7.1 Weather Analysis- Massachusetts

DSC 232R, Class 7 : Weather Data

Week 4

Weather Data: Initial Visualization

Notebook Link

For MA

```
In [1]: state=“MA”
```

Data Source

- The data is from NOAA and is called GHCND, you can get the details and download the data from AWS S3 Bucket
- We translate the raw data from the S3 Bucket into parquet files that store tables called “weather” and “stations”

Documentation

https://github.com/awslabs/open-data-docs/tree/main/docs/noaa/noaa-ghcn

Managed By

See all datasets managed by NOAA.

Contact

For questions regarding data content or quality, visit the NOAA GHCN site.
For any questions regarding data delivery or any general questions regarding the NOAA Open Data Dissemination (NODD) Program, email the NODD Team at nodd@noaa.gov.
We also seek to identify case studies on how NOAA data is being used and will be featuring those stories in joint publications and in upcoming events. If you are interested in seeing your story highlighted, please share it with the NODD team by emailing nodd@noaa.gov

How to Cite

NOAA Global Historical Climatology Network Daily (GHCN-D) was accessed on `DATE` from https://registry.opendata.aws/noaa-ghcn.

Read Data

Show 3 rows of joined weather+stations table

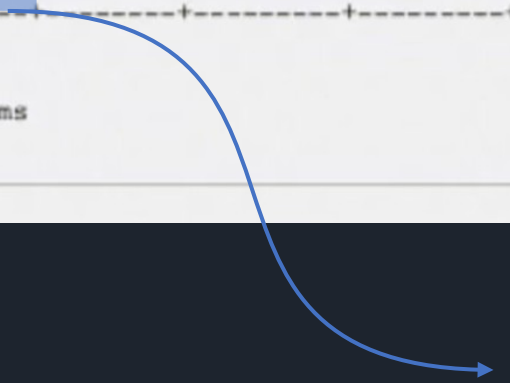
Read Data

```
In [6]: ### Total number of stations
        stations.count()

Out[6]: 119503
```

Read Data

```
In [7]: %%time
        weather=measurements.join(stations,on='station')
        weather.show(3)
```

```
+-----------+-----------+----+--------------------+--------+---------+---------+------------+--------------------+-----+-------+
|    Station|Measurement|Year|              Values|latitude|longitude|elevation|  dist2coast|                name|state|country|
+-----------+-----------+----+--------------------+--------+---------+---------+------------+--------------------+-----+-------+
|AG000060390|       TAVG|2022|[79 00 6C 00 66 0...| 36.7167|     3.25|     24.0|   8.0234375| ALGER-DAR EL BEIDA|     |Algeria|
|AGE00147716|       TAVG|2022|[85 00 83 00 7C 0...|    35.1|    -1.85|     83.0|0.5224609375|NEMOURS (GHAZAOUET)|     |Algeria|
|AGM00060360|       TMIN|2022|[5A 00 19 FC 4B 0...|  36.822|    7.809|      4.9|  3.16015625|              ANNABA|     |Algeria|
+-----------+-----------+----+--------------------+--------+---------+---------+------------+--------------------+-----+-------+
only showing top 3 rows

CPU times: user 1.92 ms, sys: 2 ms, total: 3.93 ms
Wall time: 2.64 s
```



Vectors of 365 values

Read Data

```
In [7]: %%time
        weather = measurements.join(stations,on='station')
        weather.show(3)
```

```
+-----------+-----------+----+--------------------+--------+---------+---------+------------+--------------------
|    Station|measurement|Year|              Values|latitude|longitude|elevation|  dist2coast|                ns
+-----------+-----------+----+--------------------+--------+---------+---------+------------+--------------------
|AG000060390|       TAVG|2022|[79 00 6C 00 66 0...| 36.7167|     3.25|     24.0|   8.0234375| ALGER-DAR EL BE
|AGE00147716|       TAVG|2022|[85 00 83 00 7C 0...|    35.1|    -1.85|     83.0|0.5224609375|NEMOURS (GHAZAOU
|AGM00060360|       TMIN|2022|[5A 00 19 FC 4B 0...|  36.822|    7.809|      4.9|  3.16015625|             ANN
+-----------+-----------+----+--------------------+--------+---------+---------+------------+--------------------
only showing top 3 rows

CPU times: user 1.92 ms, sys: 2 ms, total: 3.93 ms
Wall time: 2.64 s
```

```
In [8]: sqlContext.registerDataFrameAsTable(weather,'weather')
```

Read Data

In [9]:
```
ms=['TMAX', 'SNOW', 'SNWD', 'TMIN', 'PRCP', 'TOBS']
# ms=['TMAX', 'TMIN', 'TOBS']
cms='or\n'.join(['Measurement="%s" '%(m) for m in ms])

##  read all data for state
Query="""
SELECT *
FROM weather
WHERE state="%s" and
(%s)
"""%(state,cms)
print(Query)
```

```
SELECT *
FROM weather
WHERE state="MA" and
(Measurement="TMAX" or
Measurement="SNOW" or
Measurement="SNWD" or
Measurement="TMIN" or
Measurement="PRCP" or
Measurement="TOBS" )
```

Read or Compute Statistics Information for State

Read or Compute Statistics Information for Sate

```
In [12]: if os.path.isfile(pkl_filename):
             print('precomputed statistics file exists')
             with open(pkl_filename,'br') as pkl_file:
                 STAT=load(pkl_file)
         else:
             print('computing statistics')
             STAT=computeStatistics(sqlContext,weather,measurements=ms)
             with open(pkl_filename,'bw') as pkl_file:
                 dump(STAT,pkl_file)

         STAT.keys()

         precomputed statistics file exists

Out[12]: dict_keys(['TMAX', 'SNOW', 'SNWD', 'TMIN', 'PRCP', 'TOBS'])
```

Read or Compute Statistics Information for Sate

```
In [13]: print("   Name  \t                Description              \t  Size")
         print("-"*80)
         print('\n'.join(["%10s\t%40s\t%s"%(s[0],s[1],str(s[2])) for s in STAT_Descriptions]))
```

```
   Name                              Description                                 Size
--------------------------------------------------------------------------------
     UnDef              sample of number of undefs per row        vector whose length varies between measurements
        NE                   count of defined values per day      (366,)
 SortedVals                                Sample of values      vector whose length varies between measurements
      mean                                      mean value      ()
       std                                             std      ()
    low100                                   bottom 1%         ()
   high100                                      top 1%         ()
   low1000                                  bottom 0.1%        ()
  high1000                                     top 0.1%        ()
                                  Sum of values per day         (366,)
      Mean                                        E/NE         (366,)
         O                        Sum of outer products        (366, 366)
        NO                   counts for outer products        (366, 366)
       Cov                                        O/NO         (366, 366)
       Var        The variance per day = diagonal of Cov       (366,)
    eigval                              PCA eigen-values       (366,)
    eigvec                              PCA eigen-vectors      (366, 366)
```

Print Statistics for TOBS

Print Statistics for TOBS

Temp Observed

```
In [14]: S=STAT['TOBS']
         for key in ['mean', 'std', 'low100', 'high100']:
             element=S[key]
             print(key,'=',end='')
             if type(element)==numpy.float64 or type(element)==numpy.float16:
                 print('%6.2f'%element)
             elif type(element)==numpy.ndarray:
                 print (element)
             else:
                 print('unidentified type=',type(element))

         mean =   8.31
         std =  10.31
         low100 =-11.70
         high100 =  30.59
```

