Hydrol. Earth Syst. Sci., 17, 863–871, 2013
www.hydrol-earth-syst-sci.net/17/863/2013/
doi:10.5194/hess-17-863-2013

Hydrology and Earth System Sciences
Open Access

Technical Note: An open source library for processing weather radar data (*wradlib*)

M. Heistermann[1], **S. Jacobi**[1], **and T. Pfaff**[2]

[1]University of Potsdam, Institute of Earth and Environmental Sciences, Karl-Liebknecht-Str. 24–25, 14476 Potsdam, Germany
[2]University of Stuttgart, Institut für Wasser- und Umweltsystemmodellierung, Stuttgart, Germany

Correspondence to: M. Heistermann (maik.heistermann@uni-potsdam.de)

Received: 26 October 2012 – Published in Hydrol. Earth Syst. Sci. Discuss.: 2 November 2012
Revised: 18 January 2013 – Accepted: 4 February 2013 – Published: 28 February 2013

Abstract. The potential of weather radar observations for hydrological and meteorological research and applications is undisputed, particularly with increasing world-wide radar coverage. However, several barriers impede the use of weather radar data. These barriers are of both scientific and technical nature. The former refers to inherent measurement errors and artefacts, the latter to aspects such as reading specific data formats, geo-referencing, visualisation. The radar processing library *wradlib* is intended to lower these barriers by providing a free and open source tool for the most important steps in processing weather radar data for hydrometeorological and hydrological applications. Moreover, the community-based development approach of *wradlib* allows scientists to share their knowledge about efficient processing algorithms and to make this knowledge available to the weather radar community in a transparent, structured and well-documented way.

1 Introduction

The potential of weather radar observations for hydrological and meteorological research and applications is undisputed (Krajewski and Smith, 2002). Since the emergence of weather radar technology after World War II, research has strived at tapping this potential, particularly in terms of precipitation monitoring and forecasting with high spatiotemporal resolution and coverage. These features not only make weather radar data useful for severe weather detection and meso-scale flood forecasting, but potentially also for erosion studies, precision agriculture, water harvesting, or urban drainage and reservoir control (see e.g. Cruse et al., 2006; Hardegree et al., 2008; Kraemer and Verworn, 2009). The actual availability of radar coverage is no longer a limiting factor in many regions of the world (Fig. 1).

Nonetheless, many challenges remain, particularly from a hydrological perspective. From a scientific point of view, these challenges particularly arise from the multitude of potential error sources which are typically inhomogeneous in space and time (Harrison et al., 2000; Germann et al., 2006). These error sources can introduce severe bias in quantitative hydrological studies, e.g. (eco-)hydrological modelling and forecasting. Hence, it requires the combination of state-of-the-art correction algorithms to make the data useful for hydrological applications. Although many algorithms have been published in peer-reviewed journals, the level of documentation often is not sufficient for a straightforward re-implementation.

From a technical point of view, other barriers exist that prevent hydrologists (and other users) from working with weather radar data; the first being a multitude of different file formats for data storage and exchange. Although the OPERA project (http://www.knmi.nl/opera, URL will change to http://www.eumetnet.eu/opera in the future) has taken steps towards harmonizing the data exchange in Europe, different dialects still exist in addition to a large variety of legacy formats. Many radar data come in complex, and sometimes proprietary binary formats which require a lot of expertise in handling, or even the use of commercial software products. Other technical barriers are a lack of experience in working with polar coordinates and in georeferencing three dimensional scan data, as well as the lack of

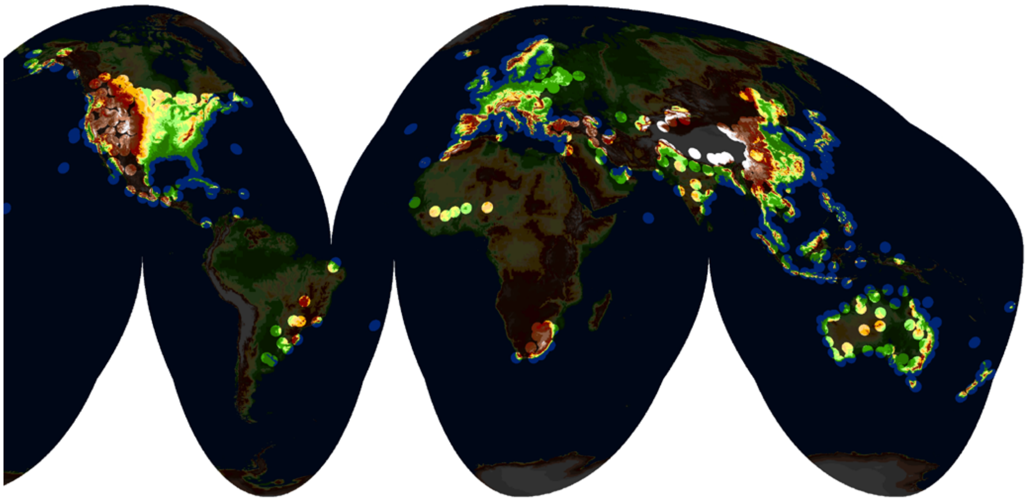

Fig. 1. Current weather radar coverage provided by national weather services (in Goode Homolosine projection). For computing the area illuminated by terrestrial weather radars, we assumed a maximum range of 200 km per radar device. The underlying database was established based on a web search of national weather services' web pages, web documents, and the WMO radar database http://wwr.dmi.gov.tr (which is quite incomplete, though, in comparison to the results shown in this figure). The sources of information are listed in the Supplement Sect. 1. The figure shows that almost complete coverage is achieved for North and Central America, as well as Western and Central Europe. Good coverage is also available for the Middle East, South, Southeast and East Asia, and Australia. Large parts of Africa and South America remain uncovered, yet, as well as vast parts of Russia (although a new network with about 140 Doppler radars is planned to be established in Russia by 2018).

