ga3

December 5, 2018

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

0.0.1 Please download the weather_data_ca.csv from https://osf.io/qvyw5/?show=view, renamed to "weather.csv" and put to same directory as ga3.ipynb

```
In [2]: #Please download the weather_data_ca.csv from https://osf.io/qvyw5/?show=view, renamed
        weather = pd.read_csv("weather.csv")
```

```
In [3]: weather.head()
```

Out[3]:

	Unnamed: 0	ID	YEARMONTHDAY	ELEMENT	DATA VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
0	91	USS0019L03S	19800101	PRCP	0	NaN	NaN	T	NaN
1	465	USC00048446	19800101	PRCP	0	NaN	NaN	0	800.0
2	490	USC00040115	19800101	PRCP	0	NaN	NaN	0	800.0
3	735	USW00023110	19800101	TMAX	172	NaN	NaN	X	NaN
4	736	USW00023110	19800101	TMIN	89	NaN	NaN	X	NaN

0.0.2 Retrieve the weather data for the relevant time periods for stations within 10 miles of Alameda County

```
In [4]: #check the time range of weather data
        (min(weather['YEARMONTHDAY'].unique()), max(weather['YEARMONTHDAY'].unique()))
```

```
Out[4]: (19800101, 20091231)
```

Weather data already in relevant time periods (1980-2009).

```
In [5]: stations_within_10miles_alameda = pd.read_csv("../group_assignment2/stations_within_10
```

```
In [6]: len(stations_within_10miles_alameda)
```

Out[6]: 88

In [7]:
```
#Remove 'TMIN' from weather data.
weather_new = weather[weather['ELEMENT'] != 'TMIN']
```

In [8]:
```
#Merge the 'stations_within_10miles_alameda' with 'weather' on station's ID.
#Get rid of all stations not appearing in 'stations_within_10miles_alameda'
merged = stations_within_10miles_alameda.merge(weather_new, on='ID', how='left')
merged.head()
```

Out[8]:

	Unnamed: 0_x	Unnamed: 0.1	ID	LATITUDE	LONGITUDE	ELEVATION	\
0	0	50462	US1CAAL0001	37.8123	-122.216	113.4	
1	0	50462	US1CAAL0001	37.8123	-122.216	113.4	
2	0	50462	US1CAAL0001	37.8123	-122.216	113.4	
3	0	50462	US1CAAL0001	37.8123	-122.216	113.4	
4	0	50462	US1CAAL0001	37.8123	-122.216	113.4	

	STATE	NAME	GSN FLAG	HCN/CRN FLAG	WMO ID	INVDIST	\
0	CA	PIEDMONT 1.0 SE	NaN	NaN	NaN	0.067894	
1	CA	PIEDMONT 1.0 SE	NaN	NaN	NaN	0.067894	
2	CA	PIEDMONT 1.0 SE	NaN	NaN	NaN	0.067894	
3	CA	PIEDMONT 1.0 SE	NaN	NaN	NaN	0.067894	
4	CA	PIEDMONT 1.0 SE	NaN	NaN	NaN	0.067894	

	Unnamed: 0_y	YEARMONTHDAY	ELEMENT	DATA VALUE	M-FLAG	Q-FLAG	S-FLAG	\
0	27876749.0	20081007.0	PRCP	0.0	NaN	NaN	N	
1	27978273.0	20081008.0	PRCP	0.0	NaN	NaN	N	
2	28079583.0	20081009.0	PRCP	0.0	NaN	NaN	N	
3	28179978.0	20081010.0	PRCP	0.0	NaN	NaN	N	
4	28279413.0	20081011.0	PRCP	0.0	NaN	NaN	N	

	OBS-TIME
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN

0.0.3 Identify the stations that meet Ranson's criteria for inclusion in each year:

Missing values:

In [9]:
```
#Check how many missing values in each column
merged.isnull().sum()
```

