by Andrea Yocom. Sorry for the html again.

In [36]:
```
with open('hw3_todo.txt') as f:
    print f.read()
```

```
hw3 To Do

Understand the file format:
Column 1 is Storm ID
Column 2 is name of storm
Column 3,4 is lat/long
Column 5/6 is wind speed and central pressure
Column 7 is year
Column 8 is some running date field in units of hours.
Column 9 is a descriptor

What options are in descriptor field? Need to identify hurricane events.

Read/write to paste into Google Code with following format:
Col 0: Decade, begin 1920 end 2010. String, number, date, datetime ok
Col 1-n: number of hurricanes in pressure bins: number only

Syntax required for input to Google Code:
       [['str','str','str',...,'str']
        ['str',#,#,#,...,#],
        [...],
        [...],
        [...]]

Program determines:
 - frequency of hurricane events
 - min(central pressure) for each hurricane event
```

1. (DONE) Isolate hurricanes only
2. Isolate distinct storms by decade
3. Calculate min central pressure for each distinct storm
4. Define pressure bins
5. Count frequency of hurricanes, filter by pressure bins

In [37]:
```
import numpy as np
import scipy as sp
```

Want: [['Year','min-min+10','min - min+20' ... 'roundup(max)-10 - roundup(max)'], [1920,count if in first bin, count if in second bin, ... , count if in last bin], ... [2010, ...]] We only want the hurricanes, and we need to know the year, the id, and the central pressure. I know from looking at the file that hurricanes have "HURRICANE" in the descriptor, so I form a dataset from the year, id, and central pressure of hurricanes only. (http://askubuntu.com/questions/336296/bash-cut-and-grep-commands-through-python and http://python4astronomers.github.io/files/asciifiles.html)

In [94]:
```
# path to the file to read from
my_file = "master1.txt"
# what to look in each line
look_for = "HURRICANE"
# variable to store relevant data from lines containing specified string
data = []

with open(my_file, "r") as file_to_read:

    for line in file_to_read:
        if look_for in line:
            #eliminate the problem of the space in the name field
            line = line.replace('NOT NAMED','NOT_NAMED')
            #get rid of newline characters
            line = line.strip()
            #split into columns
            columns = line.split()

            #the following creates a list of lists
            data.append([int(columns[6]),int(columns[0]),int(columns[5])])
```

```
#the following gets rid of data with nonzero pressures
stuff = []

for i in range(len(data)):
    if data[i][2] != 0:
        stuff.append(data[i])

#print stuff
```

The strategy I'm testing on these shorter lists is to try and wind up with a list containing only ['year','storm id','min nonzero pressure'] - so I want to try and do it by sorting through the lists, for a given year and storm id finding the min pressure and adding to a new list... so far my attempts have been fruitless. The most recent attempt is at the bottom.

In [88]:
```
lista=[[1900,371,0],[1900,371,900],[1900,380,0],[1900,380,850],[1901,390,0],[1901,390,890]]

#read each year,id,pressure:
newStorm = [lista[0][0],lista[0][1],lista[0][2]]
distinctStorms = []
for i in range(len(lista)):
    #while in the same year as previous value and same ID as previous value, if not zero, append year,pre
ssure to data
    if lista[i][0] == newStorm[0]:
        if lista[i][1] == newStorm[1]:
            if not lista[i][2] == 0:
                distinctStorms.append([lista[i][0],lista[i][2]])
        #if new storm ID, update newStorm
        else:
            newStorm[1] = lista[i][1]
    #if new year, update newStorm
    else:
        newStorm[0] = lista[i][0]
print pressures
```

```
[974, 936, 973, 970, 970, 958, 986, 976, 990, 979, 950, 977, 953, 958, 953, 989, 972, 985, 959, 976, 952,
 957]
```

In [89]:
```
lista=[[1900,371,0],[1900,371,900],[1900,380,0],[1900,380,850],[1901,390,0],[1901,390,890]]
```

```
lista=[[1900,371,0],[1900,371,900],[1900,380,0],[1900,380,850],[1901,390,0],[1901,390,890]]
print lista[:][0]
year = lista[0][0]
#print year
for i in range(len(lista)):
    if lista[i][0] == year:
        print [lista[i][1],lista[i][2]]
    else:
        year += 1
```

```
[1900, 371, 0]
[371, 0]
[371, 900]
[380, 0]
[380, 850]
[390, 890]
```

