baRcodeR: an open-source R package for sample labelling

Yihan Wu, David R. Lougheed, Stephen C. Lougheed, Kristy Moniz, Virginia K. Walker, and Robert I. Colautti*

Biology Department
Queen's University
116 Barrie St.
Kingston, ON K7L 3N6

*Corresponding Author Email: robert.colautti@queensu.ca

Keywords: Biological samples; asset tracking; barcodes; QR codes; reproducible science

Running Head: ***baRcodeR***

Abstract

1. Repeatable experiments with accurate data collection and reproducible analyses are fundamental to the scientific method but may be difficult to achieve in practice. Several flexible, open-source tools developed for the R coding environment aid the reproducibility of data analysis, but analogous tools are generally lacking for earlier stages of scientific inquiry, such as collecting, labelling, subsampling, and measuring biological samples.
2. As a step toward improving repeatability of methods for the curation and management of biological samples, we introduce the R package ***baRcodeR*** for generating biologically informative identifier (ID) codes with digitally encoded linear or 2D barcodes. Codes can be imported from an existing dataset (e.g. CSV file) or generated rapidly in ***baRcodeR***, producing scannable barcodes with customizable page layouts printing and scanning with consume-grade printers and scanners.
3. User-defined ID codes may contain a simple sequence (e.g. **`SAMPLE0427`**) or encode more meaningful sample information such as individual subjects, treatment groups, sample populations, time points, spatial coordinates, subsamples, or other associations (e.g. **`Pop22.Ind08.Time40`**). In addition to command-line functions, a graphical-user-interface (GUI) is available from the 'Addins' menu of R Studio.
4. ***baRcodeR*** can help biologists apply principles of open and reproducible science to collect and manage biological samples.

Introduction

Reproducibility is a foundation of the scientific method that is difficult to achieve in practice, contributing to a perceived 'crisis' in peer-reviewed research (Baker, 2016). A push towards more transparent, reproducible methods has inspired rapid development of new tools for data science, most notably R Markdown and R Notebooks with R Studio (RStudio Team, 2016) to generate reproducible analysis and reports in R (R Core Team, 2013). In contrast, tools to improve transparency and repeatability at earlier stages (e.g. data collection and management) have received relatively little attention. Repeatability at these earlier stages are particularly important for data sharing because all downstream treatments assume that published data are accurate within margins of reported error rates.

In many of the biological sciences, samples are collected under arduous field conditions that pose challenges for careful labelling, organizing, measuring and analyzing samples. A common approach is to collect and preserve samples for later analysis, which may include multiple users and movement to different locations, transferring to different storage media or vessels, subsampling, and archiving for further projects. Misinterpreted labels, ambiguous handwriting, cut-and-paste errors, typographical mistakes, and other problems that arise when humans associate samples with their downstream measurements can potentially go undetected and propagate through statistical analysis to compromise biological inference.

The use of computer-readable, printed labels can reduce human transcription errors and facilitate field sampling in ecology and evolution (Copp, Kennedy, & Muehlbauer, 2014). For example, imagine a field study in which we want to collect 30 to 50 individuals from each of 8 to 10 locations sampled at 3 different times, with specific numbers varying due to time, weather conditions, or other logistic constraints. Rather than hand-labelling each sample as it is collected, we could create a complete a list

of potential codes *a priori* by assuming an ideal sampling scenario (e.g. **LocA-t1-ind1** through **LocJ-t3-ind50**). Printed sheets of systematically-generated labels can allow for quick assessment of the number and type of samples collected and remaining in each group; unused barcodes can act as an additional check for missing samples long after sampling is complete. As an alternative to hierarchical labels, we could generate 1500 unique ID codes (ID0001 through ID1500) as well as 150 replicated location codes (50 individuals x 3 time points) and 500 time index codes (50 individuals x 10 locations). Upon collection, each sample would then get three labels: a unique ID label, a location label, and a time label. Additional labels could be added later, for example when allocating subsamples or when samples are moved to different locations.