Out[9]:
```
Unnamed: 0_x         0
Unnamed: 0.1         0
ID                   0
LATITUDE             0
```

```
LONGITUDE            0
ELEVATION            0
STATE                0
NAME                 0
GSN FLAG        259014
HCN/CRN FLAG    218360
WMO ID          228340
INVDIST              0
Unnamed: 0_y        49
YEARMONTHDAY        49
ELEMENT             49
DATA VALUE          49
M-FLAG          184716
Q-FLAG          257731
S-FLAG              49
OBS-TIME         67461
dtype: int64
```

There are 49 rows with missing 'ELEMENT' and 'DATA VALUE'. It is probably because some stations in 'stations.csv' don't have corresponding records in 'weather.csv', so we remove missing values.

```
In [10]: #Drop missing values in 'ELEMENT', 'DATA VALUE'
         stations_weather = merged.dropna(subset=['ELEMENT', 'DATA VALUE'])
```

Check duplicate data:

```
In [11]: #Check the length before and after dropping duplicates
         len(stations_weather) == len(stations_weather.drop_duplicates(['ID', 'YEARMONTHDAY',

Out[11]: True
```

No duplicates.

Convert 'Data Value' to correct unit:

```
In [12]: #Convert unit of TMAX data value to Fahrenheit which Ranson used:
         stations_weather['DATA VALUE'] = np.where(stations_weather['ELEMENT'] == 'TMAX',
                                                   (stations_weather['DATA VALUE']/10)* 1.8 + 3
                                                   stations_weather['DATA VALUE'])

/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  after removing the cwd from sys.path.
```

In [13]:
```
stations_weather.tail()
```

Out[13]:

	Unnamed: 0_x	Unnamed: 0.1	ID	LATITUDE	LONGITUDE	\
259009	2714	106025	USW00093228	37.6542	-122.115	
259010	2714	106025	USW00093228	37.6542	-122.115	
259011	2714	106025	USW00093228	37.6542	-122.115	
259012	2714	106025	USW00093228	37.6542	-122.115	
259013	2714	106025	USW00093228	37.6542	-122.115	

	ELEVATION	STATE	NAME	GSN FLAG	HCN/CRN FLAG	WMO ID	\
259009	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259010	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259011	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259012	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259013	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	

	INVDIST	Unnamed: 0_y	YEARMONTHDAY	ELEMENT	DATA VALUE	M-FLAG	\
259009	0.071211	36648671.0	20091229.0	PRCP	10.00	NaN	
259010	0.071211	36751411.0	20091230.0	TMAX	57.02	NaN	
259011	0.071211	36751413.0	20091230.0	PRCP	0.00	T	
259012	0.071211	36853346.0	20091231.0	TMAX	62.06	NaN	
259013	0.071211	36853348.0	20091231.0	PRCP	0.00	NaN	

	Q-FLAG	S-FLAG	OBS-TIME
259009	NaN	0	2400.0
259010	NaN	0	2400.0
259011	NaN	0	2400.0
259012	NaN	0	2400.0
259013	NaN	0	2400.0

"bias" adjustment for each weather station

In [14]:
```
stations_temp = stations_weather[stations_weather.ELEMENT=='TMAX']
```

In [15]:
```
import random
# set seed
np.random.seed(101)

ID = stations_temp.ID.unique()
# Create the starting mu
mu = np.zeros(len(ID))
mu_old = np.zeros(len(ID))

while abs(mu_old-mu).any() > 0.01 or mu_old.all() == 0:
    # update mu
    mu_old = mu
    # get reference station
    ref_idx = np.random.choice(np.arange(len(ID)),1)
    ref = ID[ref_idx][0]
```

```
            station_i = stations_temp[stations_temp.ID == ref]

            # other station
            for j in ID:
                station_idx = np.arange(len(ID))[j == ID]
                station_j = stations_temp[stations_temp.ID == j]
                # get the date that both stations reported
                shared_date = station_i[['YEARMONTHDAY']].merge(station_j[['YEARMONTHDAY']], c
                n = len(shared_date)

                # if there are shared dates
                if n>0:
                    temp_i = 0
                    for day in shared_date.YEARMONTHDAY:
                        temp_i = temp_i + float(station_i[station_i.YEARMONTHDAY==day]['DATA V

                    temp_j = 0
                    for day in shared_date.YEARMONTHDAY:
                        temp_j = temp_j + float(station_j[station_j.YEARMONTHDAY==day]['DATA V

                    mu[station_idx] = mu_old[station_idx]+float(temp_i+mu_old[ref_idx]*n - ter
                # if there's no shared date
                else:
                    mu[station_idx] = mu_old[station_idx]
```

```
In [16]: mu
```

```
Out[16]: array([ 1.45225575, -2.87482661, -5.68162948, -1.30004331, -1.20885482,
                 0.422705  ,  0.47406818, -0.1452698 , -6.97727412, -0.22802503,
                -2.13433489, -0.4030874 , -4.44915912, -0.41757235,  2.15383049,
                -0.67176254,  0.57227034,  3.53482382,  0.29042937,  2.06852628,
                 1.02454179, -5.03463377,  0.32080085])
```