TMAX
TMIN

Celsius

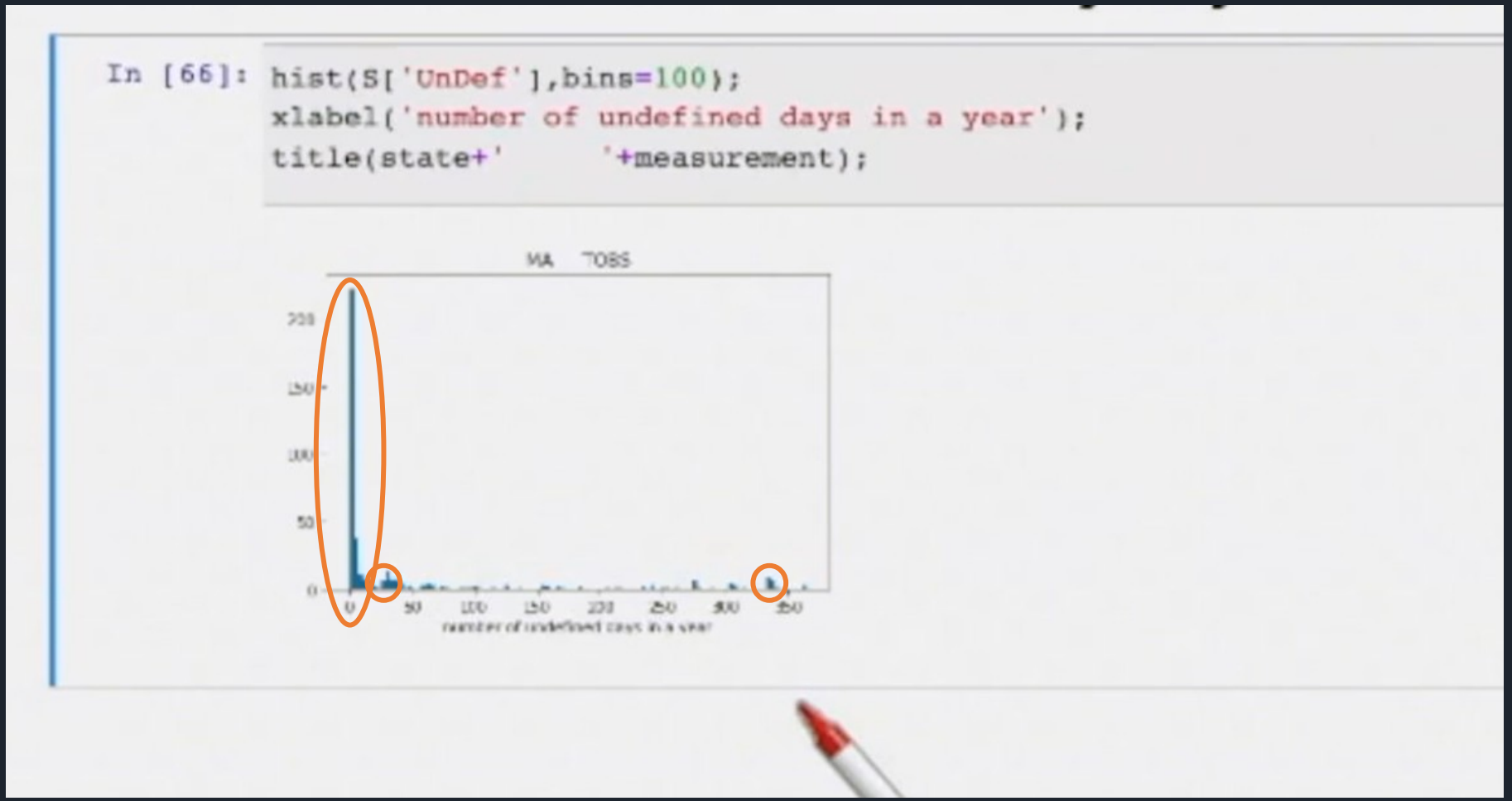
Print Statistics for TOBS

In [15]:
```
%pylab inline
measurement='TOBS'
Sobs=STAT[measurement]['SortedVals']

#figure(figsize=[15,10])
n_obs=Sobs.shape[0]
p=arange(0,1,1/n_obs)
plot(Sobs,p)
title('CDF of '+measurement)
grid()
```