As a tool to facilitate the curation and management of biological samples, we here introduce the ***baRcodeR*** package in R. Our package was designed to be (i) open-source and based in R, (ii) user-friendly, (ii) highly customizable, and (iii) inexpensive to implement at scale.

Package Overview

The ***baRcodeR*** program is an open source package for R (R Core Team, 2018) that was developed to address a need to quickly generate and print unique and informative barcodes for hundreds to thousands of samples according to a pre-defined sampling hierarchy (e.g. individuals from ten populations sampled from each of three regions and assigned to one of five treatments). We could not find any open source or commercial software packages with this flexibility, and working in the R environment enables flexible command-line operations that can quickly generate complex identifier (ID) labels (e.g. nested subsamples with user-defined codes).

We made several design choices to improve sampling for large collaborative research projects. Three main commands are available: (i) *uniqID_maker()* or (ii) *uniqID_hier_maker()* are functions to generate unique ID codes, followed by (iii) *create_PDF()* to output a portable document file (PDF) containing both

text labels and barcode images laid out in a customizable format suitable for consumer-grade printers. The command-line also includes a more user-friendly interactive mode implemented with the parameter *user=T*.

In addition to basic command-line functions, the ***baRcodeR*** package includes a ***shiny*** application (Chang, et al., 2018) for the ***R Studio*** software package (RStudio Team, 2016). This adds a 'baRcodeR GUI' option to the 'Addins' menu, which provides the user with point-and-click access to ***baRcodeR*** commands and options (see 'Graphical User Interface' section, below).

The package is available on the Comprehensive R Archive Network (CRAN), which includes a detailed tutorial vignette (Wu & Colautti, 2018a). A 2-page 'cheat-sheet' workflow summary with available commands and parameters is available on FigShare (Wu & Colautti, 2018b) and can be quickly accessed from the 'baRcodeR Cheatsheet' option in the 'Addins' menu of R Studio. A brief workflow overview is presented in Figure 1.

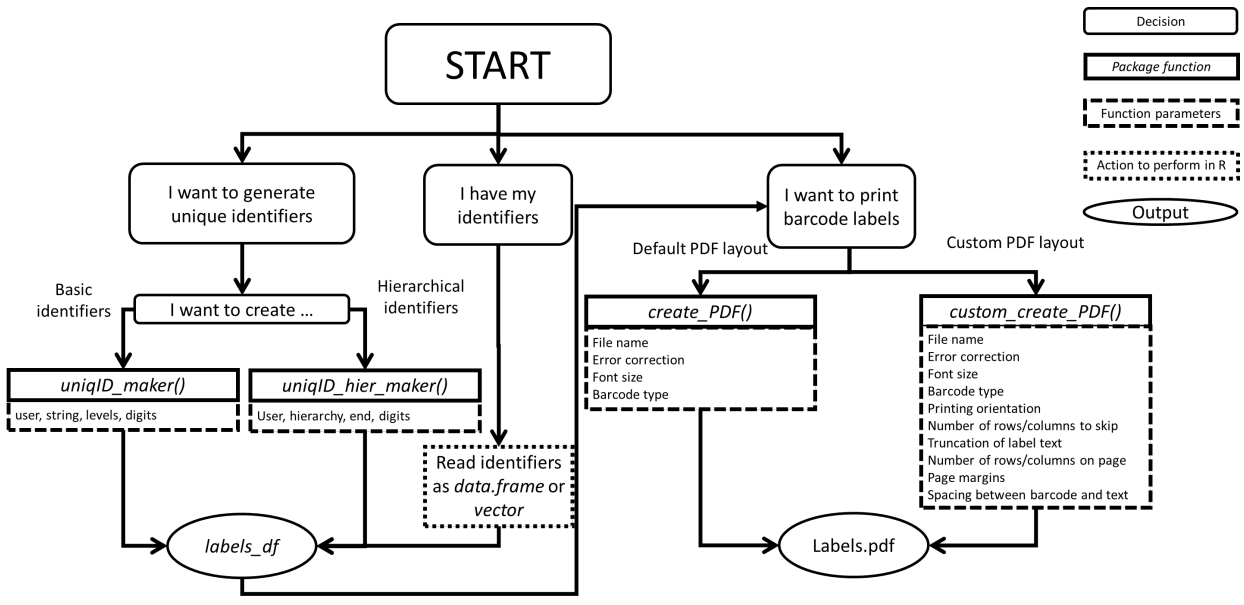

Figure 1. Flowchart of functions and workflow for the ***baRcodeR*** package, which includes options for generating unique identifier codes and formatting of labels that can be printed or scanned using consumer-grade hardware.

