ωradlib Getting Started Cheat Sheet

Learn more about *ωradlib* at http://wradlib.org

ωradlib Introduction

The *ωradlib* project has been initiated in order facilitate the use of weather radar data as well as to provide a common platform for research on new algorithms. *ωradlib* is an open source library which is well documented and easy to use. It is written in the free programming language Python.

Installation

We recommend using ***conda*** package manager alongside the ***conda-forge*** community channel:

- Install Anaconda or Miniconda [1]
- Add ***conda-forge*** channel:
  `$ conda config -add channels conda-forge`
- Create dedicated *ωradlib* environment:
  `$ conda create -name wradlib python=3.6`
- Activate *ωradlib* environment:
  `$ source activate wradlib`
- Install *ωradlib* and other needed packages:
  `(wradlib)$ conda install wradlib jupyter`

If you want to test the most recent *ωradlib* developments, then you need to get the latest **master** from *github.com* in addition:

- Clone *ωradlib* repository
  `$ git clone https://github.com/wradlib/wradlib.git`
- Activate *ωradlib* environment:
  `$ source activate wradlib`
- Install *ωradlib* from sources:
  `(wradlib)$ python setup.py install`

If you want to test the provided example notebooks, you need to download the example data [2] and extract it to an arbitrary directory. You finally need to set the **WRADLIB_DATA** environment variable pointing to that directory:

- `$ export WRADLIB_DATA=/full/path/to/wradlib-data`

[1] https://www.anaconda.com/download
https://conda.io/miniconda.html
[2] https://github.com/wradlib/wradlib-data/archive/master.zip

Getting Started

```
>>> import wradlib as wrl          Import using wrl as alias
>>> wrl.__version__                Print wradlib version
```

Reading Radar Data

- Polar Radar Data Reader

```
>>> img, meta = wrl.io.read_dx(f)                DWD's DX
>>> data = wrl.io.read_opera_hdf5(f)             ODIM_H5
>>> data = wrl.io.read_gamic_hdf5(f)             GAMIC
>>> data = wrl.io.read_edge_netcdf(f)            EDGE
>>> data = wrl.io.read_rainbow(f)                Rainbow5
>>> data = wrl.io.read_iris(f)                   Sigmet
```

- Gridded Radar Data Reader

```
>>> img, meta = read_radolan_composite(f)        RADOLAN
>>> data = wrl.io.read_rainbow(f)                Rainbow5
>>> data = wrl.io.read_iris(f)                   Sigmet
```

- Generic Data Format Reader

```
>>> data = wrl.io.read_generic_hdf5(f)           HDF5
>>> data = wrl.io.read_generic_netcdf(f)         NetCDF
```

- Raster Data Reader using GDAL

```
>>> ds = wrl.io.open_raster(f)                   open raster
>>> img, crd, proj =                             extract
    wrl.georef.extract_raster_dataset(ds)        raster data
```

Visualizing Radar Data

- Plot Polar Radar Data `img(nrays, nbins)`

```
>>> wrl.vis.plot_ppi(img)                        plot simple PPI
>>> wrl.vis.plot_ppi(img, cg=True)               Curvelinear Grid
>>> wrl.vis.plot_rhi(img)                        plot simple RHI
>>> wrl.vis.plot_rhi(img, cg=True)               Curvelinear Grid
```

- Plot Gridded Radar Data `img(nrows, ncols)`

```
>>> import matplotlib.pyplot as plt             matplotlib
>>> pl.imshow(img)                              use imshow
>>> pl.pcolormesh(img)                          use MeshPlot
>>> pl.pcolormesh(crd[..., 0],                  use coords
                  crd[..., 1], img)
```

Data Transformation

```
>>> y = wrl.trafo.rvp_to_dbz(x)                 RVP6 in dBZ
>>> dBZ = wrl.trafo.decibel(Z)                  decibel
>>> Z = wrl.trafo.idecibel(dBZ)                 inverse decibel
>>> RR = wrl.trafo.kdp_to_r(KDP)                Rainrate from KDP
>>> RR = wrl.zr.z_to_r(Z)                       Rainrate from Z
>>> Z = wrl.zr.r_to_z(RR)                       Z from RainRate
```

Data Classification

- wrl.clutter.filter_gabella() — Clutter id filter by Gabella
- wrl.clutter.filter_cloudtype() — Filter based on cloud type
- wrl.clutter.filter_window_distance() — 2D filter large gradients
- wrl.clutter.histo_cut() — Histogram clutter id
- wrl.clutter.classify_echo_fuzzy() — Dual-Pol fuzzy method

Data Correction

Gate-by-Gate approaches **wrl.atten**

- correct_attenuation_hb() — Hitschfeld&Bordan
- correct_attenuation_constrained() — iterative Kraemer (ext. by Jacobi)

Phase Processing

Phase Unfolding

- wrl.dp.unfold_phi() — unfolds ambiguous phase
- wrl.dp.unfold_phi_vulpiani() — KDP based unfolding

KDP Retrieval

- wrl.dp.kdp_from_phidp() — Lanczos derivative
- wrl.dp.process_raw_phidp_vulpiani() — 2-step PHIDP/KDP

Data Compositing

- wrl.comp.togrid() — polar to grid
- wrl.comp.compose_ko() — quality knockout criterion
- wrl.comp.compose_weighted() — quality weighted average

References

[1] Maik Heistermann, Stephan Jacobi, and Thomas Pfaff. Technical note: An open source library for processing weather radar data (wradlib). *Hydrol. Earth Syst. Sci.*, 16:863–871, 2013.

Other Resources

Check out the other available *ωradlib* Cheat Sheets which will be available shortly. Those will cover amongst others Visualisation, Georeferencing, Interpolation, Classification, Correction, Phase Processing, Compositing, Zonal Statistics, Gage Adjustment.

Contact

Web wradlib.org
Email wradlib@wradlib.org
Gitter gitter.im/wradlib