Ancient TL

Software in the context of luminescence dating: status, concepts and suggestions exemplified by the R package 'Luminescence'

Sebastian Kreutzer,[1*] Christoph Burow,[2] Michael Dietze,[3]
Margret C. Fuchs,[4] Manfred Fischer,[5] & Christoph Schmidt[5]

[1] IRAMAT-CRP2A, Université Bordeaux Montaigne, Pessac, France
[2] Institute of Geography, University of Cologne, Cologne, Germany
[3] Section 5.1: Geomorphology, Helmholtz Centre Potsdam, GFZ German Research Centre for Geosciences, Potsdam, Germany
[4] Helmholtz-Zentrum Dresden-Rossendorf, Helmholtz-Institut Freiberg for Resource Technology, Freiberg, Germany
[5] Chair of Geomorphology, University of Bayreuth, Bayreuth, Germany

*Corresponding Author: sebastian.kreutzer@u-bordeaux-montaigne.fr

Received: Nov 11, 2016; in final form: July 4, 2017

Abstract

The relevance of luminescence dating is reflected by the steadily growing quantity of published data. At the same time, the amount of data available for analysis has increased due to technological and methodological advances. Routinely, luminescence data are analysed using a mixture of commercially available software, self-written tools and specific solutions. Based on a luminescence dating literature screening we show how rarely articles report on the software used for the data analysis and we discuss potential problems arising from this. We explore the growing importance of the statistical programming language R in general and especially its reflection in recent software developments in the context of luminescence dating. Specifically, for the R package `'Luminescence'` we show how the transparency, flexibility and reliability of tools used for the data analysis have been improved. We finally advocate for more transparency if unvalidated software solutions are used and we emphasise that more attention should be paid to the tools used for analysing the data.

Keywords: R, Software, Luminescence dating, Data analysis

1. Introduction

Luminescence dating studies require comprehensive data analyses. Moreover, technological advances and methodological developments during the last decades have increased the amount of data available. However, how much emphasis is, or rather should be, put on the software used to analyse the data? Should we care about software development in general? For most of the researchers in the luminescence dating community, software is merely a tool to analyse data and conduct research. Moreover, not every update of such tools is worth publishing nor does every minor (or even major) change in, e.g., the *Analyst* (Duller, 2015) or the **R** package `'Luminescence'` (Kreutzer et al., 2012) always appeal to the vast majority of the luminescence dating community. Nevertheless, researchers may encounter problems, where no alternative software solution is readily available. Researchers are not usually skilled in programming or trained in managing software development projects, even though particular research questions sometimes demand such solutions. However, the design and the usage of self-written, specialised tools raises further challenges, such as verification and validation, bug tracking and even licensing questions. At the same time, scientific standards (e.g., documentation, transparency, reproducibility) need to be ensured.

In this study, we aim at shedding light on the role of software in the context of luminescence dating. Therefore we have conducted a literature screening and have compiled a list of software tools developed over the last years. We highlight the growing importance of the statistical programming language **R**. **R** and the package `'Luminescence'` are used to

exemplify standard software engineering practices and software test tools applied for developing research software. Finally, we discuss the advantages and challenges arising from the development of highly specialised software tools and we make suggestions for further developments. Our contribution consists of two parts: (I) a general presentation and discussion of the status quo of software tools used by the luminescence dating community and (II) a description of technical concepts embedded within the **R** package `'Luminescence'`.

Henceforth, we report names of software in *italic* letters and the names of **R** packages in single quotes and monospace letters (e.g., `'Luminescence'`). For program code we use `monospace` letters. Hyperlinks to internet resources, beyond the references, are provided as footnotes at their appropriate positions.

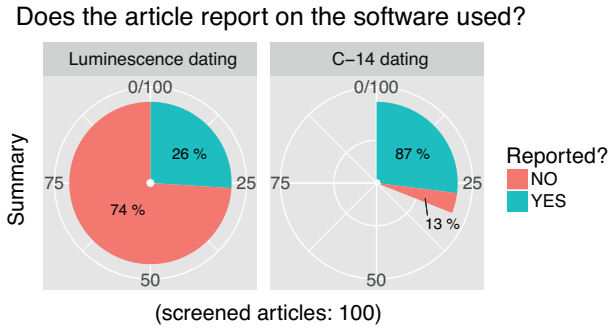

Figure 1. Results of the conducted literature study. 26 out of 100 screened articles report on the software tool used for analysing the luminescence data (left chart). By contrast, 31 out of these 100 articles additionally reported on ^{14}C dating results and from these 31 studies 27 (87 %) reported on the applied software to obtain the ^{14}C results. For details see main text. The graphic was produced using `'ggplot2'` (Wickham, 2009).

2. Software and its role in luminescence data analysis

2.1. Observations from the literature

To examine the role of software in the context of luminescence dating we conducted a literature study. The ten last published articles, presenting new luminescence dating results, of 10 international peer-reviewed journals (cf. Table S1, 100 items in total, closed volumes only, years 2016 to 2014, screening period: 2016-10-03 to 2016-10-08), were systematically screened for information given on the software used for the luminescence data analysis. Our chosen definition of data analysis includes equivalent dose modelling, age calculation and dosimetric calculations. Supplementary data provided with the screened articles were taken into account, but not referred articles. Exceptions were made for articles where substantial information on data and procedures were spread over more than one manuscript.

A table listing all screened articles is provided as supplement (Table S1). Screened journals were: *Boreas* (*BOR*), *CATENA* (*CAT*), *Earth and Planetary Science Letters* (*EPSL*), *Geomorphology* (*GM*), *Journal of Quaternary Science* (*JQS*), *Quaternary Geochronology* (*QG*), *Quaternary International* (*QI*), *Quaternary Research* (*QR*), *Quaternary Science Reviews* (*QSR*) and *The Holocene* (*HOL*). Note that this selection may be biased by preferring journals with an easy online accessibility; no further randomisation took place. Due to the applied selection articles published during the last two years were favoured. Hence, the overall explanatory power of the screening is limited.

In *CAT* and *JQS*, 5 out of 10 articles reported on the software used in the context of luminescence dating, in *EPSL*, *GM* and *QSR*, 3 out of 10 articles, in *QG*, *QR* and *HOL* it were 2 out of 10 articles each. In *BOR*, 1 out of 10 studies provided information, and none of the ten screened articles in *QI* reported on the software used. Thus, 26 out of 100 recently published articles reported on the software used for analysing the luminescence data (cf. Fig. 1). In 9 out of the 26 articles further details (e.g., software version number) were given and in only two of the articles, the references fully covered the data analysis carried out. Note: Software used for data visualisation only was not considered.

