Package 'libbib'

November 5, 2022

Type Package

Title Various Utilities for Library Science/Assessment and Cataloging

Version 1.6.4

Maintainer Tony Fischetti <tony.fischetti@gmail.com>

Description Provides functions for validating and normalizing bibliographic codes such as ISBN, ISSN, and LCCN. Also includes functions to communicate with the WorldCat API, translate Call numbers (Library of Congress and Dewey Decimal) to their subject classifications or subclassifications, and provides various loadable data files such call number / subject crosswalks and code tables.

License GPL-3

Depends R (>= 3.5.0), data.table, utils

Imports curl, methods, pbapply, stringr, xml2

Suggests assertr, testthat, knitr, magrittr, rmarkdown

Encoding UTF-8

RoxygenNote 7.2.1

VignetteBuilder knitr

NeedsCompilation no

Author Tony Fischetti [aut, cre]

Repository CRAN

Date/Publication 2022-11-05 22:50:02 UTC

R topics documented:

`books_serials_etc_sample` *Small sample of books, monographs, and serials and their information*

Description

A very small sample of books, serials, VHSs, CDs, and Computer files and some information including title, control numbers, call numbers, and call number subject classifications. Somewhat messy/inconsistent (deliberately) and mainly for testing. Will be expanded in future versions.

Usage

```
data(books_serials_etc_sample)
```

Format

An object of class `"data.table"`;

`car` *Return first element of vector*

Description

Takes a vector and returns the first element Equivalent to Lisp's `car` function

Usage

```
car(x)
```

Arguments

`x`	A vector

Details

Originally for use as a reduction function in `split_map_filter_reduce`

Value

Returns first element of vector

See Also

split_map_filter_reduce

Examples

```
car(c(8, 6, 7, 5, 3, 0, 9))      # 8
mt <- as.data.table(mtcars)
dt_del_cols(mt, "cyl", "disp", "hp")
```

`check_isbn_10_check_digit`

Check the check digit of an ISBN 10

Description

Takes a string representation of an ISBN 10 and verifies that check digit checks out

Usage

```
check_isbn_10_check_digit(x, allow.hyphens = TRUE, errors.as.false = TRUE)
```

Arguments

`x`	A string of 10 digits or 9 digits with terminal "X"
`allow.hyphens`	A logical indicating whether the hyphen separator should be allowed (default is `FALSE`)
`errors.as.false`	return false if error instead of throwing error (default is `TRUE`)

Value

Returns TRUE if check passes, FALSE if not, and NA if NA

Examples

```
check_isbn_10_check_digit("012491540X")      # TRUE
check_isbn_10_check_digit("0-124-91540-X")   # TRUE

# vectorized
check_isbn_10_check_digit(c("012491540X", "9004037812"))  # TRUE FALSE
```

check_isbn_13_check_digit

Check the check digit of an ISBN 13

Description

Takes a string representation of an ISBN 13 and verifies that check digit checks out

Usage

```
check_isbn_13_check_digit(x, allow.hyphens = TRUE, errors.as.false = TRUE)
```

Arguments

x	A string of 13 digits
allow.hyphens	A logical indicating whether the hyphen separator should be allowed (default is TRUE)
errors.as.false	return false if error instead of throwing error (default is TRUE)

Value

Returns TRUE if check passes, FALSE if not, and NA if NA

Examples

```
check_isbn_13_check_digit("9780306406157")          # TRUE
check_isbn_13_check_digit("978-0-306-40615-7")      # TRUE

# vectorized
check_isbn_13_check_digit(c("978-0-306-40615-7", "9783161484103"))  # TRUE FALSE
```

check_issn_check_digit

Check the check digit of an ISSN

Description

Takes a string representation of an ISSN and verifies that check digit checks out

Usage

```
check_issn_check_digit(x, allow.hyphens = TRUE, errors.as.false = FALSE)
```

Arguments

x	A string of 8 digits or 7 digits with terminal "X"
allow.hyphens	A logical indicating whether the hyphen separator should be allowed (default is TRUE)
errors.as.false	return false if error instead of throwing error (default is FALSE)

Value

Returns TRUE if check passes, FALSE if not, and NA if NA

Examples

```
check_issn_check_digit("2434561X")   # TRUE
check_issn_check_digit("2434-561X")  # TRUE

# vectorized
check_issn_check_digit(c("03785955", "2434561X", NA))  # TRUE TRUE NA
check_issn_check_digit(c("0378-5955", "2434-561X", NA))
# TRUE TRUE NA
```

---

convert_to_isbn_13 *Convert ISBN 10 to ISBN 13*

Description

Takes a string representation of an ISBN 10 and converts it to an ISBN 13.

Usage

```
convert_to_isbn_13(x, skip.validity.check = FALSE, errors.as.nas = FALSE)
```

Arguments

x	A string of 10 digits or 9 digits with terminal "X"
skip.validity.check	Skip the checking for whether the ISBN 10 is valid (default is FALSE)
errors.as.nas	return NA if error instead of throwing error (default is FALSE)

Value

Returns ISBN 13 as a string

Examples

```
convert_to_isbn_13("012491540X")                # 9780124915404

# vectorized
convert_to_isbn_13(c("012491540X", "9004037810"))
# "9780124915404" "9789004037816"
```

`country_code_crosswalk` *Country code / country crosswalk*

Description

A cross-walk between the country code and it's human readable version

Usage

```
data(country_code_crosswalk)
```

Format

An object of class `"data.table"`;

Source

https://www.loc.gov/marc/countries/countries_code.html

`cp_lb_attributes` *Copy special libbib attributes from one object to another*

Description

Takes two objects and copies all special libbib attributes (attributes beginning with `lb.`) from the first object to the second, by reference.

Usage

```
cp_lb_attributes(a, b)
```

Arguments

`a`	The first object (the one with the attributes to copy)
`b`	The second object (the one to copy those attributes to)

Value

Nothing, since the object is modified by reference.

Examples

```
tmp1 <- "a"
set_lb_date(tmp1, "2021-05-08")
set_lb_attribute(tmp1, "note", "just an example")

tmp2 <- "b"
cp_lb_attributes(tmp1, tmp2)
attributes(tmp2)$lb.date
# [1] "2021-05-08"
attributes(tmp2)$lb.note
# [1] "just an example"
```

---

dewey_subject_crosswalk

Dewey Decimal Classification / Subject Description crosswalk

Description

A cross-walk between the Dewey Decimal Classification code and it's human readable subject description

Usage

```
data(dewey_subject_crosswalk)
```

Format

An object of class `"data.table"`;

Source

Edited from https://www.oclc.org/content/dam/oclc/dewey/ddc23-summaries.pdf

dt_add_to_col_names *Add string to all column names in a data.table*

Description

Takes a data.table and a string. The supplied string will be added to end of the each column's name. If `prefix` is `TRUE`, the string is added to the beginning, instead.

Usage

```
dt_add_to_col_names(
  DT,
  astring,
  prefix = FALSE,
  exclude = NULL,
  include = NULL,
  fix.duplicates = FALSE
)
```

Arguments

`DT`	A data.table
`astring`	A string to add to each column name
`prefix`	A logical indicating whether the string should be added to the beginning of each column name, instead of the end. (default is `FALSE`)
`exclude`	A quoted vector or column names to exclude from renaming. Cannot co-exist with `include`
`include`	A quoted vector or column names. Changes names of only these columns. Cannot co-exist with `exclude`
`fix.duplicates`	A logical indicating whether to, if after the suffix/prefixes are added to the selected column names, correct those duplicate column names by making them unique. If `FALSE` (default), if any of the column names are duplicated, an error is raised and the new names are not set. If `TRUE`, all the column names are made unique, potentially renaming excluded column names that were not supposed to be changed.

Value

Returns data.table with string appended or prefixed to all selected column names.

