3 -ScriptWeather

July 10, 2017

1 Obtención de datos del tiempo de la web de Noaa

El objetivo de este notebook es obtener y procesar los datos del tiempo atmosférico de cada aeropuerto para cada vuelo. A continuación se irán describiendo los pasos a ejecutar.

El primer paso es importar las librerias necesarias:

2 PASO 1

```
In [ ]: import time
        import pandas as pd
        import numpy as np
        import math
        from math import cos, asin, sqrt
        import gc
        import os.path
        pd.set_option('display.max_columns', None)
```

3 PASO 2

Se lee el fichero generado anteriormente para crear nuevas columnas.

```
In [ ]: vuelos = pd.read_csv('vuelosDeparted.csv', sep=',',low_memory=False, )
        vuelos.rename(columns={'Unnamed: 0': 'index'}, inplace=True)

        vuelos["TMIN_o"]= ''
        vuelos["TMIN_d"]= ''
        vuelos["TMAX_o"]= ''
        vuelos["TMAX_d"]= ''
        vuelos["TAVG_o"]= ''
        vuelos["TAVG_d"]= ''
        vuelos["SNOW_o"]= ''
        vuelos["SNOW_d"]= ''
        vuelos["PRCP_o"]= ''
        vuelos["PRCP_d"]= ''
        vuelos["SNWD_o"]= ''
        vuelos["SNWD_d"]= ''
```

```
vuelos["ACMC_o"]= ''
vuelos["ACMC_d"]= ''
vuelos["ACSC_o"]= ''
vuelos["ACSC_d"]= ''
vuelos["AWDR_o"]= ''
vuelos["AWDR_d"]= ''
vuelos["AWND_o"]= ''
vuelos["AWND_d"]= ''
vuelos["EVAP_o"]= ''
vuelos["EVAP_d"]= ''
vuelos["FRTH_o"]= ''
vuelos["FRTH_d"]= ''
vuelos["TSUN_o"]= ''
vuelos["TSUN_d"]= ''
vuelos["WDMV_o"]= ''
vuelos["WDMV_d"]= ''


vuelos.to_csv('vuelosDatosAtmosfericos.csv', sep=',', index=False)
```

4 PASO 3

Se obtienen las estaciones metereólogicas de dónde se obtendrá el tiempo. Hay que bajarse el siguiente fichero y renombrarlo como "stations.txt" : https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt

Se puede usar el siguiente comando en entornos unix: !wget http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt -O stations.txt

```
In [ ]: !wget http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt -O s
        #si fallase el comando, habria que bajarse el archivo manualmente y renombr
```

```
In [ ]: stationstxt = ""
        with open("stations.txt") as input:
            stationstxt = input.read()

        #Extract the data from file
        stations2 = stationstxt.split("\n")
        #Remove last line
        stations2 = stations2[:-1]
        stations2 = map(lambda line: [line[0:11],float(line[13:20]),float(line[22:3
```

5 PASO 4

A continuación se obtendrá los ficheros y cargará en un dataframe con la información de las estaciones y su localización. Además se declaran varias funciones para calcular que estación es la más cercana a cada vuelo.

Hay que tener en cuenta que los ficheros de datos de Noaa tienen la información anual.
https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2017.csv.gz
https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2016.csv.gz
https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2015.csv.gz
https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2014.csv.gz
https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2013.csv.gz
https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2012.csv.gz
https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2011.csv.gz
Por lo que hay que ir ejecutando cambiando la variable "year".