county adjusted C

```
In [17]: w = stations_temp.groupby('ID').mean().INVDIST
```

```
In [18]: C = sum(w * mu)/sum(w)
         C
```

```
Out[18]: -1.1953239383398222
```

Adjust the temperature value:

```
In [19]: #Construct a dataframe for mu of stations
         d = {'ID': ID, 'station_mu': mu}
         mu_df = pd.DataFrame(data=d)
         mu_df.head()
```

Out[19]:

	ID	station_mu
0	USC00040693	1.452256
1	USC00043244	-2.874827
2	USC00044997	-5.681629
3	USC00045915	-1.300043
4	USC00046144	-1.208855

In [20]:
```
station_adjusted = stations_weather.merge(mu_df, on='ID', how='left')
station_adjusted.tail()
```

Out[20]:

	Unnamed: 0_x	Unnamed: 0.1	ID	LATITUDE	LONGITUDE	\
258960	2714	106025	USW00093228	37.6542	-122.115	
258961	2714	106025	USW00093228	37.6542	-122.115	
258962	2714	106025	USW00093228	37.6542	-122.115	
258963	2714	106025	USW00093228	37.6542	-122.115	
258964	2714	106025	USW00093228	37.6542	-122.115	

	ELEVATION	STATE	NAME	GSN FLAG	HCN/CRN FLAG	\
258960	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258961	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258962	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258963	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258964	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	

	...	INVDIST	Unnamed: 0_y	YEARMONTHDAY	ELEMENT	DATA VALUE	\
258960	...	0.071211	36648671.0	20091229.0	PRCP	10.00	
258961	...	0.071211	36751411.0	20091230.0	TMAX	57.02	
258962	...	0.071211	36751413.0	20091230.0	PRCP	0.00	
258963	...	0.071211	36853346.0	20091231.0	TMAX	62.06	
258964	...	0.071211	36853348.0	20091231.0	PRCP	0.00	

	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME	station_mu
258960	NaN	NaN	0	2400.0	0.320801
258961	NaN	NaN	0	2400.0	0.320801
258962	T	NaN	0	2400.0	0.320801
258963	NaN	NaN	0	2400.0	0.320801
258964	NaN	NaN	0	2400.0	0.320801

[5 rows x 21 columns]

In [21]:
```
# adjust tmax
station_adjusted['DATA VALUE'] = np.where(station_adjusted['ELEMENT'] == 'TMAX',
                                          station_adjusted['DATA VALUE']+ station_adju
                                          station_adjusted['DATA VALUE'])
```

bias adjustment for the Prcp

In [22]:
```
stations_prcp = stations_weather[stations_weather.ELEMENT=='PRCP']
```

In [23]:
```
stations_prcp.tail()
```

Out[23]:

	Unnamed: 0_x	Unnamed: 0.1	ID	LATITUDE	LONGITUDE	\
259005	2714	106025	USW00093228	37.6542	-122.115	
259007	2714	106025	USW00093228	37.6542	-122.115	
259009	2714	106025	USW00093228	37.6542	-122.115	
259011	2714	106025	USW00093228	37.6542	-122.115	
259013	2714	106025	USW00093228	37.6542	-122.115	

