Instructions:

1. Run import Statements and load in data
2. scroll down and run all helper functions code to lead them into the notebook
3. Begin to run analysis cells from top to bottom (AFTER you have imported/run helper functions)

Data Structures Created:

- **spark_df:** main spark sql context dataframe holding all the data (minus 'labels' column)
- **df:** The main pandas dataframe, complete.
- **station_df:** Info about stations and a groupby count of station occurances.
- **year_data:** Dict of key = measureaments and values the vector years as numpy arrays
- **df_pca:** dataframe for pca analysis, general and holds all the data
- **USC00082220_prcp_meta_df:** all the mata data, including year vectors for this station for the percipitation measurement

Import Libraries and Load Raw Data

In [344]:
```
import sys
sys.path.append('./lib')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")
# sns.set_palette(palette="colorblind")
sns.set_palette(palette="muted")
from numpy_pack import packArray, unpackArray
from spark_PCA import computeCov
from computeStats import computeOverAllDist, STAT_Descriptions
import sklearn as sk
import urllib
import math
%pylab inline
from YearPlotter import YearPlotter
from pickle import load

from ipyleaflet import (Map,
                        Marker,
                        TileLayer,
                        ImageOverlay,
                        Polyline,
                        Polygon,
                        Rectangle,
                        Circle,
                        CircleMarker,
                        GeoJSON,
                        DrawControl)
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
//anaconda/lib/python2.7/site-packages/IPython/core/magics/pylab.py:161:
 UserWarning: pylab import has clobbered these variables: ['Circle', 'Rec
tangle', 'Polygon', 'load']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

In [2]:
```
# Setting up Spark Environment

import findspark
findspark.init()

from pyspark import SparkContext
#sc.stop()
sc = SparkContext(master="local[3]",pyFiles=['lib/numpy_pack.py',
                                             'lib/spark_PCA.py',
                                             'lib/computeStats.py'])


from pyspark import SparkContext
from pyspark.sql import *
sqlContext = SQLContext(sc)
```

In [3]:
```
### Read the data frame from pickle file

data_dir = '../../Data/Weather'
file_index = 'BSSSBSBS'

#read statistics from pickle file
filename = '{}/STAT_{}.pickle'.format(data_dir,file_index)
STAT,STAT_Descriptions = load(open(filename,'rb'))
print 'keys from STAT:',STAT.keys()

#read data from parquet file
filename = '{}/US_Weather_{}.parquet'.format(data_dir,file_index)

spark_df=sqlContext.read.parquet(filename)
print 'spark_df count: ',spark_df.count()
spark_df.show(3)
```

```
keys from STAT: ['TMIN', 'TOBS', 'TMAX', 'SNOW', 'SNWD', 'PRCP']
spark_df count:  12249
+---------+--------+---------+-----------+-----------+------+------------
--------+------+--------+
|elevation|latitude|longitude|measurement|    station|undefs|
  vector|  year|   label|
+---------+--------+---------+-----------+-----------+------+------------
--------+------+--------+
|     14.9| 30.4132| -86.6635|       PRCP|US1FLOK0014|    38|[00 00 00 00
B0 5...|2009.0|BSSSBSBS|
|      6.4| 30.2119| -85.6828|       TMAX|USW00003882|     5|[40 5A F0 5A
80 5...|1999.0|BSSSBSBS|
|      6.4| 30.2119| -85.6828|       TMAX|USW00003882|     3|[20 5B 78 5B
48 5...|2000.0|BSSSBSBS|
+---------+--------+---------+-----------+-----------+------+------------
--------+------+--------+
only showing top 3 rows
```

In [4]:
```
# Column types for spark df
spark_df = spark_df.drop('label')
spark_df.printSchema()
```

```
root
 |-- elevation: double (nullable = true)
 |-- latitude: double (nullable = true)
 |-- longitude: double (nullable = true)
 |-- measurement: string (nullable = true)
 |-- station: string (nullable = true)
 |-- undefs: long (nullable = true)
 |-- vector: binary (nullable = true)
 |-- year: double (nullable = true)
```

In [5]:
```
#count number of occuring weather stations
spark_df.groupby('station').agg({'station':'count'}).show()
```

```
+-----------+--------------+
|    station|count(station)|
+-----------+--------------+
|USW00063899|            15|
|USC00013255|           185|
|USC00013251|           211|
|US1FLWT0002|             2|
|USR0000FNAV|             8|
|USC00228382|           113|
|US1ALMB0018|             6|
|US1ALMB0037|             3|
|USC00081388|           108|
|USW00003882|            33|
|USC00012813|           450|
|USW00003852|            75|
|US1FLWS0001|             2|
|USW00063869|            18|
|US1FLES0005|             5|
|USR0000ABNS|            14|
|USC00012832|            45|
|US1ALBW0041|             4|
|US1FLOK0016|             4|
|USC00086129|            52|
+-----------+--------------+
only showing top 20 rows
```

In [260]:
```
# df is the main df for the data
df = spark_df.toPandas()
df.head()
```

Out[260]:

	elevation	latitude	longitude	measurement	station	undefs	vector	year
0	14.9	30.4132	-86.6635	PRCP	US1FLOK0014	38	[0, 0, 0, 0, 176, 91, 0, 66, 0, 126, 96, 86, 0...	2009.0
1	6.4	30.2119	-85.6828	TMAX	USW00003882	5	[64, 90, 240, 90, 128, 88, 128, 81, 224, 80, 8...	1999.0
2	6.4	30.2119	-85.6828	TMAX	USW00003882	3	[32, 91, 120, 91, 72, 91, 152, 90, 0, 88, 184,...	2000.0
3	6.4	30.2119	-85.6828	TMAX	USW00003882	40	[144, 85, 224, 84, 160, 83, 160, 86, 8, 89, 12...	2001.0
4	6.4	30.2119	-85.6828	TMAX	USW00003882	12	[224, 84, 48, 84, 48, 84, 224, 85, 128, 88, 96...	2002.0

In [261]:
```
df.tail(3)
```

Out[261]:

	elevation	latitude	longitude	measurement	station	undefs	vector	year
12246	74.7	30.7244	-86.0939	SNWD	USC00082220	0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	2007.0
12247	74.7	30.7244	-86.0939	SNWD	USC00082220	0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	2008.0
12248	74.7	30.7244	-86.0939	SNWD	USC00082220	0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	2009.0

Station location Count Analysis

Looking at the location of stations and the occurances of recordings. Also looking into some high level stats about each station.

In [262]:
```
station_df = df[['latitude','longitude','station','elevation']].copy()
station_df.head()
```

Out[262]:

	latitude	longitude	station	elevation
0	30.4132	-86.6635	US1FLOK0014	14.9
1	30.2119	-85.6828	USW00003882	6.4
2	30.2119	-85.6828	USW00003882	6.4
3	30.2119	-85.6828	USW00003882	6.4
4	30.2119	-85.6828	USW00003882	6.4

In [263]:
```
station_df['count'] = station_df.groupby('station')['station'].transform('co
station_df.drop_duplicates(inplace=True)
station_df.reset_index(inplace=True)
print station_df.shape
station_df.head(5)
```

```
(177, 6)
```

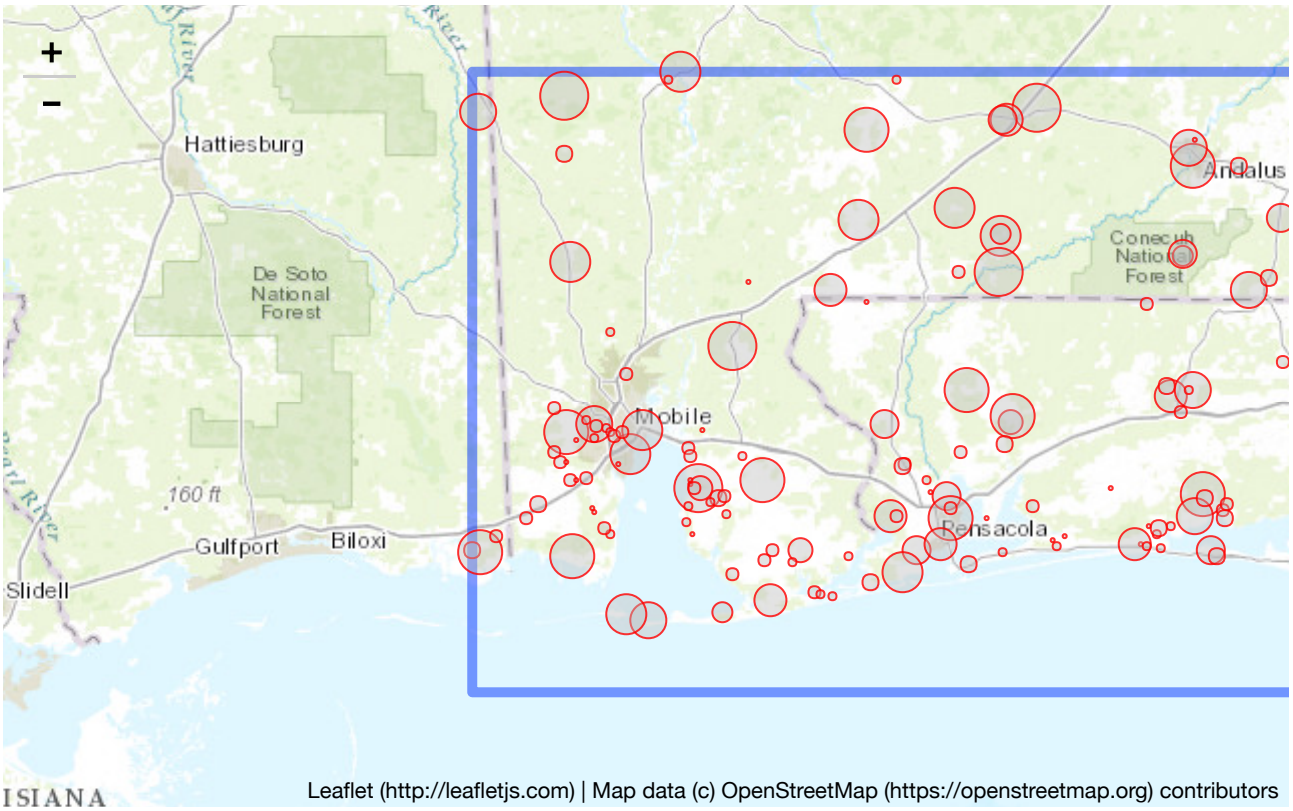
Out[263]:

	index	latitude	longitude	station	elevation	count
0	0	30.4132	-86.6635	US1FLOK0014	14.9	1
1	1	30.2119	-85.6828	USW00003882	6.4	33
2	23	31.2975	-85.8997	USC00012675	102.7	220
3	60	31.1819	-87.4389	USC00010402	91.4	270
4	108	31.3089	-86.3939	USW00053843	94.5	12

In [264]:
```
max_lat = station_df['latitude'].unique().max()
min_lat = station_df['latitude'].unique().min()
max_long = station_df['longitude'].unique().max()
min_long = station_df['longitude'].unique().min()
```

In [18]:

```
center = [(min_lat+max_lat)/2, (min_long+max_long)/2]
zoom = 9

url1 = 'http://server.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/M
provider = TileLayer(url = url1, opacity=1.0)

# station_map = Map(default_tiles=TileLayer(opacity=1.0), center=center, zoo
station_map = Map(default_tiles=provider, center=center, zoom=zoom)
boundary = Rectangle(bounds=[[min_lat,min_long],[max_lat,max_long]], weight=
station_map += boundary

lat_margin=(max_lat-min_lat)/4
long_margin=(max_long-min_long)/4

circles = []
markers = []
for index,row in station_df.iterrows():
    _lat=row['latitude']
    _long=row['longitude']
    _count=row['count']
    _station_id=row['station']
    _elevation=row['elevation']
    # taking sqrt of count so that the  area of the circle corresponds to th
    circle_obj = Circle(location=(_lat,_long), radius=int(1000*np.log(_count
            color='#F00', opacity=0.8, fill_opacity=0.4,
            fill_color='#bbb')
    marker_obj = Marker(location =(_lat,_long), opacity=0.0, title = '{}, el
    markers.append(marker_obj)
    circles.append(circle_obj)
    station_map.add_layer(circle_obj)
    station_map.add_layer(marker_obj)
station_map
```



```
Widget Javascript not detected.  It may not be installed or enabled prope
rly.
```

Station Level Analysis

In [98]:
```
# df['year'].value_counts()
# print df['year'].max()
# print df['year'].min()
df.groupby(['year'])
```

Out[98]:
```
<pandas.core.groupby.DataFrameGroupBy object at 0x120233890>
```

In [270]:
```
# test the differences between these 3 stations
s1 = ['USC00018223','US1FLSR0002','US1FLSR0004']

