n_B k_{AB} + n_D n_A k_{DA} - n_B n_D k_{AC}$$

Radex solves this high order differential equation and outputs the column densities of each molecule that satisfy the conditions.

Dust grains

Variable	Meaning
J	Grain mantle/icy out layer
K	Grain surface
CR	Cosmic ray

'J'- on a dust grain
For example:
$JC + JC = 2C$ the two carbon atoms are in gas phase
$JC + JC = J2C$ the two carbon atoms remain on the dust grain

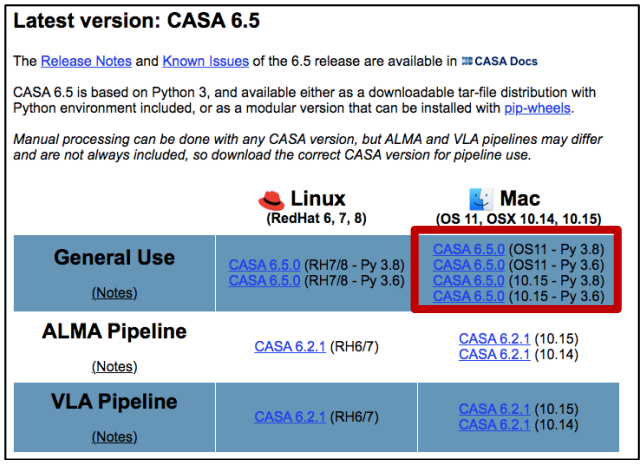
Appendix II: CASA

Downloading CASA

CASA can be downloaded at https://casa.nrao.edu/casa_obtaining.shtml . CASA works on Linux and Mac devices. While you can run Linux in a virtual environment or dual boot your Windows device, I was unable to download the software on my Windows computer successfully. If you only have a Windows device, a deep dive into the internet should help you download the software. For the rest of this document, I will be showing only the Mac interface. However, other than the installation, using the CASA software should be the same on all devices.

Download the latest verison of CASA that your system will support.

If your system will support it, use the "General Use" downloads for your macOS version.



Latest version: CASA 6.5

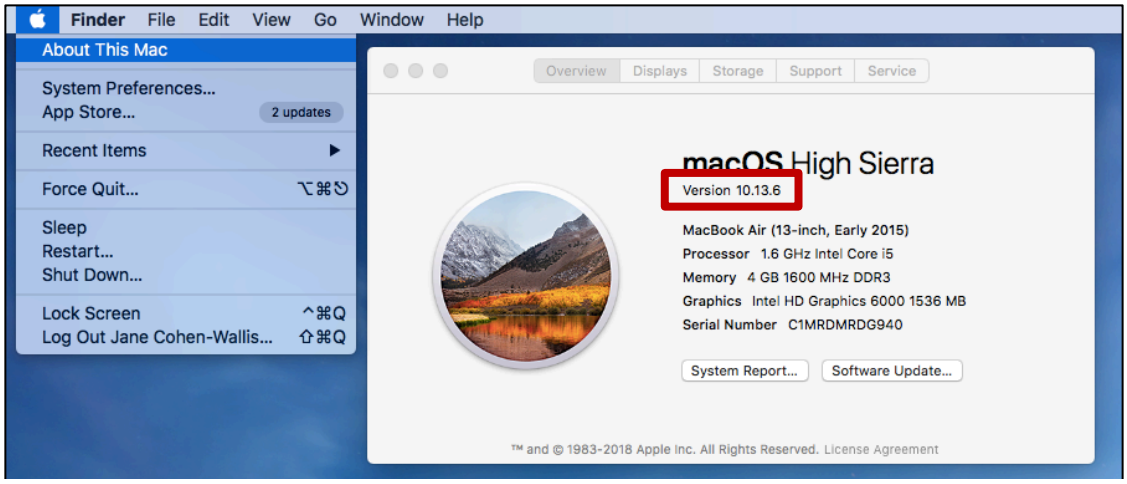
The Release Notes and Known Issues of the 6.5 release are available in CASA Docs

CASA 6.5 is based on Python 3, and available either as a downloadable tar-file distribution with Python environment included, or as a modular version that can be installed with pip-wheels.

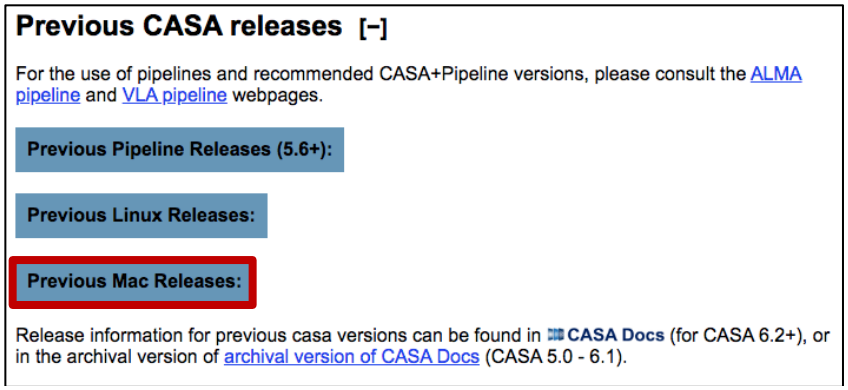
Manual processing can be done with any CASA version, but ALMA and VLA pipelines may differ and are not always included, so download the correct CASA version for pipeline use.

	Linux (RedHat 6, 7, 8)	Mac (OS 11, OSX 10.14, 10.15)
General Use (Notes)	CASA 6.5.0 (RH7/8 - Py 3.8) CASA 6.5.0 (RH7/8 - Py 3.6)	CASA 6.5.0 (OS11 - Py 3.8) CASA 6.5.0 (OS11 - Py 3.6) CASA 6.5.0 (10.15 - Py 3.8) CASA 6.5.0 (10.15 - Py 3.6)
ALMA Pipeline (Notes)	CASA 6.2.1 (RH6/7)	CASA 6.2.1 (10.15) CASA 6.2.1 (10.14)
VLA Pipeline (Notes)	CASA 6.2.1 (RH6/7)	CASA 6.2.1 (10.15) CASA 6.2.1 (10.14)

If your computer is an older OS, you can download a previous release of CASA. To determine the macOS version your computer system is, navigate to "About this Mac".

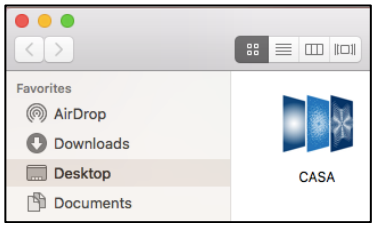


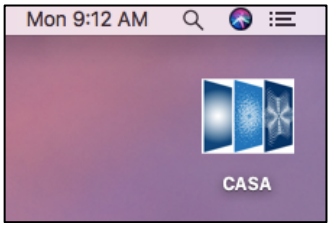
Next, select the appropriate version of CASA for your device from “Previous CASA releases”.



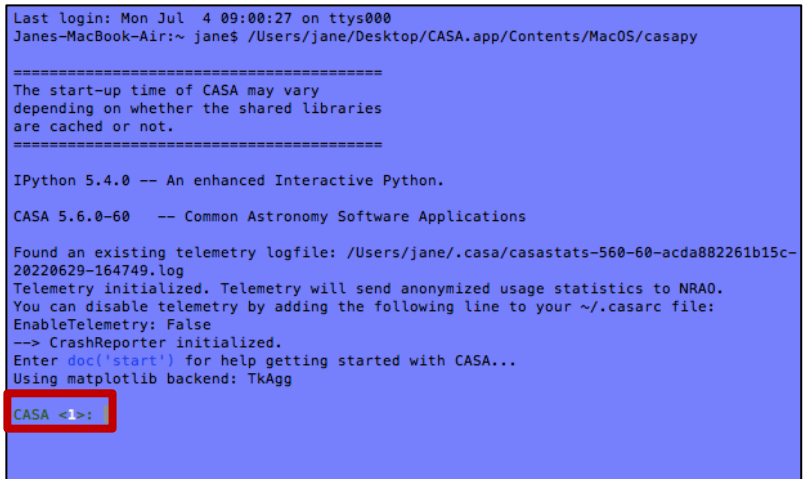
Opening CASA

You can use your terminal to open CASA if you have correctly set a path for the CASA app. However, I’ve found it easiest to navigate to where the app is stored (in my case, Finder or my desktop) and click the app icon to open CASA.





By opening the app, you are starting the CASA terminal. As seen below, ‘CASA <1>’ indicates you are now using the CASA terminal.



CASA terminal

In the CASA terminal, CASA uses Python, IPython and matplotlib which all work using 0-based indexing.

