Submitted to the
17th International Conference on Ground Penetrating Radar,
June 18–21, 2018 in Rapperswil, Switzerland

RGPR - An open-source package to process and visualize GPR data

Emanuel Huber
Applied and Environmental Geology
University of Basel
4056 Basel, Switzerland
emanuel.huber@unibas.ch

Guillaume Hans
FPInnovations,
2665 East Mall, Vancouver, BC
V6T 1Z4, Canada
guillaume.hans@fpinnovations.ca

Abstract—***RGPR, the first ground-penetrating radar (GPR) data processing package written in the R language, is presented. R is a free and open-source high-level programming language for statistical computing. RGPR is built around two main classes to process and visualize GPR data as well as to keep track of the processing steps. Although RGPR is still a work in progress, many of the basic processing methods are already implemented. The package is hosted on GitHub for collaborative development and is easily expandable. Through its openness and the rich R environment, RGPR promotes reproducible GPR research as well as GPR processing learning.***

Keywords—ground-penetrating radar; package; R language; open-source; reproducible research; processing; visualization

I. INTRODUCTION

The processing of ground-penetrating radar (GPR) data is generally an essential step to extract useful information. Because of the similarity between GPR and seismic data, many seismic processing software applications were used in the early 90's to process GPR data. Nowadays, many commercial software applications 100% dedicated to GPR data are available. However, those applications generally propose a limited range of processing methods and/or are dedicated to a very specific usage. We present RGPR, a new open-source GPR data processing package that is written in the R language [1] to allow greater freedom and flexibility for research.

R is an interpreted, high-level programming language for statistical computing and graphics that is freely available under the GNU General Public License and runs on Linux, Windows and MacOS. It is a highly versatile and extensible language to which C, C++ and Fortran code can be linked and run. Furthermore, the R developer community is very active and more than 10'000 packages are hosted on the official global package repository CRAN (Comprehensive R Archive Network, https://cran.r-project.org). To be hosted in the CRAN repository, a package must have all functions, methods and class documented according to the R package documentation standards. The last twenty years, R has strongly gained popularity and was ranked in 2017 as the sixth top programming language by the Institute of Electrical and Electronics Engineers [2].

RGPR was initially developed to compensate for shortcomings of commercial GPR data processing applications (i.e., apply specific signal processing methods, add topographical data to GPR data, align two-dimensional GPR profiles, export high-quality graphics, visualize GPR data in three dimensions). In 2015 the code was organized into a R-package based on the 'S4' system for oriented programming.

The key features of RGPR are (i) RGPR is freely available and open-source, (ii) it depends on the freely available and open-source R environment, (iii) it allows reproducible GPR data processing by means of processing scripts and storage of the processing workflow within every GPR object, (iv) it is easily expandable, and (v) it exports well-designed publication-quality plots.

Although RGPR is still a work in progress, the development version is already available on GitHub [3] under the GPL license. However, this version still presents several limitations, the most notable being that it can only handle data acquired in common-offset, common mid-point (CMP) or wide-angle reflection refraction (WARR) modes. Nevertheless, anyone can have access to the source code as well as to the history of the changes made to the files. Furthermore, anyone with a GitHub account can submit code modifications (e.g., bug fixes, new function) within the git workflow and contribute to improve the package.

In this paper, we first briefly explain how to install RGPR directly from its GitHub repository. Then, we present the structure of the RGPR package and the two main classes it uses. We also describe the main functions already implemented for GPR data analysis, processing, visualization and georeferencing. Finally, we outline the main functionalities that should be added to RGPR in the near future as well as those that would be desirable to have in a longer term.

II. PACKAGE INSTALLATION

RGPR can be installed within the R environment directly from its GitHub repository using the 'install_github()' function from the R-package devtools [4]. The code is documented in the source files and the R-package roxygen2 [5] is used to generate the standard R documentation files. Additional details as well as tutorials are available in the GitHub repository.

