APERTURA DE LOS DATOS DE RADAR DE IDEAM

Lo primero será ubicar el archivo descargado, para ello definiremos una ruta con la ubicación del mismo.

In [1]:
```
ruta = r'DATA/*'
```

Utilizaremos la librería "glob" para listar todos los archivos que se encuentren en la ruta anteriormente definida.

In [2]:
```
import glob
arch = glob.glob(ruta)
```

Procederemos a revisar el listado de los archivos en la ruta de los datos, para poder elegir aquel con el que deseemos trabajar.

In [3]:
```
list(arch)
```

Out[3]:
```
['DATA\\BAR190923000004.RAW935L',
 'DATA\\BAR200511203650.RAW82L4',
 'DATA\\BAR200511203740.RAW82LC',
 'DATA\\BAR200511203905.RAW82LJ',
 'DATA\\BAR200511204005.RAW82LK']
```

Py-ART es una librería que contiene diversos algoritmos y utilidades relacionadas al despliegue y visualización de archivos de radar (más información aquí). La función "pyart.io.read" nos permitirá abrir y editar archivos de radar con extensión ".RAW*".

Para seleccionar el archivo de la lista desplegada anteriormente debemos indicarle a python su posición, en este caso utilizaremos el primer archivo. En Python, la posición de los elementos empieza a contar desde 0.

Con el uso de esta función aparecerán algunas advertencias relacionadas a la lectura del archivo, que no necesariamente representan un error. En este ejemplo ingnoraremos dichas advertencias.

In [4]:
```
import pyart
radar = pyart.io.read(arch[0])
```

```
## You are using the Python ARM Radar Toolkit (Py-ART), an open source
## library for working with weather radar data. Py-ART is partly
## supported by the U.S. Department of Energy as part of the Atmospheric
## Radiation Measurement (ARM) Climate Research Facility, an Office of
## Science user facility.
##
## If you use this software to prepare a publication, please cite:
##
##     JJ Helmus and SM Collis, JORS 2016, doi: 10.5334/jors.119
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pyart\graph\cm.py:104: FutureWarning: elementwise compa
rison failed; returning scalar instead, but in the future will perform elementwise comparison
  if 'red' in spec:
C:\ProgramData\Anaconda3\lib\site-packages\pyart\graph\cm_colorblind.py:32: FutureWarning: elementw
ise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  if 'red' in spec:
C:\ProgramData\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the sec
ond argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.flo
at64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
C:\ProgramData\Anaconda3\lib\site-packages\pyart\io\sigmet.py:147: RuntimeWarning: invalid value enco
untered in sqrt
  sigmet_data, sigmet_metadata = sigmetfile.read_data(full_xhdr=full_xhdr)
```

La información contenida en el archivo de radar es entendida por Python como un objeto. Un objeto es una agrupación de variables y funciones que interactúan con dichas variables.

Para consultar los metadatos del radar, se debe consultar la variable "metadata" del archivo de radar. El parámetro 'compact' abreviara la información contenida en los metadatos del objeto.

In [6]:
```
radar.info('compact')
```

```
altitude: <ndarray of type: float64 and shape: (1,)>
```