For more information on using lists, dictionaries, conditionals, loops, exceptions:
https://casa.nrao.edu/casadocs/casa-5.0.0/usingcasa/python-and-casa

You can access your computer system's shell directly through the CASA terminal or use functions referred to as CASAtasks. To access your system's shell through the CASA terminal, put a '!' at the start of any command you pass through the terminal. The line will be passed (excluding the '!') to your underlying operating system.
Most of the time however, I used CASAtasks. These are simply functions you enter in the CASA terminal. I've included specific example of CASAtasks that have been used in this project. More extensive documentation can be found in "CASA Docs" at https://casadocs.readthedocs.io/en/stable/ . I found the subsections "Task list", "Using CASA" and "CASA Fundamentals" especially useful.

CASAtasks

specsmooth()

This function smooths a FITS image to minimize the number of data points or channels it has. A FITS image with less channels will have less noise and will speed up any process we wish to run using the image data. However, the smaller number of channels also results in lower resolution and information from the image is lost when the file is smoothed. We aim for **channel sizes of 0.5 km/s**.

specsmooth(imagename="original_file.fits", outfile="newfile_smooth", axis=2, function="boxcar", width=20, overwrite=True)

a. "imagename" - the original spectral data cube.
b. "outfile"- the smoothed spectral data cube.
c. "width"- how many channels we smooth over.

Here is an example where smoothing was necessary:
The W40 HCN spectral cube originally had 7742 channels that were 0.03441 km/s apart.

$$channel\ distance \times N = 0.5$$

$$0.03441 \times 15 = 0.5$$

In this case, the “width” parameter used was 15 to create an image with channels ~0.5 km/s apart.

immoments()

The spectral data cubes are, well, cubes. This function takes the three-dimensional images and makes moment maps by integrating the intensity over the specified channels. This process allows us to see where the emission is by taking the spectrally smoothed spectrum “compressing” it into a two-dimensional image.

immoments(imagename="newfile.fits", outfile="newfile_mom0", moments=0, axis='spectral', chans="startchan~endchan”)

a. “imagename” - the smoothed spectral data cube.
b. “outfile” - the moment map of the spectral data cube.
c. “moments” - make moment 0 (integrated intensity).
d. startchan, endchan: the channel numbers that encompass the line emission.
    - To find: import the smoothed cube into casa and find the channels of the brightest emissions.
    - ex) chans=("range=[111chan, 123chan], restfreq=265.9, frame=LSRK")
        - The “restfreq” is that of the molecule being observed

imregrid()

Used to regrid an image onto the same grid/pixel size as the template.

imregrid(imagename=“original_file.fits", output="temp_regrid", template="template”)

a. “imagename” – original file with its own orientation and size.
b. “output” – output file with same pixel grid as the template.
c. “template”- template regrid follows. Do not including the .fits file extension.

To **regrid an image to match another FITS file**, use the moment 0 map of the HCO+ (or HCN) as a template.

imregrid(imagename=“original_file.fits", output="temp_regrid", template="newfile_mom0”)

To **switch between coordinate systems**, follow the examples below.

Galactic to J2000:
imregrid(imagename="file.fits", output=“file_J2000", template="J2000”)

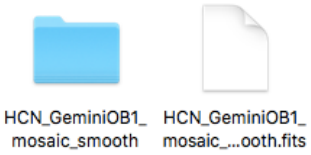
From J2000 to Galactic:
imregrid(imagename=“file.fits", output=“file_gal", template=“GALACTIC”)

exportfits()

As the name suggests, this function exports a file as a FITS file. This conversion allows us to open the image in CASA and carry out tasks that require a FITS image as an input.

exportfits(imagename=”original_file”, fitsimage=”newfile_smooth_mom0.fits”)

a. “imagename” - the non-FITS file.
b. “outfile” - the FITS image file.



The function will convert the directory (left) to a proper FITS file (right).

Another way to create a FITS file, is through the CASA viewer. This method allows you to regrid and convert to a FITS file in one step. Visit “Creating a New FITS File” below to see how.

Common fixes

File names for CASAtasks

- When writing input and output file names, the CASA documentation uses “ ”. However, I found using ‘ ’ worked and would otherwise get an error that the file did not exist. Other times, I had to delete and re-write the ‘ ‘ in the CASA terminal. It was as if the copy-pasted characters weren’t read by the terminal.

FITS files:

- If you put the .fits extension as part of the output file name for specsmooth(), immoments() and imregrid(), a file such as “filename.fits” will be produced, but this is NOT a FITS file. These functions do not output a FITS file, regardless of what you name them.
- After using any of the above functions, use exportfits() to create a proper FITS file.

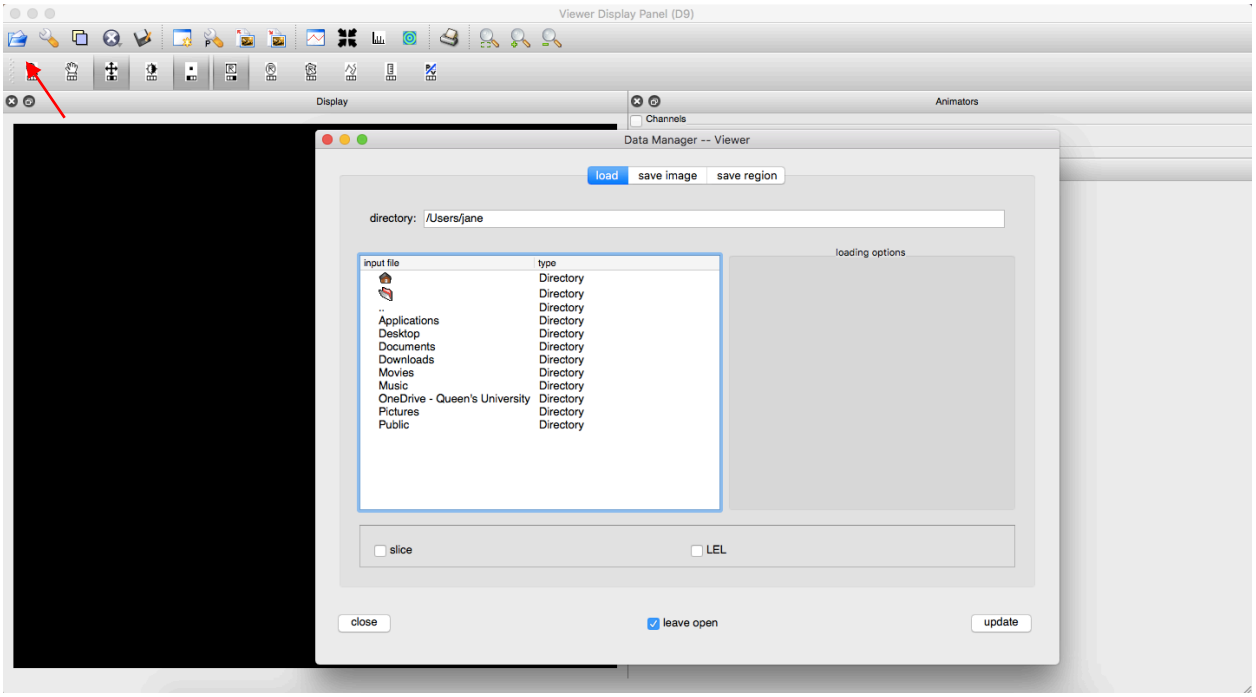
CASA imview

Type imview into the CASA terminal to view images in raster, contour, vector or marker form.

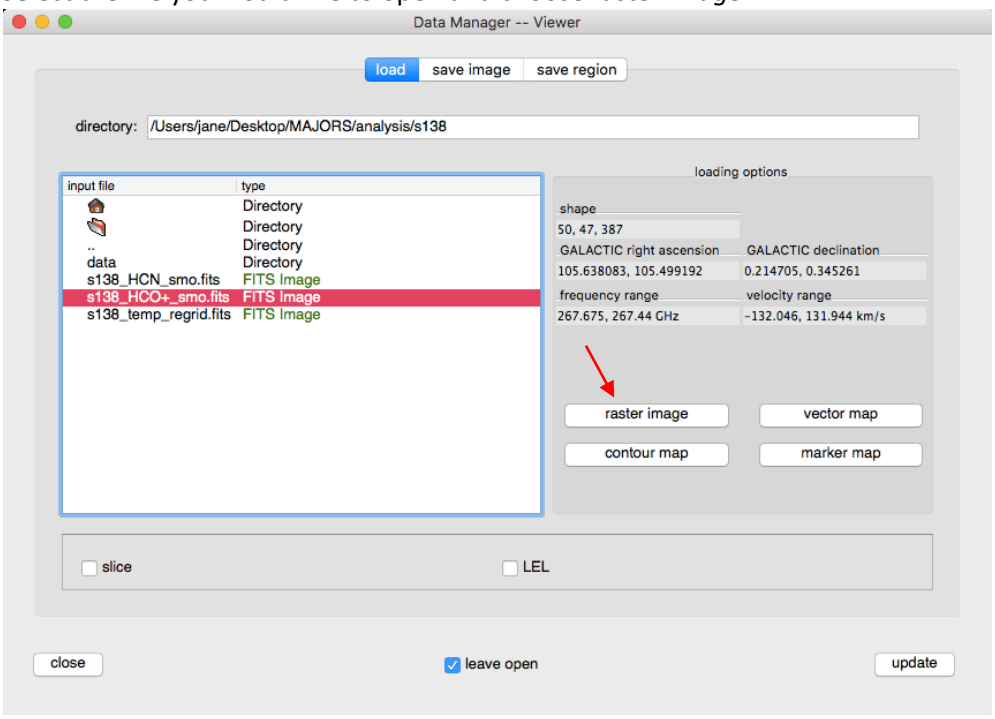
```
CASA <5>: imview
--------> imview()
```

Data Manager

To open an image, use the  icon in the top left corner. The Data Manager will open, and you can navigate to where your files are stored.