#distance from US1FLSR0002 --> US1FLSR0004 = 11.5 km
#distance from US1FLSR0002 --> USC00018223 = 138 km
```

In [271]:
```
station_df.loc[station_df['station'].isin(s1)]
```

Out[271]:

	index	latitude	longitude	station	elevation	count
103	5211	30.6513	-87.0408	US1FLSR0002	49.1	6
147	6854	30.6340	-87.1593	US1FLSR0004	57.0	6
152	6967	31.3333	-88.2500	USC00018223	70.1	8

In [277]:
```
# Unpack all the year vector data as numpy array and store it in a dict cal
# all data in 10ths of a degree C - https://earthscience.stackexchange.com/q
year_data_3 = {}
for m in STAT.keys():
    tmp_spark_df = spark_df.filter(spark_df.measurement == m)
    rows = tmp_spark_df.rdd.map(lambda row: np.append(unpackArray(row['vecto
    year_data_3[m] = np.vstack(rows)

print year_data_3.keys()
```

```
['TMIN', 'TOBS', 'TMAX', 'SNOW', 'SNWD', 'PRCP']
```

In [306]:
```
tmax_station_df = pd.DataFrame.from_dict(year_data_3['PRCP'])
tmax_station_df.rename(columns = {365:'station'}, inplace=True)
# tmax_station_df.iloc[:,:364] = tmax_station_df.iloc[:,:364].applymap(lambd

tmax_station_df.head()
```

Out[306]:

	0	1	2	3	4	5	6	7	8	9	...	356	357	358	359	36
0	0.0	0.0	246.0	3.0	nan	102.0	nan	0.0	0.0	157.0	...	nan	442.0	0.0	0.0	8.
1	nan	nan	nan	nan	nan	nan	nan	nan	0.0	nan	...	0.0	0.0	173.0	0.0	0.
2	0.0	0.0	0.0	61.0	0.0	nan	221.0	0.0	0.0	0.0	...	0.0	0.0	264.0	0.0	na
3	0.0	0.0	0.0	0.0	0.0	0.0	nan	91.0	0.0	0.0	...	0.0	0.0	0.0	61.0	0.
4	36.0	135.0	0.0	0.0	28.0	239.0	0.0	0.0	0.0	30.0	...	160.0	0.0	0.0	114.0	48.

5 rows × 366 columns

In [330]:
```
tmax_station_df_subset = tmax_station_df.loc[tmax_station_df['station'].isin
tmax_station_df_subset
# tmax_station_df_subset.groupby(['station']).mean()
```

Out[330]:

	0	1	2	3	4	5	6	7	8	9	...	356	357	358	35
920	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	...	3.0	0.0	124.0	8.
921	0.0	0.0	0.0	310.0	15.0	3.0	163.0	0.0	0.0	0.0	...	0.0	8.0	437.0	5.
922	20.0	5.0	0.0	0.0	0.0	0.0	0.0	36.0	5.0	0.0	...	0.0	0.0	0.0	130.
923	56.0	320.0	3.0	0.0	15.0	300.0	0.0	0.0	0.0	23.0	...	203.0	0.0	nan	36.
924	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	64.0	168.
2146	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	3.0	51.0	3.
2147	0.0	0.0	3.0	292.0	30.0	3.0	107.0	0.0	0.0	3.0	...	0.0	13.0	371.0	0.
2148	8.0	8.0	0.0	0.0	0.0	0.0	0.0	41.0	0.0	0.0	...	0.0	nan	0.0	130.
2149	86.0	399.0	0.0	0.0	8.0	277.0	0.0	0.0	0.0	28.0	...	130.0	0.0	0.0	46.
2150	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	61.0	152.
2259	0.0	66.0	94.0	15.0	0.0	13.0	0.0	0.0	0.0	0.0	...	nan	89.0	0.0	76.

In [331]:
```
tmax_station_df_subset['station'].value_counts()
```

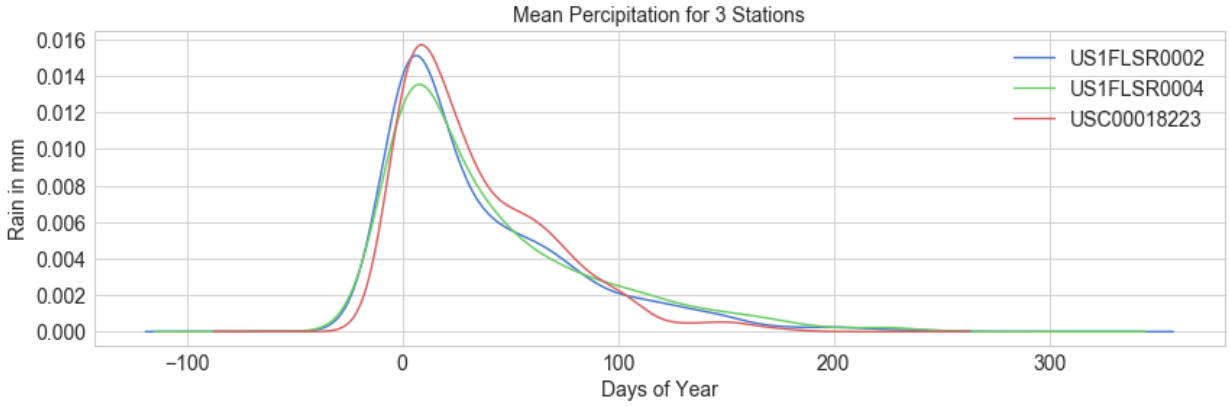
Out[331]:
```
USC00018223    8
US1FLSR0004    5
US1FLSR0002    5
Name: station, dtype: int64
```

In [332]:
```
tmax_station_df_subset.iloc[:,:364] = tmax_station_df_subset.iloc[:,:364].as
# tmax_station_df_subset.groupby(['station']).count()
```

```
/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  if __name__ == '__main__':
```

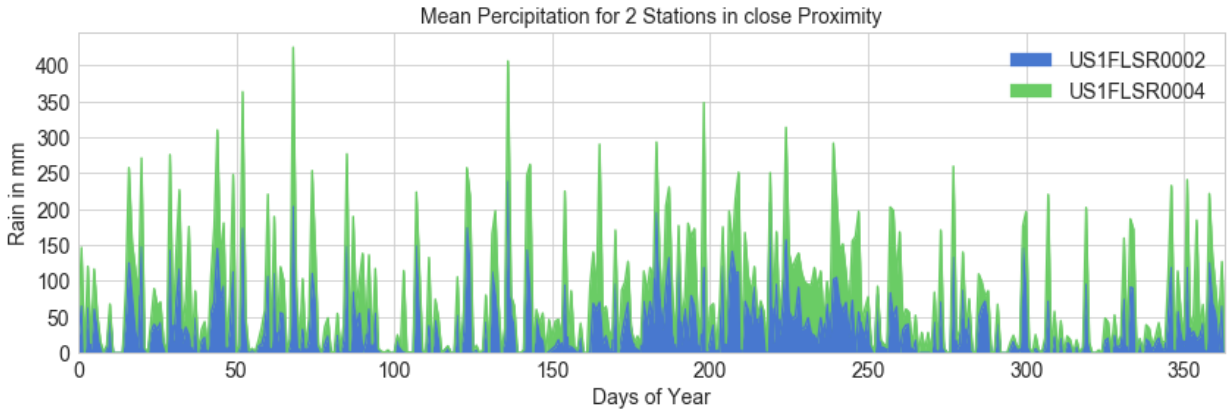
In [333]:
```
tmax_station_df_subset = tmax_station_df_subset.fillna(0).groupby(['station'
```

In [376]:
```
# tmax_station_df_subset.T.plot()

ax = tmax_station_df_subset.T.plot(kind='density',figsize=(14,4), fontsize=1
# ax.plot(epoch_std)
ax.set_xlabel("Days of Year", fontsize=14)
ax.set_ylabel("Rain in mm", fontsize=14)
ax.set_title("Mean Percipitation for 3 Stations", fontsize=14)
plt.legend(loc='best', fontsize=14)
plt.show()
```

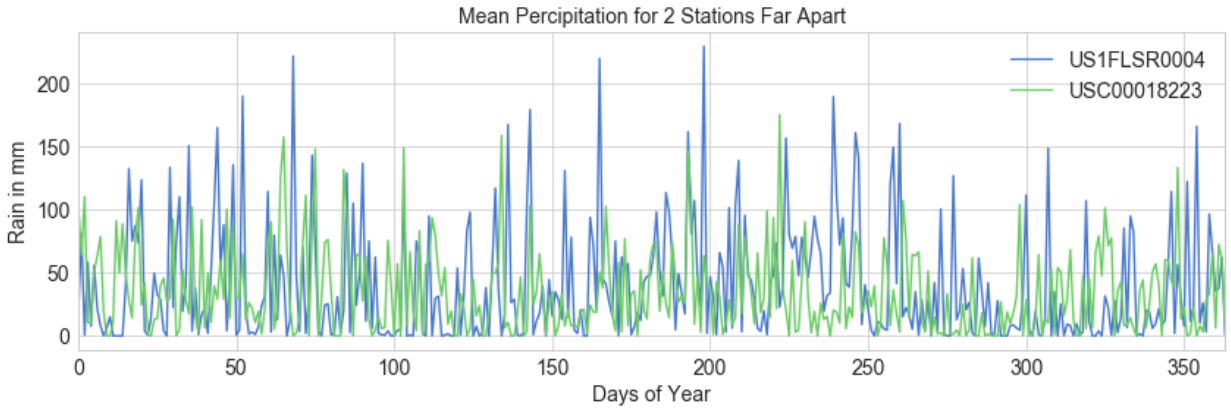


In [ ]:
```
ax = tmax_station_df_subset.T.plot(figsize=(14,4), fontsize=14)
# ax.plot(epoch_std)
ax.set_xlabel("Days of Year", fontsize=14)
ax.set_ylabel("Rain in mm", fontsize=14)
ax.set_title("Mean Percipitation for 3 Stations", fontsize=14)
plt.legend(loc='best', fontsize=14)
plt.show()
```

In [373]:
```
ax = tmax_station_df_subset.iloc[0:2].T.plot(kind='area',figsize=(14,4), for
ax.set_xlabel("Days of Year", fontsize=14)
ax.set_ylabel("Rain in mm", fontsize=14)
ax.set_title("Mean Percipitation for 2 Stations in close Proximity", fontsi
plt.legend(loc='best', fontsize=14)
plt.show()
```



In [346]:
```
ax = tmax_station_df_subset.iloc[1:3].T.plot(figsize=(14,4), fontsize=14)
ax.set_xlabel("Days of Year", fontsize=14)
ax.set_ylabel("Rain in mm", fontsize=14)
ax.set_title("Mean Percipitation for 2 Stations Far Apart", fontsize=14)
plt.legend(loc='best', fontsize=14)
plt.show()
```



In [352]:
```
tmax_station_df_subset
```

Out[352]:

station	0	1	2	3	4	5	6	7	8	9	...	354	355
US1FLSR0002	15.200	65.000	0.6	62.0	6.0	60.600	32.6	7.80	1.000	4.6	...	18.4	22.8
US1FLSR0004	18.800	81.400	0.6	58.4	7.6	56.000	21.4	8.20	0.000	6.2	...	166.2	10.2
USC00018223	102.625	63.875	110.5	10.5	10.5	51.125	63.5	78.75	19.375	0.0	...	0.0	8.0

3 rows × 364 columns

In [365]:
```
tmax_station_df_subset.iloc[1:3]
```

Out[365]:

station	0	1	2	3	4	5	6	7	8	9	...	354	355
US1FLSR0004	18.800	81.400	0.6	58.4	7.6	56.000	21.4	8.20	0.000	6.2	...	166.2	10.2
USC00018223	102.625	63.875	110.5	10.5	10.5	51.125	63.5	78.75	19.375	0.0	...	0.0	8.0

2 rows × 364 columns

In [361]:
```
import scipy.stats as stats
stats.kruskal(tmax_station_df_subset.iloc[0],tmax_station_df_subset.iloc[1])
```

Out[361]:
```
KruskalResult(statistic=1.4696584194115518, pvalue=0.22539959515258889)
```

In [363]:
```
stats.kruskal(tmax_station_df_subset.iloc[1],tmax_station_df_subset.iloc[2])
```

Out[363]:
```
KruskalResult(statistic=1.1206440692479795e-06, pvalue=0.9991553557519454
7)
```

In [353]:
```
stats.f_oneway(tmax_station_df_subset.iloc[0],tmax_station_df_subset.iloc[1
```

Out[353]:
```
F_onewayResult(statistic=1.4359230383981525, pvalue=0.23119124780827763)
```

In [359]:
```
stats.mannwhitneyu(tmax_station_df_subset.iloc[0],tmax_station_df_subset.ilo
```

Out[359]:
```
MannwhitneyuResult(statistic=62549.0, pvalue=0.095826128877260164)
```

In [366]:
```
stats.mannwhitneyu(tmax_station_df_subset.iloc[1],tmax_station_df_subset.ilo
```

Out[366]:
```
MannwhitneyuResult(statistic=62813.5, pvalue=0.11273356818902058)
```

In [357]:
```
ttest = stats.ttest_ind(tmax_station_df_subset.iloc[0],tmax_station_df_subse