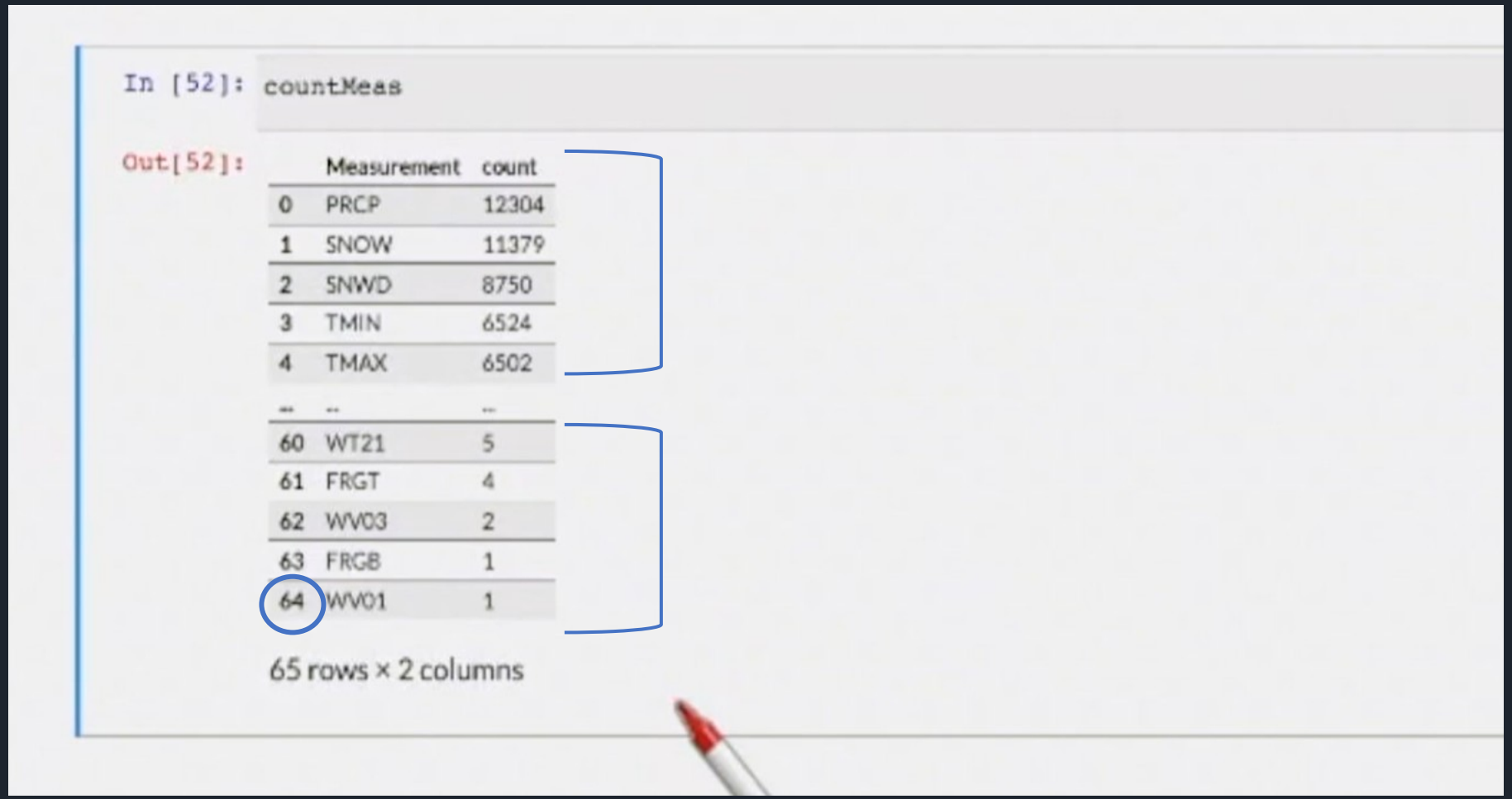
```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```

Distribution of Undefined Elements in Yearly Measurements

Distribution of Undefined Elements in Yearly Measurements



Distribution of Undefined Elements in Yearly Measurements



In [52]: countMeas

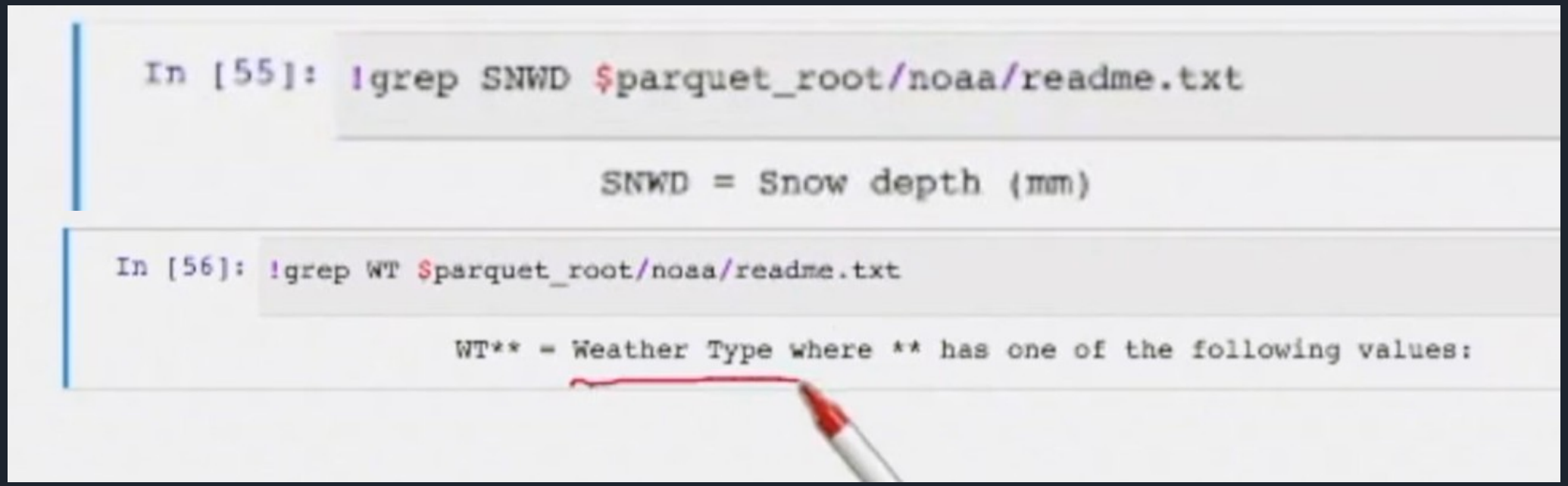
Out[52]:

	Measurement	count
0	PRCP	12304
1	SNOW	11379
2	SNWD	8750
3	TMIN	6524
4	TMAX	6502
...	...	...
60	WT21	5
61	FRGT	4
62	WV03	2
63	FRGB	1
64	WV01	1

65 rows × 2 columns

Find Out the Definition of Measurement Types

Find Out the Definition of Measurement types



Find the Number of Measurements (years) for Each Station in State

Find the Number of Measurements (years) for Each Station in State

```
In [ ]: Query="""
        select Station,count(Station) as count
        from weather
        WHERE state='%s' and Measurement='TOBS'
        GROUP BY Station
        ORDER BY count DESC
        """%state
        #print(Query)
        tmp=sqlContext.sql(Query)
        print(tmp.count())
        tmp.show(3)
```

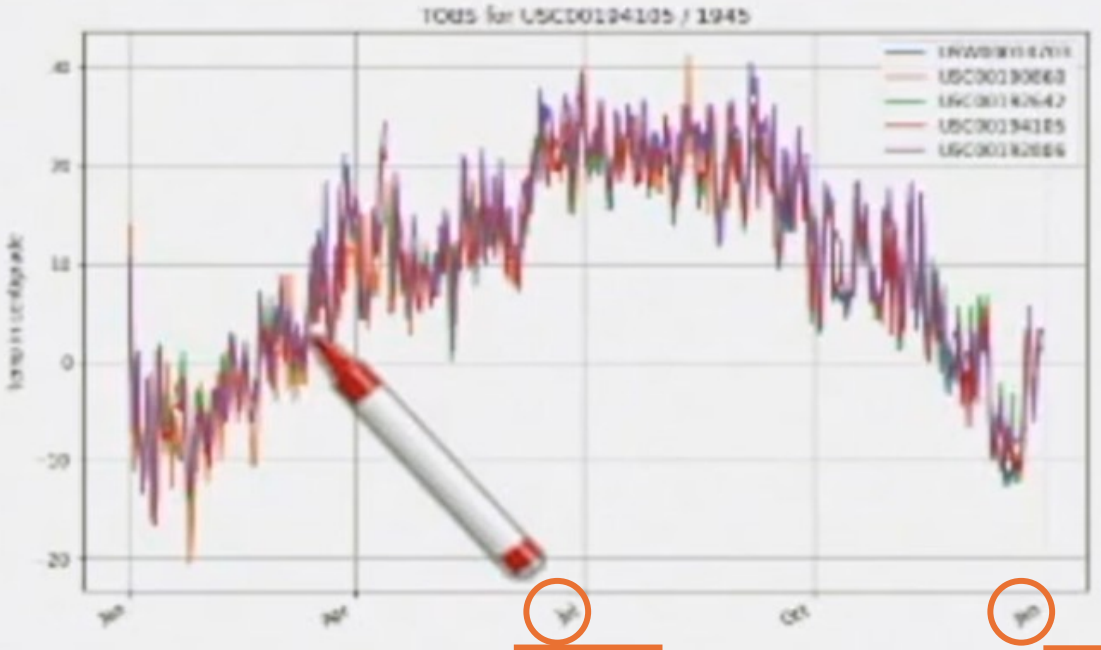
Get All Measurements for year=1945, measurement="TOBS" and state="MA"

In [60]:
```
year=1945
Query="""
SELECT *
FROM weather
WHERE Year='%s'
and Measurement='TOBS'
and State='%s'
"""%(year,state)
df=sqlContext.sql(Query)
print(df.count())
df.show(3)
```