	ELEVATION	STATE	NAME	GSN FLAG	HCN/CRN FLAG	WMO ID	\
259005	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259007	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259009	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259011	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	
259013	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	72585.0	

	INVDIST	Unnamed: 0_y	YEARMONTHDAY	ELEMENT	DATA VALUE	M-FLAG	\
259005	0.071211	36444853.0	20091227.0	PRCP	3.0	NaN	
259007	0.071211	36546191.0	20091228.0	PRCP	0.0	T	
259009	0.071211	36648671.0	20091229.0	PRCP	10.0	NaN	
259011	0.071211	36751413.0	20091230.0	PRCP	0.0	T	
259013	0.071211	36853348.0	20091231.0	PRCP	0.0	NaN	

	Q-FLAG	S-FLAG	OBS-TIME
259005	NaN	0	2400.0
259007	NaN	0	2400.0
259009	NaN	0	2400.0
259011	NaN	0	2400.0
259013	NaN	0	2400.0

In [24]:
```
import random
# set seed
np.random.seed(101)

ID = stations_prcp.ID.unique()
# Create the starting mu
mu = np.zeros(len(ID))
mu_old = np.zeros(len(ID))

while abs(mu_old-mu).any() > 0.01 or mu_old.all() == 0:
    # update mu
    mu_old = mu
    # get reference station
    ref_idx = np.random.choice(np.arange(len(ID)),1)
    ref = ID[ref_idx][0]
    station_i = stations_prcp[stations_prcp.ID == ref]

    # other station
```

```
            for j in ID:
                station_idx = np.arange(len(ID))[j == ID]
                station_j = stations_prcp[stations_prcp.ID == j]
                # get the date that both stations reported
                shared_date = station_i[['YEARMONTHDAY']].merge(station_j[['YEARMONTHDAY']], c
                n = len(shared_date)

                # if there are shared dates
                if n>0:
                    prcp_i = 0
                    for day in shared_date.YEARMONTHDAY:
                        prcp_i = prcp_i + float(station_i[station_i.YEARMONTHDAY==day]['DATA V

                    prcp_j = 0
                    for day in shared_date.YEARMONTHDAY:
                        prcp_j = prcp_j + float(station_j[station_j.YEARMONTHDAY==day]['DATA V

                    mu[station_idx] = mu_old[station_idx]+float(prcp_i+mu_old[ref_idx]*n - prc
                # if there's no shared date
                else:
                    mu[station_idx] = mu_old[station_idx]
```

In [25]:
```
mu
```

Out[25]:
```
array([ 58.69678161,  61.69686869,  61.54358209,  62.82346154,
        41.26      ,  62.68271293,  19.88      ,  55.84590308,
        56.55411765,  59.87509434, -73.74      ,  46.66      ,
        47.66      ,  33.76      ,  71.66361446,  59.26      ,
        67.06705528,  73.09512545,  67.03695246,  60.3875888 ,
        67.3106396 ,  61.47471218,  60.26098738,  55.06104712,
        68.60829174,  59.11481594,  64.47119699,  63.2054171 ,
        67.22437055,  66.04502595])
```

In [26]:
```
w = stations_prcp.groupby('ID').mean().INVDIST
```

In [27]:
```
C = sum(w * mu)/sum(w)
C
```

Out[27]:
```
55.511203007352904
```

In [28]:
```
#Construct a dataframe for mu of stations
d = {'ID': ID, 'station_prcp_mu': mu}
prcp_mu_df = pd.DataFrame(data=d)
prcp_mu_df.head()
```

Out[28]:

	ID	station_prcp_mu
0	US1CAAL0001	58.696782
1	US1CAAL0002	61.696869
2	US1CAAL0003	61.543582
3	US1CAAL0004	62.823462
4	US1CAAL0016	41.260000

```
In [29]: station_adjusted_prcp = stations_weather.merge(prcp_mu_df, on='ID', how='left')
         station_adjusted_prcp.tail()
```

Out[29]:

	Unnamed: 0_x	Unnamed: 0.1	ID	LATITUDE	LONGITUDE	\
258960	2714	106025	USW00093228	37.6542	-122.115	
258961	2714	106025	USW00093228	37.6542	-122.115	
258962	2714	106025	USW00093228	37.6542	-122.115	
258963	2714	106025	USW00093228	37.6542	-122.115	
258964	2714	106025	USW00093228	37.6542	-122.115	

	ELEVATION	STATE	NAME	GSN FLAG	HCN/CRN FLAG	\
258960	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258961	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258962	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258963	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	
258964	13.1	CA	HAYWARD AIR TERMINAL	NaN	NaN	

	...	INVDIST	Unnamed: 0_y	YEARMONTHDAY	ELEMENT	\
258960	...	0.071211	36648671.0	20091229.0	PRCP	
258961	...	0.071211	36751411.0	20091230.0	TMAX	
258962	...	0.071211	36751413.0	20091230.0	PRCP	
258963	...	0.071211	36853346.0	20091231.0	TMAX	
258964	...	0.071211	36853348.0	20091231.0	PRCP	

	DATA VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME	station_prcp_mu
258960	10.00	NaN	NaN	0	2400.0	66.045026
258961	57.02	NaN	NaN	0	2400.0	66.045026
258962	0.00	T	NaN	0	2400.0	66.045026
258963	62.06	NaN	NaN	0	2400.0	66.045026
258964	0.00	NaN	NaN	0	2400.0	66.045026

[5 rows x 21 columns]

```
In [30]: station_adjusted['station_prep_mu']=station_adjusted_prcp['station_prcp_mu']

In [31]: # adjust and convert precipitation
         station_adjusted['DATA VALUE'] = np.where(station_adjusted['ELEMENT'] == 'PRCP',
                                                   (station_adjusted['DATA VALUE']+ station_adj
                                                   station_adjusted['DATA VALUE'])

In [32]: station_adjusted.tail()
```

Out[32]:

	Unnamed: 0_x	Unnamed: 0.1	ID	LATITUDE	LONGITUDE	\
258960	2714	106025	USW00093228	37.6542	-122.115	
258961	2714	106025	USW00093228	37.6542	-122.115	
258962	2714	106025	USW00093228	37.6542	-122.115	
258963	2714	106025	USW00093228	37.6542	-122.115	
258964	2714	106025	USW00093228	37.6542	-122.115	

```
        ELEVATION STATE                    NAME  GSN FLAG HCN/CRN FLAG  \
258960       13.1    CA  HAYWARD AIR TERMINAL       NaN           NaN
258961       13.1    CA  HAYWARD AIR TERMINAL       NaN           NaN
258962       13.1    CA  HAYWARD AIR TERMINAL       NaN           NaN
258963       13.1    CA  HAYWARD AIR TERMINAL       NaN           NaN
258964       13.1    CA  HAYWARD AIR TERMINAL       NaN           NaN

                ...         Unnamed: 0_y  YEARMONTHDAY  ELEMENT  DATA VALUE  \
258960          ...            36648671.0    20091229.0     PRCP    2.053382
258961          ...            36751411.0    20091230.0     TMAX   58.536125
258962          ...            36751413.0    20091230.0     PRCP    1.053382
258963          ...            36853346.0    20091231.0     TMAX   63.576125
258964          ...            36853348.0    20091231.0     PRCP    1.053382

       M-FLAG  Q-FLAG S-FLAG OBS-TIME station_mu  station_prep_mu
258960    NaN     NaN      0   2400.0   0.320801         66.045026
258961    NaN     NaN      0   2400.0   0.320801         66.045026
258962      T     NaN      0   2400.0   0.320801         66.045026
258963    NaN     NaN      0   2400.0   0.320801         66.045026
258964    NaN     NaN      0   2400.0   0.320801         66.045026

[5 rows x 22 columns]
```

In [33]:
```
station_adjusted.to_csv("station_adjusted.csv")
```

0.0.4 bin the averaged weather data, aggregate it by month using the categories Ranson used:

"I model the daily distribution of temperatures within a month using 11 bin variables: < 10F, 10–19F, 20–29F, 30–39F, 40–49F, 50–59F, 60–69F, 70–79F, 80–89F, 90–99F; and >100F."
measure the number of days per month spent in each of 11 maximum daily temperature bins (o10 1F, 10–20 1F,..., 90–100 1F, Z100 1F) and five daily precipitation bins (0 mm, 1–4 mm, 5–14 mm, 15–29 mm, and Z 30 mm).