# ttest=stats.ttest_ind(old,new)
print 't-test independent', ttest
```

```
t-test independent Ttest_indResult(statistic=0.54842634705145643, pvalue=
0.58356778822314359)
```

In [ ]:
```

```

Unpacking vector year data into seperate numpy arrays

In [379]:
```
# Unpack all the year vector data as numpy array and store it in a dict cal
# all data in 10ths of a degree C - https://earthscience.stackexchange.com/
year_data = {}
for m in STAT.keys():
    tmp_spark_df = spark_df.filter(spark_df.measurement == m)
    rows = tmp_spark_df.rdd.map(lambda row: unpackArray(row['vector'],np.flo
    year_data[m] = np.vstack(rows)

print year_data.keys()
```

```
['TMIN', 'TOBS', 'TMAX', 'SNOW', 'SNWD', 'PRCP']
```

In [380]:
```
total = 0
for k in year_data.keys():
    rows_of_data = len(year_data[k])
    print 'rows of data for - {} - {}'.format(k, rows_of_data)
    total += rows_of_data
print total
```

```
rows of data for - TMIN - 2021
rows of data for - TOBS - 1344
rows of data for - TMAX - 2020
rows of data for - SNOW - 2107
rows of data for - SNWD - 1973
rows of data for - PRCP - 2784
12249
```

In [381]:
```
#create df and convert all 10ths of C to F
df_tmin = pd.DataFrame(year_data['TMIN']).applymap(lambda x: convert_C_to_F(
print df_tmin.shape
df_tmin.head()
```

```
(2021, 365)
```

Out[381]:

	0	1	2	3	4	5	6	7	8	9	...	355	356	357	358	359	360
0	38.1	43.7	33.1	29.2	27.0	28.1	34.8	40.3	34.2	29.8	...	39.2	35.9	32.6	31.4	29.8	35.9
1	45.3	49.2	48.7	40.9	34.8	34.8	39.8	39.2	47.0	42.6	...	31.4	29.8	32.6	35.3	35.3	37.6
2	29.8	28.7	29.2	31.4	29.8	34.8	33.1	37.0	33.7	29.2	...	36.4	44.2	37.6	35.3	33.1	30.3
3	34.2	34.2	32.6	29.8	29.8	38.7	34.8	32.0	32.0	39.2	...	40.9	44.2	45.3	36.4	34.2	33.1
4	44.2	40.3	34.2	32.6	33.1	37.0	32.6	37.6	38.7	39.2	...	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 365 columns

Univariate Analysis

In [383]:
```
#create dfs for the temp data and convert C --> F
df_tmin = pd.DataFrame(year_data['TMIN']).applymap(lambda x: convert_C_to_F(
df_tmax = pd.DataFrame(year_data['TMAX']).applymap(lambda x: convert_C_to_F(
df_tobs = pd.DataFrame(year_data['TOBS']).applymap(lambda x: convert_C_to_F(
```

In [ ]:

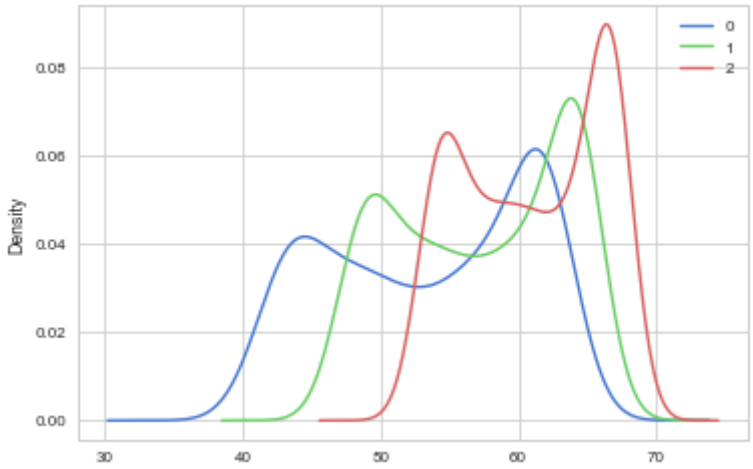
In [127]:
```
# plot all of the yearly TEMP data
plt.figure(figsize=(15,15))

plt.subplot(3,1,1)
plt.plot(mean_std(df_tmin))
plt.title('Daily Min Temp', fontsize=18)
plt.ylabel('Temp in F', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.subplot(3,1,2)
plt.plot(mean_std(df_tmax))
plt.title('Daily Max Temp', fontsize=18)
plt.ylabel('Temp in F', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.subplot(3,1,3)
plt.plot(mean_std(df_tobs))
plt.title('Daily Avg Temp', fontsize=18)
plt.ylabel('Temp in F', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.tight_layout()
plt.show()
```

In [388]:
```
pd.DataFrame(mean_std(df_tmin)).plot(kind='')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-388-d86019107cfc> in <module>()
----> 1 pd.DataFrame(mean_std(df_tmin)).plot(kind='hexbin')

//anaconda/lib/python2.7/site-packages/pandas/tools/plotting.pyc in __call__(self, x, y, kind, ax, subplots, sharex, sharey, layout, figsize, use_index, title, grid, legend, style, logx, logy, loglog, xticks, yticks, xlim, ylim, rot, fontsize, colormap, table, yerr, xerr, secondary_y, sort_columns, **kwds)
   3772                             fontsize=fontsize, colormap=colormap, table=table,
   3773                             yerr=yerr, xerr=xerr, secondary_y=secondary_y,
-> 3774                             sort_columns=sort_columns, **kwds)
   3775     __call__.__doc__ = plot_frame.__doc__
   3776
```

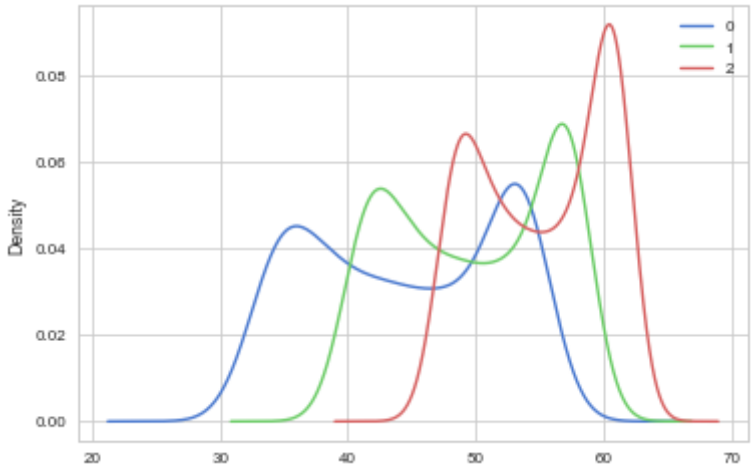
In [386]:
```
pd.DataFrame(mean_std(df_tmax)).plot(kind='kde')
```

Out[386]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x181611cd0>
```

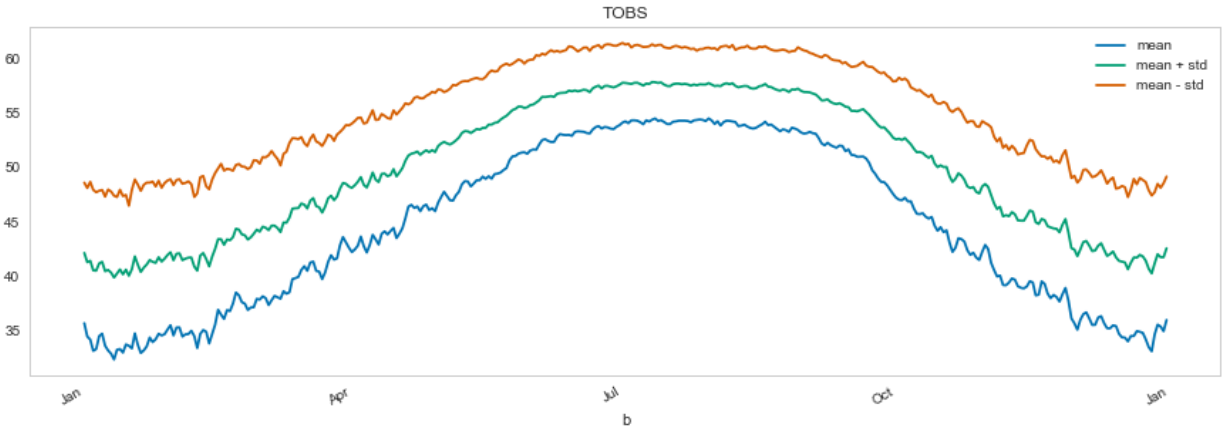


In [387]:
```
pd.DataFrame(mean_std(df_tobs)).plot(kind='kde')
```

Out[387]: <matplotlib.axes._subplots.AxesSubplot at 0x17162a110>



In [177]:
```
fig, ax = plt.subplots(figsize=(15,5));
YP=YearPlotter()
YP.plot(mean_std(df_tobs),fig,ax, labels=['mean','mean + std','mean - std'],
```



In [178]:
```
# make dfs for snow
df_snow = pd.DataFrame(year_data['SNOW'])
df_snwd = pd.DataFrame(year_data['SNWD'])
```

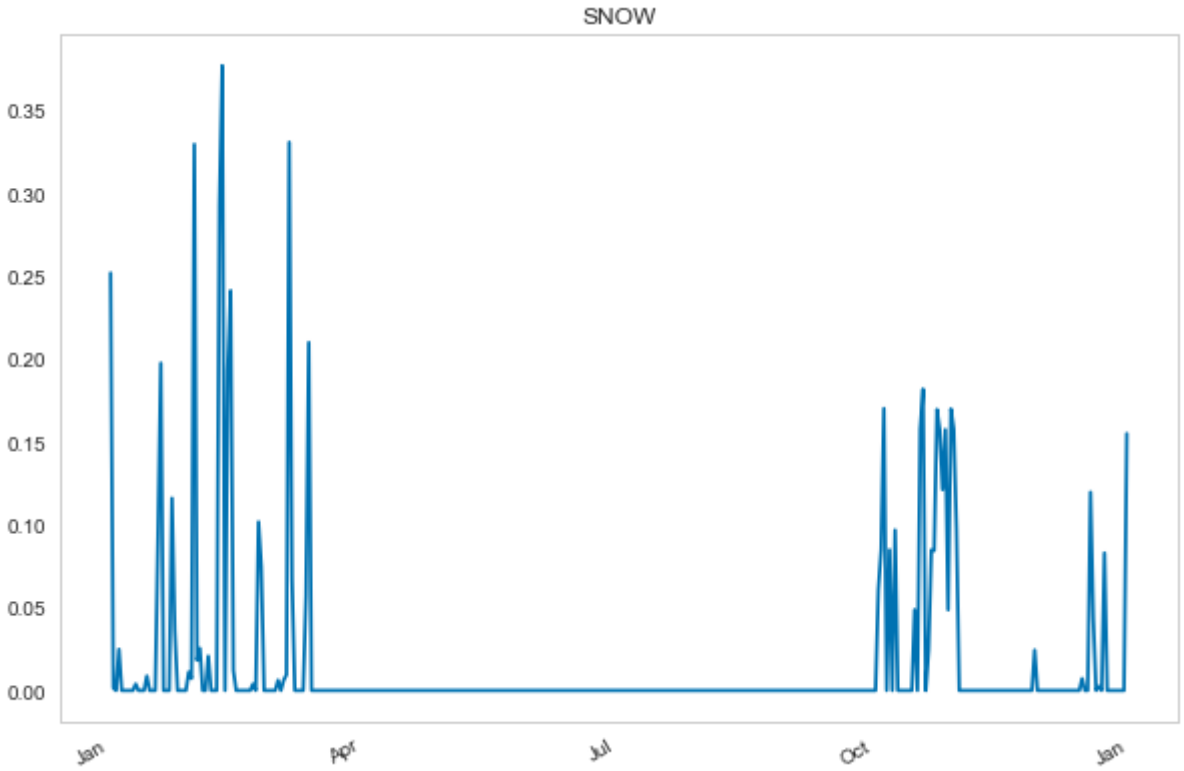
In [179]:
```
pct_zero_col = (df_snow[df_snow == 0].count(axis=0)/len(df_snow.index)).mean
print 'Percentage of Zeroes in the Snow (SNOW) Data = {}%'.format(round(100*
pct_zero_col = (df_snwd[df_snwd == 0].count(axis=0)/len(df_snwd.index)).mean
print 'Percentage of Zeroes in the Snow (SNWD) Data = {}%'.format(round(100*
```

```
Percentage of Zeroes in the Snow (SNOW) Data = 99.08%
Percentage of Zeroes in the Snow (SNWD) Data = 99.05%
```

In [181]:
```
df_snow.shape
```

Out[181]: (2107, 365)

In [163]:
```
fig, ax = plt.subplots(figsize=(10,7));
YP=YearPlotter()
YP.plot(df_snow.mean(),fig,ax,title='SNOW')
```

