pycwr

Release 0.3.4

Yu Zheng

Mar 01, 2023

开始使用

Release

V0.3.4

Date

Mar 01, 2023

PyCWR （Python based China Weather Radar Toolkit）主要面向中国天气雷达基数据的科研和应用，广泛支持中国业务天气雷达基数据（WSR98D、CINRAD/SA/SB/CB、CINRAD/CC/CCJ 和 CINRAD/SC/CD 等），方便用于科研和业务上数据处理和分析，PyCWR 现阶段主要由 NJIAS 团队进行维护和更新。

CHAPTER

ONE

使用文档

开始使用

- PyCWR 简介
- 安装方法
- 图形化界面显示
- 常见问题

1.1 PyCWR 简介

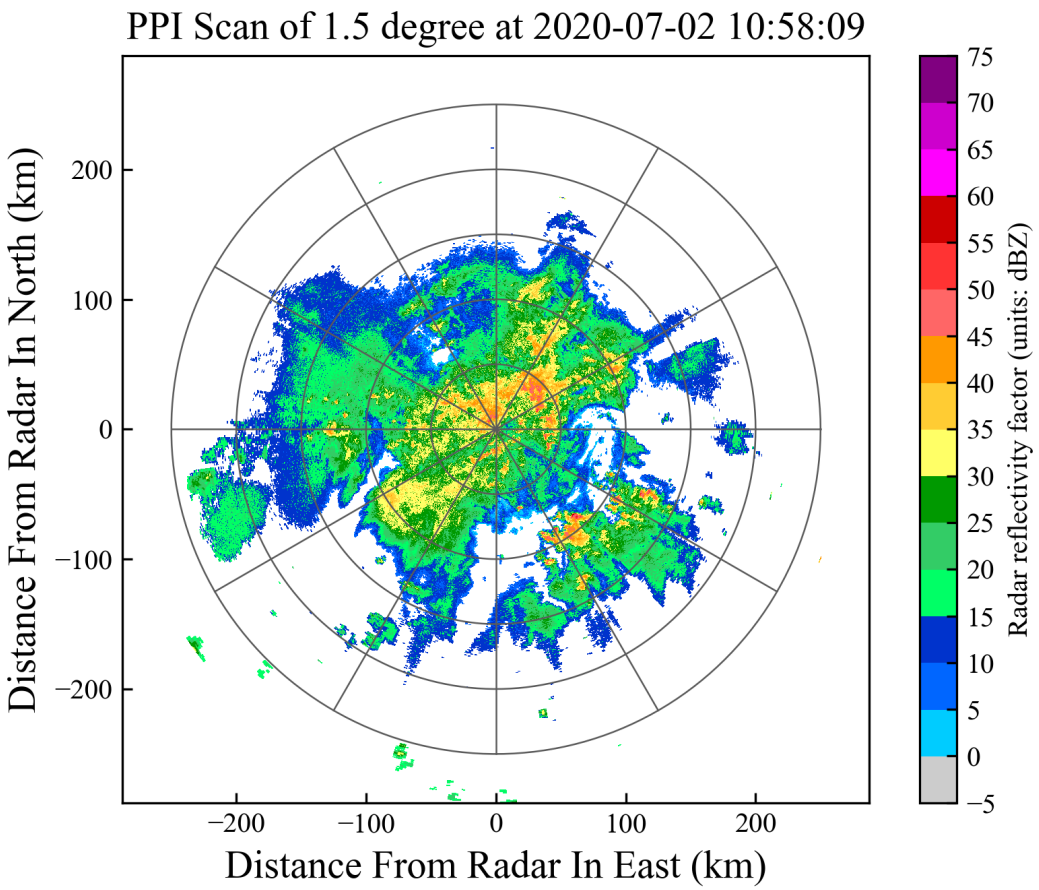
中国天气雷达基数据的格式由于历史遗留问题多种多样，文件复杂，雷达数据的处理和应用往往需要重要的先验知识和技能，各种算法难以统一实施。基于 Python 的中国天气雷达（PyCWR）库旨在解决这些问题并使雷达数据变得易于使用。

Important:

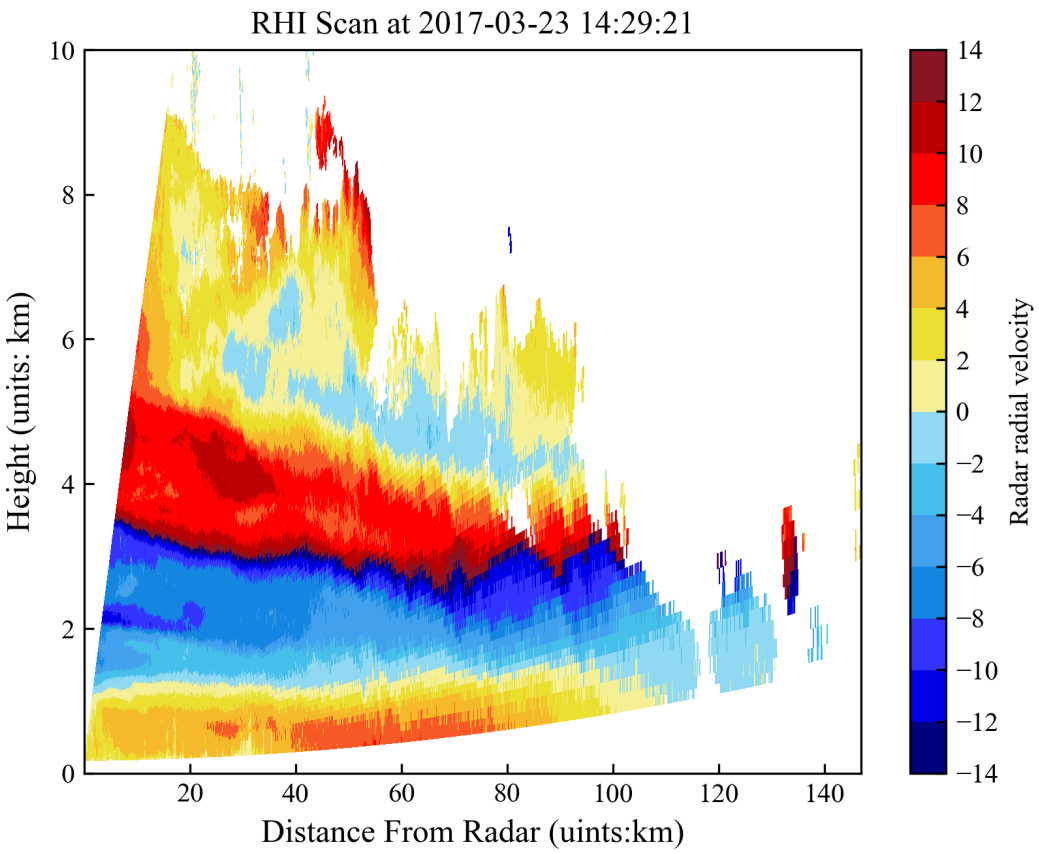
- 它支持几乎所有中国业务天气雷达基数据格式，包含 WSR98D、CINRAD/SA/SB/CB、CINRAD/CC/CCJ、CINRAD/SC/CD 和相控阵等雷达基数据格式；
- 定义用于表示雷达数据的核心 Python 对象，而该对象是使用 xarray 的 Dataset 构建的，Dataset 是类似 dict 的具有对齐尺寸的带标签数组的容器；
- 提供了经过良好测试的算法（数据质量控制，水凝物分类，定量降水估计等），并使用 Cython 加速核心算法；
- 建立开放的平台，供研究人员从天气雷达观测中开发复杂的分析和应用程序，它还提供了图形用户界面工具，以方便进行水文学和气象学分析。

