RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

METEO FRANCE

UNIVERSITÄT BONN

NCAR

MeteoSwiss

ARM

Argonne NATIONAL LABORATORY

Environment and Climate Change Canada

Environnement et Changement climatique Canada

An overview of open source software packages for radar meteorology

ERAD2022 Open Radar Short Course, Locarno, Switzerland, 28.08.2022

What is Open Source Software ?

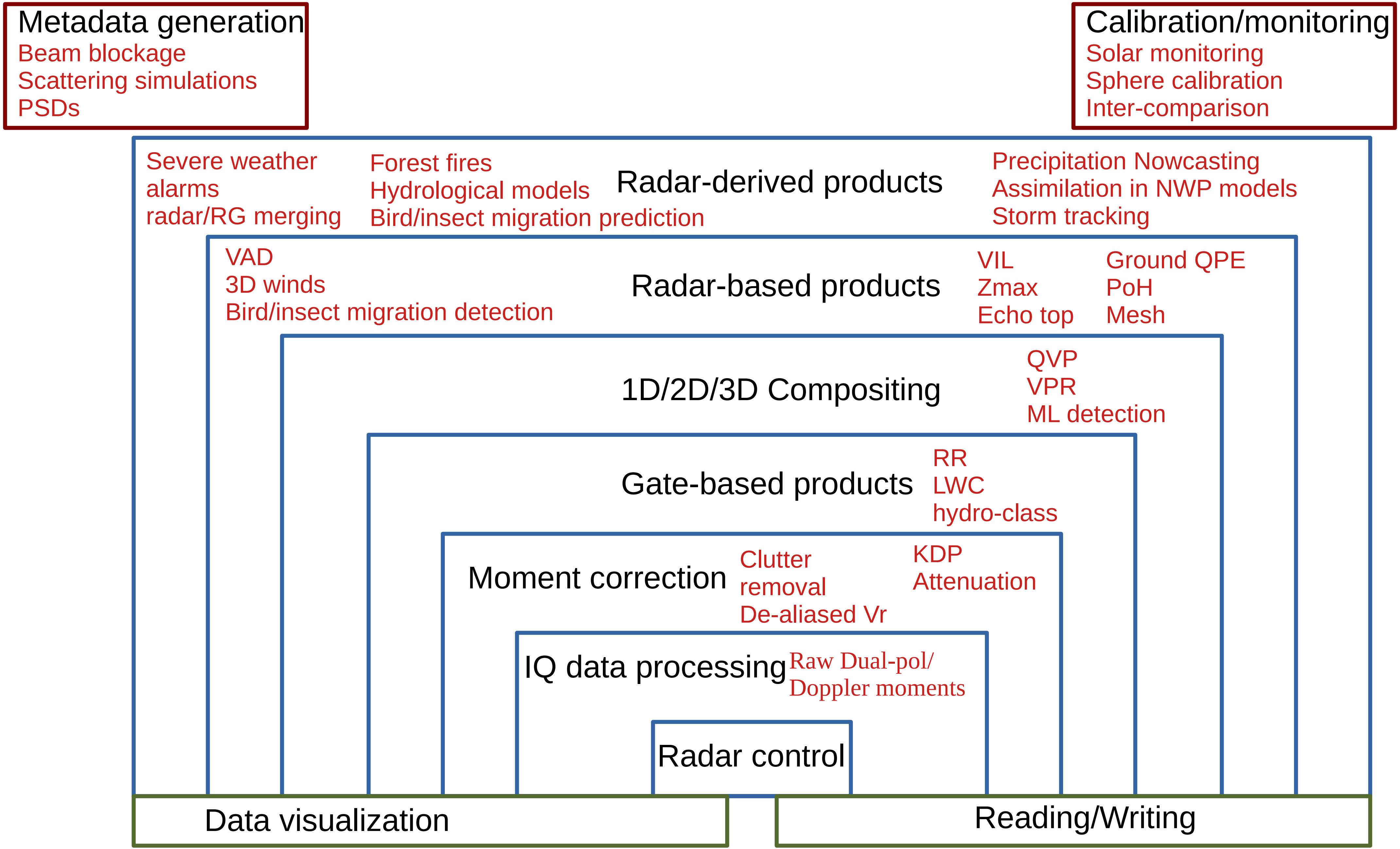
- A software package is open source if :
  - Allows **free** (senza pagare) redistribution
  - **Includes the source code** (or it is easily accessible) and allows distribution of both compiled and source code.
  - **Allows modifications** and derived works, allows distributing them under the same terms as the licence of the original software
  - <u>No discrimination</u> against persons or groups
  - <u>No discrimination</u> against fields of use (e.g. business, genetic research)
  - No need for any other licence (appart from the one of the package) to use it
  - The licence is not specific to a product, software within a distribution can be used and distributed independently
  - The licence does not restrict the use of other software
  - The licence is technology-neutral

https://opensource.org

Open source for weather radar ?