Software tools mentioned in these articles were (in alphabetical order):

- *ADELE* (Kulig, 2005, and one time *ADELE2015*, unpublished),
- *AGE* (Grün, 2009),
- *Analyst* (Duller, 2015),
- the **R** package `'Luminescence'`(Kreutzer et al., 2012, 2017),
- *DRAC* (Durcan et al., 2015),
- *DRc* (Tsakalos et al., 2015),
- *RadialPlotter* (Vermeesch, 2009),
- *PAST* (Hammer et al., 2001) (here for luminescence data regression analysis)
- and one self-written *Excel*® sheet.

Except for the programme *ADELE* (both versions) and the self-written *Excel*® sheet all cited tools were freely accessible at the time this article was written; links are given in the reference list.

Our observation differs for those articles that also include ^{14}C analysis. There, 87 % report on the software used to calculate the ^{14}C ages. In the ^{14}C community, specialised software such as *OxCal* (Ramsey, 1995) or *CALIB* (Stuiver et al., 2016) may have set a quasi-standard and their usage and citation may therefore be considered as indispensable for publishing an article. However, in other research communities (e.g., biology), Howison & Bullard (2015) encountered the similar problematic practise of missing citations and information on the software used.

2.2. Software in the wild

The previously mentioned observations from the literature are contrasted by the remarkable number of specialised software packages formally or informally released in the broader context of luminescence dating and related data analysis (in alphabetic order): *ADELE* (Kulig, 2005), *AGE* (Grün, 2009), *AgesGalore*[1] (Greilich et al., 2006), *AgesGalore2* (Greilich et al., 2015), *Analyst* (Duller, 2007, 2015), *DOSE* (Brumby, 1992), *DosiVox* (Martin et al., 2015), *DRc* (Tsakalos et al., 2015), *DRAC* (Durcan et al., 2015), *FitBin9* (Bailey, 2008), *FITT* (Grün & Macdonald, 1989), *Hybfit* (principle described in Bluszcz & Adamiec, 2006), *PTanalyse* (Lapp et al., 2009), *RadialPlotter* (Vermeesch, 2009), *RLanalyse* (Lapp et al., 2012). This list must be considered as non-exhaustive, which indicates a general demand for specialised solutions to deal with luminescence data. However, software may have not been taken into account or may have been overlooked. This includes various self-written *Excel*®, *MATLAB*®, *S* scripts and **R** scripts and individual software solutions that were never formally published, but circulate in the wild.

2.3. The R factor

Additionally, we observed a rising trend of published software based on the statistical programming language and environment **R** since 2012. Currently available **R** software (henceforth packages) dedicated to luminescence data analysis in a broader sense are listed in Table 1. All of them were first published during the last five years. The table lists the name of the package, the package maintainer as well as the latest available version and provides a short description. The columns 'Access via' (subcolumns CRAN and GitHub) and 'Licence' inform accessibility and on the selected legal statement. A licence sets the needed legal framework the software can be used in, modified and further propagated. Every software reported in Table 1 was published under GNU general public licence (GPL)[2] conditions.

3. The software dilemma

Considering the amount of software developed in the luminescence dating community over the time, it is remarkable how few articles (cf. Sec. 2.1) report on details of the software tools used for their data analysis. We consider this practice as problematic for several reasons:

1. The unanswered question "Which tool was used to analyse the data?" may lead to "How much trust can be put in presented data if important details on the data analysis remain undocumented?". Thus, reporting on the tools is indispensable to ensure transparency and reproducibility of published results.

2. Software is never free of bugs, might be inappropriately used and 'degrades' through time. Software that is no longer developed and not regularly maintained to function properly may suffer from increasingly frequent disabilities (e.g., unsupported new file structures, methods, operating systems) and eventually stop working at all. Changes are not always visible to the user and running software may undergo modifications causing changes for the data output. A proper reporting links the results to a particular tool and its version. It cannot prevent mistakes, but it helps to track them down.

3. Free and open-source software usually comes without any warranty. Hence, the user should not blindly trust every change undertaken and every new version released. Here, software quality assurance (e.g., testing, cross validation) is a responsibility that cannot be taken by the developers of analysis tools alone or should not be fully committed to them. If nothing is reported and software errors are discovered later, they are not linked to the article and can hardly be recognised.

The last point deserves further attention. Although the software tools reported in Sec. 2.2 were made freely available by the authors, the source code is not accessible to the public. By contrast, the source code of all the **R** software listed in Sec. 2.3 is available via public repositories. A non-accessible closed source tool is not by default inherently better or worse than a freely available open-source tool. However, if disagreements in the results from different software tools are encountered the ability to track down errors, such as deviations in calculations, are reduced. Open-source software balances the roles between developers and users, but only if recognised as an opportunity. At present, however, common reporting practice indicates that computational work and its developments are not yet part of the daily scientific routine in the luminescence dating community.

Beyond transparency, open-source software and in particular software published under GPL-3 (cf. Sec. 2.3) encourages code recycling, and tailored solutions can be built on existing code. The statistical programming language **R** provides a very robust and popular environment (Tippmann, 2014). The reuse and extension of code are capable of changing the way research is carried out. However, it comes at the cost of caretaking for reproducibility and reliability, and time must be invested to build up skills in programming.

In the second part of our paper, we explore some lesser known features of the programming language **R**. We further present and discuss concepts and development processing tools implemented in the **R** package `'Luminescence'`.

4. The popularity of the R environment

Since our first article on the **R** package `'Luminescence'` (Kreutzer et al., 2012), the popularity of **R** has risen remarkably. This development can be seen by the Comprehensive **R** Archive Network (CRAN)[3] statistics. The first **R** package on

[1] This software should not be confused with the **R** package `'AgesGalore'` (Greilich, 2013)

[2] https://www.gnu.org/licenses/gpl-3.0.en.html

[3] https://cran.r-project.org

Table 1. To date available **R** packages in alphabetic order dealing with luminescence and ESR data in a broader sense. URLs are given in the reference list.