---

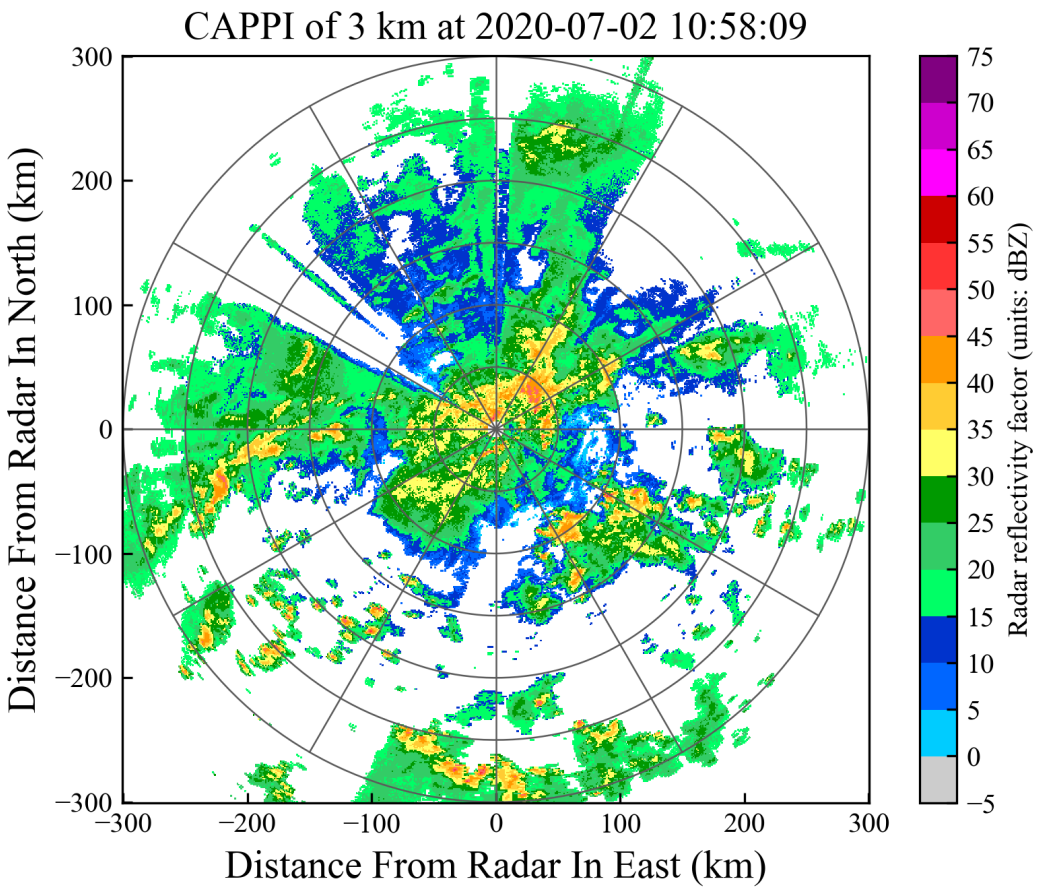
1. 天气雷达 PPI 扫描显示：



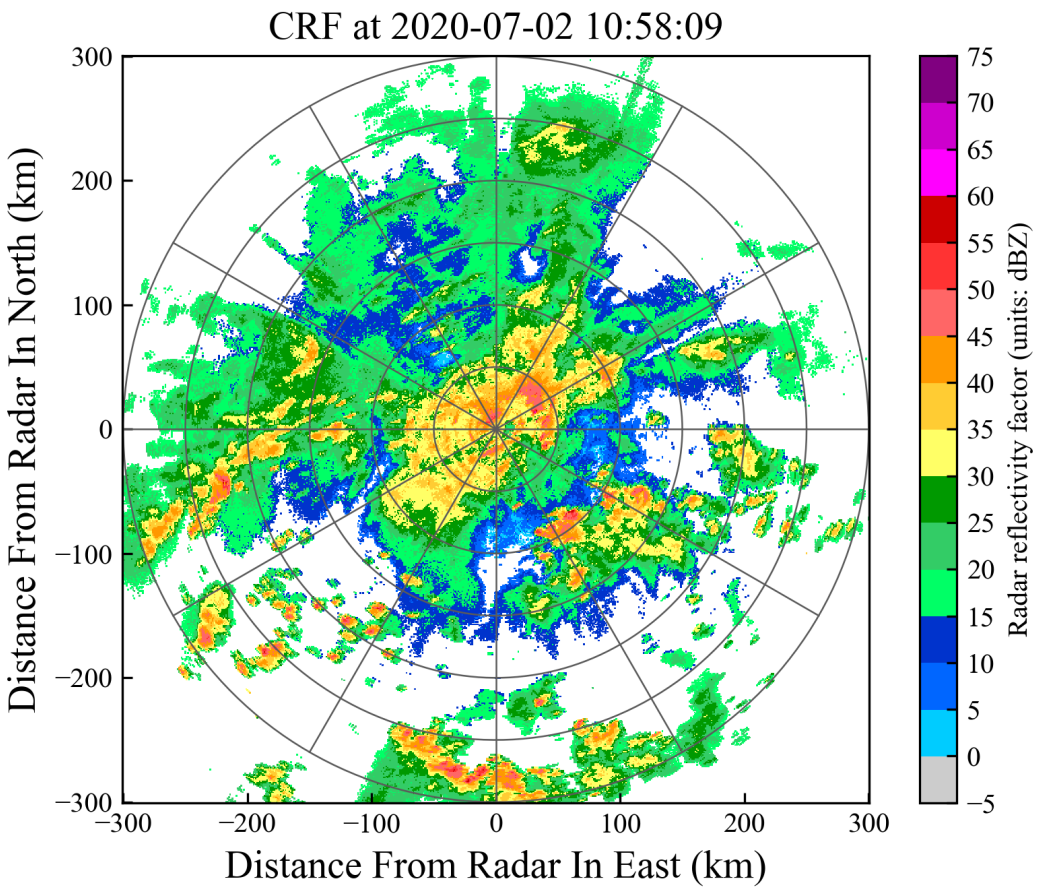
2. 天气雷达 RHI 扫描显示：



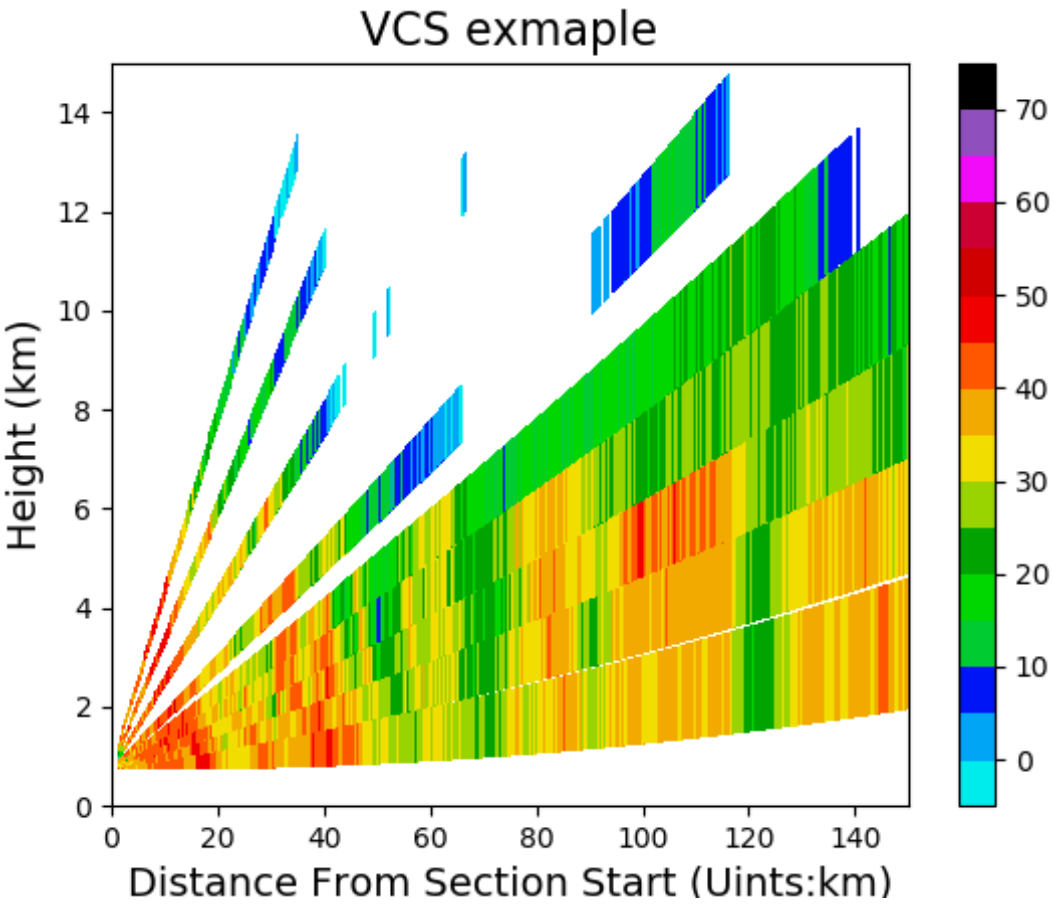
3. 天气雷达 CAPPI 插值显示：



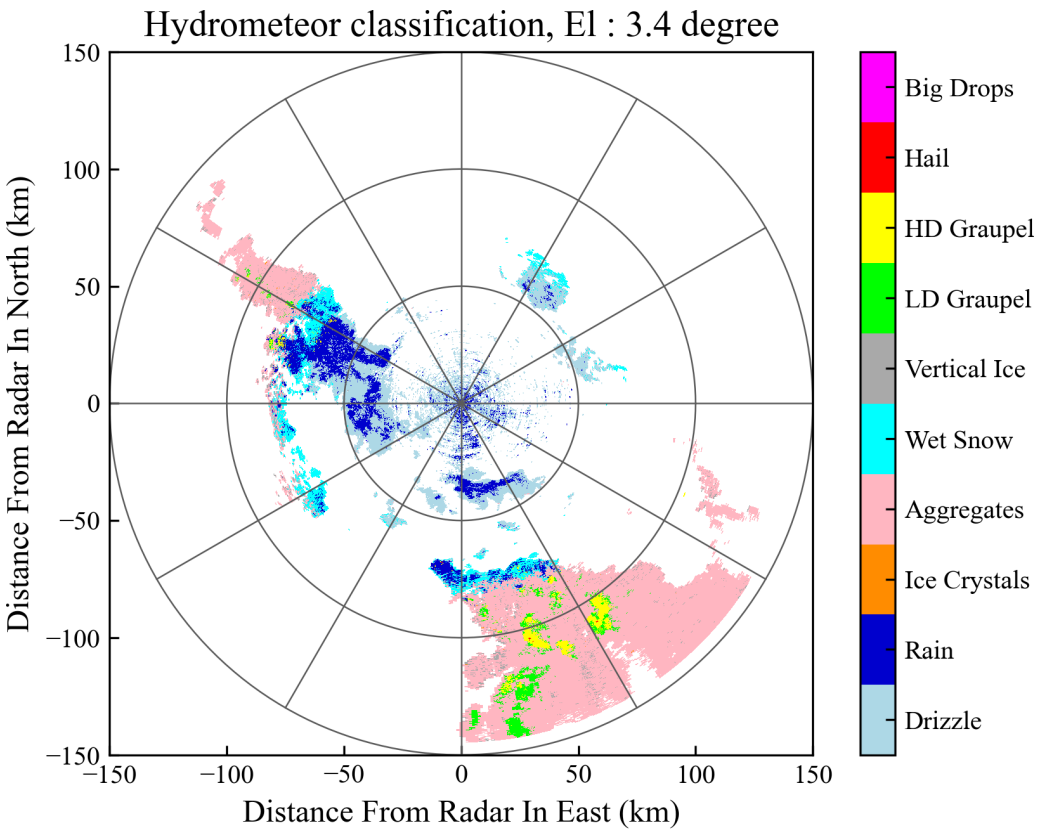
4. 天气雷达组合反射率因子显示：



5. 天气雷达 VCS 垂直剖面图：


VCS exmaple
Height (km)
0
2
4
6
8
10
12
14
0
20
40
60
80
100
120
140
Distance From Section Start (Uints:km)
70
60
50
40
30
20
10
0