- Since the late 2000s (and even before) there has been a number of major open source projects released (see e.g. https://openradarscience.org).
- Some of them are in a mature stage and are widely used in an **academic** (mostly) but also **operational environment**
- Most make use of modern tools (e.g. github, conda, docker) and practices (e.g. Continuous Integration, automatic tests) that make them easy to evolve and deploy
- Most are backed by major weather services or academic institutions
- Projects **are not competing among them** but collaborating : Best practices and inter-operability are discussed regularly and joint open source courses have been organized for years at major radar conferences (AMS, ERAD)

The weather radar business



Metadata generation
Beam blockage
Scattering simulations
PSDs

Calibration/monitoring
Solar monitoring
Sphere calibration
Inter-comparison

Radar-derived products
Severe weather alarms
radar/RG merging
Forest fires
Hydrological models
Bird/insect migration prediction
Precipitation Nowcasting
Assimilation in NWP models
Storm tracking

Radar-based products
VAD
3D winds
Bird/insect migration detection
VIL
Zmax
Echo top
Ground QPE
PoH
Mesh

1D/2D/3D Compositing
QVP
VPR
ML detection

Gate-based products
RR
LWC
hydro-class

Moment correction
Clutter removal
De-aliased Vr
KDP
Attenuation

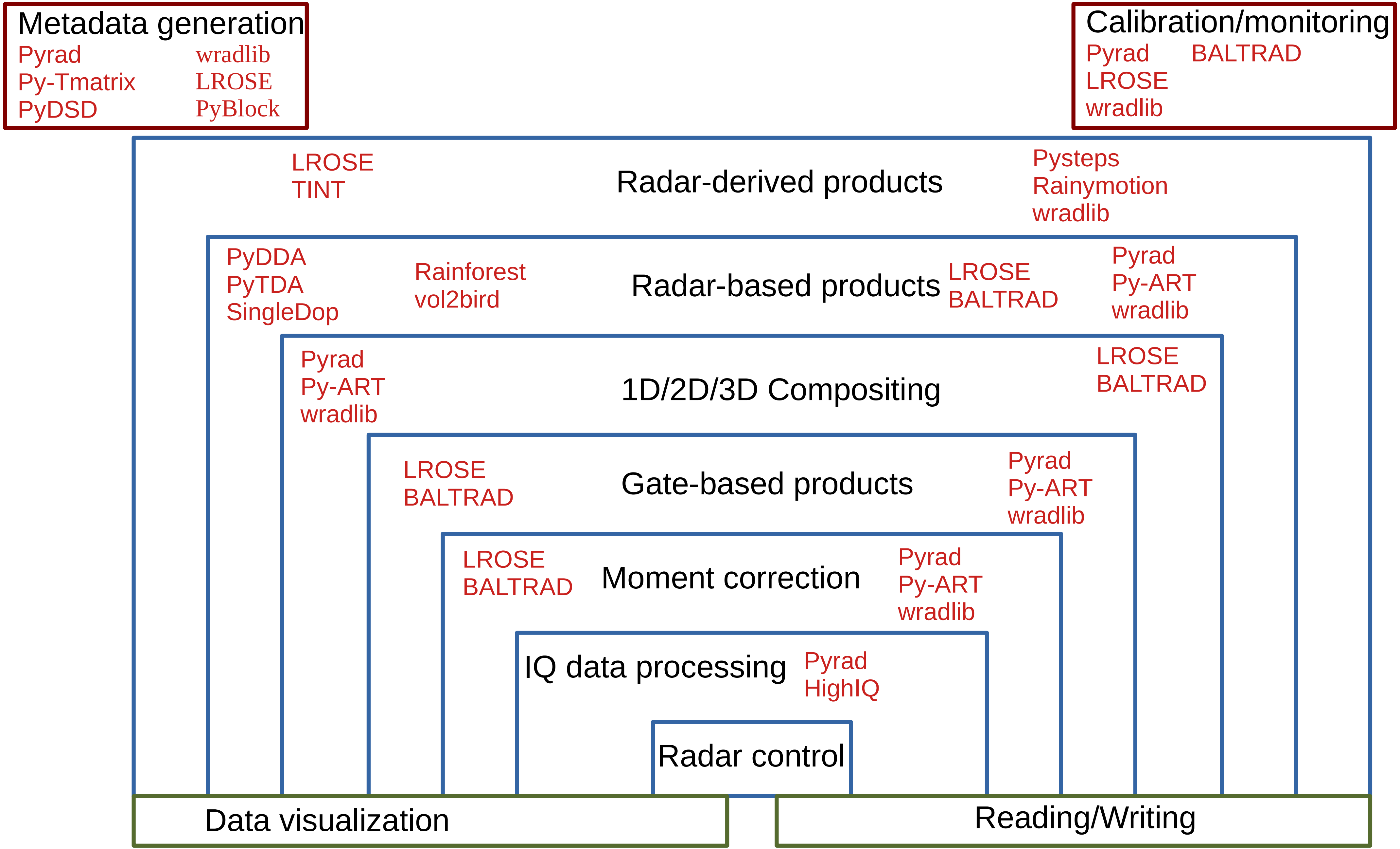
IQ data processing
Raw Dual-pol/ Doppler moments

Radar control

Data visualization

Reading/Writing

The weather radar business



Metadata generation
Pyrad
Py-Tmatrix
PyDSD
wradlib
LROSE
PyBlock

Calibration/monitoring
Pyrad
LROSE
wradlib
BALTRAD

Radar-derived products
LROSE
TINT
Pysteps
Rainymotion
wradlib

Radar-based products
PyDDA
PyTDA
SingleDop
Rainforest
vol2bird
LROSE
BALTRAD
Pyrad
Py-ART
wradlib

1D/2D/3D Compositing
Pyrad
Py-ART
wradlib
LROSE
BALTRAD

Gate-based products
LROSE
BALTRAD
Pyrad
Py-ART
wradlib

Moment correction
LROSE
BALTRAD
Pyrad
Py-ART
wradlib

IQ data processing
Pyrad
HighIQ

Radar control

Data visualization

Reading/Writing

Other useful meteorological software

Py-TROLL : satellite data processing

WRF: weather Research and Forecasting Model

MetPy: weather data visualization

Metview: Meteorological workstation

MetWork Framework: Useful modules to build meteorological applications

Radar Data Formats

Radar data takes different formats at each processing stage:

- IQ data: Time series of complex numbers
- Moments: Polar coordinates (azimuth, elevation, range)
- Composites: Cartesian/geo-referenced grids
- Radar-based products: Grids but also time-height, time-range, etc.
- Radar-derived products: ??????