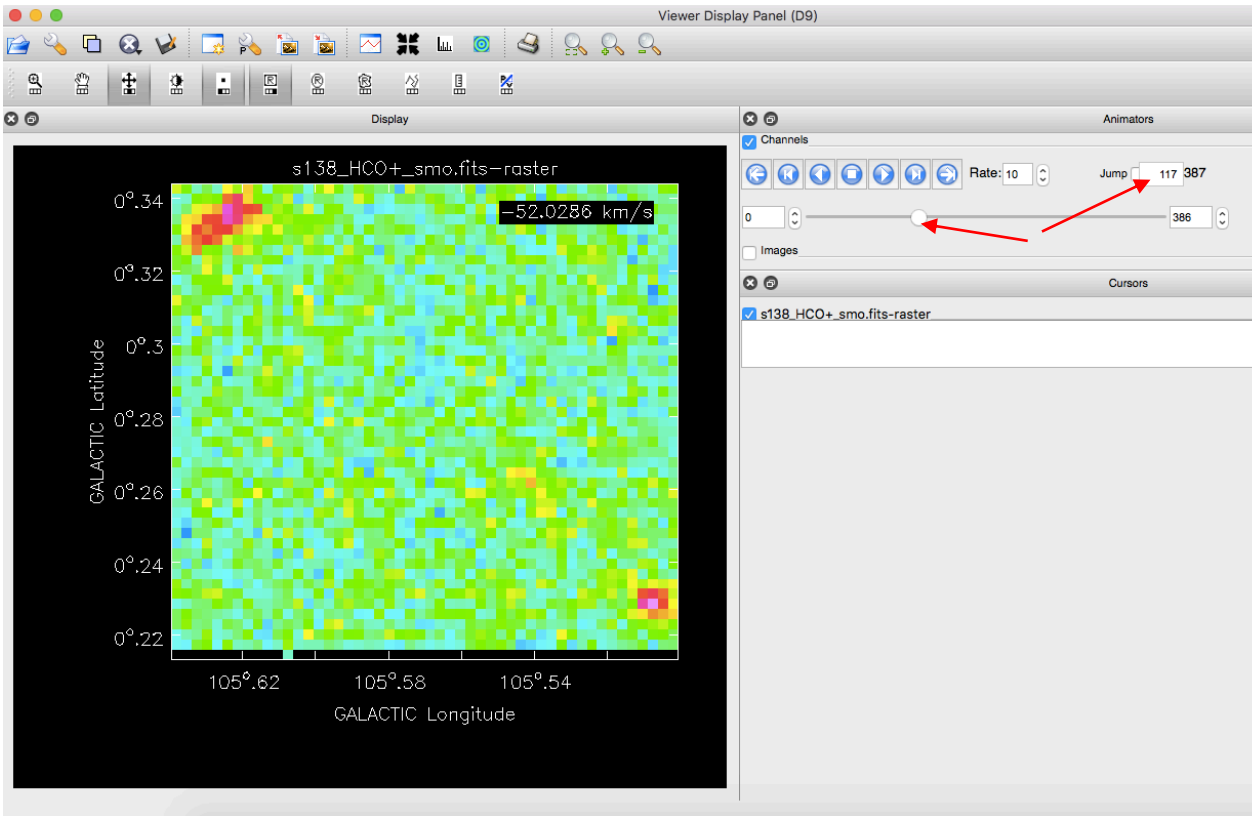
Select the file you would like to open and choose *raster image*.



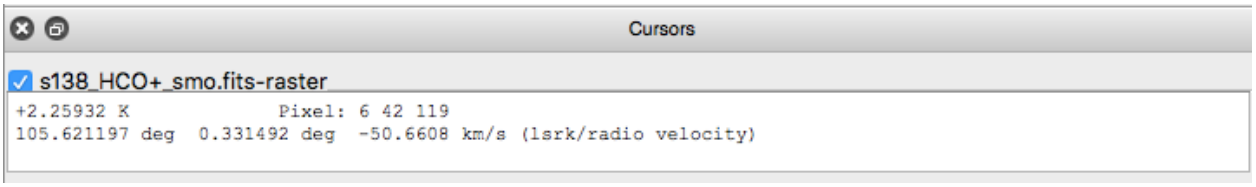
Viewer Display Panel

The fits files for HCO+ and HCN are initially three-dimensional data cubes. The “x” and “y” axes represent galactic longitude and latitude. The third dimension is the velocity at which the source is travelling. You can move through this third dimension using the *channels* slider. You can manually enter

a channel number, use the slider, move one channel at a time using  or allow the channels to play automatically using  .
Here you can see that at channel 117/387 an emission is clearly visible.



Moving your cursor over the image will give the details of the pixel your cursor is above. This data includes the location of the pixel, the temperature at this point and the velocity of the channel the image is currently displaying.



The following icons in the task bar were used frequently but the best way to get used to CASA is to experiment and explore different functions.

 Open a new image viewer window.
 Zoom in and out of the image.

 “Point Marking”- Allows you to place a marker on the image. This was mainly used when selecting a pixel to perform a Gaussian fit on.

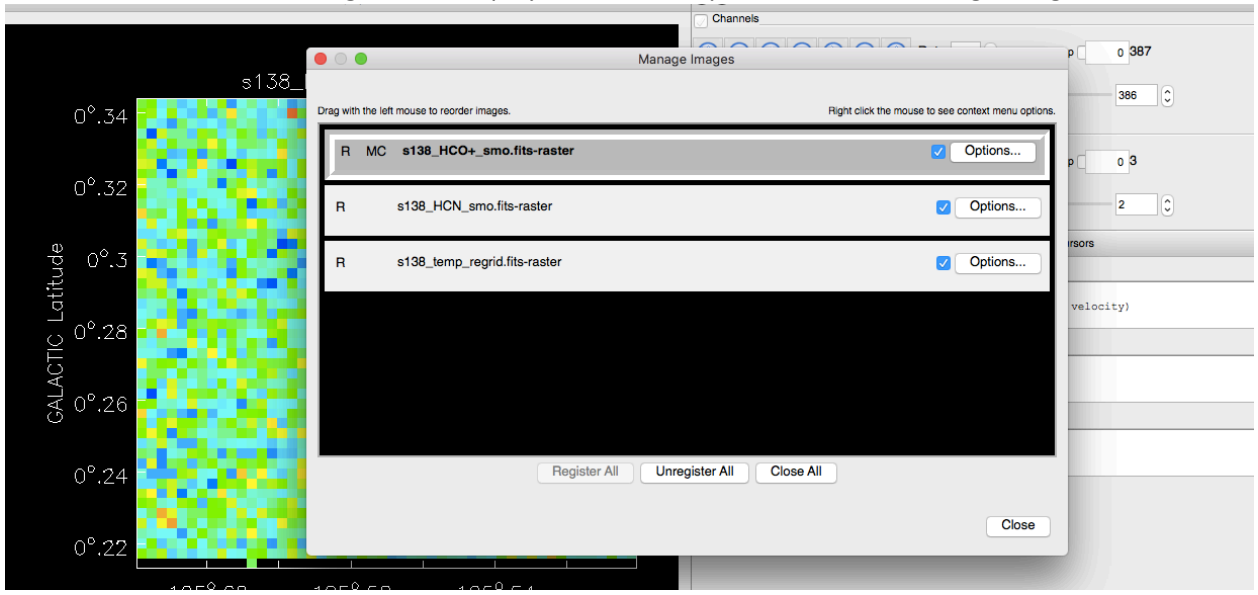
 “Zooming”- If you want to zoom into a specific area of the image, select this icon. You can then draw a rectangle on the image and double click inside the rectangle to zoom into the area contained in the rectangle.