The latest release can be installed directly in R using the command:

```
> install.packages('baRcodeR')
```

Alternatively, a pre-release of the latest features can be installed from the development website on GitHub (https://github.com/yihanwu/baRcodeR), which includes the latest code and documentation. Installation can occur either by manually downloading and installing the binaries or using the ***devtools*** package:

```
> devtools::install_github('baRcodeR')
```

Step 1: Generate Unique ID Codes

To generate standardized and biologically informative ID codes for biological samples, *uniqID_maker()* or *uniqID_hier_maker()* are used for sequential or hierarchical text labels, respectively. The *uniqID_maker()* command is particularly useful to quickly produce a large number of sequential labels with the same user-defined prefix string (e.g. *Ex-1*, *Ex-2*, *Ex-3,*…). Additionally, it is possible to pass any sequence of numbers into *uniqID_maker()*. For example, the user can specify alternating numbers (e.g. *Ex-001*, *Ex-003*, *Ex-005*):

```
> uniqID_maker(string = 'Ex', level = seq(1, 5, 2))
```

or other custom sequences (e.g. *Ex-010*, *Ex-014*, *Ex-018*, *Ex-020*, *Ex-040*):

```
> uniqID_maker(string = 'Ex', level = c(seq(10,  20,  4),  20,  40))
```

or randomly ordered sequences:

```
> uniqID_maker(string = 'Ex', level = sample(1:10, replace = F))
```

The *uniqID_hier_maker()* command expands on *uniqID_maker()*, allowing for the creation of hierarchical labels with different user-defined strings and numeric sequences for each level of the hierarchy. This is useful, for example, to generate labels for replicated populations, genetic lines nested within populations, and repeated sampling or subsampling of individuals at different times, with randomly assigned treatment categories. A single command in R can quickly define labels for ten replicate individuals, each from one of three genetic lines, with subsamples of two different tissue types:

```
> uniqID_hier_maker(hierarchy = list( c('line', 1, 3),
        c('ind', 1, 10), c('sbsmpl', 1, 2)))
```

The examples of *uniqID_maker()* and *uniqID_hier_maker()* use non-interactive mode by default, but can also be run interactively in the command line with the *user = T* parameter. In this mode, the user is prompted for input:

```
> uniqID_maker(user = T)
Please enter string for level: Ex
Enter the starting number for level: 1
Enter the ending number for level: 10
Number of digits to print for level: 3
    label ind_string ind_number
1  Ex001         Ex          1
2  Ex002         Ex          2
3  Ex003         Ex          3
4  Ex004         Ex          4
```

5	Ex005	Ex	5
6	Ex006	Ex	6
7	Ex007	Ex	7
8	Ex008	Ex	8
9	Ex009	Ex	9
10	Ex010	Ex	10

Regardless of mode (i.e. interactive or non-interactive), the output of *uniqID_maker()* and *uniqID_hier_maker()* is a *data.frame* object containing (i) a vector of unique IDs and (ii) additional columns corresponding to each level of the sampling hierarchy (e.g. columns for group and individual ID codes). Like any *data.frame* object in R, the output can be renamed with *names()* function or merged with an existing database. The output object can also be saved to a standard text file using the *write.table()* or *write.csv()* functions. These text files can then be imported into a spreadsheet, database software, or integrated with a sample management software for data collection.

Step 2: Generate Labels

A *data.frame* containing a vector of ID labels is required input for the *create_PDF()* function. The input *data.frame* may be created by one of the functions above or supplied by the user, for example by importing ID codes as a CSV file into R using the *read.csv()* command.