altitude_agl: None
antenna_transition: None
azimuth: <ndarray of type: float32 and shape: (360,)>
elevation: <ndarray of type: float32 and shape: (360,)>
fields:
    reflectivity: <ndarray of type: float32 and shape: (360, 750)>
    velocity: <ndarray of type: float32 and shape: (360, 750)>
    spectrum_width: <ndarray of type: float32 and shape: (360, 750)>
    differential_reflectivity: <ndarray of type: float32 and shape: (360, 750)>
    specific_differential_phase: <ndarray of type: float32 and shape: (360, 750)>
    differential_phase: <ndarray of type: float32 and shape: (360, 750)>
    normalized_coherent_power: <ndarray of type: float32 and shape: (360, 750)>
    cross_correlation_ratio: <ndarray of type: float32 and shape: (360, 750)>
    radar_echo_classification: <ndarray of type: float32 and shape: (360, 750)>
fixed_angle: <ndarray of type: float32 and shape: (1,)>
instrument_parameters:
    unambiguous_range: <ndarray of type: float32 and shape: (360,)>
    prt_mode: <ndarray of type: |S5 and shape: (1,)>
    prt: <ndarray of type: float32 and shape: (360,)>
    prt_ratio: <ndarray of type: float32 and shape: (360,)>
    nyquist_velocity: <ndarray of type: float32 and shape: (360,)>
    radar_beam_width_h: <ndarray of type: float32 and shape: (1,)>
    radar_beam_width_v: <ndarray of type: float32 and shape: (1,)>
    pulse_width: <ndarray of type: float32 and shape: (360,)>
latitude: <ndarray of type: float64 and shape: (1,)>
longitude: <ndarray of type: float64 and shape: (1,)>
nsweeps: 1
ngates: 750
nrays: 360
radar_calibration: None
range: <ndarray of type: float32 and shape: (750,)>
scan_rate: None
scan_type: ppi
sweep_end_ray_index: <ndarray of type: int32 and shape: (1,)>
sweep_mode: <ndarray of type: |S20 and shape: (1,)>
sweep_number: <ndarray of type: int32 and shape: (1,)>
sweep_start_ray_index: <ndarray of type: int32 and shape: (1,)>
target_scan_rate: None
time: <ndarray of type: float64 and shape: (360,)>

metadata:
    Conventions: CF/Radial instrument_parameters
    version: 1.3
    title:
    institution:
    references:
    source:
    history:
    comment:
    instrument_name: b'barrancabermeja'
    original_container: sigmet
    sigmet_task_name: b'SURVEILLANCE'
    sigmet_extended_header: false
    time_ordered: none
    rays_missing: 0

Lo siguiente será consultar la fecha en la cual fue tomada el archivo.

In [7]:
```
radar.time['units']
```

Out[7]: 'seconds since 2019-09-23T00:00:04Z'

Adicionalmente podemos consultar además, el nombre del radar que capturo la información.

In [8]:
```
radar.metadata['instrument_name']
```

Out[8]: b'barrancabermeja'

Ahora consultaremos las variables polarimétricas disponibles en el volumen escaneado por el radar.

In [9]:
```
list(radar.fields)
```

Out[9]:
['reflectivity',
 'velocity',
 'spectrum_width',
 'differential_reflectivity',

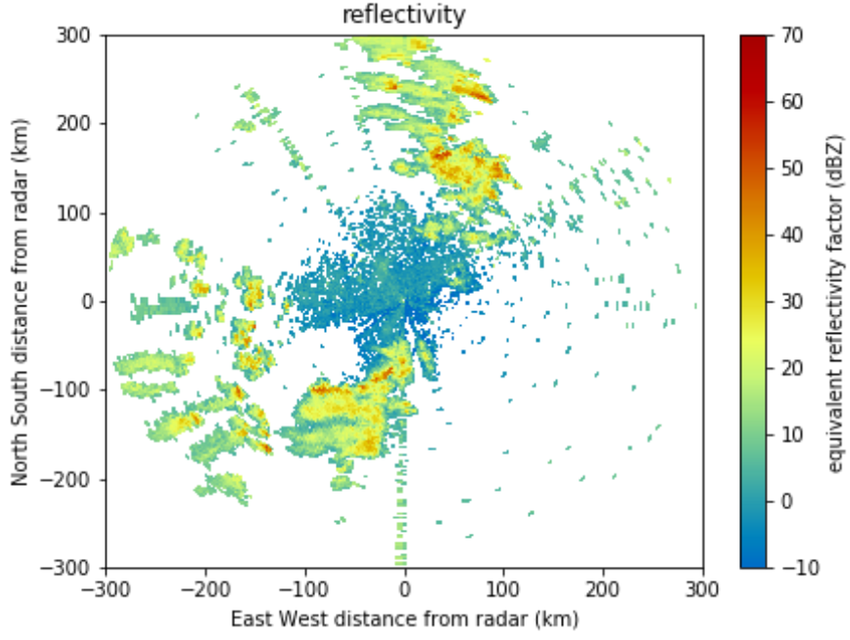
```
'differential_phase',
'normalized_coherent_power',
'cross_correlation_ratio',
'radar_echo_classification']
```

Finalmente, haremos un plot de la información contenida en el archivo de radar con ayuda de la librería matplotlib. Para ello crearemos una función que facilitará la consulta de la variable polarimétrica deseada.

