Universidad de Sonora
Licenciatura en Física

Visualización gráfica de datos usando paquetes de Python

J. Ernesto Torres Burruel

28 de febrero de 2017

Física Computacional
Carlos Lizárraga Celaya

Resumen

En esta práctica hace uso de Python para obtener gráficas de sondeos atmosféricos realizados en El Paso, Texas, para ello se hace uso de paquetes de Python. El interés es observar el comportamiento de índices y parámetros atmosféricos.

Los datos se obtuvieron desde la página del Departamento de Ciencias Atmosféricas de la *Universidad de Wyoming* en `http://weather.uwyo.edu/upperair/sounding.html`.

Índice

1. Visualización de datos

El manejo de datos acerca de un objeto, fenómeno o experimento es una labor importante para lograr comprender cómo se comportan las propiedades que se estudian, una forma de lograr una mejor interpretación de los datos es mediante la realización de gráficas que comparan datos con respecto a otros para identificar una relación y establecer cómo se comporta una característica al variar alguna otra.

La visualización gráfica de datos nos muestra estructuras que nos ayudan a crear un modelo de la situación y poder generar con este predicciones de lo que sucederá cuando el fenómeno se encuentre bajo las mismas o nuevas condiciones.

1.1. Gráficas para un sondeo atmosférico

Los sondeos atmosféricos nos permiten obtener un perfil sobre las condiciones de la atmósfera que se tienen en una región al momento de la realización del Sondeo. En este caso se analizan los datos obtenidos del sondeo realizado el 15 de febrero de 2017 en la ciudad de El Paso, Texas. Nuestros parámetros de interés son la atmósfera, la temperatura y el punto de rocío y su comportamiento con respecto a la altura.

Los datos fueron descargados desde la página del departamento de Ciencias Atmosféricas de la universidad de Wyoming: `http://weather.uwyo.edu/upperair/sounding.html`, desde la cual seleccionamos que nos genere una lista del sondeo filtrado para el día 15 de febrero a las 12Z en la estación ubicada en El Paso, Tx. que es la estación 72364.

Los datos copiados los colocaremos en un archivo nuevo creado con emacs, solo nos importan los parámetros medidos, que son los datos que se encuentran en columnas, los índices no son de nuestro interés por eso no serán incluidos en el archivo. El encabezado generado por la página le añadimos # a cada renglón para que omita el texto que se encuentre sobre ese renglón al momento de leer los datos, lo mismo se usa para el renglón que incluye los nombres de las columnas.

Con el uso de los comandos de emacs se debe cambiar el archivo para obtener un archivo con formato csv (comma separated values, por sus siglas en inglés), nuestros datos están listos para trabajar con ellos. [1]

[1] Se deben identificar los datos incompletos en el archivo y aplicar el mismo formato csv par estos

Desde una terminal con la ubicación de la carpeta (directorio) en la cual se trabajará se abre `jupyter notebook`, aquí crearemos un nuevo documento en ambiente python para escribir el código necesitado. Para leer el archivo con formato csv que contiene los datos usamos el paquete *pandas* de python[2]:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("/home/user/computacional/Actividad4/datos.csv",
names=["Col.1",.., "Col.n"])
```

De esta forma a una variable `df` asignamos los datos contenidos en el archivo, como este puede contener varios datos faltantes se debe limpiar de nuevo el conjunto de datos, pero ahora desde el ambiente de jupyter, usando el lenguaje de python y sus paquetes:

`df_cl = df.dropna()`

Esta nueva variable `df_cl` tiene solo los datos completo, es decir las filas de datos que contienen valores en todas sus columnas, con esta se trabajará la creación de gráficas.[3]

1.1.1. Presión con respecto a la altura

Para crear esta gráfica, con el archivo de datos leído anteriormente trabajaremos con las columnas respectivas a la presión y la altura (PRES y HGHT, respectivamente). En este caso se emplea matplotlib para realizar las gráficas.

Trabajando con la variable a la que se le asignaron los datos sin valores faltantes se utiliza el siguiente código:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pylab as plb
from pylab import figure, show, legend, xlabel, ylabel
 . . .
y=df_cl[u'PRES']
x=df_cl[u'HGHT']
pres=plt.plot(x,y,'b.') #grafica de color azúl y con puntos
pres_l=plt.plot(x,y, 'b-') #grafica de color azul y en forma de línea
plb.title("Gráfica de altura-presión")
plb.xlabel("Altura [m]")
plb.ylabel("Presión [hPa]")
```

[2]En el Apéndice I se encuentra el código completo utilizado para crear y trabajar los datos.

[3]Debe comprobarse que los datos son leídos como números ('float64'), usando la terminación `dtypes` al escribir la variable asignada y su columna en una celda, si no, es necesario cambiarlos a formato de número.

```
plt.grid(True)
plb.show()
```

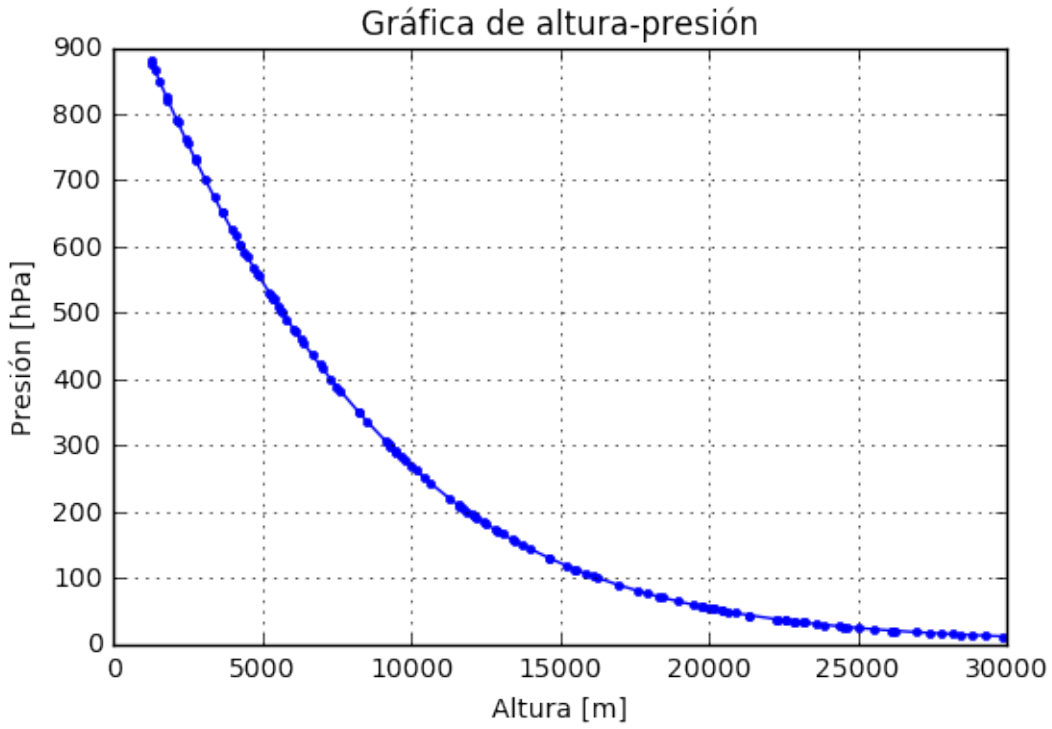

Figura 1: Gráfica generada con los datos de presión y altura

En la Figura 1 vemos la gráfica generada al tener la presión contra la altura, vemos que se comporta como una curva de tipo exponencial, conforme se aumenta la altura la presión disminuye, como es de esperarse.

1.1.2. Cambio de temperatura a distintas alturas

La temperatura es un parámetro bastante importante de la atmósfera pues en ella nos podemos basar para estimar índices atmosféricos relacionados con la energía o bien simplemente para observar cómo será la temperatura en el ambiente. en este caso nos interesa tener un perfil de la temperatura a distintas alturas, por eso es preferible tener la temperatura en el eje x y a la altura en el eje y.

Usando de nuevo los recursos de python, escribimos un código para gráficar la altura y la temperatura y observar como es el comportamiento de esta.

En el mismo contexto de antes, usando la variable que contiene la información con datos completos `df_cl` y empleando matplotlib se grafican los datos de altura (HGHT) y temperatura (TEMP:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pylab as plb
from pylab import figure, show, legend, xlabel, ylabel
...
Temp= df_cl[u'TEMP']
y= df_cl[u'HGHT']
temph=plt.plot(Temp,y, 'r.')
temph_l=plt.plot(Temp,y, 'r-')
plb.xlabel("Temperatura [C]")
plb.ylabel("Altura (m)")
plb.title("Gráfica de Temperatura - Altura")
plt.grid(True)
plb.show()
```

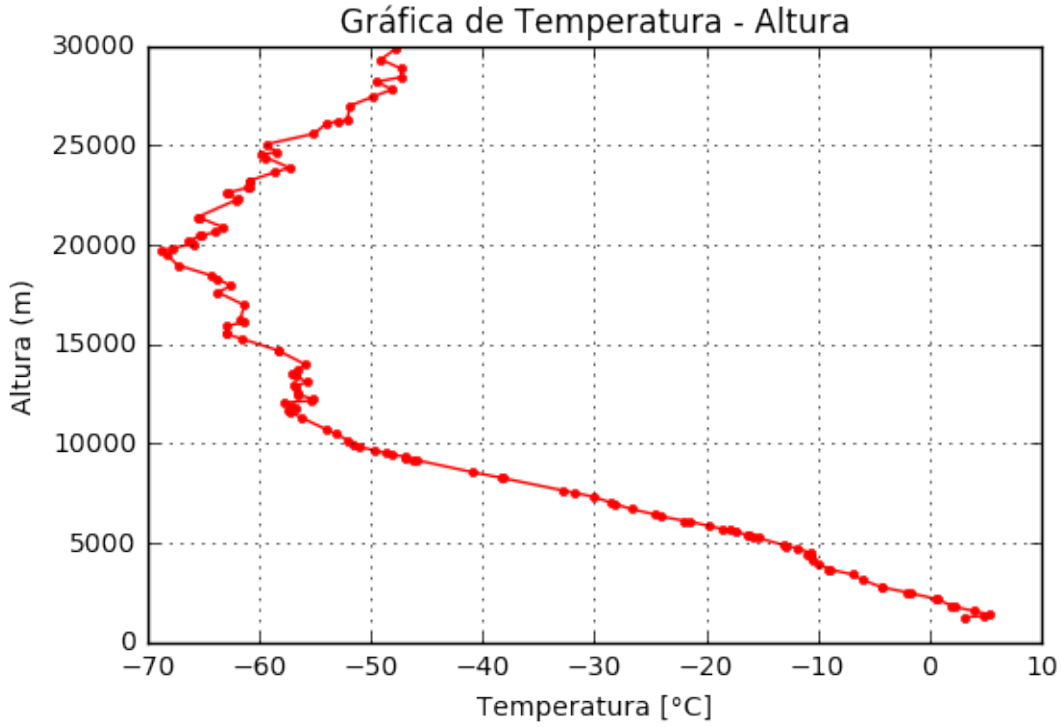

Figura 2: Gráfica obtenida por la asociación de datos de temperatura con la altura

La gráfica obtenida de la temperatura y la altura nos muestra cómo la temperatura es menor la mayor altura, lo cual se relaciona con la presión atmosférica, que pierde presión conforme se aumenta la altura. Este resultado también lo podemos asociar con la ecuación de estado del gas ideal:

$$P \cdot V = n \cdot R \cdot T \tag{1}$$

Si la temperatura disminuye también lo hace el producto de la presión con el volumen.

1.1.3. Gráfica del punto de rocío a distintas alturas

Para realizar esta gráfica debemos usar el mismo proceso que el anterior. En esta ocasión obtendremos un perfil de la temperatura en la cual el vapor de agua se condensa, conocido como punto de rocío (o dew point, del inglés), a distintas alturas para este sondeo.

Para obtener un análisis más comprensible es preferible colocar los datos referentes al punto de rocío (DWPT) 9 en el eje x y a la altura (HGHT) en el eje y.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pylab as plb
from pylab import figure, show, legend, xlabel, ylabel
...
y=df_cl[u'HGHT']
DWT=df_cl[u'DWPT']
pres=plt.plot(DWT,y, 'g.')
pres_l=plt.plot(DWT,y, 'g-')
plb.title("Gráfica de Punto de rocío-Altura")
plb.xlabel("Punto de rocío (DWPT) [C]")
plb.ylabel("Altura [m]")
plt.grid(True)
plb.show()
```

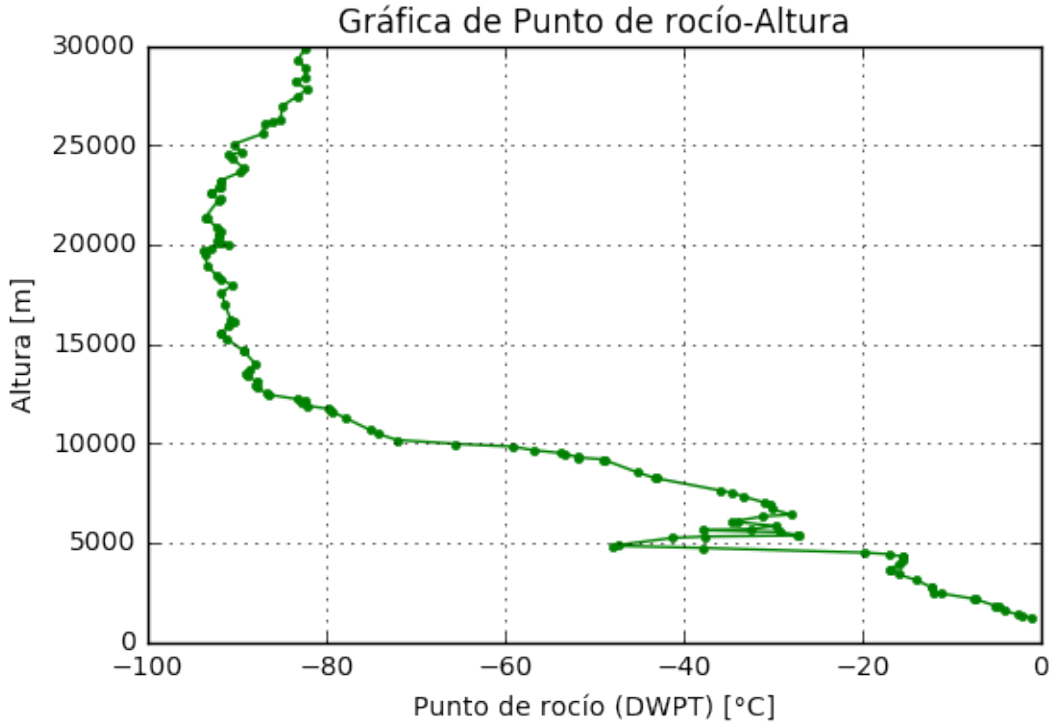

Figura 3: Gráfica generada a partir de asociar el punto de rocío a la altura en la cual se realizó la medición

De la Figura 3 vemos que a mayor altura la temperatura en la cual se condensa el agua requiere ser menor, hay algunas regiones en las que se tiene un comportamiento que se desvía. Algo que cabe destacar es que esta gráfica parece tener el mismo comportamiento que el de la temperatura y la altura, el cual compararemos más adelante.

1.1.4. Combinación de gráficas de temperatura y punto de rocío asociadas a la altura

Como vimos, el comportamiento de ambos parámetros parecen tener cierta similitud al analizarlos con respecto a la altura en la cual se realizó la medición del parámetro. Utilizando el mismo contexto que en ocasiones anteriores ahora se analizará en conjunto el punto de rocío y la temperatura asociado a la altura:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pylab as plb
from pylab import figure, show, legend, xlabel, ylabel
...
y= df_cl[u'HGHT']
DWT=df_cl[u'DWPT']
Temp= df_cl[u'TEMP']
temps=plt.plot(Temp, y, 'r-', DWT, y, 'b--')
plb.ylabel("Altura [m]")
plb.xlabel("Tamperatura (C)")

plt.annotate('Temperatura', xy=(-20, 6000))
plt.annotate('Punto de rocío', xy=(-80, 8000))
plb.show()
```

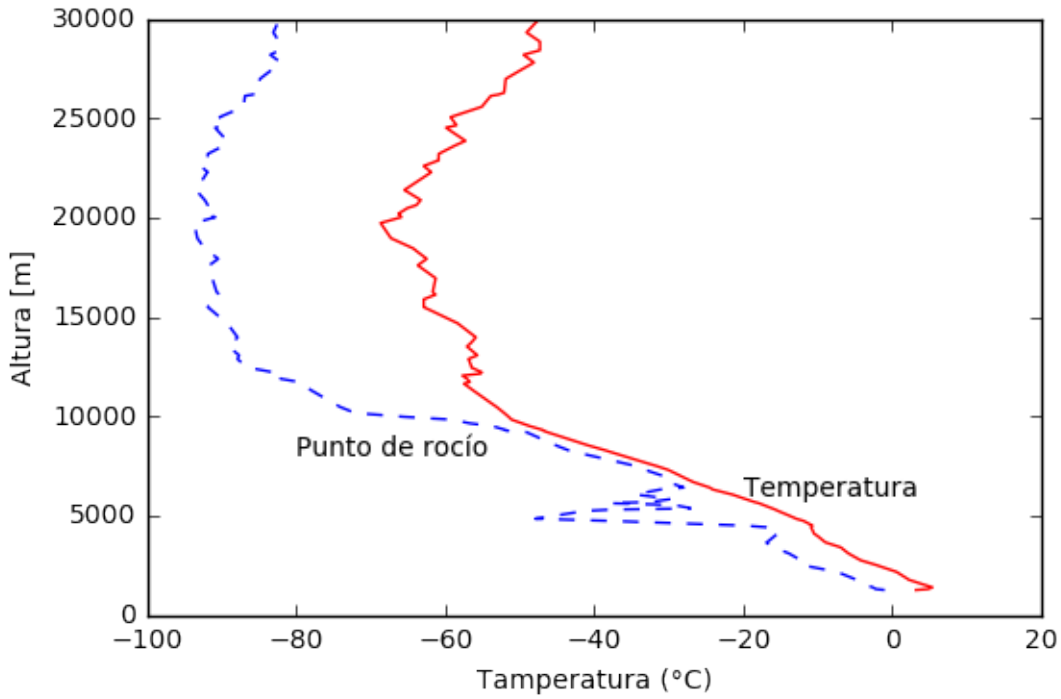

Figura 4: Gráfica comparativa entre el comportamiento de la temperatura en la atmósfera y el punto de rocío

En esta gráfica vemos que la temperatura en la cual se condensa el agua disminuye de la misma forma en que disminuye la temperatura atmosférica, aunque estas se separan mucho después de cierta altura. Podría esperarse que el punto de rocío supere en algún punto a la altura, pero lo que sucede es que la la presión también disminuye y la densidad del aire también cambia en la altura, por lo que es más difícil obtener las condiciones necesarias para condensar el vapor de agua que se tiene en la región.

2. Tefigrama

El tefigrama es un diagrama termodinámico utilizado para trazar perfiles verticales de temperatura, humedad y viento atmosféricos. El tefigrama se ha usado durante mucho tiempo para analizar la estabilidad atmosférica.

En el tefigrama se tienen distintas líneas importantes como las líneas isobáricas, isotermas, adiabáticas secas (dry adiabats) y saturadas (saturation adiabats), que se refieren a líneas de presión, temperatura y calor constantes, respectivamente; también son incluidas las líneas de razón de mezcla de saturación. Con estas líneas se forma un tefigrama, que es parecido a un diagrama oblicuo t - log p, con la diferencia que este presenta isobaras convexas.

2.1. Instalación de paquetes para trazar tefigramas

Para trazar un tefigrama con los datos obtenidos del sondeo haremos uso de el paquete para python distribuido como software libre llamado `tephi` que adquirimos mediante github, en donde se encuentra el paquete.

Pasos a seguir:

1. Primero desde tu repositorio en Github, realiza un Fork del repositorio tephi.
2. Crea la carpeta Actividad4 en tu computadora para realizar esta actividad.
3. Entra a la carpeta desde una terminal, y clona el repositorio de tephi en Github.com, utilizando el comando: git clone https://github.com/username/tephi.git
4. Se requerirá instalar la biblioteca `tephi` en tu entorno de programación, utilizando el comando `pip`.

Para realizar la instalación local del paquete, desde la terminal realizaremos lo siguiente:

```
pip install --user /home/user/Computacional/Actividad4/tephi
```

Con esto habremos instalado el paquete `tephi` como paquete de python y estará listo para emplearse y elaborar tefigramas con nuestros datos.

2.2. Creación del tefigrama

Con la ayuda de emacs debemos construír dos nuevos archivos csv que contengan dos columnas de datos: el primero con presión (PRES) y punto de rocío (DWPT), el segundo con presión (PRES) y temperatura (TEMP), que llamaremos dew.csv y temp.csv.

Ahora, desde nuestro documento de jupyter notebook trabajando en el lenguaje de python ocuparemos además de `tephi` los paquetes de `pandas` y `matplotlib`.[4]

```
import pandas as pd
import matplotlib.pyplot as plt
import tephi as tph
```

[4]Para ver el código completo utilizado para la creación de los gráficos y diagramas vea el Apéndice I

Con el paquete de pandas leemos los archivos y los asignamos a las variables que los van a almacenar:

```
dew_point = pd.read_csv("/home/user/Computacional/Actividad4/Datos/dew.csv,
names=["PRES","DWPT"])

dry_bulb = pd.read_csv("("/home/user/Computacional/Actividad4/Datos/temp.csv",
names=["PRES", "TEMP"])
```

Ahora tenemos los pares de datos que necesita `tephi` para realizar los gráficos. Solo falta utilizar este paquete para realizar el tefigrama.

```
tpg = tph.Tephigram()
tpg.plot(dew_point)
tpg.plot(dry_bulb)

plt.show()
```

Para tener un mejor resultado al realizar el tefigrama se realizaron personalizaciones en el comando para crear el tefigrama, como se ve en el código:

```
tpg = tph.Tephigram()
tpg.plot(dew_point, label='Punto de rocío', color='blue', linewidth=1, linestyle='--')
tpg.plot(dry_bulb, label='Temperatura', color='red', linewidth=1, linestyle='-')

plt.show()
```

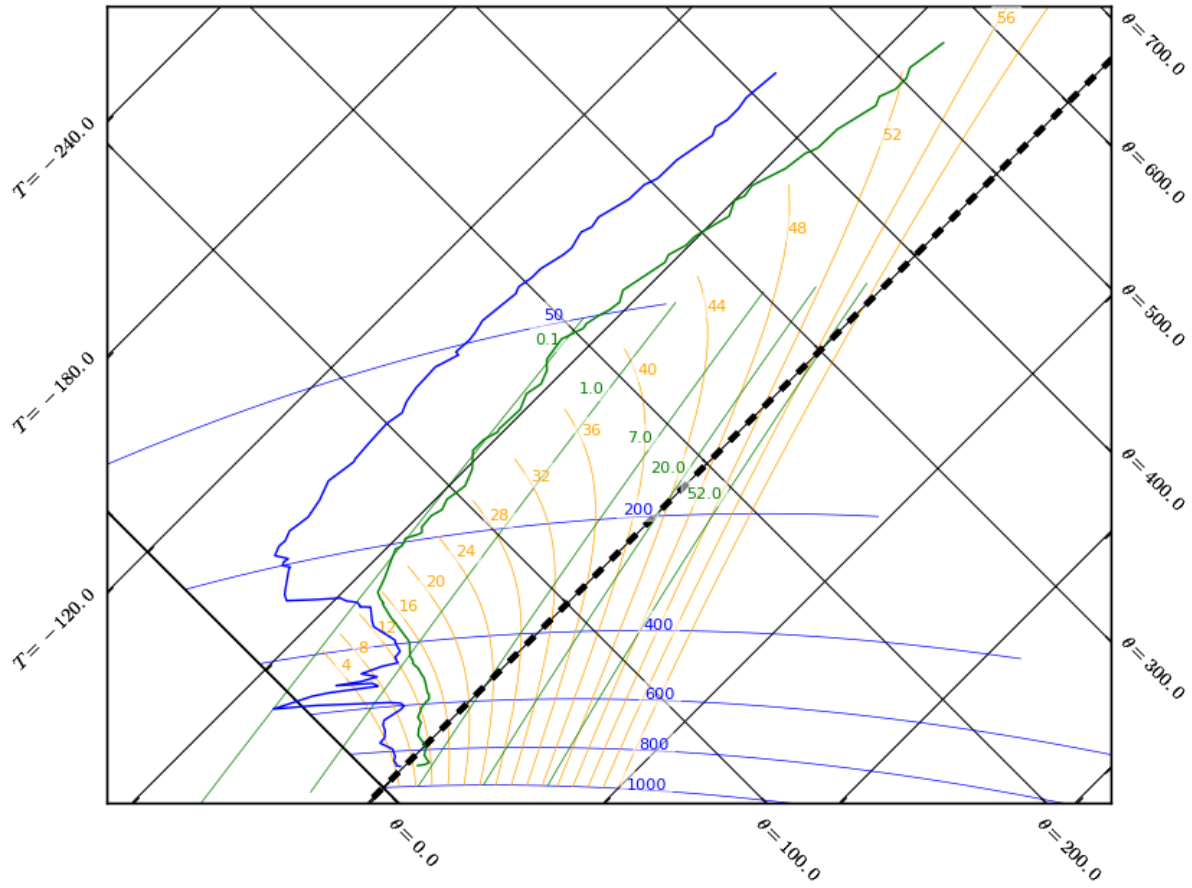

Figura 5: Tefigrama sin personalizar, creado con los datos del sondeo realizado en El Paso, Tx. el 15 de febrero a las 12Z

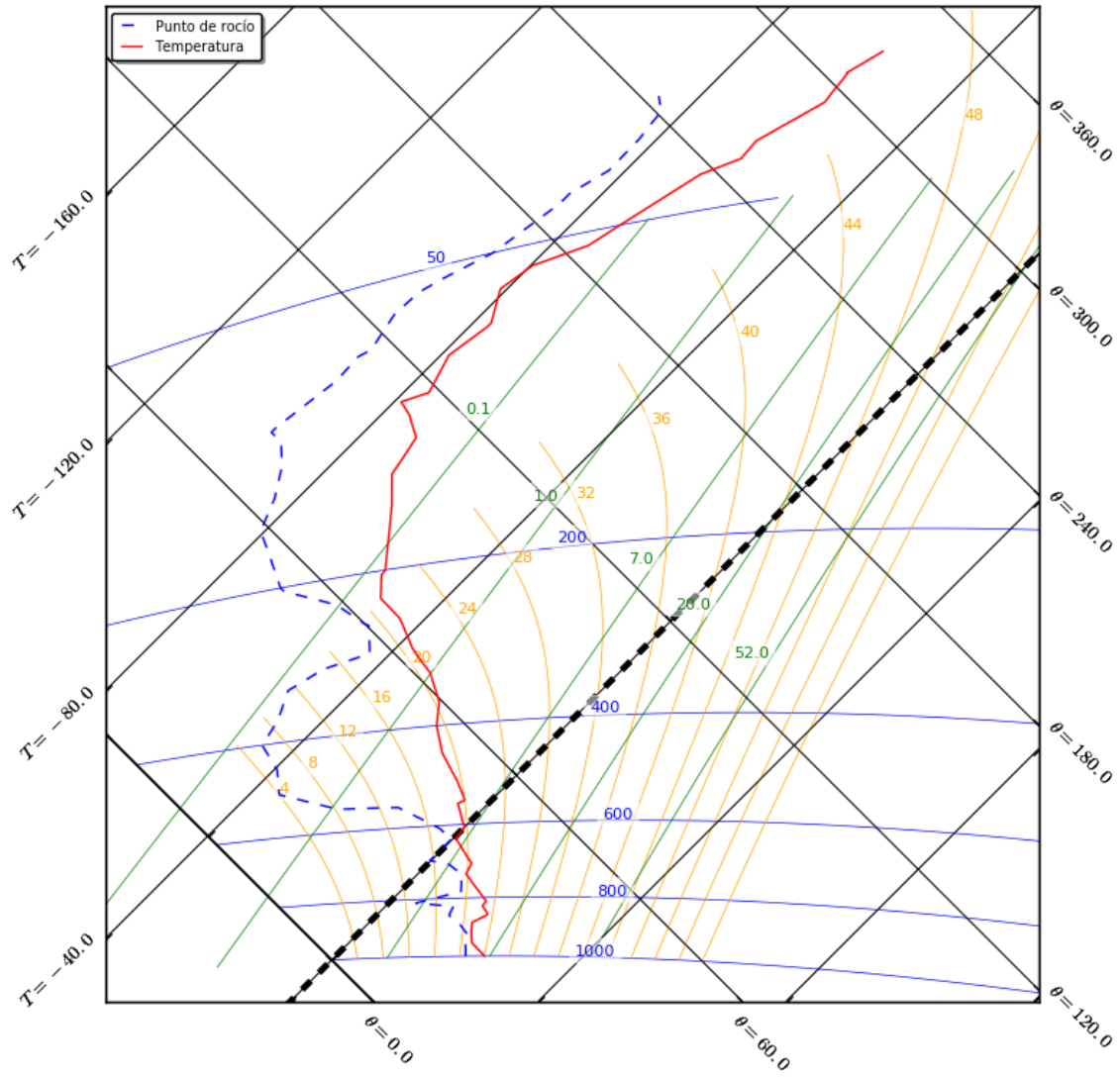

Figura 6: Tefigrama personalizado y con identificadores para los datos de temperatura y punto de rocío; creado con los datos del sondeo realizado en El Paso, Tx. el 15 de febrero a las 12Z

Con los tefigramas creados podemos tener una perspectiva de las condiciones relacionadas con los fenómenos térmicos que suceden en la atmósfera, así podemos estimar cómo fue el estado de la atmósfera en el momento que se realizó el sondeo.

3. Conclusiones

El uso de gráficos nos permite obtener una idea más clara sobre el comportamiento de los datos obtenidos con respecto a otro parámetro que conocemos y ver cuál es la relación entre ellos.

Graficar datos por medio de recursos electrónicos facilita mucho el trabajo de los investigadores que buscan analizar datos de forma más eficiente y rápida, sin tener que invertir gran cantidad de tiempo en procesar y graficar una gran cantidad de datos.

El paquete `tephi` es una herramienta que aún está en elaboración, y limita bastante al momento de realizar los diagramas termodinámicos. Existen otros paquetes que logran procesar mejor la información y generan un resultado más completo y sin complicaciones.

4. Referencias

[1] The COMET© Program. *Tephigram Mastery*. Página web. University Corporation for Atmospheric Research. 2013. Colsultado el 27 de febrero de 2017. `http://www.meted.ucar.edu/mesoprim/tephigram/index.htm`

Apéndice I: código utilizado para realizar gráficas con python

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pylab as plb
from pylab import figure, show, legend, xlabel, ylabel

df = pd.read_csv("/home/ernestotb/Computacional/Actividad4/Datos/15Feb17.csv"
, names=["PRES","HGHT","TEMP", "DWPT", "RELH", "MIXR", "DRCT", "SKNT"
, "THTA", "THTE", "THTV"])

df.PRES= pd.to_numeric(df.name, errors='coerce') #transformar datos de columnas
a números.

df_cl=df.dropna()
df.columns
#gráfica de presión contra altura

y=df_cl[u'PRES']
x=df_cl[u'HGHT']
pres=plt.plot(x,y,'b.')
pres_l=plt.plot(x,y, 'b-')
plb.title("Gráfica de altura-presión")
plb.xlabel("Altura [m]")
plb.ylabel("Presión [hPa]")
plt.grid(True)
plb.show()

#Gráfica de teperatura y altura
Temp= df_cl[u'TEMP']
y= df_cl[u'HGHT']
temph=plt.plot(Temp,y, 'r.')
temph_l=plt.plot(Temp,y, 'r-')
plb.xlabel("Temperatura [C]")
plb.ylabel("Altura (m)")
plb.title("Gráfica de Temperatura - Altura")
plt.grid(True)
plb.show()

#gráfica de punto de rocío y altura
y=df_cl[u'HGHT']
DWT=df_cl[u'DWPT']
pres=plt.plot(DWT,y, 'g.')
```

```
pres_l=plt.plot(DWT,y, 'g-')
plb.title("Gráfica de Punto de rocío-Altura")
plb.xlabel("Punto de rocío (DWPT) [C]")
plb.ylabel("Altura [m]")
plt.grid(True)
plb.show()

#combinación de gráficas de temperatura y punto de rocío con la altura
y= df_cl[u'HGHT']
DWT=df_cl[u'DWPT']
Temp= df_cl[u'TEMP']
temps=plt.plot(Temp, y, 'r-', DWT, y, 'b--')
plb.ylabel("Altura [m]")
plb.xlabel("Tamperatura (C)")

plt.annotate('Temperatura', xy=(-20, 6000))
plt.annotate('Punto de rocío', xy=(-80, 8000))
plb.show()

#creación de tefigrama
import os.path
import tephi as tph

dew_point = pd.read_csv("/home/ernestotb/Computacional/Actividad4/Datos/dew.csv",
names=["PRES", "DWPT"])

dry_bulb = pd.read_csv("/home/ernestotb/Computacional/Actividad4/Datos/temp.csv",
names=["PRES", "TEMP"])

tpg = tph.Tephigram()
tpg.plot(dew_point)
tpg.plot(dry_bulb)

plt.show()

#tefigrama estilizado
tpg = tph.Tephigram()
tpg.plot(dew_point, label='Punto de rocío', color='blue', linewidth=1,
linestyle='--')

tpg.plot(dry_bulb, label='Temperatura', color='red', linewidth=1,
linestyle='-')

plt.show()
```