out-of-the-box spatial visualisation tools outside GIS working environments.

Furthermore, specific users might have specific requirements to data processing. For example, one user might favour aggressive clutter elimination in order to assimilate the product to a numerical weather prediction (NWP) model (see e.g. Fornasiero et al., 2006), while another user might prefer a more conservative product in order to detect small scale convective features. A single product from a radar operator, even if created with the best methods currently available, will never be able to accommodate all these needs simultaneously. For many potential users, this trickles down to a *take-it-or-leave-it* decision. The Hydro-NEXRAD initiative (Krajewski et al., 2011) aims to address this problem by providing a web interface through which the user can specify a limited set of processing options; however, only for the US weather radar network NEXRAD.

Researchers themselves typically hesitate to commit to proprietary software products if they do not have the perspective to modify the software code according to their own best knowledge – which would basically require an open source concept. Until recently, no open source software for weather radar data processing has been available which lead many researchers to develop their own software code. This option often implies reinventing the wheel: writing routines for data I/O, error corrections, geo-referencing and visualisation, trying to find relevant literature and to extract algorithms from publications, which, often enough, do not offer all details required for implementation. This takes a lot of time and effort, which could be used much more efficiently if standard algorithms were available in a well- documented and easy to use manner.

In 2012, the situation changed as even two free and open source software products for weather radar data processing were released, one being the BALTRAD software (http://git.baltrad.eu) and the other being the *wradlib* library (http://wradlib.bitbucket.org). This paper aims at introducing the conceptual and technical framework of the *wradlib* library. In Sect. 2, we will present the design and development goals of *wradlib*. Section 3 will illustrate the functionality of *wradlib* based on a set of examples. Section 4 will outline the main future development challenges.

2 Design and development goals

2.1 *wradlib* is a library

It is important to understand *wradlib* not as an application, but as a library. It is designed to provide the tools required for building complete radar processing chains. It allows users to combine selected functionalities in order to create custom-tailored workflows. This is only possible because the library design is strictly modular. One of the original motivations of the development team was to offer a toolbox in order to make radar data usable for hydrological applications. However, the scope of *wradlib* is certainly not restricted to hydrology.

2.2 *wradlib* is easy to use

Using Python as the main implementation language substantially lowers the barrier to create applications based on *wradlib*. Python (http://python.org) is an open source, high level interpreted language with an extensive built-in standard library. It has a clear syntax, is well documented and easy to learn. Using Python also allows embedding lower-level programming languages such as C/C++ or FORTRAN. This option can be used to include external code, but also to optimise the performance of selected functions.

2.3 *wradlib* is platform independent

Building on top of Python makes *wradlib* platform-independent, provided that package dependencies can be satisfied on the respective platform as well. This is straightforward in Linux environments. On windows machines, most dependencies can conveniently be resolved by installing free scientific Python distributions such as Python(x, y) or the Enthought Python Distribution (EPD).

2.4 *wradlib* is a community project

wradlib is consequently open source and uses the permissive MIT License to ensure that students, researchers or companies who contributed to *wradlib* may continue to use their work even for commercial purposes. The open source license model not only allows free access to all of *wradlib*'s processing capabilities, but is also a prerequisite to put into action the concept of *wradlib* as a community-based development and exchange platform. This means that researchers and developers should understand *wradlib* as an opportunity to share their knowledge with other developers, and on the other hand to benefit from user feedback. Apart from the permissive licence model and platform independence, the following two concepts are assumed to foster this kind of exchange:

– *Distributed Version Control*: version control allows developers to track changes and to integrate code from several developers almost automatically. Distributed version control systems (DVCS) add a new level of cooperation as each developer may work with his or her own repository, having full access to the whole history, merging in changes from others when necessary, and pushing out own changes once these are deemed ready for application. *wradlib* uses the platform independent DVCS Mercurial (http://mercurial.selenic.com). The source code is hosted at Bitbucket (http://bitbucket.org/wradlib/wradlib). Apart from hosting, Bitbucket offers additional services useful for collaboration and community-building like issue-tracking, RSS feeds and e-mail notifications, or interactive comments on changesets. Developers without writing permissions on the main repository can integrate their own developments into the main *wradlib* branch by using the so-called Pull Request mechanism. Corresponding guidance is provided in the tutorials section of the *wradlib* documentation pages.

– *Documentation*: it is the belief of the authors that the paramount criterion for the usability of a library is its documentation. Documentation should not only enable the potential user to apply the software, but it should also contain key technical and scientific background information on the functions employed. Documentation is thus a requirement for efficient development cooperation, and is the main interface between developers and users. This includes documentation of the source code and the application programming interface (API), but also tutorials and code recipes to guide beginners on their first steps as well as to explain more complex applications, which cannot be covered by the individual function or class documentation. *wradlib* makes extensive use of Python's *docstring* concept to keep the library documentation as close to the actual code as possible. The docstrings are automatically extracted, incorporated in higher level documents, and finally converted into presentation formats like html or LaTeX. The resulting documentation is made available for download or can be accessed online (http://wradlib.bitbucket.org).

Most of these concepts should be taken for granted in scientific software development; however, they are not. This is why the significance is stressed so much in this context.

2.5 *wradlib* is interactive

wradlib enables interactive as well as presentation quality visualisation of data, with and without spatial contexts, and in the same environment that created the data. Together with the interactive features readily available through the Python interpreter or the IPython shell, *wradlib* may also become a valuable tool for teaching weather radar related topics.

3 Features by example

As pointed out above, *wradlib* covers many aspects of a typical radar data processing chain, with differing levels of maturity. While some modules are still experimental, others are already robust enough for operational environments (see Table 1). A full library reference is available in the *wradlib* documentation at http://wradlib.bitbucket.org/reference.html.

In the following, we will exemplify some of the features contained in *wradlib*. The code as well as example data sets are contained in the *wradlib* distribution under the *examples* directory.

3.1 Reading common radar data formats

We already mentioned cryptic binary formats as a major barrier for radar data users. *wradlib* supports a number of

Table 1. Modules in *wradlib*.

Module	General Description	Examples
`io`	Data import and export	German Weather Service DX and RADOLAN formats EDGE™ export files by EEC OPERA-conform hdf5 and BUFR files
`clutter`	Clutter identification	Acc. to Gabella and Notarpietro (2002) Based on statistics of accumulated precipitation
`atten`	Attenuation correction	Acc. to Hitschfeld and Bordan (1954) Acc. to Kraemer (2008)
`vpr`	Vertical data analyses	Data interpolation to 3-D Cartesian coordinates CAPPI and Pseudo-CAPPI
`trafo`	Unit transformations	dBZ to Z mm h^{-1} to mm
`zr`	Conversions	Reflectivity to rainfall rate Rainfall rate to reflectivity
`georef`	Spatial transformation and geo-referencing	Polar coordinates to geographic coordinates Geographic coordinates to Cartesian coordinates
`ipol`	Spatial interpolation	Nearest Neighbours Linear Interpolation Inverse Distance Weighting
`qual`	Measurement quality indicators	Height of radar beam according to Collier (1996) Pulse volume
`comp`	Building composites from multiple radar locations	Interpolation of polar data to common grids Combining multiple data sources in overlapping regions based on quality information
`adjust`	Rain gage adjustment	Based on a multiplicative error model (Brandes, 1975) Based on an additive error model Mixed additive and multiplicative error model
verify	Verification	Computation of diverse quality criteria based on comparison with rain gage observations
`vis`	Plotting and mapping	Plan Position Indicator Range Height Indicator Volumetric Slice Plot
`util`	Utility functions	Aggregation and interpolation of time series data

data formats mainly used in Europe, but also world-wide. At the moment, these formats include the OPERA BUFR (Paulitsch et al., 2010) and hdf5 formats (Michelson et al., 2011), NetCDF files generated by the commercial EDGE software, hdf5 files generated by the commercial GAMIC software, as well as the German Weather Services quantitative local scan format (DX) and the quantitative composite format (RADOLAN, see German Weather Service, 2004). Other formats will be implemented dependent on demand and availability of documentation. *wradlib* deals with data in both polar and Cartesian coordinate systems.

The basic data structure used by *wradlib* is a multi-dimensional numerical array (the *numpy.ndarray*). It is a very flexible and powerful container for homogeneous data, and has become the de-facto standard for most scientific Python packages. The *wradlib.io* module provides functions that load data from files of different formats into numpy.ndarrays. Metadata, if available, are returned as so-called dictionary objects which can be browsed and inspected using keywords. The level and detail of meta-information depends on the data source.

The following example (see Fig. 2) shows how a local plan position indicator (PPI) in polar coordinates from the German Weather Service's C-band radar Tuerkheim is loaded and inspected with a quick diagnostic plot.