There is no formally accepted standard yet for radar data at any stage

Most radar manufacturers and major Met services use their own proprietary formats

There are 3 de-facto standards for **moment data** file formats:

- ODIM_H5
- CfRadial
- NEXRAD-AR2

CfRadial

NetCDF Climate and forecast (CF) Conventions for RADAR and LIDAR data in polar coordinates

Based on Network Common Data Form (NetCDF)

Maintained by NCAR

De-facto standard for the research community

Two major versions:

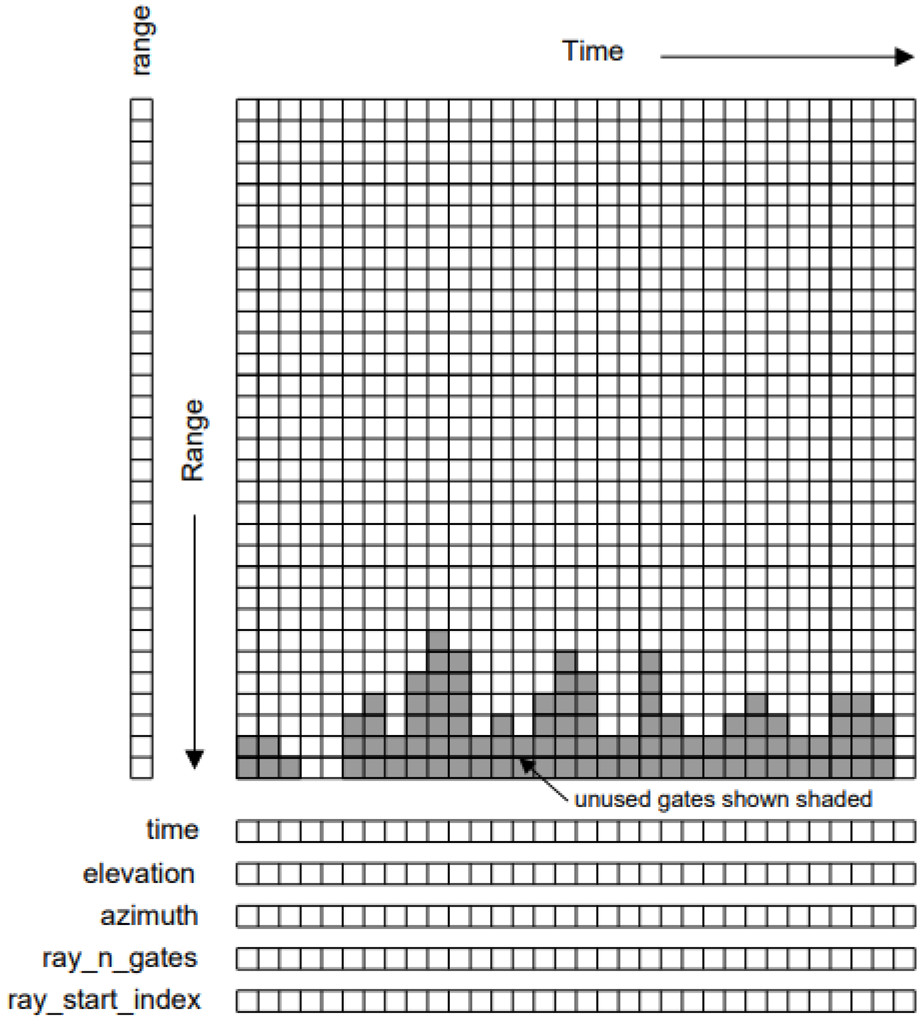
- CfRadial Version 1: (Since 2010) Classic model using NetCDF3 => Py-ART data model
  - Data stored in regular 2D (time, range) format
  - Metadata: range, time, elevation, azimuth, (ray_n_gates, ray_start_index)
- CfRadial version 2: (Since 2016) uses NetCDF4 (based on HDF5) and groups
  - Hierarchical grouping volume=>sweep=>dataset (time, range)
  - Candidate for WMO radar data standard (FM301)

Readers: wradlib, BALTRAD, Py-ART (V1), LROSE, Pyrad (V1 and (partially) V2)