 “Panning”- If you have zoomed into the image, you can pan around the image using this tool.

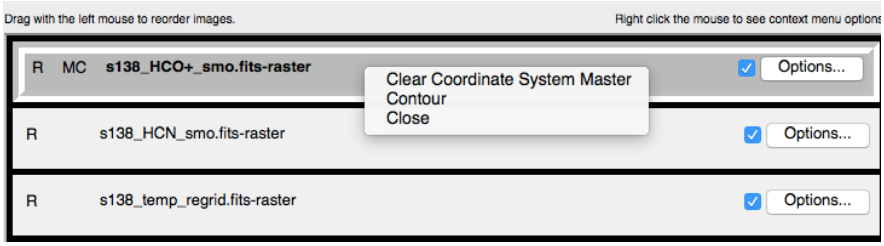
Manage Images

 Opens the window that allows you to navigate between multiple images.

In this window you can control which images show up on the Display Panel. Selecting/unselecting the blue boxes will close the image in the Display Panel but keep it listed in the Manage Images.



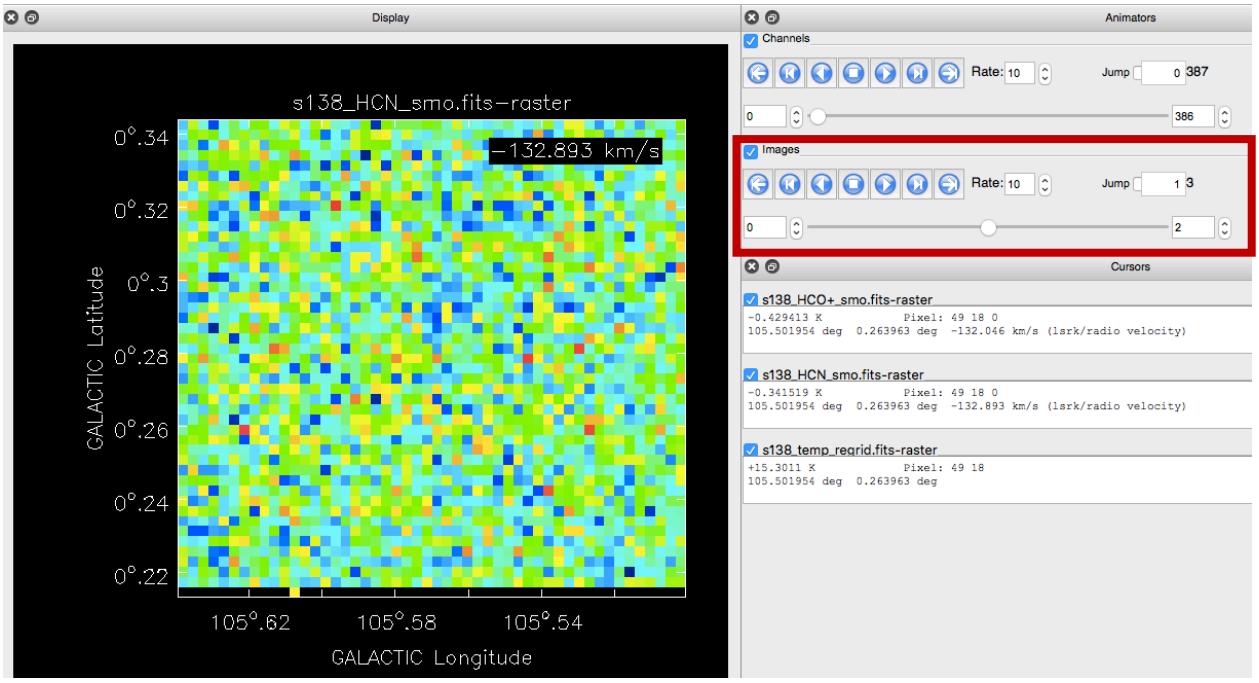
If you want to completely close the image, right click on the file and choose “Close”.



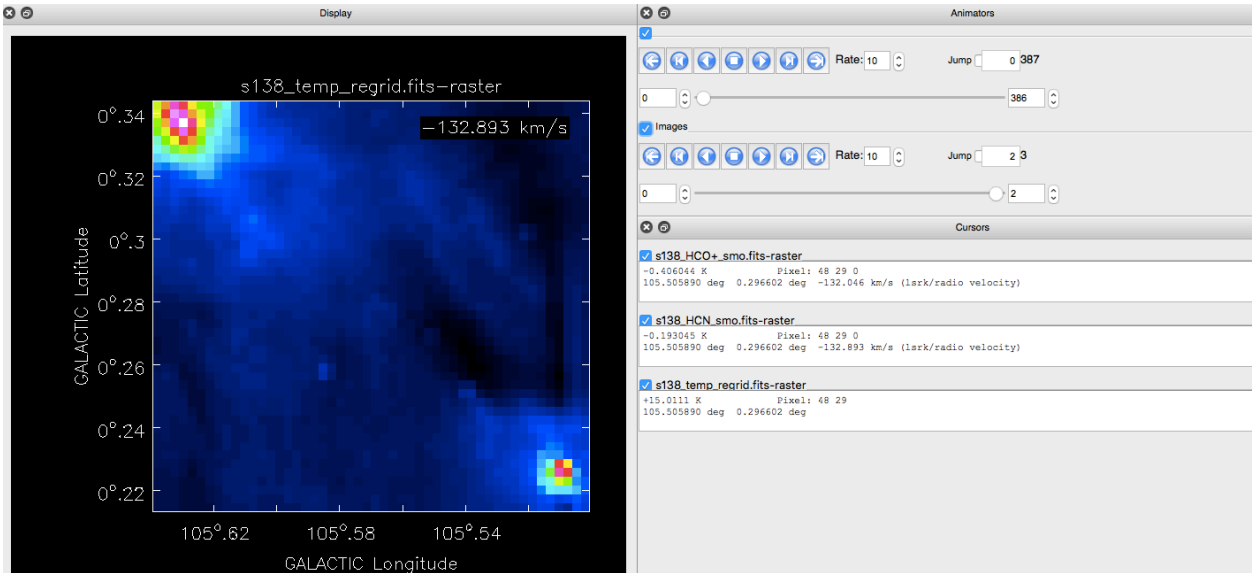
Opening multiple files

If you have opened multiple files and have them all selected in Manage Images, you can easily change between images in the Display Panel.

There will be a slider in “Images” that will allow you to switch between images, similar to switching between channels.



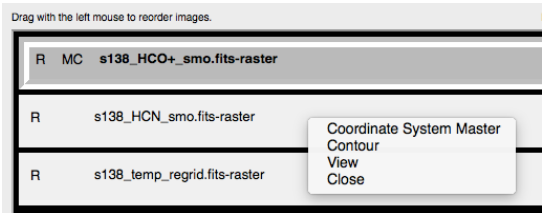
In this example, if you were to move the slider all the way to the right, the third image (the temperature map) will be shown.



Creating a new FITS file

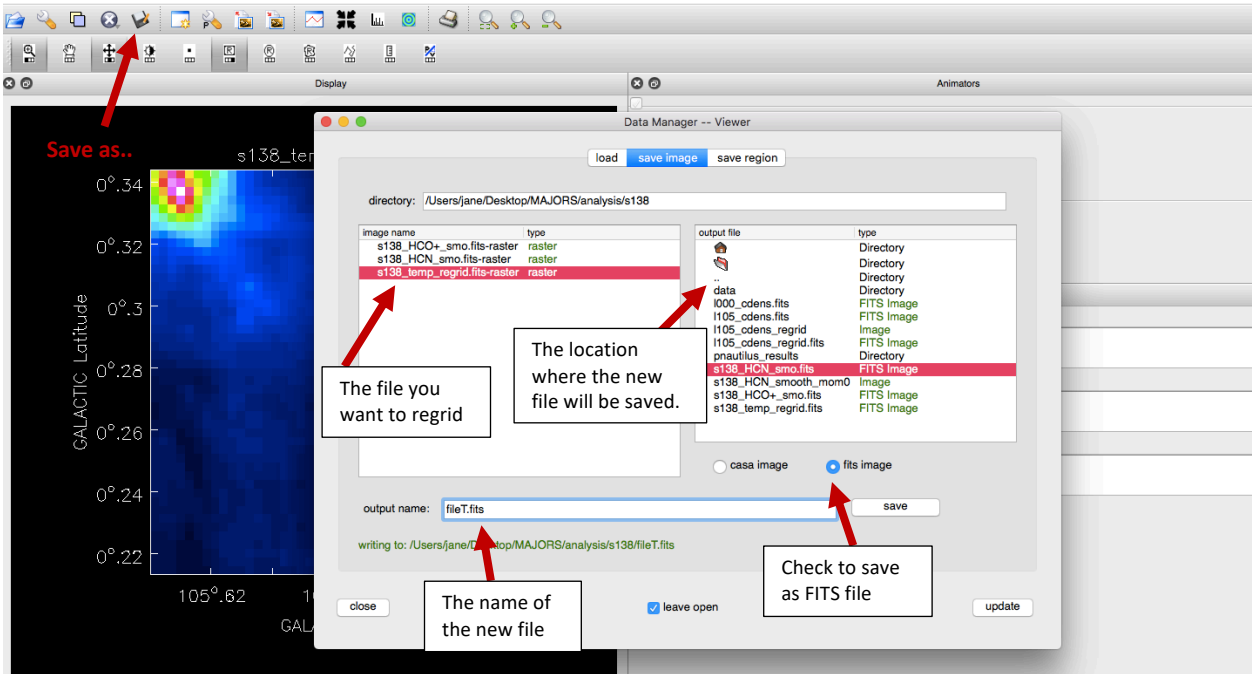
To regrid an image and save it as a FITS file, you can use imregrid() and exportfits() in the CASA terminal or you can use the following method.

