Gutermuth Criteria

CITA

Table of Contents

1. Data Preparation

1.1. Required Packages

```
import pandas as pd
import pandas.io.sql as pd_sql
import sqlite3 as sql
import numpy as np
from pandas import DataFrame
import math
```

1.2. Prepare Original Table

- Download all data in the relevant catalog into a ASCII file.
- Read into DataFrame:

```
gutermuth = pd.read_csv('/home/kecai/w49/gutermuth.tbl', sep = ',', header = None,
skiprows = 1, names = ['desig', 'ra', 'dec',
                       'm3_6', 'm4_5', 'm5_8', 'm8_0', 'm24', 'mJ', 'mH', 'mKs',
                       'dm3_6', 'dm4_5', 'dm5_8', 'dm8_0',
                       'dm24', 'dmJ', 'dmH', 'dmKs',
                       'Type', 'Phase', 'f_Type', 'alpha', 'M', 'W'
                       ], na_values = 'null')
```

`sep` defines the field delimiter in the text data. Vizier data uses `,` as the field delimiter, whereas data downloaded from Gator uses a variable number of spaces. In this case, use `sep = '\s+'`.

- Write into sqlite3

```
con = sql.connect('/home/kecai/w49/w49.db')
gutermuth.to_sql("gtw49_original", con, if_exists = 'replace', index = False)
con.commit()
```

- Inside database xxx.db, this creates a table by the name `xxx_original`. This table is only used as the data source for the master table and is not meant to be altered.

1.3. Prepare Master Table

Master table will be used to store Phase, Type and Flag values of all relevant objects. Results of each phase will be updated to this table.

- Create new table with rows from original table, computed colors and three empty columns: Phase, Type and f_Type (flag)

```
con = sql.connect('/home/kecai/w49/w49test.db')
cur = con.cursor()
cur.execute('''CREATE TABLE gtw49 AS
               SELECT desig, ra, dec, m3_6, m4_5, m5_8, m8_0, m24, mJ, mH, mKs,
               dm3_6, dm4_5, dm5_8, dm8_0, dm24, dmJ, dmH, dmKs,
               m3_6 - m4_5 AS c3645,
               m3_6 - m5_8 AS c3658,
               m3_6 - m8_0 AS c3680,
               m4_5 - m5_8 AS c4558,
               m4_5 - m8_0 AS c4580,
               m5_8 - m8_0 AS c5880,
               mH - mKs AS cHK,
               mJ - mH AS cJH,
               mKs - m3_6 AS cK36,
               m3_6 - m24 AS c3624,
               m4_5 - m24 AS c4524,
               m5_8 - m24 AS c5824,
               m8_0 - m24 AS c8024,
               Type AS oType,
               Phase AS oPhase,
               f_Type AS of_Type,
               NULL AS Type,
               NULL AS Phase,
               NULL AS f_Type FROM gtw49_original''' )
cur.execute('''CREATE INDEX gtw49_desig ON gtw49 (desig)''') ①
```

① Create an index on field `desig` because this field will be queried frequently for updating Phase, Type and Flag values.

2. Phase I

2.1. Initialize with `init`

Phase 1 is applied to all data with photometric uncertainties σ < 0.2 mag detections in all four IRAC bands.

`init` Extract data that is going to be used for Phase 1.

```
def init(con, source):
    cur = con.cursor()
    target = source + '_p1'
    cur.execute('''CREATE TABLE ''' + target + ''' AS
                   SELECT desig, m3_6, m4_5, m5_8, m8_0,
                                 dm3_6, dm4_5, dm5_8, dm8_0,
                                 c3645, c3658, c3680, c4558, c4580, c5880,
                                 Type FROM ''' + source +
                 ''' WHERE (m5_8 IS NOT NULL) AND
                           (m8_0 IS NOT NULL) AND
                           dm3_6 < ? AND
                           dm4_5 < ? AND
                           dm5_8 < ? AND
                           dm8_0 < ?''', [0.2]*4)
    return target
```

Function takes two arguments:

- `con`: connection to database
- `source`: name of the source table, in this case, the name of the master table.

Function returns the name of the table created. All data used for Phase 1 comes from this table.

2.2. Extract PAH galaxies: `extract_19`

Extract PAH galaxies (Type = 19 per Gutermuth's convention).

Extract `desig` of objects that satisfy PAH Criterion 1 **or** PAH Criterion 2.

- PAH Criterion 1:

$$\begin{aligned}
[4.5]-[5.8] &< \frac{1.05}{1.2} \times ([5.8]-[8.0]-1) \\
[4.5]-[5.8] &< 1.05 \\
[5.8]-[8.0] &> 1 \\
[4.5] &> 11.5
\end{aligned}$$

- PAH Criterion 2:

$$[3.6]-[5.8]<\frac{1.5}{2}\times([4.5]-[8.0]-1)$$
$$[3.6]-[5.8]<1.5$$
$$[4.5]-[8.0]>1$$
$$[4.5]>11.5$$

```
def extract_19(con, source):
    cur = con.cursor()
    #CREATE TABLE containing PAH galaxies
    target = source + '_19'
    cur.execute('''CREATE TABLE '''+ target + ''' AS
                SELECT desig FROM ''' + source + ''' WHERE Type IS NULL AND
                ((c4558  < (1.05/1.2) * (c5880 - 1)) AND
                 (c4558 < 1.05) AND
                 (c5880 > 1) AND
                 (m4_5 > 11.5))
                 OR
                ((c3658 < (1.5/2) * (c4580 -1)) AND
                 (c3658 < 1.5) AND
                 (c4580 > 1) AND
                 (m4_5 > 11.5))''')
    return target
```

Function takes two arguments:

- `con`: connection to database
- `source`: name of the source table, in this case, the name of the master table.

Function returns the name of the table containing extracted object `desig`. This table will be used to update object types in the master table.

2.3. Extract AGNs: `extract_29`

Extract AGNs (Type = 29 by Gutermuth's convention). Extract objects that satisfy:

ALL of the following:

$$[4.5]-[8.0]>0.5$$
$$[4.5]>13.5+([4.5]-[8.0]-2.3)/0.4$$
$$[4.5]>13.5$$

AND ONE of the following:

$$[4.5]>14+([4.5]-[8.0]-0.5)$$
$$[4.5]>14.5-([4.5]-[8.0]-1.2)/0.3$$
$$[4.5]>14.5$$

```
def extract_29(con, source):
    cur = con.cursor()
    #CREATE TABLE containing AGN
    target = source + '_29'
    cur.execute('''CREATE TABLE '''+ target + ''' AS
                   SELECT desig FROM ''' + source + ''' WHERE Type IS NULL AND
                  (c4580 > 0.5 AND m4_5 > 13.5 +(c4580 - 2.3)/0.4 AND m4_5 > 13.5)
                  AND
                  (m4_5 > 14 + (c4580 - 0.5) OR m4_5 > 14.5 - (c4580 - 1.2)/0.3 OR
m4_5 > 14.5)''')
    return target
```

Function returns the name of the table containing extracted object `desig`. This table will be used to update object types in the master table.

2.4. Extract Shock Gass Emission: `extract_9`

Extract shock gas emission (Type = 9). Extract objects that satisfy **ALL** of the following:

$$[3.6] - [4.5] > \frac{1.2}{0.55} \times (([4.5] - [5.8]) - 0.3) + 0.8$$
$$[4.5] - [5.8] \leq 0.85$$
$$[3.6] - [4.5] > 1.05$$

```
def extract_9(con, source):
    cur = con.cursor()
    #CREATE TABLE containing shock gas emission
    target = source + '_9'
    cur.execute('''CREATE TABLE '''+ target + ''' AS
                   SELECT desig FROM '''+ source + ''' WHERE Type IS NULL AND
                   (c3645 > (1.2/0.55)*(c4558 - 0.3)+0.8) AND
                   (c4558 <= 0.85) AND
                   (c3645 > 1.05)''')
    return target
```

Function returns the name of the table containing extracted object `desig`. This table will be used to update object types in the master table.