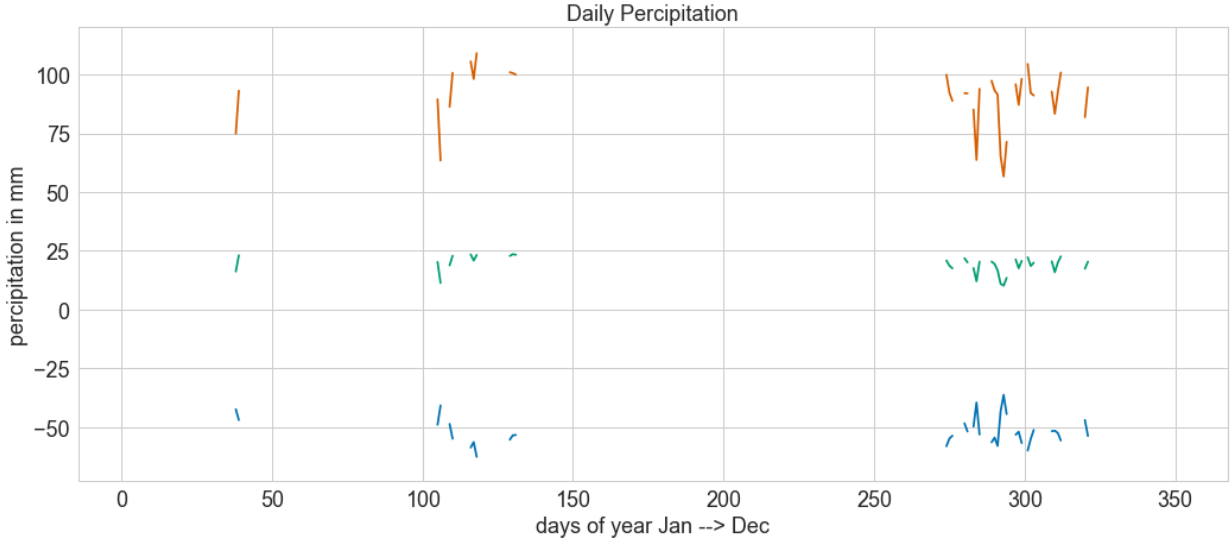


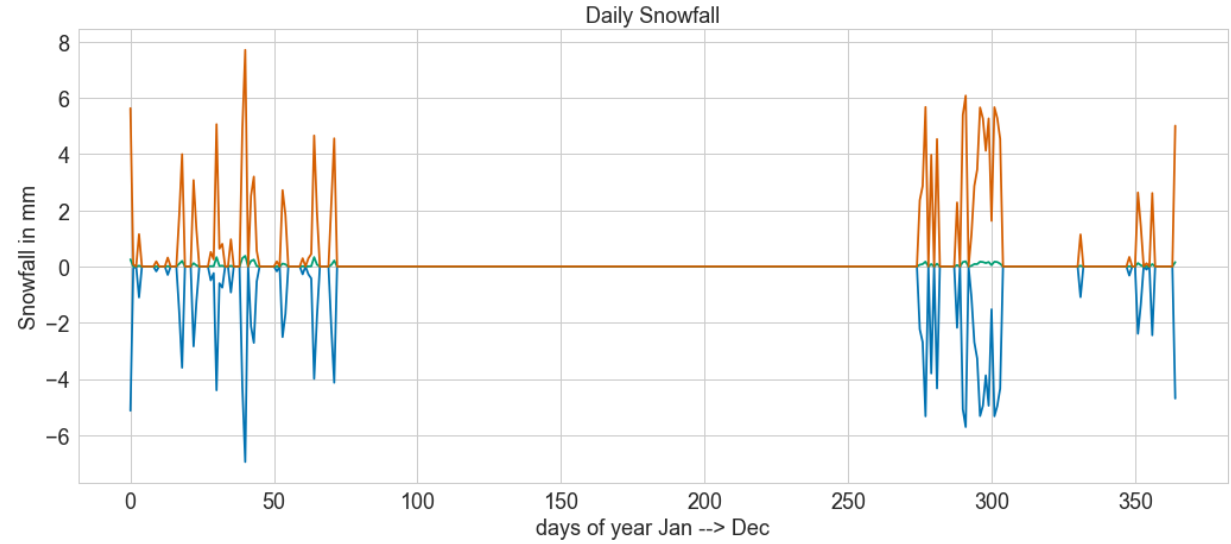
In [*]:
```
plt.figure(figsize=(15,5))

plt.subplot(2,1,1)
plt.plot(df_snow.mean())
plt.title('Daily Avg Snow', fontsize=18)
plt.ylabel('Snowfall in mm', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.subplot(2,1,2)
plt.plot(df_snwd.mean())
plt.title('Daily Avg Snow Depth', fontsize=18)
plt.ylabel('Snow Depth in mm', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.tight_layout()
plt.show()
```

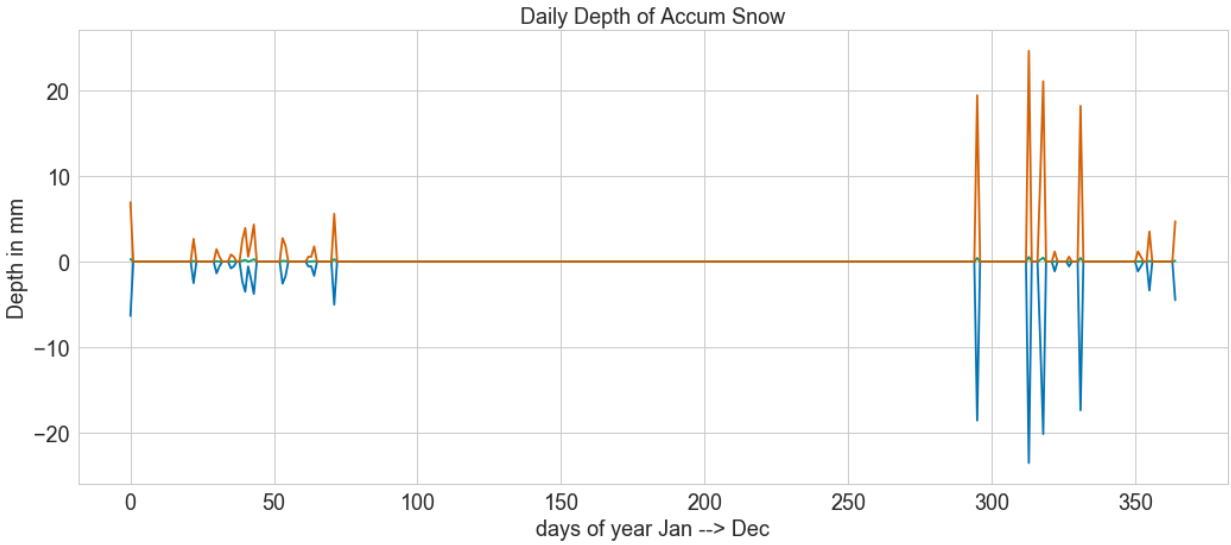
In [ ]:
```
```

In [128]:
```
#create dfs for the percipitation data and leave units in mm
df_prcp = pd.DataFrame(year_data['PRCP'])
```

In [132]:
```
# plot all of the yearly percipitation/snow
plt.figure(figsize=(15,20))

plt.subplot(3,1,1)
plt.plot(mean_std(df_prcp))
plt.title('Daily Percipitation', fontsize=18)
plt.ylabel('percipitation in mm', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.subplot(3,1,2)
plt.plot(mean_std(df_snow))
plt.title('Daily Snowfall', fontsize=18)
plt.ylabel('Snowfall in mm', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.subplot(3,1,3)
plt.plot(mean_std(df_snwd))
plt.title('Daily Depth of Accum Snow', fontsize=18)
plt.ylabel('Depth in mm', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)

plt.tight_layout()
plt.show()
```


Daily Percipitation
percipitation in mm
100
75
50
25
0
−25
−50
0
50
100
150
200
250
300
350
days of year Jan --> Dec


Daily Snowfall
Snowfall in mm
8
6
4
2
0
−2
−4
−6
0
50
100
150
200
250
300
350
days of year Jan --> Dec


Daily Depth of Accum Snow
Depth in mm
20
10
0
−10
−20
0
50
100
150
200
250
300
350
days of year Jan --> Dec

In [134]:
```
df_prcp.head(10)
```

Out[134]:

	0	1	2	3	4	5	6	7	8	9	...	355	356	357	358	
0	0.0	0.0	246.0	3.0	NaN	102.0	NaN	0.0	0.0	157.0	...	0.0	NaN	442.0	0.0	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	...	0.0	0.0	0.0	173.0	
2	0.0	0.0	0.0	61.0	0.0	NaN	221.0	0.0	0.0	0.0	...	NaN	0.0	0.0	264.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	NaN	91.0	0.0	0.0	...	0.0	0.0	0.0	0.0	(
4	36.0	135.0	0.0	0.0	28.0	239.0	0.0	0.0	0.0	30.0	...	23.0	160.0	0.0	0.0	1
5	0.0	0.0	0.0	0.0	333.0	3.0	348.0	8.0	0.0	0.0	...	0.0	25.0	0.0	36.0	
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.0	0.0	23.0	20.0	
7	3.0	0.0	0.0	0.0	0.0	0.0	20.0	0.0	0.0	0.0	...	0.0	0.0	0.0	71.0	
8	3.0	0.0	0.0	0.0	0.0	0.0	8.0	8.0	0.0	10.0	...	0.0	0.0	244.0	439.0	
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	23.0	0.0	0.0	43.0	

10 rows × 365 columns

In [148]:
```
df_prcp.shape[0]*df_prcp.shape[1]/float(df_prcp.isnull().sum().sum())
```

Out[148]:
```
44.14823825867837
```

In [140]:
```
pct_zero_col = (df_prcp[df_prcp == 0].count(axis=0)/len(df_prcp.index)).mean
print 'Percentage of Zeroes in the Percipitation Data = {}%'.format(round(10

# pct_nan_col = (df_prcp[df_prcp == NaN].count(axis=0)/len(df_prcp.index)).m
# print 'Percentage of Zeroes in the Percipitation Data = {}%'.format(pct_na
df_prcp.isnull().sum().sum()
```

```
Percentage of Zeroes in the Percipitation Data = 70.18%
```

Out[140]:
```
23017
```

In [ ]:
```

```

In [ ]:
```

```

PCA Analysis for Percepitation

In [193]:
```
df_pca = df.copy()
```

In [194]:
```
tmp_df = df_pca[df_pca['measurement'] == 'PRCP']
tmp_df['station'].value_counts().index[0]
```

Out[194]:
```
u'USC00082220'
```

In [195]:
```
#creatng meta df for station from above valuecounts()
USC00082220_prcp_meta_df = tmp_df[tmp_df['station'] == 'USC00082220']
print USC00082220_prcp_meta_df.shape
USC00082220_prcp_meta_df.head(10)
```

```
(97, 8)
```

Out[195]:

	elevation	latitude	longitude	measurement	station	undefs	vector	year
7045	74.7	30.7244	-86.0939	PRCP	USC00082220	4	[0, 0, 0, 0, 96, 86, 0, 0, 0, 0, 0, 0, 0, 0, 0...	1897.0
7046	74.7	30.7244	-86.0939	PRCP	USC00082220	0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192...	1898.0
7047	74.7	30.7244	-86.0939	PRCP	USC00082220	8	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 126, 0, 0, 1...	1899.0
7048	74.7	30.7244	-86.0939	PRCP	USC00082220	0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1900.0
7049	74.7	30.7244	-86.0939	PRCP	USC00082220	3	[120, 95, 64, 85, 192, 84, 0, 0, 0, 0, 0, 0, 0...	1901.0
7050	74.7	30.7244	-86.0939	PRCP	USC00082220	5	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1902.0
7051	74.7	30.7244	-86.0939	PRCP	USC00082220	19	[0, 0, 72, 88, 0, 0, 240, 87, 144, 85, 0, 0, 0...	1903.0
7052	74.7	30.7244	-86.0939	PRCP	USC00082220	1	[0, 0, 176, 85, 0, 73, 0, 0, 0, 0, 0, 0, 0, 168, ...	1904.0
7053	74.7	30.7244	-86.0939	PRCP	USC00082220	45	[0, 0, 192, 81, 0, 0, 0, 0, 0, 0, 0, 20, 93, 0, 0...	1905.0
7054	74.7	30.7244	-86.0939	PRCP	USC00082220	23	[0, 126, 0, 126, 48, 95, 0, 126, 0, 126, 0, 12...	1906.0

In [196]:
```
#query sql context rdd for prcp for station from above valuecounts() to unpa
Query="SELECT * FROM weather\n\tWHERE measurement='%s' and station='%s'"%('P
print Query
tmp_df = sqlContext.sql(Query)
print tmp_df.count(),'rows'
tmp_df.show(2)
rows=tmp_df.rdd.map(lambda row:unpackArray(row['vector'],np.float16)).collec
USC00082220_prcp=np.vstack(rows)
USC00082220_prcp=USC00082220_prcp
shape(USC00082220_prcp)
```

```
SELECT * FROM weather
        WHERE measurement='PRCP' and station='USC00082220'
97 rows
+---------+--------+---------+-----------+-----------+------+------------
--------+------+--------+
|elevation|latitude|longitude|measurement|    station|undefs|
  vector|  year|   label|
+---------+--------+---------+-----------+-----------+------+------------
--------+------+--------+
|     74.7| 30.7244| -86.0939|       PRCP|USC00082220|     4|[00 00 00 00
60 5...|1897.0|BSSSBSBS|
|     74.7| 30.7244| -86.0939|       PRCP|USC00082220|     0|[00 00 00 00
00 0...|1898.0|BSSSBSBS|
+---------+--------+---------+-----------+-----------+------+------------
--------+------+--------+
only showing top 2 rows
```

Out[196]: (97, 365)