Examples

```
DT <- as.data.table(iris)

dt_add_to_col_names(DT, "_post")
names(DT)
```

```
# [1] "Sepal.Length_post" "Sepal.Width_post"  "Petal.Length_post"
# [4] "Petal.Width_post"  "Species_post"

 DT <- as.data.table(iris)
dt_add_to_col_names(DT, "pre_", prefix=TRUE)
names(DT)
# [1] "pre_Sepal.Length" "pre_Sepal.Width"  "pre_Petal.Length" "pre_Petal.Width"
# [5] "pre_Species"

DT <- as.data.table(iris)
dt_add_to_col_names(DT, "_post", exclude="Species")
names(DT)
# [1] "Sepal.Length_post" "Sepal.Width_post"  "Petal.Length_post"
# [4] "Petal.Width_post"  "Species"
```

`dt_counts_and_percents` *Group by, count, and percent count in a data.table*

Description

This function takes a (quoted) column to group by, counts the number of occurrences, sorts descending, and adds the percent of occurrences for each level of the grouped-by column.

Usage

```
dt_counts_and_percents(DT, group_by_this, percent.cutoff = 0, big.mark = FALSE)
```

Arguments

`DT`	The data.table object to operate on
`group_by_this`	A quoted column to group by
`percent.cutoff`	A percent (out of 100) such that all the count percents lower than this number will be grouped into "OTHER" in the returned data.table (default is 0)
`big.mark`	If `FALSE` (default) the "count" column is left as an integer. If not `FALSE`, it must be a character to separate every three digits of the count. This turns the count column into a string.

Details

For long-tailed count distributions, a cutoff on the percent can be placed; percent of counts lower than this percent will be grouped into a category called "OTHER". The percent is a number out of 100

The final row is a total count.

The quoted group-by variable must be a character or factor. If it is not, it will be temporarily converted into one and a warning is issued.

Value

Returns a data.table with three columns: the grouped-by column, a count column, and a percent column (out of 100) to two decimal places

Examples

```
iris_dt <- as.data.table(iris)
dt_counts_and_percents(iris_dt, "Species")
mt <- as.data.table(mtcars)
mt[, cyl:=factor(cyl)]
dt_counts_and_percents(mt, "cyl")
dt_counts_and_percents(mt, "cyl", percent.cutoff=25)
```

---

dt_del_cols *Delete columns in a data.table*

Description

Takes a data.table and a quoted sequence of column names and removes the specified column names from the data.table

Usage

```
dt_del_cols(DT, ...)
```

Arguments

DT	A data.table
...	arbitrary number of column names in quotes

Value

Returns data.table with those columns removed

Examples

```
mt <- as.data.table(mtcars)
dt_del_cols(mt, "cyl", "disp", "hp")
```

dt_keep_cols *Keep columns in a data.table*

Description

Takes a data.table and a quoted sequence of column names and removes all columns but the ones specified

Usage

```
dt_keep_cols(DT, ...)
```

Arguments

DT	A data.table
...	arbitrary number of column names in quotes

Value

Returns data.table with only those columns

Examples

```
mt <- as.data.table(mtcars)
dt_keep_cols(mt, "mpg", "am", "gear", "carb")
```

dt_na_breakdown *Get a breakdown of the NA-status of a column in a data.table*

Description

This function takes a (quoted) column to group by, and tabulates the count of how many of those values are not-NA and NA, and adds the percent of occurrences. A TRUE in the first output column means the data is _not_ missing; FALSE corresponds to missing.

Usage

```
dt_na_breakdown(DT, acolumn, big.mark = FALSE)
```

Arguments

DT	The data.table object to operate on
acolumn	a quoted column name
big.mark	If FALSE (default) the "count" column is left as an integer. If not FALSE, it must be a character to separate every three digits of the count. This turns the count column into a string.

Details

The final row is a total count

The quoted group-by variable must be a character or factor

Value

Returns a data.table with three columns: the not-NA status of the column specified, a count column, and a percent column (out of 100) to two decimal places

Examples

```
iris_dt <- as.data.table(iris)
iris_dt[sample(1:.N, 10), Species:=NA_character_]
dt_na_breakdown(iris_dt, "Species")
```

---

dt_percent_not_na *Return the percentage of non-NA instances in a data.table column*

Description

This function takes a data.table and a quoted column name and returns the percentage of the data in the column that is not NA. The percent is out of 100 and contains up to two decimal places

Usage

```
dt_percent_not_na(DT, acolumn)
```

Arguments

DT	A data.table object
acolumn	a quoted column name

Value

Returns percentage of non-NA instances in column

See Also

is.na

Examples

```
mt <- as.data.table(mtcars)
mt[mpg<16, mpg:=NA]
dt_percent_not_na(mt, "mpg")        # 68.75
```

`dt_set_clean_names` *Takes a data.table and set to cleaned column names*

Description

This function takes a data.table, and returns the same data.table with column names that are cleaned and stripped of potentially troublesome names

Usage

```
dt_set_clean_names(DT, lower = TRUE)
```

Arguments

`DT`	a data.table
`lower`	A logical indicating whether all column names should be made lower case (default `TRUE`).

Details

All space/whitespace characters are replaced with underscores, as are all characters not from A-Z, a-z, an underscore, or a digit

Value

Returns the data.table but with cleaned names

See Also

`get_clean_names`

Examples

```
ejemplo <- as.data.table(iris)
setnames(ejemplo, c("Sepal Length", "Sepal@Width", "Petal Length",
                    "Petal\\nWidth", "Spêcies"))
dt_set_clean_names(ejemplo)
```

`fread_plus_date` *Read a file and set a special libbib date attribute*

Description

Takes a file name, reads it with `data.table::fread`, and sets an attribute called `lb.date` with a date extracted from the file name.

Usage

```
fread_plus_date(fname, allow.fallback.date = TRUE, ...)
```

Arguments

`fname`	The file name to read
`allow.fallback.date`	A logical indicating whether, if no matching file name with a date is found, to use today's date as the date attribute. Default is `TRUE`.
`...`	Arbitrary arguments to use with `fread`

Details

The file name can be one with a valid ISO 8601 date (yyyy-mm-dd) already in it, or it can be a file name with the date elided.

For example, if there is a file you'd like to read on your disk called "iris-2021-05-08.csv", you can call this function with either "iris.csv" or "iris-2021-05-08.csv" as the file name.

When you call this function with a file name without an ISO 8601 date (e.g. "iris.csv.gz"), the file name extension ".csv.gz" is removed and the function looks for a file name beginning with "iris", a date, and the file extension. The file extension is considered to be anything after the first period in the base name. For example, if the file name given is "./my.data/iris.csv.gz", the extension is ".csv.gz". This means no period can be present in the base file name (after any directories) with the exception of the file extension.

If you call this function with "iris.csv" and there is no file name with an ISO 8601 date appended to that file name on your disk, and `allow.fallback.date` is `TRUE`, then the `lb.date` attribute is set to the current date.

Value

A `data.table` with an attribute called `lb.date` set

Examples

```
## Not run:
  # there's a file called "iris-2021-05-08.csv" on disk
  dat <- fread_plus_date("iris.csv")
  attribute(dat)$lb.date
  # [1] "2021-05-08
```

```
  # can also read the full file name
  dat <- fread_plus_date("iris-2021-05-08.csv")
  attribute(dat)$lb.date
  # [1] "2021-05-08

## End(Not run)
```

`fwrite_plus_date` *Write a file with a date appended to the file name.*

Description

Takes a `data.table`, a file name, and writes it with `data.table::fwrite`.

Usage

```
fwrite_plus_date(
  DT,
  fname,
  from.attribute = TRUE,
  allow.fallback.date = TRUE,
  ...
)
```

Arguments

`DT`	a `data.table` to write to disk
`fname`	The file name to write the `data.table` to. The date will be appended between the file name and its file extension
`from.attribute`	A logical indicating whether the date should be taken from the `lb.date` attribute of the `data.table`, or whether it should be today's date. Default (`TRUE`) takes it from the `lb.date` attribute.
`allow.fallback.date`	A logical indicating, if there is no `lb.date` attribute in the supplied `data.table`, whether it is permissible to use today's date, instead. Default is `TRUE`.
`...`	Arbitrary arguments to pass to `fwrite`

Details

The supplied file name will be modified to include an ISO 8601 date (yyyy-mm-dd) between the file name and the file extension. Under the default settings, the date used will be from the `lb.date` attribute of the supplied `data.table`. If there is no `lb.date` attribute, the current date will be used, instead.

For example, if there is a `data.table` with an `lb.date` attribute of "2021-05-08", and you supply this function with the file name "iris.csv", the file name actually written to disk will be "iris-2021-05-08.csv". Under the default settings, if there is no `lb.date` attribute, but today's date is "2038-01-19", the file name written to disk will be "iris-2038-01-19.csv".

The ISO 8601 date is sandwiched between the file name and the file extension. The file extension is considered to be anything after the first period in the base name. For example, if the file name given is "./my.data/iris.csv.gz", the extension is ".csv.gz". This means no period can be present in the base file name (after any directories) with the exception of the file extension.

Examples

```
## Not run:

set_lb_date(iris, "2021-05-08")
fwrite_plus_date(iris, "iris.csv.gz")
# "iris-2021-05-08.csv.gz" is now written to disk


## End(Not run)
```

`get_all_lc_call_subject_letters` *Get all subject letters of LC Call Number*

Description

Takes a string representation of a Library of Congress call number and returns all the subject letters if and only if the LC Call Number is valid

Usage

```
get_all_lc_call_subject_letters(x, allow.bare = FALSE)
```

Arguments

`x`	A Library of Congress call number (string)
`allow.bare`	A logical indicating whether an LC Call with only the letters should be considered valid (default is `FALSE`)

Value

Returns all the subject letters or NA if invalid

Examples

```
get_all_lc_call_subject_letters("Q172.5.E77")
# Q
get_all_lc_call_subject_letters("AF172.5.E77")
# NA

# vectorized
get_all_lc_call_subject_letters(c("Q 172.5", "AF172", "PR6023.A93"))
# Q NA PR
```

`get_clean_names` *Takes a data.frame and returns cleaned column names*

Description

This function takes a data.frame, extracts the column names, and returns a vector of those column names but cleaned and stripped of potentially troublesome names

Usage

```
get_clean_names(dat, lower = TRUE)
```

Arguments

`dat`	A data.frame
`lower`	A logical indicating whether all column names should be made lower case (default `TRUE`).

Details

All space/whitespace characters are replaced with underscores, as are all characters not from A-Z, a-z, an underscore, or a digit

If there are duplicate column names after the cleaning, a message will show stating such and the duplicate column names will be make unique.

Value

Returns a vector of cleaned names

See Also

`make.unique`

Examples

```
ejemplo <- iris
names(ejemplo) <- c("Sepal Length", "Sepal@Width", "Petal Length",
                    "Petal\\nWidth", "Spêcies")
# c("sepal_length" "sepal_width"  "petallength"  "petal_nwidth" "sp_cies")

# c("Sepal_Length" "Sepal_Width"  "PetalLength"  "Petal_nWidth" "Sp_cies")
```

---

get_country_from_code *Conversion from country code to country name*

Description

Takes a country code (defined in the Marc standards) and returns the country name.

Usage

```
get_country_from_code(x)
```

Arguments

x	A country code (defined in the Marc standards) or a vector of country codes

Details

Interestingly, although it's called 'country' in the Marc standard, cities, states, and other non-countries also have codes

Value

Returns the country (place) name. NA if cannot be matched to country in standard.

Examples

```
get_country_from_code("ck")
# Colombia

# tolerant of case and leading/trailing whitespace
get_country_from_code(c(" PE", "not-a-country", "nyu"))
# c("Peru", NA, "New York (State)")
```

get_dewey_decimal_subject_class

Conversion from Dewey Decimal call numbers to first-level subject description

Description

Takes a string representation of a Dewey Decimal call number (DDC) and returns it's subject description. This uses the hundreds place of the DDC number and returns the most general subject classification.

Usage

```
get_dewey_decimal_subject_class(x)
```

Arguments

x	A Dewey Decimal call number

Value

Returns the most general subject classification using the hundreds places from the DDC. Returns NA if the DDC looks invalid

Examples

```
get_dewey_decimal_subject_class("709.05")     # Arts

get_dewey_decimal_subject_class("823.912")
# Literature (Belles-lettres) and rhetoric

# vectorized
get_dewey_decimal_subject_class(c("709.05", "invalid", NA, "823.912"))
# c("Arts", NA, NA, "Literature (Belles-lettres) and rhetoric")
```

get_dewey_decimal_subject_division

Conversion from Dewey Decimal call numbers to second-level subject description

Description

Takes a string representation of a Dewey Decimal call number (DDC) and returns it's subject description. This uses the first two digits of the DDC number and returns the second most general subject classification.

Usage

```
get_dewey_decimal_subject_division(x)
```

Arguments

x	A Dewey Decimal call number

Value

Returns the most general subject classification using the first two digits from the DDC. Returns NA if the DDC looks invalid

Examples

```
get_dewey_decimal_subject_division("709.05")     # Arts

get_dewey_decimal_subject_division("823.912")
# "English and Old English literatures"

# vectorized
get_dewey_decimal_subject_division(c("709.05", "invalid", NA, "823.912"))
# c("Arts", NA, NA, "English and Old English literatures")
```

---

get_dewey_decimal_subject_section

Conversion from Dewey Decimal call numbers to third-level subject description

---

Description

Takes a string representation of a Dewey Decimal call number (DDC) and returns it's subject description. This uses the first three digits of the DDC number and returns the third most general subject classification.

Usage

```
get_dewey_decimal_subject_section(x)
```

Arguments

x	A Dewey Decimal call number

Value

Returns the most general subject sectionification using the first three digits from the DDC. Returns NA if the DDC looks invalid

Examples

```
get_dewey_decimal_subject_section("709.05")
# "History, geographic treatment, biography"

get_dewey_decimal_subject_section("823.912")
# "English fiction"

# vectorized
get_dewey_decimal_subject_section(c("709.05", "invalid", NA, "823.912"))
# c("History, geographic treatment, biography", NA, NA,
#   "English fiction")
```

---

get_isbn_10_check_digit

Get ISBN 10 check digit

Description

Takes a string representation of an ISBN 10 and returns the check digit that satisfies the necessary condition. It can take a 10 digit string (and ignore the already extant check digit) or a 9 digit string (without the last digit)

Usage

```
get_isbn_10_check_digit(x, allow.hyphens = FALSE, errors.as.nas = FALSE)
```

Arguments

x	A string of 9 or 10 digits
allow.hyphens	A logical indicating whether the hyphen separator should be allowed (default is FALSE)
errors.as.nas	return NA if error instead of throwing error (default is FALSE)

Value

Returns the character check digit that satifies the mod 11 condition. Returns "X" if 10. Returns NA if input is NA

Examples

```
get_isbn_10_check_digit("012491540X")
get_isbn_10_check_digit("0-124-91540-X", allow.hyphens=TRUE)

# nine digit string
get_isbn_10_check_digit("900403781")
```

```
get_isbn_10_check_digit("onetwothre", errors.as.nas=TRUE)  # NA

# vectorized
get_isbn_10_check_digit(c("012491540X", "9004037810", "900403781"))
```

`get_isbn_13_check_digit` *Get ISBN 13 check digit*

Description

Takes a string representation of an ISBN 13 and returns the check digit that satisfies the necessary condition. It can take a 13 digit string (and ignore the already extant check digit) or a 12 digit string (without the last digit)

Usage

```
get_isbn_13_check_digit(x, allow.hyphens = FALSE, errors.as.nas = FALSE)
```

Arguments

`x`	A string of 12 or 13 digits
`allow.hyphens`	A logical indicating whether the hyphen separator should be allowed (default is `FALSE`)
`errors.as.nas`	return NA if error instead of throwing error (default is `FALSE`)

Value

Returns the character check digit that satifies the mod 10 condition. Returns NA if input is NA

Examples

```
get_isbn_13_check_digit("9780306406157")

# 12 digit string
get_isbn_13_check_digit("978030640615")

get_isbn_13_check_digit("onetwothreefo", errors.as.nas=TRUE)  # NA

# vectorized
get_isbn_13_check_digit(c("9780306406157", "9783161484100"))
```

`get_issn_check_digit` *Get ISSN check digit*

Description

Takes a string representation of an ISSN and returns the check digit that satisfies the necessary condition. It can take a 8 digit string (and ignore the already extant check digit) or a 7 digit string (without the last digit)

Usage

```
get_issn_check_digit(x, allow.hyphens = FALSE, errors.as.nas = FALSE)
```

Arguments

`x`	A string of 7 or 8 digits
`allow.hyphens`	A logical indicating whether the hyphen separator should be allowed (default is `FALSE`)
`errors.as.nas`	return NA if error instead of throwing error (default is `FALSE`)

Value

Returns the character check digit that satifies the mod 11 condition. Returns "X" if 10. Returns NA if input is NA

Examples

```
get_issn_check_digit("03785955")

get_issn_check_digit("2434-561X", allow.hyphens=TRUE)

# nine digit string
get_issn_check_digit("0378595")

# vectorized
get_issn_check_digit(c("0378595", "2434561X", NA))
```

get_language_from_code

Conversion from language code to language name

Description

Takes a language code (defined in the Marc standards) and returns the language name.