2.5. Extract PAH contaminated sources: `extract_20`

Extract PAH contaminated sources (Type = 20) Extract objects that satisfy BOTH of the following:

$$\sigma_1 := \sigma[[4.5] - [5.8]]$$
$$\sigma_2 := \sigma[[3.6] - [4.5]]$$
$$[3.6] - [4.5] - \sigma_2 \leq 1.4 \times (([4.5] - [5.8]) + \sigma_1 - 0.7) + 0.15$$
$$[3.6] - [4.5] - \sigma_2 \leq 1.65$$

```
def extract_20(con, source):
    #CREATE TABLE containing pah contaminated apertures
    cur = con.cursor()
    target = source + '_20'
    cur.execute('''CREATE TABLE '''+ target +''' AS
                SELECT desig FROM '''+ source +''' WHERE Type IS NULL AND
                c3645 - SQRT(SQUARE(dm3_6) + SQUARE(dm4_5)) <=
                  1.4 * (c4558 + SQRT(SQUARE(dm4_5) + SQUARE(dm5_8)) - 0.7) + 0.15
AND
                c3645 - SQRT(SQUARE(dm3_6) + SQUARE(dm4_5)) <= 1.65''')
    return target
```

Function returns the name of the table containing extracted object `desig`. This table will be used to update object types in the master table.

2.6. Extract Class I: `extract_1`

Extract Class I objects

Extract objects that satisfy **BOTH** of the following:

$$[4.5]-[5.8]>0.7$$
$$[3.6]-[4.5]>0.7$$

```
def extract_1(con, source):
    cur = con.cursor()
    target = source + '_1'
    cur.execute('''CREATE TABLE '''+ target +''' AS
                SELECT desig FROM '''+ source +''' WHERE
                Type IS NULL AND
                c4558 > 0.7 AND
                c3645 > 0.7''')
    return target
```

Function returns the name of the table containing extracted object `desig`. This table will be used to update object types in the master table.

2.7. Extract Class II: `extract_2`

Extract Class II objects

Extract objects that satisfy ALL of the following:

$$
\begin{aligned}
&\sigma_3 := \sigma[[4.5]-[8.0]] \\
&\sigma_4 := \sigma[[3.6]-[5.8]] \\
&[4.5]-[8.0]-\sigma_3 > 0.5 \\
&[3.6]-[5.8]-\sigma_4 > 0.35 \\
&[3.6]-[5.8]+\sigma_4 \le \frac{0.14}{0.04} \times ([4.5]-[8.0]-\sigma_3-0.5)+0.5 \\
&[3.6]-[4.5]-\sigma_2 > 0.15.
\end{aligned}
$$

```
def extract_2(con, source):
    cur = con.cursor()
    target = source + '_2'
    cur.execute('''CREATE TABLE '''+ target +''' AS
                  SELECT desig FROM '''+ source +''' WHERE Type IS NULL AND
                  c4580 - SQRT(SQUARE(dm4_5) + SQUARE(dm8_0)) > 0.5 AND
                  c3658 - SQRT(SQUARE(dm3_6) + SQUARE(dm5_8)) > 0.35 AND
                  c3658 + SQRT(SQUARE(dm3_6) + SQUARE(dm5_8))
                    <= (0.14/0.04)*(c4580 - SQRT(SQUARE(dm4_5) + SQUARE(dm8_0))-0.5)
+ 0.5 AND
                  c3645 - SQRT(SQUARE(dm3_6) + SQUARE(dm4_5)) > 0.15''')
    return target
```

2.8. Run Phase 1

run_phase1 takes two arguments: * master: name of the master table created in data preparation * constr: a connection string to the database.

run_phase1 initializes a working copy of Phase 1 relevant data, sequentially extracts different types of objects and updates their Type in the master table.

```python
def run_phase1(master, constr):
    con = sql.connect(constr)

    #add math udf to sqlite
    cu.add_math(con) ①
    #initialize and create a working copy for phase1 containing desig, four IRAC
channel mags and Type
    wc = init(con, master)  ②
    cu.update_phase(con, master, wc, 1) ③

    #extract and update PAH in the working copy
    c19 = extract_19(con, wc)  ④
    cu.update_type(con, wc, c19, 19) ⑤
    cu.update_type(con, master, c19, 19) ⑥

    #extract and update AGN
    c29 = extract_29(con, wc)
    cu.update_type(con, wc, c29, 29)
    cu.update_type(con, master, c29, 29)

    c9 = extract_9(con, wc)
    cu.update_type(con, wc, c9, 9)
    cu.update_type(con, master, c9, 9)

    c20 = extract_20(con, wc)
    cu.update_type(con, wc, c20, 20)
    cu.update_type(con, master, c20, 20)

    c1 = extract_1(con, wc)
    cu.update_type(con, wc, c1, 1)
    cu.update_type(con, master, c1, 1)

    c2 = extract_2(con, wc)
    cu.update_type(con, wc, c2, 2)
    cu.update_type(con, master, c2, 2)

    return 0
```

① Sqlite3 does not contain `SQRT` and `SQUARE` functions. They are defined in the `cu` module and need to be added to SQLite3 before running SQL containing them.

② `wc` is the initialized working copy.

③ Update phase.

④ Extract objects of a certain type.

⑤ Use extracted data to update `Type` in working copy.

⑥ Use extracted data to update `Type` in master table.

To use this function:

```
run_phase1('gtw49', '/home/kecai/w49/w49.db')
```

3. Phase II

3.1. Initialize with `init`

Intialize data used for Phase 2. Phase 2 is applied to sources that lack detections at [5.8] or [8.0], but have high quality ($\sigma < 0.1$ mag) 2MASS bands (H and Ks are required at minimum, J is used where present).

```
def init(con, source):
    cur = con.cursor()
    target = source + '_p2'
    cur.execute('''CREATE TABLE '''+ target + ''' AS
                SELECT desig,
                m3_6, m4_5, mJ, mH, mKs, dm3_6, dm4_5, dmJ, dmH, dmKs,
                c3645, cHK, cJH, cK36
                FROM '''
                + source + ''' WHERE
                (m5_8 IS NULL OR m8_0 IS NULL) AND
                (dmH < ? AND dmKs < ? AND (dmJ < ? OR mJ IS NULL)) AND
                dm3_6 < ? AND
                dm4_5 < ?''', [0.1, 0.1, 0.1, 0.2, 0.2])
                             #Relax the condition on dm3_6, dm4_5? not explicit in
paper
    return target
```

3.2. Color Excess Ratios for Computing Intrinsic Colors

Previously determined color excess ratios will be used to compute intrinsic colors:

$$\frac{E_{J-H}}{E_{H-K}} = 1.73$$

$$\frac{E_{H-K}}{E_{K-[3.6]}} = 1.49$$

$$\frac{E_{H-K}}{E_{K-[4.5]}} = 1.17$$

$$\frac{E_{[3.6]-[4.5]}}{E_{H-K}} = [\frac{E_{H-K}}{E_{K-[4.5]}}]^{-1} - [\frac{E_{H-K}}{E_{K-[3.6]}}]^{-1} = 0.184$$

3.3. Compute Intrinsic [H-K] color with `compute_icHK`

For objects that **have** J detections:

$$
\begin{aligned}
&[J-H]_0 = 0.58 \times [H-K]_0 + 0.52;\ \text{for}[H-K]_0 > 0.14\\
&[J-H]_0 = 0.6;\ \text{for}[H-K]_0 \leq 0.14\\
&[H-K]_0 = [H-K]_m - ([J-H]_m - [J-H]_0) \times \frac{E_{H-K}}{E_{J-H}}\\
&[H-K]_0 = \frac{[J-H]_m - \frac{E_{J-H}}{E_{H-K}} \times [H-K]_m - 0.52}{0.58 - \frac{E_{J-H}}{E_{H-K}}}
\end{aligned}
$$

Doing some algebra, we get the following computation friendly expressions:

$$
\begin{aligned}
&[H-K]_0 = 1.5[H-K]_m - 0.87[J-H]_m + 0.45;\ \text{if}[H-K]_m > -0.21 + 0.58[J-H]_m\\
&[H-K]_0 = [H-K]_m - 0.58[J-H]_m + 0.35;\ \text{if}[H-K]_m \leq -0.21 + 0.58[J-H]_m
\end{aligned}
$$

