Kurt Hornik and Duncan Murdoch and Achim Zeileis

Who Did What? The Roles of R Package Authors and How to Refer to Them

Who Did What? The Roles of R Package Authors and How to Refer to Them

by Kurt Hornik, Duncan Murdoch and Achim Zeileis

Abstract Computational infrastructure for representing persons and citations has been available in R for several years, but has been restructured through enhanced classes `"person"` and `"bibentry"` in recent versions of R. The new features include support for the specification of the roles of package authors (e.g. maintainer, author, contributor, translator, etc.) and more flexible formatting/printing tools among various other improvements. Here, we introduce the new classes and their methods and indicate how this functionality is employed in the management of R packages. Specifically, we show how the authors of R packages can be specified along with their roles in package 'DESCRIPTION' and/or 'CITATION' files and the citations produced from it.

R packages are the result of scholarly activity and as such constitute scholarly resources which must be clearly identifiable for the respective scientific communities and, more generally, today's information society. In particular, packages published by standard repositories can be regarded as *reliable sources* which can and should be referenced (i.e. cited) by scientific works such as articles or other packages. This requires conceptual frameworks and computational infrastructure for describing bibliographic resources, general enough to encompass the needs of communities with an interest in R. These needs include support for exporting bibliographic metadata in standardized formats such as BIBTEX (Berry and Patashnik, 2010), but also facilitating bibliometric analyses and investigations of the social fabric underlying the creation of scholarly knowledge.

The latter requires a richer vocabulary than commonly employed by reference management software such as BIBTEX, identifying *persons* and their *roles* in relation to bibliographic resources. For example, a thesis typically has an author and advisors. Software can have an (original) author and a translator to another language (such as from S to R). The maintainer of an R package is not necessarily an author.

In this paper, we introduce the base R infrastructure (as completely available in R since version 2.14.0) for representing and manipulating such scholarly data: objects of class `"person"` (hereafter, *person objects*) hold information about persons, possibly including their roles; objects of class `"bibentry"` (hereafter, *bibentry objects*) hold bibliographic information in enhanced BIBTEX style, ideally using person objects when referring to persons (such as authors or editors). Furthermore, we indicate how this functionality is employed in the management of R packages, in particular in their 'CITATION' and 'DESCRIPTION' files.

Persons and their roles

Person objects can hold information about an arbitrary positive number of persons. These can be obtained by one call to `person()` with list arguments, or by first creating objects representing single persons and combining these via `c()`.

Every person has at least one *given* name, and natural persons typically (but not necessarily, Wikipedia, 2012) have a *family* name. These names are specified by arguments `given` and `family`, respectively. For backward compatibility with older versions, there are also arguments `first`, `middle` and `last` with obvious (Western-centric) meaning; however, these are deprecated as not universally applicable: some cultures place the given after the family name, or use no family name.

An optional `email` argument allows specifying persons' email addresses, which may serve as unique identifiers for the persons.

Whereas person names by nature may be arbitrary non-empty character strings, the state of the art in library science is to employ standardized vocabularies for person roles. R uses the "Relator and Role" codes and terms (`http://www.loc.gov/marc/relators/relaterm.html`) from MARC (MAchine-Readable Cataloging, Library of Congress, 2012), one of the leading collections of standards used by librarians to encode and share bibliographic information. Argument `role` specifies the roles using the MARC codes; if necessary, the `comment` argument can be used to provide (human-readable) character strings complementing the standardized role information.

As an example (see Figure 1 and also `?person`), consider specifying the authors of package **boot** (Canty and Ripley, 2012): Angelo Canty is the first author (`"aut"`) who wrote the S original; Brian D. Ripley is the second author (`"aut"`), who translated the code to R (`"trl"`), and maintains the package (`"cre"`).