In Manage Images, one file will always be the “Coordinate System Master” or CSM. The CSM is the image in Manage Images that has a darker box. Right click a file and select “Coordinate System Master” to set it as the CSM.



Recall the way you can switch between multiple images but the coordinates do not change? This is because all the open images are on the coordinate system of the CSM. Therefore, to regrid a file, the file acting as the template should be the CSM.
Now, any file saved will be aligned with the coordinates of the CSM and will be a proper FITS file.

Use  to save your new file.

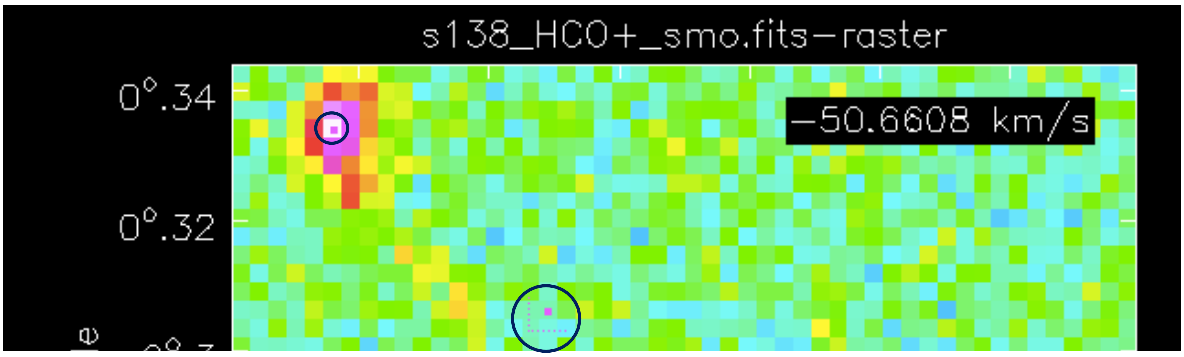


Note: you must select an output file, even if you don't wish to overwrite it. Choose one in the correct directory and by changing the name, the selected file will not be overwritten.

Spectral Viewer

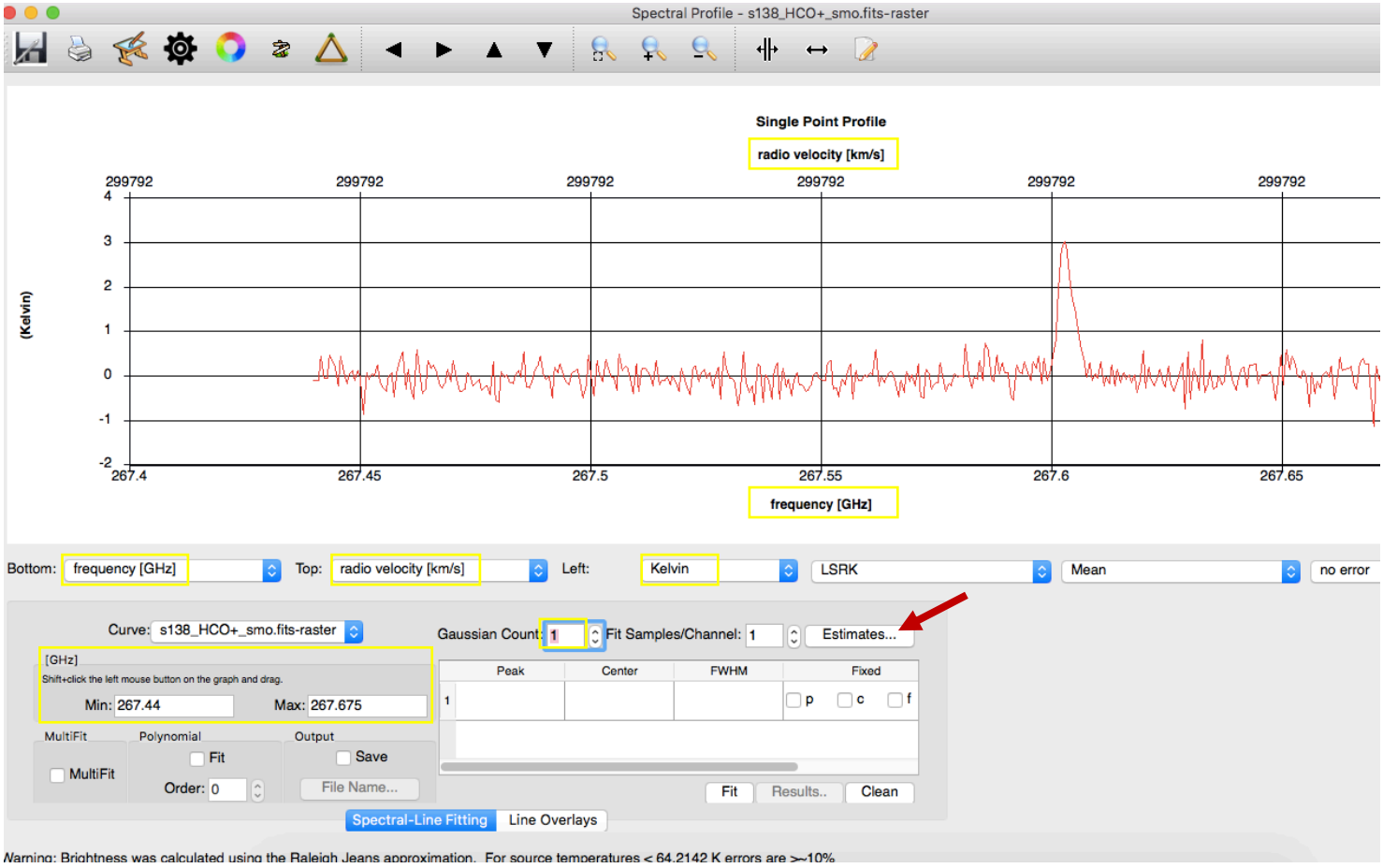
The spectral viewer has many features that I have not explored. For this project, it was used to create Gaussian fits of the data. These fits were compared to the plots produced by the code to ensure the code was creating a correct fit.

To select which pixel to perform the Gaussian fit on, use the Point Marking cursor and select the pixel. This will place a small pink square on the pixel you have selected, as seen below (top left). To get rid of a marker, hover your cursor over the pixel and dotted lines (lower pixel) will appear around the pixel. Click 'esc' and the marker will be removed.



Once you have selected one pixel, use  to open the Spectral Viewer.