Usage

```
get_language_from_code(x)
```

Arguments

x	A language code (defined in the Marc standards) or a vector of language codes

Value

Returns the language name. NA if cannot be matched to language in standard.

Examples

```
get_language_from_code("yor")
# Yoruba

# tolerant of case and leading/trailing whitespace
get_language_from_code(c("yor", " SPA ", "not-a-language", "nah", NA))
# c("Yoruba", "Spanish", NA, "Nahuatl", NA)
```

get_lc_call_first_letter

Get the first letter of LC Call Number

Description

Takes a string representation of a Library of Congress call number and returns the first letter if and only if the LC Call Number is valid

Usage

```
get_lc_call_first_letter(x, allow.bare = FALSE)
```

Arguments

x	A Library of Congress call number (string)
allow.bare	A logical indicating whether an LC Call with only the letters should be considered valid (default is FALSE)

Value

Returns first letter or NA if invalid

Examples

```
get_lc_call_first_letter("Q172.5.E77")
# Q
get_lc_call_first_letter("AF172.5.E77")
# NA

# vectorized
get_lc_call_first_letter(c("Q 172.5", "AF172", "PR6023.A93"))
# Q NA P
```

---

get_lc_call_subject_classification

Conversion from Library of Congress Call number to subject classification

---

Description

Takes a string representation of a Library of Congress call number and returns either the broad subject classification description (default) based on the first letter, or a second-level subclassification description based on the all the letters

Usage

```
get_lc_call_subject_classification(
  x,
  subclassification = FALSE,
  already.parsed = FALSE,
  allow.bare = TRUE
)
```

Arguments

x	A Library of Congress call number (string)
subclassification	A logical indicating whether the letters of call number past the first should be used to match to a subject subclassification (default is FALSE)

already.parsed Skips the extraction of the subject letters and jumps to the subject matching (default is FALSE)

allow.bare A logical indicating whether an LC Call with only the letters should be considered valid (default is TRUE)

Value

Returns either the broad (top-level) subject classification description or the second level subject subclassification description. Returns "NA" if no subject could not be matched or call number is invalid

Examples

```
get_lc_call_subject_classification("ND 237.S18 $b S87 1997")
# Fine Arts

get_lc_call_subject_classification("ND 237.S18 $b S87 1997", subclassification=TRUE)
# Painting

get_lc_call_subject_classification("PQ2246.M3")
# Language and Literature

get_lc_call_subject_classification("PQ2246.M3",
                                   subclassification=TRUE)
# "French, Italian, Spanish, and Portuguese literature"

get_lc_call_subject_classification("PQ2246.M3", already.parsed=TRUE)
# NA

get_lc_call_subject_classification("PQ", already.parsed=TRUE,
                                   subclassification=TRUE)
# "French, Italian, Spanish, and Portuguese literature"

# vectorized
get_lc_call_subject_classification(c("ND 237", "\\\\$a ND 2", "PQ2246.M3"),
                                   subclassification=TRUE)
# c("Painting", NA, "French, Italian, Spanish, and Portuguese literature")
```

---

is_valid_isbn_10 *Return TRUE if valid ISBN 10*

Description

Takes a string representation of an ISBN 10 verifies that it is valid. An ISBN 10 is valid if it is a 10 digit string or a 9 digit string with a terminal "X" AND the check digit matches

Usage

```
is_valid_isbn_10(x, allow.hyphens = TRUE, lower.x.allowed = TRUE)
```

Arguments

x	A string of 10 digits or 9 digits with terminal "X"
allow.hyphens	A logical indicating whether the hyphen separator should be allowed (default is TRUE)
lower.x.allowed	A logical indicating whether ISBN 10s with a check digit with a lower-case "x" should be treated as valid (default is TRUE)

Value

Returns TRUE if checks pass, FALSE if not, and NA if NA

Examples

```
is_valid_isbn_10("012491540X")    # TRUE
is_valid_isbn_10("0-124-91540-X") # TRUE

# vectorized
is_valid_isbn_10(c("012491540X", "9004037812"))      # TRUE FALSE
is_valid_isbn_10(c("012491540X", "hubo un tiempo"))  # TRUE FALSE
```

---

is_valid_isbn_13 *Return TRUE if valid ISBN 13*

Description

Takes a string representation of an ISBN 13 verifies that it is valid. An ISBN 13 is valid if it is a 13 digit string and the check digit matches

Usage

```
is_valid_isbn_13(x, allow.hyphens = TRUE)
```

Arguments

x	A string of 13
allow.hyphens	A logical indicating whether the hyphen separator should be allowed (default is TRUE)

Value

Returns TRUE if checks pass, FALSE if not, and NA if NA

Examples

```
is_valid_isbn_13("9780306406157")          # TRUE
is_valid_isbn_13("978-0-306-40615-7")      # TRUE

# vectorized
is_valid_isbn_10(c("012491540X", "9004037812"))  # TRUE FALSE
is_valid_isbn_13(c("978-0-306-40615-7", "9783161484103"))  # TRUE FALSE
is_valid_isbn_13(c("978-0-306-40615-7", "hubo un tiempo"))  # TRUE FALSE
```

---

is_valid_issn *Return TRUE if valid ISSN*

Description

Takes a string representation of an ISSN verifies that it is valid. An ISSN is valid if it is a 8 digit string or a 7 digit string with a terminal "X" AND the check digit matches

Usage

```
is_valid_issn(x, allow.hyphens = TRUE, lower.x.allowed = TRUE)
```

Arguments

x	A string of 8 digits or 7 digits with terminal "X"
allow.hyphens	A logical indicating whether the hyphen separator should be allowed (default is TRUE)
lower.x.allowed	A logical indicating whether ISSNs with a check digit with a lower-case "x" should be treated as valid (default is TRUE)

Value

Returns TRUE if checks pass, FALSE if not, and NA if NA

Examples

```
is_valid_issn("2434561X")           # TRUE
is_valid_issn("2434-561X")          # TRUE

# vectorized

is_valid_issn(c("2434-561X", "2434-5611", "0378-5955", NA))
# TRUE FALSE TRUE NA
```

is_valid_lc_call *Check if LC Call Number is valid*

Description

Takes a string representation of a Library of Congress call number and returns either TRUE or FALSE based on whether or not the input fits the canonical LC Call Number pattern

Usage

```
is_valid_lc_call(x, allow.bare = FALSE)
```

Arguments

x	A Library of Congress call number (string)
allow.bare	A logical indicating whether an LC Call with only the letters should be considered valid (default is FALSE)

Value

Returns either TRUE or FALSE based on whether the call number is valid

Examples

```
is_valid_lc_call("Q172.5.E77")
# TRUE
is_valid_lc_call("AF172.5.E77")
# FALSE

# vectorized
is_valid_lc_call(c("Q 172.5", "AF172", "PR6023.A93"))
# TRUE FALSE TRUE
```

language_code_crosswalk *language code / language crosswalk*

Description

A cross-walk between the language code and it's human readable version

Usage

```
data(language_code_crosswalk)
```

Format

An object of class "data.table";

Source

https://www.loc.gov/marc/languages/language_code.html

---

lc_subject_classification

LC Call Subject Code Classification / Classification name crosswalk

---

Description

A cross-walk between the LC Subject classification and and it's human readable name (first letter of LC Call Number)

Usage

```
data(lc_subject_classification)
```

Format

An object of class "data.table";

Source

https://www.loc.gov/catdir/cpso/lcco/

---

lc_subject_subclassification

LC Subject Subclassification / Subclassification name crosswalk

---

Description

A cross-walk between the LC Subject subclassification and and it's human readable name (all letters in an LC Call Number)

Usage

```
data(lc_subject_subclassification)
```

Format

An object of class "data.table";

Source

https://www.loc.gov/catdir/cpso/lcco/

`loc_permalink_from_lccn` *Get Library of Congress catalog permalinks from LCCNs*

Description

Takes a string representation of an LCCNs. Returns permalinks to the Library of Congress catalog entries using those LCCNs.

Usage

```
loc_permalink_from_lccn(x, normalize = TRUE, format = "")
```

Arguments

`x`	A string (or vector of strings) of LCCNs
`normalize`	a logical indicating whether the LCCN should be normalized prior to creating the permalink (default is `TRUE`)
`format`	One of "", "marcxml", "mods", "mads", or "dublin" to return the link to the main permalink page, or the link directly to the record's MARCXml, MODS, MADS, or Dublin Core representation, respectively.

Details

If normalize=TRUE and the LCCN is invalid, the permalink is NA. If normalize=FALSE, the permalink may be invalid. No validity check on the URL is performed

Value

Library of Congress permalinks using LCCNs.

Examples

```
loc_permalink_from_lccn("n78-890351")        # "https://lccn.loc.gov/n78890351"
loc_permalink_from_lccn("85-2 ")             # "https://lccn.loc.gov/85000002"
loc_permalink_from_lccn("75-425165//r75")    # "https://lccn.loc.gov/75425165"

# vectorized
loc_permalink_from_lccn(c("###78890351#", NA, "n78-890351"))

# MARCXML metadata format
loc_permalink_from_lccn("73167510", format="marcxml")
# "https://lccn.loc.gov/73167510/marcxml"
```

`marc_008_get_info` *Get info from MARC control field 008*

Description

Takes one or more MARC 008 fields (string/strings) and returns a `data.table` containing the publication date, publication place code, and language code.

Usage

```
marc_008_get_info(
  x,
  original.pub.date = FALSE,
  include.questionable.dates = FALSE
)
```

Arguments

`x`	A string (or vector of strings) of LCCNs
`original.pub.date`	If `TRUE` and if applicable, return the original publication date, not the re-issue publication date. (Default is `FALSE`)
`include.questionable.dates`	A logical indicating whether "questionable" dates should be replaced with `NA`. Questionable dates are when the "type of date" in character position 06 is "q". (default is `FALSE`)

Details

If any date element is "unknown" (contains a "u"), the returned date is NA. The returned date is always an integer.

Value

A `data.table`

Examples

```
# reissue publication date
marc_008_get_info("950622r19701880ru            000 0 rus d")
#     pub_date pub_place_code lang_code
#        <int>         <char>    <char>
#  1:     1970             ru       rus

# The Brothers Karamazov (1970 reissue but original publication date)
marc_008_get_info("950622r19701880ru            000 0 rus d",
                  original.pub.date=TRUE)
```

```
#    pub_date pub_place_code lang_code
#       <int>         <char>    <char>
# 1:     1880             ru       rus

# vectorized
marc_008_get_info(c("101106s1992    gr            000 1 gre d", NA,
                    "180528s2017    ag            000 j spa d"))
#      pub_date pub_place_code lang_code
#         <int>         <char>    <char>
#   1:     1992             gr       gre
#   2:       NA           <NA>      <NA>
#   3:     2017             ag       spa
```

`marc_leader_get_info` *Get info from MARC leader*

Description

Takes one or more MARC leaders (string/strings) and returns a `data.table` containing the record type and bib level

Usage

```
marc_leader_get_info(x)
```

Arguments

`x`	A string (or vector of strings) of MARC leaders

Value

A `data.table`

Examples

```
marc_leader_get_info("00000cam a22000008i 4500")
#          record_type      bib_level
#               <char>         <char>
# 1: Language Material Monograph/Item

# vectorized
marc_leader_get_info(c("00000cam a2200000Ma 4500", NA,
                       "00000cem a2200000Ma 4500"))
#              record_type      bib_level
#                    <char>         <char>
#  1:     Language Material Monograph/Item
#  2:                   <NA>           <NA>
#  3: Cartographic material Monograph/Item
```

normalize_isbn *Attempt to enforce validity and canonical form to an ISBN*

Description

Takes a string representation of an ISBN (10 or 13). This function uses tries to normalize the string as a ISBN 13, then an ISBN 10. If one of those methods are able to salvage the ISBN, the canonicalized ISBN is returned. User can specify whether "aggressive" measures should be taken to salvage the malformed ISBN string.

Usage

```
normalize_isbn(x, aggressive = TRUE, convert.to.isbn.13 = FALSE)
```

Arguments

x	A string
aggressive	A logical indicating whether aggressive measures should be taken to try to get the "ISBN 10" into a valid form. See "Details" for more info (default is TRUE)
convert.to.isbn.13	A logical indicating whether the ISBN 10 should be converted into an ISBN 13 (default is FALSE)

Details

If aggressive is TRUE, aggressive measures are taken to try to salvage the malformed ISBN string. Since this function attempts to salvage both an ISBN 10 and 13, to learn about examples of the aggressive methods, see normalize_isbn_10 and normalize_isbn_13

Value

Returns valid ISBN if possible, NA if not

See Also

normalize_isbn_10 normalize_isbn_13

Examples

```
normalize_isbn("012491540x")                          # "012491540X"
normalize_isbn("012491540x", convert.to.isbn.13=TRUE)
"9780124915404"

# vectorized
normalize_isbn(c("513213012491540x245",
                 "978966819^*!X7918",
                 NA,
```

```
                "9781572411579799781572411579"))
# "012491540X", "9789668197918", NA, "9781572411579"
```

`normalize_isbn_10` *Attempt to enforce validity and canonical form to ISBN 10*

Description

Takes a string representation of an ISBN 10. Strips all non-digit and non-"X" characters and checks if it is valid (whether the check digit works out, etc). User can specify whether "aggressive" measures should be taken to salvage the malformed ISBN 10 string.

Usage

```
normalize_isbn_10(x, aggressive = TRUE, convert.to.isbn.13 = FALSE)
```

Arguments

`x`	A string
`aggressive`	A logical indicating whether aggressive measures should be taken to try to get the "ISBN 10" into a valid form. See "Details" for more info (default is `TRUE`)
`convert.to.isbn.13`	A logical indicating whether the ISBN 10 should be converted into an ISBN 13 (default is `FALSE`)

Details

If `aggressive` is TRUE, aggressive measures are taken to try to salvage the malformed ISBN 10 string. If the ISBN 10, for example, is 9 digits, and either adding an "X" to the end, or leading "0"s fix it, this function will return the salvaged ISBN 10. If the ISBN 10 has garbage digits/characters in the front and has an "X" check digit, it will return the salvaged ISBN 10.

Value

Returns valid ISBN 10 if possible, NA if not

See Also

`normalize_isbn` `normalize_isbn_13`

Examples

```
normalize_isbn_10("012491540x")                    # "012491540X"
normalize_isbn_10("012491540x xe32ea")             # "012491540X"
normalize_isbn_10("012491540x", convert.to.isbn.13=TRUE)
# "9780124915404"
normalize_isbn_10("513213012491540x")              # "012491540X"
```

`normalize_isbn_13` *Attempt to enforce validity and canonical form to ISBN 13*

Description

Takes a string representation of an ISBN 13. Strips all non-digit characters and checks if it is valid (whether the check digit works out, etc). User can specify whether "aggressive" measures should be taken to salvage the malformed ISBN 13 string.

Usage

```
normalize_isbn_13(x, aggressive = TRUE)
```

Arguments

`x`	A string
`aggressive`	A logical indicating whether aggressive measures should be taken to try to get the "ISBN 13" into a valid form. See "Details" for more info (default is `TRUE`)

Details

If `aggressive` is TRUE, aggressive measures are taken to try to salvage the malformed ISBN 13 string. If the ISBN 13, for example, is more than 13 characters, this function will attempt to make a valid ISBN 13 from the first 13 digits.

Value

Returns valid ISBN 13 if possible, NA if not

See Also

`normalize_isbn` `normalize_isbn_10`

Examples

```
normalize_isbn_13("978966819^*!X7918")        # "9789668197918"

# vectorized
normalize_isbn_13(c("978-9-66-819791-8", "__9__781572411579"))
# "9789668197918" "9781572411579"
```

normalize_issn *Attempt to enforce validity and canonical form to ISSN*

Description

Takes a string representation of an ISSN. Strips all non-digit and non-"X" characters and checks if it is valid (whether the check digit works out, etc). User can specify whether "aggressive" measures should be taken to salvage the malformed ISSN string.