```
35
+-----------+-----------+----+--------------------+--------+---------+---------+-----------+--------------------+-----+-------------+
|    Station|Measurement|Year|              Values|latitude|longitude|elevation| dist2coast|                name|state|      country|
+-----------+-----------+----+--------------------+--------+---------+---------+-----------+--------------------+-----+-------------+
|USW00014703|       TOBS|1945|[43 00 D4 FF A...|    42.2| -72.5333|     75.0|    97.3125|CHICOPEE FALLS WE...|   MA|United States|
|USC00190860|       TOBS|1945|[8B 00 EF FF 91...| 42.0475| -71.0081|     22.9|  21.046875|            BROCKTON|   MA|United States|
|USC00192642|       TOBS|1945|[53 00 C3 FF FA F...| 41.7167| -71.1333|     57.9|2.447265625|          FALL RIVER|   MA|United States|
+-----------+-----------+----+--------------------+--------+---------+---------+-----------+--------------------+-----+-------------+
only showing top 3 rows
```

5 of the 35 ; TOBS measurements

```
In [62]: from lib.YearPlotter import YearPlotter
         k=5
         _title='TOBS for %s / %d'%(station,year)
         fig, ax = plt.subplots(figsize=_figsize);
         YP=YearPlotter()
         YP.plot(M[:k,:366].T,fig,ax,title=_title,labels=_labels)# ,labels=labels);
         ylabel('temp in centigrade');
```

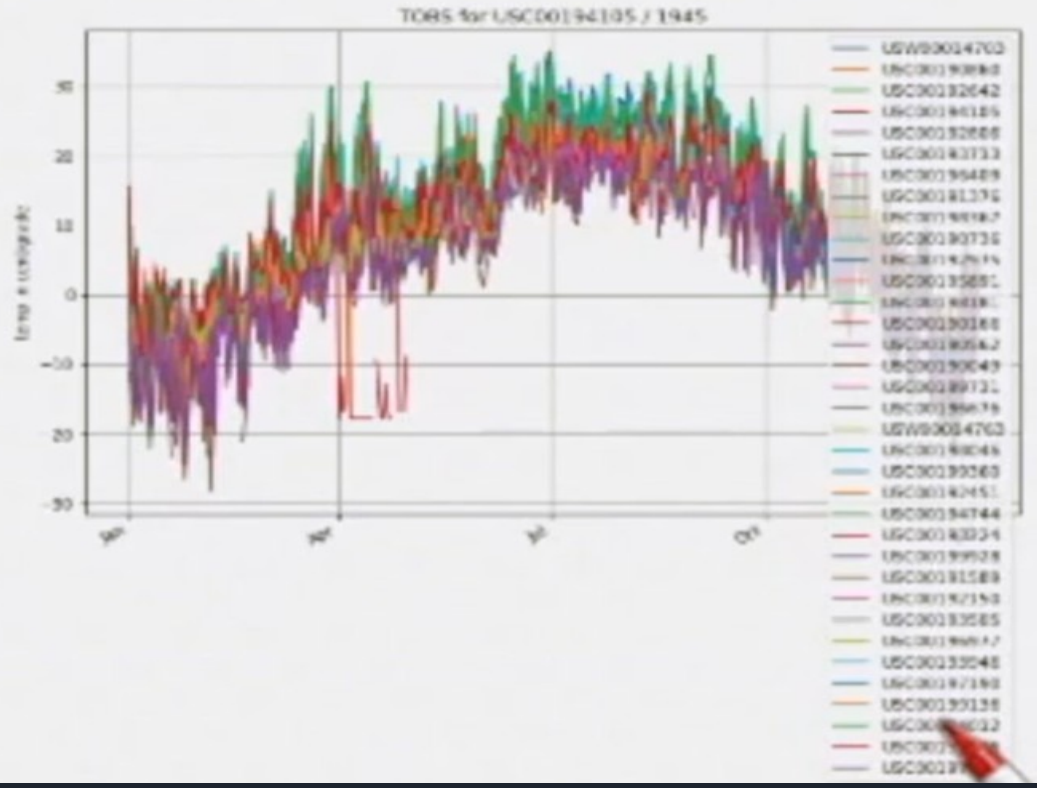


July

Jan

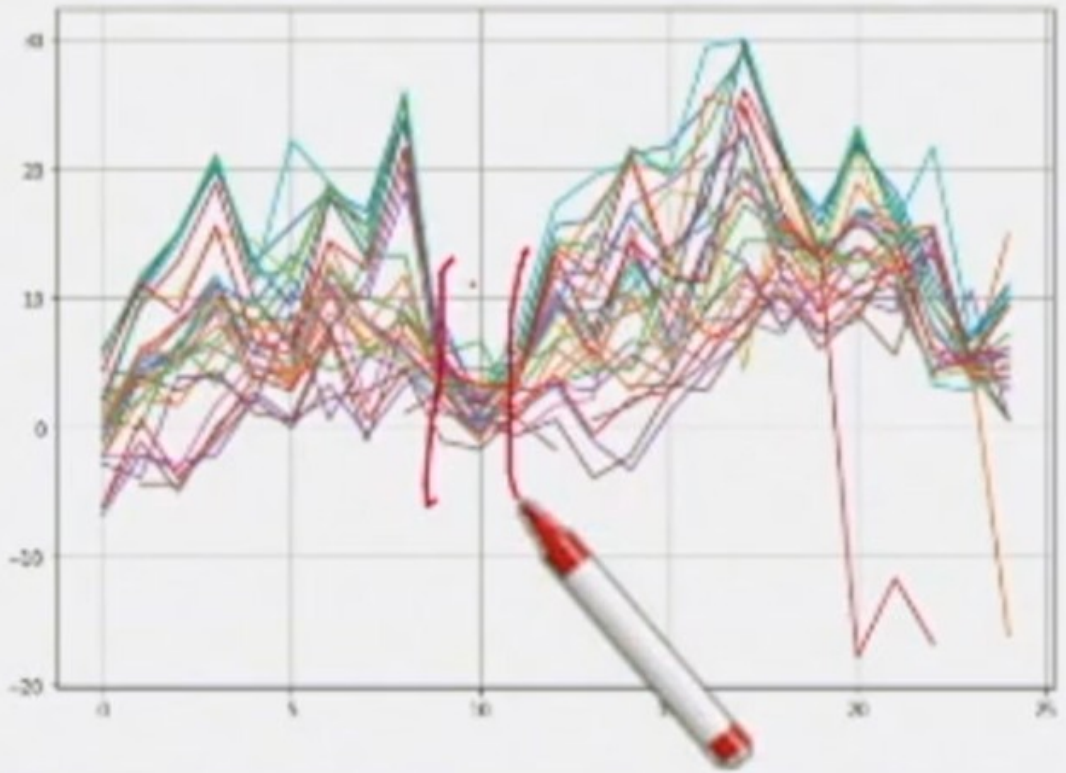
Plot all 35 the TOBS measurements

