Proyecto de Machine Learning "Climate Change: Earth Surface Temperature Data"

Pasos

1. visualizacion de datos (mapas)
2. Mapas interactivos (con animacion o barras interactivas)
3. Tarea de Prediccion Temporal o Espacial

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
%run -i calhousing.py
```

```
<div class="bk-root">
    <a href="http://bokeh.pydata.org" target="_blank" class="bk-logo bk-logo-small
bk-logo-notebook"></a>
    <span id="f4813684-7609-43d0-8830-aad139380f1c">Loading BokehJS ...</span>
</div>
```

Importo los Datos

NOTA: En las coordenadas N es positivo y S Negativo, E es positivo y W Negativo

```
dglobal = pd.read_csv("GlobalTemperatures.csv")
print dglobal.shape
dglobal.tail()
```

```
(3192, 9)
```

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemper
3187	2015-08-01	14.755	0.072	20.699
3188	2015-09-01	12.999	0.079	18.845
3189	2015-10-01	10.801	0.102	16.450
3190	2015-11-01	7.433	0.119	12.892

3191	2015-12-01	5.518	0.100	10.725

```
dcity = pd.read_csv("GlobalLandTemperaturesByCity.csv")
print dcity.shape
dcity.head()
```

```
(8599212, 7)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	La
0	1743-11-01	6.068	1.737	Århus	Denmark	57
1	1743-12-01	NaN	NaN	Århus	Denmark	57
2	1744-01-01	NaN	NaN	Århus	Denmark	57
3	1744-02-01	NaN	NaN	Århus	Denmark	57
4	1744-03-01	NaN	NaN	Århus	Denmark	57

```
dcountry = pd.read_csv("GlobalLandTemperaturesByCountry.csv")
print dcountry.shape
dcountry.head()
```

```
(577462, 4)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
0	1743-11-01	4.384	2.294	Åland
1	1743-12-01	NaN	NaN	Åland
2	1744-01-01	NaN	NaN	Åland
3	1744-02-01	NaN	NaN	Åland

4	1744-03-01	NaN	NaN	Åland

```
dMcity = pd.read_csv("GlobalLandTemperaturesByMajorCity.csv")
print dMcity.shape
dMcity.head()
```

```
(239177, 7)
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	La
0	1849-01-01	26.704	1.435	Abidjan	Côte D'Ivoire	5.
1	1849-02-01	27.434	1.362	Abidjan	Côte D'Ivoire	5.
2	1849-03-01	28.101	1.612	Abidjan	Côte D'Ivoire	5.
3	1849-04-01	26.140	1.387	Abidjan	Côte D'Ivoire	5.
4	1849-05-01	25.427	1.200	Abidjan	Côte D'Ivoire	5.

Conociendo los Datos

Visualizacion de temperaturas por ciudad

1. Intente Graficar, con la funcion que el profesor Raul Ramos Fabrico en las notas del curso, pero curiosamente no logre hacer que funcionara
2. este mal funcionamiento, puedo pensar que es por la diferencia en los datos, Los datos utilizados por el profesor tenian coordenadas totalmente numericas, y estos datos poseen coordenadas en String

```
smp = dcity.iloc[:500]
plot_map(smp["Latitude"], smp["Longitude"], smp["AverageTemperature"])
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-9-aba110e4e9b8> in <module>()
      1 smp = dcity.iloc[:500]
```

```
----> 2 plot_map(smp["Latitude"], smp["Longitude"], smp["AverageTemperature"])


/home/aronf/Machine Learning/DATASETML/calhousing.py in plot_map(lat, lon, color,
size)
    102 def plot_map(lat, lon, color=None, size=10):
    103     cmap = cm.rainbow
--> 104     wlat, wlong = latlng_to_meters(lat, lon)
    105     if color is not None:
    106         colors = MinMaxScaler(feature_range=(0,255)).fit_transform(color)


/home/aronf/Machine Learning/DATASETML/calhousing.py in latlng_to_meters(lat, lng)
     49
     50 def latlng_to_meters(lat, lng):
---> 51     lat1, lng1 = latlng_str_to_num(lat,lng)
     52     origin_shift = 2 * np.pi * 6378137 / 2.0
     53     mx = lng1 * origin_shift / 180.0


/home/aronf/Machine Learning/DATASETML/calhousing.py in latlng_str_to_num(lat,
lng)
     38     elif ltN > 0:
     39         lat = lat.replace("N", "0")
---> 40         lat = float(lat)
     41     if lnW > 0:
     42         lng = lng.replace("W", "0")


ValueError: invalid literal for float(): 0      57.050
1      57.050
2      57.050
3      57.050
4      57.050
5      57.050
6      57.050
7      57.050
8      57.050
9      57.050
10     57.050
11     57.050
12     57.050
13     57.050
14
```

```
sample = dcity.iloc[6337428]
lat, lng = sample["Latitude"], sample["Longitude"]
print lat.lstrip("3"), lat, lng #PROBANDO EL LSTRIP para quitar cadenas de la
izquierda
```

```
4.56N 34.56N 116.76W
```