0.0.5 Please refer to above step for how the dataset is generated.

In [2]:
```
station_adjusted = pd.read_csv("station_adjusted.csv")
station_adjusted.head()
```

```
/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:3020: DtypeWarning: Co
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:
```
   Unnamed: 0  Unnamed: 0_x  Unnamed: 0.1           ID  LATITUDE  LONGITUDE  \
0           0             0         50462  US1CAAL0001   37.8123   -122.216
1           1             0         50462  US1CAAL0001   37.8123   -122.216
2           2             0         50462  US1CAAL0001   37.8123   -122.216
3           3             0         50462  US1CAAL0001   37.8123   -122.216
4           4             0         50462  US1CAAL0001   37.8123   -122.216
```

```
   ELEVATION STATE             NAME  GSN FLAG    ...       INVDIST  \
0      113.4    CA  PIEDMONT 1.0 SE       NaN    ...      0.067894
1      113.4    CA  PIEDMONT 1.0 SE       NaN    ...      0.067894
2      113.4    CA  PIEDMONT 1.0 SE       NaN    ...      0.067894
3      113.4    CA  PIEDMONT 1.0 SE       NaN    ...      0.067894
4      113.4    CA  PIEDMONT 1.0 SE       NaN    ...      0.067894

   Unnamed: 0_y  YEARMONTHDAY  ELEMENT  DATA VALUE M-FLAG  Q-FLAG S-FLAG  \
0    27876749.0    20081007.0     PRCP         0.0    NaN     NaN      N
1    27978273.0    20081008.0     PRCP         0.0    NaN     NaN      N
2    28079583.0    20081009.0     PRCP         0.0    NaN     NaN      N
3    28179978.0    20081010.0     PRCP         0.0    NaN     NaN      N
4    28279413.0    20081011.0     PRCP         0.0    NaN     NaN      N

  OBS-TIME station_mu
0      NaN        NaN
1      NaN        NaN
2      NaN        NaN
3      NaN        NaN
4      NaN        NaN

[5 rows x 22 columns]
```

```
In [3]: station_adjusted.columns
```

```
Out[3]: Index(['Unnamed: 0', 'Unnamed: 0_x', 'Unnamed: 0.1', 'ID', 'LATITUDE',
               'LONGITUDE', 'ELEVATION', 'STATE', 'NAME', 'GSN FLAG', 'HCN/CRN FLAG',
               'WMO ID', 'INVDIST', 'Unnamed: 0_y', 'YEARMONTHDAY', 'ELEMENT',
               'DATA VALUE', 'M-FLAG', 'Q-FLAG', 'S-FLAG', 'OBS-TIME', 'station_mu'],
              dtype='object')
```

```
In [4]: station_adjusted.shape
```

```
Out[4]: (258965, 22)
```

```
In [5]: station_TMAX = station_adjusted[station_adjusted['ELEMENT'] == 'TMAX']
        station_PRCP = station_adjusted[station_adjusted['ELEMENT'] == 'PRCP']
        station_TMAX['wi_val'] = station_TMAX['INVDIST'] * station_TMAX['DATA VALUE']
        station_PRCP['wi_val'] = station_PRCP['INVDIST'] * station_PRCP['DATA VALUE']
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  after removing the cwd from sys.path.
```

```
In [6]: weighted_TMAX = station_TMAX.groupby('YEARMONTHDAY', as_index = False).agg({'wi_val' :
        weighted_PRCP = station_PRCP.groupby('YEARMONTHDAY', as_index = False).agg({'wi_val' :
        weighted_PRCP.tail()
```