Name	Maintainer	Version	Description	Licence	Access via CRAN	Access via GitHub[1]	Reference[2]
'AgesGalore'	Steffen Greilich	0.0.3 [2013-12-16]	Collection of routines complementing *AgesGalore 2*	GPL-3	-	-	unpublished, Greilich (2013)
'ArchaeoPhases'	Anne Philippe	1.2 [2017-06-13]	Post-Processing of the Markov Chain Simulated by *ChronoModel*, *Oxcal* or *BCal*	GPL-3	X	-	Philippe & Vibet (2017a)
'ESR'	Christoph Burow	0.1.0.9031 [2017-07-03]	Analysing and plotting Electron Resonance Spin (ESR) data	GPL-3	-	X	unpublished, Burow (2015)
'KMS'[2]	Jun Peng	no number [2015-11-04]	Collection of kinetic models for simulating quartz luminescence	custom & GPL-3	-	X	Peng & Pagonis (2016)
'Luminescence'	Sebastian Kreutzer	0.7.5 [2017-06-26]	Comprehensive luminescence dating data analysis	GPL-3	X	X	Kreutzer et al. (2012, 2017)
'LumReader'	David Strebler	0.1.0 [2017-01-27]	Package to simulate and visualise technical aspects of a luminesce reader	GPL-3	X	X	Strebler (2017)
'numOSL'	Jun Peng	2.3 [2017-05-18]	Numerical routines dealing for OSL dating, e.g., D_e calculation, dose response curve fitting	GPL-3	X	-	Peng et al. (2013); Peng & Li (2017)
'RChronoModel'	Anne Philippe	0.4 [2017-01-12]	Collection of functions for post-processing data returned by the software *ChronoModel* (Lanos et al., 2015), this includes chronological frameworks based on luminescence dating data	GPL-3	X	-	Philippe & Vibet (2017b)
'RLumModel'	Johannes Friedrich	0.2.1 [2017-04-13]	Simulate luminescence signals based on published models, e.g., Bailey (2001)	GPL-3	X	X	Friedrich et al. (2016, 2017)
'RLumShiny'	Christoph Burow	0.2.0 [2017-06-26]	Graphical interface for the **R** package 'Luminescence'	GPL-3	X	X	Burow et al. (2017, 2016a)
'rxylib'	Sebastian Kreutzer	0.1.1 [2017-07-07]	Functions to import xy-data into **R** (e.g., from γ-ray spectrometer)	GPL-3	X	X	Kreutzer (2017)
'TLdating'	David Strebler	0.1.3 [2016-08-31]	Functions dealing with TL data using the MAAD and SAR protocol	GPL-3	X	X	Strebler et al. (2016); Strebler (2016)
'tgcd'	Jun Peng	2.0 [2016-09-06]	Functions for TL curve deconvolution	GPL-3	X	-	Peng et al. (2016); Peng (2016)

[1] The source code of every **R** package on CRAN is additionally available on GitHub, but here only listed if the source code is actively managed by the package author(s) on GitHub
[2] A software is considered as published if it is (a) released via CRAN and/or (b) presented in a peer-reviewed journal
[3] Not available as a distinct **R** package, but as a collection of **R** functions
CRAN: comprehensive **R** Archive Network. https://cran.r-project.org
GitHub: online platform for the open-source version control system *Git*. https://github.com

CRAN was released in 1997. In 2012, the year the **R** package 'Luminescence' was introduced, the CRAN counted almost 4,000 packages and the package 'Luminescence' became number 3,918[4]. When this article was written there were 10,255 active **R** packages (07/2017: 11,018), 1,322 packages of which have been added from January to August 2016 alone (Hornik & Zeilis, 2016) and CRAN counted 6 to 7 million downloaded individual packages every week[5].

[4] https://gist.github.com/daroczig/3cf06d6db4be2bbe3368, accessed: 2016-10-06

[5] http://www.r-pkg.org; accessed: 2017-03-12 & 2017-07-10

Every package published on CRAN is well integrated in the **R** environment (convenient download and installation) and available for all three major platforms (*Windows*®, *macOS*® and *Linux*®) along with the source code and a reference manual. GitHub[6] is a commercial repository to maintain, develop and host the source code of software written in all kinds of programming languages. GitHub uses the version control system *Git* (e.g., Chacon & Straub, 2014) and can be used free of charge as long as the source code is made public and open-source. The popularity of GitHub amongst the **R** community may result from its good integration provided by the **R** community itself, e.g., packages under development can be directly downloaded and installed out of the **R** environment.

In Table 1 the columns CRAN and GitHub (Table 1) inform on how the package itself and the program code are made available to the public. As mentioned before, the term CRAN refers to the location a package can be submitted to after it has successfully passed some technical tests (cf. Sec. 5.4) and as long as it does not violate the repository rules[7]. At present, the source code of all **R** packages on CRAN is additionally mirrored on GitHub even if the package is not developed on GitHub itself. By contrast, Table 1 only lists packages available via GitHub if they are actively developed and maintained via GitHub.

General advantages of using **R** for luminescence data analyses have been outlined already elsewhere (Kreutzer et al., 2012; Dietze et al., 2013; Fuchs et al., 2015; Dietze et al., 2016). A lesser known fact is that CRAN automatically archives all packages ever published on the network. Thus even after a package was updated or removed, the older versions are still available. Additionally, automatic checks are run by the CRAN before every package submission and regularly after the package was released on CRAN. These tests provide a certain degree of technical compatibility and stability. They ensure, amongst other things, that the package can be installed without error (e.g., all dependencies are available) and it provides tests for all package examples. Packages failing the automatic tests are usually removed from CRAN and become archived until the developers have addressed the raised issues.

5. Transparency, flexibility and reliability

The development of software for scientific research demands extra care regarding transparency, flexibility and reliability. During the last five years of developing the **R** package `'Luminescence'`, we have adapted concepts established in computational science of which the most important aspects are presented below.

[6] https://github.com

[7] https://cran.r-project.org/submit.html, accessed: 2016-10-03

5.1. Transparent development process

All packages on CRAN are, including the source code, freely accessible. The development process itself is not always visible to the user. This can lead to situations where the developer might already be aware of a critical bug leading to a wrong calculation and eventually fixes that bug, but if the bug fix is not announced when the new version is released, the user might not even become aware of a faulty version that previously produced erroneous results. To improve the transparency of the development process, for the package `'Luminescence'` it was decided to move the entire development process, including the bug tracking, to an open repository, namely: GitHub.

In agreement with the GPL-3 licence conditions, the package still comes without any warranty, but now offers maximum transparency and an open handling of bugs. All modifications made to the software are recorded as so-called commits, usually enhanced by comments. Each of these commits keeps track of all individual changes, which enables a side-by-side comparison of a newer and older version of a piece of code or file. As shown in Table 1 this step is not limited to the `'Luminescence'` package and it is also not a new software development concept, but it is an important step which allow a proper peer-reviewing process of tools used for the data analysis.