```
In [29]: #looking at all of the year creates a mess
         from lib.YearPlotter import YearPlotter
         k=35
         fig, ax = plt.subplots(figsize=_figsize);
         YP=YearPlotter()
         YP.plot(M[:k,:366].T,fig,ax,title=_title,labels=_labels)# ,labels=labels);
         ylabel('temp in centigrade');
```



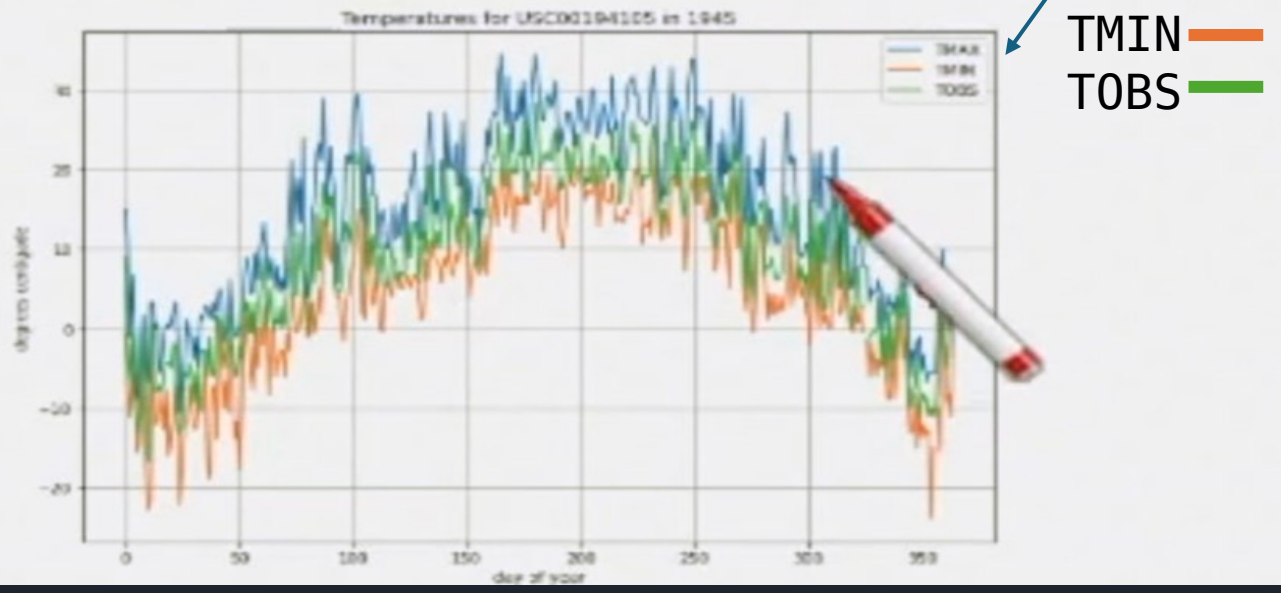
Zooming Into a Small Number of Days the Correlations between Stations are Clear

In [64]:
```
## Looking at a small range of days  across all stations in state reveals
## Interesting clustering
figure(figsize=_figsize)
plot(M[:,70:95].transpose());
grid()
```



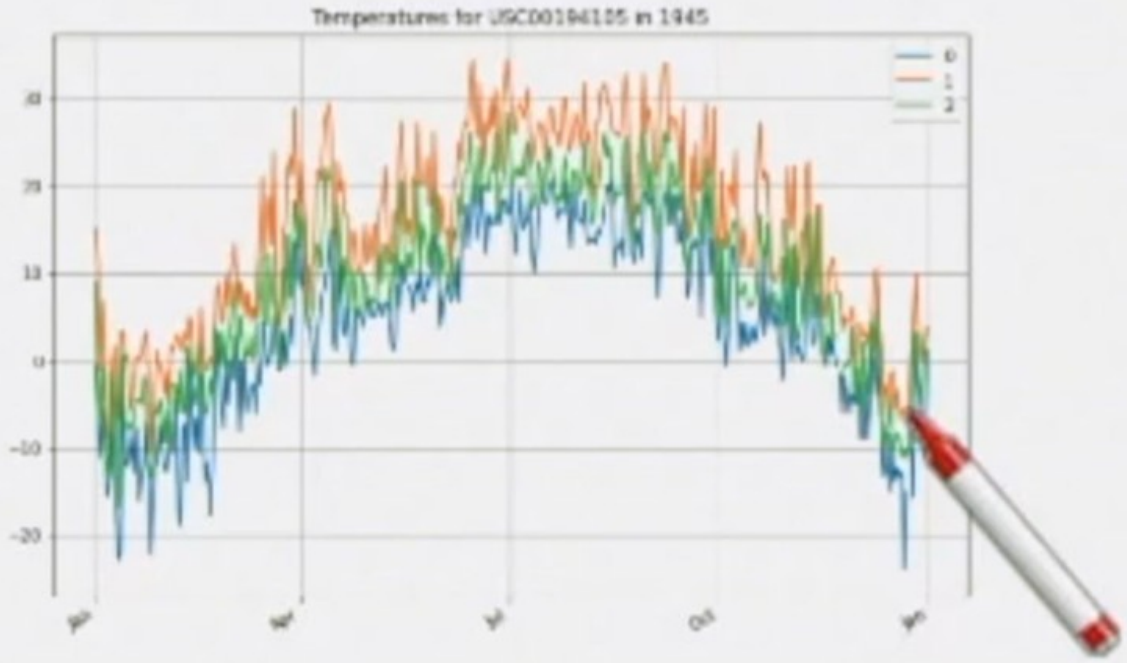
How Different Measurements Behave, Relate to Each Other

```
In [32]: _tmax=unpackArray(pandas_df.loc['TMAX','Values'],np.int16)/10.
         _tmin=unpackArray(pandas_df.loc['TMIN','Values'],np.int16)/10.
         _tobs=unpackArray(pandas_df.loc['TOBS','Values'],np.int16)/10.
         figure(figsize=_figsize)
         plot(_tmax,label='TMAX');
         plot(_tmin,label='TMIN');
         plot(_tobs,label='TOBS');

         xlabel('day of year')
         ylabel('degrees centigade')
         title('Temperatures for %s in %d'%(station,year))
         legend()
         grid()
```



Script for Plotting Yearly Plots

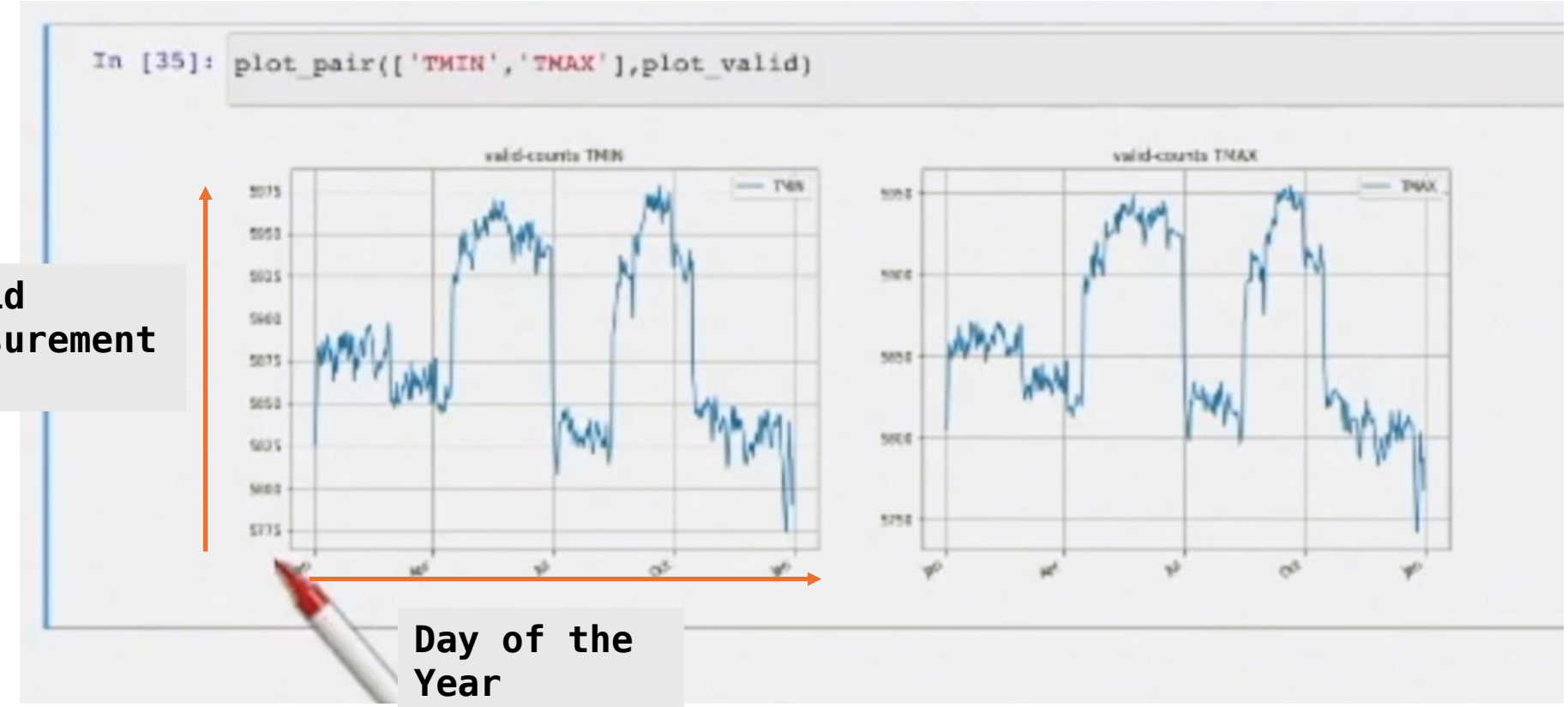
In [33]:
```
from lib.YearPlotter import YearPlotter
T=np.stack([_tmin,_tmax,_tobs])