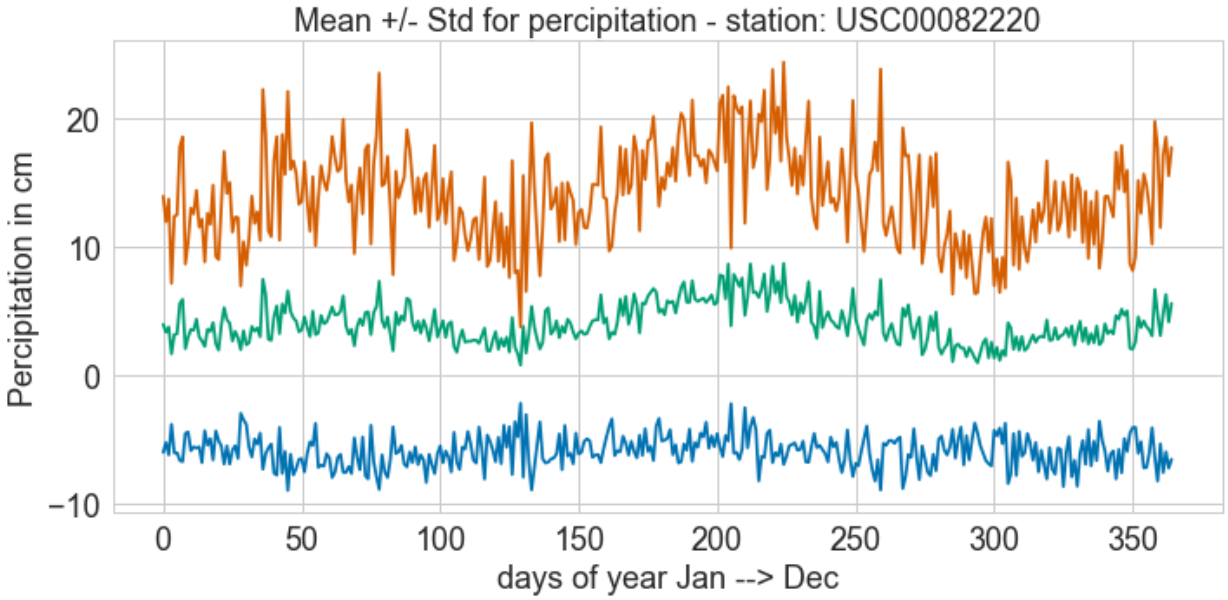
In [197]:
```
#creating year vector dataframe for weather station from above for PCA analy
USC00082220_prcp_df = pd.DataFrame(USC00082220_prcp)
USC00082220_prcp_df.head()
```

Out[197]:

	0	1	2	3	4	5	6	7	8	9	...	355	356	357	358	359	36
0	0.0	0.0	102.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	229.0	102.0	0.0	0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	76.0	0.0	102.0	...	102.0	0.0	0.0	38.0	0.0	0
2	0.0	0.0	0.0	0.0	0.0	NaN	0.0	13.0	0.0	216.0	...	0.0	0.0	254.0	0.0	0.0	0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	18.0	10.0	0.0	0.0	0.0	343
4	478.0	84.0	76.0	0.0	0.0	0.0	0.0	0.0	5.0	30.0	...	0.0	76.0	0.0	0.0	89.0	0

5 rows × 365 columns

In [218]:
```
# plot all of the yearly TEMP data
plt.figure(figsize=(10,5))

plt.subplot(1,1,1)
plt.plot(mean_std_nparray(USC00082220_prcp/10)) #convert to cm
plt.title('Mean +/- Std for percipitation - station: USC00082220', fontsize=
plt.ylabel('Percipitation in cm', fontsize=18)
plt.xlabel('days of year Jan --> Dec', fontsize=18)
plt.tick_params(labelsize=18)


plt.tight_layout()
plt.show()
```



In [225]:
```
USC00082220_prcp_df
```

	0	1	2	3	4	5	6	7	8	9	...	355	356	357	35
12	0.0	0.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	584.0	66.0	0.0	0
13	51.0	25.0	10.0	0.0	0.0	0.0	25.0	533.0	0.0	0.0	...	0.0	0.0	0.0	0
14	0.0	0.0	81.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
15	0.0	0.0	0.0	0.0	0.0	36.0	142.0	0.0	0.0	0.0	...	0.0	0.0	0.0	38
16	0.0	0.0	0.0	0.0	0.0	0.0	5.0	91.0	0.0	0.0	...	NaN	NaN	NaN	Na
17	0.0	0.0	0.0	0.0	0.0	0.0	132.0	0.0	165.0	0.0	...	559.0	0.0	0.0	384
18	0.0	0.0	198.0	0.0	124.0	0.0	122.0	0.0	0.0	0.0	...	5.0	0.0	51.0	0
19	8.0	0.0	0.0	0.0	23.0	224.0	0.0	130.0	18.0	221.0	...	NaN	NaN	NaN	Na
20	0.0	0.0	127.0	13.0	605.0	140.0	503.0	20.0	0.0	0.0	...	0.0	0.0	0.0	53
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	25
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	0.0	0.0	...	0.0	0.0	0.0	0

In [ ]:

epoched temp analysis

In [295]:
```
import array
import struct


tmin = []
df_tmin = df.copy()
```

In [296]:
```
df_tmin = df_tmin[['vector', 'year']]
df_tmin.head()
```

Out[296]:

	vector	year
0	[0, 0, 0, 0, 176, 91, 0, 66, 0, 126, 96, 86, 0...	2009.0
1	[64, 90, 240, 90, 128, 88, 128, 81, 224, 80, 8...	1999.0
2	[32, 91, 120, 91, 72, 91, 152, 90, 0, 88, 184,...	2000.0
3	[144, 85, 224, 84, 160, 83, 160, 86, 8, 89, 12...	2001.0
4	[224, 84, 48, 84, 48, 84, 224, 85, 128, 88, 96...	2002.0

In [306]:
```
for index, row in df_tmin.iterrows():
    tmp_year = []
    tmp_year.append(row[1])
    for n in row[0]:
        tmp_year.append(n)
#         print tmp_year
    tmin.append(tmp_year)
    break
len(tmin)
```

Out[306]: 12252

In [303]:
```
tmin[0]
```

...

In [ ]:
```
year_data_2 = {}
for m in []:
# for m in STAT.keys():
    tmp_spark_df = spark_df.filter(spark_df.measurement == 'TMIN')
    rows = tmp_spark_df.rdd.map(lambda row: unpackArray(row['vector'],np.flo
    print rows
    break
    year_data['TMIN'] = np.vstack(rowsrow['year'])

print year_data_2.keys()
```

In [17]:
```
Unpack all the year vector data as numpy array and store it in a dict called
all data in 10ths of a degree C - https://earthscience.stackexchange.com/ques
ar_data_2 = {}
r m in STAT.keys():
  tmp_spark_df = spark_df.filter(spark_df.measurement == m)
  rows = tmp_spark_df.rdd.map(lambda row: np.append(unpackArray(row['vector']
  year_data_2[m] = np.vstack(rows)

int year_data_2.keys()
```

```
['TMIN', 'TOBS', 'TMAX', 'SNOW', 'SNWD', 'PRCP']
```

In [48]:
```
tmin_df = pd.DataFrame.from_dict(year_data_2['TMIN'])
tmin_df.head()
```

Out[48]:

	0	1	2	3	4	5	6	7	8	9	...	356	357	358	359
0	61.0	117.0	11.0	-28.0	-50.0	-39.0	28.0	83.0	22.0	-22.0	...	39.0	6.0	-6.0	-22.0
1	133.0	172.0	167.0	89.0	28.0	28.0	78.0	72.0	150.0	106.0	...	-22.0	6.0	33.0	33.0
2	-22.0	-33.0	-28.0	-6.0	-22.0	28.0	11.0	50.0	17.0	-28.0	...	122.0	56.0	33.0	11.0
3	22.0	22.0	6.0	-22.0	-22.0	67.0	28.0	0.0	0.0	72.0	...	122.0	133.0	44.0	22.0
4	122.0	83.0	22.0	6.0	11.0	50.0	6.0	56.0	67.0	72.0	...	NaN	NaN	NaN	NaN

5 rows × 366 columns

In [49]:
```
tmin_df.iloc[:,365] = tmin_df.iloc[:,365].astype(str).apply(lambda x: x.spli
# tmin_df.iloc[:,365] = pd.to_datetime(tmin_df.iloc[:,365], format='%Y')
tmin_df.head()
```

Out[49]:

	0	1	2	3	4	5	6	7	8	9	...	356	357	358	359
0	61.0	117.0	11.0	-28.0	-50.0	-39.0	28.0	83.0	22.0	-22.0	...	39.0	6.0	-6.0	-22.0
1	133.0	172.0	167.0	89.0	28.0	28.0	78.0	72.0	150.0	106.0	...	-22.0	6.0	33.0	33.0
2	-22.0	-33.0	-28.0	-6.0	-22.0	28.0	11.0	50.0	17.0	-28.0	...	122.0	56.0	33.0	11.0
3	22.0	22.0	6.0	-22.0	-22.0	67.0	28.0	0.0	0.0	72.0	...	122.0	133.0	44.0	22.0
4	122.0	83.0	22.0	6.0	11.0	50.0	6.0	56.0	67.0	72.0	...	NaN	NaN	NaN	NaN

5 rows × 366 columns

In [51]:
```
tmin_df.rename(columns = {365:'year'}, inplace=True)
tmin_df.head()
```

In [52]:
```
tmin_df.head()
```

Out[52]:

	0	1	2	3	4	5	6	7	8	9	...	356	357	358	359
0	61.0	117.0	11.0	-28.0	-50.0	-39.0	28.0	83.0	22.0	-22.0	...	39.0	6.0	-6.0	-22.0
1	133.0	172.0	167.0	89.0	28.0	28.0	78.0	72.0	150.0	106.0	...	-22.0	6.0	33.0	33.0
2	-22.0	-33.0	-28.0	-6.0	-22.0	28.0	11.0	50.0	17.0	-28.0	...	122.0	56.0	33.0	11.0
3	22.0	22.0	6.0	-22.0	-22.0	67.0	28.0	0.0	0.0	72.0	...	122.0	133.0	44.0	22.0
4	122.0	83.0	22.0	6.0	11.0	50.0	6.0	56.0	67.0	72.0	...	NaN	NaN	NaN	NaN

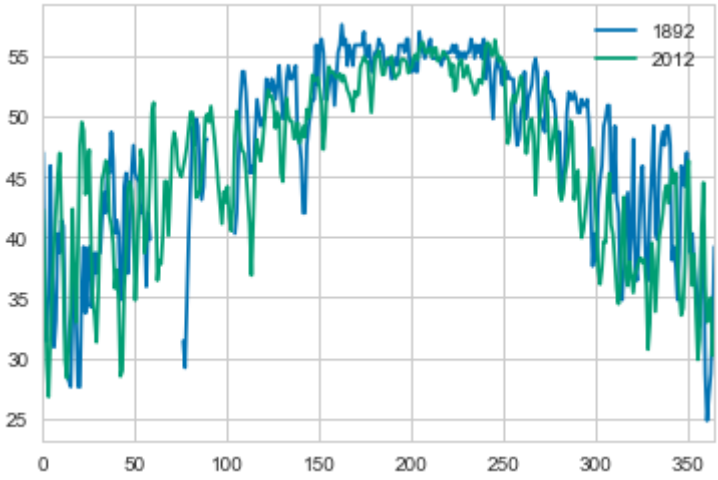
5 rows × 366 columns

In [94]:
```
new = tmin_df.groupby(['year']).mean().applymap(lambda x: convert_C_to_F(x))
```

In [100]:
```
old = tmin_df.groupby(['year']).mean().applymap(lambda x: convert_C_to_F(x))
```

In [115]:
```
t_df = pd.concat([old,new], axis=1)#.plot()
t_df.plot()
```

Out[115]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x125e1a690>
```



In [113]:
```
from __future__ import division
from scipy import stats

ttest = stats.ttest_ind(old.dropna(), new.dropna())

# ttest=stats.ttest_ind(old,new)
print 't-test independent', ttest
```

```
t-test independent Ttest_indResult(statistic=2.4501102478116636, pvalue=
0.014526108030847289)
```

In [114]:
```
print old.mean(), new.mean()
```