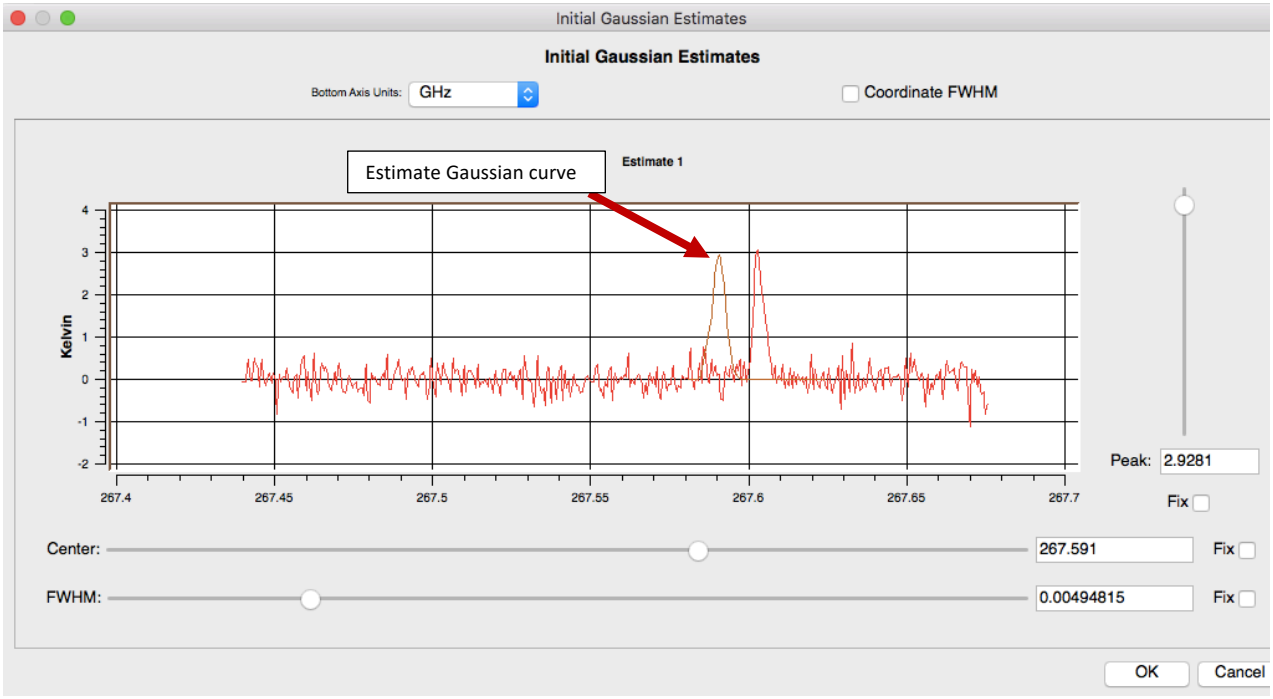
This is the initial window you will see:



The setting you will set:

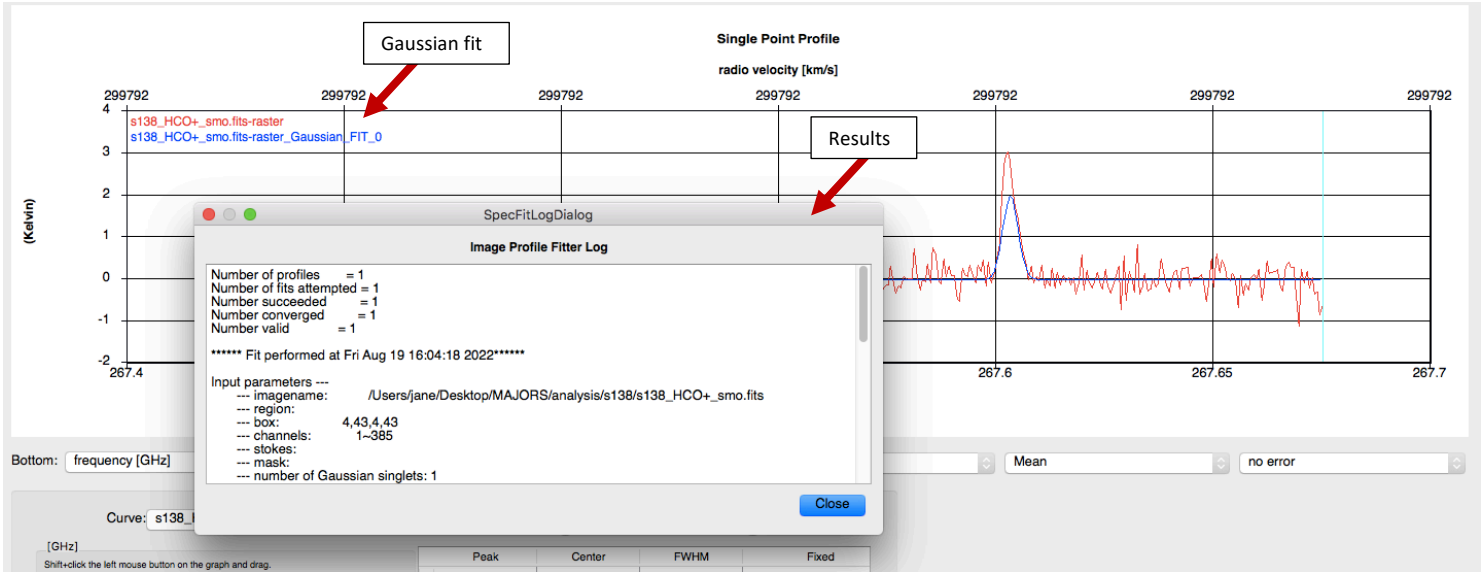
- Bottom: frequency [GHz]
- Top: radio velocity [km/s]
- Left: Kelvin
- Gaussian Count: 1
- In  make sure the “optical” option is NOT selected.

Next, select estimates and the following window will open:



Using the sliders, the estimate Gaussian curve (yellow) can be created. Here I've moved the centre estimate away from the centre of the actual plot to show it clearly. However, the centre of both curves should be lined up.

Next select "OK" and then "Fit". The Gaussian fit will appear with the numerical results and the plot.



We ran into several problems while using the Spectral Viewer to create Gaussian fits. We were able to work around them and produce acceptable plots. However, we do not know why our tricks worked. But here they are:

- When you mark a pixel and then open the Spectral Viewer, the range of the emission will be automatically inputted. This range will include the entire emission. Changing the range to eliminate noise and be closer to each side of the curve results in the fit not working. You must not change the frequency range.
- The top and bottom axes cannot be switched. When the centre of the curve is at a negative number (in this case -50.6608 km/s), the fit will not work. Therefore, we put frequency at the bottom.

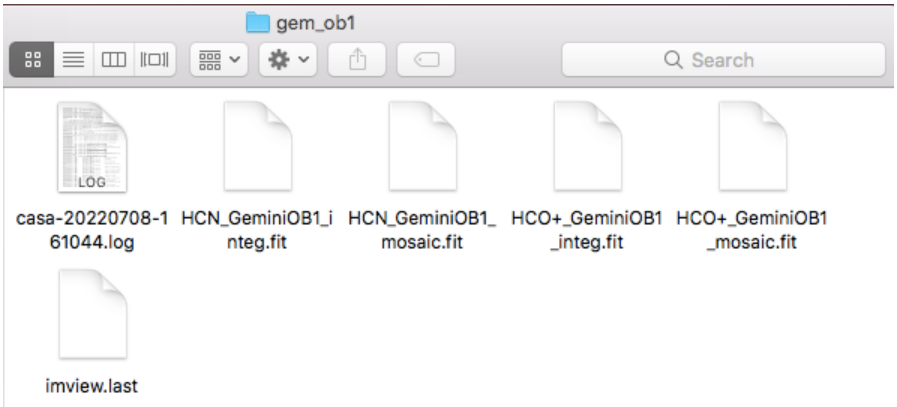
Appendix III: Set Up for jc_combined.ipynb

The following is the process I followed after receiving the first files for gem_ob1. It is a complete process from the very start, running through all the steps it takes to use the jc_combined.ipynb script.

Molecule Data

I've organized my files as Desktop>MAJORS>analysis. Within 'analysis' I have a folder for each new source.

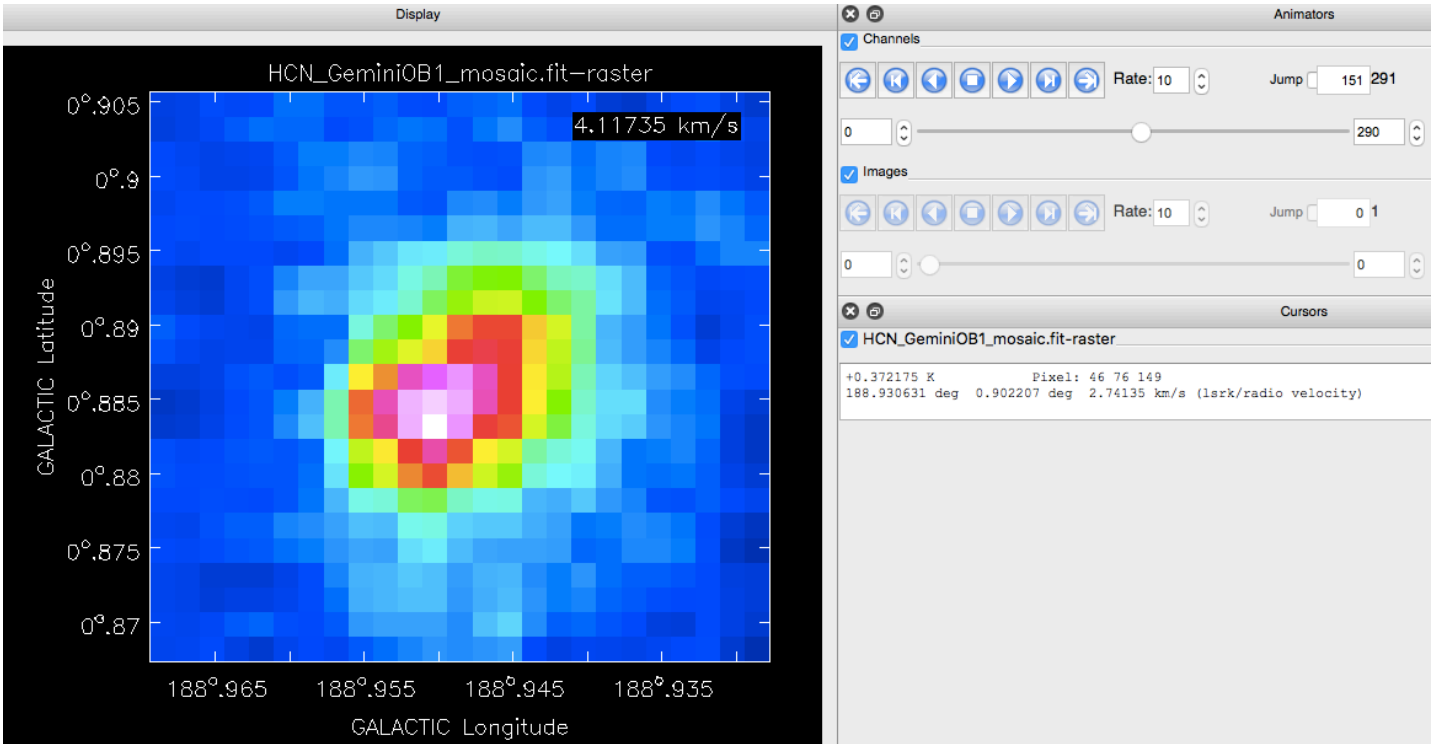
Start with files from your project head and move them to their respective folder:



You must have the two spectral data cubes. In this case they are called XXX_GeminiOB1_mosaic.fits.

Now go retrieve a temperature map and column density map for the source.

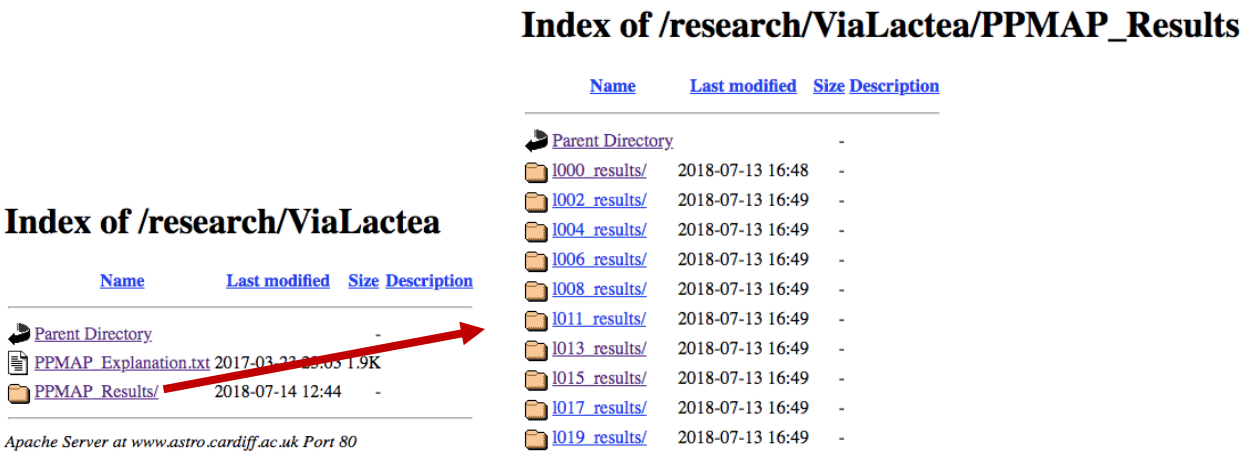
Open the files on CASA imview to get a feeling for what the source looks like:



The above image is a spectral data cube for HCN from gem_ob1. Two axes dictate the spatial coordinates and a third gives the velocity of the source. Changing the channel allows us to look through velocities and determine the velocity when the source gives off the brightest emission. In this case, that is channel 151/291 at a velocity of 4.11735 km/s. You can also see the coordinates of the source and moving your mouse over the source will give you an idea of the temperatures/intensities of the source.

Temperature and H2 Column Density

The temperature and H2 column density files are not part of the JCMT survey, so we must obtain them ourselves. Visit http://www.astro.cardiff.ac.uk/research/ViaLactea/ and navigate to PPMAP Results/.



Index of /research/ViaLactea

Name	Last modified	Size	Description
Parent Directory		-	
PPMAP_Explanation.txt	2017-03-22 05:03	1.9K	
PPMAP_Results/	2018-07-14 12:44	-	

Apache Server at www.astro.cardiff.ac.uk Port 80

Index of /research/ViaLactea/PPMAP_Results

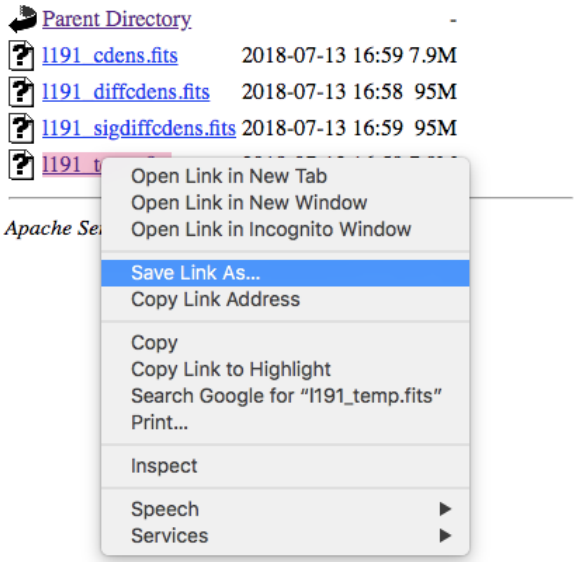
Name	Last modified	Size	Description
Parent Directory		-	
l000_results/	2018-07-13 16:48	-	
l002_results/	2018-07-13 16:49	-	
l004_results/	2018-07-13 16:49	-	
l006_results/	2018-07-13 16:49	-	
l008_results/	2018-07-13 16:49	-	
l011_results/	2018-07-13 16:49	-	
l013_results/	2018-07-13 16:49	-	
l015_results/	2018-07-13 16:49	-	
l017_results/	2018-07-13 16:49	-	
l019_results/	2018-07-13 16:49	-	

Naming convention: “lNNN” where the first character is a lower-case L and the three digits are the approximate galactic longitude of the center of the map’s location.

For example, s138 has a galactic longitude range of 105.501937 - 105.638050 degrees. The map “l105” can be used for s138.

The two maps to use:
lNNN_cdens.fits = 2D map of integrated line-of-sight column density (NH2)
lNNN_temp.fits = 2D map of mean line-of-sight dust temperature

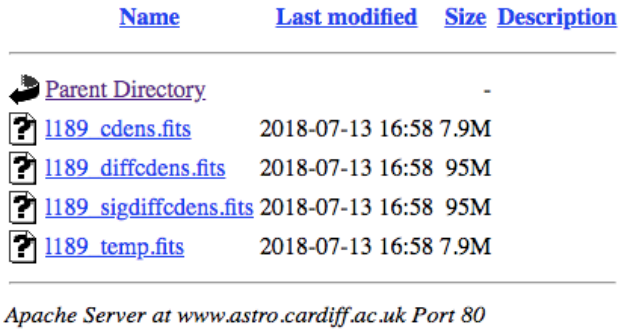
They can be saved as a FITS IMAGE by right clicking the map and selecting ‘Save Link As…’



You do not need to add the extension- it will download as a FITS file.

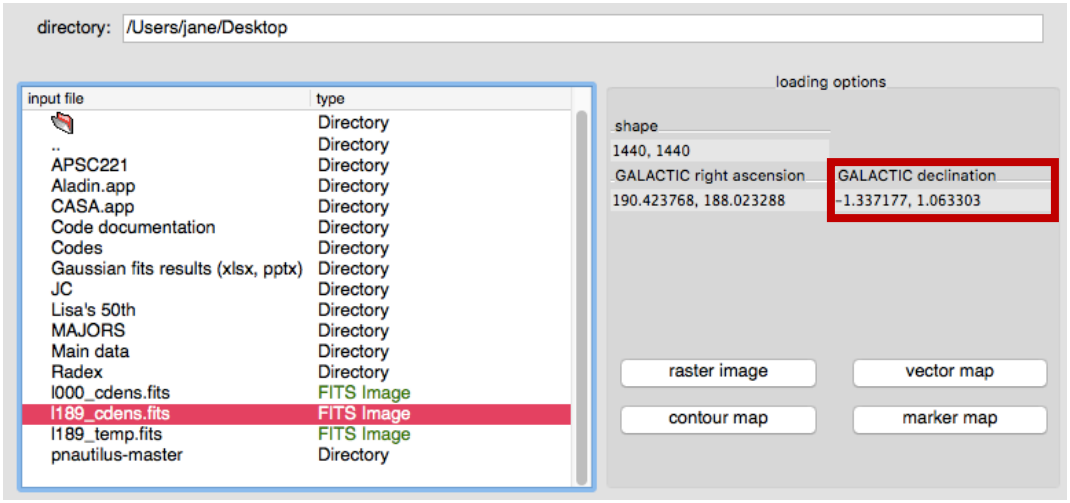
It is important to make sure the galactic longitude range of the map matches that of the source. If it does not, the above resource cannot be used to retrieve the PPMAPs for the source. Contact Plume if this is the case and maps from other scholars can be used.