fig, ax = plt.subplots(figsize=_figsize);
YP=YearPlotter()
YP.plot(T.transpose(),fig,ax,title='Temperatures for %s in %d'%(station,year));
plt.savefig('percipitation.png')
#title('A sample of graphs');
```



Statistics Across the State

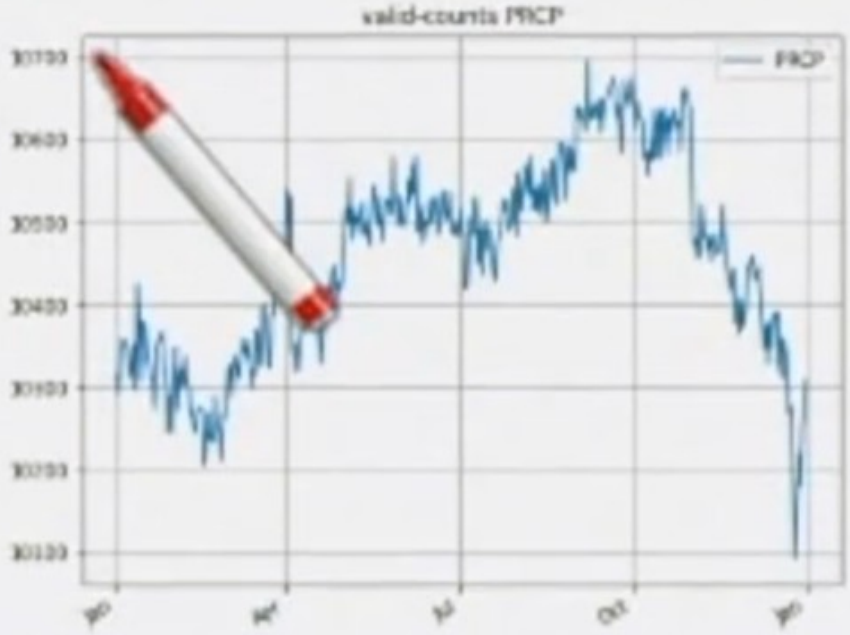
Distribution of Missing Observations

- The distribution of missing observations is not uniform throughout the year. We visualize it below



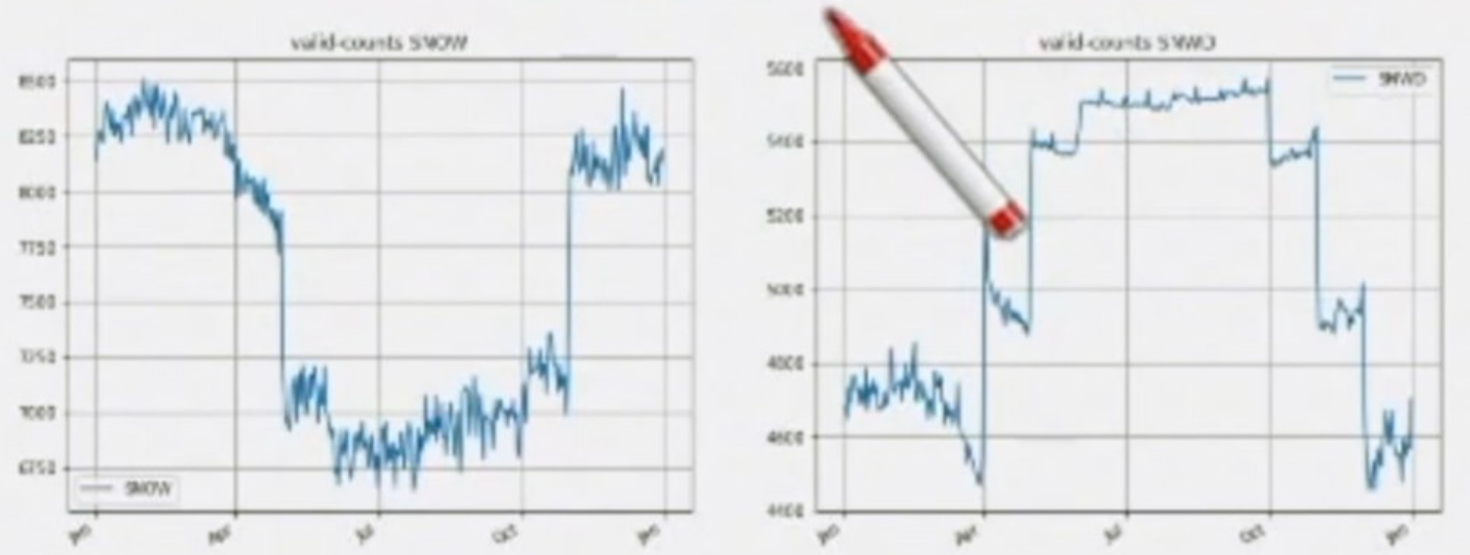
Statistics Across the State:
T Observed & Precipitation

```
In [36]: plot_pair(['TOBS','PRCP'],plot_valid)
```



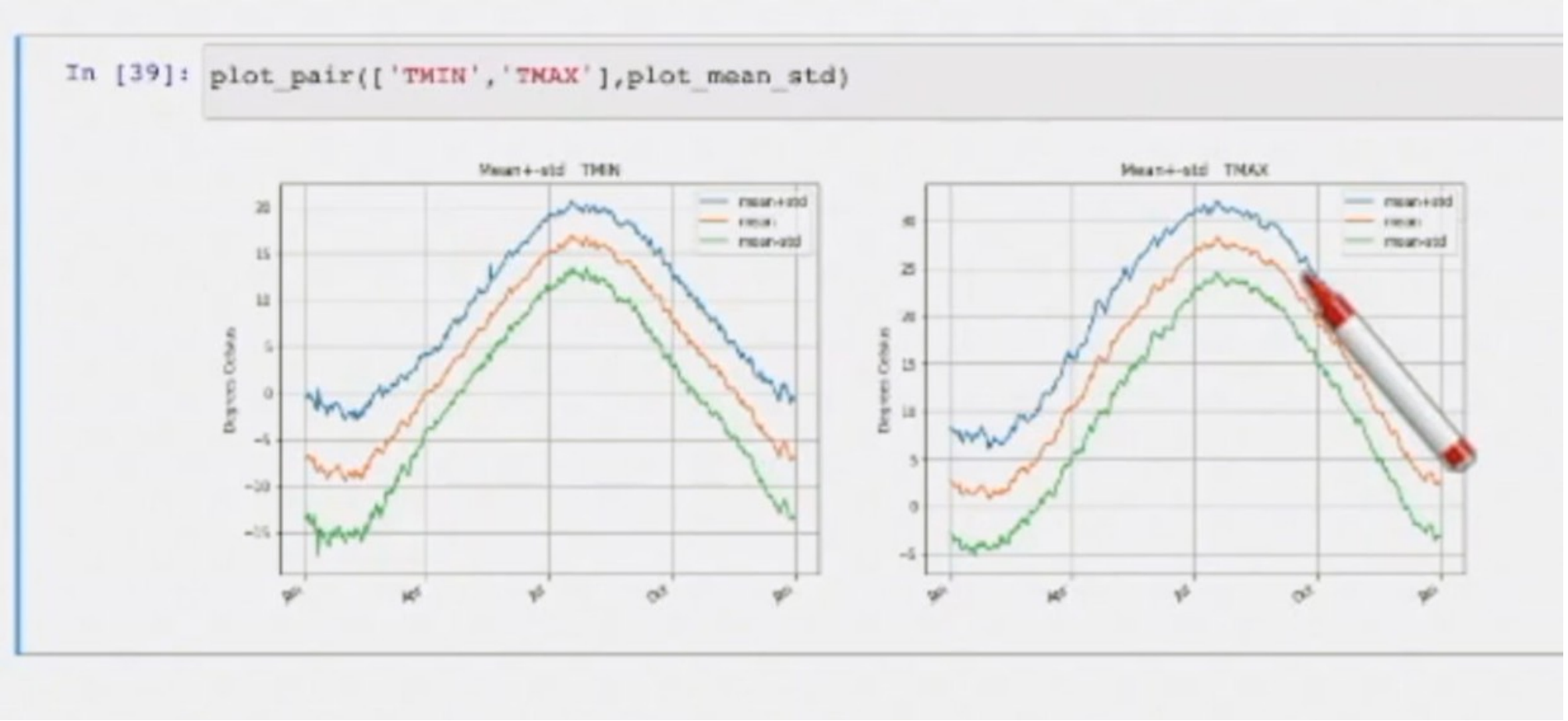
Statistics Across the State: Snow Depth