Note that the MARC definition of the relator term "Creator" is "Use for a person or organization responsible for the intellectual or artistic content of a work.", which maps nicely to the notion of the maintainer of an R package: the maintainer creates the package as a bibliographic resource for possible redistribution (e.g. via the CRAN package repository), and is the person (currently) responsible for

```
R> ## specifying the authors of the "boot" package
R> p <- c(
+    person("Angelo", "Canty", role = "aut", comment =
+      "S original, http://statwww.epfl.ch/davison/BMA/library.html"),
+    person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
+      comment = "R port, author of parallel support", email = "ripley@stats.ox.ac.uk"))
R> p

[1] "Angelo Canty [aut] (S original, http://statwww.epfl.ch/davison/BMA/library.html)"
[2] "Brian D. Ripley <ripley@stats.ox.ac.uk> [aut, trl, cre] (R port, author of parallel support)"

R> ## subsetting
R> p[-1]

[1] "Brian D. Ripley <ripley@stats.ox.ac.uk> [aut, trl, cre] (R port, author of parallel support)"

R> ## extracting information (as list for multiple persons)
R> p$given

[[1]]
[1] "Angelo"

[[2]]
[1] "Brian" "D."

R> ## extracting information (as vector for single person)
R> p[2]$given

[1] "Brian" "D."

R> ## flexible formatting
R> format(p, include = c("family", "given", "role"),
+     braces = list(family = c("", ","), role = c("[Role(s): ", "]")),
+     collapse = list(role = "/"))

[1] "Canty, Angelo [Role(s): aut]"           "Ripley, Brian D. [Role(s): aut/trl/cre]"

R> ## formatting for BibTeX
R> toBibtex(p)

[1] "Angelo Canty and Brian D. Ripley"

R> ## alternative BibTeX formatting "by hand"
R> paste(format(p, include = c("family", "given"),
+    braces = list(family = c("", ","))), collapse = " and ")

[1] "Canty, Angelo and Ripley, Brian D."
```

Figure 1: Example for `person()`, specifying the authors of the **boot** package.

the package, being the "interface" between what is inside the package and the outside world, and warranting in particular that the license and representation of intellectual property rights are appropriate. For the persons who originally "created" the package code, typically author roles should be employed.

In general, while all MARC relator codes are supported, the following usage is suggested when giving the roles of persons in the context of authoring R packages:

`"aut"` (Author): Full authors who have made substantial contributions to the package and should show up in the package citation.

`"com"` (Compiler): Persons who collected code (potentially in other languages) but did not make further substantial contributions to the package.

`"ctb"` (Contributor): Authors who have made smaller contributions (such as code patches etc.) but should not show up in the package citation.

"cph" (Copyright holder): Copyright holders.

"cre" (Creator): Package maintainer.

"ths" (Thesis advisor): Thesis advisor, if the package is part of a thesis.

"trl" (Translator): Translator to R, if the R code is a translation from another language (typically S).

Person objects can be subscripted by field (using `$`) or by position (using `[`), and allow for flexible formatting and printing. By default, email, role and comment entries are suitably "embraced" by enclosing with angle brackets, square brackets and parentheses, respectively, and suitably collapsed. There is also a `toBibtex()` method which creates a suitable BIBTEX representation. Finally, the default method for `as.person()` is capable of inverting the default person formatting, and also can handle formatted person entries collapsed by comma or '`and`' (with appropriate white space). See Figure 1 and `?person` for various illustrative usages of these methods.

R code specifying person objects can also be used in the `Authors@R` field of 'DESCRIPTION' files. This information is used by the R package management system (specifically, `R CMD build` and `R CMD INSTALL`) to create the `Author` and `Maintainer` fields from `Authors@R`, unless these are still defined (statically) in 'DESCRIPTION'. Also, the person information is used for reliably creating the author information in auto-generated package citations (more on this later). Note that there are no default roles for `Authors@R`: all authors must be given an author role ("aut"), and the maintainer must be identified by the creator role ("cre").

Bibliographic information

Bibentry objects can hold information about an arbitrary positive number of bibliography entries. These can be obtained by one call to `bibentry()` with list arguments, or by first creating objects representing single bibentries and combining these via `c()`.

Bibentry objects represent bibliographic information in "enhanced BIBTEX style", using the same entry types (such as 'Article' or 'Book') and field names (such as 'author' and 'year') as BIBTEX. A single bibentry is created by calling `bibentry()` with arguments `bibtype` (typically used positionally) and `key` (optionally) giving the type of the entry and a key for it, and further fields given in *tag* `=` *value* form, with *tag* and *value* the name and value of the field, respectively. See `?bibentry` for details on types and fields. The author and editor field values can be person objects as discussed above (and are coerced to such using `as.person()` if they are not). In addition, it is possible to specify header and footer information for the individual entries (arguments `header` and `footer`) and the collection of entries (arguments `mheader` and `mfooter`).