III. PACKAGE STRUCTURE

A. General overview

RGPR is built around two main classes, *GPRsurvey* and *GPRvirtual* (Fig. 1) that allow codes and method names to be recycled and the interface for the end-users to be simplified. The same method names can be intuitively used for different classes with specific behavior. For example, the same method 'plot()' can display a map of the GPR measurement lines or the GPR data themselves when applied to an object of the class *GPRsurvey* or *GPRvirtual*, respectively. GPR data carry additional information (e.g., trace position, time signal, antenna separation, time-zero) that can be modified during various processing steps. This additional information along with the GPR data are stored as attributes within the objects and updated by the methods if necessary. Hence, the user does not have to worry about modifying this additional information as data are being processed.

B. The GPRsurvey class

The class *GPRsurvey* is designed to deal with any mapping, georeferencing or topographic correction operation that need to be performed with the GPR data.

An object of the class *GPRsurvey* does only contain a link to the GPR data files as well as information about the GPR data such as trace coordinates. It is further possible to apply batch processing to the GPR data referenced by an object of the class *GPRsurvey*. The processed GPR data are then saved on the disk as temporary files and their links in the object are updated. These data can be saved later by the user as GPR files (see Section IV.D.).

C. The GPRvirtual abstract class

GPRvirtual is an abstract class that has four implementing classes designed for four different types of GPR data: *GPR*, *GPRstack*, *GPRcube*, and *GPRslice* (Fig. 1). Each of these classes possesses, at least, the attributes listed in TABLE I.

The *GPR* class represents two-dimensional GPR data such as common-offset or common mid-point GPR data with their attributes. This is the simplest class that allows its objects to be manipulated as two-dimensional arrays (i.e., matrix) objects. Classical mathematical and matrix operations (e.g., addition, scalar and matrix multiplication, absolute value, logarithm) as well as subsetting work on objects of the *GPR* class.

TABLE I: ATTRIBUTES OF GPRvirtual

Attributes	Type	Description
version	character	version of RGPR in use
data	array (2D/3D)	GPR data (one trace per column)
z	numeric	depth position or sampling time
x	numeric	position of the traces along the survey
dz	numeric	time or depth sampling interval
dx	numeric	spatial trace sampling interval
coord	array	trace coordinates (x,y,z)
rec	array	receiver coordinates (x,y,z)
trans	array	transmitter coordinates (x,y,z)
units	character	time/depth and spatial unit
crs	character	coordinate reference system
time0	numeric	time-zero of every trace
freq	numeric	antenna frequency (in MHz)
antsep	numeric	antenna separation
surveymode	character	survey mode (reflection/CMP)
fid	character	fiducial marks
ann	character	annotation (e.g., intersections)
name	character	name of the GPR data
description	character	description of the GPR data
filepath	character	file path of the GPR data
date	character	survey date
proc	character	processing steps applied to the data
vel	list	velocity model
hd	list	relevant additional information