Usage

```
normalize_issn(x, aggressive = TRUE, pretty = FALSE)
```

Arguments

x	A string
aggressive	A logical indicating whether aggressive measures should be taken to try to get the "ISSN" into a valid form. See "Details" for more info (default is TRUE)
pretty	A logical indicating whether the ISSN should be prettily hyphenated (default is FALSE)

Details

If aggressive is TRUE, aggressive measures are taken to try to salvage the malformed ISSN string. If the ISSN, for example, is 7 digits, and either adding an "X" to the end, or leading "0"s fix it, this function will return the salvaged ISSN. If the ISSN has garbage digits/characters in the front and has an "X" check digit, it will return the salvaged ISSN.

Value

Returns valid ISSN if possible, NA if not

Examples

```
# adds leading zero
normalize_issn("3785955")                          # "03785955"

# adds X to 7 digit ISSN if valid
normalize_issn("2434561")                          # "2434561X"

normalize_issn("2434561", pretty=TRUE)             # "2434-561X"

# finds correct sequence
normalize_issn("21335212434561X")                  # "2434561X"

# vectorized
normalize_issn(c("__2434__561X", "2434561", "21335212434561X"))
```

```
# "2434561X" "2434561X" "2434561X"
```

`normalize_lccn` *Attempt to enforce validity of LCCN and convert to normalized form*

Description

Takes a string representation of an LCCN. Returns a normalized one

Usage

```
normalize_lccn(userlccns, allow.hyphens = TRUE)
```

Arguments

`userlccns`	A string (or vector of strings) of LCCNs
`allow.hyphens`	a logical indicating whether hyphens separating the year and serial should be handled. Adds complexity and time to the function. (default is `TRUE`)

Details

Normalization procedure is documented here: https://www.loc.gov/marc/lccn-namespace.html

This does not include revisions or use "#" as a padding character The normalized LCCN is not always the same number of characters

Value

Returns valid LCCN if possible, NA if not

Examples

```
normalize_lccn("n 78890351 ")                 # "n78890351"

normalize_lccn("###78890351#")                # "78890351"

normalize_lccn(" 79139101 /AC/r932")          #  "79139101"

normalize_lccn("85-2 ")                       #  "85000002"
normalize_lccn("85-2 ", allow.hyphens=FALSE)  #  NA

# vectorized
normalize_lccn(c("85-2 ", " 79139101 /AC/r932", "n 78890351 "))
# c("85000002", "79139101", "n78890351"))
```

`oclc_classify_link_from_standard_num` *Get OCLC Classify link from a standard number*

Description

Takes a string representation of ISSNs, ISBNs, UPC, or OCLC numbers. Returns a link to the OCLC's experimental classify service which provides the most frequent call numbers, FAST subject headings, etc...

Usage

```
oclc_classify_link_from_standard_num(x)
```

Arguments

`x`	A string (or vector of strings) of a standard number. Must be an ISSN, ISBN, UPC, and/or OCLC numbers.

Details

Since this can take a variety of standard numbers, no normalization can be performed. The numbers much be normalized before the call to this function. No validity check on the URL is performed

Value

Links to OCLC's Classify web service

Examples

```
oclc_classify_link_from_standard_num("629725006")
# "http://classify.oclc.org/classify2/ClassifyDemo?search-standnum-txt=629725006&startRec=0"

oclc_classify_link_from_standard_num(c("039333712X", NA, "629725006"))
# [1] "http://classify.oclc.org/classify2/ClassifyDemo?search-standnum-txt=039333712X&startRec=0"
# [2] NA
# [3] "http://classify.oclc.org/classify2/ClassifyDemo?search-standnum-txt=629725006&startRec=0"
```

recombine_with_sep_closure *Return a function that will combine/contatenate a vector*

Description

This function takes and optional separator, and returns a function that takes a vector and pastes the elements of that vector together

Usage

```
recombine_with_sep_closure(sep = ";")
```

Arguments

sep	A character to use in between the elements (default is a semicolon character)

Details

Can be used as a reduction function in split_map_filter_reduce

Value

Returns a closure/function

See Also

split_map_filter_reduce

paste

Examples

```
lambda <- recombine_with_sep_closure()
lambda(c(8, 6, 7))                              # "8;6;7"

# directly
recombine_with_sep_closure()(c(8,6,7))          #  "8;6;7"
lambda <- recombine_with_sep_closure(" ")
lambda(c("this", "that", NA,"the-other"))       # "this that NA the-other"
```

`remove_duplicates_and_nas` *Remove duplicate elements and NAs from a vector*

Description

Takes a vector and returns the same vector without duplicate elements and without NA values

Usage

```
remove_duplicates_and_nas(x)
```

Arguments

x	A vector

Details

Can be used as a filtering function in `split_map_filter_reduce`

Value

Returns vector with duplicates and NAs removed

See Also

`split_map_filter_reduce`

Examples

```
remove_duplicates_and_nas(c(8, 6, 7, 5, 3, 0, 9, 6, NA, 3))
# 8 6 7 5 3 0 9

remove_duplicates_and_nas(c(NA, NA))
# NA
```

set_lb_attribute *Set special libbib attribute on object*

Description

Takes an object, attribute name, and a value and sets a special libbib attribute by reference

Usage

```
set_lb_attribute(x, type, value)
```

Arguments

x	An object to set the attribute on
type	The name of the attribute to set. `lb.` will be appended to this attribute name. For example, if this argument is `source`, and attribute called `lb.source` will be set on the object with the value specified
value	The value of the attribute

Value

Nothing, since the object is modified by reference.

Examples

```
set_lb_attribute(mtcars, "source", "R built-in dataset")

versicolor <- iris[iris$Species=="versicolor", ]
set_lb_attribute(versicolor, "note", "modified built-in dataset")
attributes(versicolor)$lb.note
# [1] "modified built-in dataset"
```

set_lb_date *Set special libbib date attribute on object*

Description

Takes an object and a date and sets a special attribute, "lb.date" by reference

Usage

```
set_lb_date(x, value)
```

Arguments

x	An object to set the attribute on
value	Either a value of class Date or a string in ISO 8601 date format (yyyy-mm-dd) which will be converted into a Date

Value

Nothing, since the object is modified by reference.

Examples

```
set_lb_date(mtcars, "2021-05-08")
attributes(mtcars)$lb.date
# [1] "2021-05-08

set_lb_date(mtcars, Sys.Date())
```

---

split_map_filter_reduce

Split, Map, Filter, and Reduce a string vector

Description

This function takes a vector of strings, splits those strings on a particular character; string; or regex patters, applies a user-specified function to each sub-element of the now split element, filters those sub-elements using a user-specified function, and, finally, recombines each element's sub-elements using a user specified reduction function.

Usage

```
split_map_filter_reduce(
  x,
  sep = ";",
  fixed = TRUE,
  mapfun = identity,
  filterfun = identity,
  reduxfun = car,
  cl = 0
)
```

Arguments

x	A vector of strings
sep	A character to use containing a character, string, or regular expression pattern to split each element by. If fixed=TRUE, the separator will be used exactly; If not, a Perl-compatible regular expression can be used (default is ";")

fixed	Should it be split by a fixed string/character or a regular expression (default is TRUE)
mapfun	A vectorized function that will be applied to the sub-elements (after splitting) of each element in x (default is identity which would leave the sub-elements unchanged)
filterfun	A vectorized function that, when given a vector returns the same vector with unwanted elements removed (default is identity which would not remove any sub-elements)
reduxfun	A vectorized function that, when given a vector, will combine all of it's elements into one value (default is car, which would return the first element only)
cl	An integer to indicate the number of child processes should be used to parallelize the work-load. If 0, the workload will not be parallelized. Can also take a cluster object created by 'makeCluster' (default is 0)

Details

Since this operation cannot be vectorized, if the user specifies a non-zero cl argument, the workload will be parallelized and cl many child processes will be spawned to do the work. The package pbapply will be used to do this.