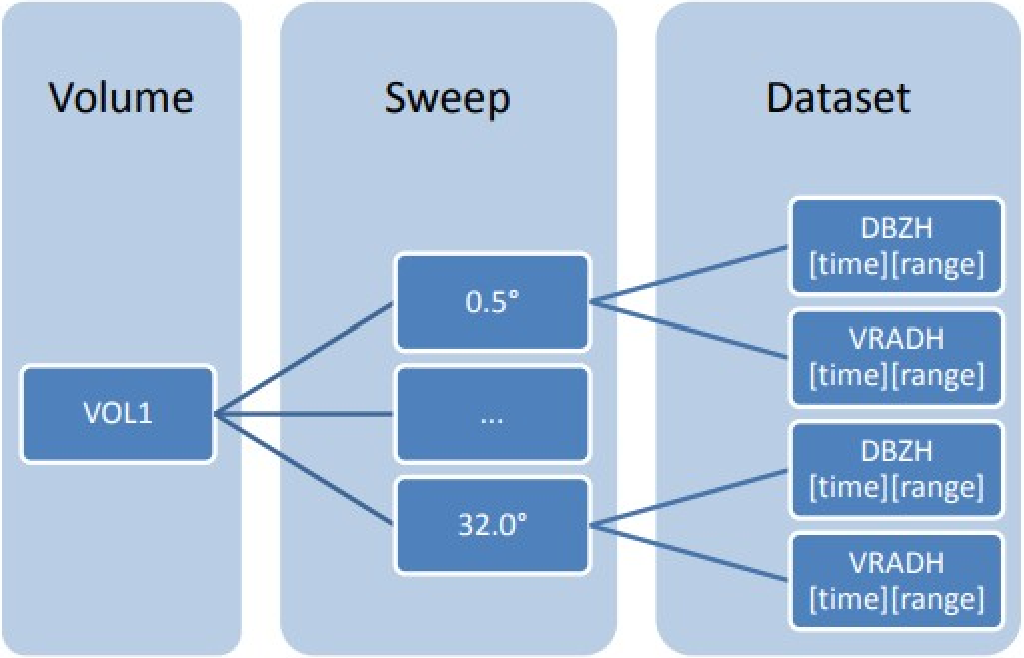
https://ncar.github.io/CfRadial/

CFRadial

CFRadial 1



CFRadial 2



ODIM_H5

OPERA Data Information Model for HDF5

Based on HDF5

Maintained by the OPERA programme of EUMETNET

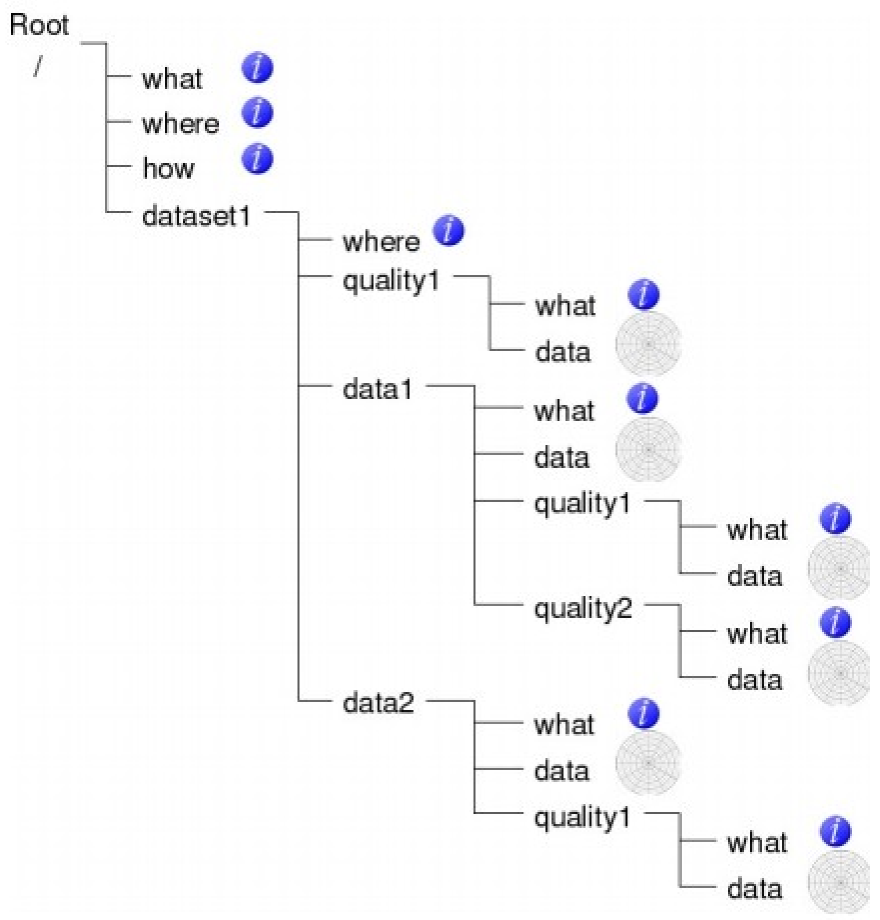
European standard for the exchange of radar data

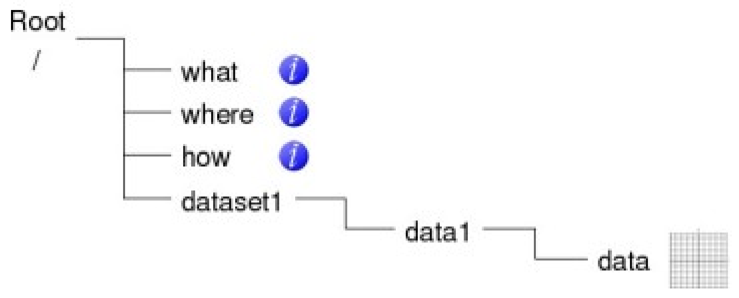
Defined for exchange of polar AND Cartesian data

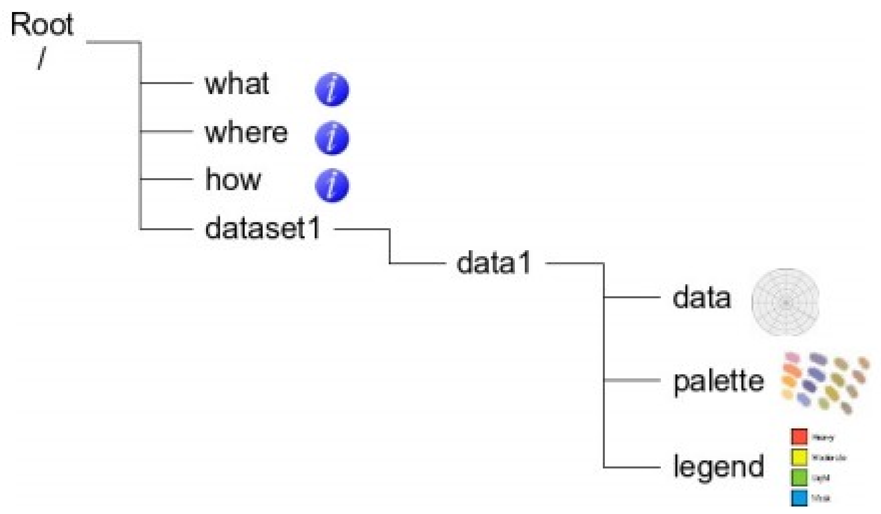
Uses groups

Readers: wradlib, Py-ART, Pyrad, BALTRAD, LROSE

ODIM


Root
/
what
where
how
dataset1
where
quality1
what
data
data1
what
data
quality1
what
data
quality2
what
data
data2
what
data
quality1
what
data