5.2. Object standardisation

Working with **R** can sometimes become a rather disjointed experience, especially if different packages are involved (Boettiger, 2015). Many packages are tailored to deal with rather specific problems or to solve only one particular task. Some packages comprise only a few functions and even these few functions within one package may work with a different logic.

For example, the first version of the package `'Luminescence'` was just a collection of functions that could be used independently for one or the other purpose, e.g., analysing linearly modulated optically stimulated luminescence (LM-OSL) signals or plotting equivalent dose (D_e) distributions. Both are still possible, but due to the standardisation of function argument names and a changed package structure, more comprehensive data analyses are now possible.

Figure 2 shows the general package structure for the last version of the `'Luminescence'` package. The development of this structure was guided by the idea that the luminescence data need to be first imported independently of the initial data format and is then transformed into a coherent internal structure.

Therefore, an interface was integrated that converts all kinds of luminescence input data (e.g., a BIN- or XSYG-file) into a new unified data structure consisting of so-called `RLum-objects`. The details of this structure are beyond the scope of this contribution and may change in the future. Once the data are available as `RLum-objects`, they can be passed from one function to another. Thus, as long as the

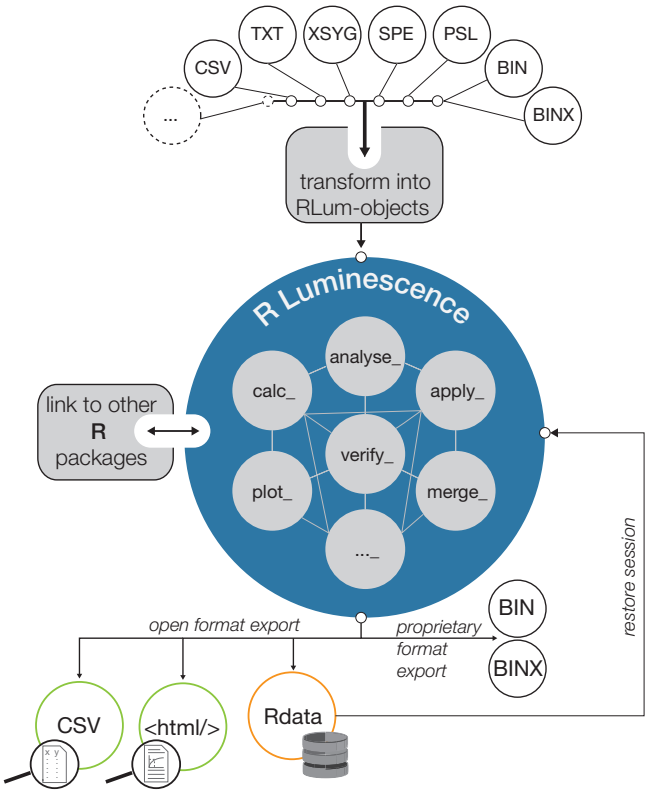

Figure 2. **R** ‘Luminescence’ package structure with indicated input/output interfaces. Once the raw measurement data have been transformed into an RLum-object, data can be passed without further transformation from one function to another within the package environment.

data are processed within the package environment (blue circle in Fig. 2), they can be analysed and combined in manifold ways, paving the way for new types of data analysis, e.g., data mining and big data analysis. For example, Burow et al. (2016b) reported the photo-ionisation cross section obtained from CW-OSL fitting of 5,488 CW-OSL curves extracted from 58 BIN-files (348 aliquots, 17 fine grain quartz loess samples from Europe). Their preliminary work showed the potential of data analyses carried out on a large scale, determining realistic distributions of expected natural variations.

The analytical output can be a graphic, a printed text in the **R** terminal, a new **R** object, an individual object created by the package ‘Luminescence’ (RLum-object) itself, or a combination of these possibilities. Numerical output can be exported to various formats natively supported by **R** (e.g., a CSV-file) or even proprietary formats (e.g., a BIN/BINX-file using the ‘Luminescence’ package). **R** allows for the export or archiving of an entire session (Rdata-file), which can be used to continue the data analysis with all previously created objects at any other time or to load the data and objects into a new session.

Another way to share analytical output with persons not familiar with **R** is the implemented possibility to export a function output to an HTML-file by creating a report using the function report_RLum(). Detailed examples are given in the package manual itself; an exemplary output is shown in Fig. 3.

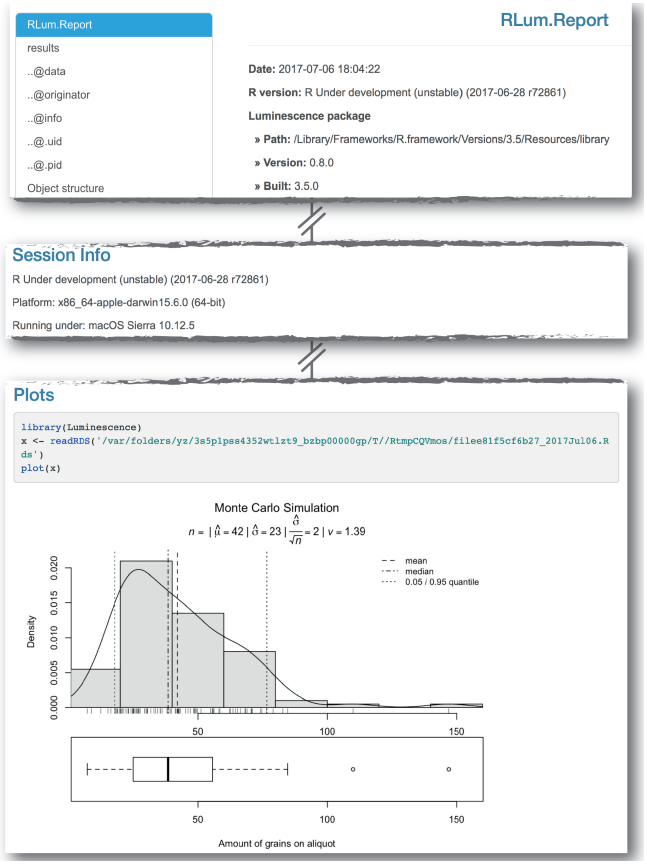

Figure 3. Screenshot of an HTML report produced using the function report_RLum() from the **R** package ‘Luminescence’. The output object used for the example was produced by the function calc_AliquotSize(). The output has been manually reduced for this figure.

The HTML-file can be opened with any modern web browser. It reports analytical output and parameters used for data analysis when producing the object, e.g., the **R** version, the package version, the operating system etc. Standard plot outputs are partly included. In this way, the provided information on a performed data analysis is optimised regarding a maximum transparency. Since HTML-files are human readable (non-binary) and can be opened by a web browser or by any text editor, the included information are available to every reviewer and reader.