6. 水凝物分类算法：



7. 更多功能更新中…

1.2 安装方法

1.2.1 Anaconda

目前 *PyCWR* 仅支持 Python3，推荐使用 Python3.8 及以上版本的 Anaconda。

由于国内墙下载速度的限制，推荐在 清华镜像站 下载并安装 Anaconda，提高下载速度。

Anaconda 安装完毕后请按照 conda 源修改教程 修改 conda 源为清华源，按照以下教程修改 pip 源为豆瓣源以提高库的安装速度，节省您的时间。

1. Linux/Mac 平台

   Linux/Mac 用户将它命名为 pip.conf

```
[global]
timeout = 60
index-url = http://pypi.douban.com/simple
trusted-host = pypi.douban.com
```

然后将该文件放在 `$HOME/.pip/pip.conf` 位置

2. Windows 平台

   Windows 用户将它命名为 pip.ini

```
[global]
timeout = 60
index-url = http://pypi.douban.com/simple
trusted-host = pypi.douban.com
```

   然后将该文件放在 `%HOME%\pip\pip.ini` 位置

1.2.2 直接安装 *PyCWR* 在 base 环境

由于 Cartopy 采用 pip 安装容易出错，但 *PyCWR* 画图部分需要依赖 Cartopy，因此需要先采用 conda 来安装 Cartopy，再使用 pip 安装 *PyCWR*。

1. 首先使用 conda 安装 Cartopy

```
conda install cartopy -c conda-forge --yes
```

2. 然后使用 pip 安装 PyCWR

```
pip install pycwr==0.3.6
```

1.2.3 隔离环境安装 *PyCWR* （推荐）

为防止 *PyCWR* 环境与 base 环境冲突，推荐使用隔离环境的方式安装。

1. 首先使用 conda 隔离环境并安装 Cartopy

```
conda create -n cwr cartopy -c conda-forge --yes
```

2. 再切换到隔离的 cwr 环境

```
conda activate cwr
```

3. 最后在 cwr 环境中使用 pip 安装 PyCWR

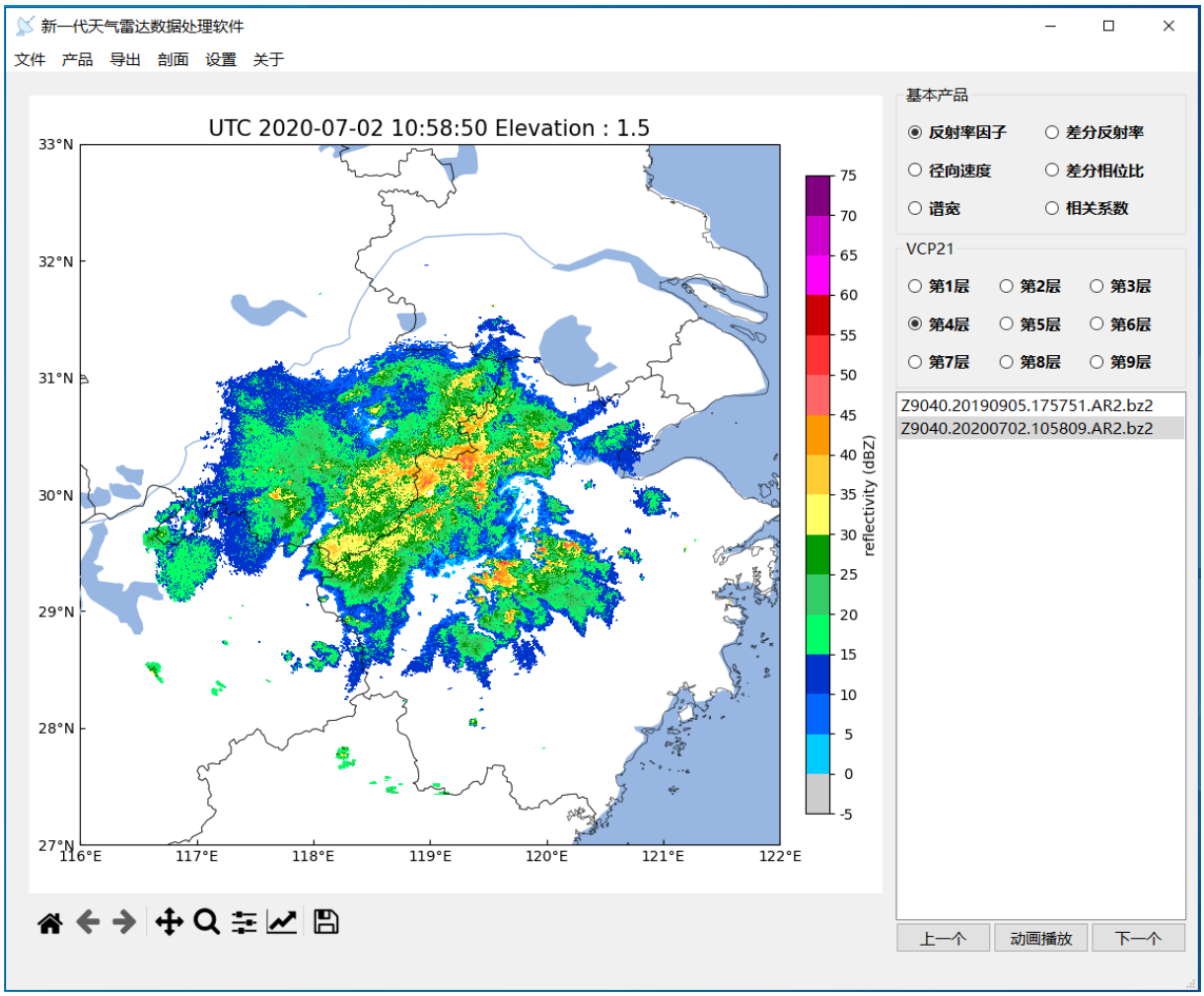
```
pip install pycwr==0.3.6
```

1.3 图形化界面显示

用 Python 执行此脚本可以启动图形化界面程序：

```
import warnings
warnings.filterwarnings('ignore')
import os, sys
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from pycwr.GraphicalInterface.RadarInterface import MainWindow
from PyQt5 import QtWidgets
app = QtWidgets.QApplication(sys.argv)
ui = MainWindow()
ui.show()
sys.exit(app.exec_())
```

图形化界面打开后如下图：



在右侧可以选择要显示的 **产品**和 **仰角**，在 **设置-> 显示设置-> 叠加地图**可以选择显示的图像是否叠加地图。

1.4 常见问题

用户指南

- 数据读取
- 数据结构
- 绘图
- 导出数据
- 选取数据
- 水凝物分类
- 衰减订正
- 雷达数据插值
- 组合反射率产品
- CAPPI 产品
- 风场反演

1.5 数据读取

双偏振雷达普及之前，我国新一代天气雷达数据主要有 CINRAD/SA/SB/CB、CINRAD/CC/CCJ、CINRAD/SC/CD 等格式，随着双偏振雷达数据的普及，我国逐渐将数据统一为 WSR98D 格式。

文档中采用的数据可通过百度网盘下载，链接: https://pan.baidu.com/s/1MhW9jE7r3LYHO7enUOfRsw 提取码: 8ba9

1.5.1 自动读取

可以使用 read_auto 函数实现雷达数据类型的识别，并转化为 pycwr 内置的雷达数据存储格式 PRD（Polarimetry Radar Data）类型：

```
from pycwr.io import read_auto
PRD = read_auto(r"./data/Z_RADR_I_Z9898_20190828181529_O_DOR_SAD_CAP_FMT.bin.bz2")
print(PRD.scan_info)
print(PRD.fields)
```

read_auto 函数在 `pycwr.io` 下，可以传入站点的经纬度高度参数，如不传入该参数，则会根据站号或者站点信息识别站点的经纬度数据。

read_auto 函数 `read_auto(filename, station_lon=None, station_lat=None, station_alt=None)` 共有四个参数：

