Enhancing Analyst by integrating the R package ‘Luminescence’

G.A.T. Duller

Aberystwyth Luminescence Research Laboratory (ALRL), Department of Geography and Earth Sciences, Aberystwyth University, Ceredigion, United Kingdom SY23 3DB

*Corresponding Author: ggd@aber.ac.uk

Received: November 29, 2018; in final form: December 18, 2018

Abstract

A new facility within the software package Analyst is described, allowing users to run R scripts with little or no need to interact directly with the R language themselves. This tight integration of R within Analyst provides users with a much more powerful platform in which to undertake selection and analysis of their data than is currently available. The current implementation focusses on the use of the R ‘Luminescence’ package for visualisation and statistical modelling of equivalent dose distributions, but the wide variety of other capabilities of the R statistical language can also be accessed using this new system. For users with more experience of using R, this new version of Analyst allows them to tailor their analysis in a way that is specific to their application, whilst retaining the simplicity of the Analyst interface. The ability to exploit the complex statistical methods available in R from within the Analyst environment provides a powerful new approach to data analysis in luminescence research.

Keywords: Analyst, Data analysis, R

1. Introduction

The number of steps involved in protocols being used for dose reconstruction using luminescence is expanding (e.g. the multiple elevated temperature (MET) IR protocol of Li & Li (2011), and the violet stimulated luminescence (VSL) SAR protocol of Ankjærgaard et al. 2013), as is the range of luminescence signals being measured (thermoluminescence (TL), isothermal luminescence, optically stimulated luminescence (OSL), pulsed OSL, spatially resolved TL or OSL etc.).

The Analyst software package provides a simple user interface allowing researchers to process their data (Duller, 2015). The aims of Analyst have been fourfold: (1) to allow users to select which parts of a sequence of data measurements should be used for analysis; (2) to allow users to visualise each measurement in a simple manner; (3) to undertake analyses such as dose response curve fitting and equivalent dose determination, OSL curve fitting, and g-value determination; and (4) to make it simple to export data from Analyst so that it can be analysed in other software. The approach used in Analyst is to make interaction with data a very visual and direct process. Changes are made using the mouse or keystrokes, and the impact of these is immediately seen. However, a major disadvantage of Analyst is that it cannot be configured by users to undertake new calculations, new methods of analysis, or new types of display.

In recent years a highly flexible package, ‘Luminescence’ (Kreutzer et al., 2012), for the language R specifically aimed at assisting research in luminescence has been developed. R is an open source statistical language (R Core Team, 2018) that has become extremely widely used, especially in scientific work (e.g., Tippmann, 2014). One of the most powerful aspects of the language is the ability of users to write their own functions, and collect these together to create a package, and to make this package available to users around the world via CRAN, the Comprehensive R Archive Network (https://cran.r-project.org). Documentation is always provided with these functions, so that users can obtain information about the purpose of a function and how to use it.

A common reason why many luminescence researchers may have begun to use the statistical language R is to run some of the age models described by Galbraith et al. (1999)
such as the central age model (CAM) or the minimum age model (MAM). These were originally written in the language S, but the models run with little modification in R which is freely available. New models such as the IEU model are also now available (Thomsen et al., 2007; Smedley, 2015). The ‘Luminescence’ package (Kreutzer et al., 2012) provides a wide range of analytical capabilities specifically dedicated to the analysis of luminescence data, especially its use in dating. Since its launch in 2012 the package has undergone rapid expansion in the range of functions that it provides.

However, using the ‘Luminescence’ package requires at least some understanding of the R language, and while detailed examples of how to use the package for analysis of data for dating have been published (e.g., Dietze et al., 2013; Fuchs et al., 2015), the need to learn some elements of the R language has been an impediment to some researchers adopting this new approach. Burow et al. (2016) provide an alternative route with the software ‘RLumShiny’, that creates a simple graphical user interface (GUI), but here the functionality is fixed by the designer, and not by the user, and so only part of the power of the ‘Luminescence’ package and the R language are available. This paper describes changes made in the latest version (v4.57) of Analyst which allows users either to exploit the power of the ‘Luminescence’ package without having to write any R code of their own, or allows users to write simple scripts (or modify existing scripts). All of this is achieved from within the Analyst package, to simplify access to the power of R.