Root
/
what
where
how
dataset1
data1
data


Root
/
what
where
how
dataset1
data1
data
palette
legend

NEXRAD-AR2 data

Data from the US Weather radar network

NEXRAD Level-II (Base) Data: reflectivity, mean radial velocity, spectrum width, (differential reflectivity, correlation coefficient, differential phase)

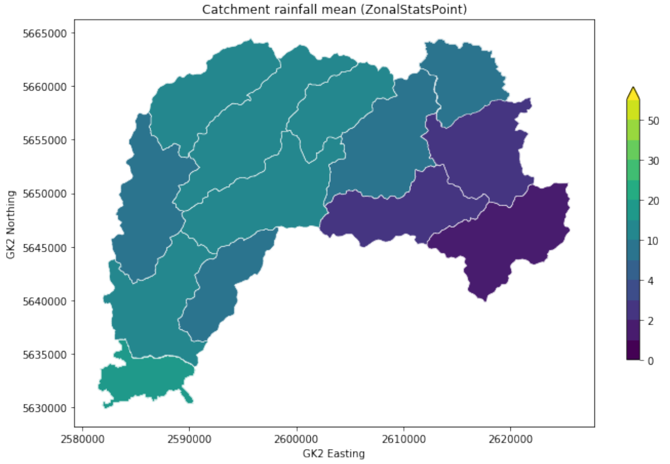
NEXRAD Level-III Products: More than 75 products

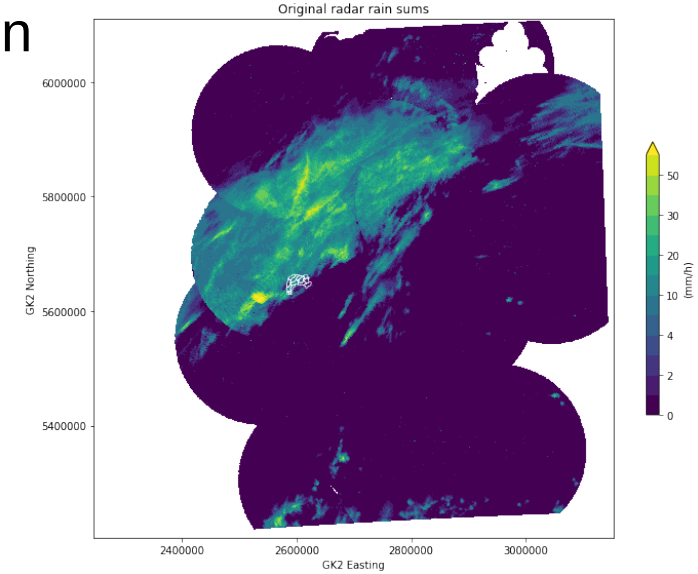
Readers: wradlib, Py-ART, Pyrad (level II), LROSE

wradlib

Philosophy : Keep the magic to the minimum (and let the user decide)

- One of the oldest packages (2011)
- Open platform for collaborative development of algorithms
- Python-based
- Linux/Windows/Mac
- Flat data model that allows maximum flexibility to interact with the data. xarray readers available
- Comprehensively addresses the full radar processing chain
- Mainly geared to interactive use in research but used in operations too
- Easy to install (PyPI, conda, Docker Hub)
- https://wradlib.org


Catchment rainfall mean (ZonalStatsPoint)