- filename : 雷达基数据路径，基数据可以是 bz2 和 gz 格式的压缩文件（不需要解压）。
- station_lon : 雷达站点的经度信息，范围是-180 ~ 180。（units: degree east）
- station_lat : 雷达站点的纬度信息，范围是-90 ~ 900。（units: degree north）
- station_alt : 雷达站点的高度信息。（units: meters）

1.5.2 指定类型读取

如果已经知道雷达数据格式的类型，可以直接指定格式进行读取。

1. 数据类型是 WSR98D：

```
from pycwr.io import WSR98DFile
filename = "./data/Z_RADR_I_Z9898_20190828181529_O_DOR_SAD_CAP_FMT.bz2"
PRD = WSR98DFile.WSR98D2NRadar(WSR98DFile.WSR98DBaseData(filename,station_lon=None,\
 station_lat=None, station_alt=None)).ToPRD()
print(PRD.scan_info)
print(PRD.fields)
```

2. 数据类型是 SA/SB：

```
from pycwr.io import SABFile
filename = "./data/Z9396_BASE_SB_20180724_055400.bin.bz2"
PRD = SABFile.SAB2NRadar(SABFile.SABBaseData(filename, station_lon=None,\
 station_lat=None, station_alt=None)).ToPRD()
print(PRD.scan_info)
print(PRD.fields)
```

3. 数据类型是 CC/CCJ：

```
from pycwr.io import CCFile
filename = "./data/2016070817.48V.gz"
PRD = CCFile.CC2NRadar(CCFile.CCBaseData(filename, station_lon=None,\
 station_lat=None, station_alt=None)).ToPRD()
print(PRD.scan_info)
print(PRD.fields)
```

4. 数据类型是 SC/CD：

```
from pycwr.io import SCFile
filename = "./data/Z_RADR_I_Z9240_20190703101340_O_DOR_SC_CAP.bin.bz2"
PRD = SCFile.SC2NRadar(SCFile.SCBaseData(filename, station_lon=None,\
 station_lat=None, station_alt=None)).ToPRD()
```

(continues on next page)

(continued from previous page)

```
print(PRD.scan_info)
print(PRD.fields)
```

5. 数据类型是 Phase Array：

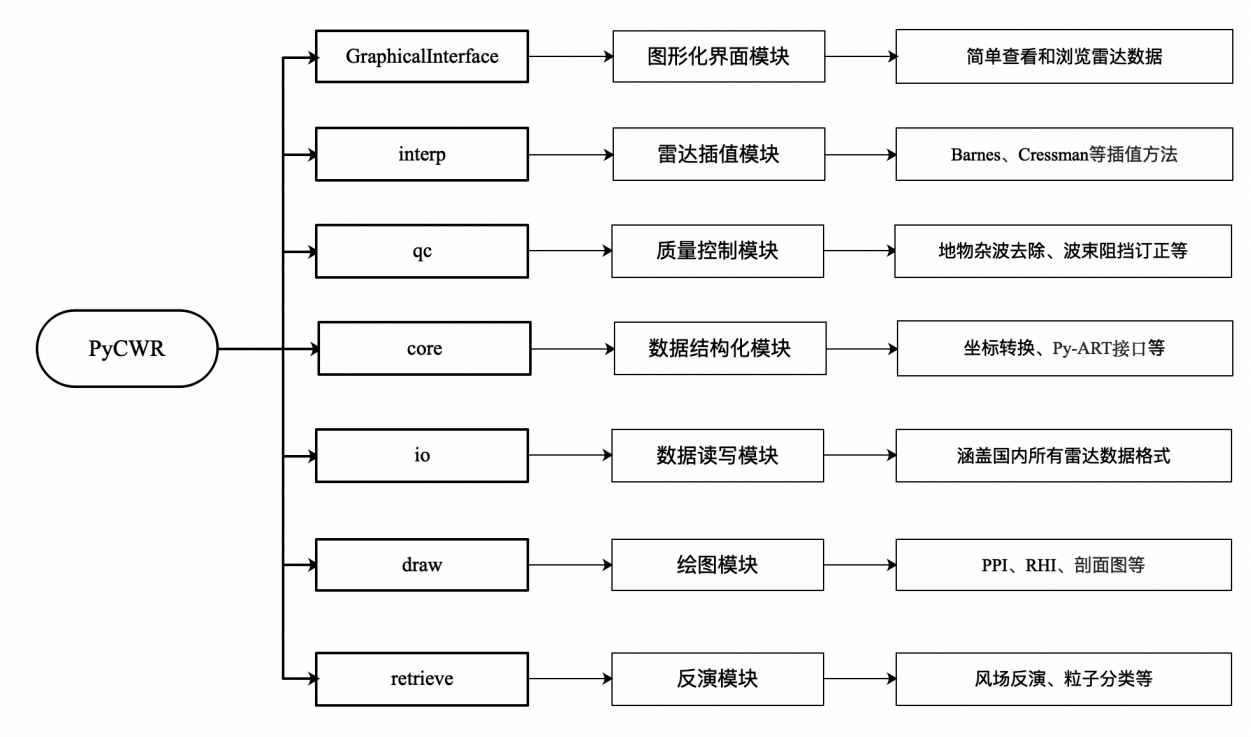
```
from pycwr.io import read_PA
filename = "./data/Z_RADR_I_ZGZ01_20200820220246_O_DOR_DXK_CAR.bin.bz2"
PRD = read_PA(filename)
print(PRD.scan_info)
print(PRD.fields)
```

1.5.3 配合 Py-ART 库对雷达数据进行处理

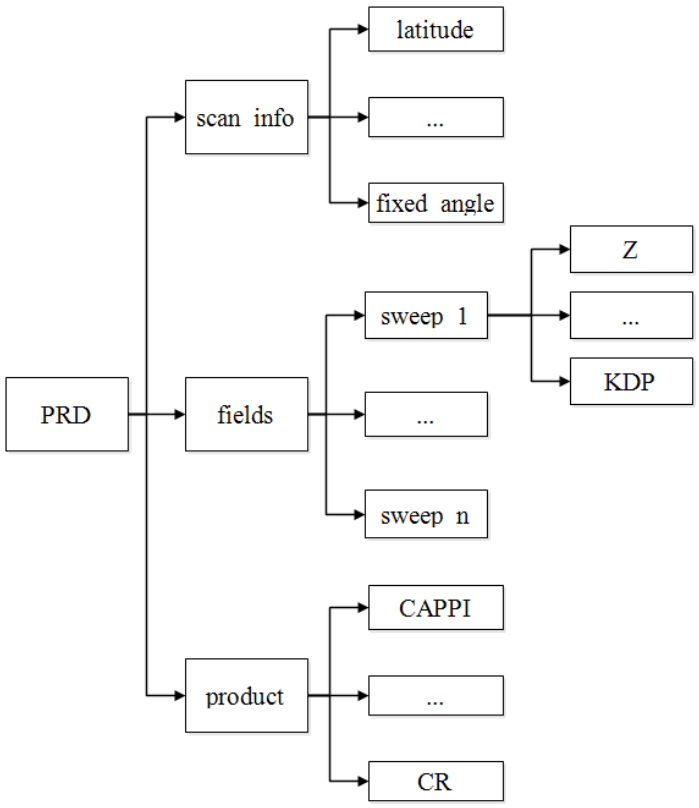
```
from pycwr.io import read_auto
PRD = read_auto(r"./data/Z_RADR_I_Z9898_20190828181529_O_DOR_SAD_CAP_FMT.bin.bz2")
print(PRD.scan_info)
print(PRD.fields)
PyartRadar = PRD.ToPyartRadar()
```

1.6 数据结构

PyCWR 的主要模块及作用：



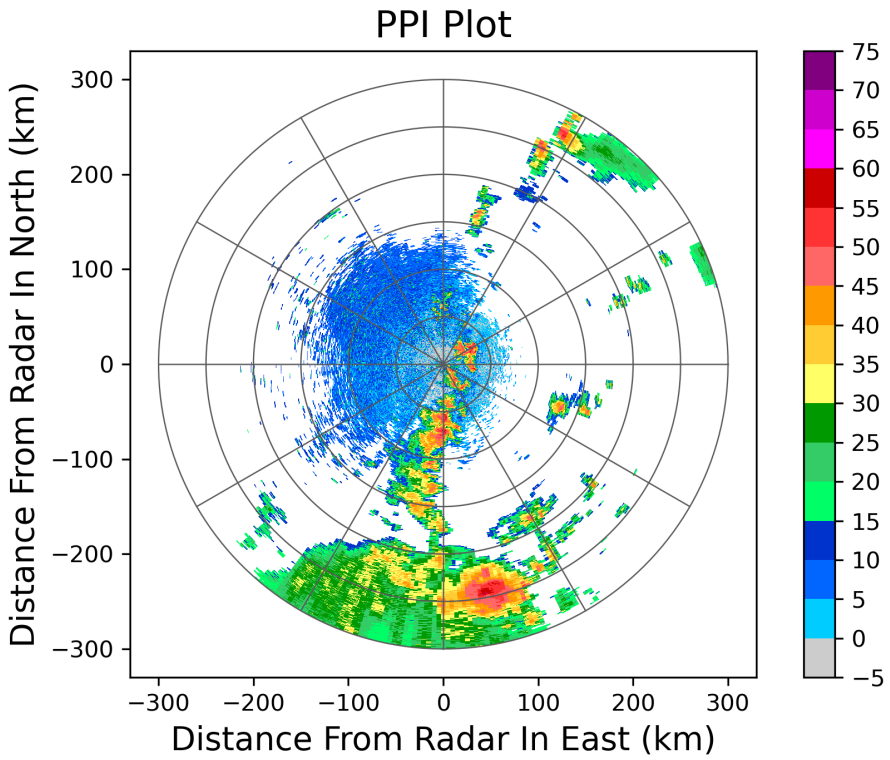
为了增加雷达数据的易用性，读取后的雷达数据会转换为偏振雷达数据（PRD, Polarimetric Radar Data）类，该类由两部分构成，一部分是扫描信息（scan_info），它主要包含了雷达站点的经纬度、高度、扫描类型、雷达频率、不模糊速度等信息，另一部分是观测数据（fields），它是一个列表容器，存储不同仰角的雷达扫描数据，列表的每个元素都是 xarray 的 DataArray 类，DataArray 类提供了一个围绕 numpy 的 ndarray 类的包装器，该包装器使用标注的维度和坐标来支持元数据操作，极大的提高了数据的可读性和可理解性。其结构如下：