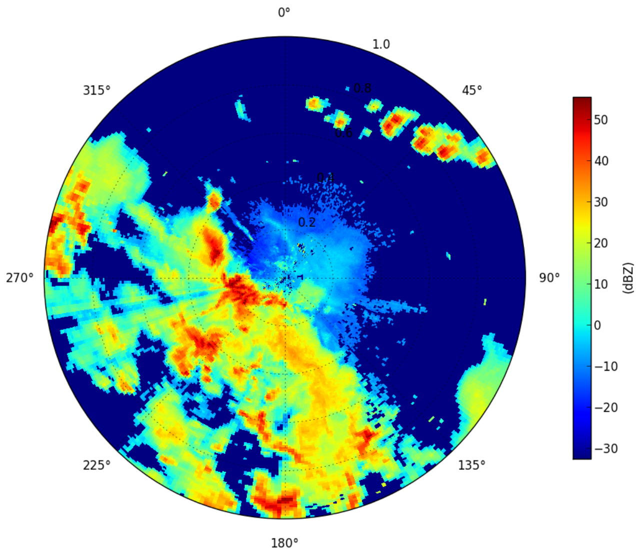

Fig. 2. Simple diagnostic plot of polar reflectivity.

```
Example 1:
>>> import wradlib
>>> data, metadata=wradlib.io.readDX('sample.dx')
>>> wradlib.vis.polar_plot(data)
```

3.2 Error correction

There are several modules dedicated to the correction of the major sources of error in weather radar data. *wradlib.clutter* implements functions for clutter detection, *wradlib.atten* and *wradlib.vpr* deal with attenuation and errors due to the vertical profile of reflectivity (algorithms for VPR correction are currently underway). In the following example, a map of clutter signals is identified by using the filter described in Gabella and Notarpietro (2002).

```
Example 2:
>>> clutter=wradlib.clutter.filter_gabella
        (data, tr1=12, n_p=6, tr2=1.1)
```

The behaviour of this filter can be adjusted to the user's needs. In this example the three parameters of the filter are set explicitly. If they were not given default values (usually those presented in the respective publication) would have been used.

The Boolean map returned by the function can then be used to correct the original data, for example by interpolation from surrounding radar bins:

```
Example 3:
>>> corrected=wradlib.ipol.interpolate_polar
        (data, clutter)
```

For attenuation correction several implementations of the recursive methods presented by Nicol and Austin (2003) and Kraemer (2008) exist. An implementation of the method presented by Marzoug and Amayenc (1994) is underway as well as methods addressing the vertical profile of reflectivity.

3.3 Data transformation

The *wradlib.trafo* module contains utility functions to do common unit transformations like converting from and to decibel or transform rainfall intensities to depths depending on the integration interval. Due to the importance and the diversity of algorithms for Z–R relations, corresponding functions are collected in the *wradlib.zr* module. Currently, the standard power-law relation and the 3-part relation currently in operational use by the German Weather Service according to German Weather Service (2004) are implemented. For instance, the clutter corrected reflectivity (in dBZ) from Example 3 can be converted to a 5 min (300 s) rainfall depth using the following code sequence:

```
Example 4:
>>> Z=wradlib.trafo.idecibel(corrected)
>>> R=wradlib.zr.z2r(Z)
>>> depth=wradlib.trafo.r2depth(R, 300)
```

An additional configuration can be passed to many functions using Python's keyword mechanism. By default *wradlib.zr.z2r* uses the parameters $a = 200$ and $b = 1.6$.

3.4 Geo-referencing

Geo-referencing includes the conversion from polar coordinates to geographical coordinates, and the projection of geographical coordinates to Cartesian coordinate systems such as Universal Transverse Mercator (UTM), Gauss–Krueger, or Azimuthal Equidistant. This functionality is particularly important since the only geographical reference typically provided with radar data is the location of the radar device. The radar location (in terms of latitude, longitude, and altitude) together with the azimuth, range and the elevation angle of the radar beam allows the computation of geographical coordinates and the height of the radar beam (line 4 of example 5) as well as the subsequent projection to a Cartesian reference system (lines 5–6 of example 5), in this case Gauss–Krueger Zone 3:

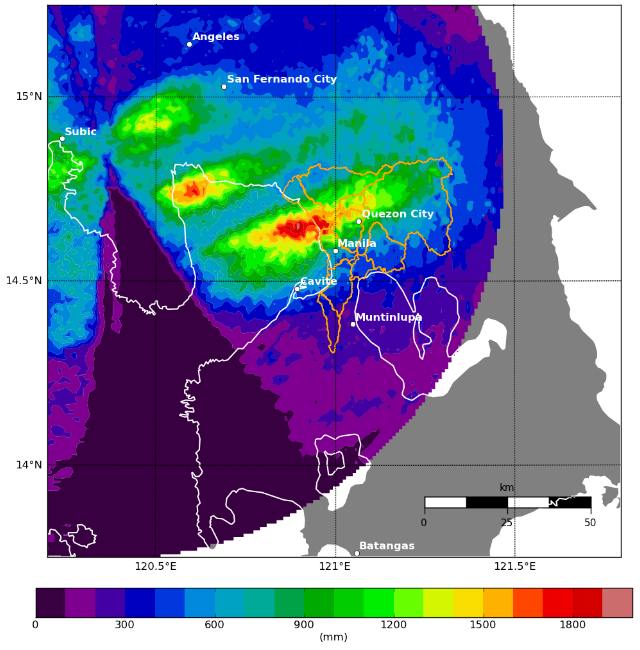

Fig. 3. Rainfall accumulation for Metropolitan Manila region, Philippines, from 6–9 August 2012, based on the S-band radar located near Subic city, using wradlib modules io (reading NetCDF data), georef (geo-referencing), clutter (clutter identification), ipol (gap filling), adjust (for mean field bias adjustment), and vis (for plotting over a map). The orange lines indicate watersheds draining to Manila Metropolitan region. Note that the southern sectors of the radar exhibit significant beam shielding caused by Mount Natib, a volcano and caldera complex located in the province of Bataan. Please note that a corresponding code recipe for this figure has not been added to the wradlib documentation page because the code is part of a more complex application built on wradlib.

```
Example 5:
>>> range=100000.
>>> azimuth=45.
>>> elev=0.5
>>> radar_location=(48.5861, 9.7839, 500.)
>>> lat, lon, alt=wradlib.georef.polar2latlonalt
    (range, azimuth, elev, radar_location)
>>> gk3=wradlib.georef.create_projstr("gk", zone=3)
>>> x, y=wradlib.georef.project(lat, lon, gk3)
```

Geo-referencing also allows plotting radar data over a map, including arbitrary, user-defined geographical features such as coastlines, borders, cities or catchments, using the *wradlib.vis* module. A typical result of such a procedure is shown in Fig. 3, containing a four day rainfall accumulation for the Metropolitan Manila region in the Philippines for the disastrous rainfall event in August 2012.

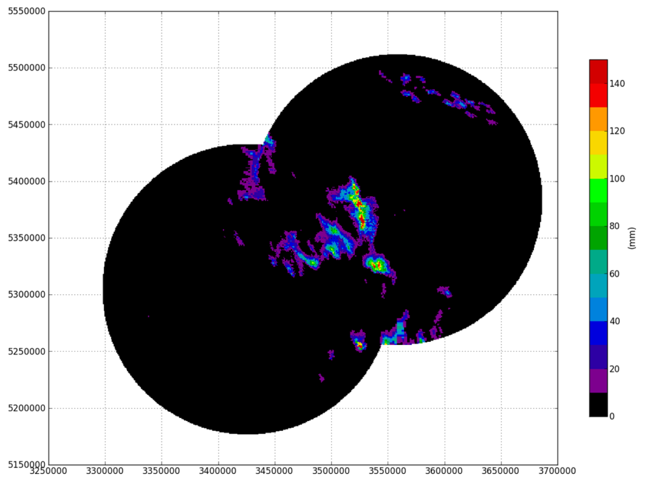

Fig. 4. Composite of a two hours event accumulation of clutter and attenuation corrected data for 2 June 2008, 16:00–18:00 UTC, including the German Weather Service's C-band radars Tuerkheim and Feldberg (5 min scan interval, 1° angular resolution, 128 km range, and 1 km range resolution). This event caused a disastrous flash flood in the central map region. Get details about the underlying code at http://wradlib.bitbucket.org/recipes.html. For two hours of data, the complete workflow takes about 20 s of computation time (on a 32bit Windows machine, using one out of eight threads of an Intel 2.93 GHz Quad-Core processor).

3.5 Interpolation and composition

wradlib provides nearest neighbour, inverse distance weighting and linear interpolation. Various Kriging methods are currently being implemented. All methods have a unified interface so that only minimal changes are necessary to switch between different approaches. The interpolation methods are used by several routines like infilling of missing or flagged data, or the transfer from polar to Cartesian representations.

Combining data from several radar locations on one common grid is usually referred to as *composition*. Creating composites is particularly challenging in areas where several radar circles overlap. In these regions of overlap, *wradlib* applies criteria related to data quality in order to weight different radar data sources. Users are free to define quality criteria; however, the *wradlib.qual* module supports the evaluation of radar data quality for different criteria such as beam height over ground, or pulse volume. Figure 4 shows the result of a composition based on a weighted combination, using the sampling volume as a quality criterion.

3.6 Gage adjustment

Using rain gage observations to adjust the radar measurements has been shown to substantially reduce the error of radar-based rainfall estimates (Goudenhoofdt and Delobbe, 2009). The procedure of gage adjustment is considered indispensable for hydrological applications (Heistermann and

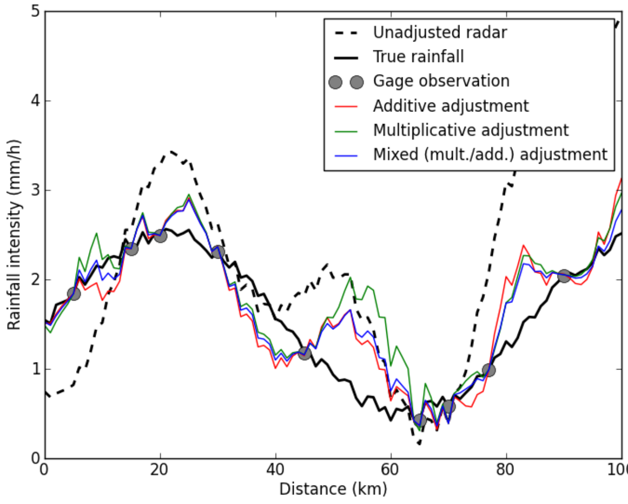

Fig. 5. A 1-dimensional example with synthetic data for illustrating different adjustment approaches in wradlib. The true rainfall was generated by combining a sine curve with white noise. Contaminating the true rainfall with spatially variable additive and multiplicative error terms yields the unadjusted radar observation. Three adjustment methods are then applied via wradlib.adjust – using the gage observations (grey circles) which are random samples of the true rainfall. See Fig. 6 for a verification of this example adjustment.

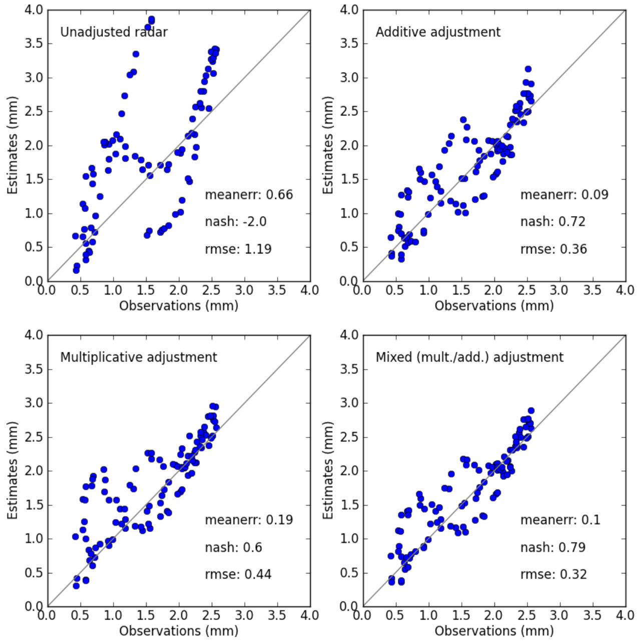

Fig. 6. Verification of the adjustment techniques, using the example shown in Fig. 5. The scatter plot and the error metrics were created using wradlib.verify. The metrics are computed by comparing the unadjusted and the adjusted radar observations against the synthetic true rainfall (just a small selection of error metrics is shown here).

Kneis, 2011). The *wradlib.adjust* module provides different adjustment techniques in order to identify the most effective approach for a specific region and application. At the moment, the implemented adjustment approaches mainly differ with respect to the error model (additive, multiplicative, and a mixed multiplicative and additive approach), and whether the error is assumed to be spatially homogeneous or not. An illustrative one-dimensional example fo different adjustment techniques is given in Fig. 5.

3.7 Verification

Typically, radar-based precipitation estimation and the effectiveness of the underlying correction and adjustment methods are verified by comparing the results against rain gage observations on the ground (Kneis and Heistermann, 2009). This can be done by cross-validation (in case rain gages were already used for gage adjustment), or by using a set of independent, high-quality precipitation measurements on the ground. This procedure has been criticized due to the limited spatial representativeness of point observations and the problems resulting from a comparison to the volume integrated radar observation (Einfalt et al., 2004; Germann and Joss, 2004; Kitchen and Blackall, 1992; Michelson and Koistinen, 2000; Seed et al., 1996). Indeed, alternatives have been proposed, e.g. based on the hydrological catchment response (Heistermann and Kneis, 2011). Nonetheless, the use of rain gage observations remains the main verification strategy. Therefore, *wradlib.verify* provides procedures not only to extract the radar values at specific gage locations, but also a set of error metrics which are computed from gage observations and the corresponding radar-based precipitation estimates (including standard metrics such as RMSE, mean error and Nash–Sutcliffe efficiency). *wradlib.adjust* also contains a standard procedure for leave-one-out cross validation which is applicable to each of the available adjustment method. Figure 6 illustrates the verification output for the one-dimensional example that was provided in Fig. 5.

4 Conclusions and outlook

Currently, *wradlib* is used for different purposes and by different working groups in Germany, but also in Europe and Southeast Asia (mainly in research environments). Although there is not yet a systematic comparison between the BALTRAD and the *wradlib* software, we suggest thinking of *wradlib* as a lightweight, flexible, and experimental counterpart of the BALTRAD software which is geared more towards larger scale operational weather radar networks, and focuses on speed and efficiency rather than flexibility. Nonetheless, *wradlib* also has the potential to be applied in operational environments. For example, it is currently being used for operational rainfall estimation in the Philippine government project NOAH (National Operational Assessment of Hazards). Beyond, features such as interactive use and platform independency might be used to distinguish *wradlib* from BALTRAD.

Apart from the user's perspective, *wradlib* could support radar related research by providing a unified interface to a growing number of standard and research algorithms. Thereby, *wradlib* could enable extensive comparison studies, and increase the reproducibility of research results. In addition, the open source concept allows public code review of any algorithm provided in the library. It already turned out that, together with the community features and the DVCS concept, errors are rapidly identified and fixed.

In the future, the main development efforts in *wradlib* will have to be invested in the integration of algorithms related to the use of

– polarimetric radar moments e.g. the classification of hydrometeors, for the optimisation of Z–R relation parameters, or the correction of clutter, beam blockage and attenuation (e.g. Matrosov et al., 2007; Friedrich et al., 2007; Lang et al., 2009; Cifelli et al., 2011; Tabary et al., 2011; Vulpiani et al., 2012; and Cao et al., 2012, for a general introduction to weather radar polarimetry);

– and 3-dimensional scan information to correct for the effects of non-homogeneous vertical profiles of reflectivity (e.g. Franco et al., 2006; Tabary, 2007; Bouilloud et al., 2010; Kirstetter et al., 2010).

The other *wradlib* modules also are highly dynamic and offer a huge potential for further improvements (e.g. related to interpolation, gage adjustment, quality assessment, and visualisation).

5 How to use wradlib?

– Get the wradlib source distribution at http://www.bitbucket.org/wradlib/wradlib;

– see the wradlib documentation at http://www.wradlib.bitbucket.org;

– get community support by joining the mailing list https://groups.google.com/forum/?fromgroups#!forum/wradlib-users.

Supplementary material related to this article is available online at: http://www.hydrol-earth-syst-sci.net/17/863/2013/hess-17-863-2013-supplement.pdf.

Acknowledgements. The development of *wradlib* was supported by the University of Stuttgart, the University of Potsdam, and the German Federal Ministry for Education and Research (BMBF) through the PROGRESS project. The example radar data shown in this paper were provided by courtesy of the German Weather Service, and the Philippine Atmospheric, Geophysical and Astronomical Services Administration (PAGASA). Until now, the feedback of many *wradlib* users has already allowed for substantial improvements in the software library. Finally, we would like to thank Hidde Leijnse and an anonymous referee for reviewing the manuscript.

Edited by: R. Uijlenhoet

References

Bouilloud, L., Delrieu, G., Boudevillain, B., and Kirstetter, P. E.: Radar rainfall estimation in the context of post-event analysis of flash-flood events, J. Hydrol., 394, 17–27, 2010.

Brandes, E. A.: Optimizing Rainfall Estimates with the Aid of Radar, J. Appl. Meteorol., 14, 1339–1345, 1975.

Cao, Q., Yeary, M. B., Zhang, G.: Efficient Ways to Learn Weather Radar Polarimetry, IEEE T. Educ., 55, 58–68, 2012.

Cifelli, R., Chandrasekar, V., Lim, S., Kennedy, P. C., Wang, Y., and Rutledge, S. A.: A New Dual-Polarization Radar Rainfall Algorithm: Application in Colorado Precipitation Events, J. Atmos. Ocean. Tech., 28, 352–364, 2011.

Collier, C. G.: Applications of weather radar systems: A guide to uses of radar data in meteorology and hydrology, 2nd Edn., John Wiley and Sons, New York, 1996.

Cruse, R., Flanagan, D., Frankenberger, J., Gelder, B., Herzmann, D., James, D., Krajewski, W., Kraszewski, M., Laflen, J., Opsomer, J., and Todey, D.: Daily estimates of rainfall, water runoff, and soil erosion in Iowa, J. Soil Water Conserv., 61, 191–199, 2006.

Einfalt, T., Arnbjerg-Nielsen, K., Golz, C., Jensen, N. E., Quirmbach, M., Vaes, G., and Vieux, B.: Towards a roadmap for use of radar rainfall data in urban drainage, J. Hydrol., 299, 186–202, 2004.

Fornasiero, A., Bech, J., and Alberoni, P. P.: Enhanced radar precipitation estimates using a combined clutter and beam blockage correction technique, Nat. Hazards Earth Syst. Sci., 6, 697–710, doi:10.5194/nhess-6-697-2006, 2006.

Franco, M., Sanchez-Diezma, R., and Sempere-Torres, D.: Improvements in weather radar rain rate estimates using a method for identifying the vertical profile of reflectivity from volume radar scans, Meteorol. Z., 15, 521–536, 2006.

Friedrich, K., Germann, U., Gourley, J. J., and Tabary, P.: Effects of radar beam shielding on rainfall estimation for the polarimetric C-band radar, J. Atmos. Ocean. Tech., 24, 1839–1859, 2007.

Gabella, M. and Notarpietro, R.: Ground clutter characterization and elimination in mountainous terrain. In Proceedings of ERAD 2002, Delft, The Netherlands, 305–311, available at: http://www.copernicus.org/erad/online/erad-305.pdf, 2002.

Germann, U. and Joss, J.: Operational measurement of precipitation in mountainous terrain, in: Weather Radar: Principles and Advanced Applications, Physics of Earth and Space Environments, edited by: Meischner, P., 52–77, Springer, Berlin, Germany, 2004.

Germann, U., Galli, G., Boscacci, M., and Bolliger, M.: Radar precipitation measurement in a mountainous region, Q. J. Roy. Meteor. Soc., 132, 1669–1692, 2006.

German Weather Service: Projekt RADOLAN – Routineverfahren zur Online-Aneichung der Radarniederschlagsdaten mit

Hilfe von automatischen Bodenniederschlagsstationen, Final Report, Offenbach, Germany, available at: http://www.dwd.de/RADOLAN (last access: February 2013), 2004 (in German).

Goudenhoofdt, E. and Delobbe, L.: Evaluation of radar-gauge merging methods for quantitative precipitation estimates, Hydrol. Earth Syst. Sci., 13, 195–203, doi:10.5194/hess-13-195-2009, 2009.

Hardegree, S. P., Van Vactor, S. S., Levinson, D. H., and Winstra, A. H.: Evaluation of NEXRAD radar precipitation products for natural resource applications, Rangeland Ecol. Manage., 61, 346–353, 2008.