Original radar rain sums

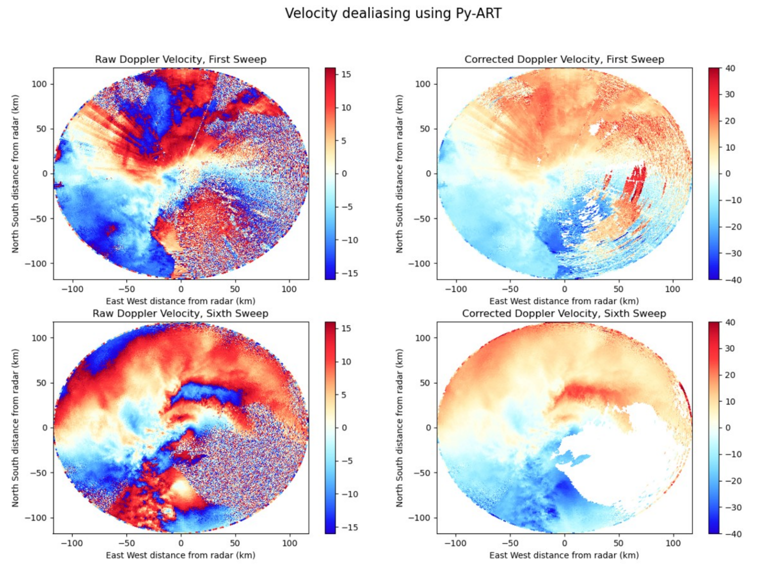
Wradlib functionality

Module	Functionality	Comments
adjust	Gage adjustment	
atten	Attenuation Correction	Hitschfeld, PIA from KDP
classify	Hydrometeor Classification	Fuzzy logic classifier
clutter	Clutter Identification	
comp	Composition	Multiple Radar compositing
dp	Dual-Pol and Differential Phase	KDP retrieval, texture computation, de-polarization ratio computation
georef	Georeferencing	
io	Raw data I/O	Many readers, some put data in xarrays
ipol	Interpolation	Interpolation functions
qual	Data Quality	Beam blockage calculations, Bright band contamination
trafo	Data Transformation	e.g. linear to dB
util	Utility Functions	Despeckle, derivate, etc.
verify	Verification	Comparison between radar-base precipitation and ground truth
vis	Visualization	PPI, RHI, etc.
vpr	Vertical Profile of Reflectivity	Create and work with 3D grids
zonalstats	Zonal Statistics	
zr	Z-R Conversions	

DOE-ARM Py-ART

Philosophy : It’s all about the data model

- Created in the context of the ARM programme (2013)
- Open platform for collaborative development of algorithms
- Mostly Python-based (some modules in C, Cython and FORTRAN)
- Linux/Windows/Mac
- Core : Radar object that structures the radar data and metadata mirroring the C/F Radial standard
- Limited scope. Base block to built upon
- Rich ecosystem of packages :
  - ART-VIEW, PyTDA, PyDDA, TINT, Pyrad...
- Easy to install (PyPI, conda)
- https://arm-doe.github.io/pyart/


Velocity dealiasing using Py-ART

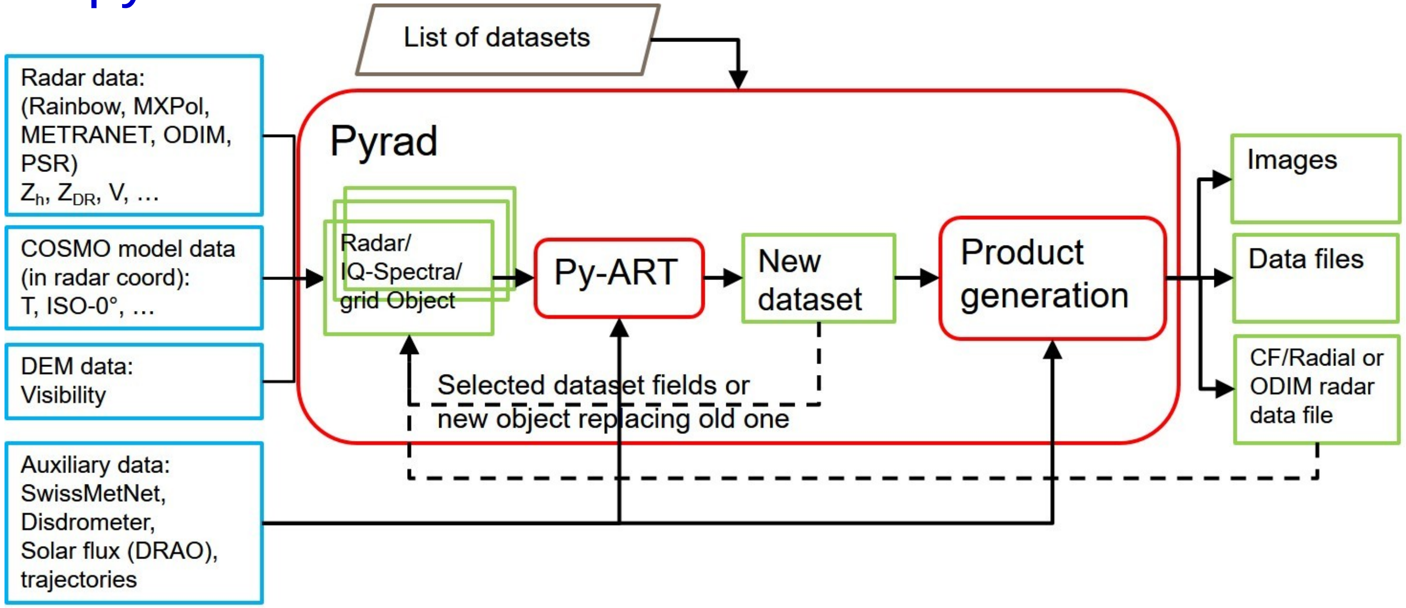
Py-ART functionality

Category	Module	Functionality
Reading/writing	io and aux_io	Reading and writing gridded and polar data
Correcting radar fields	correct	Attenuation, bias, noise in RhoHV, Doppler velocity de-aliasing, PhiDP processing, Despeckling and clutter filtering
Retrieving	retrieve	Secondary moments (e.g. Noise, SNR, CDR, KDP) VAD, hydrometeor classification, RR
Plotting	graph	Horizontal grid plotting, 3D grid slice plotting, PPI, RHI, ray, Pseudo-PPI,
Compositing	map	
Filtering	filters	According to temperature, position with respect to iso-0°, moments, moments and textures

Pyrad

Philosophy : Flexible and replicable data processing chains with no programming

- Initially developed at MeteoSwiss. Now shared development between MeteoSwiss and Météo-France
- Python-based weather radar data processing framework capable of operating in **real time** or **off-line**
- Core based on **ARM-DOE Py-ART** (Pyrad major contributor)
- Easy to install (PyPI, conda)
- https://github.com/MeteoSwiss/pyrad


List of datasets
Radar data: (Rainbow, MXPol, METRANET, ODIM, PSR) Z_h, Z_{DR}, V, …
COSMO model data (in radar coord): T, ISO-0°, …
DEM data: Visibility
Auxiliary data: SwissMetNet, Disdrometer, Solar flux (DRAO), trajectories
Pyrad
Radar/ IQ-Spectra/ grid Object
Py-ART
New dataset
Product generation
Selected dataset fields or new object replacing old one
Images
Data files
CF/Radial or ODIM radar data file