2. Implementation within Analyst

Analyst runs scripts (sets of commands written in the R language) using the programme RScript that is a core part of R itself. Analyst writes the script chosen by the user to a file, calls RScript with the name of this file as an argument, and then collects the output from RScript to present it within Analyst (Fig. 1). Currently, the part of Analyst where R can be accessed with data is the single aliquot regeneration section. Analyst provides the opportunity to analyse many single aliquots, or single grains, to generate many equivalent dose ($D_e$) values. These $D_e$ values are written to a file and then read by the R script. As well as the $D_e$ and its uncertainty, parameters such as the intensity of the luminescence signals, the $L_x/T_x$ ratios and their uncertainties, the recycling ratio, recuperation, the fitting parameters for the equation used to fit the dose response curve, and many others are exported and hence are available for further analysis within the R script.

The range of parameters exported to the R script is substantially larger than was available in the summary table of earlier versions of Analyst. The most important enhancement has been to export the dose response data for each aliquot. For each point in a regenerative dose measurement sequence, the dose, the signal from the regenerative dose ($L_x$), its background, the signal from the test dose ($T_x$), its background, the ratio of $L_x/T_x$ (and its uncertainty) are all exported. This opens up many possibilities for data analysis in R. For instance, it makes it possible to plot multiple dose response curves (Fig. 2). An R script can be run using a single click of a mouse, and any text results are presented within a window that is part of the Analyst programme (Figure 2). Graphical output produced by the R script is also shown within the Analyst programme, and can be copied to other packages via the Windows\textsuperscript{TM} clipboard.

3. Scripts and variables

A number of scripts are automatically available when Analyst is installed, and can be seen as soon as R is detected (the user is responsible for installing R on their computer). These scripts are simple sets of R commands. There is no limit to the length of the scripts. The scripts consist of a series of lines of commands and comments. Extensive use of comments is encouraged in order to make the intention of the
code clear. An example of a script is given in Listing 1. To make it simpler to discuss the script in this article, line numbers have been added in Listing 1, but these are not part of the script itself.

Lines 1 to 10 are simply comments, and give the name of the script (line 2) and its purpose (lines 4 to 7). The script first ensures that the ‘Luminescence’ library is activated (lines 12 to 14), and then reads the data file (lines 18–19) that has automatically been created by Analyst from the single aliquot analyses. Line 20 removes any results that do not contain finite $D_e$ values and uncertainties on the $D_e$. The table of results is stored in the variable LumData, and by attaching this object (line 24) the various fields within the structure (the different columns of data within the Analyst table) can be accessed just by writing their name. For instance, the list of $D_e$ values and their uncertainty can be accessed simply using ED and ED_Err (line 28). To see a list of the names of the columns of data in the file written by Analyst, users can run the script List_SAR_data_fields which is included with Analyst.

Listing 1. RAN-file example - MAM_Abanico

```r
#==========================================#
# Name: MAM_Abanico                           #
# # Purpose: To fit the Minimum Age Model     #
# to the De dataset and then create          #
# an Abanico plot of the data                #
# showing the derived MAM De                 #
# # GAT Duller, August 2017                  #
#==========================================#

if(!require(Luminescence)){
  library("Luminescence")
}

#Remove any incomplete rows of #data where ED or ED_Err are not present
LumData <- read.delim("AnalystDedata.txt", sep="\t",header=TRUE)
LumData <- LumData[is.finite(LumData$ED) & is.finite(LumData$ED_Err),]
```

Figure 2. Output from running an R in Analyst script to plot multiple dose response curves for different aliquots of a single sample. The user interface shown here is within the Analyst program. The four different sections of this interface are labelled. The R script to be run is shown in the upper left hand panel. Any variables defined by the user (see later in the text) are listed in the upper right hand panel, and the values of these variables can be changed when the user runs the script. Once the script is run (from the Analyst Function Menu item), any text output from R is shown in the left hand panel, and any graphical output is shown in the bottom right hand panel.
The minimum age model of Galbraith et al. (1999) is run in lines 39–40 using the function `calc_MinDose()` which is part of the `Luminescence` package. This function requires a list of $D_0$ values and their uncertainties, and these are in the variable `AnData` that was created in line 28 from the list generated by `Analyst`. The function also requires users to specify a value of `sigmab`, the overdispersion exhibited by a well bleached population of $D_e$ values. The correct choice of `sigmab` is critical to the running of the minimum age model, and the subject of much research (Galbraith & Roberts, 2012). It would be possible to write a script where the value of `sigmab` was fixed, but if the user wished to see the impact of using a different value then they would have to edit the script, save it, and then rerun it. A more flexible approach is to use a variable. In the script shown in Listing 1 the user can change the value of `sigmab` without having to edit and save the script.