6 PASO 5

En este paso se indica con que se año se desea trabajar. Se ha desarrollado el script para obtener y tratar datos de un año en concreto.

```
In [ ]: year = '2017'
```

```
In [ ]: u = 'http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/'+year+'.csv.gz'
        w = year+'.csv.gz'
        !wget $u
        !gunzip $w

        #si el comando falla, recomendamos bajar manualmente el fichero de la ruta.
```

7 PASO 6

```
In [ ]: start = time.time()

        weatherdf = pd.read_csv(year+".csv",header=None)
        weatherdf.columns = ["id","date","type","Value1","Value2","Value3","Value4"

        ##nos quedamos con las latitudes de las estaciones.
        stationsdf = pd.DataFrame(stations2)
        stationsdf.columns = ["id","lat","lng","name"]
        lats = stationsdf[['lat','lng']]

        #funciones para calcular la distancia entre los aeropuertos y estacion mete
        def distance(lat1, lon1, lat2, lon2):
            p = 0.017453292519943295
            a = 0.5 - cos((lat2-lat1)*p)/2 + cos(lat1*p)*cos(lat2*p) * (1-cos((lon2
            return 12742 * asin(sqrt(a))

        def closest(data, v):
            return min(data, key=lambda p: distance(v[0],v[1],p[0],p[1]))


        #para que la fecha no de problemas al comparar (quizas no este bien del too
```

```
def tratarFecha(fecha):
    fecha = str(fecha).split(' ')[0]
    fecha = fecha.replace('-','')
    fecha = int(fecha)
    return fecha



del weatherdf['Value3']
del weatherdf['Value4']
del weatherdf['Value5']
del weatherdf['Value2']
weather = weatherdf ##creamos bck



def obtenerEstacion2(row):
    lat = row['board_lat']
    lon = row['board_lon']
    coor = [lat,lon]
    EMC= closest(np.asarray(lats), coor)
    ##obtenemos el id de la estacion de board
    aux = stationsdf[stationsdf['lat']==EMC[0]]
    aux = stationsdf[stationsdf['lng']==EMC[1]]
    EMC = aux['id']
    s =EMC.to_string()
    stationid = str(s)
    row['stationid'] = stationid.split()[1]
    return row

def obtenerEstacion3(row):
    lat = row['off_lat']
    lon = row['off_lon']
    coor = [lat,lon]
    EMC= closest(np.asarray(lats), coor)
    ##obtenemos el id de la estacion de board
    aux = stationsdf[stationsdf['lat']==EMC[0]]
    aux = stationsdf[stationsdf['lng']==EMC[1]]
    EMC = aux['id']
    s =EMC.to_string()
    stationid = str(s)
    row['stationid'] = stationid.split()[1]
    return row

def aplicarEstacionesOrigen(row):
    row['board_stationid']= coordenadasOrigen[(coordenadasOrigen.board_lat
    return row
```

```
        def aplicarEstacionesDestino(row):
            row['off_stationid']= coordenadasDestino[(coordenadasDestino.off_lat ==
            return row


        end = time.time()
        print(end - start)
```

8 PASO 7

```
In [ ]: #aplicamos y obtenemos las estaciones
        if 'board_stationid' in vuelos:
            pass
        else:
            coordenadasOrigen = vuelos[['board_lat','board_lon']]
            coordenadasDestino = vuelos[['off_lat','off_lon']]
            coordenadasOrigen = coordenadasOrigen.drop_duplicates(subset=['board_la
            coordenadasDestino = coordenadasDestino.drop_duplicates(subset=['off_la
            coordenadasOrigen = coordenadasOrigen.apply(lambda x:obtenerEstacion2(x
            coordenadasDestino = coordenadasDestino.apply(lambda x:obtenerEstacion3
            vuelos = vuelos.apply(lambda x: aplicarEstacionesOrigen(x),axis = 1)
            vuelos = vuelos.apply(lambda x: aplicarEstacionesDestino(x),axis = 1)
```