```
47.519760479 46.1523561644
```

In [ ]:
```
try cumsum for years to show upwards trend?, also
look at stack overflow on right for folding of rows...
```

In [ ]:

In [ ]:

In [ ]:

In [126]:
```
tmin_df_mean = tmin_df.groupby(['year']).median().applymap(lambda x: convert
```

In [131]:
```
tmin_df_mean.shape
```

Out[131]:
```
(123, 365)
```

In [129]:
```
tmin_df_mean.head(10)
```

Out[129]:

year	0	1	2	3	4	5	6	7	8	9	...	355	356	357	
1890	43.10	49.20	46.40	46.40	46.40	47.00	50.90	49.20	40.30	45.30	...	41.40	42.00	42.6	4
1891	46.40	35.30	30.30	30.90	33.70	35.90	35.90	38.70	40.30	38.70	...	45.30	44.80	44.8	4
1892	47.00	33.70	31.40	35.30	45.90	37.60	30.90	33.10	40.30	38.70	...	34.20	32.60	39.2	4
1893	36.70	35.35	36.70	37.00	36.75	31.70	33.40	30.65	30.90	32.85	...	37.60	45.90	44.8	4
1894	31.70	32.80	40.05	45.90	50.30	50.05	45.60	45.90	45.35	44.50	...	41.15	39.75	42.0	4
1895	31.45	36.70	34.20	34.50	40.90	45.85	47.25	35.60	31.40	31.15	...	40.60	43.40	40.9	4
1896	29.20	30.90	39.20	26.75	26.20	30.05	37.30	41.45	37.00	35.05	...	31.45	36.15	34.2	2
1897	44.80	46.40	40.90	34.80	30.90	28.10	27.60	28.10	30.90	32.60	...	39.20	34.20	37.0	3
1898	30.05	24.50	27.00	33.95	37.80	37.30	39.20	42.00	46.45	47.55	...	46.45	38.65	35.9	3
1899	29.20	30.30	37.00	42.00	43.70	40.90	29.80	29.80	39.20	43.10	...	40.90	39.20	35.9	3

10 rows × 365 columns

In [192]:
```
tmin_epoch = []
start,end = 0, 16
while end < tmin_df_mean.shape[0]:
    tmin_epoch.append(tmin_df_mean.ix[start:end].median())
    start += 15
    end += 15
```

In [193]:
```
tmin_epoch_df = pd.DataFrame(tmin_epoch)
```

In [194]:
```
tmin_epoch_df
```

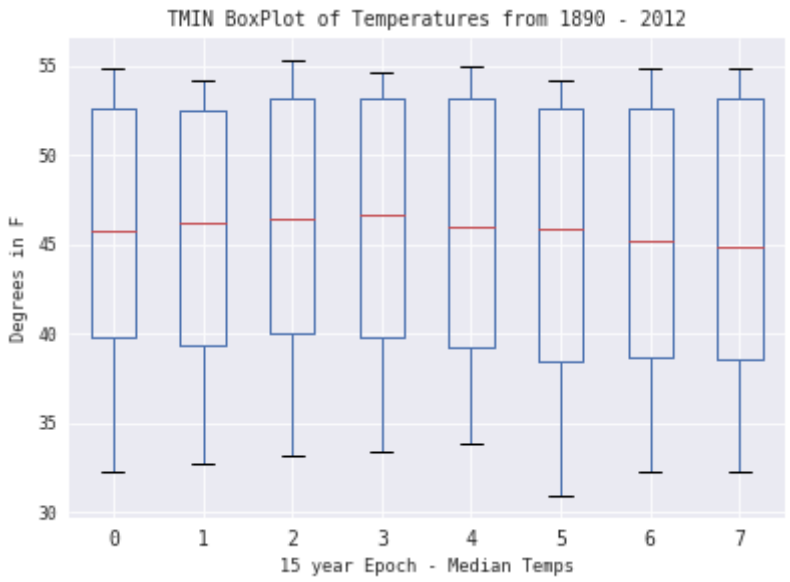
Out[194]:

	0	1	2	3	4	5	6	7	8	9	...	355	356
0	32.300	35.325	36.000	34.65	34.775	36.450	34.350	34.350	37.700	38.700	...	38.150	38.925
1	36.425	40.200	35.875	34.65	36.850	38.675	33.375	33.400	33.825	36.175	...	36.750	36.150
2	37.025	33.125	38.375	39.50	40.900	40.200	40.175	41.275	40.725	35.600	...	37.550	35.725
3	40.600	37.975	38.700	37.55	34.775	37.550	37.975	37.300	38.800	37.150	...	36.750	38.650
4	35.900	37.000	37.850	36.75	38.700	38.950	36.700	34.650	35.050	35.600	...	35.600	37.600
5	39.250	37.000	38.250	35.35	33.825	34.500	35.600	34.500	36.400	35.600	...	35.475	34.925
6	38.700	37.125	37.600	35.90	32.700	32.300	35.200	35.050	35.850	37.000	...	39.200	38.950
7	37.125	37.300	35.200	35.05	32.400	34.500	33.100	34.800	35.900	33.700	...	34.375	36.700

8 rows × 365 columns

In [195]:
```
ax = tmin_epoch_df.T.boxplot()
ax.set_xlabel("15 year Epoch - Median Temps")
ax.set_ylabel("Degrees in F")
ax.set_title("TMIN BoxPlot of Temperatures from 1890 - 2012")
# ax = tmin_epoch_df.T.plot(type=boxplot,title='box')
```

Out[195]: `<matplotlib.text.Text at 0x13cd80ed0>`



In [196]:

```
tmax_df = pd.DataFrame.from_dict(year_data_2['TMAX'])
tmax_df.iloc[:,365] = tmax_df.iloc[:,365].astype(str).apply(lambda x: x.spli
tmax_df.rename(columns = {365:'year'}, inplace=True)
tmax_df_mean = tmax_df.groupby(['year']).median().applymap(lambda x: convert

tmax_epoch = []
start,end = 0, 16
while end < tmax_df_mean.shape[0]:
    tmax_epoch.append(tmax_df_mean.ix[start:end].median())
    start += 15
    end += 15


tmax_epoch_df = pd.DataFrame(tmax_epoch)
tmax_epoch_df
```

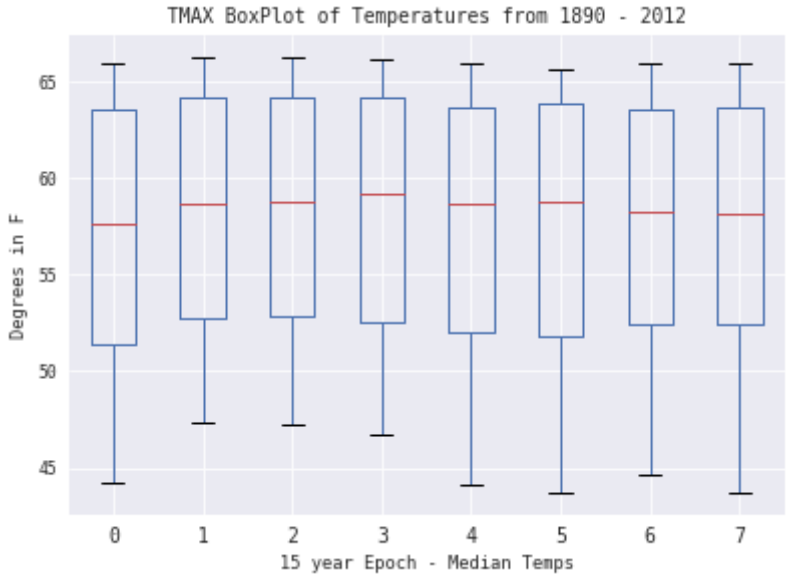
Out[196]:

	0	1	2	3	4	5	6	7	8	9	...	355	356
0	46.875	48.225	47.600	47.850	48.40	48.675	50.200	49.500	48.150	50.050	...	48.550	50.30
1	51.400	49.500	50.625	50.350	50.35	49.200	47.950	49.625	48.275	48.950	...	50.175	47.60
2	48.700	48.150	49.750	49.925	50.35	51.000	51.150	51.450	50.475	50.075	...	49.225	51.00
3	52.850	51.150	50.475	50.325	49.20	51.150	49.075	50.050	46.750	48.400	...	50.900	51.45
4	47.600	50.450	51.025	48.525	50.05	49.250	48.700	48.100	49.375	50.175	...	50.175	50.85
5	49.250	47.125	49.500	48.550	46.70	48.700	49.800	49.350	46.150	45.600	...	47.850	47.85
6	49.200	47.850	46.400	45.200	45.45	48.550	49.800	49.500	46.450	47.850	...	49.350	51.70
7	50.050	50.300	48.950	49.625	49.25	50.050	48.125	48.825	49.750	47.425	...	50.450	50.35

8 rows × 365 columns

In [197]:
```
ax = tmax_epoch_df.T.boxplot()
ax.set_xlabel("15 year Epoch - Median Temps")
ax.set_ylabel("Degrees in F")
ax.set_title("TMAX BoxPlot of Temperatures from 1890 - 2012")
```

Out[197]: <matplotlib.text.Text at 0x13e0cb510>



In [210]:
```
tobs_df = pd.DataFrame.from_dict(year_data_2['TOBS'])
tobs_df.iloc[:,365] = tobs_df.iloc[:,365].astype(str).apply(lambda x: x.spli
tobs_df.rename(columns = {365:'year'}, inplace=True)
tobs_df_mean = tobs_df.groupby(['year']).median().applymap(lambda x: convert

tobs_epoch = []
start,end = 0, 16
while end < tobs_df_mean.shape[0]+1:
    tobs_epoch.append(tobs_df_mean.ix[start:end].median())
    start += 15
    end += 15


tobs_epoch_df = pd.DataFrame(tobs_epoch)
tobs_epoch_df
```

Out[210]:

	0	1	2	3	4	5	6	7	8	9	...	355	356
0	45.850	45.575	41.750	41.050	43.950	42.850	43.675	41.450	43.950	46.275	...	44.200	42.000
1	42.250	44.500	44.925	46.700	47.575	44.475	44.800	45.350	46.200	45.625	...	42.150	42.400
2	43.825	43.950	44.650	44.800	44.350	46.725	47.000	42.700	40.900	42.425	...	46.275	47.575
3	42.575	45.475	45.050	44.500	45.050	44.775	41.700	44.500	45.450	43.950	...	44.625	45.025
4	39.775	42.150	43.400	41.975	39.750	43.100	41.450	40.900	42.825	39.075	...	43.100	42.150
5	41.450	37.150	38.925	37.850	35.325	39.075	42.000	38.275	36.275	39.200	...	41.575	42.850
6	40.600	41.700	42.150	36.700	38.950	39.200	37.425	38.825	39.750	39.500	...	39.750	41.275

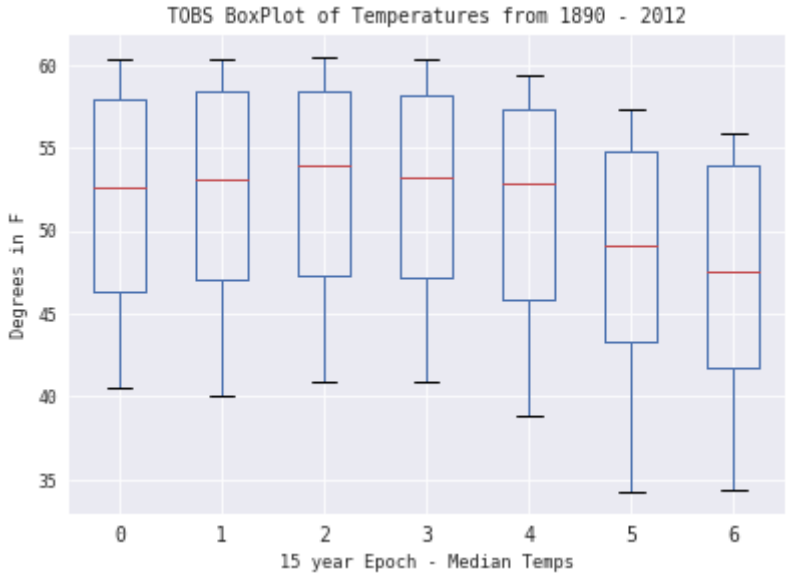
7 rows × 365 columns

In [214]:
```
tmax_df.shape
```

Out[214]: (2020, 366)

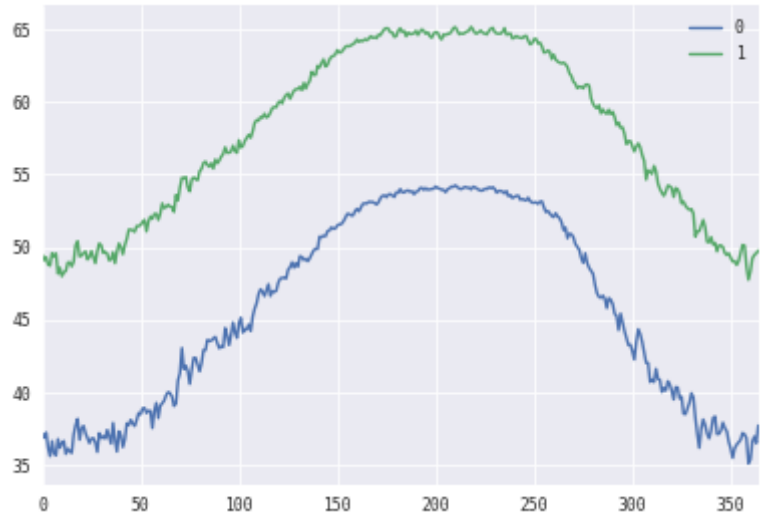
In [207]:
```
ax = tobs_epoch_df.T.boxplot()
ax.set_xlabel("15 year Epoch - Median Temps")
ax.set_ylabel("Degrees in F")
ax.set_title("TOBS BoxPlot of Temperatures from 1890 - 2012")
```

Out[207]: <matplotlib.text.Text at 0x13f57fc50>



In [215]:
```
epoch_std = pd.concat([tmin_epoch_df.std(),tobs_epoch_df.std()],axis=1)#(tma
# epochs = [tmin_epoch_df.std(),tmax_epoch_df.std()]
epoch_std = pd.DataFrame(epochs)
epoch_std.T.plot()
```

Out[215]: <matplotlib.axes._subplots.AxesSubplot at 0x13f759410>

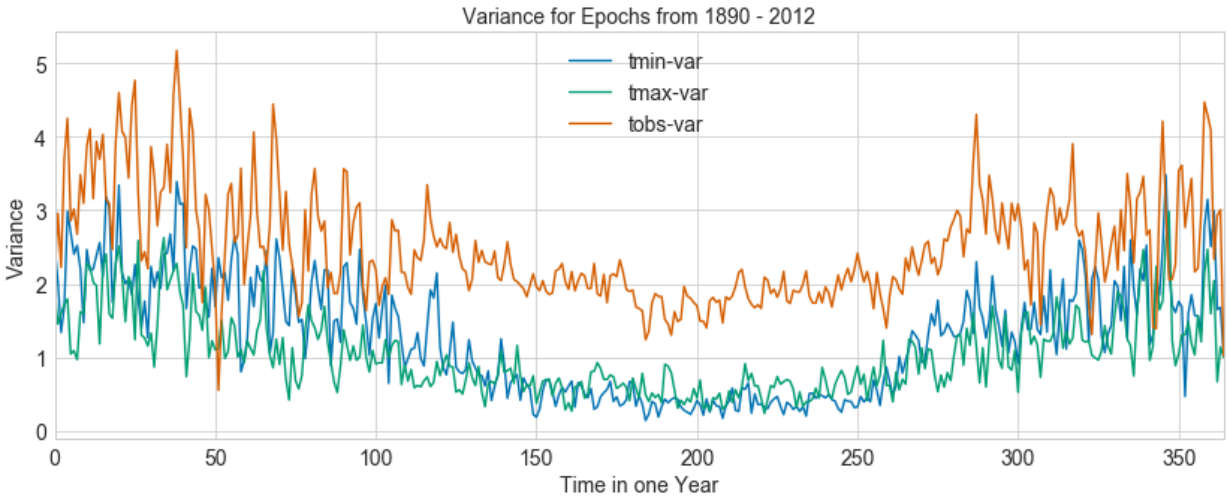


In [237]:
```
epoch_std.head()
```

Out[237]:

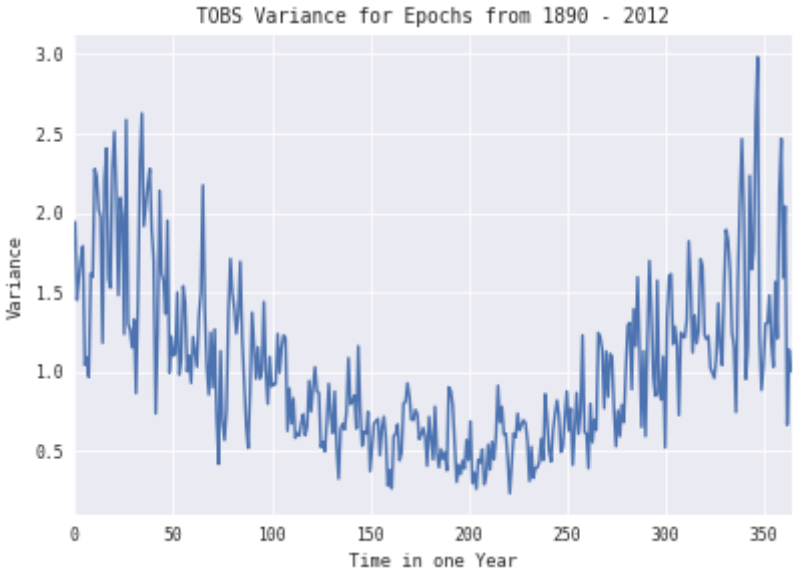
	tmin-var	tmax-var	tobs-var
0	2.516798	1.943025	2.040716
1	2.035565	1.451892	2.958266
2	1.336490	1.593566	2.224037
3	1.688194	1.718583	3.705173
4	2.986216	1.791635	4.252447

In [251]:
```
epochs = [tmin_epoch_df.std(),tmax_epoch_df.std(),tobs_epoch_df.std()]
epoch_std = pd.DataFrame(epochs)
epoch_std = epoch_std.T
epoch_std.columns = ['tmin-var','tmax-var','tobs-var']