Bibentry objects can be subscripted by field (using `$`) or by position (using `[`), and have `print()` and `format()` methods providing a choice between at least six different styles (as controlled by a `style` argument): plain text, HTML, LATEX, BIBTEX, R (for formatting bibentries as (high-level) R code), and citation information (including headers/footers and a mixture of plain text and BIBTEX). The first three make use of a `.bibstyle` argument using the `bibstyle()` function in package **tools** which defines the display style of the citation by providing a collection of functions to format the individual parts of the entry. Currently the default style "JSS", which corresponds to the bibliography style employed by the *Journal of Statistical Software* (http://www.jstatsoft.org/), is the only available style, but it can be modified by replacing the component functions, or replaced with a completely new style. A style is implemented as an environment, and is required to contain functions to format each type of bibliographic entry (`formatArticle()`, `formatBook()`, etc.) as well a function `sortKeys()` to sort entries. In addition to these required functions, the "JSS" style includes several dozen auxiliary functions that are employed by the required ones. Users modifying it should read the help page `?tools::bibstyle`, which includes an example that changes the sort order. Those implementing a completely new style should consult the source code.

Figure 2 shows an illustrative example for using `bibentry()`: A vector with two bibentries is created for the **boot** package and the book whose methods it implements. In the first bibentry the person object `p` from Figure 1 is reused while the second bibentry applies `as.person()` to a string with author and role information. Subsequently, the bibentry is printed in "JSS" style and transformed to BIBTEX. Note that the roles of the persons in the author field are *not* employed in `toBibtex()` as BIBTEX does not support that concept. (Hence, the role information in the `as.person()` call creating `b[2]` would not have been necessary for producing BIBTEX output.)

Currently, the primary use of bibentry objects is in package 'CITATION' files, which contain R code for defining such objects.[1] Function `citation()` collects all of the defined bibentry objects into a single object for creating package citations, primarily for visual inspection and export to BIBTEX. (Technically, `citation()` creates an object of class "citation" inheriting from "bibentry", for which the `print()`

[1]Typically, the R code should contain `bibentry()` calls but special constructs for reusing the information from the 'DESCRIPTION' are also available as is some legacy functionality (`citEntry()` etc.). More details are provided in the following.

```
R> b <- c(
+     bibentry(
+       bibtype = "Manual",
+       title = "{boot}: Bootstrap {R} ({S-Plus}) Functions",
+       author = p,
+       year = "2012",
+       note = "R package version 1.3-4",
+       url = "http://CRAN.R-project.org/package=boot",
+       key = "boot-package"
+     ),
+
+     bibentry(
+       bibtype = "Book",
+       title = "Bootstrap Methods and Their Applications",
+       author = "Anthony C. Davison [aut], David V. Hinkley [aut]",
+       year = "1997",
+       publisher = "Cambridge University Press",
+       address = "Cambridge",
+       isbn = "0-521-57391-2",
+       url = "http://statwww.epfl.ch/davison/BMA/",
+       key = "boot-book"
+     )
+  )
R> b

Canty A and Ripley BD (2012). _boot: Bootstrap R (S-Plus) Functions_. R package version
1.3-4, <URL: http://CRAN.R-project.org/package=boot>.

Davison AC and Hinkley DV (1997). _Bootstrap Methods and Their Applications_. Cambridge
University Press, Cambridge. ISBN 0-521-57391-2, <URL:
http://statwww.epfl.ch/davison/BMA/>.

R> ## formatting for BibTeX
R> toBibtex(b[1])

@Manual{boot-package,
  title = {{boot}: Bootstrap {R} ({S-Plus}) Functions},
  author = {Angelo Canty and Brian D. Ripley},
  year = {2012},
  note = {R package version 1.3-4},
  url = {http://CRAN.R-project.org/package=boot},
}
```

Figure 2: Example for `bibentry()`, specifying citations for the **boot** package (i.e. the package itself and the book which it implements).

method uses a different default style.)