Funcion para convertir las coordenadas tipo String a tipo Float

Esta funcion fue pensada explicitamente para quitar la letra de las coordenadas (Latitud y Longitud) y convertir estos datos de String a Float y poder asi utilizar la funcion de plotmap del profesor. Funciona correctamente, pero al retornar

los datos a la funcion de PLOT ocurre un error, al parecer agrega elementos como index.

```
def latlng_str_to_num(lat,lng):
    lat = str(lat)
    lng = str(lng)
    ltS = lat.count("S")
    ltN = lat.count("N")
    lnW = lng.count("W")
    lnE = lng.count("E")
    if ltS > 0:
        lat = lat.replace("S", "0")
        lat = float(lat)
        lat = lat*(-1)
    elif ltN > 0:
        lat = lat.replace("N", "0")
        lat = float(lat)
    if lnW > 0:
        lng = lng.replace("W", "0")
        lng = float(lng)
        lng = lng*(-1)
    elif lnE > 0:
        lng = lng.replace("E", "0")
        lng = float(lng)
    return lat, lng
```

```
i = np.random.randint(0,8599212)
print "Dato numero :", i
sample = dcity.iloc[i]
lat = sample["Latitude"]
print "Latitud Original :", lat
lng = sample["Longitude"]
print "Longitud Original :", lng
x, y = latlng_str_to_num(lat, lng)
print "Latitud y Longitud transformadas por la funcion creada previamente :", x, y
```

```
Dato numero : 3787389
Latitud Original : 32.95N
Longitud Original : 50.74E
Latitud y Longitud transformadas por la funcion creada previamente : 32.95 50.74
```