In [10]:
```
import matplotlib.pyplot as plt
def graf(data, var, vmin, vmax):
    display = pyart.graph.RadarMapDisplay(data)
    fig = plt.figure(figsize = (15, 23))
    fig.add_subplot(int(str(42) + str(1)))
    display.plot_ppi(var, 0, vmin = vmin, vmax = vmax, title = var)
```

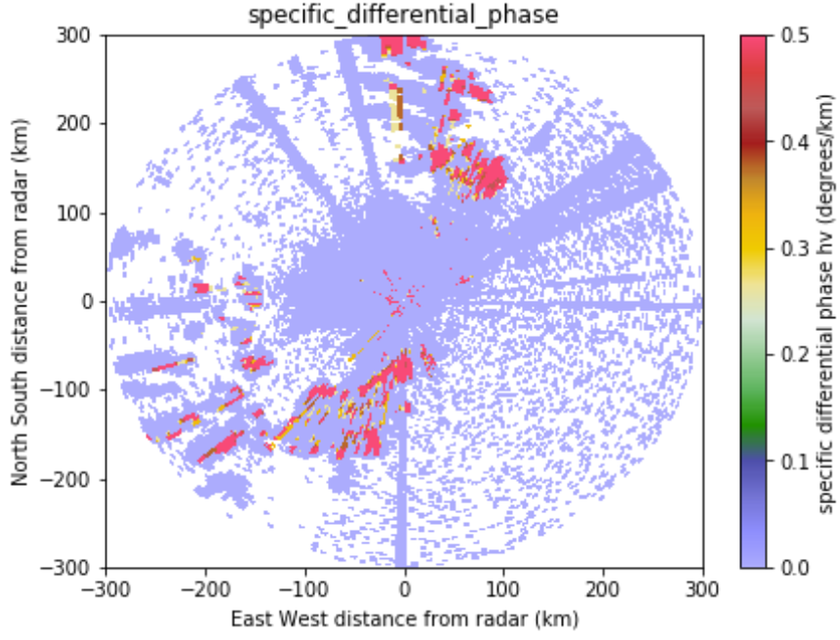
Usaremos la función "graf" para ver los datos de reflectividad.

In [11]:
```
graf(radar, 'reflectivity', -10, 70)
```



Ahora visualizaremos los datos del diferencial especifico de fase (KDP).

In [12]:
```
graf(radar, 'specific_differential_phase', 0, 0.5)
```



Si quisiéramos tener un plot con más de una variable polarimétrica debemos hacer una nueva función que nos permita agrupar varias imágenes un una sola.