9 PASO 8

La siguiente función trata y junta los datos.

```
In [ ]: def tratar2(row):

            board_station = row['board_stationid']
            off_station = row['off_stationid']
            fecha = row['actual_time_of_departure']
            fecha_d = row['actual_time_of_arrival']
            ################################
            #primero tratamos las fechas
            fecha = tratarFecha(fecha)
            fecha_d = tratarFecha(fecha_d)
            ##############################################################################
            data_board = weatherdf[(weatherdf["id"]==board_station)&(weatherdf['dat
            data_off   = weatherdf[(weatherdf["id"]==off_station)&(weatherdf['date'
            ###################################################
            data_board["Value1"]=data_board["Value1"]/10
            data_board = data_board.reset_index()
            data_off["Value1"]=data_off["Value1"]/10
            data_off = data_off.reset_index()
                ########################################################
            T1 = data_board[data_board['type']=='TMIN']['Value1']
```

```
T2 = data_off[data_off['type']=='TMIN']['Value1']
T3 = data_board[data_board['type']=='TMAX']['Value1']
T4 = data_off[data_off['type']=='TMAX']['Value1']
T5 = data_board[data_board['type']=='TAVG']['Value1']
T6 = data_off[data_off['type']=='TAVG']['Value1']
T7 = data_board[data_board['type']=='SNOW']['Value1']
T8 = data_off[data_off['type']=='SNOW']['Value1']
T9 = data_board[data_board['type']=='PRCP']['Value1']
T10 = data_off[data_off['type']=='PRCP']['Value1']
T11 = data_board[data_board['type']=='SNWD']['Value1']
T12 = data_off[data_off['type']=='SNWD']['Value1']
T13 = data_board[data_board['type']=='ACMC']['Value1']
T14 = data_off[data_off['type']=='ACMC']['Value1']
T17 = data_board[data_board['type']=='ACSC']['Value1']
T18 = data_off[data_off['type']=='ACSC']['Value1']
T21 = data_board[data_board['type']=='AWDR']['Value1']
T22 = data_off[data_off['type']=='AWDR']['Value1']
T23 = data_board[data_board['type']=='AWND']['Value1']
T24 = data_off[data_off['type']=='AWND']['Value1']
T39 = data_board[data_board['type']=='EVAP']['Value1']
T40 = data_off[data_off['type']=='EVAP']['Value1']
T47 = data_board[data_board['type']=='FRTH']['Value1']
T48 = data_off[data_off['type']=='FRTH']['Value1']
T75 = data_board[data_board['type']=='TSUN']['Value1']
T76 = data_off[data_off['type']=='TSUN']['Value1']
T89 = data_board[data_board['type']=='WDMV']['Value1']
T90 = data_off[data_off['type']=='WDMV']['Value1']


##############################
##tratamos los valores nulos aplicando un valor nulo
if T1.empty:
    T1 = 9999
if T2.empty:
    T2 = 9999
if T3.empty:
    T3 = 9999
if T4.empty:
    T4 = 9999
if T5.empty:
    T5 = 9999
if T6.empty:
    T6 = 9999
if T7.empty:
    T7 = 9999
if T8.empty:
    T8 = 9999
if T9.empty:
```

```
    T9 = 9999
if T10.empty:
    T10 = 9999
if T11.empty:
    T11 = 9999
if T12.empty:
    T12 = 9999
if T13.empty:
    T13 = 9999
if T14.empty:
    T14 = 9999
if T17.empty:
    T17 = 9999
if T18.empty:
    T18 = 9999
if T21.empty:
    T21 = 9999
if T22.empty:
    T22 = 9999
if T23.empty:
    T23 = 9999
if T24.empty:
    T24 = 9999
if T39.empty:
    T39 = 9999
if T40.empty:
    T40 = 9999
if T47.empty:
    T47 = 9999
if T48.empty:
    T48 = 9999
if T75.empty:
    T75 = 9999
if T76.empty:
    T76 = 9999
if T89.empty:
    T89 = 9999
if T90.empty:
    T90 = 9999


#actualizamos la fila
row['TMIN_o']= float(T1)
row['TMIN_d']= float(T2)
row['TMAX_o']= float(T3)
row['TMAX_d']= float(T4)
row['TAVG_o']= float(T5)
row['TAVG_d']= float(T6)
```

```
        row['SNOW_o']= float(T7)
        row['SNOW_d']= float(T8)
        row['PRCP_o']= float(T9)
        row['PRCP_d']= float(T10)
        row['SNWD_o']= float(T11)
        row['SNWD_d']= float(T12)
        row['ACMC_o']= float(T13)
        row['ACMC_d']= float(T14)
        row['ACSC_o']= float(T17)
        row['ACSC_d']= float(T18)
        row['AWDR_o']= float(T21)
        row['AWDR_d']= float(T22)
        row['AWND_o']= float(T23)
        row['AWND_d']= float(T24)
        row['EVAP_o']= float(T39)
        row['EVAP_d']= float(T40)
        row['FRTH_o']= float(T47)
        row['FRTH_d']= float(T48)
        row['TSUN_o']= float(T75)
        row['TSUN_d']= float(T76)
        row['WDMV_o']= float(T89)
        row['WDMV_d']= float(T90)


        return row

    def functionWeatherDay(year, month, day):
        u = int(month+day)
        d = int(year*10000)
        weatherdf = weather##restauramos el bck
        weatherdf = weatherdf[weatherdf['date']-d==u]
        return weatherdf

    def functionPlus(year, month, day):
        start = time.time()
        vuelosSinTratar = vuelos[vuelos['TAVG_o'].isnull()]
        x = vuelosSinTratar[vuelosSinTratar['anyoSalida']==int(year)]
        x = x[x['mesSalida']==month]
        x = x[x['diaSalida']==day]
        x = x.apply(lambda x:tratar2(x),axis = 1)
        #para escribir ya
        #variable auxiliar

        pd.options.mode.chained_assignment = None  # default='warn'
        vuelosSinTratar['TMIN_o'] = x.set_index(['index'])['TMIN_o'].combine_fi
        vuelosSinTratar['TMIN_d'] = x.set_index(['index'])['TMIN_d'].combine_fi
        vuelosSinTratar['TMAX_o'] = x.set_index(['index'])['TMAX_o'].combine_fi
        vuelosSinTratar['TMAX_d'] = x.set_index(['index'])['TMAX_d'].combine_fi
```

```
vuelosSinTratar['TAVG_o'] = x.set_index(['index'])['TAVG_o'].combine_fi
vuelosSinTratar['TAVG_d'] = x.set_index(['index'])['TAVG_d'].combine_fi
vuelosSinTratar['SNOW_o'] = x.set_index(['index'])['SNOW_o'].combine_fi
vuelosSinTratar['SNOW_d'] = x.set_index(['index'])['SNOW_d'].combine_fi
vuelosSinTratar['PRCP_o'] = x.set_index(['index'])['PRCP_o'].combine_fi
vuelosSinTratar['PRCP_d'] = x.set_index(['index'])['PRCP_d'].combine_fi
vuelosSinTratar['SNWD_o'] = x.set_index(['index'])['SNWD_o'].combine_fi
vuelosSinTratar['SNWD_d'] = x.set_index(['index'])['SNWD_d'].combine_fi
vuelosSinTratar['ACMC_o'] = x.set_index(['index'])['ACMC_o'].combine_fi
vuelosSinTratar['ACMC_d'] = x.set_index(['index'])['ACMC_d'].combine_fi
vuelosSinTratar['ACSC_o'] = x.set_index(['index'])['ACSC_o'].combine_fi
vuelosSinTratar['ACSC_d'] = x.set_index(['index'])['ACSC_d'].combine_fi
vuelosSinTratar['AWDR_o'] = x.set_index(['index'])['AWDR_o'].combine_fi
vuelosSinTratar['AWDR_d'] = x.set_index(['index'])['AWDR_d'].combine_fi
vuelosSinTratar['AWND_o'] = x.set_index(['index'])['AWND_o'].combine_fi
vuelosSinTratar['AWND_d'] = x.set_index(['index'])['AWND_d'].combine_fi
vuelosSinTratar['EVAP_o'] = x.set_index(['index'])['EVAP_o'].combine_fi
vuelosSinTratar['EVAP_d'] = x.set_index(['index'])['EVAP_d'].combine_fi
vuelosSinTratar['FRTH_o'] = x.set_index(['index'])['FRTH_o'].combine_fi
vuelosSinTratar['FRTH_d'] = x.set_index(['index'])['FRTH_d'].combine_fi
vuelosSinTratar['TSUN_o'] = x.set_index(['index'])['TSUN_o'].combine_fi
vuelosSinTratar['TSUN_d'] = x.set_index(['index'])['TSUN_d'].combine_fi
vuelosSinTratar['WDMV_o'] = x.set_index(['index'])['WDMV_o'].combine_fi
vuelosSinTratar['WDMV_d'] = x.set_index(['index'])['WDMV_d'].combine_fi
```

```
vuelos['TMIN_o'] = vuelosSinTratar.set_index(['index'])['TMIN_o'].combi
vuelos['TMIN_d'] = vuelosSinTratar.set_index(['index'])['TMIN_d'].combi
vuelos['TMAX_o'] = vuelosSinTratar.set_index(['index'])['TMAX_o'].combi
vuelos['TMAX_d'] = vuelosSinTratar.set_index(['index'])['TMAX_d'].combi
vuelos['TAVG_o'] = vuelosSinTratar.set_index(['index'])['TAVG_o'].combi
vuelos['TAVG_d'] = vuelosSinTratar.set_index(['index'])['TAVG_d'].combi
vuelos['SNOW_o'] = vuelosSinTratar.set_index(['index'])['SNOW_o'].combi
vuelos['SNOW_d'] = vuelosSinTratar.set_index(['index'])['SNOW_d'].combi
vuelos['PRCP_o'] = vuelosSinTratar.set_index(['index'])['PRCP_o'].combi
vuelos['PRCP_d'] = vuelosSinTratar.set_index(['index'])['PRCP_d'].combi
vuelos['SNWD_o'] = vuelosSinTratar.set_index(['index'])['SNWD_o'].combi
vuelos['SNWD_d'] = vuelosSinTratar.set_index(['index'])['SNWD_d'].combi
vuelos['ACMC_o'] = vuelosSinTratar.set_index(['index'])['ACMC_o'].combi
vuelos['ACMC_d'] = vuelosSinTratar.set_index(['index'])['ACMC_d'].combi
vuelos['ACSC_o'] = vuelosSinTratar.set_index(['index'])['ACSC_o'].combi
vuelos['ACSC_d'] = vuelosSinTratar.set_index(['index'])['ACSC_d'].combi
vuelos['AWDR_o'] = vuelosSinTratar.set_index(['index'])['AWDR_o'].combi
vuelos['AWDR_d'] = vuelosSinTratar.set_index(['index'])['AWDR_d'].combi
vuelos['AWND_o'] = vuelosSinTratar.set_index(['index'])['AWND_o'].combi
vuelos['AWND_d'] = vuelosSinTratar.set_index(['index'])['AWND_d'].combi
vuelos['EVAP_o'] = vuelosSinTratar.set_index(['index'])['EVAP_o'].combi
vuelos['EVAP_d'] = vuelosSinTratar.set_index(['index'])['EVAP_d'].combi
```

```
vuelos['FRTH_o'] = vuelosSinTratar.set_index(['index'])['FRTH_o'].combi
vuelos['FRTH_d'] = vuelosSinTratar.set_index(['index'])['FRTH_d'].combi
vuelos['TSUN_o'] = vuelosSinTratar.set_index(['index'])['TSUN_o'].combi
vuelos['TSUN_d'] = vuelosSinTratar.set_index(['index'])['TSUN_d'].combi
vuelos['WDMV_o'] = vuelosSinTratar.set_index(['index'])['WDMV_o'].combi
vuelos['WDMV_d'] = vuelosSinTratar.set_index(['index'])['WDMV_d'].combi


#vuelos.to_csv('vuelos.csv', sep=',', index=False)
end = time.time()
print(end - start)
```

10 PASO 9

La ejecución de la siguiente celda hace que se procesen todos los vuelos de los días y meses del año. Recordar que hay que repetir lo mismo para cada año.

```
In [ ]: #variables de inicio

        dia = 1
        mes = 1
        anyo = int(year)
        for mes in range(1,13):
            print 'empiezo el mes', mes
            mesw = str(mes)
            for dia in range(1,32):
                print 'empiezo el dia - ' ,dia
                diaw = str(dia)
                if dia < 10:
                    diaw = '0'+diaw

                weatherdf = functionWeatherDay(anyo,mesw,diaw)
                functionPlus(year,mes,dia)
```

11 PASO 10

```
In [ ]: vuelos.to_csv('vuelosDatosAtmosfericos.csv', sep=',', index=False)
```

NOTA IMPORTANTE, LEE ANTES DE CONTINUAR: Si se han procesado todos los años (2017, 2016, 2015, 2014, 2013, 2012, 2011) se puede continuar a la Segunda parte. Si no, se debe volver al Paso nº 5 y cambiar la variable "year" por el año que falte procesar. Es importante procesar todos los años, ya que ha sido lo que hemos hecho nosotros.

12 Segunda parte

Si se han procesado todos los años, se quiere decir que se han procesado todos los vuelos. Pero nos encontramos con el problema de tener demasiados nulos. Es decir, vuelos sin datos del tiempo.

Entonces se ha decidido buscar la siguiente estación más cercana para disminuir este problema.

13 PASO 11

```
In [ ]: def getLatstation(row):
            row['lat'] = stationsdf[stationsdf['id']==row['id']]['lat'].values[0]
            row['lng'] = stationsdf[stationsdf['id']==row['id']]['lng'].values[0]
            return row

        def obtenerEstacionBoard(row):
            lat = row['board_lat']
            lon = row['board_lon']
            coor = [lat,lon]
            EMC= closest(np.asarray(lista), coor)
            EMC[0] = round(EMC[0],3)
            EMC[1] = round(EMC[1],3)
            ##obtenemos el id de la estacion de board
            aux = stationsdf[(stationsdf.lat==EMC[0])|(stationsdf.lng==EMC[1])]
            EMC = aux['id']
            s = EMC.to_string()
            stationid = str(s)
            row['stationid'] = stationid.split()[1]
            return row


        def aplicarEstacionesOrigen(row):
            n = 'board_stationid_o_'+var
            row[n] = coordenadasOrigen[(coordenadasOrigen.board_lat == row['board_l
            return row

        def obtenerEstacionOff(row):
            lat = row['off_lat']
            lon = row['off_lon']
            coor = [lat,lon]
            EMC= closest(np.asarray(lista), coor)
            EMC[0] = round(EMC[0],3)
            EMC[1] = round(EMC[1],3)
            ##obtenemos el id de la estacion de board
            aux = stationsdf[(stationsdf.lat==EMC[0])|(stationsdf.lng==EMC[1])]
            EMC = aux['id']
            s = EMC.to_string()
            stationid = str(s)
            row['stationid'] = stationid.split()[1]
            return row

        def aplicarEstacionesDestino(row):
            n = 'off_stationid_d_'+var
```

```
    row[n] = coordenadasDestino[(coordenadasDestino.off_lat == row['off_lat
    return row

def roundF(row):
    row['lat'] = round(row['lat'],3)
    return row

def roundFF(row):
    row['lng'] = round(row['lng'],3)
    return row


def obtenerDatoEstacion(row):
    if var_.split('_')[1]=='o':
        u = 'board_stationid_o_'+ var
    else:
        u = 'off_stationid_d_'+ var

    board_station = row[u]
    fechaAux = tratarFecha(row['actual_time_of_departure'])
    x = stationsX[stationsX.date == fechaAux]
    if x.empty:
        pass
    else:
        x= x[x.id == board_station]
        if x.empty:
            pass

        else:
            row[var_] = float(x.Value1.values[0])/10

    del x
    return row

def functionGest(year, month, day):

    vuelosSinTratar = vuelos[vuelos[var_]==9999.0]
    x = vuelosSinTratar[vuelosSinTratar['anyoSalida']==year]
    x = x[x['mesSalida']==month]
    x = x[x['diaSalida']==day]
    x = x.apply(lambda x:obtenerDatoEstacion(x),axis = 1)
    pd.options.mode.chained_assignment = None
    vuelosSinTratar[var_] = x.set_index(['index'])[var_].combine_first(vuel
    vuelos[var_] = vuelosSinTratar.set_index(['index'])[var_].combine_first
```

14 PASO 12

Como en pasos anteriores la siguiente celda habrá que modificarla y ejecutar los siguientes pasos varias veces. Una para cada dato a procesar.

(Se intentó montar un bucle, pero como tarda mucho, se ha particionado la ejecución para controlar la ejecución)

Para las siguientes variables de las cuales queremos quitar nulos, hay que hacer ejecuciones por cada año. - PRCP_o
- PRCP_d - TAVG_o - TAVG_d - TMAX_o - TMAX_d - TMIN_o - TMIN_d

NOTA IMPORTANTE:

Hay 7 años y 8 variables,y son muchas ejecuciones. Hay años con pocos datos como de 2011 - 2013 . Y se podría saltar si se desea. Nosotros hemos ejecutado todos los años. Se puede coger el año 2016 por ejemplo para seguir. Si no se tiene demasiado tiempo para ejecutar. Se dejo así por que era más cómodo para controlar lo que ibamos ejecutando y tratando.

```
In [ ]: weatherdf = weather #reseteamos los datos del tiempo por si acaso.
```

```
In [ ]: year = '2017'  #cambiar el año para cada ciclo
```

```
In [ ]: stationstxt = ""
        with open("stations.txt") as input:
            stationstxt = input.read()

        stations2 = stationstxt.split("\n")
        stations2 = stations2[:-1]
        stations2 = map(lambda line: [line[0:11],float(line[13:20]),float(line[22:3

        ########################  ir comentando y descomentando (dejar solo 1 desc
        var_ = "PRCP_o"
        #var_ = "PRCP_d"
        #var_ = "TAVG_o"
        #var_ = "TAVG_d"
        #var_ = "TMAX_o"
        #var_ = "TMAX_d"
        #var_ = "TMIN_o"
        #var_ = "TMIN_d"
        ########################


        var = var_.split("_")[0]
        stationsX = weatherdf[weatherdf["type"]==var]
        stationsX = stationsX.sort_values(by=['id'])
```

15 PASO 13

```
In [ ]: #suele tardar un poco.
```

```
f = 'lista'+var+year+'.csv'

if os.path.exists(f):
    lista = pd.read_csv(f,sep = ',')
    del lista['id']
else:
    lista = stationsX['id'].unique()
    lista = pd.DataFrame(lista)
    lista.columns = ['id']
    lista = lista.apply(lambda x:getLatstation(x),axis=1)
    lista.to_csv(f, sep=',', index=False)
    del lista['id']
```

16 PASO 14

```
In [ ]: m = 'board_stationid_o_'+var
        if m in vuelos:
            pass
        else:
            stationsdf = stationsdf.apply(lambda x: roundF(x),axis = 1)
            stationsdf = stationsdf.apply(lambda x: roundFF(x),axis = 1)

            coordenadasOrigen = vuelos[['board_lat','board_lon']]
            coordenadasOrigen = coordenadasOrigen.drop_duplicates(subset=['board_la
            coordenadasOrigen = coordenadasOrigen.apply(lambda x:obtenerEstacionBoa
            vuelos = vuelos.apply(lambda x: aplicarEstacionesOrigen(x),axis = 1)

        vuelos.to_csv('vuelosDatosAtmosfericos.csv', sep=',', index=False)
```

17 PASO 15

```
In [ ]: m = 'off_stationid_d_'+var
        if m in vuelos:
            pass
        else:
            stationsdf = stationsdf.apply(lambda x: roundF(x),axis = 1)
            stationsdf = stationsdf.apply(lambda x: roundFF(x),axis = 1)

            coordenadasDestino = vuelos[['off_lat','off_lon']]
            coordenadasDestino = coordenadasDestino.drop_duplicates(subset=['off_la
            coordenadasDestino = coordenadasDestino.apply(lambda x:obtenerEstacionD
            vuelos = vuelos.apply(lambda x: aplicarEstacionesDestino(x),axis = 1)
```

18 PASO 16

Los siguientes bucles repasan día a día los vuelos del año y rellenan los datos.

```
In [ ]: vuelos.to_csv('vuelos.csv', sep=',', index=False)
        year2 = int(year)
        for mes in range(1,13):
            print 'empiezo el mes', mes
            mesw = str(mes)
            for dia in range(1,32):
                print 'empiezo el dia - ' ,dia
                diaw = str(dia)
                if dia < 10:
                    diaw = '0'+diaw

                functionGest(year2,mes,dia)

        vuelos.to_csv('vuelosDatosAtmosfericos.csv', sep=',', index=False)
```

IMPORTANTE: si no se han procesado todos los datos es necesario ir al paso 12 de nuevo y cambiar la configuración con nuevos datos a procesar.

19 PASO 17

Borramos columnas que no se necesitarán

```
In [ ]: del vuelos['ACMC_o']
        del vuelos['ACMC_d']
        del vuelos['ACSC_o']
        del vuelos['ACSC_d']
        del vuelos['AWDR_o']
        del vuelos['AWDR_d']
        del vuelos['AWND_o']
        del vuelos['AWND_d']
        del vuelos['EVAP_o']
        del vuelos['EVAP_d']
        del vuelos['FRTH_o']
        del vuelos['FRTH_d']
        del vuelos['TSUN_o']
        del vuelos['TSUN_d']
        del vuelos['WDMV_o']
        del vuelos['WDMV_d']
        del vuelos['board_stationid']
        del vuelos['off_stationid']
        del vuelos['board_stationid_o_TMIN']
        del vuelos['board_stationid_o_TMAX']
        del vuelos['board_stationid_o_TAVG']
        del vuelos['board_stationid_o_PRCP']
        del vuelos['off_stationid_d_PRCP']
        del vuelos['off_stationid_d_TAVG']
        del vuelos['off_stationid_d_TMAX']
        del vuelos['off_stationid_d_TMIN']
```

```
del vuelos['SNWD_d']
del vuelos['SNWD_o']
```

20 PASO 18

Dejamos a nulos lo valores no tratados o no encontrados

```
In [ ]: def functionTreatNull(row):

            if row.TMIN_o == 9999.0:
                row.TMIN_o = None
            if row.TMIN_d == 9999.0:
                row.TMIN_d = None
            if row.TMAX_o == 9999.0:
                row.TMAX_o = None
            if row.TMAX_d == 9999.0:
                row.TMAX_d = None
            if row.TAVG_o == 9999.0:
                row.TAVG_o = None
            if row.TAVG_d == 9999.0:
                row.TAVG_d = None
            if row.SNOW_o == 9999.0:
                row.SNOW_o = None
            if row.SNOW_d == 9999.0:
                row.SNOW_d = None
            if row.PRCP_o == 9999.0:
                row.PRCP_o = None
            if row.PRCP_d == 9999.0:
                row.PRCP_d = None
            if row.SNWD_o == 9999.0:
                row.SNWD_o = None
            if row.SNWD_d == 9999.0:
                row.SNWD_d = None

            return row

        vuelos = vuelos.apply(lambda x: functionTreatNull(x), axis=1)
```

21 PASO 19

```
In [ ]: vuelos.to_csv('vuelosDatosAtmosfericos.csv', sep=',', index=False)
```

Escribimos los datos en un fichero csv para seguir trabajando en el siguiente paso.