Nuevo Intento de graficar, con las herramientas de * Basemap * y * animation *

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from mpl_toolkits.basemap import Basemap
from matplotlib import animation, rc
from IPython.display import HTML
```

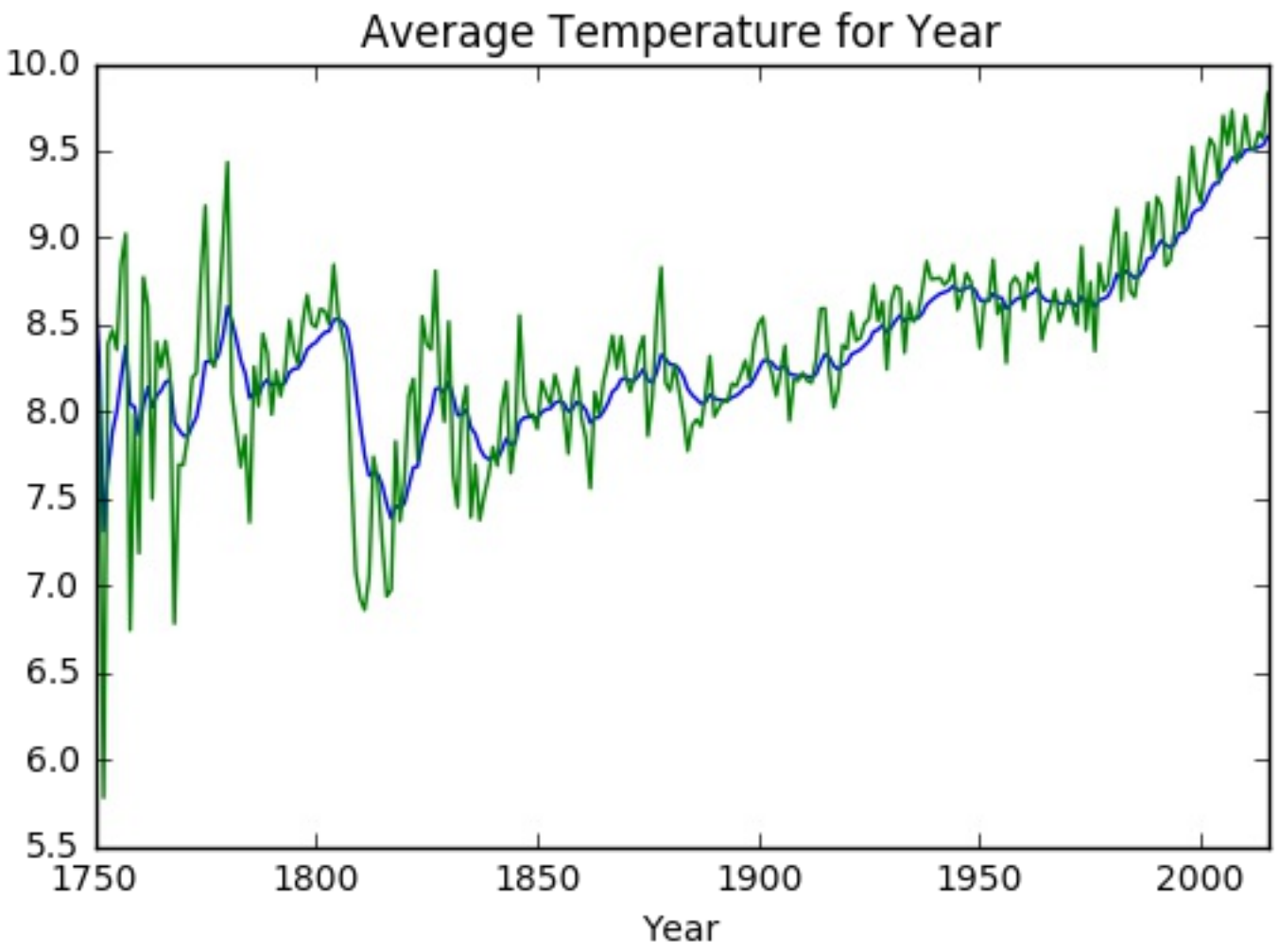
```
dglobal = pd.read_csv("GlobalTemperatures.csv", parse_dates=['dt'])
dglobal.head()
```

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTempera
0	1750-01-01 00:00:00	3.034	3.574	NaN
1	1750-02-01 00:00:00	3.083	3.702	NaN
2	1750-03-01 00:00:00	5.626	3.076	NaN
3	1750-04-01 00:00:00	8.490	2.451	NaN
4	1750-05-01 00:00:00	11.573	2.072	NaN

Primer intento de Grafica con las medias de las temperaturas mundiales.

```
year_temp = dglobal.groupby(dglobal.dt.dt.year).mean()
pd.stats.moments.ewma(year_temp.LandAverageTemperature, 5).plot()
year_temp.LandAverageTemperature.plot(linewidth=1)
plt.title('Average Temperature for Year') #Lo escribo en ingles, porque ocurre error con la letra "ñ"
plt.xlabel('Year')
```

```
<matplotlib.text.Text at 0x7f715eb823d0>
```



Se puede evidenciar en la grafica anterior, un incremento NOTABLE en la temperatura en los ultimos 150-170 años mas o menos.

- Ahora analizaremos temperaturas por ciudad

```
bycities = pd.read_csv('GlobalLandTemperaturesByCity.csv', parse_dates=['dt'])
#Hay algunas ciudades con el mismo nombre pero en paises diferentes
bycities[['City', 'Country']].drop_duplicates()
bycities.City = bycities.City.str.cat(bycities.Country, sep=' ')
bycities = bycities[bycities.dt.dt.year >= 1900]
bycities.head()
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country
1874	1900-01-01	-0.989	0.588	Århus Denmark	Denmark
1875	1900-02-01	-2.799	0.882	Århus Denmark	Denmark
1876	1900-	0.592	0.429	Århus	Denmark

	03-01			Denmark	
1877	1900-04-01	4.630	0.417	Århus Denmark	Denmark
1878	1900-05-01	9.576	0.521	Århus Denmark	Denmark

Utilizaremos el promedio de la temperatura anual, para crear una tabla donde relacionemos la ciudad y el pais. (Se hace con los maximos y minimos para tener mas perspectivas a la hora de ver los datos)

```
city_means = bycities.groupby(['City', bycities.dt.dt.year])
['AverageTemperature'].mean().unstack()
city_mins = bycities.groupby(['City', bycities.dt.dt.year])
['AverageTemperature'].min().unstack()
city_maxs = bycities.groupby(['City', bycities.dt.dt.year])
['AverageTemperature'].max().unstack()
city_means.head()
```

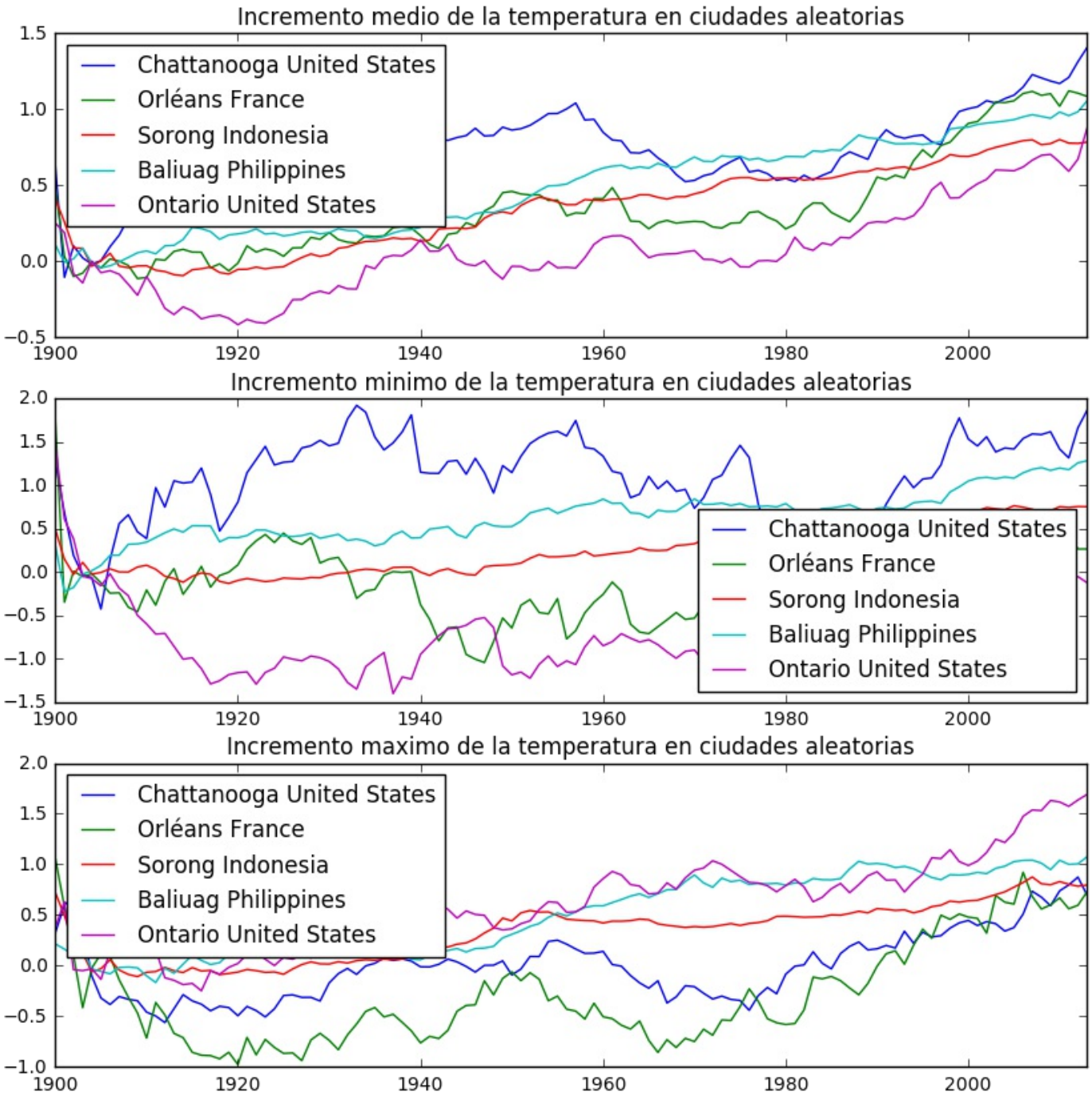
dt	1900	1901	1902	1903	1904	1905
City						
A Coruña Spain	13.267917	12.773417	12.828333	13.028167	13.349083	12.688000
Aachen Germany	9.132500	8.339750	8.133583	8.994000	8.987333	8.571083
Aalborg Denmark	7.375250	7.875667	6.542667	7.830833	7.577083	7.808917
Aba Nigeria	26.418833	26.455333	26.001750	25.875917	25.342083	26.315250
Abadan Iran	25.016167	25.770750	25.459083	24.242750	24.777750	24.393500