# fig = plt.figure(figsize=(14,5))
# ax = fig.add_subplot(111)
ax = epoch_std.plot(figsize=(14,5), fontsize=14)
# ax.plot(epoch_std)
ax.set_xlabel("Time in one Year", fontsize=14)
ax.set_ylabel("Variance", fontsize=14)
ax.set_title("Variance for Epochs from 1890 - 2012", fontsize=14)
plt.legend(loc='best', fontsize=14)
plt.show()
```



In [205]:
```
ax = tobs_epoch_df.std().plot()
ax.set_xlabel("Time in one Year")
ax.set_ylabel("Variance")
ax.set_title("TOBS Variance for Epochs from 1890 - 2012")
```

Out[205]: <matplotlib.text.Text at 0x13f435590>

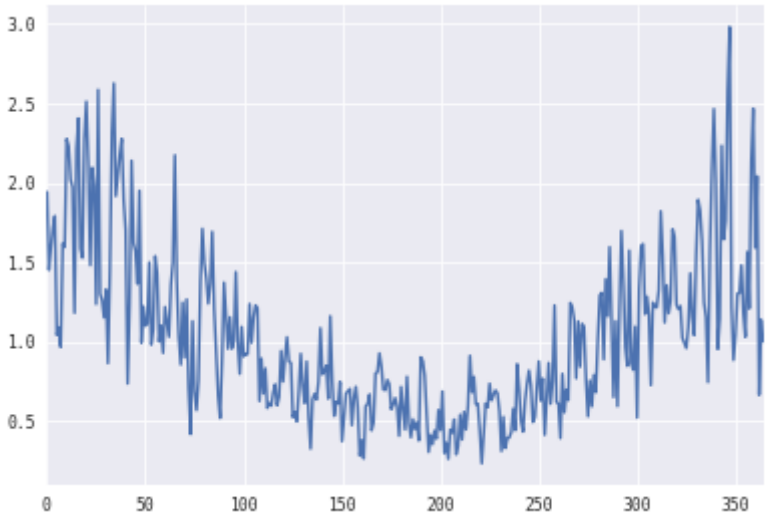


In [168]:
```
tmin_epoch_df.std().plot()
```

Out[168]: <matplotlib.axes._subplots.AxesSubplot at 0x12884d690>

In [169]:
```
tmax_epoch_df.std().plot()
```

Out[169]: <matplotlib.axes._subplots.AxesSubplot at 0x13773ded0>



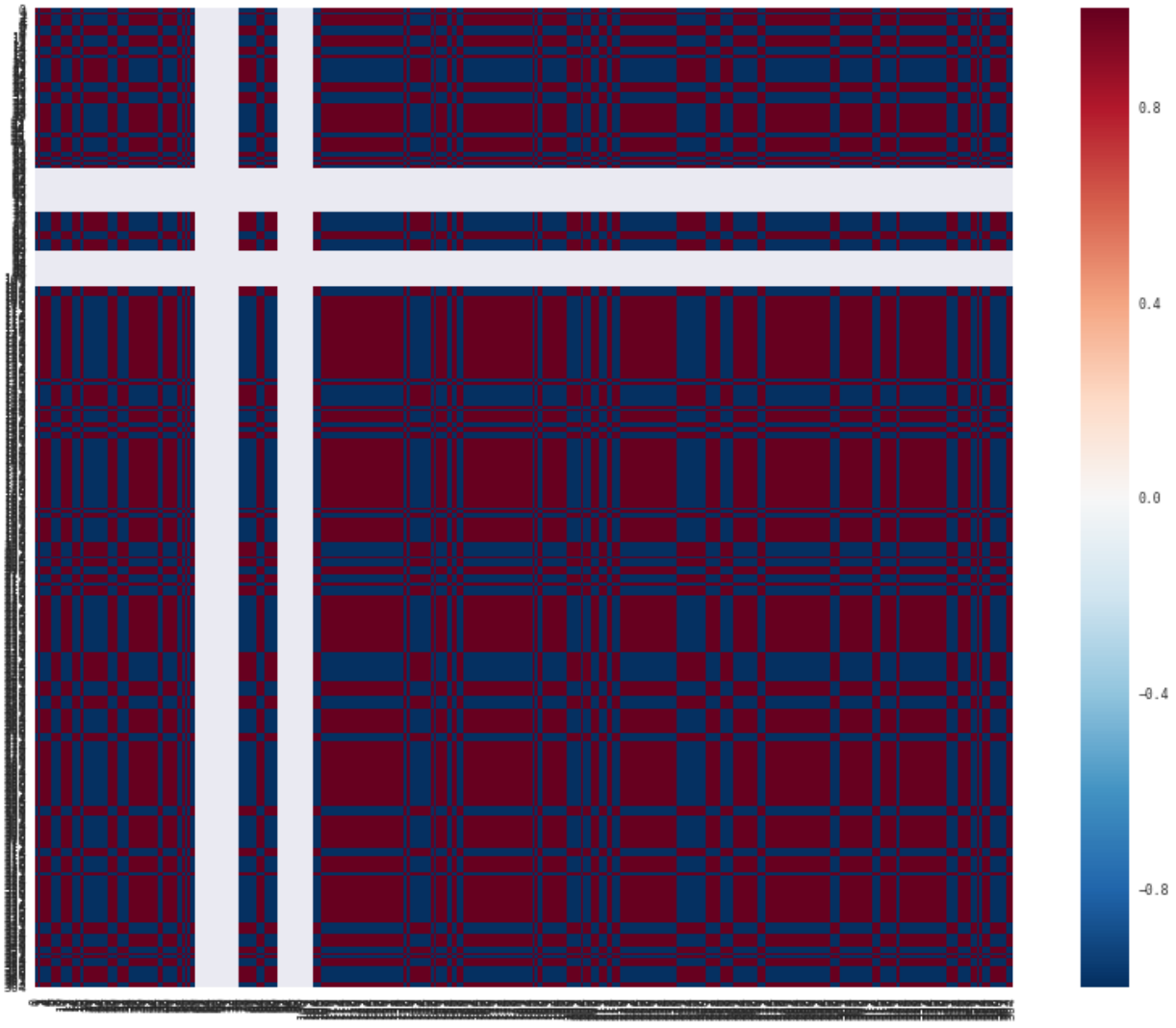
In [ ]:

In [ ]:

In [ ]:

In [117]:

```
sns.set(context="paper", font="monospace")

# Load the datset of correlations between cortical brain networks
# df t_df #= sns.load_dataset("brain_networks", header=[0, 1, 2], index_col=
corrmat = t_df.T.corr()

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(12, 9))

# Draw the heatmap using seaborn
sns.heatmap(corrmat, vmax=.8, square=True)

# Use matplotlib directly to emphasize known networks
# networks = corrmat.columns.get_level_values("network")
# for i, network in enumerate(networks):
#     if i and network != networks[i - 1]:
#         ax.axhline(len(networks) - i, c="w")
#         ax.axvline(i, c="w")
f.tight_layout()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [70]:
```
tmin_df.groupby(['year']).first()
```

Out[70]:

year	0	1	2	3	4	5	6	7	8	9	...	355	356	357
1890	111.0	172.0	144.0	144.0	144.0	150.0	189.0	172.0	83.0	133.0	...	94.0	100.0	106.0
1891	144.0	33.0	-17.0	-11.0	17.0	39.0	39.0	67.0	83.0	67.0	...	133.0	128.0	128.0
1892	150.0	17.0	-6.0	33.0	139.0	56.0	-11.0	11.0	83.0	67.0	...	22.0	6.0	72.0
1893	72.0	17.0	72.0	50.0	89.0	0.0	6.0	17.0	-22.0	11.0	...	56.0	139.0	128.0
1894	-6.0	-17.0	61.0	100.0	183.0	194.0	144.0	111.0	111.0	172.0	...	94.0	94.0	100.0
1895	-28.0	50.0	44.0	17.0	50.0	144.0	161.0	61.0	-6.0	-17.0	...	72.0	100.0	106.0
1896	0.0	22.0	72.0	-44.0	-44.0	17.0	67.0	106.0	83.0	44.0	...	17.0	61.0	61.0
1897	161.0	178.0	122.0	22.0	0.0	-11.0	-17.0	22.0	33.0	56.0	...	72.0	33.0	50.0
1898	-11.0	-67.0	-33.0	61.0	111.0	122.0	100.0	150.0	161.0	178.0	...	156.0	72.0	67.0
1899	-6.0	0.0	44.0	106.0	122.0	122.0	-11.0	-22.0	83.0	111.0	...	94.0	117.0	72.0

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [105]:
```
STAT['PRCP'].keys()
```

Out[105]:
```
['std',
 'UnDef',
 'E',
 'Cov',
 'high1000',
 'NE',
 'O',
 'low100',
 'NO',
 'high100',
 'eigvec',
 'low1000',
 'Var',
 'eigval',
 'mean',
 'SortedVals',
 'Mean']
```