```
Out[6]:        YEARMONTHDAY     wi_val
        10953    20091227.0  12.167668
        10954    20091228.0   0.404646
        10955    20091229.0   1.561178
        10956    20091230.0   1.885155
        10957    20091231.0   0.120179
```

```
In [7]: # Warning: Extremely long runtime!!!!!! Don't repeatedly run
        dict_date_TMAX = {}
        for i in station_TMAX['YEARMONTHDAY']:
            only_i = station_TMAX[station_TMAX['YEARMONTHDAY'] == i]
            dict_date_TMAX[i] = sum(only_i['wi_val'])/sum(only_i['INVDIST'].unique())
```

```
In [8]: #sanity check
        series_date_TMAX = pd.Series(data = dict_date_TMAX)
        series_date_TMAX.describe()
```

```
Out[8]: count    10958.000000
        mean        68.620590
        std         11.253764
        min         33.934814
        25%         59.317645
        50%         68.152397
        75%         77.041568
        max        104.816593
        dtype: float64
```

```
In [9]: series_date_TMAX.head()
```

```
Out[9]: 19800101.0    58.245443
        19800102.0    54.548382
        19800103.0    55.110284
        19800104.0    53.382759
        19800105.0    54.689231
        dtype: float64
```

```
In [10]: # Warning: Extremely long runtime!!!!!! Don't repeatedly run
         dict_date_PRCP = {}
         for i in station_PRCP['YEARMONTHDAY']:
             only_i = station_PRCP[station_PRCP['YEARMONTHDAY'] == i]
             dict_date_PRCP[i] = sum(only_i['wi_val'])/sum(only_i['INVDIST'].unique())
```

```
In [11]: #sanity check, NorCal is dry so this makes sense
         series_date_PRCP = pd.Series(data = dict_date_PRCP)
         series_date_PRCP.describe()

Out[11]: count    10958.000000
         mean         1.354813
         std          4.311938
         min          0.000000
         25%          0.000000
         50%          0.000000
         75%          0.115445
         max         74.749216
         dtype: float64

In [12]: series_date_PRCP.head()

Out[12]: 20081007.0    0.000000
         20081008.0    0.015742
         20081009.0    0.000000
         20081010.0    0.000000
         20081011.0    0.000000
         dtype: float64

In [13]: #Put bias-adjusted temperature into bins
         import math
         #define bins
         TMAX_df = pd.DataFrame({'DATE':series_date_TMAX.index, 'TMAX':series_date_TMAX.values
         temp_bins = [-math.inf,10,20,30,40,50,60,70,80,90,100,math.inf]
         group_names_temp = ['<10F','10-19F','20-29F','30-39F','40-49F','50-59F','60-69F','70-
         TMAX_df['temp_bins'] = pd.cut(TMAX_df['TMAX'], temp_bins, labels = group_names_temp)

In [14]: TMAX_df.tail()

Out[14]:             DATE       TMAX temp_bins
         10953  19970504.0  72.517124    70-79F
         10954  19970505.0  72.994418    70-79F
         10955  19970510.0  75.223254    70-79F
         10956  19970511.0  75.222487    70-79F
         10957  20090126.0  50.899420    50-59F

In [15]: #Put precipitation into bins
         prcp_bins = [-math.inf,0.000001,5,15,30,math.inf]
         group_names_prcp = ['0mm','1-4mm','5-14mm','15-29mm','>30mm']
         PRCP_df = pd.DataFrame({'DATE':series_date_PRCP.index, 'PRCP':series_date_PRCP.values
         PRCP_df['prcp_bins'] = pd.cut(PRCP_df['PRCP'], prcp_bins, labels = group_names_prcp)

In [16]: PRCP_df['prcp_bins'].unique()

Out[16]: [0mm, 1-4mm, 15-29mm, 5-14mm, >30mm]
         Categories (5, object): [0mm < 1-4mm < 5-14mm < 15-29mm < >30mm]
```

```
In [17]: #count frequancy of bin for temperature in each month
         TMAX_df['YearMonth'] = TMAX_df['DATE'].astype(str).str[:6]
         #TMAX_df.groupby(['YearMonth','temp_bins']).temp_bins.count()