5 rows × 114 columns

```
first_years_mean = city_means.iloc[:, :5].mean(axis=1) # Media de la temperaturaen
los primeros 5 años
city_means_shifted = city_means.subtract(first_years_mean, axis=0)

def plot_temps(cities, city_ser, ax):
    first_years_mean = city_ser.iloc[:, :5].mean(axis=1)
    city_ser = city_ser.subtract(first_years_mean, axis=0)
    for city in cities:
```

```
        row = city_ser.loc[city]
        pd.stats.moments.ewma(row, 10).plot(label=row.name, ax=ax)
    ax.set_xlabel('')
    ax.legend(loc='best')

fig, axes = plt.subplots(3,1, figsize=(10,10))

n = 5
random_cities = city_means_shifted.sample(n).index

plot_temps(random_cities, city_means, axes[0])
plot_temps(random_cities, city_mins, axes[1])
plot_temps(random_cities, city_maxs, axes[2])

axes[0].set_title("Incremento medio de la temperatura en ciudades aleatorias")
axes[1].set_title("Incremento minimo de la temperatura en ciudades aleatorias")
axes[2].set_title("Incremento maximo de la temperatura en ciudades aleatorias")
```

```
<matplotlib.text.Text at 0x7f70f30ac0d0>
```



Se nota a simple vistaa que la temperatura promedio crece a traves de los años, PERO, no podemos decir lo mismo si hablamos de los minimos y maximos, estos tienen un comportamiento menos obvio, un poco mas caotico.

Graficar puntos en el mapa, y animarlo a traves de los años

```
cities_info = bycities.groupby(['City']).first()
cities_info.head()
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Country	Latitu

City					
A Coruña Spain	1900-01-01	8.089	0.655	Spain	42.59N
Aachen Germany	1900-01-01	2.079	0.545	Germany	50.63N
Aalborg Denmark	1900-01-01	-0.989	0.588	Denmark	57.05N
Aba Nigeria	1900-01-01	25.722	0.892	Nigeria	5.63N
Abadan Iran	1900-01-01	11.636	1.253	Iran	29.74N

Puntos en el mapa

Los puntos en el mapa tienen dos caracteristicas muy importantes y que es necesario explicarlas para poder entender a que va el siguiente mapa.

- Color
- Tamaño

El color:

Nos da la informacion sobre la temperatura, si ese año la temperatura de la ciudad fue mas calida (rojo) o mas fria (azul).

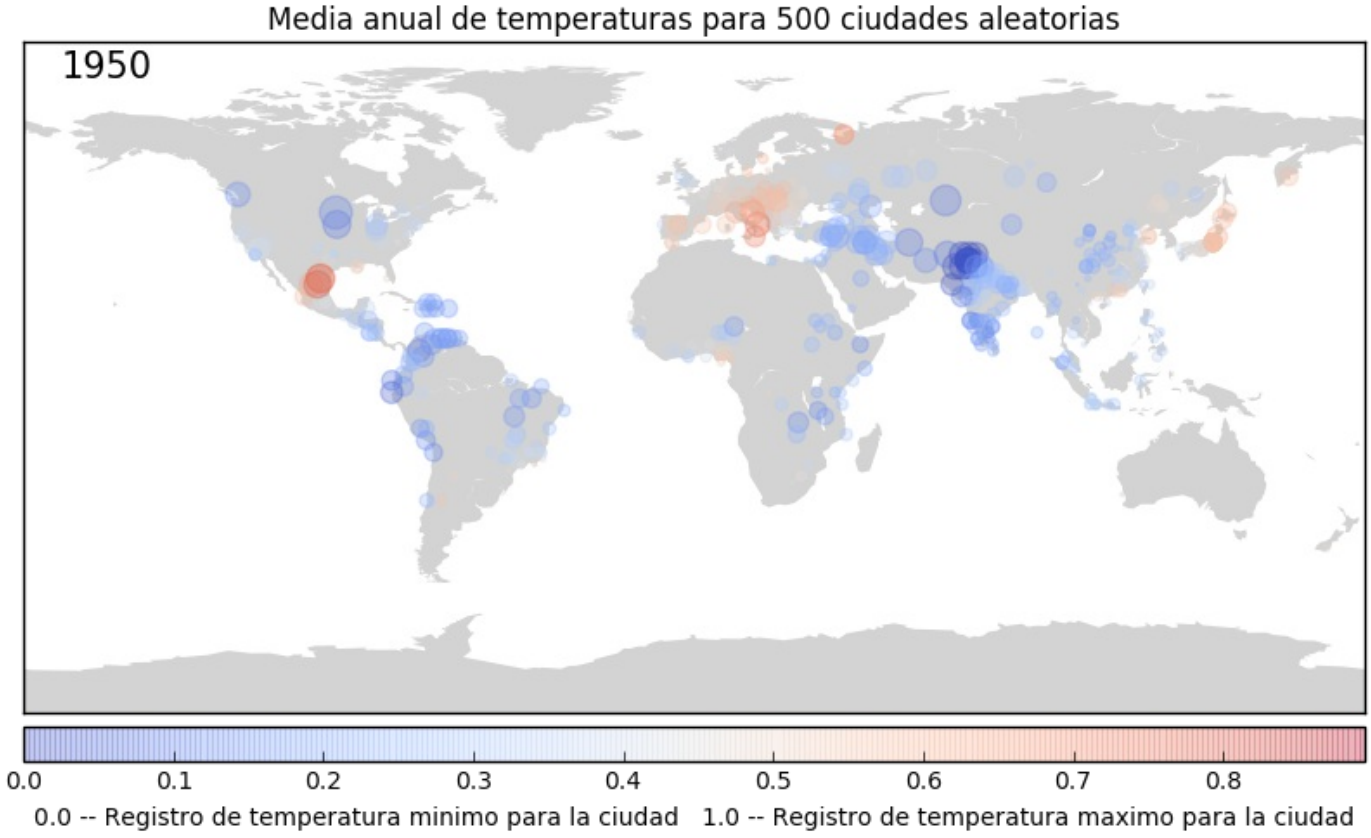
El tamaño:

Representa la diferencia absoluta entre la temperatura media de la ciudad y la temperatura recurrente en ese año.