In [106]:
```
STAT_Descriptions
```

Out[106]:
```
[('SortedVals',
  'Sample of values',
  'vector whose length varies between measurements'),
 ('UnDef',
  'sample of number of undefs per row',
  '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%', ()),
 ('E', 'Sum of values per day', (365,)),
 ('NE', 'count of values per day', (365,)),
 ('Mean', 'E/NE', (365,)),
 ('O', 'Sum of outer products', (365, 365)),
 ('NO', 'counts for outer products', (365, 365)),
 ('Cov', 'O/NO', (365, 365)),
 ('Var', 'The variance per day = diagonal of Cov', (365,)),
 ('eigval', 'PCA eigen-values', (365,)),
 ('eigvec', 'PCA eigen-vectors', (365, 365))]
```

In [ ]:
```
# root mean square analysis, substracting the mean from ach point , e.g. ra
#no 1 / 0 if mean was 0.9 subtracting mean would be 0.1 / -0.9 thus you get
#diff
```

In [107]:
```
def plot_mean_std(m,fig,axis):
    mean=STAT[m]['Mean']
    std=np.sqrt(STAT[m]['Var'])
    graphs=np.vstack([mean-std,mean,mean+std]).transpose()
    YP.plot(graphs,fig,axis,title='Mean+-std   '+m)
```

In [ ]:
```
# group by station split into sep DFs
# get df with lat long station number and count to use for a map
# univariate analysis fr each feature
#elevation analysis
# outliers
# years, stations by year analysis
# undefinned analysis for years
#count for each measurement
# start drilling down into each of the seperate measurements.
#look at temp (assuming that there is global warming climate chnage, can I s
#     temp is unifomly increasing, according to the website it says yes, so
#     take tmax tmin, plot curves of the temp by the year, break data set in
#     1920,1940 etc plot the mean temp throughout the year, do these curves
#     gets warmer in the summer, if take 1886 - 1996 as my baseline curve, i
#     I should get the 0 I should get +/- if there is a trend, get the rsidu
```

In [ ]:

In [ ]:

In [ ]:
```
#extract longitude and latitude for each station
feature='coeff_1'
sqlContext.registerDataFrameAsTable(df,'weather')
Query="SELECT station, latitude,longitude,elevation,%s FROM weather"%feature
print(Query)
df1 = sqlContext.sql(Query)
df1.show(4)
```

In [ ]:

In [ ]:
```
df2=df1.groupby(['station','latitude','longitude','elevation']).agg({"static
pdf=df2.toPandas()
pdf.sort_values(by=['station'],inplace=True)
pdf.head(2)
```

In [ ]:

In [ ]:
```
#define a mapping from the range of the value to hex colors.
from matplotlib.colors import rgb2hex
_avg='avg(%s)'%feature
_min=pdf[_avg].min()
_max=pdf[_avg].max()
_min,_max

import pylab as plt
cmap=plt.get_cmap('jet')
def get_color(val):
    x=(val-_min)/(_max-_min)
    return(rgb2hex(cmap(x)[:3]))

get_color(1000.)
```

In [109]:
```
def is_pos_def(x):
    return np.all(np.linalg.eigvals(x) > 0)

for measure in STAT.keys():
    print '{}\'s cov is positive semi def: {}'.format(measure, is_pos_def(ST
```

```
TMIN's cov is positive semi def: False
TOBS's cov is positive semi def: False
TMAX's cov is positive semi def: False
SNOW's cov is positive semi def: False
SNWD's cov is positive semi def: False
PRCP's cov is positive semi def: False
```

In [255]:
```
for stat in STAT.keys():
    print stat, STAT[stat]['eigval'].max()
```

```
TMIN 100485.309663
TOBS 376528.613346
TMAX 36345.0352096
SNOW 312.258081267
SNWD 1416.53530402
PRCP 85913.3689075
```

In [256]:
```
for stat in STAT.keys():
    print stat, STAT[stat]['eigval'].min()
```

```
TMIN -652.90998029
TOBS -1191.96054368
TMAX -983.174275106
SNOW -0.269567080872
SNWD -0.74636558989
PRCP 1116.03735189
```

In [258]:
```
STAT['PRCP']['eigval']
```

Out[258]:
```
array([ 85913.3689075 ,  67192.36274194,  55580.87137412,  50224.96555409,
        43039.42529795,  37844.42529572,  37120.31217476,  37015.44902827,
        36021.90897655,  35483.71871433,  34111.97657463,  33577.36187282,
        32351.77080235,  31886.36818315,  30752.92505023,  29866.09943278,
        29485.53714561,  28518.77069333,  27347.43940654,  26838.18011544,
        26440.72632203,  26025.57532505,  25423.933581  ,  25292.52389595,
        24793.37196138,  24272.14958042,  23994.36681891,  23675.04448826,
        22964.68523373,  22416.28263313,  21988.01919787,  21936.11529697,
        21511.1459879 ,  21113.57133171,  20485.89090914,  20391.96892536,
        20249.83681133,  20095.93658391,  19774.06084388,  19552.34638974,
        19180.92318338,  18937.68827361,  18682.29839571,  18476.74005777,
        18150.6096536 ,  18016.86232027,  17890.32694521,  17348.77768165,
        17251.86873049,  17038.95831777,  16967.32561368,  16738.96156284,
        16573.6041552 ,  16197.2977163 ,  16000.11504499,  15910.37831466,
        15821.88497407,  15455.07066009,  15269.87294643,  15192.84337385,
        15049.91545648,  14723.43763215,  14588.63665154,  14320.17160477,
        14156.93582587,  14065.14907809,  14004.13221452,  13903.78778502,
        13749.87057291,  13649.45592491,  13491.03159207,  13373.86051124,
        13326.48685031,  13180.48886539,  13050.25830129,  12998.23755194,
        12801.576115  ,  12608.86774958,  12578.42789434,  12447.77589742,
        12358.06360214,  12233.67726126,  12166.91066254,  12103.27501104,
        12051.1242726 ,  11955.30761067,  11882.63012468,  11720.73392883,
        11637.56345344,  11590.01821906,  11530.73043645,  11415.58751269,
        10816.49016187,  11313.88442244,  11217.18996026,  11177.11170999,
        10948.59975424,  11044.04734267,  11025.36360314,  10677.95954101,
        10553.28403401,  10507.05152221,  10424.49780796,  10315.78355288,
        10272.00647157,  10182.53013521,  10133.49694952,  10093.24657951,
        10059.38516137,  10013.13621587,   9954.35171932,   9852.06517266
```

7,
9746.4266147 , 9703.94829619, 9630.01220082, 9564.2927398
5,
9416.15296516, 9380.25383754, 9367.48600878, 1116.0373518
9,
9248.90778356, 9206.77121938, 9151.23906016, 9071.3675441
9,
9044.12701548, 8932.95957042, 8900.43273203, 8868.4096821
3,
8770.98635391, 8730.87781235, 8668.34277218, 8295.9202612
,
8614.4863026 , 8375.35520338, 8557.80087535, 8522.1338121
4,
8475.69233988, 8456.44004967, 1421.42684621, 1430.8216353
,
1477.61487751, 1543.32584281, 1557.2689826 , 1592.9543177
,
1622.4485773 , 1640.51222521, 1666.0221068 , 8286.2725392
5,
8199.43306237, 8168.10628798, 8114.28707587, 8071.2491636
7,
8047.1835427 , 7984.34171001, 7901.16096844, 7934.5119760
8,
1719.98615986, 1732.05057036, 1758.55609547, 1803.7492536
8,
1795.6055933 , 1838.38945539, 7877.50319927, 7798.1462182
9,
7763.49013768, 7545.32834001, 7709.88036826, 7660.9298832
9,
7635.70330627, 1881.77663764, 1917.81063996, 1899.2702892
7,
1890.15136272, 1948.43424688, 1988.59316064, 7558.3870140
8,
7487.42096413, 7412.59806026, 7370.38164162, 7294.3404545
9,
7096.5822477 , 7154.1436519 , 7170.4803292 , 7243.7173711
,
7282.77691415, 2004.9419056 , 2026.64261713, 2051.4852945
,
2069.09961489, 2120.98523373, 2210.43677015, 2191.4131632
1,
2139.09033405, 2170.41641041, 7039.99079241, 6970.8840840
3,
6942.19327936, 6919.2161451 , 6857.58048545, 6818.4680821
5,
6748.20965482, 6534.3259871 , 6550.19138576, 6584.8120752
2,
6627.68685882, 6678.09589333, 6674.94097714, 2269.8782676
7,
2249.83460205, 2256.84543056, 6473.10223662, 6388.5678414
5,
6434.87208907, 6364.03330766, 6339.51208506, 6297.4208118
6,
2302.2366679 , 2345.73847269, 2329.81116107, 2332.0655445
8,
6264.46608269, 6240.19086199, 6183.24826443, 6163.2996415
6,

6146.92163538, 2361.26539228, 2397.06322003, 2426.27111013,
2468.97104054, 2461.13505756, 2439.25658163, 6084.99092331,
5970.19658791, 6059.64072667, 6003.9218226 , 6050.70081361,
5931.21702898, 5881.78753637, 5848.82021156, 2449.22737232,
2497.78201805, 2504.93360414, 2536.89505166, 2562.38003786,
2583.76145596, 2605.86207566, 2638.78922573, 2654.11232377,
5777.02153729, 5759.6011399 , 5693.93956345, 5663.13713996,
5614.21855885, 5573.12033482, 5546.70535083, 5531.84038113,
5511.19571882, 5496.70867337, 5484.96749307, 2553.26281472,
2688.20296545, 2700.24906778, 2719.49729586, 2732.39520112,
2772.93675901, 2873.86568449, 2889.26051369, 2794.56158601,
2847.78023163, 2832.60281454, 2811.48439825, 2918.26834859,
5374.74344141, 5316.55159085, 3023.52074446, 2949.44557083,
2979.01949155, 2966.79820536, 5299.62261788, 5279.44500152,
5261.90502298, 5243.09268666, 5165.02226581, 5188.43779525,
5215.4159451 , 3139.12552823, 3109.92093237, 3095.84384008,
3045.79066779, 3041.99781978, 3169.82112014, 3203.43786136,
3233.59861142, 3267.98158253, 3288.09347868, 3310.04406541,
5100.08476012, 5048.94728442, 5114.0911626 , 5022.66440637,
4994.10412747, 4973.63988343, 4929.79656212, 4891.75285499,
4914.80222585, 3341.48734385, 3355.93807836, 3409.32302018,
3178.58389621, 3440.03935969, 4857.65396427, 4836.32941787,
4705.32441383, 4647.93942162, 4635.28160059, 4739.28843929,
4056.52053072, 4079.50163503, 4600.69522269, 4387.17995045,
4328.63320258, 4353.41817513, 4265.51379095, 3388.20439659,
4791.70013028, 4756.56580418, 3444.28602414, 3520.31487463,
3542.63835566, 3660.25831172, 4556.60206069, 4417.47061697,
4480.47513064, 4492.52240363, 4431.54552929, 4504.74742437,
4242.21199956, 4217.47239759, 4130.54569308, 4296.98649076,

```
3,
        3701.47340477,   4013.08178571,   4146.43072686,   3760.3923021
6,
        3733.28359627,   3782.59084482,   3859.34511444,   3896.0454522
8,
        3806.41375035,   3837.9114554 ,   3940.93496909,   3957.4063123
2,
        4722.95363107,   3599.92377476,   3723.76705515,   4529.5905473
9,
        3489.22610583,   3559.76431045,   3584.41672385,   3606.9681032
8,
        4198.27716872,   3824.95470231,   4005.46115205,   3920.3315933
 ,
        3585.75771988])
```

In [105]:
```
STAT_Descriptions
```

Out[105]:
```
[('SortedVals',
  'Sample of values',
  'vector whose length varies between measurements'),
 ('UnDef',
  'sample of number of undefs per row',
  '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%', ()),
 ('E', 'Sum of values per day', (365,)),
 ('NE', 'count of values per day', (365,)),
 ('Mean', 'E/NE', (365,)),
 ('O', 'Sum of outer products', (365, 365)),
 ('NO', 'counts for outer products', (365, 365)),
 ('Cov', 'O/NO', (365, 365)),
 ('Var', 'The variance per day = diagonal of Cov', (365,)),
 ('eigval', 'PCA eigen-values', (365,)),
 ('eigvec', 'PCA eigen-vectors', (365, 365))]
```

In [ ]:
```
```

Helper Functions

In [65]:
```
def mean_std_nparray(array):
    '''input X size df returns data mean and +/- std as a 3 col by X rows df
    mean = np.nanmean(array, axis=0, dtype=np.float64)
    std = np.nanstd(array, axis=0, dtype=np.float64)
    return np.vstack([mean-std, mean, mean+std]).transpose()
```

In [66]:
```
def mean_std(df):
    '''input X size df returns data mean and +/- std as a 3 col by X rows df
    mean = df.mean()
    std = df.std()
    return np.vstack([mean-std, mean, mean+std]).transpose()
```

In [67]:
```
def convert_C_to_F(temp):
    '''converts temperatures in tenths of a C to F'''
    return round((temp/10.0*(9/5)+32),2) #divide temp/10 NOAA data
```

In [ ]:

In [ ]:

In [ ]: