Package 'rbibutils'

November 7, 2025

Type Package

Title Read 'Bibtex' Files and Convert Between Bibliography Formats

Version 2.4

Description Read and write 'Bibtex' files. Convert between bibliography formats, including 'Bibtex', 'Biblatex', 'PubMed', 'Endnote', and 'Bibentry'. Includes a port of the 'bibutils' utilities by Chris Putnam <https://sourceforge.net/projects/bibutils/>. Supports all bibliography formats and character encodings implemented in 'bibutils'.

License GPL-2

URL https://geobosh.github.io/rbibutils/ (doc), https://github.com/GeoBosh/rbibutils (devel)

BugReports https://github.com/GeoBosh/rbibutils/issues

Depends R (>= 2.10)

Imports utils, tools

Suggests testthat

Encoding UTF-8

NeedsCompilation yes

Config/Needs/memcheck devtools, rcmdcheck

Author Georgi N. Boshnakov [aut, cre] (R port, R code, new C code and modifications to bibutils' C code, conversion to Bibentry (R and C code), comment.ORCID: 0000-0003-2839-346X),
Chris Putman [aut] (src/*, author of the bibutils libraries, https://sourceforge.net/projects/bibutils/),
Richard Mathar [ctb] (src/addsout.c),
Johannes Wilm [ctb] (src/biblatexin.c, src/bltypes.c),
R Core Team [ctb] (base R's bibentry and bibstyle implementation)

Maintainer Georgi N. Boshnakov <georgi.boshnakov@manchester.ac.uk>

Repository CRAN

Date/Publication 2025-11-07 16:20:02 UTC

Contents

`rbibutils-package` *Read 'Bibtex' Files and Convert Between Bibliography Formats*

Description

Read and write 'Bibtex' files. Convert between bibliography formats, including 'Bibtex', 'Biblatex', 'PubMed', 'Endnote', and 'Bibentry'. Includes a port of the 'bibutils' utilities by Chris Putnam <https://sourceforge.net/projects/bibutils/>. Supports all bibliography formats and character encodings implemented in 'bibutils'.

Details

Package **rbibutils** provides an R port of the `bibutils` programs plus additional facilities. The main function, `bibConvert`, offers all conversions between bibliography formats supported by library `bibutils`. In addition, package **rbibutils** converts to and from R's `bibentry` Bibtex-based bibliography format.

The core functionality is provided by the `bibutils` programs which convert between various bibliography formats using a common MODS XML intermediate format, see the source cited below.

Currently we provide the function `bibConvert` for conversion between supported bibliography formats. For complete list of formats supported by the package, see the documentation of the original bibutils library.

`readBib` and `writeBib` import/export BiBTeX files. `readBibentry` and `writeBibentry` import/export R source files in which the references are represented by `bibentry()` calls. These functions were originally just wrappers around `bibConvert`. `readBib` has acquired additional features, including a direct import (without going through `bibConvert`) from BiBTeX files.

All encodings supported by the bibutils library are available for `bibConvert`.

Further functionality may be provided in future releases, in particular, the underlying C functions could be exposed to package authors. Further R wrappers may be added, as well. However, the scope of the package will remain conversion between formats based on `bibutils` and manipulation of the MODS XML intermediate format. **rbibutils** can be used also as an alternative to package **bibtex** (Francois 2020). For bibliography management see package **RefManager** (McLean 2017). For citations in R documentation (Rd or roxygen2) see package **Rdpack** (Boshnakov 2020).

Supported input and output formats

Most formats are supported for both input and output, see the listings below. A format supported for input can be converted to any of the output formats.

The input is first converted to *MODS XML intermediate*, the latter is then converted to the requested output format. In **rbibutils** there are currently two exceptions to this rule. First, the conversion from bibtex to bibentry offers the option to bypass the conversion to *MODS XML intermediate* and parse directly the bibtex file, see `readBib` for details. Second, the conversion from `bibentry` to `BibTeX` just uses a `print` method provided by R.

In the table below column Abbreviation shows the abbreviation for arguments `informat` and `outformat`, column FileExt gives the default file extension for that format, column Input (Output) contains TRUE if the format is supported for input (output) and FALSE otherwise. Column Description gives basic description of the format.

Abbreviation	FileExt	Input	Output	Description
ads	ads	FALSE	TRUE	ADS reference format
bib	bib	TRUE	TRUE	BibTeX
bibtex	bibtex	TRUE	TRUE	BibTeX
biblatex	biblatex	TRUE	TRUE	BibLaTeX
copac	copac	TRUE	FALSE	COPAC format references
ebi	ebi	TRUE	FALSE	EBI XML
end	end	TRUE	TRUE	EndNote (Refer format)
endx	endx	TRUE	FALSE	EndNote XML
isi	isi	TRUE	TRUE	ISI web of science
med	med	TRUE	FALSE	Pubmed XML references
nbib	nbib	TRUE	TRUE	Pubmed/National Library of Medicine nbib format
ris	ris	TRUE	TRUE	RIS format
R, r, Rstyle	R	TRUE	TRUE	R source file containing bibentry commands
rds	rds	TRUE	TRUE	bibentry object in a binary file created by saveRDS()
xml	xml	TRUE	TRUE	MODS XML intermediate
wordbib	wordbib	TRUE	TRUE	Word 2007 bibliography format

`bibentry` is the native R variant of BibTeX. It can be input directly from an R source file or from a binary `rds` file. The `"rds"` format is a compressed binary format. The `rds` file should contain a `bibentry` R object, saved from R with `saveRDS`. An R source file should contain one or more bibentry instructions, see `readBibentry` for details of the contents.

A `bibentry` object can be written to a file as a binary (`"rds"`) object or as an R source file, see `bibConvert` and `writeBib` for details.

ADS is the reference format of the Smithsonian Astrophysical Observatory (SAO) and National Aeronautics and Space Administration (NASA) Astrophyics Data System.

For COPAC, see `https://en.wikipedia.org/wiki/Copac`.

Note

The `bibutils` library is included in a number of software packages. These include include pandoc and a library for Haskell. Executable programs for conversion are available for Linux distributions but seem not easily available for Windows. Executable and libraries can also be generated out-of-the-box from the `bibutils` disribution (on Windows under MSYS).

rbibutils adds conversions to/from R's bibentry format and direct conversion from bibtex, which preserves non-standard fields from the bibtex source. There is also improved support for mathematical expressions in bibtex files.

Author(s)

Georgi N. Boshnakov (R code and R port of bibutils), Chris Putnam (author of bibutils library)

References

Georgi N Boshnakov (2020). "Rdpack: Update and Manipulate Rd Documentation Objects." doi:10.5281/zenodo.3925612, R package version 2.0.0.

Damiano Fantini (2019). "easyPubMed: Search and Retrieve Scientific Publication Records from PubMed." R package version 2.13, https://CRAN.R-project.org/package=easyPubMed.

Romain Francois (2014). *bibtex: bibtex parser*. R package version 0.4.0.

Mathew William McLean (2017). "RefManageR: Import and Manage BibTeX and BibLaTeX References in R." *The Journal of Open Source Software*. doi:10.21105/joss.00338.

Chris Putnam (2020). "Library bibutils, version 6.10." https://sourceforge.net/projects/bibutils/.

See Also

`bibConvert` for further `details` and `examples`

`readBib`,

`charToBib`,

`readBibentry`, `writeBibentry`

`bibConvert` *Convert between bibliography formats*

Description

Read a bibliography file in one of the supported formats, convert it to another format, and write that to a file.

Usage

```
bibConvert(infile, outfile, informat, outformat, ..., tex, encoding,
           options)
```

Arguments

infile	input file, a character string.
outfile	output file, a character string.
informat	input format, a character string, see sections “Supported formats” and “Details”.
outformat	output format, a character string, see sections “Supported formats” and “Details”.
...	not used.
tex	TeX/LaTeX specific options, see section “Details”, a character vector.
encoding	character(2), a length two vector specifying input and output encodings. Default to both is "utf8", see section “Details”.
options	mainly for debugging: additional options for the converters, see section “Details”.

Details

Arguments informat and outformat can usually be omitted, since bibConvert infers them from the extensions of the names of the input and output files, see section "File extensions" below. However, there is ambiguity for the extension "bib", since it is used for Bibtex and BibLaTeX entries. For this extension, the default for both, informat and outformat, is "bibtex".

Package **rbibutils** supports format "bibentry", in addition to the formats supported by the bibutils library. A bibentry object contains one or more references. Two formats are supported for "bibentry" for both input and output. A bibentry object previously saved to a file using saveRDS (default extension "rds") or an R source file containing one or more bibentry commands. The "rds" file is just read in and should contain a bibentry object.

When bibconvert outputs to an R source file, two variants are supported: "R" and "Rstyle". When (outformat = "R", there is one bibentry call for each reference, just as in a Bibtex file, each reference is a single entry. outformat = "Rstyle" uses the format of print(be, style = "R"), i.e., the bibentry calls are output as a comma separated sequence wrapped in c(). For input, it is not necessary to specify which variant is used.

Note that when the input format and output formats are identical, the conversion is not necessarilly a null operation (except for xml, and even that may change). For example, depending on the arguments the character encoding may change. Also, input BibTeX files may contain additional instructions, such as journal abbreviations, which are expanded and incorporated in the references but not exported. It should be remembered also that there may be loss of information when converting from one format to another.

For a complete list of supported bibliography formats, see section “Supported formats” below. The documentation of the original bibutils library (Putnam 2020) gives further details.

Argument encoding is a character vector containing 2 elements, specifying the encoding of the input and output files. If the encodings are the same, a length one vector can be supplied. The default encodings are UTF-8 for input and output. A large number of familiar encodings are supported, e.g. "latin1" and "cp1251" (Windows Cyrillic). Some encodings have two or more aliases and they are also accepted. If an unknown encoding is requested, a list of all supported encodings will be printed.

Argument tex is an unnamed character vector containing switches for bibtex input and output (mostly output). Currently, the following are available:

uppercase write bibtex tags/types in upper case.

no_latex do not convert latex-style character combinations to letters.

brackets use brackets, not quotation marks surrounding data.

dash use one dash `"-"`, not two `"--"`, in page ranges.

fc add final comma to bibtex output.

By default latex encodings for accented characters are converted to letters. This may be a problem if the output encoding is not UTF-8, since some characters created by this process may be invalid in that encoding. For example, a BibTeX file which otherwise contains only cyriilic and latin characters may have a few entries with authors containing latin accented characters represented using the TeX convention. If those characters are not converted to Unicode letters, they can be exported to `"cp1251"` (Windows Cyrillic) for example. Specifying the option `no_latex` should solve the problem in such cases.

Argument `options` is mostly for debugging and mimics the command line options of the bibutils' binaries. The argument is a named character vector and is supplied as `c(tag1= val1, tag2 = val2, ...)`, where each tag is the name of an option and the value is the corresponding value. The value for options that do not require one is ignored and can be set to `""`. Some of the available options are:

h help, show all available options.

nb do not write Byte Order Mark in UTF8 output.

verbose print intermediate output.

debug print even more intermediate output.

Value

The function is used for the side effect of creating a file in the requested format. It returns a list, currently containing the following components:

`infile`	name of the input file,
`outfile`	name of the output file,
`nref_in`	number of references read from the input file,
`nref_out`	number of references written to the output file.
`bib`	when `outformat` is one of `"R"`, `"r"` or `"bibentry"`), a bibentry object obtained by reading `outfile`; otherwise not present.

Normally, `nref_in` and `nref_out` are the same. If some references were imported successfully but failed on export, `nref_out` may be smaller than `nref_in`. In such cases informative messages are printed during processing. (If this happens silently, it is probably a bug and please create an issue on Github.)

Supported formats

If an input or output format is not specified by arguments, it is inferred, if possible, from the file extension.

In the table below column Abbreviation shows the abbreviation for arguments `informat` and `outformat`, column FileExt gives the default file extension for that format, column Input (Output) contains TRUE if the format is supported for input (output) and FALSE otherwise. Column Description gives basic description of the format.

Abbreviation	FileExt	Input	Output	Description
ads	ads	FALSE	TRUE	ADS reference format
bib	bib	TRUE	TRUE	BibTeX
bibtex	bibtex	TRUE	TRUE	BibTeX
biblatex	biblatex	TRUE	TRUE	BibLaTeX
copac	copac	TRUE	FALSE	COPAC format references
ebi	ebi	TRUE	FALSE	EBI XML
end	end	TRUE	TRUE	EndNote (Refer format)
endx	endx	TRUE	FALSE	EndNote XML
isi	isi	TRUE	TRUE	ISI web of science
med	med	TRUE	FALSE	Pubmed XML references
nbib	nbib	TRUE	TRUE	Pubmed/National Library of Medicine nbib format
ris	ris	TRUE	TRUE	RIS format
R, r, Rstyle	R	TRUE	TRUE	R source file containing bibentry commands
rds	rds	TRUE	TRUE	bibentry object in a binary file created by saveRDS()
xml	xml	TRUE	TRUE	MODS XML intermediate
wordbib	wordbib	TRUE	TRUE	Word 2007 bibliography format

The file `"easyPubMedvig.xml"` used in the examples for Pubmed XML (`"med"`) was obtained using code from the vignette in package **easyPubMed** (Fantini 2019).

Author(s)

Georgi N. Boshnakov

References

Damiano Fantini (2019). "easyPubMed: Search and Retrieve Scientific Publication Records from PubMed." R package version 2.13, https://CRAN.R-project.org/package=easyPubMed.

Chris Putnam (2020). "Library bibutils, version 6.10." https://sourceforge.net/projects/bibutils/.

See Also

`readBib`,

`charToBib`,

`readBibentry`, `writeBibentry`

Examples

```
fn_biblatex <- system.file("bib", "ex0.biblatex",  package = "rbibutils")
fn_biblatex
## file.show(fn_biblatex)

## convert a biblatex file to xml
modl <- tempfile(fileext = ".xml")
bibConvert(infile = fn_biblatex, outfile = modl, informat = "biblatex", outformat = "xml")
## file.show(modl)

## convert a biblatex file to bibtex
bib <- tempfile(fileext = ".bib")
bib2 <- tempfile(fileext = ".bib")
bibConvert(infile = fn_biblatex, outfile = bib, informat = "biblatex", outformat = "bib")
## file.show(bib)

## convert a biblatex file to bibentry
rds <- tempfile(fileext = ".rds")
fn_biblatex
rds
be <- bibConvert(fn_biblatex, rds, "biblatex", "bibentry")
bea <- bibConvert(fn_biblatex, rds, "biblatex") # same
readRDS(rds)

## convert to R source file
r <- tempfile(fileext = ".R")
bibConvert(fn_biblatex, r, "biblatex")
## file.show(r)
cat(readLines(r), sep = "\n")

fn_cyr_utf8 <- system.file("bib", "cyr_utf8.bib",  package = "rbibutils")

## Can't have files with different encodings in the package, so below
## first convert a UTF-8 file to something else.
##
## input here contains cyrillic (UTF-8) output to Windows Cyrillic,
## notice the "no_latex" option
a <- bibConvert(fn_cyr_utf8, bib, encoding = c("utf8", "cp1251"), tex = "no_latex")

## now take the bib file and convert it to UTF-8
bibConvert(bib, bib2, encoding = c("cp1251", "utf8"))


## Latin-1 example: Author and Title fileds contain Latin-1 accented
##   characters, not real names. As above, the file is in UTF-8
fn_latin1_utf8  <- system.file("bib", "latin1accents_utf8.bib", package = "rbibutils")
## convert to Latin-1, by default the accents are converted to TeX combinations:
b <- bibConvert(fn_latin1_utf8, bib , encoding = c("utf8", "latin1"))
cat(readLines(bib), sep = "\n")
## use "no_latex" option to keep them Latin1:
c <- bibConvert(fn_latin1_utf8, bib , encoding = c("utf8", "latin1"), tex = "no_latex")
## this will show properly in Latin-1 locale (or suitable text editor):
```

```
##cat(readLines(bib), sep = "\n")

## gb18030 example (Chinese)
##
## prepare some filenames for the examples below:
xeCJK_utf8    <- system.file("bib/xeCJK_utf8.bib", package = "rbibutils")
xeCJK_gb18030 <- system.file("bib/xeCJK_gb18030.bib", package = "rbibutils")
fn_gb18030 <- tempfile(fileext = ".bib")
fn_rds <- tempfile(fileext = ".rds")

## input bib file utf8, output bib file gb18030:
bibConvert(xeCJK_utf8, fn_gb18030, encoding = c("utf8", "gb18030"))

## input bib file utf8, output file rds (and the rds object is returned
bibConvert(xeCJK_utf8, fn_rds)
unlink(fn_gb18030)
unlink(fn_rds)

## a Pubmed file
fn_med <- system.file("bib/easyPubMedvig.xml", package = "rbibutils")
## convert a Pubmed file to bibtex:
bibConvert(fn_med, bib, informat = "med")
## convert a Pubmed file to rds and import:
#bibConvert(fn_med, rds, informat = "med")


unlink(c(modl, bib, bib2, r, rds))
```

`bibentryExtra` *Work with 'bibentryExtra' objects*

Description

Objects from class "`bibentryExtra`" represent a collection of bibliographic references. This page documents functions to create such objects or convert other compatible objects to "`bibentryExtra`", as well as methods for subsetting and replacing parts of them.

Usage

```
bibentryExtra(bibtype = NULL, ...)

as.bibentryExtra(x, ...)

## S3 method for class 'bibentryExtra'
x[[i, j, drop = TRUE]]

## S3 replacement method for class 'bibentryExtra'
x[[i]] <- value
```

```
## S3 method for class 'bibentryExtra'
x[i, j, drop = TRUE]

## S3 replacement method for class 'bibentryExtra'
x$name <- value
```

Arguments

bibtype	a character string specifying the type of the bib entry. Can also be a character vector to create an object containing more than one entry, see bibentry.
x	for bibentryExtra, an object to be converted; otherwise an object from class "bibentryExtra".
i	the bib entry to extract or assign to, a character string (the key), a singe integer number (position), or a list of length 2. For the extractor, "[[", i can also be of length more than 1 (character or integer) when j is missing completely. See section 'Details' for complete details.
j	field(s) to extract, a character vector or missing, see section 'Details'.
name	field to extract.
drop	if TRUE, each entry in the returned list will contain the attributes (e.g., "bibtype" and "key").
value	value(s) to use for replacement, a list. For "[<-", an object inheriting from "bibentry" (so, including "bibentryExtra"), containing exactly one bib item. Alternatively, a character vector or a list, see section 'Details'.
...	for bibentryExtra, any arguments that bibentry accepts.

Details

bibentryExtra creates a "bibentryExtra" object. It has the same arguments as bibentry, see its help page for full details. The main difference is that for bibentryExtra, the bibentry type, is not restricted to have values from the list of standard Bibtex types (which is the case for bibtype).

bibentryExtra sets the "names" attribute to the keys of the bib entries in it. However, further changes in the names and/or the keys can make them different. If you want to keep the names always consistent with the keys, set names(bee) <- NULL. This will cause names to dynamically collect the keys when called.

as.bibentryExtra is a generic function for conversion of objects to class "bibentryExtra", most notably from class "bibentry".

Details on the subsetting methods are given below. The main thing that needs to be pointed out is that the bracket operators take as first argument the values of one or more keys, while for the dollar operators the argument is the name of a field (e.g., journal). This is convenient but can cause confusion, since usually $ and [[are (mostly) equivalent. This can be avoided by using argument j, since in the (i,j) pair i is always a key (or keys) and j the names of fields.

The subsetting methods aim to provide convenient access to components of "bibentryExtra" objects. In comparison to the corresponding methods for "bibenry" objects, the methods for "bibentryExtra" provide some additional features. Most notably, some of them include argument j, see the details below. If you want similar access for "bibenry" objects, just convert them to "bibentryExtra" using as.bibentryExtra.

It is convenient to think of the "bibentryExtra" as a ragged array with 'rows' (the bib entries) and 'columns' (the fields in the items). This is a peculiar ragged array, where each bib item (row) may have a different collection of fields.

Additional functionality for the subscript operators that admit argument j is provided for "bibentryExtra" objects. We will say that argument j *is missing completely* if it is missing and there is no placeholder for it in the call. For example be[i] and be[i, drop = FALSE] are examples when j is missing completely. On the other hand, be[i,] and be[i, , drop = FALSE] are examples when j is missing, but not completely.

When j is completely missing, the subscript operations for "bibentryExtra" objects work exactly as for bibentry objects (the latter don't have methods that use j). Alternatively, rather than use both i and j, one can set i to be a list of length 2, whose two components stand for i and j, respectively. The latter syntax can be used also for the methods that don't have j as an argument (but do have i).

The methods for the dollar operator:

"$" extracts the specified field. "bibentryExtra" inherits the "bibentry" method for "$". The result is a list with one component for each bib entry.

Note that the result is a list for several reasons. First, some fields may have more than one element. Second, some values are likely to be NULL. Third, some fields may be compound objects, e.g. 'Author', which is from class "person". However, if the returned list has one element, the enlosing list is removed (similarly to matrices with one row).

The "bibentryExtra" method for subset-assigment, "$<-", assigns a new value to the specified field. In most cases, value should be a list of the same length as x. Otherwise, it will be wrapped in a list If it is not a list, the result may be unexpected. For example, a character vector will replace the specified field in all bib entries.

To change the values of the keys, specify field "key". Again, remember that you get a list of all keys, unless there is only one bib entry in x. So, value should be a list of the same length.

The double bracket extractor method:

The method for "[[" accepts, unusually, two indices, i and j, reflecting the above interpretation.

i is typically a character string or a positive integer number identifying the bib item to extract, while j is a character vector specifying the required fields of that bib item. The result is a list. If drop is FALSE, then the attributes of this list are set to those of the bib item (e.g., bibtype and key).

If j is missing, all fields of the item are included in the result. However it depends on how j is missing. If it is missing as in x[[i,]] or x[[i, , drop = FALSE]], the result is as above and includes all entries in bib item i.

Alternatively, i can be a list of length 2 and j omitted. This is equivalent to a call with i = i[[1]] and j = i[[2]].

If j is missing completely (i.e., there is no redundant comma in the call), as in x[[i]] or x[[i, drop = FALSE]], then i must be a single index value (positive integer number or character string). The result is a "bibentryExtra" object. This case is compatible with the method for "bibentry" objects, which does not have argument j. Note though that for the "bibentry" method i can be of length more than 1.

The double bracket assignment method:

The assignment version of "[[" does not have argument j, so the two-element list form for i is used when fields are needed, see above.

If value inherits from "bibentry" (in particular, it can be a "bibentryExtra" object), then i must be a single character string or a positive integer specifying the bibitem to replace with value. Notice that the new item may have a different key.

Otherwise, value should be a named list and i a list of length 2. In this case, i[[1]] should be a character string of a positive integer identifyng the bib item on which replacement will take place, while i[[2]] is a character vector specifying the fields to replace. As a special case, i[[2]] can be the character string "*", which specifies that all elements of value should be used.

Usually value has names and these are interpreted as names of fields. In this case, the fields specified by i[[2]] are replaced by the corresponding fields in value.

If value has no names and i[[2]] is not equal to "*" (see above), value must have the same length as i[[2]] and its names are set to i[[2]].

The single bracket extractor method:

For "[", i is an index vector specifying which bib entries to extract. If j is missing completely, the bib entries are extracted and returned as a "bibentryExtra" object.

If j is used, it is a character vector specifying which fields to keep. For example, this could be a list of all standard bibtex fields. Only fields from this list are kept. Note that this may leave some bib entries invalid (i.e., missing compulsory fields).

Note that in any case the returned object has class "bibentryExtra".

Value

for as.bibentryExtra and bibentryExtra, a "bibentryExtra" object;

for "$", a list containing the requested fields with one list component for each key. If only one key was specified, the outer list is dropped;

for "[[", typically a list, but if argument x is missing completely, the result is a "bibentryExtra" object, as described in section 'The double bracket extractor method';

for "[", a "bibentryExtra" object;

for the assignment operators ("[[<-", "[<-" and "$<-"), the modified object x.

for names, always a character vector. If attribute names is NULL, the keys are put in a character vector and returned.

Note

This is somewhat experimental but incompatible changes are unlikely.

Author(s)

Georgi N. Boshnakov

See Also

the vignette,

readBib for importing bibtex files,

readBibentry for importing from R source files containing bibentry expressions,

charToBib for converting character vectors containing bibentry expressions.

Examples

```
## example bib from ?bibentry
bref <- c(
    bibentry(
        bibtype = "Manual",
        title = "boot: Bootstrap R (S-PLUS) Functions",
        author = c(
            person("Angelo", "Canty", role = "aut",
                   comment = "S original"),
            person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
                   comment = "R port, author of parallel support",
                   email = "ripley@stats.ox.ac.uk")
        ),
        year = "2012",
        note = "R package version 1.3-4",
        url = "https://CRAN.R-project.org/package=boot",
        key = "boot-package"
    ),

    bibentry(
        bibtype = "Book",
        title = "Bootstrap Methods and Their Applications",
        author = as.person("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"
    )
  )

brefExtra <- as.bibentryExtra(bref)

## error: j is present, so i must have length 1:
## brefExtra[[1:2, "title"]]

## the returned list doesn't have attributes:
                            brefExtra[[1, c("title", "author")]] # drop = TRUE by default

## now it does:
brefExtra[[1, "title", drop = FALSE]]
brefExtra[["boot-package", "title", drop = FALSE]]

brefExtra[["boot-book", ]]
brefExtra[["boot-book"]]
```

```
## assignment "[[<-"
b2 <- brefExtra
b2

## use all elements of 'value'
b2[[list(1, "*")]] <- list(title = "New title", note = "a new note")

## replace title
b2[[list(1, "title")]] <- list(title = "New title A")
b2

## no change,'year' is not in 'value'
b2[[list(1, "year")]] <- list(title = "New title A2")
b2

## remove 'year'
b2[[list(1, "year")]] <- list(title = "New title A", year = NULL) ## removes 'year'
b2

## a bibentry 'value'
b2[[2]] <- bibentry(bibtype = "Misc", title = "Dummy title",
                    author = "A A Dummy", organization = "none")
b2
```

`rbibutils_formats` *Supported bibliography formats*

Description

Supported bibliography formats in package rbibutils.

Usage

```
rbibutils_formats
```

Format

A data frame with 16 observations on the following 5 variables:

`Abbreviation` a character vector.

`FileExt` a character vector.

`Input` a logical vector.

`Output` a logical vector.

`Description` a character vector.

Details

Each row in rbibutils_formats gives information about a supported bibliography format in package **rbibutils**.

Abbreviation is the name to use in arguments informat and outformat in bibConvert.

FileExt is the default extension used to infer the input/output format if the corresponding argument is missing.

Input (Output) is TRUE if the format is supported for input (output).

Description gives a brief description of the format.

See bibConvert, section "Supported formats", for further details.

Examples

```
rbibutils_formats
```

---

readBib *Read and write bibtex files*

Description

Read and write bibtex files.

Usage

```
readBib(file, encoding = NULL, ..., direct = FALSE,
        texChars = c("keep", "convert", "export", "Rdpack"),
        macros = NULL, extra = FALSE, key, fbibentry = NULL)

writeBib(object, con = stdout(), append = FALSE)

charToBib(text, informat, ...)
```

Arguments

file	name or path to the file, a character string.
encoding	the encoding of file, a character string.
direct	If TRUE parse file directly to bibentry, otherwise convert first to intermediate XML, then to bibentry.
texChars	What to do with characters represented by TeX commands (for example, accented Latin charaters? If "export", export as TeX escapes when possible. If "convert", convert to the target encoding. If "keep", output the characters as they were in the input file, like "export", but don't convert normal characters to TeX escapes. "Rdpack" is mainly for internal use and its actions may be changed. It is equivalent to "keep" plus some additional processing, see https://github.com/GeoBosh/rbibutils/issues/7#issue-1020385889.

macros	additional bib files, usually containing bibtex macros, such as journal abbreviations.
object	a bibentry object.
con	filename (a character string) or a text connection
append	if TRUE append to the file.
text	a character vector.
informat	the input format, defaults to "bibtex".
key	a character vectors of key(s) to use for entries without cite keys. Should have the same number of elements as the number of such entries.
...	for charTobib, arguments to be passed on to readBib or bibConvert, see section "Details". Not used by readBib and writeBib (which throw error to avoid silently ignoring unknown arguments).
extra	if TRUE, allow non-standard bibtex types.
fbibentry	a function to use for generating bib objects instead of bibentry(), see section "Details".

Details

readBib is wrapper around bibConvert for import of bibtex files into bibentry objects.

If direct = FALSE, the bibtex file is converted first to XML intermediate, then the XML file is converted to bibentry. The advantage of this is that it gives a standardised representation of the bibtex input. Fields that cannot be mapped to the intermediate format are generally omitted.

If direct = TRUE the input file is converted directly to bibentry, without the XML intermediate step. This means that non-standard fields in the bib entries are preserved in the bibentry object.

Argument texChars, currently implemented only for the case direct = TRUE, gives some control over the processing of TeX sequences representing characters (such as accented Latin characters): If it is "keep" (the default), such sequences are kept as in the input. "convert" causes them to be converted to the characters they represent. Finally, "export" exports characters as TeX sequences, whenever possible.

The difference between "keep" and "export" is that "keep" does not convert normal characters to TeX escapes, while "export" does it if possible. For example, if the input file contains the TeX sequence \"o representing the letter o-umlaut, "keep" and "export" will keep it as TeX sequence, while "convert" will convert it to the character o-umlaut in the output encoding (normally UTF-8). On the othe hand, if the input file contains the character o-umlaut, then "keep" and "convert" will convert it to the output encoding of o-umlaut, while "export" will export it as \"o.

Currently, texChars = "export" does not process properly mathematical formulas.

fbibentry, if supplied, will be used in place of utils::bibentry to create bib objects in R. The arguments of fbibentry should be the same as for utils::bibentry.

writeBib writes a bibentry object to a bibtex file.

charTobib is a convenience function for reading or converting bibliography information, accepting the input from a character vector rather than a file. If informat is missing it calls readBib, otherwise bibConvert. In both cases the remaining arguments are passed on and should be suitable for the called function.

The files specified by argument `macros` are read in before those in `file`. Currently this is implemented by concatenating the files in the order they appear in `c(macros, file)`. It is ok for `macros` to be `character(0)`.

Value

for `readBib`, a `bibentry` object. If `extra` is `TRUE` it can also be `bibentryExtra` (which inherits from `bibentry`). If `fbibentry` is a function the return value is whatever it returns.

for `writeBib`, the `bibentry` object (invisibly)

Author(s)

Georgi N. Boshnakov

See Also

`readBibentry` and `writeBibentry` for import/export to R code,

`bibConvert`

Examples

```
## create a bibentry object
bibs <- readBib(system.file("REFERENCES.bib", package = "rbibutils"),
                encoding = "UTF-8")
## write bibs to a file
fn <- tempfile(fileext = ".bib")
writeBib(bibs, fn)

## see the contents of the file
readLines(fn) # or: file.show(fn)

## import a bib file containing Chineese characters encoded with UTF-8:
ch_bib <- readBib(system.file("bib/xeCJK_utf8.bib", package = "rbibutils"))
ch_bib
print(ch_bib, style = "R")

## import a bib file encoded with the official Chineese encoding:
ch_bib2 <- readBib(system.file("bib/xeCJK_gb18030.bib", package = "rbibutils"),
                   encoding = "gb18030")


## a dummy reference with accented characters
## (in the file some are utf8, others are TeX escapes)
bibacc <- system.file("bib/latin1accents_utf8.bib", package = "rbibutils")

## export as UTF-8 characters
## this will print as true characters in suitable locale:
be <- readBib(bibacc, direct = TRUE, texChars = "convert")
print(be, style = "R")
print(be, style = "bibtex")
## compare to the input file:
```

```
readLines(bibacc)


be1 <- readBib(bibacc, direct = TRUE)
be1a <- readBib(bibacc, direct = TRUE, texChars = "keep") # same
be1
print(be1, style = "R")
print(be1, style = "bibtex")

## export as TeX escapes, when possible
be2 <- readBib(bibacc, direct = TRUE, texChars = "export") ## same
be2
print(be2, style = "R")
print(be2, style = "bibtex")

## in older versions (up to 2.2.4) of rbibutils, "convert" converted
##     a lot of TeX commands representing symbols  to characters.
## This is no longer the case:
be3 <- readBib(bibacc, direct = TRUE, texChars = "convert")
## be3
print(be3, style = "R")
## print(be3, style = "bibtex")

## charToBib
##
## get a bibtex reference for R
Rcore <- format(citation(), style = "bibtex")
## add a citation key
Rcore <- sub("@Manual{", "@Manual{Rcore", Rcore, fixed = TRUE)
cat(Rcore, sep = "\n")
beRcore <- charToBib(Rcore)
beRcore
class(beRcore)
print(beRcore, style = "R")

## bibtex entries generated by citation() don't have cite keys.
## this sets the key to 'Rcore'
beRcore <- charToBib(toBibtex(citation()), key = "Rcore")
beRcore$key == "Rcore"  # TRUE

## this sets two keys
bemore <- charToBib(toBibtex( c(citation(), citation("rbibutils"))),
    key = c("Rcore", "Rpackage:rbibutils"))
all.equal(names(bemore), c("Rcore", "Rpackage:rbibutils"))


## a large example with several files - needs internet access;
## it is better to clone repository https://github.com/iridia-ulb/references
## and work on local files
##
## iridia_mac <- c("abbrev.bib", "authors.bib", "journals.bib", "crossref.bib")
## iridia_biblio <- "biblio.bib"
##
```

```
## iridia_raw_url <- "https://raw.githubusercontent.com/iridia-ulb/references/master"
## iridia_mac_url <- file.path(iridia_raw_url, iridia_mac)
## iridia_biblio_url <- file.path(iridia_raw_url, iridia_biblio)
##
## bibdir <- tempdir()
## iridia_mac_loc <- file.path(bibdir, iridia_mac)
## iridia_biblio_loc <- file.path(bibdir, iridia_biblio)
##
## ## download the files to bibdir
## sapply(c(iridia_biblio_url, iridia_mac_url),
##        function(x) download.file(x, file.path(bibdir, basename(x))))
##
## iridia <- readBib(iridia_biblio_loc, direct = TRUE, macros = iridia_mac_loc)
## iridia[1]
## print(iridia[1], style = "R")
## toBibtex(iridia[1])  # or: print(iridia[1], style = "bibtex")
## length(iridia)  # 2576 at the time of writing

unlink(fn)
```

`readBibentry` *Read and write bibentry files or read bibtex strings*

Description

Read and write bibentry files.