Package citations can also be auto-generated from the package 'DESCRIPTION' metadata, see Figure 3 for an example: The bibentry title field is generated from the package metadata fields `Package` and `Title` which should be capitalized (in title case), and not use markup or end in a period. (Hence, it may need manual touch-ups, e.g. protecting `{S}` or `{B}ayesian` etc.) The year is taken from `Date/Publication` (for CRAN packages) or `Date` (if available), and the note field is generated from the package's `Version`. If the package was installed from CRAN, the official stable CRAN package URL (here: `http://CRAN.R-project.org/package=boot`) is given in the url field.[2] Finally, the bibentry author field is generated from the description `Author` field (unless there is an `Authors@R`, see below). As the `Author` field in 'DESCRIPTION' is a plain text field intended for human readers, but not necessarily for automatic processing, it may happen that the auto-generated BIBTEX is incorrect (as in Figure 3). Hence, `citation()` provides an ATTENTION note to this

[2] Note that the current physical URLs, e.g. `http://CRAN.R-project.org/web/packages/boot/index.html` are not guaranteed to be stable and may change in the future (as they have in the past).

```
R> citation("boot", auto = TRUE)

To cite package 'boot' in publications use:

  Angelo Canty and Brian Ripley (2012). boot: Bootstrap Functions (originally by Angelo
  Canty for S). R package version 1.3-4. http://CRAN.R-project.org/package=boot

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {boot: Bootstrap Functions (originally by Angelo Canty for S)},
    author = {Angelo Canty and Brian Ripley},
    year = {2012},
    note = {R package version 1.3-4},
    url = {http://CRAN.R-project.org/package=boot},
  }
```

Figure 3: Citation for the **boot** package auto-generated from its 'DESCRIPTION' file. (Note that this is not the recommended citation of the package authors, see `citation("boot")` without `auto = TRUE` for that.)

end. One can overcome this problem by providing an `Authors@R` field in 'DESCRIPTION', from which the names of all persons with author roles can reliably be determined and included in the author field. Note again that author roles (`"aut"`) are not "implied" as defaults, and must be specified explicitly.

As illustrated in Figure 3, one can inspect the auto-generated package citation bibentry by giving `auto = TRUE` when calling `citation()` on a package – even if the package provides a 'CITATION' file in which case `auto` defaults to `FALSE`. The motivation is that it must be possible to refer to the package *itself*, even if the preferred way of citing the package in publications is *via* the references given in the package 'CITATION' file. Of course, both approaches might also be combined as in the **boot** package where one of the bibentries in 'CITATION' corresponds essentially to the auto-generated package bibentry (see `citation("boot")`). To facilitate incorporating the auto-generated citations into 'CITATION', one can include a `citation(auto = meta)` entry: when evaluating the code in the 'CITATION' file, `citation()` automagically sets `meta` to the package metadata. This eliminates the need for more elaborate programmatic constructions of the package citation bibentry (or even worse, manually duplicating the relevant package metadata), provided that auto-generation is reliable (i.e. correctly processes author names and does not need additional LaTeX markup or protection in the title). To check whether this is the case for a particular package's metadata and hence `citation(auto = meta)` can conveniently be used in 'CITATION', one can employ constructs such as `citation(auto = packageDescription("boot"))`.

Actions for package maintainers

Package maintainers should provide an `Authors@R` entry in their 'DESCRIPTION' files to allow computations on the list of authors, in particular to ensure that auto-generation of bibliographic references to their packages works reliably. Typically, the `Author` and `Maintainer` entries in 'DESCRIPTION' can then be omitted in the sources (as these are automatically added by `R CMD build`).

Packages wanting to employ the auto-generated citation as (one of) the officially recommended reference(s) to their package can use the new simplified `citation(auto = meta)` to do so.[3] Note that if this is used then the package's citation depends on R at least 2.14.0, suggesting to declare this requirement in the `Depends` field of 'DESCRIPTION'.

As of 2012-04-02, CRAN had 3718 packages in total, with about 20% (734) having 'CITATION' files, but less than 2.5% (92) already providing an `Authors@R` entry in their package metadata: so clearly, there is substantial room for future improvement.