For objects that **lack** J detections:

$$
\begin{aligned}
&[H-K]_0 = 1.33 \times [[3.6]-[4.5]]_0 + 0.133; && \text{for}[[3.6]-[4.5]]_0 > 0.06\\
&[H-K]_0 = 0.2; && \text{for}[[3.6]-[4.5]]_0 \leq 0.06\\
&[H-K]_0 = [H-K]_m - \frac{E_{[3.6]-[4.5]}}{E_{H-K}}^{-1}([[3.6-[4.5]]_m - [[3.6]-[4.5]]_0) &&\\
&[H-K]_0 = \frac{1.33 \times (\frac{E_{[3.6]-[4.5]}}{E_{H-K}}[H-K]_m - [[3.6]-[4.5]]_m) - 0.133}{1.33\frac{E_{[3.6]-[4.5]}}{E_{H-K}} - 1} &&
\end{aligned}
$$

Doing some algebra, we get the following computation friendly expressions:

$$
\begin{aligned}
&[H-K]_0 = 1.76 \times [[3.6]-[4.5]]_m - 0.32[H-K]_m + 0.176;\ \text{if}[H-K]_m < -0.11 + 5.4[[3.6]-[4.5]]_m\\
&[H-K]_0 = 0.2;\ \text{if}[H-K]_m \geq -0.13 + 5.4[[3.6]-[4.5]]_m
\end{aligned}
$$

Note that these conditions have overlap.

```
def compute_icHK(con, source):
    cur = con.cursor()
    target = source + '_icHK'
    cur.execute('''CREATE TABLE '''+ target + ''' AS SELECT *,
               CASE
               WHEN (mJ IS NOT NULL) AND cHK > 0.58 * cJH - 0.21
                 THEN 1.5 * cHK - 0.87 * cJH + 0.45
               WHEN (mJ IS NOT NULL) AND cHK <= 0.58 * cJH - 0.21
                 THEN cHK - 0.58 * cJH + 0.35
               WHEN mJ IS NULL AND cHK < -0.12 + 5.4 * c3645
                 THEN 1.76 * c3645 - 0.32 * cHK + 0.176
               WHEN mJ IS NULL AND cHK >= -0.12 + 5.4 * c3645
                 THEN 0.2
               END AS icH_K
               FROM '''+ source)
    return target
```

3.4. Compute Intrinsic [K - [3.6]] and [[3.6] - [4.5]] Colors with `compute_icK36_ic3645`

$$[K-[3.6]]_0 = [K-[3.6]]_m - ([H-K]_m - [H-K]_0)\frac{E_{K-[3.6]}}{E_{H-K}}$$
$$[[3.6]-[4.5]]_0 = [[3.6]-[4.5]]_m - ([H-K]_m - [H-K]_0)\frac{E_{[3.6]-[4.5]}}{E_{H-K}}$$

```
def compute_icK36_ic3645(con, source):
    cur = con.cursor()
    target = source + '_icK36_ic3645'
    cur.execute('''CREATE TABLE '''+ target + ''' AS SELECT *,
                cK36 - (cHK - icH_K) * 0.671 AS icK_36,
                c3645 - (cHK - icH_K) * 0.184 AS ic36_45 FROM '''+ source)
    return target
```

3.5. Extract Addtional YSO's with `extract_yso`

Identify additional YSO's as those sources whose intrinsic colors obey all of the following:

$$\sigma_2 := \sigma\{[[3.6]-[4.5]]_m\}$$
$$\sigma_5 := \sigma\{[[K]-[3.6]]_m\}$$
$$[[3.6]-[4.5]]_0 - \sigma_2 > 0.101$$
$$[K-[3.6]]_0 - \sigma_5 > 0$$
$$[K-[3.6]]_0 - \sigma_5 > -2.85714 \times ([[3.6]-[4.5]]_0 - \sigma_2 - 0.101) + 0.5$$

```
def extract_yso(con, source):
    cur = con.cursor()
    target = source + '_yso'
    cur.execute('''CREATE TABLE '''+ target + ''' AS SELECT * FROM '''+ source + '''
                WHERE
                ic36_45 - SQRT(SQUARE(dm3_6) + SQUARE(dm4_5)) > 0.101 AND
                icK_36 - SQRT(SQUARE(dmKs) + SQUARE(dm3_6)) > 0 AND
                icK_36 - SQRT(SQUARE(dmKs) + SQUARE(dm3_6)) > -2.85714 * (ic36_45 -
SQRT(SQUARE(dm3_6) + SQUARE(dm4_5)) - 0.101) + 0.5''' )
    return target
```

3.6. Extract Class I from Additional YSO's with `extract_1`

From the table created by `extract_yso`, identify as Class I objects that satisfy:

$$[K-[3.6]]_0 - \sigma_5 > -2.85714 \times ([[3.6]-[4.5]]_0 - \sigma_2 - 0.401]) + 1.7$$
$$[3.6] < \zeta_{361}$$

where ζ_{361} is a distance-dependent brightness cutoff for Class I objects to minimize inclusion of dim extraglactic contaminants.

```
def extract_1(con, source, cutoff361):
    cur = con.cursor()
    target = source + '_1'
    cur.execute('''CREATE TABLE '''+ target +''' AS SELECT desig FROM '''+ source +'''
                   WHERE
                   ic36_45 - SQRT(SQUARE(dm3_6) + SQUARE(dm4_5)) > -2.85714 * (ic36_45
- SQRT(SQUARE(dm3_6) + SQUARE(dm4_5)) - 0.401) + 1.7
                   AND
                   m3_6 < ?''', [cutoff361])
    return target
```

3.7. Extract Class II from Additional YSO's with `extract_2`

All objects that were produced by `extract_yso` but not Class I are considered Class II. To minimize inclusion of extraglactic contaminants, apply distance-dependent [3.6] brightness cutoff:

$$[3.6]<\zeta_{362}$$

```
def extract_2(con, yso, c1, cutoff362):
    cur = con.cursor()
    target = yso+'_2'
    cur.execute('''CREATE TABLE '''+ target +''' AS SELECT desig FROM '''+ yso +'''
    WHERE
    m3_6 < ?
    AND
    desig NOT IN (SELECT desig FROM '''+ c1 +''')''', [cutoff362])
    return target
```

3.8. Running Phase 2 with `run_phase2`

```python
def run_phase2(master, constr, cutoff361, cutoff362):
    con = sql.connect(constr)

    #add math udf to sqlite
    cu.add_math(con)

    #initialize a working copy containing relevant data
    wc = init(con, master)
    cu.update_phase(con, master, wc, 2)

    #compute three intrinsic colors
    icHK = compute_icHK(con, wc)
    icK36_ic3645 = compute_icK36_ic3645(con, icHK)

    yso = extract_yso(con, icK36_ic3645)

    c1 = extract_1(con, yso, cutoff361)
    cu.update_type(con, master, c1, 1)

    c2 = extract_2(con, yso, c1, cutoff362)
    cu.update_type(con, master, c2, 2)

    return 0
```

To run `run_phase2`, do

```python
run_phase2('gtw49', '/home/kecai/w49/w49.db', 20, 20)
```

4. Phase III

Adding and checking YSOs with MIPS 24 micron photometry.

4.1. Initialize Data with `init`

Extract data with MIPS 24 micron detection with σ < 0.1 mag.

```
def init(con, source):
    cur = con.cursor()
    target = source + '_p3'
    cur.execute('''CREATE TABLE '''+ target +''' AS SELECT
                   desig,
                   m3_6, m4_5, m5_8, m8_0, m24,
                   c3624, c4524, c5824, c8024, c3658,
                   Type
                   FROM '''+ source +''' WHERE
                   (m24 IS NOT NULL) AND
                   dm24 < 0.1 AND
                   (dm3_6 < 0.2 OR m3_6 IS NULL) AND
                   (dm4_5 < 0.2 OR m4_5 IS NULL) AND
                   (dm5_8 < 0.2 OR m5_8 IS NULL) AND
                   (dm8_0 < 0.2 OR m8_0 IS NULL)''')
    return target
```

4.2. Extract Transition Disks with `extract_3`

Sources that were considered photospheric (Class I or Class II) in Phase 1 or Phase 2 are identified as "transition disks" if they satisfy ONE of the following:

$$[5.8]-[24]>2.5$$
$$[4.5]-[24]>2.5$$

Also apply a distance-dependent brightness cutoff ζ_{36} to [3.6] to minimize inclusion of extragalactic contaminants:

$$[3.6]<\zeta_{36}$$

```
def extract_3(con, source, cutoff36):
    #flag what were considered photospheric to transition disks
    cur = con.cursor()
    target = source + '_3'
    cur.execute('''CREATE TABLE '''+ target +''' AS SELECT
                   desig FROM '''+ source +''' WHERE
                   (Type = 1 OR 2)AND
                   (c5824 > 2.5 OR c4524 > 2.5)
                   AND m3_6 < ?
                   ''', [cutoff36])
    return target
```

4.3. Extract Class I*

Identify sources as deeply embedded protostar if they lack detections in some IRAC bands but satisfy BOTH of the following (bright MIPS photometry):

$$[24] < \zeta_{24}$$
$$[X] - [24] > 4.5$$

where ζ_{24} is the distant-dependent brightness cutoff to mitigate extragalactic contamination and [X] is the photometry for the longest wavelength IRAC detection available in the catalog.

```
def extract_18(con, source, cutoff24):
    cur = con.cursor()
    target = source + '_18'
    cur.execute('''CREATE TABLE '''+ target + ''' AS
                     SELECT desig FROM (SELECT desig,
                      CASE
                      WHEN m8_0 IS NOT NULL THEN c8024
                      WHEN (m8_0 IS NULL) AND (m5_8 IS NOT NULL) THEN c5824
                      WHEN (m8_0 IS NULL) AND (m5_8 IS NULL) AND (m4_5 IS NOT NULL)
THEN c4524
                      WHEN (m8_0 IS NULL) AND (m5_8 IS NULL) AND (m4_5 IS NULL) AND
(m3_6 IS NOT NULL) THEN c3624
                      END AS cX_24
                     FROM ''' + source + ''' WHERE m24 < ?)
                    WHERE cX_24 > 4.5''', [cutoff24])
    return target
```

4.4. Reinclude Protostars (Class I)

Reinclude sources as protostars (Class I) that were previously identified as AGNs and shock emissions if they satisfy ALL of the following:

$$[24] < \zeta_{24}$$
$$[3.6] - [5.8] > 0.5$$
$$[4.5] - [24] > 4.5$$
$$[8.0] - [24] > 4$$

where ζ_{24} is a distance-dependent brightness cutoff for 24 micron to eliminate extragalactic contaminants.

```
def reflag_299_to_1(con, source, cutoff24):
    cur = con.cursor()
    target = source + '_1_from_29_9'
    cur.execute('''CREATE TABLE '''+ target +''' AS SELECT desig FROM '''+ source +'''
WHERE
                   (Type = 29 OR Type = 9) AND
                   m24 > ? AND
                   c3658 > 0.5 AND
                   c4524 > 4.5 AND
                   c8024 > 4''', [cutoff24])
    return target
```

4.5. Re-identify Protostars (Class I) as Class II

All previously identified protostars (Class I) that have 24 micron detections are identified as Class II if they satisfy EITHER of the following:

$$[5.8] - [24] \leq 4; \text{ if possess 5.8 micron}$$
$$[4.5] - [24] \leq 4; \text{ if do not possess 5.8 micron}$$

```
def reflag_1_to_2(con, source):
    cur = con.cursor()
    target = source+'_2_from_1'
    cur.execute('''CREATE TABLE '''+ target +''' AS SELECT desig FROM '''+source+'''
WHERE
                   Type = 1 AND
                   (m5_8 IS NOT NULL AND c5824 <=4) OR
                   (m5_8 IS NULL AND c4524 <=4)''')
    return target
```

4.6. Run Phase 3

```python
def run_phase3(master, constr, cutoff36, cutoff24):
    con = sql.connect(constr)
    cur = con.cursor()

    wc = init(con, master)
    cu.update_phase(con, master, wc, 0) ①

    c3 = extract_3(con, wc, cutoff36)
    cu.update_type(con, master, c3, 3)
    cu.update_type(con, wc, c3, 3)

    #extract ClassI*
    c18 = extract_18(con, wc, cutoff24)
    cu.update_type(con, master, c18, 1)
    cu.update_type(con, wc, c18, 1)

    #reflag AGN and shock emissions as classI
    c1f299 = reflag_299_to_1(con, wc, cutoff24) ②
    cu.update_type(con, master, c1f299, 1)
    cu.update_type(con, wc, c1f299, 1)

    #reflag previous class I to classII
    c2f1 = reflag_1_to_2(con, wc)
    cu.update_type(con, master, c2f1, 2)
    cu.update_type(con, wc, c2f1, 2)

    return 0
```

① We follow Gutermuth's Phase labeling convention. Objects that are identified/updated in Phase III have value `0` in their `Phase` field.

② Reflagged sources are extracted into a new table and then used to update Type in the master table.

To use `run_phase3`, do

```python
run_phase3('gtw49', '/home/kecai/w49/w49.db', 20, 20)
```

5. Phase AGB

If the star formation region is very far away (e.g. W49), possible higher level of contamination should be taken into account. Phase AGB relabel previously identified YSOs as AGB stars.

5.1. Initialize Data

Extract objects that are previously identified as Class I and II:

```
def init(con, source):
    cur = con.cursor()
    target = source+'_pa'
    cur.execute('''CREATE TABLE '''+ target +''' AS
                   SELECT desig, m3_6, m4_5, m5_8, m8_0,
                   c3645, c3680,
                   Phase, Type, f_Type
                   FROM '''+ source +''' WHERE Type = 1 OR Type = 2''')
    return target
```

5.2. Extract AGB stars: `extract_12`

Identify as AGBs if objects satisfy EITHER of the following two pairs of conditions:

$$3 < [3.6] < 9.5 \text{and} 0.2 < [3.6] - [4.5] < 1.25$$
$$3.5 < [3.6] < 9.5 \text{and} 0.4 < [3.6] - [8.0] < 2.6$$

```
def extract_12(con, source):
    cur = con.cursor()
    target = source+'_12'
    cur.execute('''CREATE TABLE '''+ target +''' AS SELECT desig FROM '''+ source +'''
WHERE
                   (m3_6 > 3 AND
                    m3_6 < 9.5 AND
                    c3645 > 0.2 AND
                    c3645 < 1.25)
                   OR
                   (m3_6 > 3.5 AND
                    m3_6 < 9.5 AND
                    c3680 > 0.4 AND
                    c3680 < 2.6)''')
    return target
```

5.3. Flag as `uc`

Extragalactic and background contaminant elimination require 5.8 and 8.0 micron detections. For YSOs classified in Phase II that lack these detections, apply a distant dependent brightness cutoff to

[3.6]:

$$[3.6] \geq \zeta_{36}$$

Objects that satisfy the condition are extracted and will be used to update `f_Type` as `uc`:

```
def extract_uc(con, source, cutoff36):
    cur = con.cursor()
    target = source + '_uc'
    cur.execute('''CREATE TABLE '''+ target + ''' AS SELECT desig FROM '''+ source +
''' WHERE
                   Phase = 2 AND
                   (m5_8 IS NULL OR m8_0 IS NULL) AND
                   m3_6 > ?''', [cutoff36])
    return target
```

5.4. Run Phase AGB

```
def run_phaseagb(master, constr, cutoff36):
    con = sql.connect(constr)

    wc = init(con, master)
    cu.update_phase(con, master, wc, 3)

    c12 = extract_12(con, wc)
    cu.update_type(con, master, c12, 12)
    cu.update_type(con, wc, c12, 12)

    cuc = extract_uc(con, wc, cutoff36)
    cu.update_flag(con, master, cuc, 'uc')
    cu.update_flag(con, wc, cuc, 'uc')

    return 0
```

To use this function, do:

```
run_phaseagb('gtw49', '/home/kecai/w49/w49.db', 20)
```

6. Compute Spectral Index

Spectral index

$$\alpha = \frac{\partial \log(\lambda F_\lambda)}{\partial \log(\lambda)}$$

Compute values of

$$\frac{\log(\lambda_1 F_{\lambda_1}) - \log(\lambda_2 F_{\lambda_2})}{\log(\lambda_1) - \log(\lambda_2)}$$

between neighboring passbands for bands 2.16 micron (2MASS Ks), 3.6 micron, 4.5 micron, 5.8 micron, 8.0 micron and 24 micron.

If catalog only reports magnitudes, use the equivalent expression:

$$1 - \frac{0.4(m_1 - m_2)}{\log(\lambda_1 / \lambda_2)}$$

- α's | Bands | $\log(\lambda_2 / \lambda_1)$
- α_1 | 2.16 and 3.6 | $\log(2.16/3.6) = -0.222$
- α_2 | 3.6 and 4.5 | $\log(3.6/4.5) = -0.0969$
- α_3 | 4.5 and 5.8 | $\log(4.5/5.8) = -0.110$
- α_4 | 5.8 and 8.0 | $\log(5.8/8.0) = -0.140$
- α_5 | 8.0 and 24 | $\log(8.0/24) = -0.477$

```
def compute_alpha(constr, source):
    con = sql.connect(constr)
    cur = con.cursor()
    target = source+'_alpha'
    cur.execute('''CREATE TABLE '''+ target +''' AS
                   SELECT desig, Type, mKs, m3_6, m4_5, m5_8, m8_0, m24,
                   1 + 0.4 * cK36/0.222 AS alpha1,
                   1 + 0.4 * c3645/0.0969 AS alpha2,
                   1 + 0.4 * c4558/0.110 AS alpha3,
                   1 + 0.4 * c5880/0.140 AS alpha4,
                   1 + 0.4 * c8024/0.477 AS alpha5
                   FROM '''+ source +''' WHERE Type = 1 OR Type = 2''')
    return 0
```

To use this function, do

```
compute_alpha('/home/kecai/w49/w49test.db', 'gtw49')
```

7. Analysis

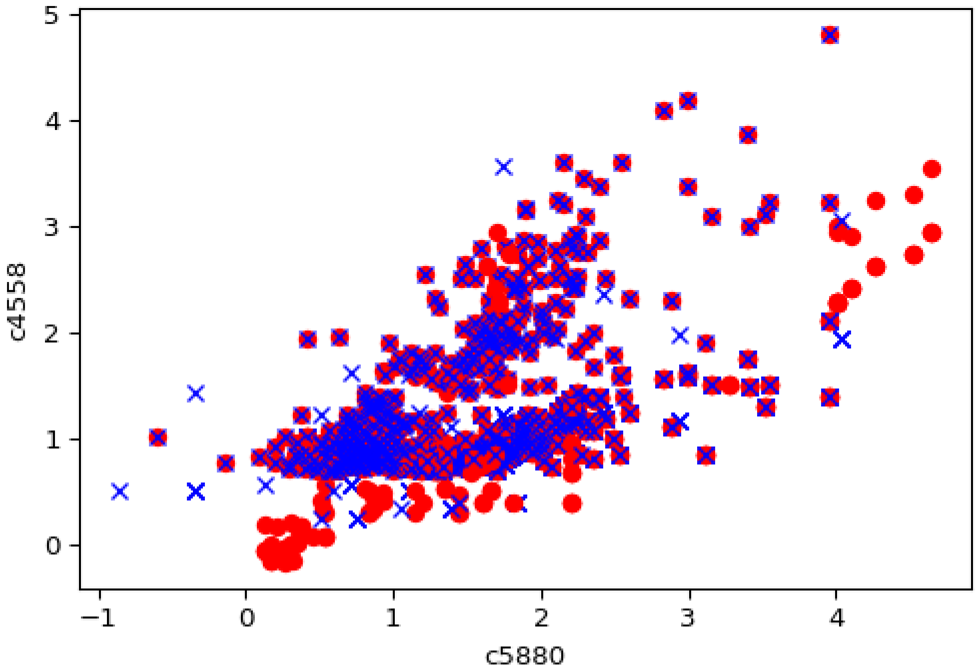
7.1. Possible Reasons of Difference

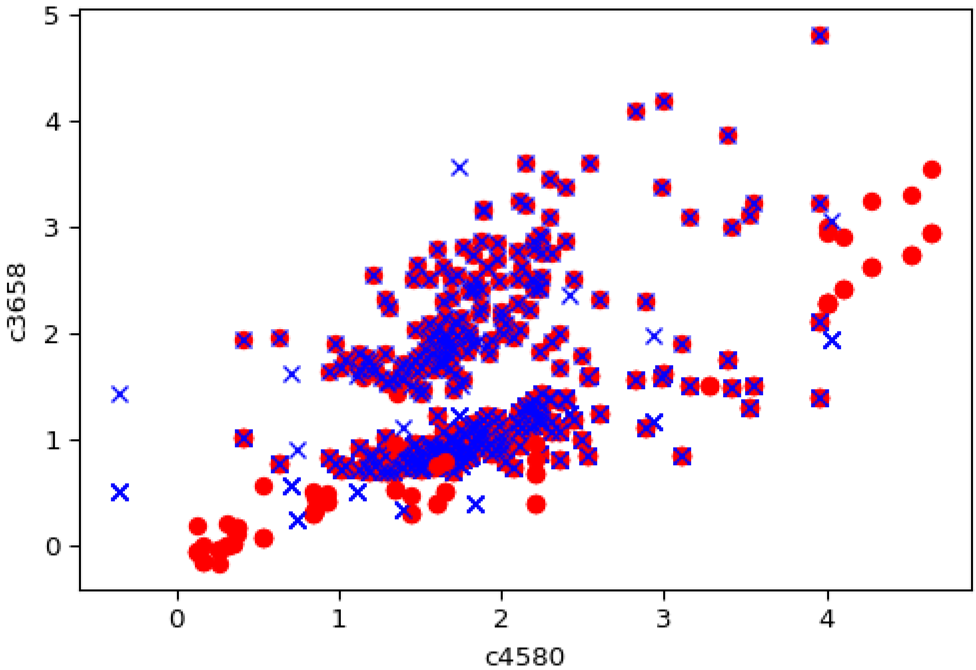
- Distance dependent brightness cutoff
- 1 Standard deviation fuzzying
- Class 0 classification