See examples for more information and ideas on why this might be useful for, as an example, batch normalizing ISBNs that, for each bibliographic record, is separated by a semicolon

Value

Returns a vector

See Also

car

remove_duplicates_and_nas

recombine_with_sep_closure

Examples

```
someisbns <- c("9782711875177;garbage-isbn;2711875172;2844268900",
               "1861897952; 978-1-86189-795-4")

# will return only the first ISBN for each record
split_map_filter_reduce(someisbns)
# "9782711875177" "1861897952"

# will return only the first ISBN for each record, after normalizing
# each ISBN
split_map_filter_reduce(someisbns, mapfun=function(x){normalize_isbn(x, convert.to.isbn.13=TRUE)})
# "9782711875177" "9781861897954"

# will return all ISBNs, for each record, separated by a semicolon
```

```
# after applying normalize_isbn to each ISBN
# note the duplicates introduced after normalization occurs
split_map_filter_reduce(someisbns, mapfun=function(x){normalize_isbn(x, convert.to.isbn.13=TRUE)},
                        reduxfun=recombine_with_sep_closure())
# "9782711875177;NA;9782711875177;9782844268907" "9781861897954;9781861897954"

# After splitting each items ISBN list by semicolon, this runs
# normalize_isbn in each of them. Duplicates are produced when
# an ISBN 10 converts to an ISBN 13 that is already in the ISBN
# list for the item. NAs are produced when an ISBN fails to normalize.
# Then, all duplicates and NAs are removed. Finally, the remaining
# ISBNs, for each record, are pasted together using a space as a separator
split_map_filter_reduce(someisbns, mapfun=function(x){normalize_isbn(x, convert.to.isbn.13=TRUE)},
                        filterfun=remove_duplicates_and_nas,
                        reduxfun=recombine_with_sep_closure(" "))
# "9782711875177 9782844268907" "9781861897954"
```

`worldcat_api_bib_read_info_by`

Get bibliographic info from a standard number

Description

Access the results of a WorldCat bib read API search by ISBN, ISSN, or OCLC number. The MARCXML returned by the API is parsed and the function returns a `data.table` containing the oclc number, ISBN, ISSN, title, author, MARC leader, and the 008 control field, respectively.

Usage

```
worldcat_api_bib_read_info_by_oclc(
  x,
  wskey = getOption("libbib.wskey", NULL),
  more = FALSE,
  debug = FALSE
)

worldcat_api_bib_read_info_by_isbn(
  x,
  wskey = getOption("libbib.wskey", NULL),
  more = FALSE,
  debug = FALSE
)

worldcat_api_bib_read_info_by_issn(
  x,
  wskey = getOption("libbib.wskey", NULL),
  more = FALSE,
```

```
  debug = FALSE
)
```

Arguments

x	A string representation of the standard number that the function chosen accepts.
wskey	A WorldCat API key (default is `getOption("libbib.wskey")`)
more	A logical indicating whether more infomation from the MARCXML should be returned (publisher, bib level etc....) In the interest of memory consumption, the default is `FALSE`
debug	A logical indicating whether the HTTP and bib read API responses should be printed (for debugging) (default is `FALSE`)

Details

Though this function gets all standard numbers (OCLC, ISBN, ISSN) from the MARCXML, the standard number that was supplied to the function will be the one in the returned `data.table`. For example, if you use `worldcat_api_bib_read_info_by_isbn`, the returned `data.table` will have that ISBN in the ISBN column, not the ISBN in the MARC record.

If something went wrong, all columns (except the one corresponding to the supplied standard number) will be NA.

This function is helpful to call before attempting to use the Location and Classify API functions as it will ensure that the supplied standard number actually resolves to a OCLC work.

As with all API access functions in this package, it's up to the user to limit their API usage so as to not get blocked. These functions are deliberately not vectorized for this reason; they only accept one standard number at a time.

This (and other) WorldCat API communication functions require a WorldCat API key. The easiest way to use these functions is to set a global options with your key: `options("libbib.wskey"="YOUR KEY HERE")`

Final note: all of these API functions seem to work better with OCLC numbers than any other standard number. If multiple standard numbers are available, using the OCLC number is always preferred.

Value

A `data.table` containing the OCLC number, ISBN, ISSN, title, author, MARC leader, and the 008 control field, respectively,

Examples

```
## Not run:
worldcat_api_bib_read_info_by_isbn("9780984201006")
#         oclc          isbn   issn                        title
#       <char>        <char> <char>                       <char>
# 1: 462894360 9780984201006   <NA> The Great Debate about Art /
#          author                  leader
```

```
#          <char>                   <char>
# 1: Harris, Roy, 00000cam a2200000 a 4500
#                                       oh08
#                                      <char>
# 1: 091031s2010    ilua     b    000 0 eng c

worldcat_api_bib_read_info_by_issn("13602365")
#        oclc   isbn     issn                        title author
#      <char> <char>   <char>                       <char> <char>
# 1: 37787277   <NA> 14664410 The journal of architecture.   <NA>
#                      leader                                    oh08
#                      <char>                                   <char>
# 1: 00000cas a2200000 a 4500 971015c19969999enkbx pso     0   a0eng c


## End(Not run)
```

`worldcat_api_classify_by` *Search WorldCat classify API by ISBN, ISSN, or OCLC number*

Description

Access the results of a WorldCat classify API search by ISBN, ISSN, or OCLC number to get the most frequent call numbers (DDC and LCC) associated with a work. Returns a `data.table` with those call numbers and various other metadata. See "Details" for more information.

Usage

```
worldcat_api_classify_by_oclc(x, debug = FALSE)

worldcat_api_classify_by_isbn(x, debug = FALSE)

worldcat_api_classify_by_issn(x, debug = FALSE)
```

Arguments

`x`	A string representation of the standard number that the function chosen accepts.
`debug`	A logical indicating whether the HTTP and classify API responses should be printed (for debugging) (default is `FALSE`)

Details

The returned `data.table` contains fields for various pieces of metadata returned by the API request. These fields include the ISBN/ISSN/OCLC number used, title of work, author, total number of holdings, total number of electronic holdings, call number type, call number recommendation (by most popular), number of holdings using that call number, the HTTP status code, and the Classify API response code.

For each ISBN/ISSN/OCLC number used, two rows will be returned; one for the DDC and one for the LCC. Common information (work metadata) will the the same in both rows. If one of the call numbers is missing, the recommendation and holdings fields will be NA.

The API can be persnickety, and there are many things that can go wrong. For example, the API can respond with multiple works for a single standard number (ISBN 9780900565748, for example). If this happens, no attempt is made to follow one of the results, and the returned `data.table` will return no useful information.

If the `http_status_code` is 200 and the `classify_response_code` is 0, you've received good results.If the `classify_response_code` is 4, the standard number may have returned multiple works.

The `http_status_code` should never not be 200.

If something went wrong (for example, the status/response codes are not 200 and 0, respectively), you may want to re-run the function call with `print.api.responses` set to `TRUE`. This will print the HTTP status code and the raw XML text response from the API.

As with all API access functions in this package, it's up to the user to limit their API usage so as to not get blocked. These functions are deliberately not vectorized for this reason; they only accept one standard number at a time.

Final note: all of these API functions seem to work better with OCLC numbers than any other standard number. If multiple standard numbers are available, using the OCLC number is always preferred.

Value

A `data.table` with most popular DDC and LCC call numbers and various other metadata. See "Details" for more information.

Examples

```
## Not run:
  worldcat_api_classify_by_oclc("93976650")
   #         oclc   title           author total_holdings total_eholdings call_type
   #       <char>  <char>           <char>          <int>           <int>    <char>
   # 1: 939766505 Lobster King, Richard J.            244             534       DDC
   # 2: 939766505 Lobster King, Richard J.            244             534       LCC
   #    recommendation holdings http_status_code classify_response_code
   #            <char>   <char>            <int>                 <int>
   # 1:        641.395      767              200                     0
   # 2:      QL444.M33      318              200                     0


## End(Not run)
```

`worldcat_api_locations_by` *Get holding libraries by standard number*

Description

Access the results of a WorldCat location API search by ISBN, ISSN, or OCLC number. Returns a `data.table` with rows corresponding to each holding institution. The columns contain the standard number provided, the institution identifier, the institution name, number of copies held by that insitution, and, by default, the bibliographic information provided by `worldcat_api_bib_read_info_by_...`. This information is helpful to ensure thaat the standard number provided successfully resolved to a single OCLC work.