Pyrad capabilities (volume data)

18 dataset groups, 80+ different processings, 40+ different products

Echo classification & filtering	Ψ_{DP} processing and attenuation correction	Monitoring, calibration & noise corr	Retrievals	Special functions
• Clutter ID and filtering • SNR filter • Visibility filter • Outlier filter • Hydrometeor classification (semi-supervised cluster)	• Φ_{DP0} correction • Φ_{DP} smoothing (1, 2 windows) • Least square K_{DP} retrieval (1, 2 wind) • Φ_{DP}, K_{DP} retrieval Maesaka • Linear Programming Φ_{DP}, K_{DP} retrieval • Φ_{DP}, K_{DP} retrieval Vulpiani • ZPhi & PhiLinear att corr	• Bias correction • ρ_{HV} noise correction • ρ_{HV} in rain estimation • Z_{dr} in moderate rain estimation • Z_{dr} in snow estimation • Z_{dr} in birdbath scan • Self-consistency Z_h bias estimation • Time averaging • Ground clutter monitoring • Radar inter-comparison • Sun signal monitoring	• Signal power • SNR • Radial noise power • Clutter Correction ratio • L parameter • CDR • RCS • Melting layer detection • Wind velocity • Wind shear • Various rainrate algorithms • Rainfall accumulation • Velocity dealiasing • Bird density • Velocity dealias • VAD	• Volume cutting • Gridding • Trajectory • Point of interest • Data gridding • Cumulative distribution functions • Quasi Vertical Profiles • Temporal statistics • Fixed range/fixed range span data

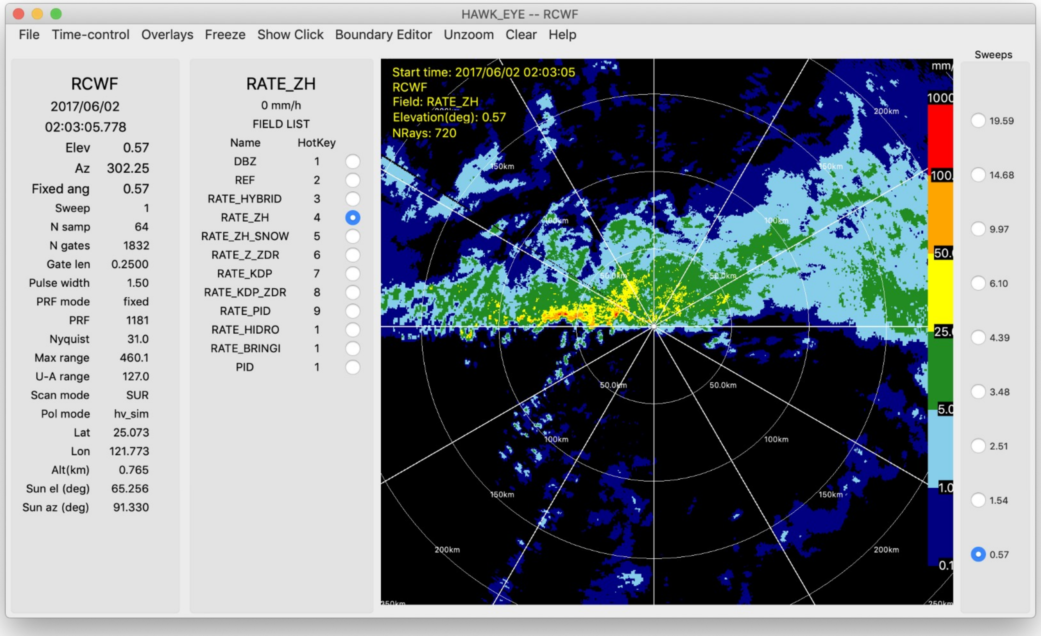
Pyrad capabilities and products (other data)

	IQ data	Spectral data	Gridded data
Capabilities	• Computation of polarimetric and Doppler moments (lag-N estimators) • Transformation into spectral data (FFT)	• 0-Doppler filtering • sRhoHV filtering • Spectral noise filtering (Spectral clipping) • Point of interest • Region of interest • Spectral power, phase, reflectivity, ZDR, RhoHV, PhiDP • Noise estimation • Computation of polarimetric and Doppler moments • Transformation into IQ data (inverse FFT)	• Point of interest • Region of interest • Temporal statistics
Products	• Range/Angle/Time-Doppler plot • Save data in netcdf		• Mapped image/contour • Cross-sections • Histograms • Save data in netcdf

LROSE

Philosophy : High quality building blocks for complex workflows

- Based on legacy of NCAR and CSU tools
- Fast native cross-platform applications
- Mostly C++
- Linux/Mac/partially Windows
- Many stand-alone tools
- Stores data in CF/Radial
- http://lrose.net/



RCWF
2017/06/02
02:03:05.778

Elev	0.57
Az	302.25
Fixed ang	0.57
Sweep	1
N samp	64
N gates	1832
Gate len	0.2500
Pulse width	1.50
PRF mode	fixed
PRF	1181
Nyquist	31.0
Max range	460.1
U-A range	127.0
Scan mode	SUR
Pol mode	hv_sim
Lat	25.073
Lon	121.773
Alt(km)	0.765
Sun el (deg)	65.256
Sun az (deg)	91.330