`Analyst` allows users to define as many variables are they like, and these are numbered from 1 upwards. A variable is shown by preceding it with a dollar symbol (e.g., variable 3 is `$3$`). In the script (line 40) the value of `sigmab` is variable 1 (`$1$`). When the script is ready to be run, `Analyst` shows a list of variables (right hand side of Fig. 3), and whatever value a user types into this box will be used to replace any occurrence of the variable `$1$`. In this example, variables are also used so that users can select whether the logged or unlogged MAM is used (variable 2, `$2$`), and whether the 3 parameter or 4 parameter MAM is used (variable 3, `$3$`). Variables give users a very rapid method of exploring the impact of changing a value, and allows scripts to be more flexible than if all the parameters were fixed. Variables can also be used to alter the size of symbols in plots, the text displayed on axes, and many other aspects of the script.

As an example of using R within `Analyst`, Fig. 4 shows the results of running the script shown in Listing 1 for a single grain quartz data set for a fluvial sample from central southern Africa (Larkin et al., 2017).

As well as providing access to functions that have been included in existing packages (such as `Luminescence`), the ability to write ones own R scripts opens up enormous opportunities for data exploration, and for adopting novel methods of analysis. A good example is provided by the suggestion of Thomsen et al. (2016) to analyse single grain $D_e$ distributions by taking into account the characteristic saturation dose ($D_0$) of each individual grain. All of the information needed to undertake the procedure described in that paper is available within `Analyst`, and the ability to write R scripts allows the user to develop a script that could automate this analysis whilst the user remains within the `Analyst` GUI environment.

4. Limitations of implementing R in `Analyst`

The ability to run R scripts from within `Analyst` is designed to allow users to benefit from the power of R (and especially the `Luminescence` package) without having to have a deep understanding of the R language and syntax. While the implementation in `Analyst` provides a great simplification compared with running in the R terminal itself, there are a number of limitations and drawbacks.

The most important of these is that if researchers wish to write new scripts or edit existing ones, there is no help within `Analyst` about the R language, or about the parameters needed for different functions. One of the benefits of the system of packages in R is that they all provide documentation of how to use the functions, along with examples. Although `Analyst` does not provide access to this help, the site https://www.rdocumentation.org/ provides a very convenient interface to all of the help files for R packages.
Typing ‘Luminescence’ or the name of a specific function (e.g., calc_MinDose()) into this search tool will bring up help associated with that topic, and this will normally include example scripts demonstrating how to use a function.

A second disadvantage is that programming environments such as RStudio® (https://www.rstudio.com), provide a lot of assistance when writing R code, such as autocomplete (similar to predictive text on a mobile phone), but this type of help is not available within Analyst. An effective solution to these limitations for more experienced users is to work both within RStudio® and Analyst when developing scripts, then to use copy and paste so that scripts developed in RStudio® can be run in Analyst.

5. Conclusions and future developments

This new functionality in Analyst was first demonstrated at a workshop that preceded the 15th International Luminescence and Electron Spin Resonance Dating conference held in Cape Town in September 2017. In the 4 hour afternoon session dedicated to Analyst, delegates whose previous experience of Analyst varied from almost zero to many years, and whose experience of R varied from none to very experienced, all managed to successfully run R scripts within Analyst to run the MAM on a set of $D_e$ values, and to use R to plot these on an Abanico plot (Dietze et al., 2016). The feedback from that session has been used to improve the current version, especially to overcome some of the issues with permissions within Windows for installing the software. It is hoped that other colleagues will find this new facility relatively simple to use, and that it will provide a great deal of power to analyse luminescence data. Analyst (v4.57) can be downloaded for free from http://users.aber.ac.uk/ggd.

While the scripts automatically installed with this latest version of Analyst have been designed to utilise the functions within the ‘Luminescence’ package, other parts of the R language, and other R packages can also be used (e.g., ‘numOSL’, Peng et al. 2013).

Scripts written for R within Analyst can be saved as so-called RAN-files. These files can then be imported into Analyst at a later date, or into Analyst installed on a different computer. The ability to save these scripts is so that users can archive the scripts, or so they may make the scripts available to other researchers. In this way it is hoped that the types of analyses that can be undertaken on luminescence data can be expanded, and that where authors show results from novel types of analysis they are able to make the scripts available as RAN-files so that other colleagues are able to use these new methods. A website where RAN-files can be shared is under construction and the author welcomes contributions of RAN-files from colleagues.

Acknowledgments

I would like to thank the R ‘Luminescence’ package development team for inviting me to their development meeting in 2016 where the ideas for this integration with Analyst took shape. I would also like to specifically thank Sebastian Kreutzer for his assistance and his feedback on this new aspect of Analyst.