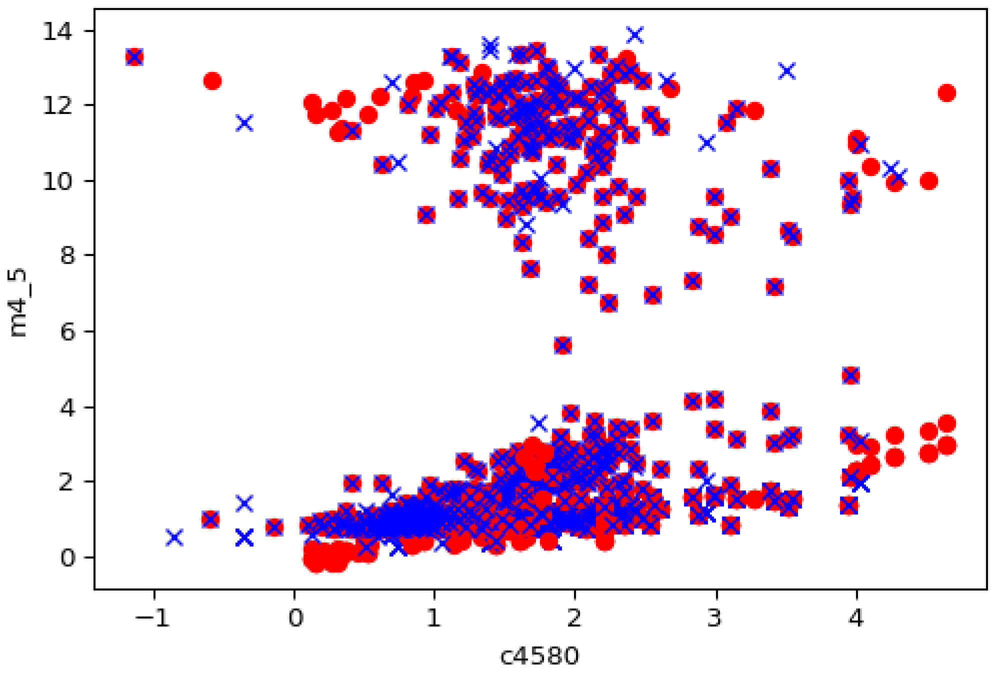
7.2. Type Counts

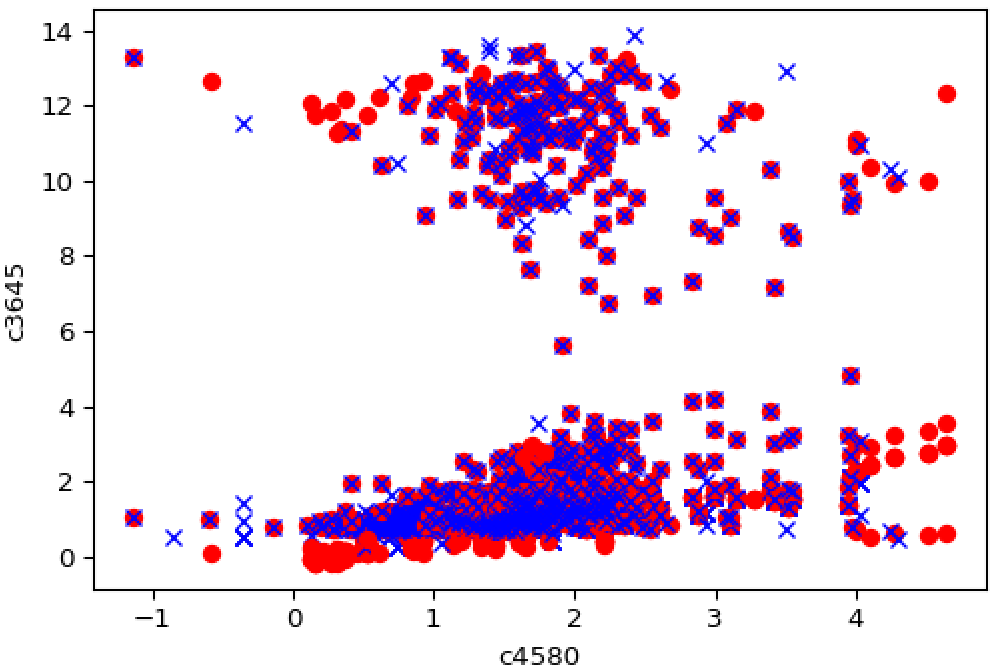
```
select Type, count(*) from gtw49 group by Type;
|328054
0|166
1|208
2|3054
3|72
9|11
12|198
19|29
20|646
29|4
select oType, count(*) from gtw49 group by oType;
-100|81097
0|46
1|250
2|2839
3|149
9|11
12|212
19|53
20|639
29|4
99|247142
```