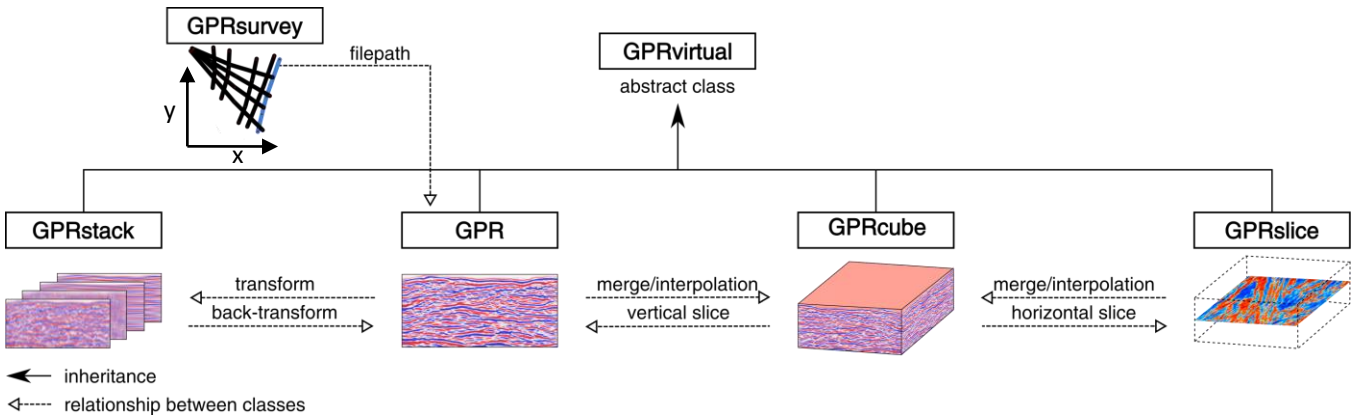

Fig. 1. RGPR class system

Some multi-dimensional decomposition transforms applied to two-dimensional GPR data results in a series of two-dimensional arrays with identical size (e.g., singular value decomposition, Fourier transform, Hilbert transform). The *GPRstack* class is built to represent such arrays of two-dimensional data, to process each array of two-dimensional data individually and to back-reconstruct the GPR data. Objects of the *GPRstack* class can be manipulated as array objects.

The *GPRcube* class represents GPR data cubes (e.g., GPR data collected along a regularly spaced grid). Objects of the *GPRcube* class can be manipulated as array objects and are created from GPR or *GPRslice* objects. The extraction of a horizontal slice from a *GPRcube* object results in an object of the *GPRslice* class that can be manipulated as a matrix (Fig. 1).

IV. AVAILABLE FUNCTIONS AND METHODS

A. Creating a GPR object

RGPR supports file formats from various GPR manufacturers. Importing GPR data in R is therefore straightforward and can be performed using the 'readGPR()' function (file formats currently supported are listed in the help page of this function). Furthermore, any matrix or array within the R environment can be coerced into an object of the GPR class.

B. Analysis and processing methods

RGPR currently provides a few analysis methods to explore the GPR data (TABLE II) as well as some basic one-dimensional and two-dimensional processing technics (TABLE III). A few more advanced processing methods such as mixed-phase deconvolution [6], topographic Kirchhoff migration [7,8], and two-dimensional adaptive smoothing (based on the R-package adimpro [9]) are also already implemented. Furthermore, RGPR allows the user to apply its own functions to the GPR data through wrapper functions for one-dimensional, two-dimensional and moving-window based processing.

TABLE II: ANALYSIS METHODS

1D (trace) analysis	2D analysis
• trace amplitude	• frequency-wave spectrum
• frequency and phase spectrum	• CMP analysis
• first wave break estimation	• structure tensor estimation

C. GPRsurvey methods

The methods of the GPRsurvey class exclusively manipulate the traces coordinates. More specifically, the methods currently allow to:

- display map of the GPR measurement lines that can be superimposed on any raster or vector data (Fig. 2.A),
- display GPR data in three-dimensional interactive graphics (fence diagram, Fig. 2.B) with the R-package RGL [10],
- add or interpolate trace coordinates (the elevation coordinates can be interpolated from raster data),
- georeference the trace coordinates (e.g., from a local coordinate reference system to a regional reference coordinate system),
- estimate the spatial shift between two (parallel) GPR profiles.

TABLE III: PROCESSING METHODS

1D (trace) processing	2D processing
• dewow	• f-k filter
• DC-shift	• 2D convolution
• trace average	• 3x3-median filter
• amplitude correction:	• 2D adaptive smoothing
- power gain	• topographic Kirchhoff migration
- exponential gain	
- automatic gain control	
• frequency filter	
• constant-offset correction	
• 1D convolution	
- deconvolution	
- minimum-phase	

D. Writing files

Once analyzed and processed, GPR objects can be saved in the R native '.rds' format for later use. In addition, GPR data can be exported back in their original file format or any other format currently supported by RGPR (including text file). Furthermore, RGPR allows GPR data to be exported as high-quality PDF file with full control on the aspect ratio. Any plot can be exported to almost any image format using the various R libraries dedicated to that purpose (e.g., Cairo [11], jpeg [12]).

Trace coordinates can be exported as spatial vector data (e.g., ESRI shapefiles). Hence, RGPR provide an interface between standard GPR file formats and files used in geographical information systems.

V. WHAT'S NEXT?

The current development version of RGPR offers access to the most popular GPR signal processing methods coupled with the flexibility of working in the R environment. However, as mentioned in the introduction, RGPR is still a work in progress and many functionalities remain to be added. For example, the ability to process transillumination GPR profiles. The structure of RGPR will allow to quickly and conveniently implementing additional methods in the near future. In the long term, adding more complex functionalities might require to bring modification to the class structure. RGPR will be most likely brought to evolve but its readily availability on GitHub will allow all users to easily assimilate these changes.