5.3. Task modularisation

A programmer’s mantra is: do not repeat yourself. In other words, existing code designed for a particular task should be reused instead of implementing new code. With package modularisation, the **R** environment is well suited to reuse solutions developed by others. The ‘Luminescence’ package takes direct and indirect advantage of $>$ 50 other **R** packages. The packages are imported during installation or later (upon request) and are connected to the

'Luminescence' package based on the ability to link packages in the **R** environment and import functionalities from one package into another. Figure 4 illustrates how the package 'Luminescence' can be (and is) connected to other **R** packages.

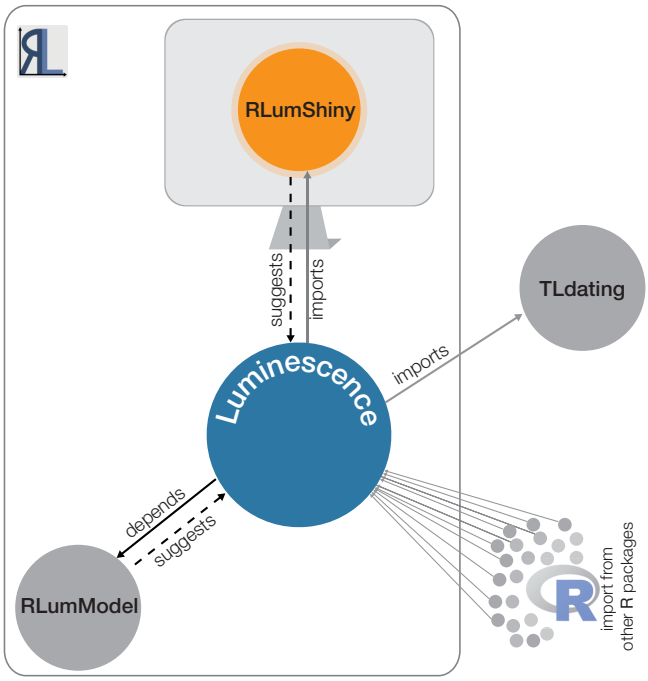

Figure 4. **R** package 'Luminescence' dependency sketch. For example: The package 'RLumShiny' is suggested by the package 'Luminescence' and therefore imported. Thus, the package 'Luminescence' can run without the package 'RLumShiny', but not the other way around.

To date, three packages import the 'Luminescence' package and using its core functionalities (e.g., data import, RLum-object structure). The package 'RLumShiny' (Burow et al., 2017) enhances the 'Luminescence' package by providing a graphical user interface for selected functions. 'RLumShiny' is not required to analyse data and is not installed by default while installing the 'Luminescence' package, but once installed it can improve the usability of supported functions tremendously (e.g., plot_AbanicoPlot()). The package 'RLumModel' (Friedrich et al., 2016, 2017) is a package to simulate quartz luminescence signals and it can be called out of the 'Luminescence' package using a so-called wrapper function (model_LuminescenceSignals()). To take advantage of its full functionality and for more complex use cases, the package functions should be called directly. The link established between both packages allows a very quick and focussed development of the 'RLumModel' package. It uses the entire object structure of the 'Luminescence' package, i.e., simulation outputs can be transferred to the analysis and plot functions of the 'Luminescence' package and can be treated as measurement output, i.e., as if it were produced by a luminescence reader. This connection makes writing custom plot and analysis functions for the 'RLumModel' package unnecessary and still allows a very efficient and independent development of the 'RLumModel' package, while keeping the model functions out of the 'Luminescence' package, where they are not needed for routine data analyses. The package 'TLdating' (Strebler, 2016; Strebler et al., 2016) took a different path. It imports functions (e.g., for plotting) from the 'Luminescence' package, but was further modified based on the underlying code structure of the 'Luminescence' package.

In all cases, the idea is similar concerning a preferred flexibility and task orientation of the packages, but avoiding a doubling of code wherever possible to reduce coding errors and to improve the overall reliability. The various possibilities to link **R** packages combined with the universally applicable object structure gives other package authors an independent platform for their projects without the need of taking care of the constraints provided by the 'Luminescence' package. For example, due to the complexity of 'Luminescence' not every function can be modified substantially without breaking other code and introducing errors.

5.4. Improving the code quality

Every new 'Luminescence' package release has passed internal tests performed by the programmers or by users working with the developer version from GitHub. But even after all of those tests, chance remains that bugs persist or that new bugs are introduced, which may lead to unexpected substantial errors or behaviours, i.e., the calculation output changes and remains undetected. And not every unexpected behaviour is a real bug, i.e. a coding mistake. For the package 'Luminescence', unexpected behaviour occurred with the change from package version 0.3.4 to 0.4.0. With the new version, the function calc_FiniteMixture() produced a different output; not because the function itself was changed, but the likelihood optimisation routine was taken from another package. The old and the new version gave different results. The results returned by the old version were not wrong, but the new results are considered to be more precise. Still, this behaviour remained unnoticed at first. Unexpected program behaviour and software errors can hardly be avoided, but can be reduced by adequate testing. Unfortunately, software testing is a tedious and time-consuming business that requires skilled users developing test scenarios. It is not the easily recognisable error in the graphical output that is most concerning, but the hard-to-track-down errors in basic calculations. Thus, errors may lead to a chain of misinterpretations and wrong scientific conclusions. Given that code errors will always exist, the aim is to recognise and reduce them. Figure 5 illustrates the development and testing process as implemented for the 'Luminescence' package (version $\geqslant$ 0.7.0).