```
In [37]: # Note that for "SNOW" there are more missing measurements in the summer
         # While for SNWD there are less missing in the summer
         # Question: do these anomalies involve the same stations?
         plot_pair(['SNOW', 'SNWD'],plot_valid)
```

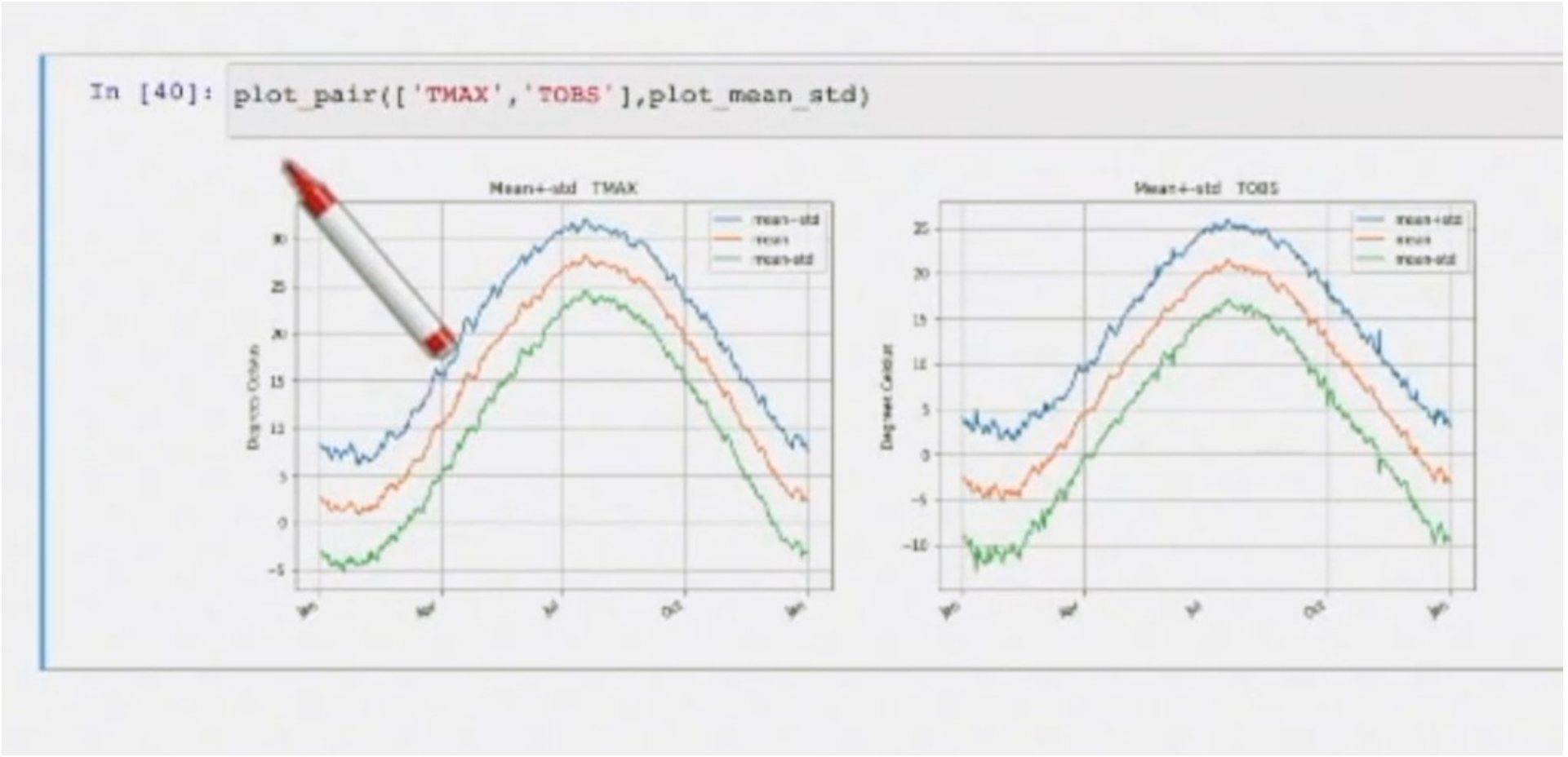


Plots of Mean and Standard Deviation(STD) of Observations

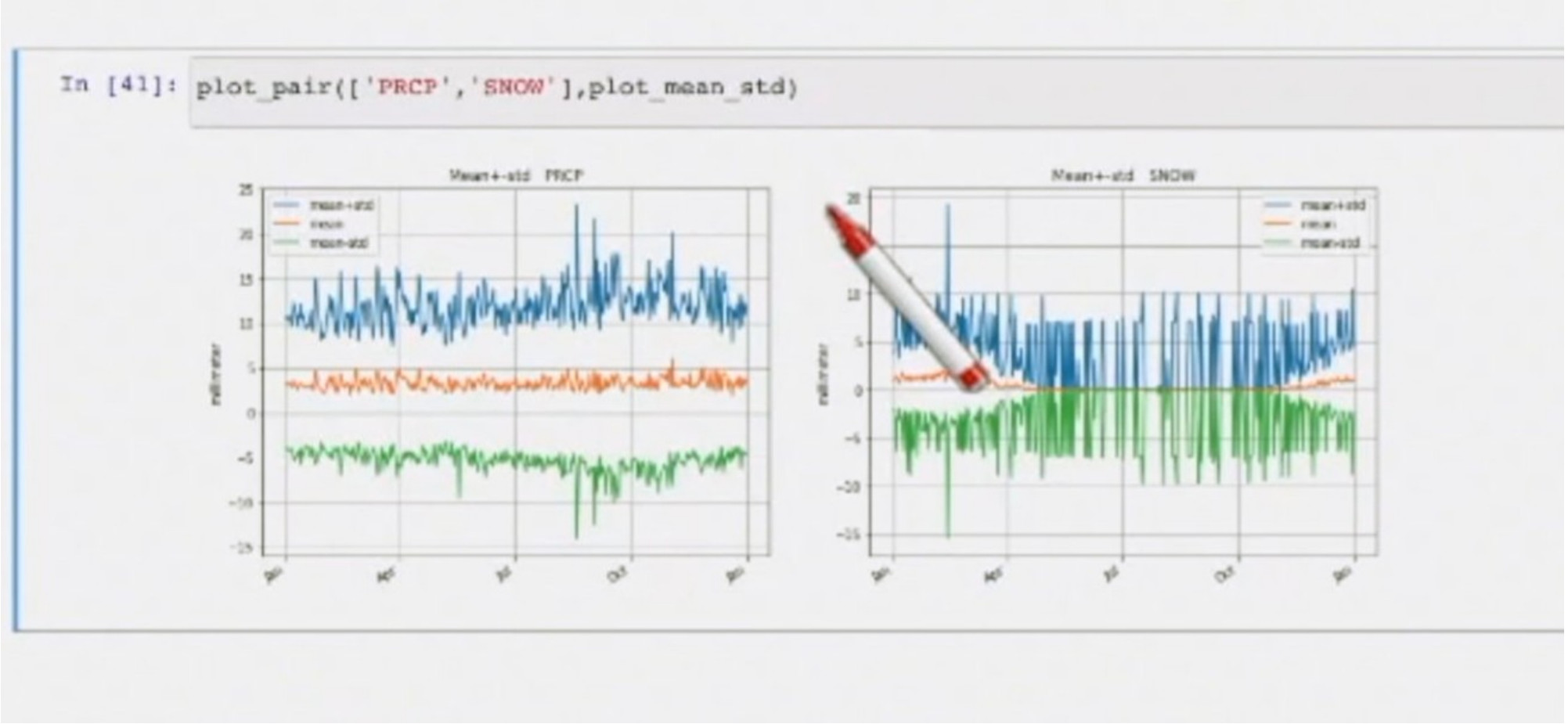
Plots of Mean and STD of Observations



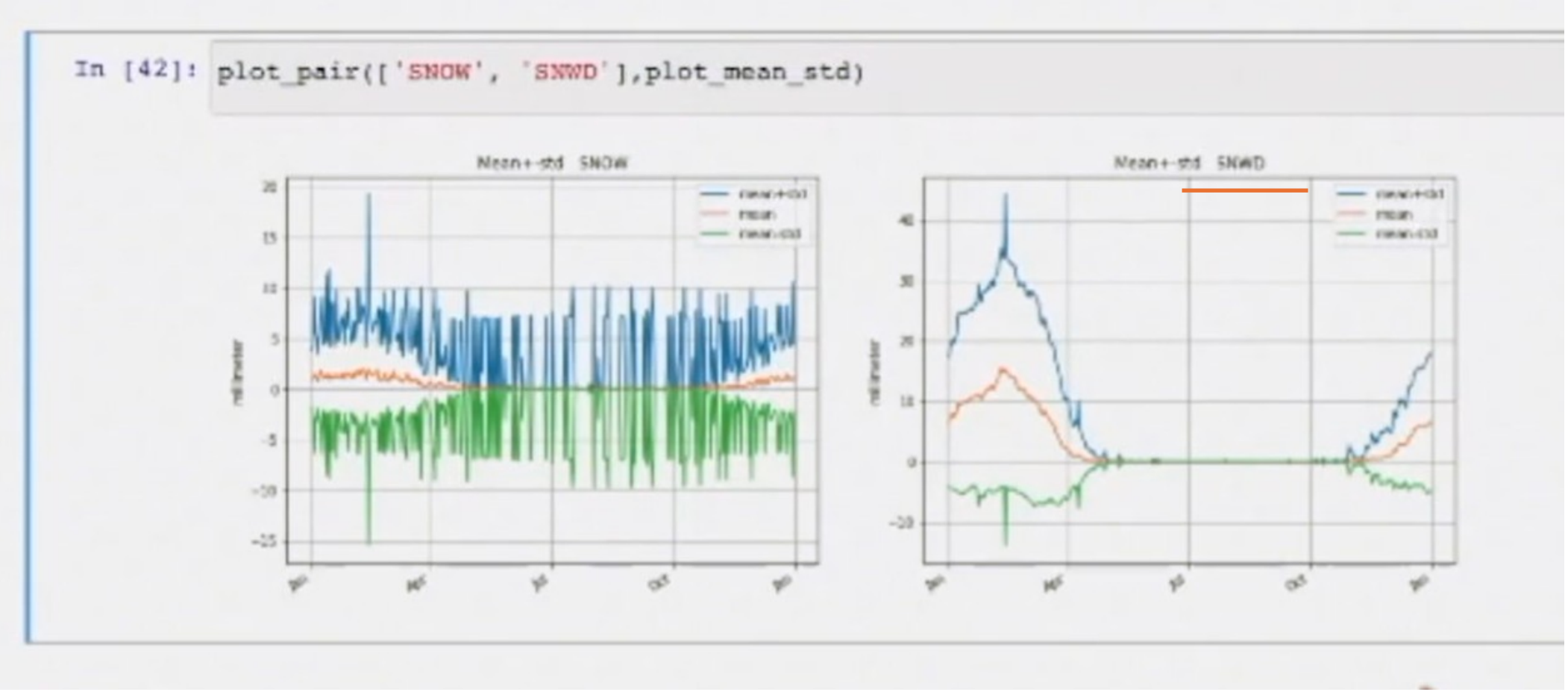
Plots of Mean and STD of Observations



Plots of Mean and STD of Observations (Precipitation and Snow)



Plots of Mean and STD of Observations (Snow Depth)



Conclusion

- We loaded the weather data from Parquet files
- We explored statistics for the data
- We explored where there are a lot of empty cells – limits the accuracy of the statistics
- We visualized different measurements as a function of the day of the year

NEXT: Using PCA for more refined analysis