Conclusions and outlook

R has provided functionality for representing persons and generating package citations for several years now. Earlier versions used 'CITATION' files with calls to `citEntry()` (and possibly `citHeader()` and `citFooter()`). R 2.12.0 introduced the new bibentry functionality for improved representation and manipulation of bibliographic information (the old mechanism is still supported and implemented by the new one). R 2.12.0 also added the new per-

[3]Thus, the information from the 'DESCRIPTION' can be reused in the 'CITATION'. However, the reverse is not possible as the 'DESCRIPTION' needs to be self-contained.

son functionality (changing to the more appropriate given/family scheme and adding roles). R 2.14.0 added more functionality, including the `Authors@R` field in package 'DESCRIPTION' files from which `Author` and `Maintainer` fields in this file as well as package bibentries can reliably be auto-generated.

Having reliable comprehensive information on package "authors" and their roles which can be processed automatically is also highly relevant for the purpose of understanding the roles of persons in the social fabric underlying the remarkable growth of the R package multiverse, opening up fascinating possibilities for research on R (of course, with R), e.g. using social network analysis and related methods.

Since R 2.10.0, R help files have been able to contain R code in `\Sexpr` macros to be executed before the help text is displayed. In the future, it is planned to use this facility to auto-generate `\references` sections based on citations in the text, much as LaTeX and BibTeX can automatically generate bibliographies.

These enhancements will clearly make dealing with bibliographic information in Rd files much more convenient and efficient, and eliminate the need to "manually" (re)format entries in the `\references` sections. Of course, this should not come at the price of needing to manually generate bibentries for references already available in other bibliographic formats. For BibTeX format '.bib' databases (presumably the format typically employed by R users), conversion is straightforward: one can read the entries in '.bib' files into bibentry objects using `read.bib()` in CRAN package **bibtex** (François, 2011), use `format()` with `style = "R"` to obtain a character vector with `bibentry()` calls for each bibentry, and use `writeLines()` to write (or add) this to a bibentry database file (see `?bibentry` for examples of the last two steps). For other formats, one could first transform to BibTeX format: e.g. Bibutils (Putnam, 2010) can convert between the bibliography formats COPAC, EndNote refer, EndNote XML, Pubmed XML, ISI Web of Science, US Library of Congress MODS XML, RIS, and Word 2007. However, this may lose useful information: e.g. the MODS format can provide role data which (as discussed above) will be dropped when converting to BibTeX format. We are thus planning to provide a **bibutils** package which instead converts to the MODS XML format first, and creates bibentry objects from this. However, as this functionality requires both external software and package **XML** (Temple Lang, 2012), it cannot be made available within base R.

With these functionalities in place, managing bibliographic information in R documentation files will become much more convenient, and R will become a much sharper knife for cutting-edge bibliometric and scientometric analyses. In particular, it will become possible to programmatically analyze package citations, and use these to learn about the processes of knowledge dissemination driving the creation of R packages, and the relatedness of package authors and concepts.

Bibliography

K. Berry and O. Patashnik. BibTeX, 2010. URL `http://www.ctan.org/tex-archive/biblio/bibtex/base`.

A. Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2012. URL `http://CRAN.R-project.org/package=boot`. R package version 1.3-4.

R. François. *bibtex:* BibTeX *Parser*, 2011. URL `http://CRAN.R-project.org/package=bibtex`. R package version 0.3-0.

Library of Congress. MARC standards. URL `http://www.loc.gov/marc/`, accessed 2012-04-02, 2012.

C. Putnam. *Bibutils: Bibliography Conversion Utilities*. The Scripps Research Institute, 2010. URL `http://sourceforge.net/p/bibutils/`.

D. Temple Lang. *XML: Tools for Parsing and Generating XML within R and S-Plus*, 2012. URL `http://CRAN.R-project.org/package=XML`. R package version 3.9-4.

Wikipedia. Personal name – Wikipedia, the free encyclopedia. URL `http://en.wikipedia.org/wiki/Personal_name`, accessed 2012-04-02, 2012.

Kurt Hornik
Department of Finance, Accounting and Statistics
WU Wirtschaftsuniversität Wien
Augasse 2–6, 1090, Wien
Austria
`Kurt.Hornik@R-project.org`

Duncan Murdoch
Department of Statistical and Actuarial Sciences
University of Western Ontario
London, Ontario
Canada
`murdoch@stats.uwo.ca`

Achim Zeileis
Faculty of Economics and Statistics
Universität Innsbruck
Universitätsstr. 15, 6020 Innsbruck
Austria
`Achim.Zeileis@R-project.org`