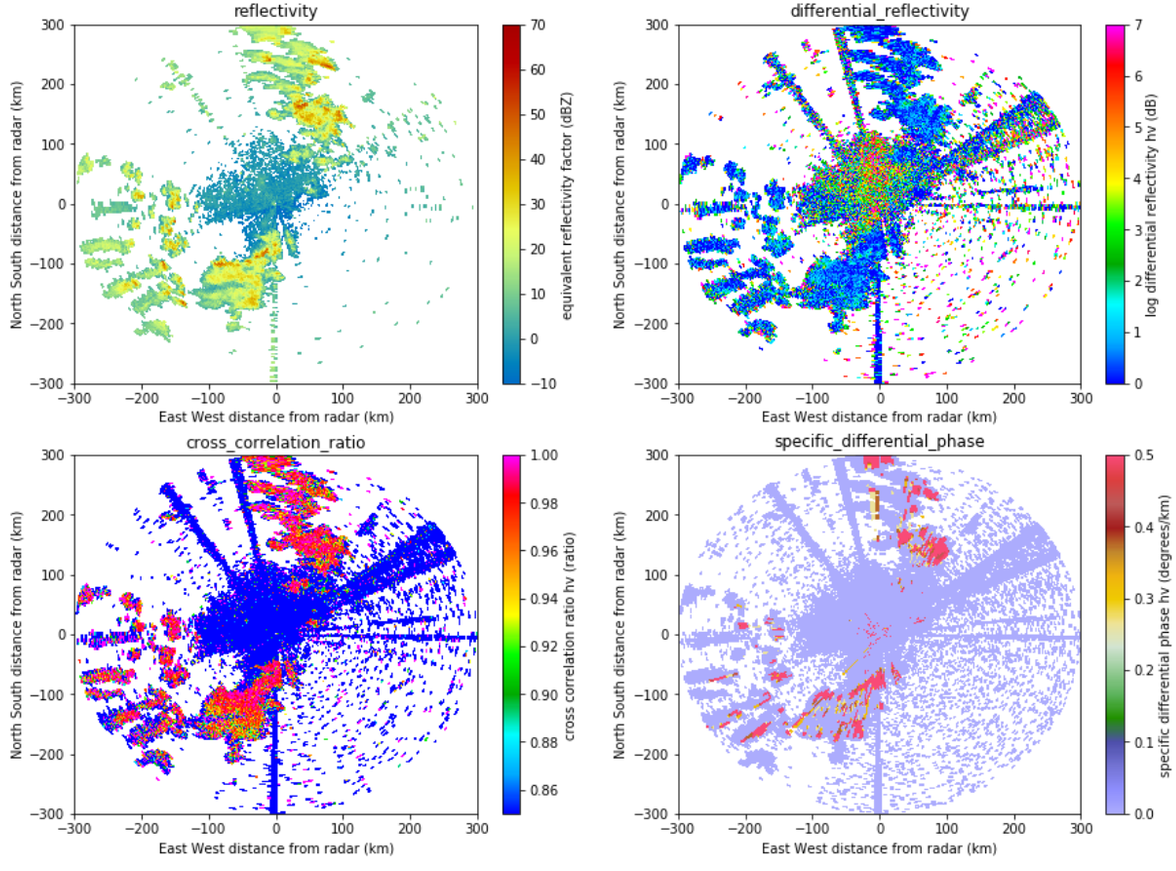
In [15]:
```
def graf_m(data , x , lims):

    display = pyart.graph.RadarMapDisplay(data)
    fig = plt.figure(figsize = (15, 23))

    for i in range(0, len(x)):
        fig.add_subplot(int(str(42) + str(i+1)))
        display.plot_ppi(x[i], 0, vmin = lims[i][0], vmax = lims[i][1], title = x[i] )
```

La nueva función "graf_m" nos permitirá realizar un plot múltiple; sin embargo, el ingreso de las valores máximos y mínimos de cada variable polarimétrica debe ser a través de una matriz 2 x n, para esto utilizaremos la librería "numpy", la cual nos permitirá crear una matriz a partir de los vectores "vmin" y "vmax"

In [17]:
```
import numpy as np
vmin = [-10, 0, 0.85, 0.]
vmax = [70., 7., 1., 0.5]
lims = np.column_stack((vmin, vmax))
x = ['reflectivity', 'differential_reflectivity', 'cross_correlation_ratio', 'specific_differential_phase']
graf_m(radar, x, lims)
```



IDEAM
Bogotá, D.C. Colombia - Sur América
Sede correspondencia
Calle 25 D No. 96 B - 70 Bogotá D.C. Codigo postal: 110911
PBX (571) 3527160 Fax Server: 3075621 - 3527160 Opc.2
Línea Nacional 018000110012 - Pronóstico y Alertas (571) 3527180
Sede Puente Aranda: Calle 12 No 42B – 44 Bogotá D.C. PBX. 2681070
- www.ideam.gov.co