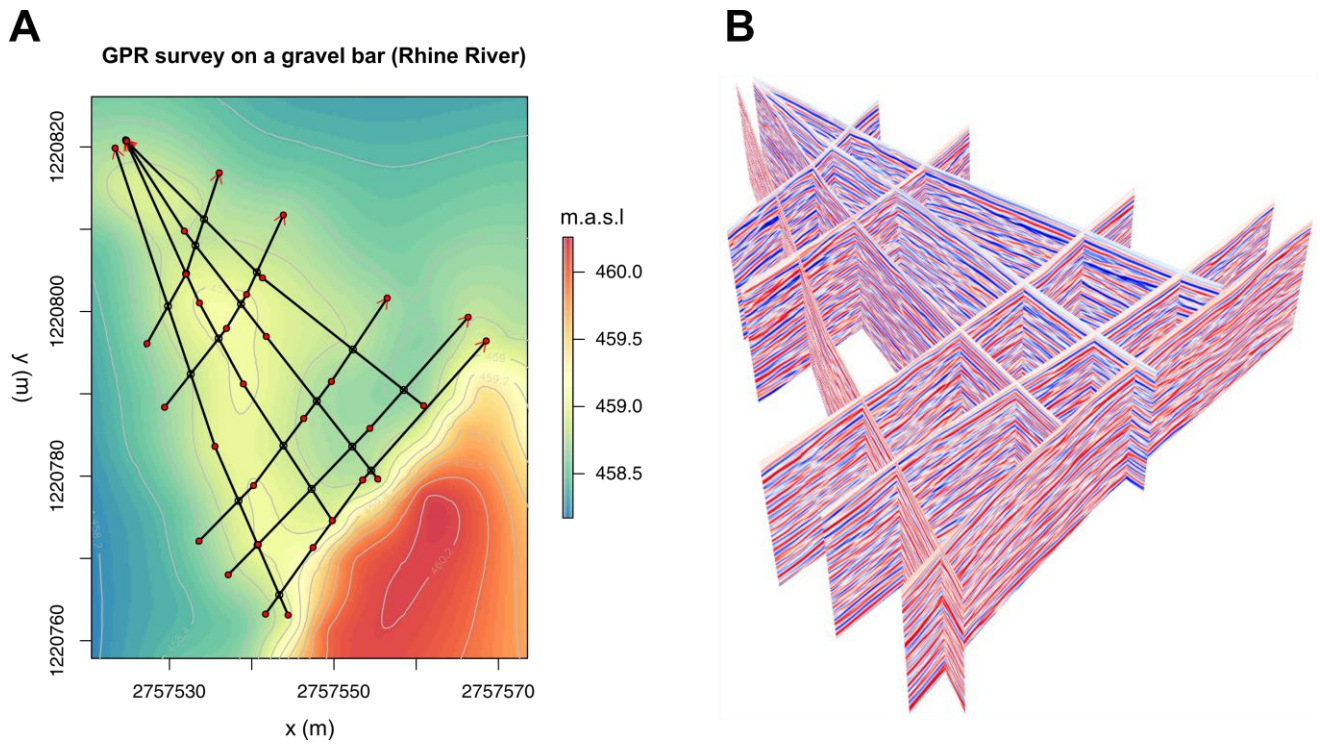

Fig. 2. (A) Graphical output of the 'plot()' method applied to an object of the *GPRsurvey* class superimposed on a raster elevation data: the black lines represent the GPR profile coordinates in the Swiss coordinate system, the red dots the fiducial markers, and the red arrows the survey direction. (B) Graphical output of the 'plot3DRGL()' method applied to an object of the *GPRsurvey* class: GPR fence diagram corresponding to (A).

A. Short term improvements

In the near future, a depth-varying velocity model for migration from CMP will be added to RGPR as well as three-dimensional processing technics and various transform functions (e.g., eigenvector decomposition, Hilbert transform, Stockwell transform). Furthermore, methods to delineate structures on GPR data will be developed including three-dimensional surface interpolation. Finally, the 'readGPR()' function will be enhanced to support more file formats.