RATE_ZH
0 mm/h
FIELD LIST

Name	HotKey	
DBZ	1	○
REF	2	○
RATE_HYBRID	3	○
RATE_ZH	4	●
RATE_ZH_SNOW	5	○
RATE_Z_ZDR	6	○
RATE_KDP	7	○
RATE_KDP_ZDR	8	○
RATE_PID	9	○
RATE_HIDRO	1	○
RATE_BRINGI	1	○
PID	1	○

Start time: 2017/06/02 02:03:05
RCWF
Field: RATE_ZH
Elevation(deg): 0.57
NRays: 720

Sweeps
○ 19.59
○ 14.68
○ 9.97
○ 6.10
○ 4.39
○ 3.48
○ 2.51
○ 1.54
● 0.57

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

METEO FRANCE

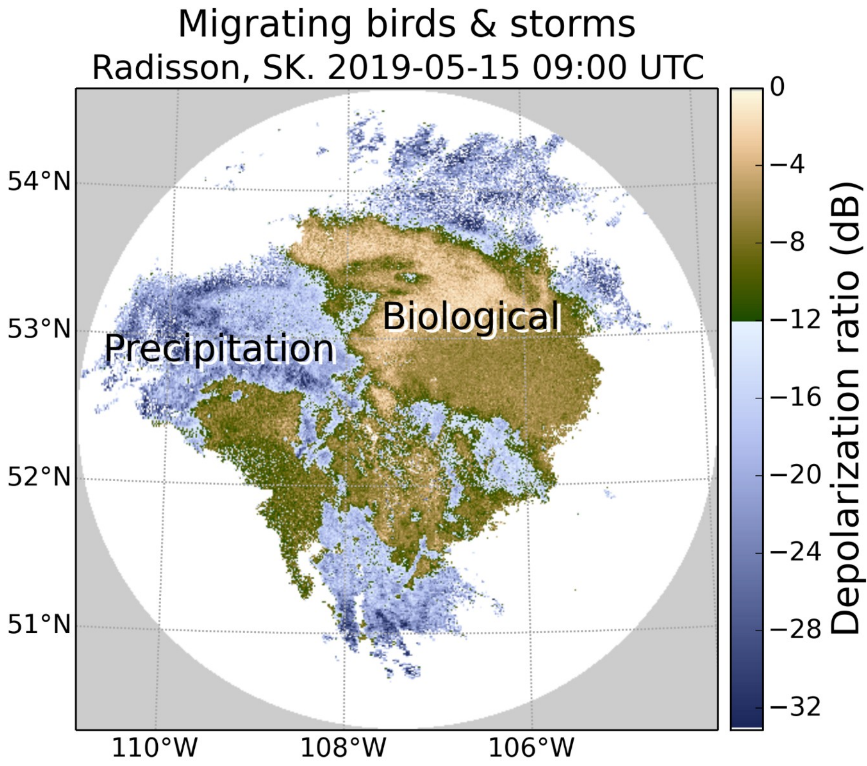
LROSE tools

Convert	RadxPrint: Print file properties and determine if it is supported by Radx RadxConvert and RadxBufr: Conversion from 25 formats to CfRadial
Display	HawkEye
Quality Control	14 tools: compare merge and filter fields Detect sun hits and analyse them
Grid	Radx2Grid
Echo	23 tools: KDP and Attenuation Particle Identification, hydrometeor classification Rain rate and rainfall accumulation Beam blockage estimation Convective/stratiform Mesocyclones Refractivity and moisture Titan (Thunderstorm Identification, Tracking, Analysis and Nowcasting)
Wind	5 tools: VAD Multi-Doppler retrieval Vortex Optical Flow

BALTRAD

Philosophy : Advanced Weather Radar Network

- Heritage from the Nordic Network NORDRAD. Partly funded by the EU. BALTRAD and BALTRAD+ projects (2009-2014). 13 partners in 10 countries
- Real-time data exchange and data processing
- Sub-packages written in different languages
  - Data exchange: JAVA
  - Data processing: C and Python
- Linux/Mac
- Distributed networking, partners exchange polar data and process them using a common toolbox
- Uses ODIM-H5
- Docu: https://baltrad.github.io/
- Code: https://github.com/baltrad



BALTRAD packages

Package	Environment	Description
baltrad-db	Python, Java	Database manager subsystem
BaltradDex	Java	Distribution and Exchange subsystem
baltrad_wms	OGC Map Server	Web map services
bbufr	C, Python	BALTRAD interface to EUMETNET OPERA's BUFR Software
beamb	C, Python	Beam blockage correction
beast	Java	Task manager/scheduler subsystem
bRopo	C, Python	Anomaly (non-precipitation echo) detection and removal
GoogleMapsPlugin	Python	Creation of PNG images to use in Google Maps
node-installer	Python	Installation wizard
OdimH5	Java	Data injector using ODIM_H5 and Rainbow file formats
RAVE	C, Python	Product generation framework and toolbox
baltrad_wrwp	C, Python	Wind products
baltrad-ppc	C, Python	Polarimetric processing chain

Final thoughts

- If you use open source :
  - **Acknowledge it** : To get the backing of the institutions that finance the PIs it is important to show that it is used
  - **Contribute back** : reports on bugs, enhancements, even feedback on how you use it are crucial to improve the code (and boost the morale of the developers).
  - **Do not be afraid to contribute** your code even if you are not confident about your programming skills: Your code will be reviewed before merging and you will learn a lot in the process
- Open source should not just be putting your program somewhere in a server. If you create a new project you should be prepared to provide a minimum support (good documentation, code consistency, some level of engagement with the user, etc.)
- **PERSONAL THOUGHT :** We are lucky that the weather radar open source environment is quite rich and mature. Before creating your own project from scratch think whether it can fit in an existing project

GRAZIE !

MERCI!

THANK YOU !

GRÀCIES!