For example, the galactic longitude range of gem_ob1 is 188.760208 - 189.006917 degrees.
Here is the corresponding directory for gem_ob1:



The galactic latitude range of gem_ob1 is 0.774841 - 1.074843 degrees.

Opening the file in CASA imviewer:



The galactic latitude (or declination) is in the same region as the source. It does not line up exactly, but it has enough overlap to give us ample data points.

Formatting Files Using CASA

Next, each file will be manipulated using the CASA terminal to ensure they are all on the same coordinate system and are a proper FITS IMAGE.

A. Spectral smoothing

```
specsmooth(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic
.fit',
outfile='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_smooth',
axis=2, function="boxcar", width=1, overwrite=True)
```

For this project, we aim for a channel size of 0.5 km/s. In the case of gem_ob1, the channel size was already ~0.5 km/s. Therefore, smoothing was only carried out for sake of consistency and naming convention, hence the width of 1.

B. Regridding: J2000 to Galactic

```
imregrid(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_s
mooth',
output='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_smooth_gal',
template="Galactic", overwrite=True)
```

The data file was already given in Galactic coordinates, so the regridding step was not required. If this was not the case, the above command would have been used.

C. Converting to a FITS IMAGE

```
exportfits(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic
_smooth',
fitsimage='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_smooth.fit
s', velocity=True, overwrite=True)
```

D. Creating a moment map

```
immoments(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_
smooth.fits',
outfile='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_smooth_mom0',
moments=0, axis='spectral', chans=('range=[141chan, 161chan], restfreq=265.9,
frame=LSRK'))
```

The channels were chosen by selecting the channel of the spectral cube with the brightest emission. In this case, it is channel 151, as seen in the mosaic image on CASA above. Using a range of 20 channels, this gives us range=[141chan, 161chan]. The frequency used is that of HCN: 265.9 GHz.

Again the image must be converted to a FITS IMAGE.

```
exportfits(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_smooth
_mom0',
fitsimage='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_smooth_mom0.fits'
, velocity=True, overwrite=True)
```

E. Regridding: using a template image

```
imregrid(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_s
mooth.fits',
output='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_smooth_regrid
', template='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_smooth_mom0',
overwrite=True)
```

Note here the regridding was redundant because the moment map was created using the "HCN_GeminiOB1_mosaic_smooth.fits" file. But it is good practice to ensure all your maps are using the same coordinates.

And again, convert to FITS IMAGE.

```
exportfits(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic
_smooth_regrid',
fitsimage='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCN_GeminiOB1_mosaic_smooth_reg
rid.fits', velocity=True, overwrite=True)
```

Repeat for HCO+. The only differences will be the file names and the change of frequency for immoments().

Next, the N(H2) must be formatted using the CASA terminal.

```
imregrid(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_cdens.fits',
output='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_cdens_regrid',
template='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCO+_GeminiOB1_smooth_mom0',
overwrite=True)

exportfits(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_cdens_regrid',
fitsimage='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_cdens_regrid.fits',
velocity=True, overwrite=True)
```

And temperature maps.

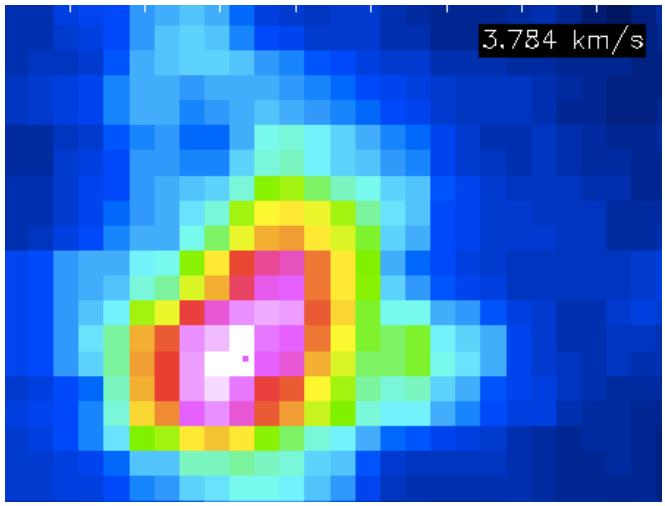
```
imregrid(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_temp.fits',
output='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_temp_regrid',
template='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/HCO+_GeminiOB1_smooth_mom0',
overwrite=True)

exportfits(imagename='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_temp_regrid',
fitsimage='/Users/jane/Desktop/MAJORS/analysis/gem_ob1/l189_temp_regrid.fits',
velocity=True, overwrite=True)
```

Global Parameters

Now that all the images are in the correct form, the parameters block can be filled out. I've highlighted the parameters that will be changed for each new source.

```
################################################
# Path to directories. Change for each new computer used.

# directory where the data are located and the files are to be written
datadir = '/Users/jane/Desktop/MAJORS/analysis/gem_ob1/'

# Path to the radex molecular data files
radexpath = '/Users/jane/Desktop/Radex/data/'

# Extension to the molecular data files
extension = '.dat'

# Path to the radex executable program (how executable command will be called)
radexexec = '/Users/jane/Desktop/Radex/bin/radex'

# How executable command will be called: os.system(radexexec+ '<'
+radexpath+'/'+inputFile+ '> /dev/null')


################################################
# Input files. Already exist. Refer to external isntructions for creating files.

# Name of the spectral data cubes on which to perform gaussian fitting
hco_spec_filename='HCO+_GeminiOB1_mosaic_smooth.fits'    # input fits file name
hcn_spec_filename='HCN_GeminiOB1_mosaic_smooth.fits'     # input fits file name

# Name of the input temperature map/FITS file
# If temperature file DNE for source, use 'none' and set constant temp. in "Tkin Data
Retrieval" block.
temp_fitsfilename = 'l189_temp_regrid.fits'

# N(H2) file
NH2regrid = 'l189_cdens_regrid.fits'


################################################
# Parameters for the Gaussian fitting.  CHANGE FOR EACH NEW SOURCE

vel_guess = 3.5   # The initial guess for the centroid velocity of the gaussian fit
```

Found using the velocity that appears to have the brightest emission.



For gem_ob1, it appears to be at 3.784 km/s.

We can check by performing a Gaussian fit on the brightest pixel.
The fit gives Center : 3.517 +/- 0.030 km/s. Therefore, taking 3.5 km/s as a velocity guess is appropriate.

```
peak_guess = 10     # the initial guess for the maximum line strength (sets the y
scaling of the plots)
```

The Gaussian fit of the brightest pixel is used again to find the peak temperature guess. The fit gives:
Peak : 9.07 +/- 0.12 K. Rounding up to the next whole number gives a guess of 10 K.

```
dv_min = 1          # set a minimum acceptable line width
dv_max = 10         # set a maximum acceptable line width
snr_min = 3.0       # set a minumum acceptable SNR
```

dv_min, dv_max and snr_min do not need to be adjusted for each source.

```
# set the range of pixels in the x direction over which to loop
min_xpix = 27
max_xpix = 45

# set the range of pixels in the y direction over which to loop
min_ypix = 58
max_ypix = 80
```

The more pixels selected, the longer it will take to compute the Gaussian fits and column density of each pixel. I chose to use a range between 20 and 40 pixels in each direction. The pixels were chosen using CASA imviewer and include the brightest emission.

```
################################################

# density for the RADEX models
nh2 = 1.0e5       # nH2 cm^-3

################################################

# Names of various output files.  YOU ONLY NEED TO CHANGE THESE IF YOU WANT
```

```
# DIFFERENT NAMES THAN THE DEFAULT ONES
```

I've taken these out to save space- but they appear in the parameter block and you can change them if you want to.

Jupyter Lab

Now duplicate the code from any source and rename it for the source you are using. In this case, I duplicated jc_combined_w40.ipynb and renamed it jc_combined _gemob1.ipynb.

You are now ready to go!