B. Long term perspectives

In the long term, a GPR signal forward simulator could be added to RGPR to perform full-waveform inversion based on finite-difference time-domain (FDTD) solutions to Maxwell's equations. This could be implemented, for example, by calling the GPR simulator developed by [13] and coded in C directly in R.

VI. Conclusion

RGPR is a free and open-source R-package to process and visualize GPR data that is still in development. It is hosted on GitHub for collaborative development and intended for academic work. The structure of RGPR was designed with the primary focus of ensuring that signal processing steps can be easily reproduced and tracked. RGPR provides the first interface for GPR data analysis, processing and visualization in R. It opens infinite possibilities for researchers to implement, develop and evaluate their own data processing methods.

The ultimate goal of RGPR is to promote GPR related research by providing access to the flexible and rich R environment. RGPR has also a didactic vocation by encouraging students and young researchers to learn about GPR signal processing through various tutorials available on the RGPR GitHub repository and the R documentation.

Acknowledgment

Work by the first author was partially funded by the Swiss National Science Foundation (grants no. CRSI22_132249/1 and P2BSP2_161955).

References

[1] R. Ihaka, and R. Gentleman, "R: A language for data analysis and graphics," Journal of Computational and Graphical Statistics, vol. 5, pp. 299-314, 1996.

[2] N. Diakopoulos, and S. Cass, "Interactive: The Top Programming Languages 2017," https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017, July 2017, consulted on January 17th, 2018.

[3] E. Huber, and G. Hans, "R-package with S4 classes for ground-penetrating radar (GPR) data visualization, analysis, and processing", https://github.com/emanuelhuber/RGPR, consulted on January 17th, 2018.

[4] H. Wickham, W. Chang, RStudio, and R Core team, "devtools: Tools to Make Developing R Packages Easier", https://cran.r-project.org/web/packages/devtools/index.html, November 2017, consulted on January 17th, 2018.

[5] H. Wickham, P. Danenberg, M. Eugster, and RStudio, "roxygen2: In-Line Documentation for R", https://cran.r-project.org/web/packages/roxygen2/index.html, February 2017, consulted on January 17th, 2018.

[6] C. Schmelzbach, and E. Huber, “Efficient Deconvolution of Ground-Penetrating Radar Data”, IEEE Transactions on Geoscience and Remote Sensing, vol. 53, pp. 5209-5217, April 2015.

[7] F. Lehmann, and A. G. Green, “Topographic migration of georadar data: Implications for acquisition and processing”, Geophysics, vol. 65, pp. 836-848, May-June 2000.

[8] J.-R. Dujardin, and M. Bano, “Topographic migration of GPR data: Examples from Chad and Mongolia”, Comptes Rendus Géoscience, vol. 345, pp.73-80, February 2013.

[9] K. Tabelow, and J. Polzehl, “adimpro: Adaptive Smoothing of Digital Images”, https://cran.r-project.org/web/packages/adimpro/index.html, September 2016, consulted on January 25th, 2018.

[10] D. Adler, and D. Murdoch, “rgl: 3D Visualization Using OpenGL”, https://cran.r-project.org/web/packages/rgl/index.html, January 2018, consulted on January 25th, 2018.

[11] S. Urbanek, and J. Horner, “Cairo: R graphics device using cairo graphics library for creating high-quality bitmap (PNG, JPEG, TIFF), vector (PDF, SVG, PostScript) and display (X11 and Win32) output”, https://cran.r-project.org/web/packages/Cairo/index.html, September 2015, consulted on January 17th, 2018.

[12] S. Urbanek, “jpeg: Read and write JPEG images”, https://cran.r-project.org/web/packages/jpeg/index.html, January 2014, consulted on January 17th, 2018.

[13] J.R. Ernst, H. Maurer, A. G. Green, and K. Holliger, “Full-waveform inversion of crosshole radar data based on 2-D finite-difference time domain solutions of Maxwell’s equations”, IEEE Transactions on Geoscience and Remote Sensing, vol. 45, pp. 2807-2828, August 2007.