Harrison, D. L., Driscoll, S. J., and Kitchen, M.: Improving precipitation estimates from weather radar using quality control and correction techniques, Meteorol. Appl., 7, 135–144, 2000.

Heistermann M. and Kneis, D.: Benchmarking quantitative precipitation estimation by conceptual rainfall-runoff modeling, Water Res. Resour., 47, W06514, doi:10.1029/2010WR009153, 2011.

Hitschfeld, W. and Bordan, J.: Errors inherent in the radar measurement of rainfall at attenuating wavelengths, J. Meteorol., 11, 58–67, 1954.

Kirstetter, P. E., Andrieu, H., Delrieu, G., and Boudevillain, B.: Identification of Vertical Profiles of Reflectivity for Correction of Volumetric Radar Data Using Rainfall Classification, J. Appl. Meteorol. Clim., 49, 2167–2180, 2010.

Kitchen, M. and Blackall, R. M.: Representativeness errors in comparisons between radar and gauge measurements of rainfall, J. Hydrol., 134, 13–33, 1992.

Kneis, D. and Heistermann, M.: Quality assessment of radar-based precipitation estimates with the example of a small catchment, Hydrol. Wasserbewirts., 53, 160–171, 2009.

Kraemer, S.: Quantitative Radardatenaufbereitung für die Niederschlagsvorhersage und die Siedlungsentwässerung. Mitteilungen Institut für Wasserwirtschaft, Hydrologie und Landwirtschaftlichen Wasserbau Gottfried Wilhelm Leibniz Universität Hannover, Heft 92, ISSN 0343-8090, Hanover, Germany, 2008.

Kraemer, S. and Verworn, H. R.: Improved radar data processing algorithms for quantitative rainfall estimation in real time, Water Sci. Technol., 60, 175–184, 2009.

Krajewski, W. F. and Smith, J. A.: Radar hydrology: rainfall estimation, Adv. Water Res., 25, 1387–1394, 2002.

Krajewski, W. F., Kruger, A., Smith, J. A., Lawrence, R., Gunyon, C., Goska, R., Seo, B. C., Domaszczynski, P., Baeck, M. L., Ramamurthy, M. K., Weber, J., Bradley, A. A., DelGreco, S. A., and Steiner, M.: Towards better utilization of NEXRAD data in hydrology: an overview of Hydro-NEXRAD, J. Hydroinform., 13, 255–266, 2011.

Lang, T. J., Nesbitt, S. W., and Carey, L. D.: On the Correction of Partial Beam Blockage in Polarimetric Radar Data, J. Atmos. Ocean. Tech., 26, 943–957, 2009.

Marzoug, M. and Amayenc, P.: A Class of Single-and Dual-Frequency Algorithms for Rain-Rate Profiling from a Spaceborne Radar. Part I: Principle and Tests from Numerical Simulations, J. Atmos. Ocean. Tech., 11, 1480–1506, 1994.

Matrosov, S. Y., Clark, K. A., and Kingsmill, D. E.: A polarimetric radar approach to identify rain, melting-layer, and snow regions for applying corrections to vertical profiles of reflectivity, J. Appl. Meteorol. Clim., 46, 154–166, 2007.

Michelson, D. B. and Koistinen, J.: Gauge-radar network adjustment for the Baltic Sea Experiment, Phys. Chem. Earth Pt. B, 25, 915–920, 2000.

Michelson, D. B., Lewandowski, R., Szewczykowski, M., and Beekhuis, H.: EUMETNET OPERA weather radar information model for implementation with the HDF5 file format, Version 2.1. OPERA Working Document WD_2008_03, available at: http://www.knmi.nl/opera/opera3/OPERA_2008_03_WP2.1b_ODIM_H5_v2.1.pdf (last access: February 2013), 2011.

Nicol, J. C. and Austin, G. L.: Attenuation correction constraint for single-polarisation weather radar, Meteorol. Appl., 10, 345–354, 2003.

Paulitsch, H., Fuchsberger, J., and Köck, K.: FM94-BUFR Encoding and Decoding Software, User Guidelines Version 1.6 For BUFR Software Version 3, available at: http://www.knmi.nl/opera/bufr/doc/bufr_sw_desc.pdf, 2010.

Seed, A. W., Nicol, J., Austin, G. L., Stow, C. D., and Bradley, S. G.: The impact of radar and raingauge sampling errors when calibrating a weather radar, Meteorol. Appl., 3, 43–52, 1996.

Tabary, P.: The new French operational radar rainfall product. Part I: Methodology, Weather Forecast., 22, 409–427, 2007.

Tabary, P., Boumahmoud, A.-A., Andrieu, H., Thompson, R. J., Illingworth, A. J., Le Bouar, E., and Testud, J.: Evaluation of two “integrated” polarimetric Quantitative Precipitation Estimation (QPE) algorithms at C-band, J. Hydrol., 405, 248–260, 2011.

Vulpiani, G., Montopoli, M., Passeri, L. D., Gioia, A. G., Giordano, P., and Marzano, F. S.: On the Use of Dual-Polarized C-Band Radar for Operational Rainfall Retrieval in Mountainous Areas, J. Appl. Meteorol. Clim., 51, 405–425, 2012.