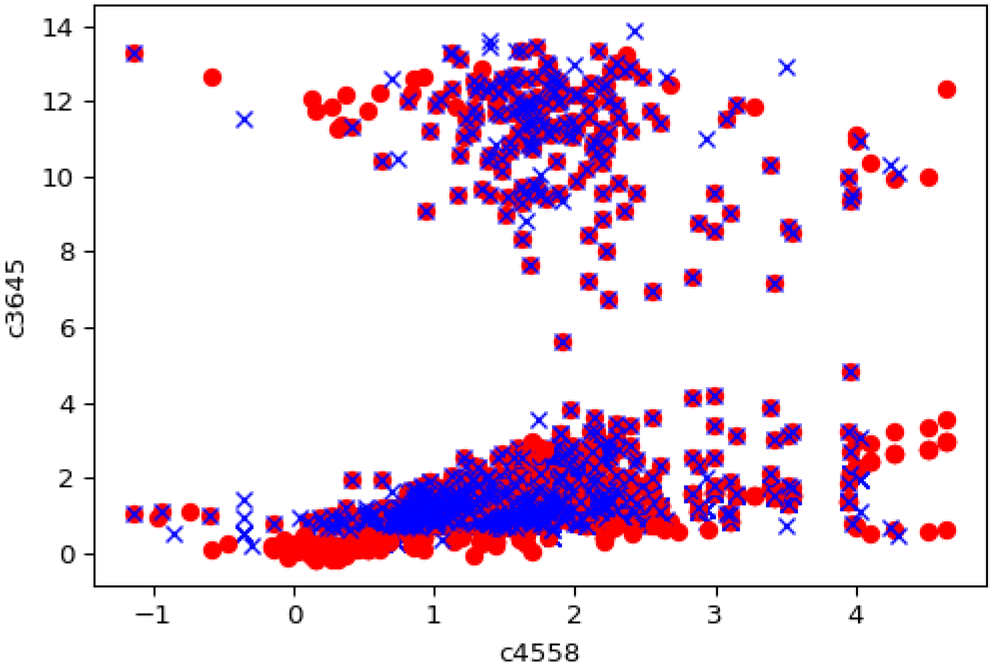
7.3. Class I

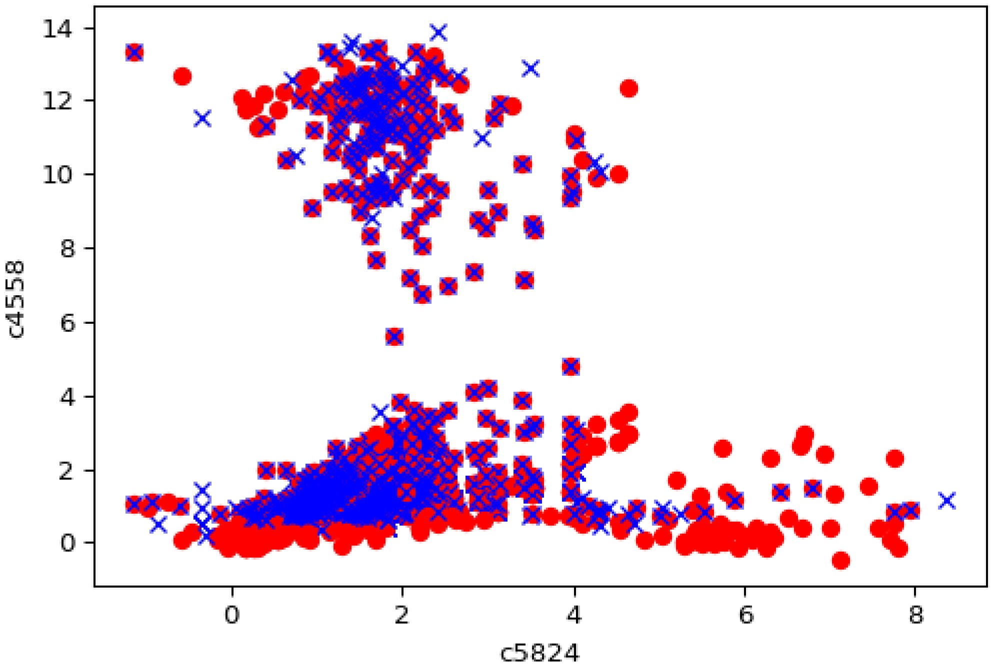

c4558
c5880

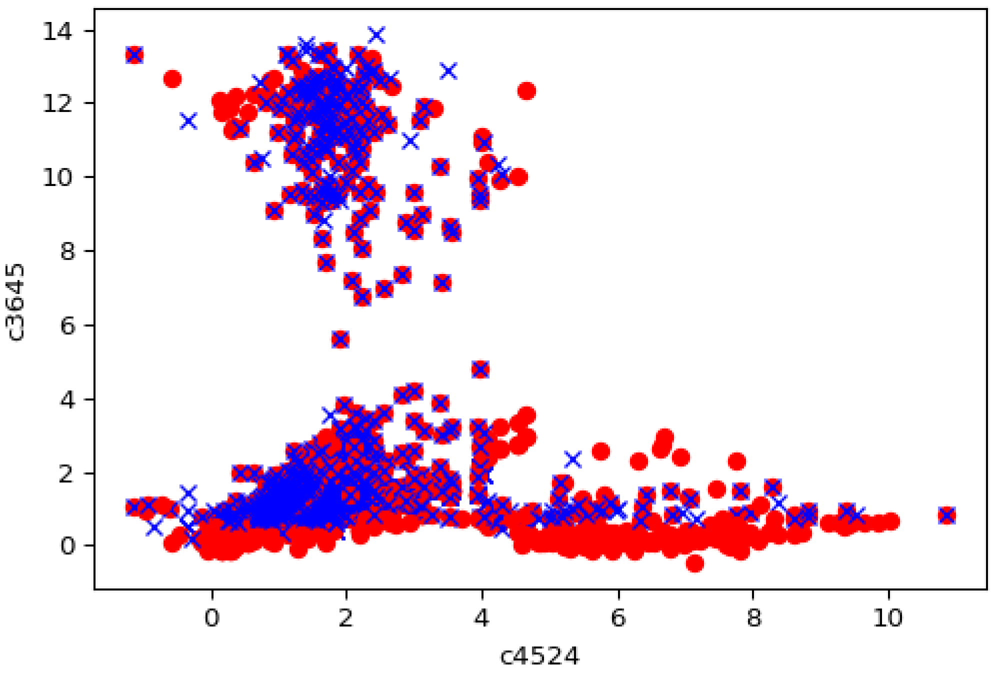

c3658
c4580


m4_5
c4580
14
12
10
8
6
4
2
0
−1
0
1
2
3
4

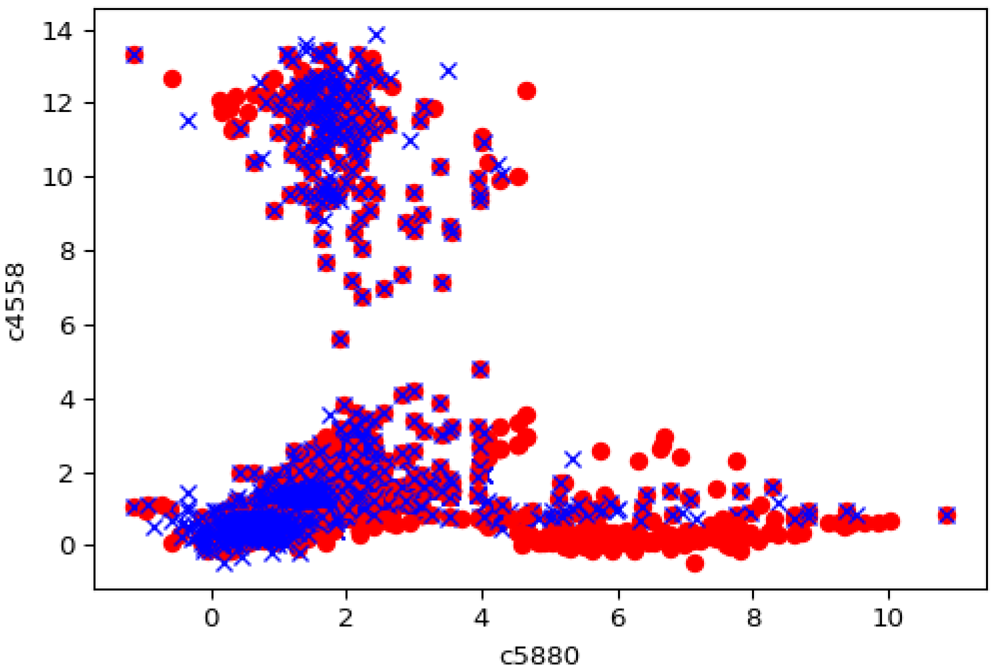

c3645
c4580
14
12
10
8
6
4
2
0
−1
0
1
2
3
4

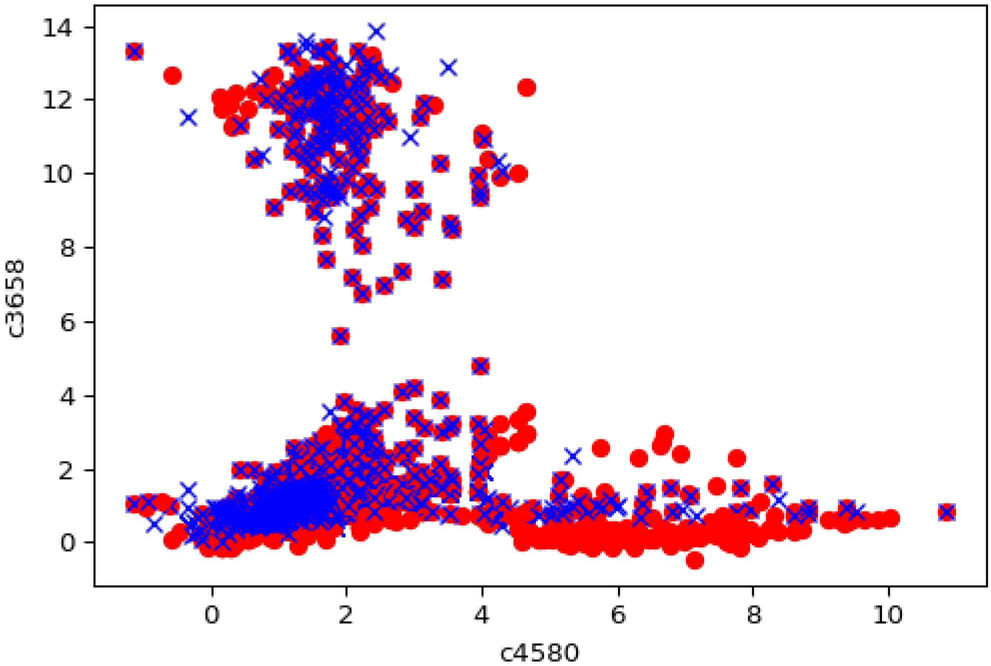

c3645
c4558
14
12
10
8
6
4
2
0
−1
0
1
2
3
4


c4558
c5824
14
12
10
8
6
4
2
0
0
2
4
6
8

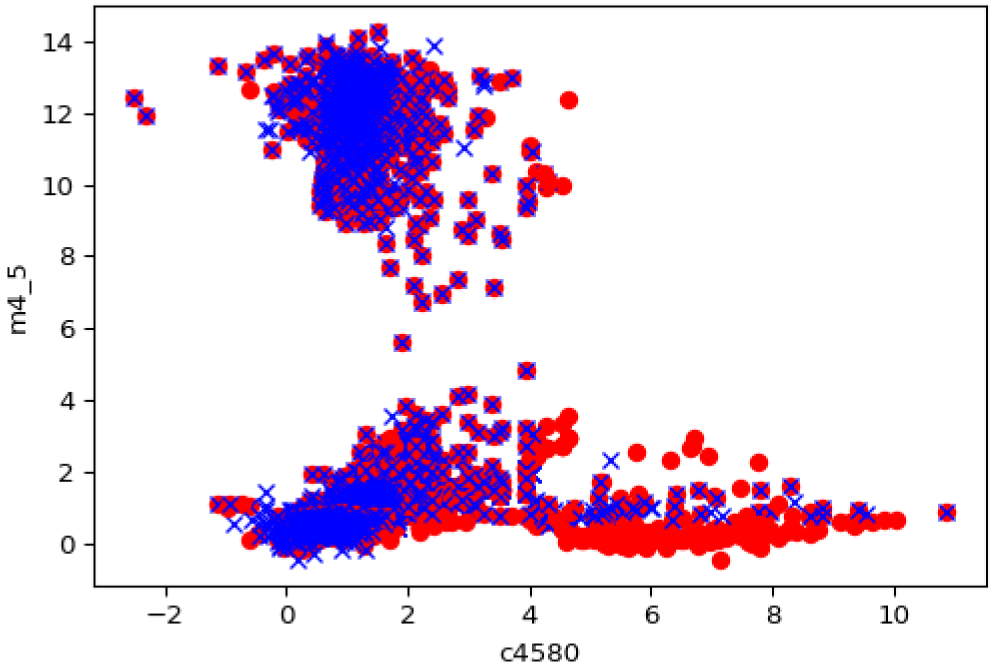