In [91]:

```
lista=[[1900,371,0],[1900,371,900],[1900,380,0],[1900,380,850],[1901,390,0],[1901,390,890]]

#split into discrete lists
years = []
ids = []
pressures = []
for i in range(len(lista)):
    years.append(lista[i][0])
    ids.append(lista[i][1])
    pressures.append(lista[i][2])

print years
print ids
print pressures

#print the unique ids

for i in list(set(ids))
```

```
#get rid of zero pressures
stuff = []

for i in range(len(pressures)):
    if pressures[i] != 0:
        stuff.append([years[i],ids[i],pressures[i]])
print stuff
```

```
  File "<ipython-input-91-d42e06e2e2bd>", line 18
    for i in list(set(ids))
                          ^
SyntaxError: invalid syntax
```

In [81]:
```
lista=[[1900, 371, 974], [1900, 371, 936], [1901, 381, 973], [1902, 393, 970], [1903, 396, 970], [1903, 3
96, 958], [1903, 396, 986], [1903, 397, 976], [1903, 398, 990], [1906, 416, 979], [1906, 418, 950], [1906
, 419, 977], [1906, 420, 953], [1906, 420, 958], [1906, 422, 953], [1908, 432, 989], [1909, 442, 972], [1
909, 444, 985], [1909, 444, 959], [1909, 448, 976], [1909, 448, 952], [1909, 450, 957]]
#keep only data with nonzero pressure reading
stuff = []

for i in range(len(lista)):
    if lista[i][2] != 0:
        stuff.append(lista[i])

#split into discrete lists
years = []
ids = []
pressures = []
for i in range(len(stuff)):
    years.append(stuff[i][0])
    ids.append(stuff[i][1])
    pressures.append(stuff[i][2])

print list(set(years))
print list(set(ids))
print pressures
```

```
[1900, 1901, 1902, 1903, 1906, 1908, 1909]
[416, 448, 418, 419, 420, 422, 393, 396, 397, 398, 432, 450, 371, 442, 444, 381]
[974, 936, 973, 970, 970, 958, 986, 976, 990, 979, 950, 977, 953, 958, 953, 989, 972, 985, 959, 976, 952,
 957]
```

In [93]:

```
lista=[[1900, 371, 974], [1900, 371, 936], [1901, 381, 973], [1902, 393, 970], [1903, 396, 970], [1903, 3
96, 958], [1903, 396, 986], [1903, 397, 976], [1903, 398, 990], [1906, 416, 979], [1906, 418, 950], [1906
, 419, 977], [1906, 420, 953], [1906, 420, 958], [1906, 422, 953], [1908, 432, 989], [1909, 442, 972], [1
909, 444, 985], [1909, 444, 959], [1909, 448, 976], [1909, 448, 952], [1909, 450, 957]]

#lista contains only nonzero pressure readings for first decade

#initialize a list to compare each list within lista against
dummy = [0,0,100000]
keep = []
for i in range(len(lista)):

    #loop through unique storms. If new storm, update the keep list with old data, update [year,storm] in
 dummy, reset dummy pressure
    if lista[i][1] != dummy[1]:
        keep.append(dummy)
        dummy[0] = lista[i][0]
        dummy[1] = lista[i][1]
        dummy[2] = 100000
    #else if we are still on the same storm, we still have to check for lower pressures, but don't update
 dummy

    if lista[i][2] <= dummy[2]:
        #update pressure in dummy if pressure is a new low
        dummy[2] = lista[i][2]
    #else if we have reached a higher pressure, we loop back to the next row.

    #else if we are still on the same year, we still have to check for unique storms, but no need to upda
te dummy.
    #therefore year loop is redundant. Included below for reference.
```

```
    #loop through unique years row by row
    #if lista[i][0] != dummy[0]:
    #    dummy[0] = lista[i][0]
print keep
```

```
[[1909, 450, 957], [1909, 450, 957], [1909, 450, 957], [1909, 450, 957], [1909, 450, 957], [1909, 450, 95
7], [1909, 450, 957], [1909, 450, 957], [1909, 450, 957], [1909, 450, 957], [1909, 450, 957], [1909, 450,
 957], [1909, 450, 957], [1909, 450, 957], [1909, 450, 957], [1909, 450, 957]]
```