First, the implementation of a new feature (e.g., new function) starts with a feature request (induced internally or externally). After that, the developer drafts the first version and runs tests until the feature appears sufficiently implemented. Before making the new function part of the pack-

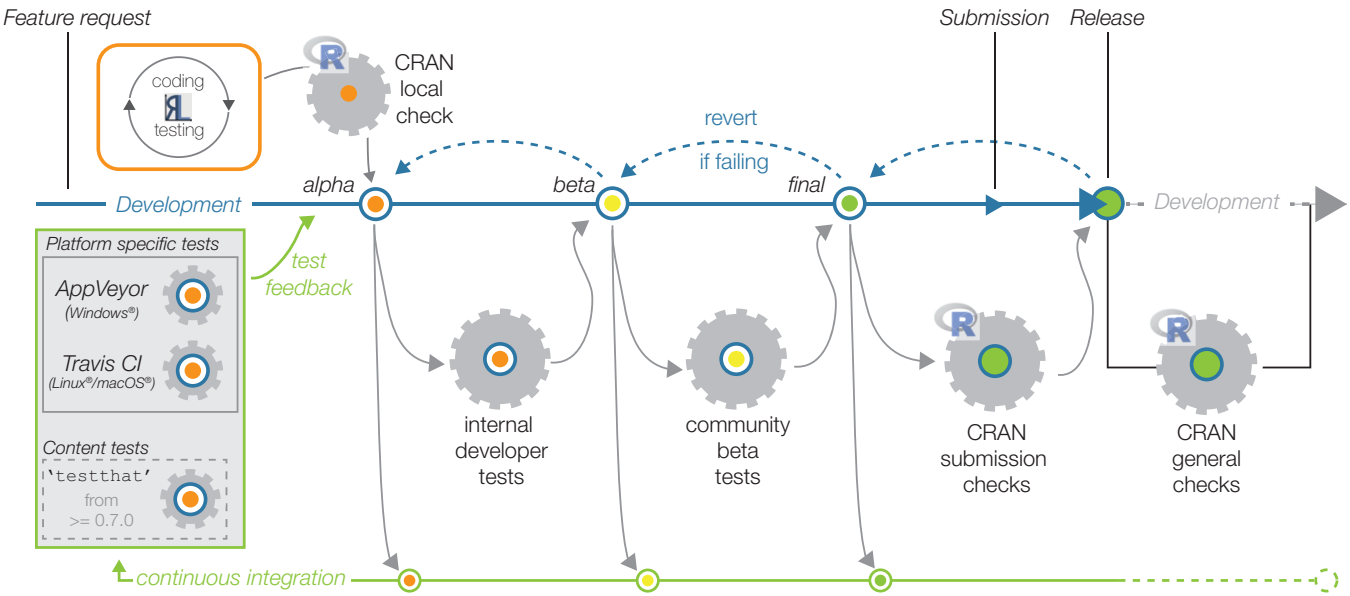

Figure 5. Development and testing process as implemented in the **R** package `'Luminescence'`. Once the development process has been initialised by a feature request (e.g., a new function), the new package version evolves over several development versions. Starting with an alpha version, stability and quality are improved until a final version, ready for submission to CRAN, is reached. The entire development process is supported by automatic platform and unit tests, which are run continuously.

age, local CRAN checks are performed, i.e., the same tests that are run by CRAN when submitting a package. If these tests were passed successfully, the entire package is send to two external resources, namely *AppVeyor*[8] and *Travis CI*[9], to conduct platform specific tests. Both resources are service platforms that provide continuous integration tests on virtual machines (virtual computers) for various platforms and are used in combination with GitHub. Currently, these services are free of charge for open-source projects.

The tests have the same intention as the local CRAN test, namely to run technical tests including the package examples, but they are run on different systems, newly set up, in a separate virtual machine for each test. On top of that, with version 0.7.0, special unit tests were defined and run using the package `'testthat'` (Wickham & RStudio, 2016; Wickham, 2011). In contrast to the tests before, complex test scenarios can be established and function output can be compared to predefined output values. For example, the function `calc_FiniteMixture()` can be run with predefined values while the output is compared against reference values. Any mismatch between calculated and pre-defined reference values causes an immediate test-error message.

Until this point, all tests (except in the first step, the developer test) run automatically. After passing, the new feature will be implemented in a first 'alpha' version. Now a test phase using "human resources" starts until the scheduled CRAN submission date. The CRAN submission can be compared with the submission of a scientific article for peer-review. The CRAN is volunteered by only a few members of the **R** community running further automated or semi-automated tests, though the review process only cares about technical aspects and takes usually not more than 24 h. After the package with the new feature is released on CRAN it is still continuously tested, i.e. as soon as a change in another package prevents the `'Luminescence'` package from functioning this would be recognised by the CRAN team.

6. Discussion

The previous section has demonstrated an increasingly complex system that is necessary to fulfil basic scientific standards, balancing transparency and reproducibility with enhanced tool functionality. The question is whether this effort is needed and justified.

Humans are imperfect, which justifies the established, peer-reviewed procedure in science. However, for software tools used in data analysis, this system appears to be underutilised. The required innovations and the dynamics of software development do not favour the long lasting peer-review procedures that every new version would require, and it would perhaps exhaust existing capacities. Take the package `'Luminescence'` as an example. It would have required a minimum of 25 publications and a minimum of up to twice as many reviewers within the last five years. Accordingly, to ensure basic scientific standards, we implemented the measures as discussed in Sec. 5.

Furthermore, tools of limited complexity may also only encounter problems of limited complexity. A pseudo LM-OSL curve conversion requires only a rather simple script, which can be easily validated by third parties. However, the

[8] https://www.appveyor.com
[9] https://travis-ci.org

additional benefit of such scripts remains limited. By contrast, a full equivalent dose determination routine requires a set of tools working together and thus becomes more complicated, but carries more overall value. The presented programming environment **R** favours simple scripts which can be combined to produce complex scenarios. Hence, the popularity of **R** within the scientific community might be partly explained by its package structure, which allows for aggregate packages to tackle rather complex tasks.

Nonetheless, **R** serves only as an example, and the drawbacks should not be overlooked. The nearly unlimited flexibility comes at the cost of lacking a native, easy-to-handle graphical user interface (GUI), which poses serious obstacles for beginners, as basic knowledge on the **R** environment is needed before packages can be used. And the advantage that **R** software is open for inspection by the user is of interest only to a small circle of **R** users. Even the package structure itself may partly cause a fragmentation of the system. Spreading needed functionalities across packages with limited compatibility gives no direct benefit to the user. Packages are not necessarily compatible with each other in terms of data exchange and functionality.

In contrast to the established peer-reviewed publishing procedure, software development is a far more agile and fast moving process, poorly suited to the established peer-review process. We may, therefore, advocate for a changed perception of scientific software developments. The correct and complete documentation of the applied tools would be an important first step. A minimum reporting standard for the tools used in luminescence dating studies should at least include the correct name of the tool, the version number (alternatively the release date) and an appropriate reference. If more complex procedures were applied (e.g., age models), used parameters might be provided as well to make the presented data comprehensible. To what extent such reporting is necessary depends on the particular case. For example, researchers may consider whether the raw data should be made available, or whether the computational work can be reproduced easily. Both discussions are beyond the scope of this article.

We furthermore suggest that every non-trivial piece of self-written software or script should be made freely accessible or published along with the study. Ideally, the source code itself would be open-source, provided under a commonly accepted open-source licence and available via platforms similar to those presented here (i.e., CRAN or GitHub).

7. Conclusions

The role of software for analysing luminescence data is increasingly important. In our article, we investigated the role of software in the luminescence dating community.

1. A literature screening was carried out, showing that only one-quarter of the screened articles reports on the software used for the data analysis. We argue for minimum reporting standards for the software applied to the data processing in a luminescence dating study, including the name of the tool, the version number and any relevant reference.

2. We listed software developed by the community to analyse luminescence data.

3. We explored the popularity of the statistical programming environment **R** and presented development concepts implemented for the **R** package `'Luminescence'`.

4. We explained how transparency, flexibility and reliability of the developed code and tools can be improved. To this end, the software development process for the `'Luminescence'` package was moved to an open repository and the software code is now largely tested automatically. The opening-up of the development process is believed to increase the transparency and reliability of the developed tools.

5. We suggest that the code of freely available, self-developed tools should be made accessible to the public.

Finally, we advocate for more attention to software developments, since they considerably influence the research from which scientific conclusions are drawn.

Acknowledgements

The manuscript benefitted considerably from suggestions by Rainer Grün and Nathan Brown. Johannes Friedrich is thanked for his help getting access to the *Boreas*' articles. Steve Grehl is thanked for fruitful discussions on object-based programming and questions on how to take advantage of this concept for the **R** package `'Luminescence'`. The work of Sebastian Kreutzer is financed by a programme supported by the ANR - n° ANR-10-LABX-52. Collaboration and personal exchange between the authors are gratefully supported by the DFG (SCHM 3051/3-1) in the framework of the programme Scientific Networks.

Appendix

*How do I cite an **R** package?*

R has an implemented functionality to get conclusive information on how a package should be cited and referenced. For the **R** package `'Luminescence'` this information will be shown by typing the following code line into the **R** terminal:

```
library(Luminescence)
citation("Luminescence", auto = TRUE)
```

References

Bailey, D J E. *Development of LM OSL Analysis Techniques for Applications to Optical Dating*. PhD thesis, Research Laboratory for Archæology and the History of Art Keble College, 2008.

Bailey, R M. *Towards a general kinetic model for optically and thermally stimulated luminescence of quartz*. Radiation Measurements, 33: 17–45, 2001.

Bluszcz, A and Adamiec, G. *Application of differential evolution to fitting OSL decay curves*. Radiation Measurements, 41(7-8): 886–891, 2006.

Boettiger, C. *An introduction to Docker for reproducible research*. ACM SIGOPS Operating Systems Review, 49(1): 71–79, 2015.

Brumby, S. *Regression analysis of ESR/TL dose-response data*. International Journal of Radiation Applications and Instrumentation. Part D. Nuclear Tracks and Radiation Measurements, 20 (4): 595–599, 1992.

Burow, C. *ESR: Package to analyse ESR data (under development)*. CRAN, version 0.1.0.9031, 2015. Developer version on GitHub: https://github.com/tzerk/ESR.

Burow, C, Kreutzer, S, Dietze, M, Fuchs, M C, Fischer, M, Schmidt, C, and Brückner, H. *RLumShiny - A graphical user interface for the R Package 'Luminescence'*. Ancient TL, 34: 22–32, 2016a.

Burow, C, Zens, J, Kreutzer, S, Dietze, M, Fuchs, M C, Fischer, M, Schmidt, C, and Brückner, H. *Exploratory data analysis using the R package 'Luminescence'. Towards data mining in OSL applications*. In *UK LED Liverpool, 2016-08-11 to 2016-08-13*, Liverpool, 2016b. Poster presentation.

Burow, C, Wolpert, U T, and Kreutzer, S. *RLumShiny: 'Shiny' Applications for the R Package 'Luminescence'*. CRAN, version 0.2.0, 2017. URL https://CRAN.R-project.org/package=RLumShiny. Developer version on GitHub: https://github.com/tzerk/RLumShiny.

Chacon, S and Straub, B. *Pro Git*. Everything you need to know about git. Apress, 2nd edition edition, 2014. web resource: https://git-scm.com.

Dietze, M, Kreutzer, S, Fuchs, M C, Burow, C, Fischer, M, and Schmidt, C. *A practical guide to the R package Luminescence*. Ancient TL, 31(1): 11–18, 2013.

Dietze, M, Kreutzer, S, Burow, C, Fuchs, M C, Fischer, M, and Schmidt, C. *The abanico plot: visualising chronometric data with individual standard errors*. Quaternary Geochronology, 31: 12–18, 2016.

Duller, G A T. *Assessing the error on equivalent dose estimates derived from single aliquot regenerative dose measurements*. Ancient TL, 25(1): 15–24, 2007.

Duller, G A T. *The Analyst software package for luminescence data: overview and recent improvements*. Ancient TL, 33(1): 35–42, 2015. URL http://users.aber.ac.uk/ggd/.

Durcan, J A, King, G E, and Duller, G A T. *DRAC: Dose Rate and Age Calculator for trapped charge dating*. Quaternary Geochronology, 28: 54–61, 2015. source code on GitHub: https://github.com/drac-calculator/DRAC-calculator.

Friedrich, J, Kreutzer, S, and Schmidt, C. *Solving ordinary differential equations to understand luminescence: 'RLumModel', an advanced research tool for simulating luminescence in quartz using R*. Quaternary Geochronology, 35(C): 88–100, 2016.

Friedrich, J, Kreutzer, S, and Schmidt, C. *RLumModel: Modelling Ordinary Differential Equations Leading to Luminescence*. CRAN, version 0.2.1, 2017. URL https://CRAN.R-project.org/package=RLumModel. Developer version on GitHub: https://github.com/R-Lum/RLumModel.

Fuchs, M C, Kreutzer, S, Burow, C, Dietze, M, Fischer, M, Schmidt, C, and Fuchs, M. *Data processing in luminescence dating analysis: An exemplary workflow using the R package 'Luminescence'*. Quaternary International, 362: 8–13, 2015.

Greilich, K-S, Harney, H L, Woda, C, and Wagner, G A. *AgesGalore—A software program for evaluating spatially resolved luminescence data*. Radiation Measurements, 41(6): 726–735, 2006.

Greilich, S. *AgesGalore: R routines for AgesGalore data*. 0.03, 2013. URL http://www.agesgalore.net.

Greilich, S, Gribenski, N, Mittelstraß, D, Dornich, K, Huot, S, and Preusser, F. *Single-grain dose-distribution measurements by optically stimulated luminescence using an integrated EMCCD-based system*. Quaternary Geochronology, 29(C): 70–79, 2015.

Grün, R. *The "AGE" program for the calculation of luminescence age estimates*. Ancient TL, 27(2): 45–46, 2009. URL to the *AGE* program: http://www.ecu.edu/cs-cas/physics/ancient-timeline/upload/ATL27-2_supplement_Grun.zip.

Grün, R and Macdonald, P D M. *Non-linear fitting of TL/ESR dose-response curves*. International Journal of Radiation Applications and Instrumentation. Part A. Applied Radiation and Isotopes, 40 (10-12): 1077–1080, 1989.

Hammer, Ø, Harper, D A T, and Ryan, P D. *PAST: Paleontological Statistics Software Package for Education and Data Analysis*. Palaeontologia Electronica, 4(1): 1–9, 2001. Software download: http://nhm2.uio.no/norlex/past/download.html.

Hornik, K and Zeilis, A. *Changes on CRAN*. The R Journal, 8(1): 402–403, 2016.

Howison, J and Bullard, J. *Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature*. Journal of the Association for Information Science and Technology, 67(9): 2137–2155, 2015.

Kreutzer, S. *rxylib: Import XY-Data into R*. CRAN, 0.1.1, 2017. URL https://CRAN.R-project.org/package=rxylib. Developer version on GitHub: https://github.com/R-Lum/rxylib.

Kreutzer, S, Schmidt, C, Fuchs, M C, Dietze, M, Fischer, M, and Fuchs, M. *Introducing an R package for luminescence dating analysis*. Ancient TL, 30(1): 1–8, 2012.

Kreutzer, S, Burow, C, Dietze, M, Fuchs, M C, Schmidt, C, Fischer, M, and Friedrich, J. *Luminescence: Comprehensive Luminescence Dating Data Analysis*. CRAN, version 0.7.5, 2017. URL https://CRAN.R-project.org/package=Luminescence. Developer version on GitHub: https://github.com/R-Lum/Luminescence.

Kulig, G. *Erstellung einer Auswertesoftware zur Altersbestimmung mittels Lumineszenzverfahren unter besonderer Berücksichtigung des Einflusses radioaktiver Ungleichgewichte in der 238-U-Zerfallsreihe*. Master's thesis, Inst. F. Informatik, TU Bergakademie Freiberg, 2005.

Lanos, P, Philippe, A, Lanos, H, and Dufrense, P. *Chronomodel : Chronological Modelling of Archaeological Data using Bayesian Statistics*. version 1.5, 2015. URL http://www.chronomodel.fr.

Lapp, T, Jain, M, Ankjærgaard, C, and Pirtzel, L. *Development of pulsed stimulation and Photon Timer attachments to the Risø TL/OSL reader*. Radiation Measurements, 44: 571–575, 2009.

Lapp, T, Jain, M, Thomsen, K J, Murray, A S, and Buylaert, J P. *New luminescence measurement facilities in retrospective dosimetry*. Radiation Measurements, 47: 803–808, 2012.

Martin, L, Incerti, S, and Mercier, N. *DosiVox: Implementing Geant 4-based software for dosimetry simulations relevant to luminescence and ESR dating techniques* . Ancient TL, 33(1): 1–10, 2015.

Peng, J. *tgcd: Thermoluminescence Glow Curve Deconvolution*. CRAN, version 2.0, 2016. URL https://CRAN.R-project.org/package=tgcd.

Peng, J and Li, B. *numOSL: Numeric Routines for Optically Stimulated Luminescence Dating*. CRAN, version 2.3, 2017. URL https://CRAN.R-project.org/package=numOSL.

Peng, J and Pagonis, V. *Simulating comprehensive kinetic models for quartz luminescence using the R program KMS*. Radiation Measurements, 86: 63–70, 2016. URL https://github.com/pengjunUCAS/KMS.

Peng, J, Dong, Z B, Han, F Q, Long, H, and Liu, X J. *R package numOSL: numeric routines for optically stimulated luminescence dating*. Ancient TL, 31(2): 41–48, 2013.

Peng, J, Dong, Z, and Han, F. *tgcd: An R package for analyzing thermoluminescence glow curves*. SoftwareX, 5: 112–120, 2016.

Philippe, A and Vibet, M-A. *ArchaeoPhases: Post-Processing of the Markov Chain Simulated by 'ChronoModel', 'Oxcal' or 'BCal'*. CRAN, version 1.2, 2017a. URL https://CRAN.R-project.org/package=ArchaeoPhases.

Philippe, A and Vibet, M-A. *RChronoModel: Post-Processing of the Markov Chain Simulated by ChronoModel*. CRAN, version 0.4, 2017b. URL https://CRAN.R-project.org/package=RChronoModel.

Ramsey, C B. *Radiocarbon Calibration and Analysis of Stratigraphy: The OxCal Program*. Radiocarbon, 37(2): 425–430, 1995.

Strebler, D. *TLdating: Tools for Thermoluminescences Dating*. CRAN, version 0.1.3, 2016. URL https://CRAN.R-project.org/package=TLdating. Developer version on GitHub: https://github.com/dstreble/TLdating.

Strebler, D. *LumReader: TL/OSL Reader simulator (under development)*. CRAN, 0.1.0, 2017. URL https://CRAN.R-project.org/package=LumReader. Developer version on GitHub: https://github.com/dstreble/LumReader.

Strebler, D, Burow, C, Brill, D, and Brückner, H. *Using R for TL dating*. Quaternary Geochronology, pp. 1–27, 2016.

Stuiver, M, Reimer, P J, and Reimer, R. *CALIB: Radiocarbon Calibration*. 7.1, 2016. URL http://calib.org/calib/.

Tippmann, S. *Programming tools: Adventures with R*. Nature, 517 (7532): 109–110, 2014.

Tsakalos, E, Christodoulakis, J, and Charalambous, L. *The Dose Rate Calculator (DRc) for Luminescence and ESR Dating-a Java Application for Dose Rate and Age Determination*. Archaeometry, 58(2): 347–352, 2015. Software download: http://www.ims.demokritos.gr/download/.

Vermeesch, P. *RadialPlotter: A Java application for fission track, luminescence and other radial plots*. Radiation Measurements, 44(4): 409–410, 2009. URL http://www.ucl.ac.uk/~ucfbpve/radialplotter/.

Wickham, H. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org.

Wickham, H. *testthat: Get Started with Testing*. R Journal, 3(1): 5–10, 2011.

Wickham, H and RStudio. *testthat: Unit Testing for R*. 1.02, 2016. URL https://CRAN.R-project.org/package=testthat.

Reviewer

Nathan Brown