7.4. Class II

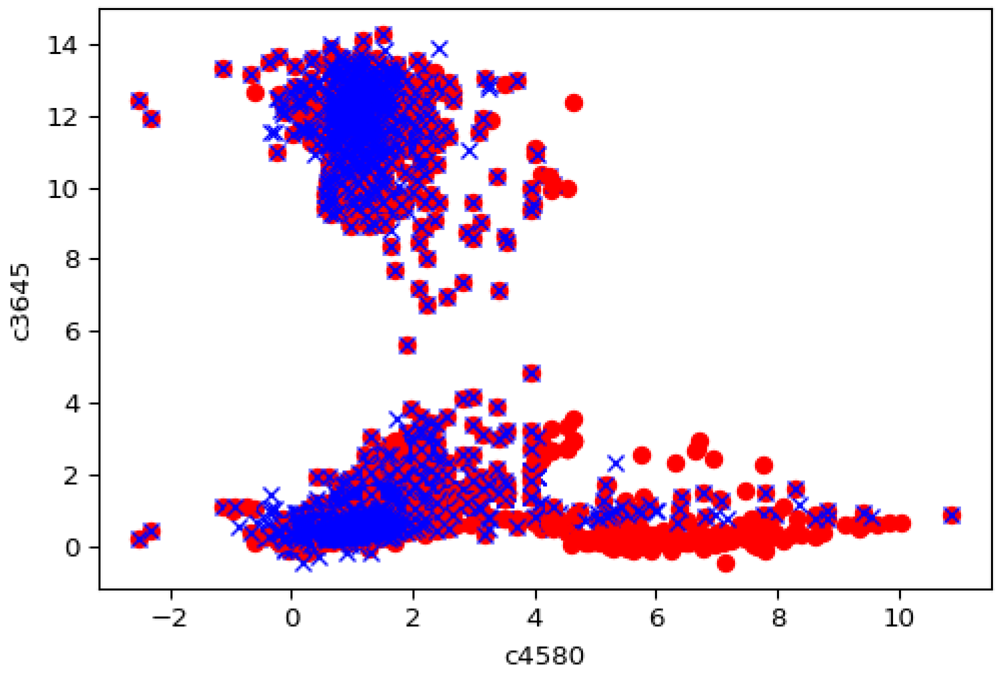




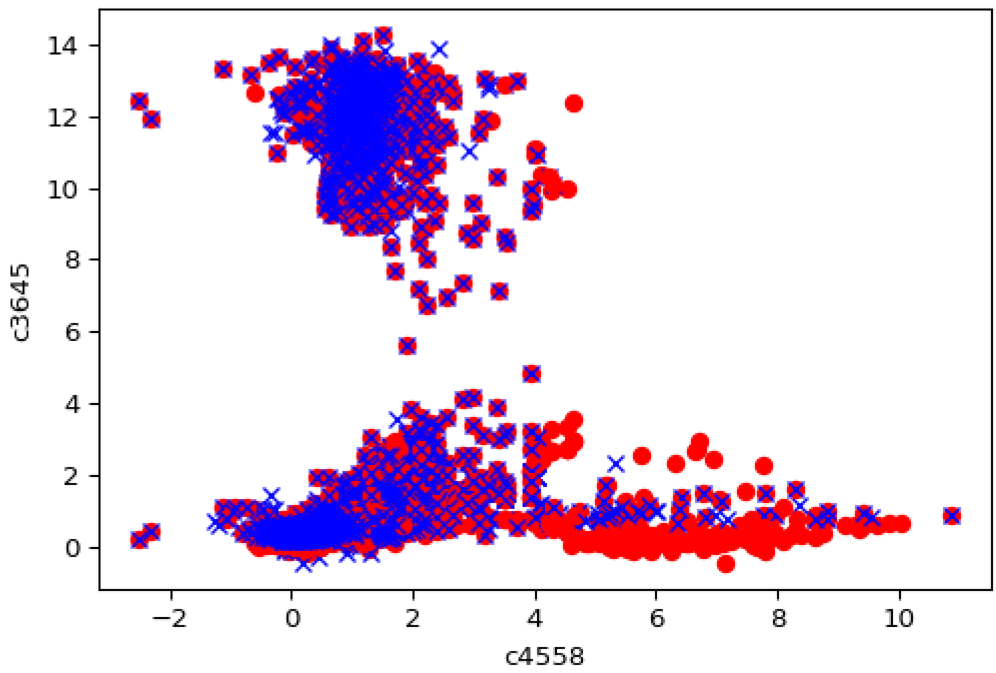
14
12
10
8
6
4
2
0
c3658
0
2
4
6
8
10
c4580



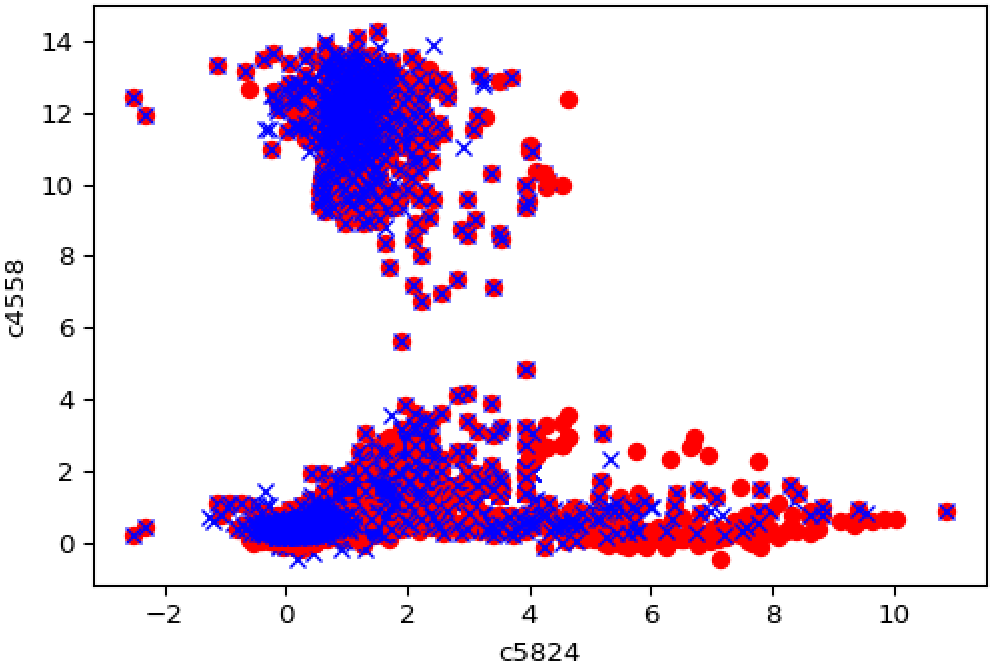
14
12
10
8
6
4
2
0
m4_5
−2
0
2
4
6
8
10
c4580

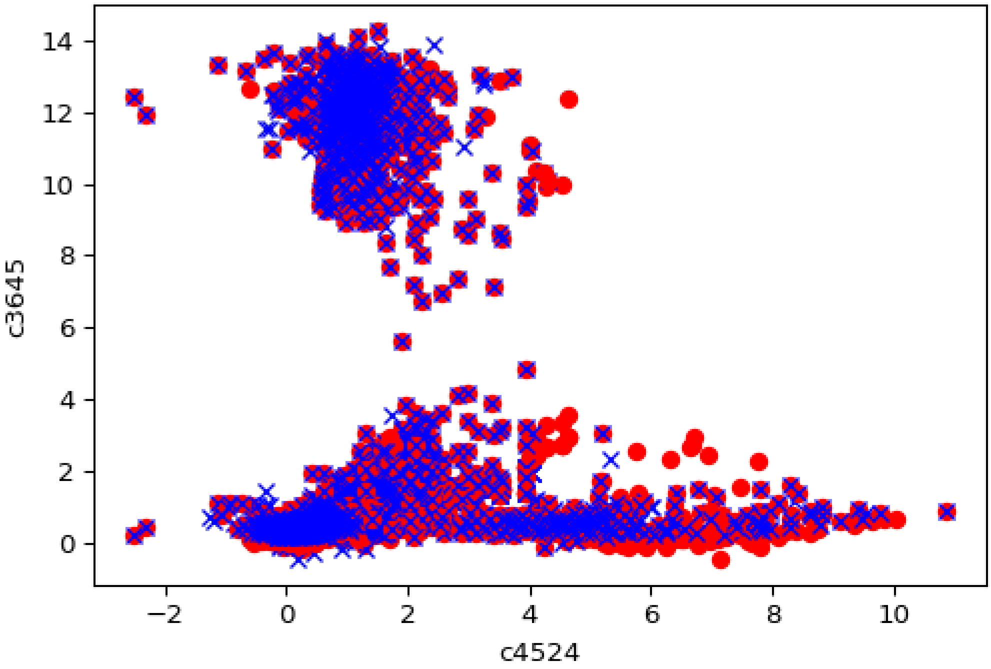


c3645
c4580



c3645
c4558





7.5. Notes

lynn Hillerbrand, mass luminosity relation L(K)

dN/dK, dN/dM3.8, peak of dN/dK// single color luminosity of and bolometric luminosity

integral of dN/dK