1.7 绘图

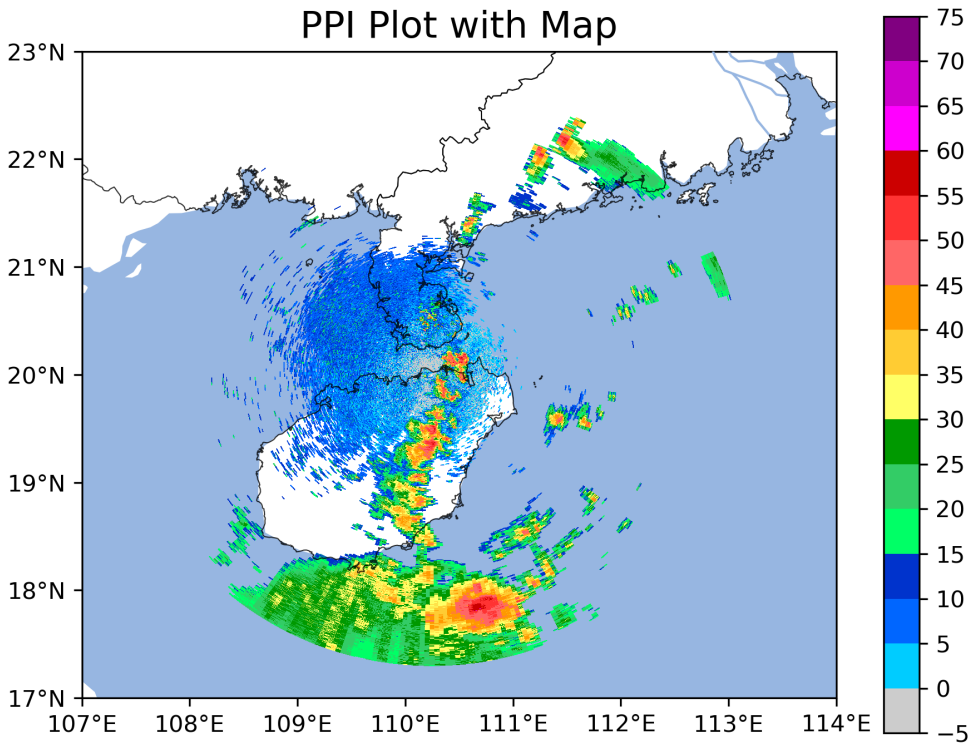
PPI 绘图不叠加地图：

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import Graph

filename = r"./data/Z_RADR_I_Z9898_20190828181529_O_DOR_SAD_CAP_FMT.bin.bz2"
PRD = read_auto(filename)
fig, ax = plt.subplots()
graph = Graph(PRD)
graph.plot_ppi(ax, 0, "dBZ", cmap="CN_ref") ## 0 代表第一层, dBZ 代表反射率产品
graph.add_rings(ax, [0, 50, 100, 150, 200, 250, 300])
ax.set_title("PPI Plot", fontsize=16)
ax.set_xlabel("Distance From Radar In East (km)", fontsize=14)
ax.set_ylabel("Distance From Radar In North (km)", fontsize=14)
plt.show()
```



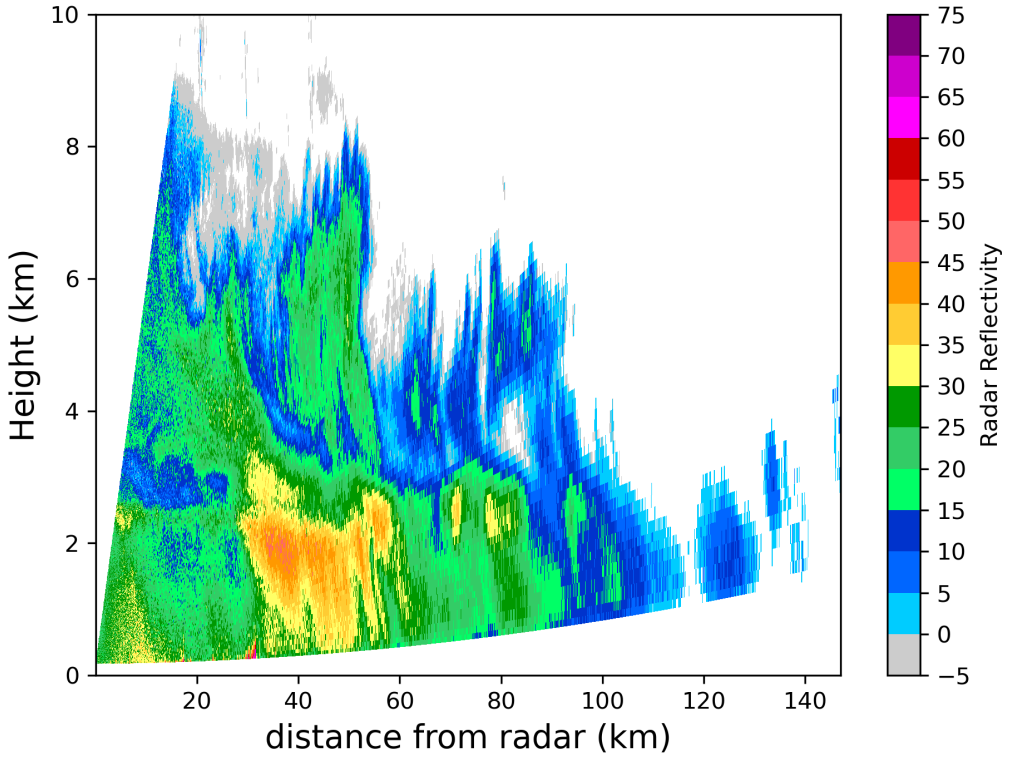
PPI 绘图叠加地图：

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import GraphMap
import cartopy.crs as ccrs
filename = r"./data/Z_RADR_I_Z9898_20190828181529_O_DOR_SAD_CAP_FMT.bin.bz2"
PRD = read_auto(filename)

ax = plt.axes(projection=ccrs.PlateCarree())
graph = GraphMap(PRD, ccrs.PlateCarree())
graph.plot_ppi_map(ax, 0, "dBZ", cmap="CN_ref") ## 0 代表第一层, dBZ 代表反射率产品, cmap
ax.set_title("PPI Plot with Map", fontsize=16)
plt.tight_layout()
plt.show()
```



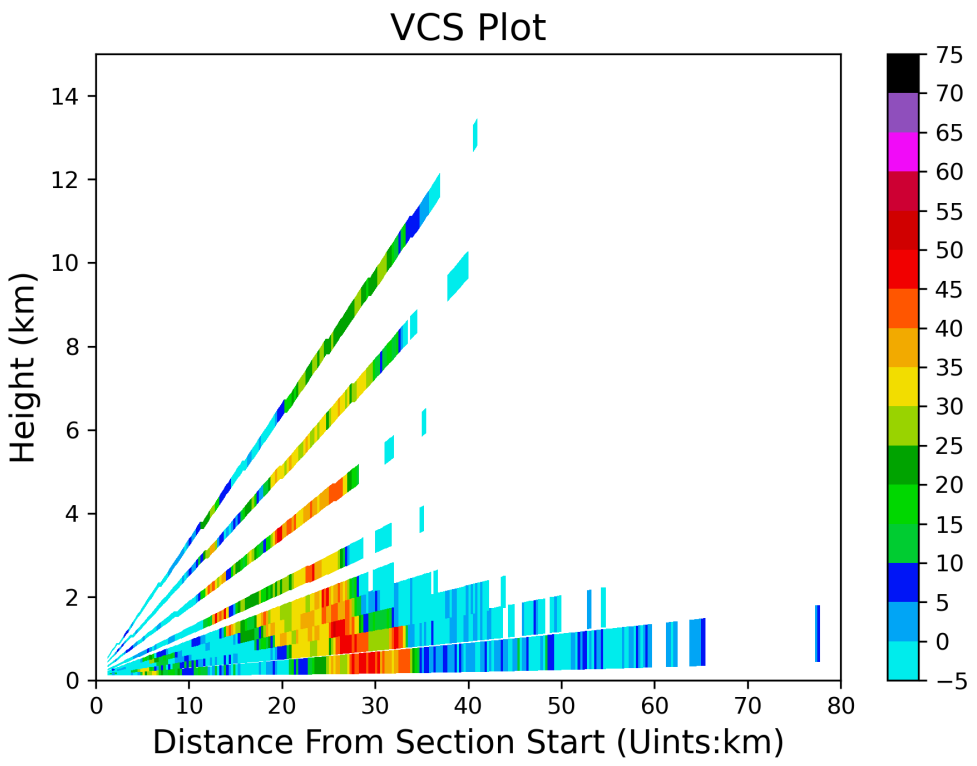
雷达 RHI 绘图:

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import Graph

filename = r"./data/NUIST.20170323.142921.AR2"
PRD = read_auto(filename)

fig, ax = plt.subplots()
graph = Graph(PRD)
graph.plot_rhi(ax, 0, field_name="dBZ", cmap="CN_ref", clabel="Radar Reflectivity")
ax.set_ylim([0, 10]) # 设置 rhi 的高度范围 (units: km)
ax.set_xlabel("distance from radar (km)", fontsize=14)
ax.set_ylabel("Height (km)", fontsize=14)
plt.tight_layout()
plt.show()
```



天气雷达剖面图：

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import Graph

filename = r"./data/Z_RADR_I_Z9898_20190828181529_O_DOR_SAD_CAP_FMT.bin.bz2"
PRD = read_auto(filename)

fig, ax = plt.subplots()
graph = Graph(PRD)
graph.plot_vcs(ax, (0,0), (150, 0), "dBZ", cmap="copy_pyart_NWSRef") # 起点，终点
(units: km)
ax.set_ylim([0, 15])
ax.set_xlim([0, 80])
ax.set_ylabel("Height (km)", fontsize=14)
ax.set_xlabel("Distance From Section Start (Uints:km)", fontsize=14)
ax.set_title("VCS Plot", fontsize=16)
plt.tight_layout()
plt.show()
```



1.8 导出数据

目前可以通过 pyart 将数据导出为 cfradial/UF 格式

```
from pycwr.io import read_auto
import pyart
import warnings
warnings.filterwarnings("ignore")

file = "./data/Z_RADR_I_Z9898_20190828181529_O_DOR_SAD_CAP_FMT.bin.bz2"
PRD = read_auto(file)
pyart_radar = PRD.ToPyartRadar()
pyart.io.write_cfradial("./cfradial.nc", pyart_radar)
```

1.9 选取数据

fields 属性存放了雷达体扫数据:

```
>>> from pycwr.io import read_auto
>>> PRD = read_auto("./data/NUIST.20150627.002438.AR2.bz2")
```

通过 [层数] 方法索引对应产品, 层数从 0 开始:

```
>>> print(PRD.fields[0])
<xarray.Dataset>
Dimensions:    (range: 1933, time: 668)
Coordinates:
    azimuth    (time) float64 9.149 9.69 10.22 10.77 ... 8.094 8.611 9.168 9.693
    elevation  (time) float64 0.5933 0.5933 0.5933 ... 0.5054 0.5054 0.5054
    x          (time, range) float64 11.92 23.85 35.77 ... 2.439e+04 2.44e+04
    y          (time, range) float64 74.04 148.1 222.1 ... 1.428e+05 1.429e+05
    z          (time, range) float64 174.8 175.6 176.3 ... 2.688e+03 2.689e+03
    lat        (time, range) float64 32.21 32.21 32.21 ... 33.49 33.49 33.49
    lon        (time, range) float64 118.7 118.7 118.7 ... 119.0 119.0 119.0
  * range      (range) float64 75.0 150.0 225.0 ... 1.448e+05 1.449e+05 1.45e+05
  * time       (time) datetime64[ns] 2015-06-27T00:24:38.640231 ... 2015-06-2...
Data variables:
    dBZ        (time, range) float64 -13.6 17.34 24.1 ... 18.62 18.24 19.32
    dBT        (time, range) float64 -13.6 17.34 24.1 ... 18.62 18.24 19.32
    V          (time, range) float64 0.0 -0.08 -1.12 -1.55 ... -1.76 -3.22 -3.41
    W          (time, range) float64 0.01 0.01 0.01 1.12 ... 1.63 1.43 1.5 1.68
    ZDR        (time, range) float64 -0.81 13.73 0.41 -0.39 ... 1.1 0.67 0.43
    KDP        (time, range) float64 7.03 6.13 5.44 4.78 ... 0.5 0.47 0.48 0.51
    PhiDP      (time, range) float64 100.0 33.3 1.02 2.86 ... 103.2 104.5 102.6
    CC         (time, range) float64 1.0 0.57 0.938 0.994 ... 0.923 0.938 0.96
    SQI        (time, range) float64 1.0 1.0 1.0 0.966 ... 0.913 0.905 0.898
    SNRH       (time, range) float64 67.77 81.21 81.95 ... 14.46 14.08 15.16
```

通过 [层数][关键词] 方法索引对应产品, 关键词详见表 1:

```
>>> print(PRD.fields[0]["dBZ"])
<xarray.DataArray 'dBZ' (time: 668, range: 1933)>
array([[-13.60000038,  17.34000015,  24.10000038, ...,  16.87999916,
         16.76000023,  17.52000046],
       [-13.60000038,  16.65999985,  24.09000015, ...,  18.37000084,
         16.42000008,  17.86000061],
       [-13.60000038,  18.03000069,  23.57999992, ...,  16.98999977,
         20.04000092,  20.85000038],
```

(continues on next page)

(continued from previous page)

```
       ...,
       [-13.60000038,  12.61999989,  24.        , ...,  15.63000011,
         14.63000011,  13.81000042],
       [-13.60000038,  15.75      ,  25.35000038, ...,  12.85999966,
         17.29999924,  14.15999985],
       [-13.60000038,  14.97999954,  24.55999947, ...,  18.62000084,
         18.23999977,  19.31999969]])
Coordinates:
    azimuth    (time) float64 9.149 9.69 10.22 10.77 ... 8.094 8.611 9.168 9.693
    elevation  (time) float64 0.5933 0.5933 0.5933 ... 0.5054 0.5054 0.5054
    x          (time, range) float64 11.92 23.85 35.77 ... 2.439e+04 2.44e+04
    y          (time, range) float64 74.04 148.1 222.1 ... 1.428e+05 1.429e+05
    z          (time, range) float64 174.8 175.6 176.3 ... 2.688e+03 2.689e+03
    lat        (time, range) float64 32.21 32.21 32.21 ... 33.49 33.49 33.49
    lon        (time, range) float64 118.7 118.7 118.7 ... 119.0 119.0 119.0
  * range      (range) float64 75.0 150.0 225.0 ... 1.448e+05 1.449e+05 1.45e+05
  * time       (time) datetime64[ns] 2015-06-27T00:24:38.640231 ... 2015-06-2...
Attributes:
    units:          dBZ
    standard_name:  equivalent_reflectivity_factor
    long_name:      Reflectivity
    valid_max:      80.0
    valid_min:      -30.0
    coordinates:    elevation azimuth range
```

scan_info 存放了雷达位置信息以及扫描方式:

```
>>> print(PRD.scan_info)
<xarray.Dataset>
Dimensions:            (sweep: 15)
Coordinates:
  * sweep              (sweep) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Data variables:
    latitude           float64 32.21
    longitude          float64 118.7
    altitude           int64 87
    scan_type          <U3 'ppi'
    frequency          float64 5.592
    start_time         datetime64[ns] 2015-06-27T00:24:38.640231
    end_time           datetime64[ns] 2015-06-27T00:31:47.207645
    nyquist_velocity   (sweep) float32 13.4 13.4 13.4 13.4 ... 26.81 26.81 26.81
    unambiguous_range  (sweep) int32 145000 145000 145000 ... 70000 70000 10000
    rays_per_sweep     (sweep) int64 668 668 668 668 668 ... 669 668 668 668 668
    fixed_angle        (sweep) float32 0.5 1.5 2.4 3.4 ... 14.0 16.7 19.5 90.0
```

(continues on next page)

(continued from previous page)

```
    beam_width          (sweep) float64 0.5389 0.5389 0.5389 ... 0.5389 0.5389
```

表 1，不同变量对应索引的关键词:

key	variable
dBT	total_power
dBZ	reflectivity
V	velocity
W	spectrum_width
SQI	normalized_coherent_power
CPA	clutter_phase_alignment
ZDR	differential_reflectivity
LDR	linear_depolarization_ratio
CC	cross_correlation_ratio
PhiDP	differential_phase
KDP	specific_differential_phase
CP	clutter_probability
Flag	flag_of_rpv_data
HCL	hydro_class
CF	clutter_flag
Zc	corrected_reflectivity
Vc	corrected_velocity
Wc	spectrum_width_corrected
SNRH	horizontal_signal_noise_ratio
SNRV	vertical_signal_noise_ratio

1.10 水凝物分类

水凝物分类示例程序:

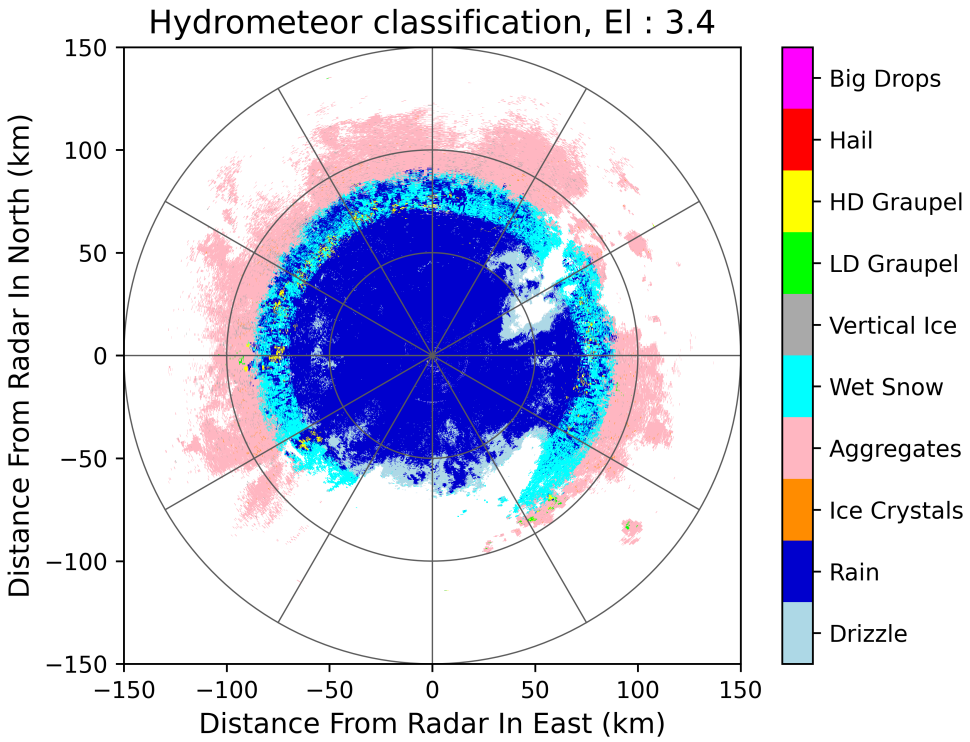
```
# -*- coding: utf-8 -*-
from pycwr.io import read_auto
from pycwr.retrieve.HID import fhc_HCL
import matplotlib.pyplot as plt
import numpy as np
from pycwr.draw.RadarPlot import plot_xy, add_rings
import pandas as pd

file = r"./data/NUIST.20150627.002438.AR2.bz2"
```

(continues on next page)

(continued from previous page)

```
file_t = r"./data/20150627.csv"
temp = pd.read_csv(file_t, index_col=0, header=None, names=['temp'])

NRadar = read_auto(file)
num = 3
dBZ = np.where(NRadar.fields[num].CC>0.9, NRadar.fields[num].dBZ, np.nan)
KDP = np.where(NRadar.fields[num].CC>0.9, NRadar.fields[num].KDP, np.nan)
ZDR = np.where(NRadar.fields[num].CC>0.9, NRadar.fields[num].ZDR, np.nan)
CC = np.where(NRadar.fields[num].CC>0.9, NRadar.fields[num].CC, np.nan)
temp_2d = np.interp(NRadar.fields[num].z.values/1000., temp.index, temp['temp'])
dBZ[:,0] = np.nan
ticks = np.arange(1, 11, 1)
ticklabels = ['Drizzle', 'Rain', 'Ice Crystals', 'Aggregates', 'Wet Snow', 'Vertical␣
↪Ice',
               'LD Graupel', 'HD Graupel', 'Hail', 'Big Drops']

hcl = fhc_HCL(dBZ=dBZ, KDP=KDP, ZDR=ZDR, CC = CC, T=temp_2d)
fig, ax = plt.subplots()
plot_xy(ax, NRadar.fields[num].x, NRadar.fields[num].y, hcl,
        cmap="CN_hcl", bounds=np.arange(0.5,10.6,1),
        cbar_ticks=ticks, cbar_ticklabels=ticklabels)
add_rings(ax=ax, rings=[0, 50, 100, 150])
ax.set_xlim([-150, 150])
ax.set_ylim([-150, 150])
ax.set_xlabel("Distance From Radar In East (km)", fontsize=12)
ax.set_ylabel("Distance From Radar In North (km)", fontsize=12)
ax.set_title("Hydrometeor classification, El : 3.4", fontsize=14)
plt.savefig(r"./201506270024_HC.png", dpi=600)
plt.show()
```

水凝物分类效果:



1.11 衰减订正

1.12 雷达数据插值

1.13 组合反射率产品

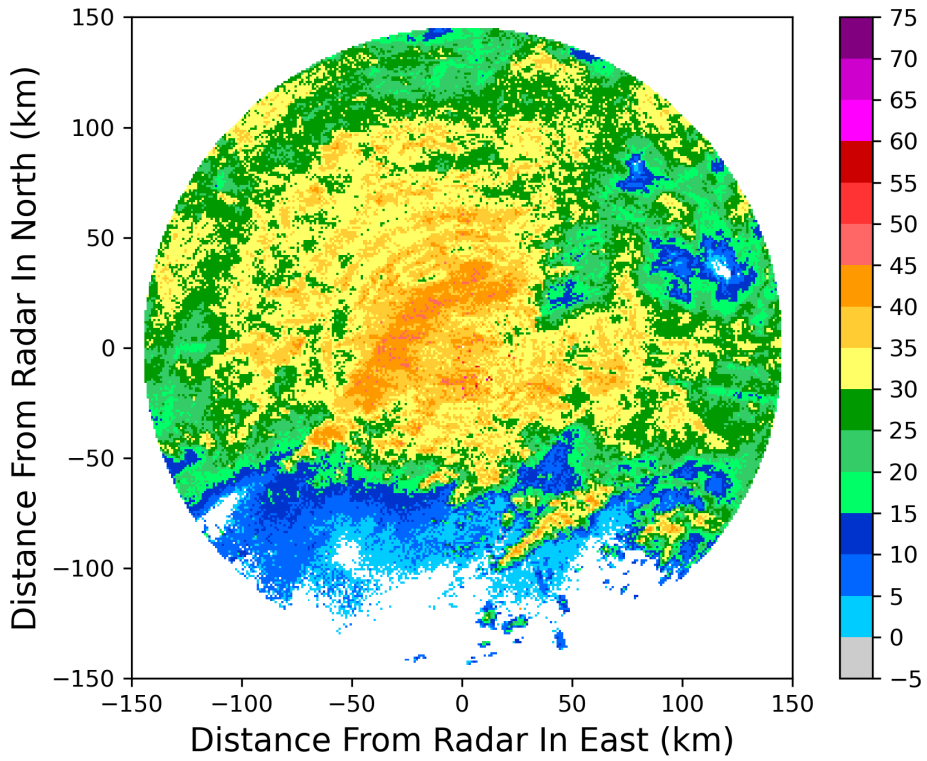
以雷达为中心，生成笛卡尔坐标的组合反射率网格产品

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import plot_xy
import numpy as np

filename = r"../../data/NUIST.20150627.002438.AR2.bz2"
PRD = read_auto(filename)

x1d = np.arange(-150000, 150001, 1000) ##x 方向 1km 等间距, -150km~150km 范围
y1d = np.arange(-150000, 150001, 1000) ##y 方向 1km 等间距, -150km~150km 范围
PRD.add_product_CR_xy(XRange=x1d, YRange=y1d)
print(PRD.product)
```

(continues on next page)

(continued from previous page)

```
grid_x, grid_y = np.meshgrid(x1d, y1d, indexing="ij")
fig, ax = plt.subplots()
plot_xy(ax, grid_x, grid_y, PRD.product.CR) ## 画图显示
ax.set_xlabel("Distance From Radar In East (km)", fontsize=14)
ax.set_ylabel("Distance From Radar In North (km)", fontsize=14)
plt.tight_layout()
plt.show()
```



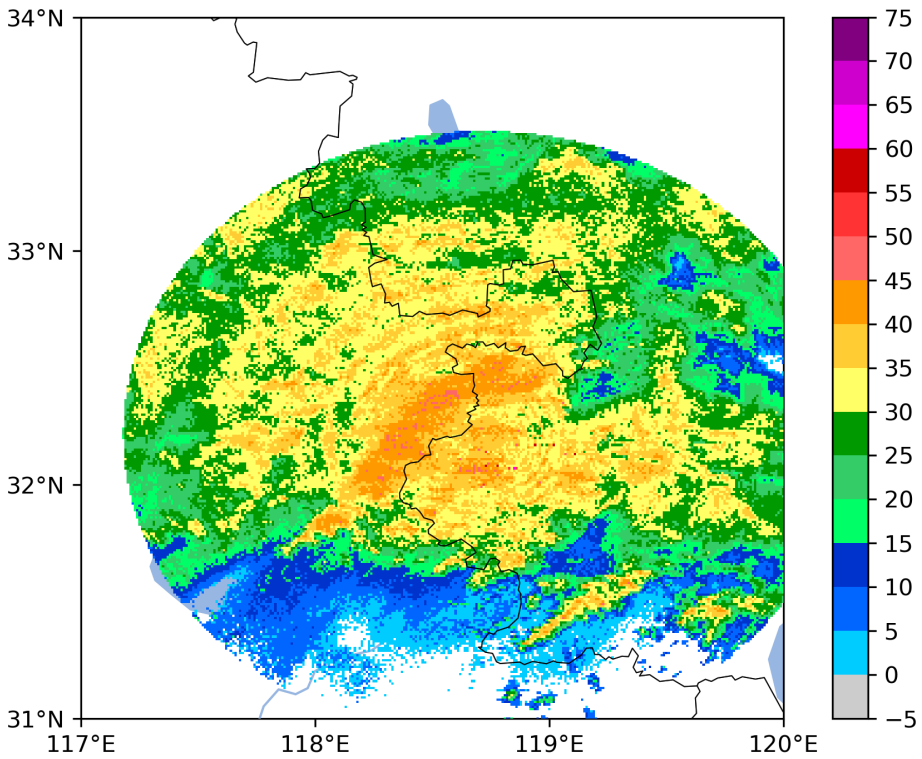
利用经纬度坐标信息生成组合反射率网格产品

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import plot_lonlat_map
import cartopy.crs as ccrs
import numpy as np

filename = r"./data/NUIST.20150627.002438.AR2.bz2"
PRD = read_auto(filename)

lon1d = np.arange(117, 120.001, 0.01) ##lon 方向 0.01 等间距, 117-120 范围
lat1d = np.arange(31, 34.001, 0.01) ##lat 方向 0.01 等间距, 31-34 度范围
PRD.add_product_CR_lonlat(XLon=lon1d, YLat=lat1d)
```