Usage

```
readBibentry(file, extra = FALSE, fbibentry = NULL)

writeBibentry(be, file = stdout(), style = c("Rstyle", "loose"))
```

Arguments

`be`	a bibentry object.
`file`	filename, a character string or a connection. For `readBibentry`, input from the console can be specified by `file = ""`. The default for `writeBibentry` is `stdout` (effectively, to write on the screen).
`extra`	if `TRUE` allow non-standard bibtex types.
`style`	if `"Rstyle"` (default), wrap in `c()`, otherwise don't wrap and don't put commas between the entries, see section "Details".
`fbibentry`	a function to use for generating bib objects. The default is `utils::bibentry()`.

Details

These functions read/write `bibentry` objects from/to R source files. Two styles are supported. `"Rstyle"` is the format used by `print(be, style = "R")`, which writes the `bibentry` calls as a comma separated sequence wrapped in `c()` (i.e., the file contains a single R expression). Style `"loose"` writes the entries without separators and no wrapping.

`writeBibentry` writes the object to the specified file in the requested style (default is `"Rstyle"`). The file is neatly formatted for humans to read and edit.

`readBibentry` reads the file and creates a `bibentry` object. It doesn't have argument for style, since that is inferred from the contents of the file.

`bibentry()` calls that throw errors are not included in the returned object. The errors are intercepted and converted to warnings, identifying the corresponding `bibentry()` calls by their keys, if present (otherwise the text of the whole bibentry is shown).

Value

for `writeBibentry`, `NULL` (invisibly)

for `readBibentry`, a `bibentry` object with the keys as names

Author(s)

Georgi N. Boshnakov

See Also

`readBib` and `writeBib` for reading/writing bib files,

`bibConvert`

`charToBib` for reading from a character vector

Examples

```
bibs <- readBib(system.file("REFERENCES.bib", package = "rbibutils"),
                encoding = "UTF-8")
fn <- tempfile(fileext = ".bib")

writeBibentry(bibs, file = fn) # style = "Rstyle" (default)
cat(readLines(fn), sep = "\n")

writeBibentry(bibs, file = fn, style = "loose")
cat(readLines(fn), sep = "\n")

unlink(fn)
```

`register_JSSextra` *Create and register bibstyle JSSextra*

Description

Create and register bibstyle JSSextra.

Usage

```
register_JSSextra(make_default = FALSE, reset = FALSE, parent_style = "JSS")
```

Arguments

`make_default`	if `TRUE` make `"JSSextra"` default.
`reset`	recreate bibstyle `"JSSextra"`.
`parent_style`	the style from which to derive `"JSSextra"`.

Details

`register_JSSextra` creates style `"JSSextra"` and registers it for use in the current R session. This means that it can be specified for functions which accept a bibstyle argument, most notably printing objects from class `"bibentry"` and `"bibentryExtra"`. In normal use `register_JSSextra` is called once in a session.

Functions accepting a bibstyle argument use a default style if such an argument is not provided. In most cases it is `"JSS"`.

Using `tools::bibstyle()` the default style can be changed at any time to any of the styles currently registered in the session. A list of these styles can be obtained with `tools::getBibstyle(TRUE)`. The currently default style can be seen with `tools::getBibstyle()`. As a convenience `register_JSSextra(TRUE)` makes and registers `"JSSextra"` as the default style.

The remaining arguments should rarely be needed in normal circumstances.

`register_JSSextra` stores the bibstyle object it creates and just uses it when called again. `reset = TRUE` can be used to force a fresh copy of `"JSSextra"` to be created.

By default `"JSSextra"` is derived from `"JSS"`. To base it on a different style, use argument `parent_style`.

Value

`register_JSSextra` is used mainly for the side effect of registering and setting the style as default. It returns the created style (an environment) but it can be discarded.

Author(s)

Georgi N. Boshnakov

Examples

```
## current default style
tools::getBibstyle()
tools::getBibstyle(TRUE)  # all styles, currently "JSS" only

register_JSSextra()       # register "JSSextra"
tools::getBibstyle(TRUE)  # now it is available
tools::getBibstyle()      # ... but not default

register_JSSextra(TRUE)  # this makes it default
tools::getBibstyle()

## setting default style with bibstyle():
tools::bibstyle("JSS", .default = TRUE)
tools::getBibstyle()
```

Index