# Practica 3: Sondeos meteorológicos de la Atmósfera

Barajas Ibarria Juan Pedro

14 de Febrero del 2018

## 1. Introducción

Un sondeo es realizar la exploración de un terreno, en el contexto de este reporte un sondeo atmosférico es donde se analiza la distribución física de las propiedades atmosféricas como lo son la presión, temperatura, velocidad y dirección de los vientos, humedad, etc. [1]

Este sondeo se puede llevar a cabo gracias a la utilización de un globo meteorológico que es una especie de globo aerostático que se utiliza para dar información sobre las propiedades atmosféricas ya mencionadas. [2]

Este tipo de globo aerostático se utiliza para elevar instrumentos de medición a alturas de alrededor de los 8 kilómetros en donde se encuentra la estratosfera. Con este globo también es posible medir la radiación ultravioleta (UV).

Encontramos como sustento los datos proporcionados el sondeo atmosférico de la Universidad de Wyoming, que fueron realizados por el departamento de ciencias atmosféricas de la misma universidad.

Este análisis de datos corresponde a la ciudad de Brest, Francia la cual es una ciudad situada en el departamento de Finisterre, en la región de Bretaña. Para esto utilizamos Jupyter notebook con el lenguaje Python 3 el cual nos facilito de manera extraordinaria el análisis de datos para encontrar el sentido físico de los datos recabados.

## 2. Análisis de datos

Al momento de hacer un sondeo meteorológico al nivel básico en el que nos encontramos, primeramente descargamos los datos el sitio de sondeo atmosféricos de la Universidad de Wyoming. [3]

Estos datos al momento de descargar tuvimos ciertas dificultades con ellos, como lo fueron las líneas de innecesarias o el tipo de objeto en el cual estaban configurados los datos.

Primero subimos los archivos a pandas, son dos los cuales son los de Diciembre y de Junio diferenciados como BrestD.txt y BrestJ.txt respectivamente

```
In [2]:  df0 = pd.read_csv('BrestD.txt', skiprows=3, sep='\s+')
         df1 = pd.read_csv('BrestJ.txt', skiprows=3, sep='\s+')
```

En la presentación de los datos vemos que tienen defectos como espacios innecesarios con los dos archivos

```
In [3]: df0.head()
```

Out[3]:

| | PRES | HGHT | TEMP | DWPT | RELH | \ |
|---|---|---|---|---|---|---|
| 0 | hPa | m | C | C | % | |
| 1 | ----------...- | NaN | NaN | NaN | NaN | |
| 2 | 1026.0 | 98 | 10.8 | 10.3 | 97 | |
| 3 | 1022.0 | 130 | 11.2 | 7.3 | 77 | |
| 4 | 1010.0 | 228 | 10.4 | 9.9 | 97 | |

| | MIXR | DRCT | SKNT | THTA | THTE | THTV |
|---|---|---|---|---|---|---|
| 0 | g/kg | deg | knot | K | K | K |
| 1 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 7.72 | 260 | 8 | 281.9 | 303.2 | 283.2 |
| 3 | 6.31 | 261 | 8 | 282.6 | 300.2 | 283.7 |
| 4 | 7.63 | 263 | 9 | 282.8 | 303.9 | 284.1 |

```
In [4]: df1.head()
```

Out[4]:

| | PRES | HGHT | TEMP | DWPT | RELH | \ |
|---|---|---|---|---|---|---|
| 0 | hPa | m | C | C | % | |
| 1 | ----------...- | NaN | NaN | NaN | NaN | |
| 2 | 1006.0 | 98 | 21.2 | 14.2 | 64 | |
| 3 | 1000.0 | 146 | 19.2 | 14.2 | 73 | |
| 4 | 999.0 | 155 | 18.8 | 13.9 | 73 | |

| | MIXR | DRCT | SKNT | THTA | THTE | THTV |
|---|---|---|---|---|---|---|
| 0 | g/kg | deg | knot | K | K | K |
| 1 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 10.21 | 295 | 5 | 293.9 | 323.3 | 295.6 |
| 3 | 10.28 | 290 | 6 | 292.4 | 321.8 | 294.2 |
| 4 | 10.09 | 290 | 6 | 292.0 | 320.9 | 293.8 |

Se procede a reparar a estructura de los datos eliminando renglones y acomodando las columnas. Vemos que ya tienen una estructura aceptable.

```
In [5]: #Elimina columnas de datos, en este caso son renglones
        dfD=df0.drop(df0.index[[0,1]])

In [6]: dfJ=df1.drop(df1.index[[0,1]])

In [7]: dfD.head()
```

Out[7]:

| | PRES | HGHT | TEMP | DWPT | RELH | MIXR | DRCT | SKNT | THTA | THTE | THTV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1026.0 | 98 | 10.8 | 10.3 | 97 | 7.72 | 260 | 8 | 281.9 | 303.2 | 283.2 |
| 3 | 1022.0 | 130 | 11.2 | 7.3 | 77 | 6.31 | 261 | 8 | 282.6 | 300.2 | 283.7 |
| 4 | 1010.0 | 228 | 10.4 | 9.9 | 97 | 7.63 | 263 | 9 | 282.8 | 303.9 | 284.1 |
| 5 | 1000.0 | 310 | 10.0 | 9.5 | 97 | 7.50 | 265 | 9 | 283.1 | 304.0 | 284.4 |
| 6 | 925.0 | 955 | 6.6 | 6.0 | 96 | 6.38 | 285 | 11 | 286.1 | 304.1 | 287.1 |

```
In [8]: dfJ.head()
```

Out[8]:

| | PRES | HGHT | TEMP | DWPT | RELH | MIXR | DRCT | SKNT | THTA | THTE | THTV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1006.0 | 98 | 21.2 | 14.2 | 64 | 10.21 | 295 | 5 | 293.9 | 323.3 | 295.6 |
| 3 | 1000.0 | 146 | 19.2 | 14.2 | 73 | 10.28 | 290 | 6 | 292.4 | 321.8 | 294.2 |
| 4 | 999.0 | 155 | 18.8 | 13.9 | 73 | 10.09 | 290 | 6 | 292.0 | 320.9 | 293.8 |
| 5 | 925.0 | 809 | 12.6 | 11.3 | 92 | 9.17 | 280 | 17 | 292.2 | 318.5 | 293.8 |
| 6 | 904.0 | 1001 | 10.6 | 10.3 | 98 | 8.77 | 276 | 21 | 292.1 | 317.2 | 293.6 |

Damos la estructura de datos necesaria para la manipulación.

```
In [9]: # Dar estructura de datos (DataFrame)
        dfD1 = pd.DataFrame(dfD)
        dfJ1 = pd.DataFrame(dfJ)
```

Vemos que los datos tuvieron una complicación ya que al momento de guardarlos en un arreglo de datos estos fueron por default objetos lo que complica la lectura de graficación, para esto realizamos un cambio de tipo, esto fue pasando de datos .ºbjeto» "numérico".

```
In [10]: #Cambio de tipo de dato de cada una de las columnas.
         dfD1.PRES=pd.to_numeric(dfD1.PRES, errors='raise', downcast=None)
         dfJ1.PRES=pd.to_numeric(dfJ1.PRES, errors='raise', downcast=None)

In [11]: dfD1.HGHT=pd.to_numeric(dfD1.HGHT, errors='raise', downcast=None)
         dfJ1.HGHT=pd.to_numeric(dfJ1.HGHT, errors='raise', downcast=None)

In [12]: dfD1.TEMP=pd.to_numeric(dfD1.TEMP, errors='raise', downcast=None)
         dfJ1.TEMP=pd.to_numeric(dfJ1.TEMP, errors='raise', downcast=None)

In [13]: dfD1.DWPT=pd.to_numeric(dfD1.DWPT, errors='raise', downcast=None)
         dfJ1.DWPT=pd.to_numeric(dfJ1.DWPT, errors='raise', downcast=None)

In [14]: dfD1.RELH=pd.to_numeric(dfD1.RELH, errors='raise', downcast=None)
         dfJ1.RELH=pd.to_numeric(dfJ1.RELH, errors='raise', downcast=None)

In [15]: dfD1.MIXR=pd.to_numeric(dfD1.MIXR, errors='raise', downcast=None)
         dfJ1.MIXR=pd.to_numeric(dfJ1.MIXR, errors='raise', downcast=None)
```

```
In [16]: dfD1.MIXR=pd.to_numeric(dfD1.MIXR, errors='raise', downcast=None)
         dfJ1.MIXR=pd.to_numeric(dfJ1.MIXR, errors='raise', downcast=None)

In [17]: dfD1.DRCT=pd.to_numeric(dfD1.DRCT, errors='raise', downcast=None)
         dfJ1.DRCT=pd.to_numeric(dfJ1.DRCT, errors='raise', downcast=None)

In [18]: dfD1.SKNT=pd.to_numeric(dfD1.SKNT, errors='raise', downcast=None)
         dfJ1.SKNT=pd.to_numeric(dfJ1.SKNT, errors='raise', downcast=None)

In [19]: # Dar estructura de datos (DataFrame)
         dfD1 = pd.DataFrame(dfD1)
         dfJ1 = pd.DataFrame(dfJ1)

In [20]: dfD1.dtypes
```

Vemos a continuación que los datos ya tienen una estructura numérica.

```
Out[20]: PRES    float64
         HGHT      int64
         TEMP    float64
         DWPT    float64
         RELH      int64
         MIXR    float64
         DRCT    float64
         SKNT    float64
         THTA     object
         THTE     object
         THTV     object
         dtype: object

In [21]: dfJ1.dtypes

Out[21]: PRES    float64
         HGHT      int64
         TEMP    float64
         DWPT    float64
         RELH      int64
         MIXR    float64
         DRCT      int64
         SKNT      int64
         THTA     object
         THTE     object
         THTV     object
         dtype: object
```

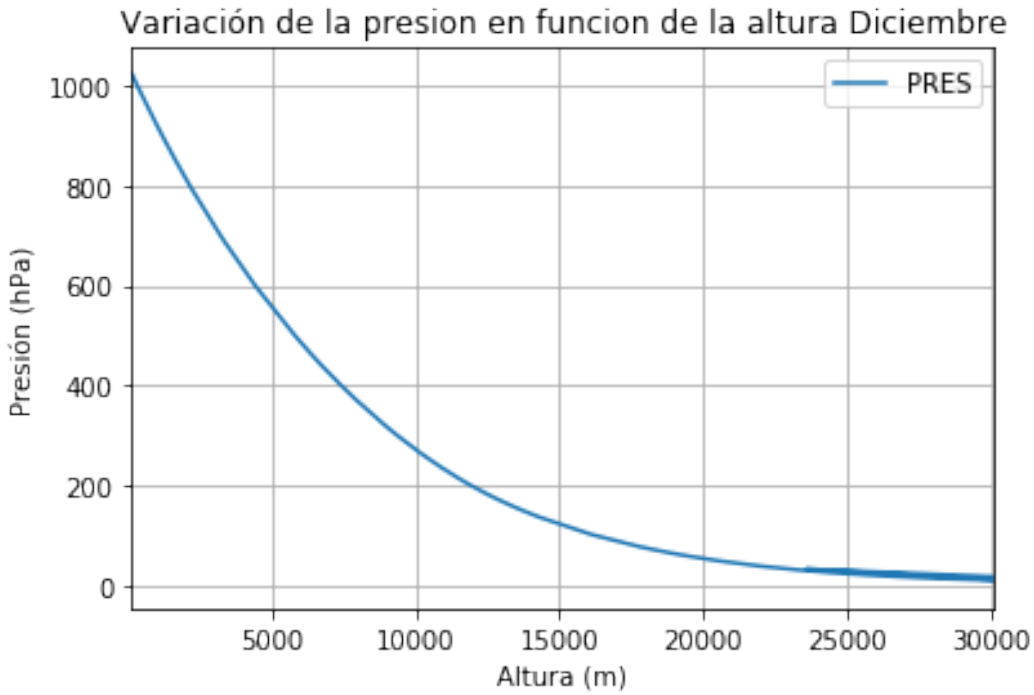
# 3. Resultados e interpretación

A continuación generamos las gráficas para poder darles una interpretación real a los datos.

1. La primeras dos gráficas corresponden al ejemplo reproducido de la presión con respecto a la altura.

```
In [28]: df01 = dfD1[['HGHT','PRES']]
         plot.figure(); df01.plot(x='HGHT'); plot.legend(loc='best')
         plot.title('Variación de la presion en funcion de la altura Diciembre')
         plot.ylabel('Presión (hPa)')
         plot.xlabel('Altura (m)')
         plot.grid(True)
         plot.show()
```

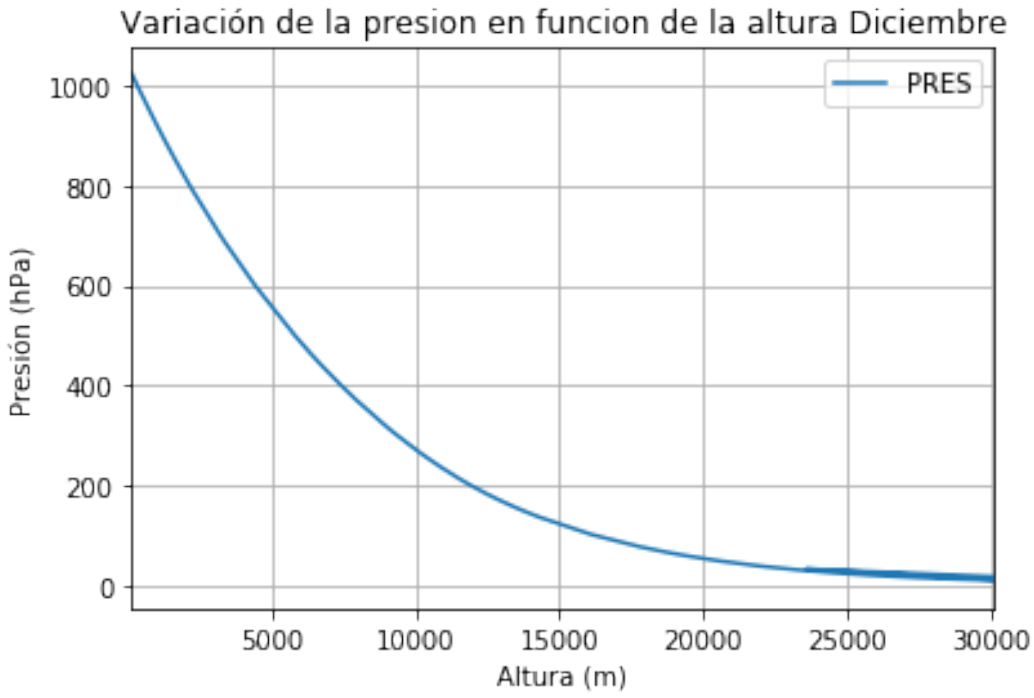
```
<matplotlib.figure.Figure at 0x7fd05c52ef98>
```




```
In [29]: df02 = dfJ1[['HGHT','PRES']]
         plot.figure(); df01.plot(x='HGHT'); plot.legend(loc='best')
         plot.title('Variación de la presion en funcion de la altura Diciembre')
```

```
plot.ylabel('Presión (hPa)')
plot.xlabel('Altura (m)')
plot.grid(True)
plot.show()
```

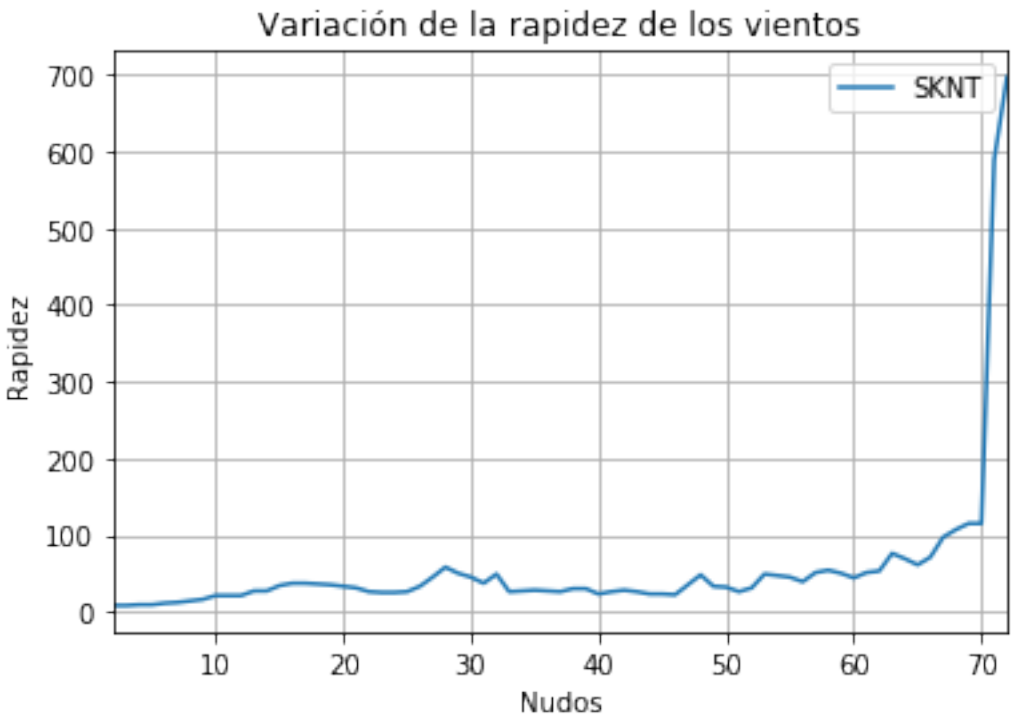
```
<matplotlib.figure.Figure at 0x7fd05c3b88d0>
```




2. Las segundas dos gráficas corresponden a la variación de los vientos, en como estos fueron variando dependiendo la altura en la que se encontraban la cual incrementaba conforme la altura crecía.

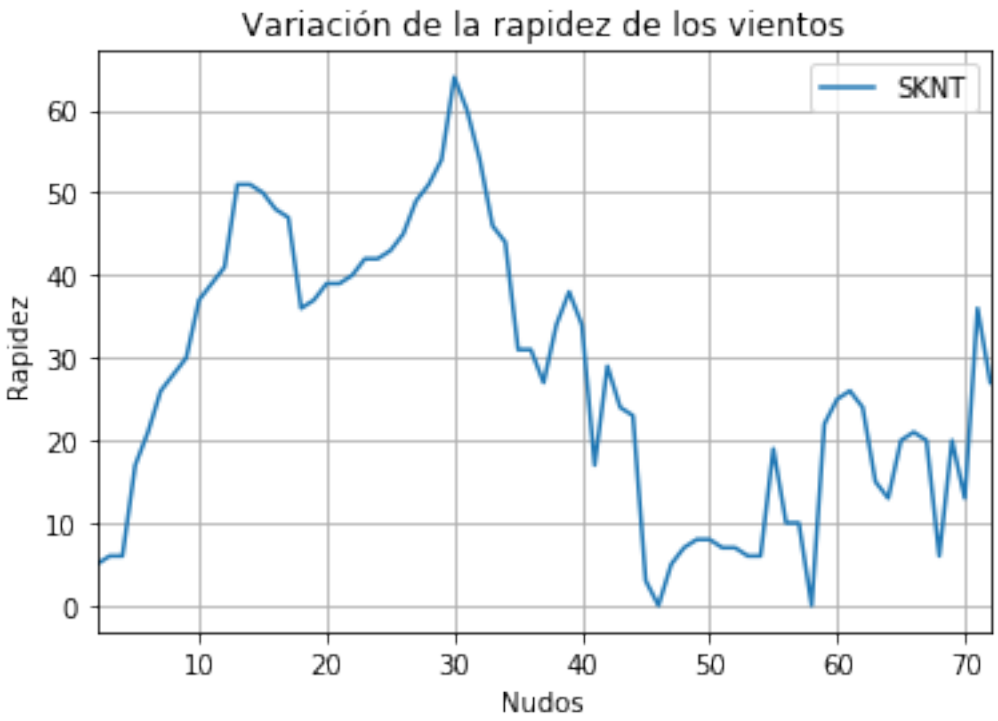
```
In [31]: df01 = dfD1[['SKNT']]
         plot.figure(); df01.plot(y='SKNT'); plot.legend(loc='best')
         plot.title('Variación de la rapidez de los vientos')
         plot.ylabel('Rapidez')
         plot.xlabel('Nudos')
         plot.grid(True)
         plot.show()
```

```
<matplotlib.figure.Figure at 0x7fd05c44c160>
```




```
In [32]: df01 = dfJ1[['SKNT']]
         plot.figure(); df01.plot(y='SKNT'); plot.legend(loc='best')
         plot.title('Variación de la rapidez de los vientos')
         plot.ylabel('Rapidez')
         plot.xlabel('Nudos')
         plot.grid(True)
         plot.show()
```

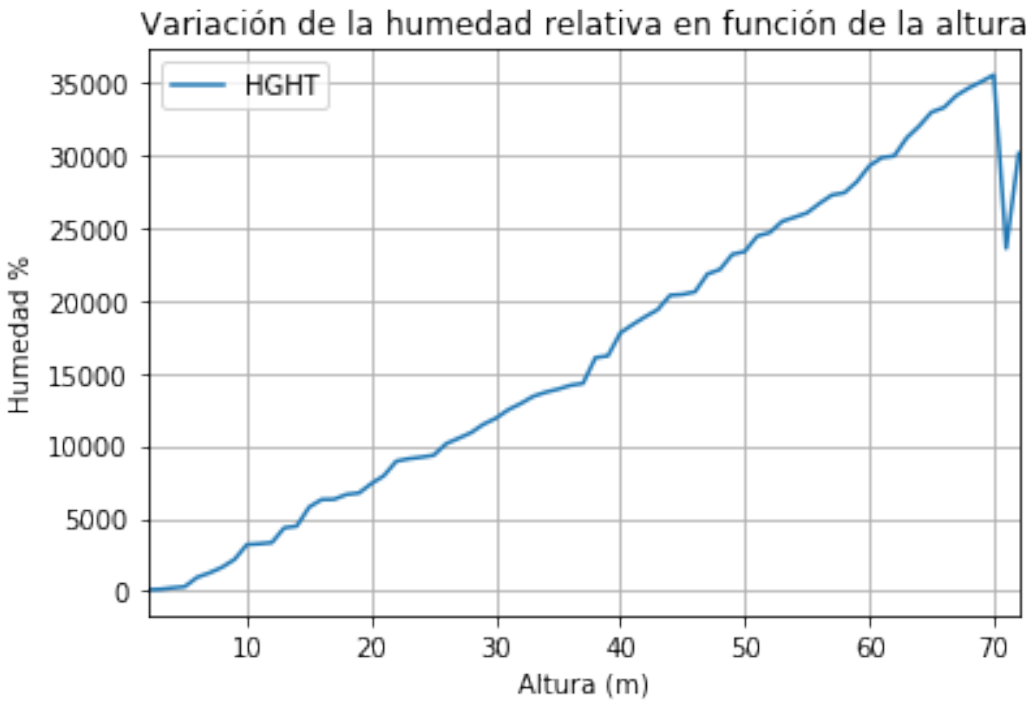
```
<matplotlib.figure.Figure at 0x7fd05c2a4198>
```




3. Después generamos las gráficas de humedad relativa con respecto a la altura en la que se encontraba el medidor.

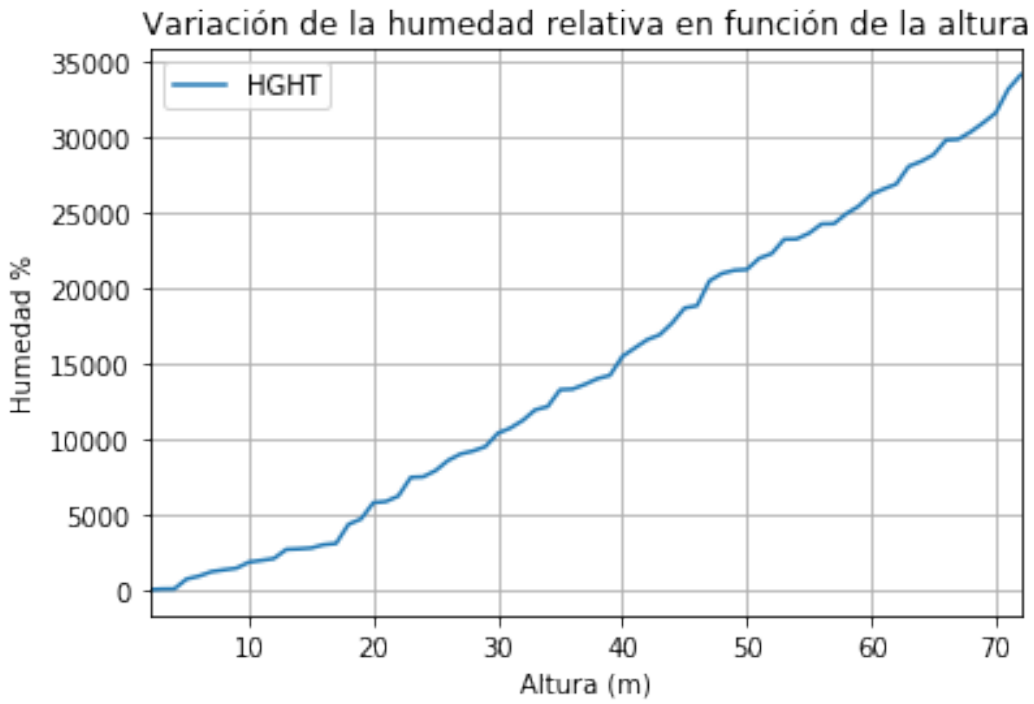
```
In [34]: df01 = dfD1[['HGHT','RELH']]
         plot.figure(); df01.plot(y='HGHT'); plot.legend(loc='best')
         plot.title('Variación de la humedad relativa en función de la altura')
         plot.ylabel('Humedad %')
         plot.xlabel('Altura (m)')
         plot.grid(True)
         plot.show()
```

```
<matplotlib.figure.Figure at 0x7fd05c2b0518>
```




```
In [38]: df01 = dfJ1[['HGHT','RELH']]
         plot.figure(); df01.plot(y='HGHT'); plot.legend(loc='best')
         plot.title('Variación de la humedad relativa en función de la altura')
         plot.ylabel('Humedad %')
         plot.xlabel('Altura (m)')
         plot.grid(True)
         plot.show()
```

```
<matplotlib.figure.Figure at 0x7fd05c291160>
```




Vemos que las gráficas son muy parecidas, solo que en diciembre un leve decrecimiento de en la humedad durante un tiempo muy pequeño.

4. Por ultimo utilice la función

```
In [39]: dfD1.describe()
```

Out[39]:

| | PRES | HGHT | TEMP | DWPT | RELH | M |
|---|---|---|---|---|---|---|
| count | 71.000000 | 71.000000 | 71.000000 | 71.000000 | 71.000000 | 71.000 |
| mean | 259.257746 | 16407.352113 | -48.774648 | -66.522535 | 24.154930 | 0.949 |
| std | 310.834576 | 10738.156332 | 27.830249 | 36.288633 | 31.665054 | 2.138 |
| min | 4.000000 | 98.000000 | -82.100000 | -100.100000 | 0.000000 | 0.000 |
| 25% | 20.500000 | 7077.500000 | -70.000000 | -96.000000 | 1.000000 | 0.000 |
| 50% | 136.000000 | 14299.000000 | -60.000000 | -79.700000 | 7.000000 | 0.010 |
| 75% | 419.000000 | 25588.000000 | -30.000000 | -46.500000 | 35.000000 | 0.150 |
| max | 1026.000000 | 35518.000000 | 11.200000 | 10.300000 | 99.000000 | 7.720 |

| | DRCT | SKNT |
|---|---|---|
| count | 71.000000 | 71.000000 |
| mean | 237.846479 | 55.159155 |
| std | 148.155853 | 103.763204 |
| min | 0.000000 | 8.000000 |

```
25%      35.000000   26.000000
50%     290.000000   33.000000
75%     319.500000   49.000000
max     696.700000  696.900000
```

```
In [40]: dfJ1.describe()
```

Out[40]:

| | PRES | HGHT | TEMP | DWPT | RELH | MI |
|---|---|---|---|---|---|---|
| count | 71.000000 | 71.000000 | 71.000000 | 71.000000 | 71.000000 | 71.0000 |
| mean | 306.752113 | 14651.605634 | -31.460563 | -52.661972 | 23.126761 | 1.7373 |
| std | 327.126598 | 10264.059724 | 27.963673 | 36.079596 | 27.690086 | 3.2272 |
| min | 7.000000 | 98.000000 | -59.700000 | -89.700000 | 1.000000 | 0.0000 |
| 25% | 34.000000 | 5301.000000 | -54.650000 | -83.800000 | 1.000000 | 0.0100 |
| 50% | 159.000000 | 13647.000000 | -41.800000 | -71.900000 | 8.000000 | 0.0800 |
| 75% | 537.000000 | 23420.000000 | -6.800000 | -27.300000 | 42.500000 | 0.7650 |
| max | 1006.000000 | 34047.000000 | 21.200000 | 14.200000 | 98.000000 | 10.2800 |

| | DRCT | SKNT |
|---|---|---|
| count | 71.000000 | 71.000000 |
| mean | 199.760563 | 27.605634 |
| std | 86.857595 | 16.633425 |
| min | 0.000000 | 0.000000 |
| 25% | 113.500000 | 13.000000 |
| 50% | 250.000000 | 27.000000 |
| 75% | 265.000000 | 40.500000 |
| max | 348.000000 | 64.000000 |

# 4. Conclusión

Para terminar podemos decir que el uso de estas herramientas de análisis de datos son sumamente útiles ya que nos permiten interpretar los datos de los sondeos climáticos que también son muy importantes para poder explicar y predecir los fenómenos meteorológicos y así poder aumentar la calidad de vida.

# Referencias


[1] Anon. Atmospheric sounding. https://en.wikipedia.org/wiki/Atmospheric_sounding, 2017.

[2] Anon. Globo meteorológico. https://es.wikipedia.org/wiki/Globo_meteorol%C3%B3gico, October 2018.

[3] University of Wyoming. Sounding. http://weather.uwyo.edu/upperair/sounding.html, 2017.

# Apéndice

1. ¿Cuál es tu opinión general de esta actividad?

    - La utilización de Python me gusto un poco mas esta vez ya que lo utilizamos con mas comandos y nos dio mas herramientas para hacer mas cosas en pandas.

2. ¿Qué fue lo que más te agradó? ¿Lo que menos te agradó?

    - Lo que mas me agrado de esta actividad fue el aprender nuevos comandos de Python, lo que menos me agrado fue el que seguimos trabajando solo datos atmosféricos.

3. ¿Qué consideras que aprendiste en esta actividad?

    - Aprendí a intercambiar tipos de datos y otros comandos en pandas.

4. ¿Qué le faltó? ¿Ó le sobró?

    - Creo que necesito un poco mas de utilización de código.

5. ¿Qué mejoras sugieres a la actividad?

    - Pudiéramos encontrar otras opciones de tipo de datos para utilizar e interpretar.