In [18]: #count frequancy of bin for precipitation in each month
         PRCP_df['YearMonth'] = PRCP_df['DATE'].astype(str).str[:6]
         PRCP_df.groupby(['YearMonth','prcp_bins']).prcp_bins.count()
```

Out[18]:

YearMonth	prcp_bins	
198001	0mm	14
	1-4mm	9
	5-14mm	5
	15-29mm	3
198002	0mm	15
	1-4mm	4
	5-14mm	5
	15-29mm	4
	>30mm	1
198003	0mm	17
	1-4mm	10
	5-14mm	4
198004	0mm	17
	1-4mm	12
	15-29mm	1
198005	0mm	26
	1-4mm	4
	5-14mm	1
198006	0mm	28
	1-4mm	2
198007	0mm	28
	1-4mm	2
	5-14mm	1
198008	0mm	31
198009	0mm	30
198010	0mm	27
	1-4mm	4
198011	0mm	23
	1-4mm	7
198012	0mm	22
		..
200902	5-14mm	6
	15-29mm	3
200903	0mm	19
	1-4mm	7
	5-14mm	4
	15-29mm	1
200904	0mm	23
	1-4mm	6

```
                5-14mm        1
        200905  0mm          25
                1-4mm         3
                5-14mm        3
        200906  0mm          24
                1-4mm         6
        200907  0mm          28
                1-4mm         3
        200908  0mm          29
                1-4mm         2
        200909  0mm          23
                1-4mm         7
        200910  0mm          21
                1-4mm         6
                5-14mm        2
                >30mm         2
        200911  0mm          17
                1-4mm        13
        200912  0mm           8
                1-4mm        18
                5-14mm        4
                15-29mm       1
        Name: prcp_bins, Length: 1011, dtype: int64
```

```
In [19]: TMAX_df.head()
```

Out[19]:

	DATE	TMAX	temp_bins	YearMonth
0	19800101.0	58.245443	50-59F	198001
1	19800102.0	54.548382	50-59F	198001
2	19800103.0	55.110284	50-59F	198001
3	19800104.0	53.382759	50-59F	198001
4	19800105.0	54.689231	50-59F	198001

```
In [20]: TMAX_df['temp_bins'].unique()
```

```
Out[20]: [50-59F, 60-69F, 70-79F, 80-89F, 90-99F, 40-49F, 30-39F, >100F]
         Categories (8, object): [30-39F < 40-49F < 50-59F < 60-69F < 70-79F < 80-89F < 90-99F
```

```
In [21]: TMAX_data = TMAX_df.pivot_table(index='temp_bins',
                                columns='YearMonth',
                                values='TMAX',
                                fill_value = 0,
                                aggfunc='count').unstack().to_frame()
```

```
In [22]: TMAX_data = TMAX_data.reset_index().set_index('YearMonth')
         TMAX_data.head()
```

Out[22]:

	temp_bins	0
YearMonth		

```
          198001       30-39F   0
          198001       40-49F   0
          198001       50-59F  26
          198001       60-69F   5
          198001       70-79F   0
```

```
In [23]: TMAX_data = TMAX_data.sort_index()
```

```
In [24]: PRCP_data = PRCP_df.pivot_table(index='prcp_bins',
                                columns='YearMonth',
                                values='PRCP',
                                fill_value = 0,
                                aggfunc='count').unstack().to_frame().sort_index()
```

```
In [25]: PRCP_data.tail()
```

```
Out[25]:                        0
          YearMonth prcp_bins
          200912    0mm         8
                    1-4mm      18
                    5-14mm      4
                    15-29mm     1
                    >30mm       0
```

```
In [26]: TMAX_data.to_csv('../group_assignment3/TMAX_data.csv')
          PRCP_data.to_csv('../group_assignment3/PRCP_data.csv')
```

```
In [63]: TMAX_data_2 = TMAX_df.groupby(['YearMonth','temp_bins']).temp_bins.count().to_frame()
```

```
In [64]: TMAX_data_2.to_csv('TMAX_data.csv')
```