The output of *create_PDF()* is a printable PDF file containing human-readable (plain text) IDs and digital barcode images. The layout can be customized for printing on consumer-grade printers and can be scanned with standard barcode scanners or image recognition software for cellphones or other devices (Fig. 2). The print layout can be customized using spacing parameters in *create_PDF()*, but default parameters create a PDF output file that will fit S-19297 labels (ULINE.ca or ULINE.com). This layout was chosen because the S-19297 labels are high-density (80 per page) weatherproof vinyl labels that can be printed on a standard laser printer. Our own tests demonstrate that printed labels do not fade or

degrade even after two years of storage at -80 °C or three years of exposure to sunlight in outdoor field experiments. Standard laser printer toner is similarly robust to ultraviolet light and freezing to ultralow temperatures. The label glue is less robust but transparent packing tape can be used to better secure the labels without affecting their function.

Figure 2. Examples of conventional, hand-labelled tubes (top panel), compared with ***baRcodeR***-generated labels of tubes (mid panel) and plastic markers (lower panel).

The *create_PDF()* function can be run interactively with the *user = T* parameter or in default non-interactive mode with user-defined parameters for barcode type, custom label layouts and other parameters like file name and font size. Simple, linear code 128 barcodes may be generated in addition to the 2D barcodes shown in Figure 2. Linear barcodes are encoded in ***baRcodeR*** following the Code 128 algorithm, whereas two-dimensional QR codes are generated by calling the the *qrcode()* function from

the ***qrcode*** package (Teh, 2015). Code 128 and QR codes were chosen because they are alphanumeric and can be scanned by most hardware capable of reading 2D barcodes, but future versions of ***baRcodeR*** could incorporate other digital barcode types as new coding algorithms become available for R. For example, the ***zintr*** package (https://github.com/carlganz/zintr) is a wrapper for the C++ zint library (http://zint.org.uk/) that provides access to many other encoding algorithms. However, zintr can be difficult for casual R users to implement as it has no repository on CRAN, it requires command-line installation of C libraries, and it generates a separate png file for each input code rather than an R vector-based object.

One parameter of note for 2D barcodes is the error correction level (L, M, Q or H), which represent a trade-off between barcode size and redundancy, ranging from larger, high-redundancy (H) to smaller low-redundancy (L) 2D barcodes. H-codes can be recognized by barcode scanners after losing up to 30% of their surface while L-codes labels are less robust but can be printed at a smaller size. Custom page layout, label layout and other printing options can be specified by passing in arguments through *create_PDF()* to *custom_create_PDF().* See the ***baRcodeR*** vignette for details (Wu & Colautti, 2018a).

As an example, the code below will create a custom page layout (Avery Address Labels 1" x 2-5/8", Catalog #8160; https://www.avery.com) which has 10 rows and 3 columns of labels on a page, a 3/16" horizontal page margin and 0.5" vertical page margin. In addition, we specify that the first label starts at the second column of the fourth row of labels. Starting further down the page like this is useful for re-using the same label sheet when only a few labels at a time:

```
> labels <- uniqID_maker(string = 'Ex', level = sample(1:10, replace = F))
> create_PDF(user=FALSE, Labels = labels, name = 'Avery_Layout', Fsz = 8,
      ERows = 3, ECols = 1, numrow = 10, numcol = 3, width_margin = 3/16,
      height_margin = 0.5, label_width = 2.625, label_height = 1)
```

Graphical User Interface

The text commands outlined above may be entered directly in the command line but can also be accessed via a graphical-user interface (GUI) available in R Studio after installing and loading the ***baRcodeR*** library. Details of the GUI are provided as a quick-start guide in the README file on the CRAN repository (Wu & Colautti, 2018a). The GUI add-in is a ***shiny*** application (Chang et al., 2018) that can be run directly from the 'baRcodeR GUI' option in the 'Addins' menu of the R Studio toolbar (Fig. 3). Three tabs in the GUI window correspond to three main functions: 'Simple Label Creation' calls *uniqID_maker()*, 'Hierarchical Label Creation' calls *uniqID_hier_maker()*, and 'Barcode Generation' calls *create_PDF()* (Fig. 1). Effect of parameters on codes and layout are previewed before final generation and will be automatically saved in the R working directory upon creation. Code snippets are provided for the user-specified parameter settings and can be exported and archived for repeatable experimental design and reproducible label creation – running the code snippet in the command line will reproduce the labels.

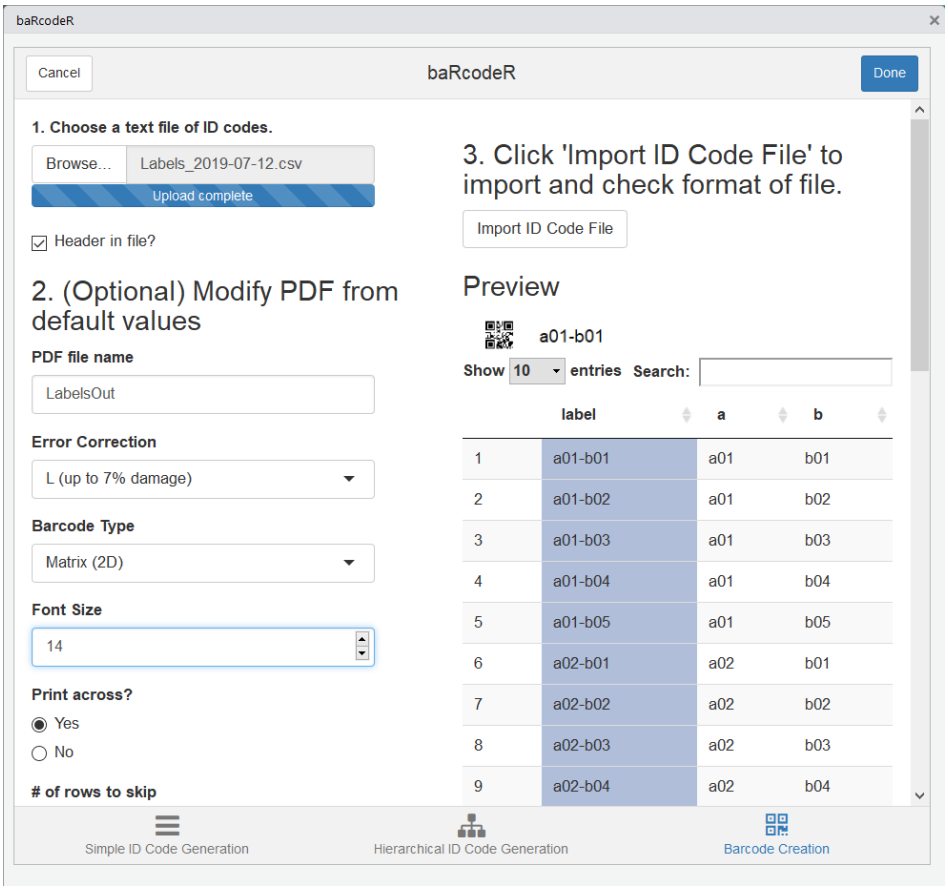

Figure 3. Screenshot of the ***baRcodeR*** graphical user interface showing an uploaded hierarchical label file with two levels, PDF output options, a preview of the label generated, and a table showing the selected column of labels.

In addition to using *create_PDF()* to generate printable barcodes, users can load saved labels into other barcode printers, such as those that print directly on sample tubes, plant markers, and other material. For example, the TubeWriter 360 (https://tubewriter.com/) and TubeMarker 2 (https://www.4ti.co.uk/) can generate and print barcodes from standard text files, allowing users to create ID codes in *uniqID_maker()* or *uniqID_hier_maker()* and saving the output to a text file using the *write.table()* or *write.csv()* functions in base R.

Once a *data.frame* or CSV file is created, additional headings can be added for data entry. Manual data entry is inherently susceptible to human error but enforcing and restricting entry formats can help to ensure repeatability of analysis. A variety of software is available to help reduce errors in data entry, ranging from fully-featured propriety software suites with optical recognition of hand-written data such as Viking Data Entry (http://vikingsoft.com), to open-access programs such as Epidata (Christiansen & Lauritsen, 2010) and OpenRefine (http://www.openrefine.org). Microsoft Excel, Libreoffice Calc, and Google Sheets also have data form creators for restricting data entry types. Within R Studio, interactive data entry can be performed using the *editData* add-in (Moon, 2017), which can be appended to the biological sample *data.frame* created from ***baRcodeR***.

Summary

The scientific method rests on the fundamental concept of repeatable observation. Many key and fundamental results of published research in the sciences and social sciences are not reproducible, due at least partly to insufficient published detail about experimental methods, data collection and statistical analysis. The use of open-source tools for documenting data wrangling and analysis in scientific studies are important steps towards improving reproducibility and transparency of data collection in published research. To complement these tools, we have introduced ***baRcodeR*** to improve transparency and repeatability upstream of data wrangling and analysis.

Acknowledgements

We thank E. Bao for contributions to ***baRcodeR*** code, Dr. P. V.C. de Groot for his efforts in the TSFN fish and BEARWATCH sampling as well as E. Jensen, R. Clemente-Carvalho, K. Flock, the Gjoa Haven, NU community residents, the Hunters and Trappers Association (HTA), the Government of Nunavut, Natural Sciences and Engineering Research Council (Canada) and large-scale Genome-Canada Projects, and

these projects' associated supporters. This work was partially funded by the Government of Canada through Genome Canada and the Ontario Genomics Institute (OGI-096 and OGI-123). We would like to acknowledge that Queen's University is situated on traditional Anishinaabe and Haudenosaunee territory and we are grateful to be able to be live and learn on these lands.

Author's Contributions

YW wrote the final ***baRcodeR*** package from initial code by RIC. All authors contributed ideas and writing to the manuscript and gave approval for publication.

Data Accessibility

baRcodeR is available on the Comprehensive R Archive Network (CRAN): https://cran.r-project.org/package=baRcodeR and GitHub: https://github.com/yihanwu/baRcodeR with 2-page 'cheat-sheet' outline available on FigShare: https://figshare.com/articles/baRcodeR_Cheat_Sheet/7043309

Literature Cited

Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature News*, *533*(7604), 452. doi:10.1038/533452a

Copp, A. J., Kennedy, T. A., & Muehlbauer, J. D. (2014). Barcodes are a useful tool for labeling and tracking ecological samples. *The Bulletin of the Ecological Society of America*, *95*(3), 293–300. doi:10.1890/0012-9623-95.3.293

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2018). *Shiny: Web Application Framework for R*. Retrieved from https://CRAN.R-project.org/package=shiny

Christiansen, T., & Lauritsen, J. (2010). *EpiData - Comprehensive Data Management and Basic Statistical Analysis System*. Retrieved from https://www.epidata.dk/credit.htm

Moon, K.-W. (2017). editData: 'RStudio' Addin for Editing a 'data.frame' (Version 0.1.2). Retrieved from https://CRAN.R-project.org/package=editData

R Core Team. (2013). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

RStudio Team. (2016). *RStudio: Integrated Development Environment for R*. Boston, MA: RStudio, Inc. Retrieved from http://www.rstudio.com/

Teh, V. (2015). qrcode: QRcode Generator for R (Version 0.1.1). Retrieved from https://CRAN.R-project.org/package=qrcode

Wu, Y., & Colautti, R. I. (2018a, October 26). baRcodeR Vignette. The Comprehensive R Archive Network (CRAN). Retrieved from https://cran.r-project.org/package=baRcodeR/baRcodeR.pdf

Wu, Y., & Colautti, R. I. (2018b, September 19). baRcodeR Cheat-Sheet. FigShare. Retrieved from https://figshare.com/articles/baRcodeR_Cheat_Sheet/7043309