```
def get_temp_markers(city_names, year):
    points = np.zeros(len(city_names), dtype=[('lon', float, 1),
                                      ('lat', float, 1),
                                      ('size',     float, 1),
                                      ('color',    float, 1)])
    cmap = plt.get_cmap('coolwarm')

    for i, city in enumerate(city_names):
        city_temps = city_means.loc[city]
        _MIN, _MAX, _MEDIAN = city_temps.min(), city_temps.max(), city_temps.median()
        temp = city_temps.loc[year]

        coords = cities_info.loc[city][['Latitude', 'Longitude']].values
        lat = float(coords[0][:-1]) * (-1 if coords[0][-1] == 'S' else 1)
        lon = float(coords[1][:-1]) * (-1 if coords[1][-1] == 'W' else 1)
```

```
        points['lat'][i] = lat
        points['lon'][i] = lon
        points['size'][i] = 100 * abs(temp - _MEDIAN)
        points['color'][i] = (temp - _MIN) / (_MAX - _MIN)

    return points
```

```
fig = plt.figure(figsize=(10, 10))
cmap = plt.get_cmap('coolwarm')

map = Basemap(projection='cyl')
map.drawmapboundary()
map.fillcontinents(color='lightgray', zorder=1)

START_YEAR = 1950
LAST_YEAR = 2013

n_cities = 500
random_cities = city_means.sample(n_cities).index
year_text = plt.text(-170, 80, str(START_YEAR),fontsize=15)

temp_markers = get_temp_markers(random_cities, START_YEAR)
xs, ys = map(temp_markers['lon'], temp_markers['lat'])
scat = map.scatter(xs, ys, s=temp_markers['size'], c=temp_markers['color'],
cmap=cmap, marker='o',
                   alpha=0.3, zorder=10)

def update(frame_number):
    current_year = START_YEAR + (frame_number % (LAST_YEAR - START_YEAR + 1))

    temp_markers = get_temp_markers(random_cities, current_year)
    xs, ys = map(temp_markers['lon'], temp_markers['lat'])

    scat.set_offsets(np.dstack((xs, ys)))
    scat.set_color(cmap(temp_markers['color']))
    scat.set_sizes(temp_markers['size'])

    year_text.set_text(str(current_year))

# # # Se construye la animacion, usando la funcion de actualizacion de animation

ani = animation.FuncAnimation(fig, update, interval=500, frames=LAST_YEAR -
START_YEAR + 1)

cbar = map.colorbar(scat, location='bottom')
cbar.set_label('0.0 -- Registro de temperatura minimo para la ciudad   1.0 --
Registro de temperatura maximo para la ciudad')
plt.title('Media anual de temperaturas para {} ciudades aleatorias'.format(n_cities))
plt.show()
```



Creo un Archivo .GIF con lo anterior y lo guardo como tempanimation.gif

```
ani.save('tempanimation.gif', writer='imagemagick', fps=2)
```

Observacion de

Se puede notar que en los ultimos años la temperatura aumento rapidamente, en los ultimos 10 años (aproximadamente) la temperatura crecio de una manera significativa, y algo importante es que este crecimiento mencionado es mucho mayor en lugares mas poblados, como se puede observar en Europa y sus ciudades, y en lugares menos poblados este crecimiento es mas pequeño como en algunos lugares de Africa y Sur-America.

Tarea de Prediccion

Para la tarea de prediccion, se podria pensar en dos tipos:

- Prediccion en el tiempo
- Prediccion Espacial

Prediccion en el tiempo:

se refiere a construir un modelo predictivo que nos permita predecir la temperatura media en algunos lugaresa medida que pasan los años, con el fin de poder obtener un modelo predictivo que pueda funcionar en cualquier lugar del mundo si se le proporcionan los datos necesarios para el entrenamiento.

Prediccion Espacial:

Con prediccion espacial se refiere a predecir la temperatura (desconocida) de un punto (ciudad) en el mundo, conociendo previamente las temperaturas de algunos de los puntos cercanos a este. Teniendo el mismo fin que en la anterior tarea, encontrar un modelo predictivo general, que al proporcionarle los datos necesarios y suficientes pueda funcionar en cualquier parte del mundo.

* Bibliografia *

- Global Warming Confirmed - https://www.kaggle.com/pavelevap/d/berkeleyearth/climate-change-earth-surface-temperature-data/global-warming-confirmed-basemap-animation/notebook
- Documentacion de Python basemap - http://matplotlib.org/basemap/
- Documentacion de Python Pandas - http://pandas.pydata.org/pandas-docs/stable/
- Documentacion de Python Animation(matplotlib) - https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/
- Documentacion interna con el comando "Help()"