(continues on next page)

(continued from previous page)

```
# XLon:np.ndarray, 1d, units:degrees
# YLat:np.ndarray, 1d, units:degrees
# level_height: 常量, 要计算的高度 units:meters
grid_lon, grid_lat = np.meshgrid(lon1d, lat1d, indexing="ij")
ax = plt.axes(projection=ccrs.PlateCarree())
plot_lonlat_map(ax, grid_lon, grid_lat, PRD.product.CR_geo, transform=ccrs.
↪PlateCarree())
ax.set_extent([117, 120, 31, 34], crs = ccrs.PlateCarree()) # 设置显示范围
plt.tight_layout()
plt.show()
```



1.14 CAPPI 产品

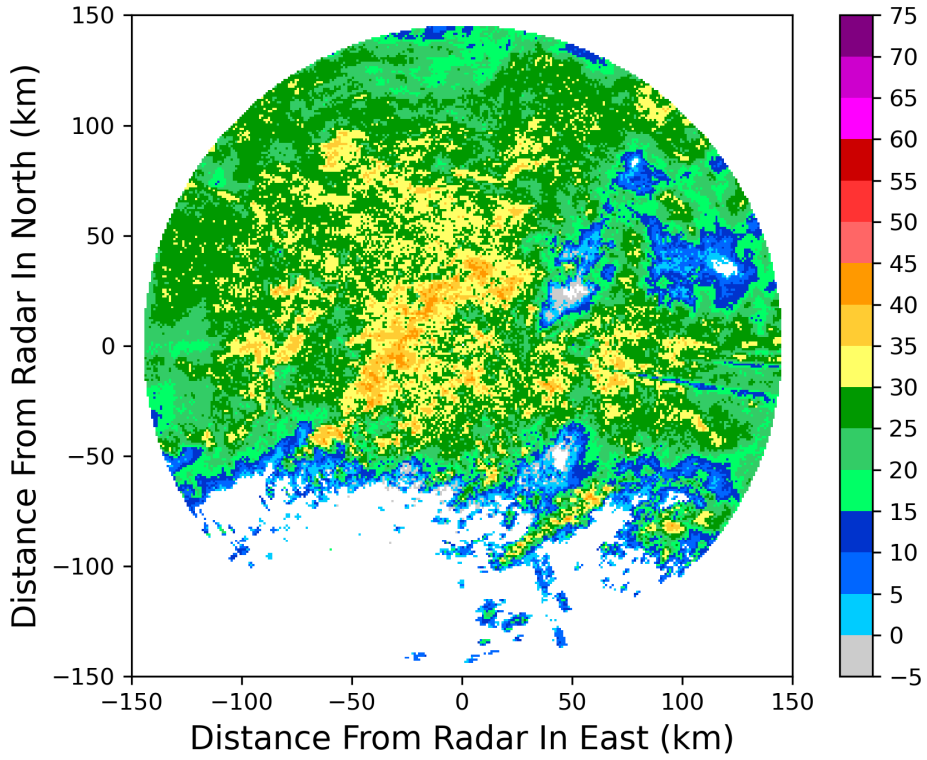
以雷达为中心，生成笛卡尔坐标的 CAPPI 网格产品

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import plot_xy
import numpy as np

```

(continues on next page)

(continued from previous page)

```
filename = r"./data/NUIST.20150627.002438.AR2.bz2"
PRD = read_auto(filename)

x1d = np.arange(-150000, 150001, 1000) ##x 方向 1km 等间距， -150km~150km 范围
y1d = np.arange(-150000, 150001, 1000) ##y 方向 1km 等间距， -150km~150km 范围
PRD.add_product_CAPPI_xy(XRange=x1d, YRange=y1d, level_height=3000) ##level height␣
↪units:meters
print(PRD.product)
grid_x, grid_y = np.meshgrid(x1d, y1d, indexing="ij")
fig, ax = plt.subplots()
plot_xy(ax, grid_x, grid_y, PRD.product.CAPPI_3000) ## 画图显示
ax.set_xlabel("Distance From Radar In East (km)", fontsize=14)
ax.set_ylabel("Distance From Radar In North (km)", fontsize=14)
plt.tight_layout()
plt.show()
```



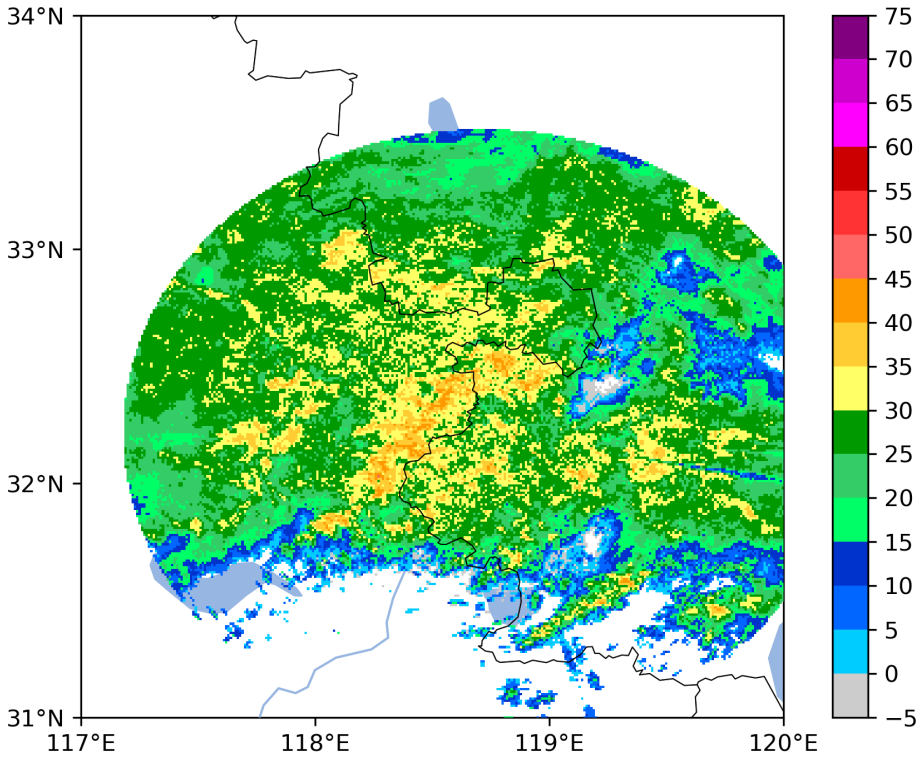
利用经纬度坐标信息生成 CAPPI 网格产品

```
from pycwr.io import read_auto
import matplotlib.pyplot as plt
from pycwr.draw.RadarPlot import plot_lonlat_map
import cartopy.crs as ccrs
```

(continues on next page)

(continued from previous page)

```
import numpy as np

filename = r"./data/NUIST.20150627.002438.AR2.bz2"
PRD = read_auto(filename)

lon1d = np.arange(117, 120.001, 0.01) ##lon 方向 0.01 等间距，117-120 范围
lat1d = np.arange(31, 34.001, 0.01) ##lat 方向 0.01 等间距， 31-34 度范围
PRD.add_product_CAPPI_lonlat(XLon=lon1d, YLat=lat1d, level_height=3000) ## 插值 1500m 高度的
# XLon:np.ndarray, 1d, units:degrees
# YLat:np.ndarray, 1d, units:degrees
# level_height: 常量，要计算的高度 units:meters
grid_lon, grid_lat = np.meshgrid(lon1d, lat1d, indexing="ij")
ax = plt.axes(projection=ccrs.PlateCarree())
plot_lonlat_map(ax, grid_lon, grid_lat, PRD.product.CAPPI_geo_3000, transform=ccrs.PlateCarree())
ax.set_extent([117, 120, 31, 34], crs = ccrs.PlateCarree()) # 设置范围
plt.tight_layout()
plt.show()
```



1.15 风场反演

CHAPTER

TWO

开发者信息

作者

郑玉[1] ; 李南[2] ; 魏鸣[2] ; 楚志刚[2]

地址

南京市建邺区雨顺路 6 号江苏省气象预警中心

邮箱

yuzheng1206@outlook.com; zhengyu@cma.gov.cn

贡献

特别感谢李扬、张昕、贾鹏程、吕星超和樊丝慧等人对项目的支持。

[1] 中国气象局交通气象重点开放实验室，南京气象科技创新研究院

[2] 大气物理学院，南京信息工程大学