Usage

```
worldcat_api_locations_by_oclc(
  x,
  location = "10032",
  include.bib.info = TRUE,
  max_libraries = Inf,
  servicelevel = "full",
  frbrGrouping = "on",
  libtype = NULL,
  wskey = getOption("libbib.wskey", NULL),
  print.progress = TRUE,
  debug = FALSE
)

worldcat_api_locations_by_isbn(
  x,
  location = "10032",
  include.bib.info = TRUE,
  max_libraries = Inf,
  servicelevel = "full",
  frbrGrouping = "on",
  libtype = NULL,
  wskey = getOption("libbib.wskey", NULL),
  print.progress = TRUE,
  debug = FALSE
)

worldcat_api_locations_by_issn(
  x,
  location = "10032",
  include.bib.info = TRUE,
  max_libraries = Inf,
  servicelevel = "full",
```

```
  frbrGrouping = "on",
  libtype = NULL,
  wskey = getOption("libbib.wskey", NULL),
  print.progress = TRUE,
  debug = FALSE
)
```

Arguments

x	The standard number to search using. Must be a string.
location	The holding institutions are sorted roughly by geographic proximity to this zip-code, country code, etc... If max_libraries is Inf (the default), the starting location doesn't matter since all holding institutions are returned. Defaults to the zip code of Washington Heights, NYC.
include.bib.info	A logical indicating whether to include bibliographic metadata associated with the work (provided by worldcat_api_bib_read_info_by_...). This is very useful for error checking so default is TRUE.
max_libraries	The maximum number of libraries to return. Must be a number between 0 and 100 or Inf. If Inf (default), the function will automatically make all follow-up requests to retrieve all holding institutions. Beware that each page of 100 institutions counts as one API request. If the bib searched for is popular, set this to non-Inf.
servicelevel	Either "full" (the default) or "default". If "full", the number of holding libraries returned is the same as if a user logged in to an institution when making a WorldCat search. If "default", the results are a subset of WorldCat libraries, namely those that participate in worldcat.org. In this way, the number of holding libraries is tantamount to if a non-logged-in user searched WorldCat. The number of results with "full" is always at least as high as with "default", so the default is "full". If this package is being used in an application where a user is not logged in to an institution, set this to "default". It is up to you to respect the WorldCat API's conditions.
frbrGrouping	With this parameter set to "on" (default), an attempt is made by the WorldCat API to group together similar editions and present only the top held record as the representative record for that group. If not, only institutions holding the exact standard number specified will be returned.
libtype	One of NULL (default), "academic", "public", "government", or "other". NULL will return all library subsets. The others will only search for holdings from insitutions of that library type.
wskey	A WorldCat API key (default is getOption("libbib.wskey"))
print.progress	A logical indicating whether a message should be displayed for each API request. If max_libraries is TRUE a message will be displayed for every group of 100 institutions the function fetches. (default is TRUE)
debug	A logical indicating whether the HTTP and API responses should be printed (for debugging) (default is FALSE)

Details

Numerous parameters are provided that change the API url parameters. See parameter section for details on each.

If something went wrong, most columns (especially the bibliographic info columns) will be NA. You should always check the output.

As with all API access functions in this package, it's up to the user to limit their API usage so as to not get blocked. These functions are deliberately not vectorized for this reason; they only accept one standard number at a time.

This (and other) WorldCat API communication functions require a WorldCat API key. The easiest way to use these functions is to set a global options with your key: `options("libbib.wskey"="YOUR KEY HERE")`

Final note: all of these API functions seem to work better with OCLC numbers than any other standard number. If multiple standard numbers are available, using the OCLC number is always preferred. In this function, for example, searching for ISSN: 14664410 (Journal of Architecture) will (at time of writing) return only one insitution, whereas searching by it's OCLC number (958283020) will yield many more (660, at time of writing, with default parameters).

Value

A `data.table` with each row corresponding to a holding library.

Examples

```
## Not run:
# worldcat_api_locations_by_oclc("877749545", max_libraries=10,
#                                include.bib.info=FALSE)
#         oclc institution_identifier
#       <char>                 <char>
# 1: 877749545                    NLE
# 2: 877749545                    NLW
# 3: 877749545                    EUM
# 4: 877749545                    LTU
# 5: 877749545                    ELU
# 6: 877749545                  UKUAL
#                                   institution_name copies
#                                             <char> <char>
# 1:                  National Library of Scotland      1
# 2:                     National Library of Wales      1
# 3:              University of Manchester Library      1
# 4: University of Leicester, David Wilson Library      1
# 5:     University of London Senate House Library      1
# 6:                 University of the Arts London      1


## End(Not run)
```

`worldcat_api_search` *Use the WorldCat Search API*

Description

Searches WorldCat using a CQL query. Returns a `data.table` containing the bibliographic metadata of the results, along with the total number of results.

Usage

```
worldcat_api_search(
  sru,
  max_records = 10,
  sru_query_assist = getOption("libbib.sru_query_assist", TRUE),
  frbrGrouping = "on",
  start_at = 1,
  wskey = getOption("libbib.wskey", NULL),
  more = TRUE,
  print.progress = TRUE,
  debug = FALSE
)
```

Arguments

`sru`	The search query (in CQL syntax). See `examples` section for some examples.
`max_records`	The maximum number of search results to return. Must be a number between 0 and 100 or `Inf`. If `Inf`, the function will automatically make all follow-up requests to retrieve all search results. For safety, the default is 10.
`sru_query_assist`	A logical indicating whether translation from more human-readable aliases to the SRU search index codes should be allowed. See details for more information. (default is `TRUE`). You can control this parameter globally by setting `options("libbib.sru_query_assist")`.
`frbrGrouping`	With this parameter set to "on" (default), an attempt is made by the WorldCat API to group together similar editions and present only the top held record as the representative record for that group.
`start_at`	The search result to start at (default is 1)
`wskey`	A WorldCat API key (default is `getOption("libbib.wskey")`)
`more`	A logical indicating whether more information from the MARCXML search results should be returned (publisher, bib level, etc....). (Default is `TRUE`)
`print.progress`	A logical indicating whether a message should be displayed for each API request. If `max_records` is `Inf` a message will be displayed for every group of 100 search results the function fetches. (default is `TRUE`)
`debug`	A logical indicating whether the HTTP and API responses should be printed (for debugging) (default is `FALSE`)

Details

There is an entire vignette dedicated to this function; to view it, execute `vignette("using-the-worldcat-search-api")`

By default, this function allows for the usage of more human-readable aliases to the arcane SRU search index codes. This allows you, for example, to search using "$title" instead of "srw.ti". This behavior is controlled using the 'sru_query_assist' parameter. If it is `TRUE` (the default) you can still use the formal search index codes. See `vignette("using-the-worldcat-search-api")` for more information.

As with all API access functions in this package, it's up to the user to limit their API usage so as to not get blocked. These functions are deliberately not vectorized for this reason; they only accept one standard number at a time.

This (and other) WorldCat API communication functions require a WorldCat API key. The easiest way to use these functions is to set a global options with your key: `options("libbib.wskey"="YOUR KEY HERE")`

Value

A `data.table` containing the bibliographic metadata of the results, along with the total number of results.

Examples

```
## Not run:

# A title search for "The Brothers Karamazov"
worldcat_api_search('$title = "Brothers Karamazov"')

# An exact title search for "The Brothers Karamazov"
worldcat_api_search('$title exact "Brothers Karamazov"')

# Search for title "Madame Bovary" by author "Gustave Flaubert"
# in language Greek (all results)
# (queries may span multiple lines)
sru <- '$author = "Gustave Flaubert" and $title="Madame Bovary"
          and $language=greek'
worldcat_api_search(sru, max_records=Inf)

# Hip Hop (subject) materials on Cassette, CD, or wax from years 1987 to 1990
sru <- '(($material_type=cas or $material_type=cda or $material_type=lps)
           and $subject="Rap") and $year="1987-1990"'
worldcat_api_search(sru)

# all materials with keyword "Common Lisp" at The New York Public Library
sru <- '$keyword="common lisp" and $holding_library=NYP'
worldcat_api_search(sru, max_records=Inf)

# 19th century materials on ethics (Dewey code 170s / LC Call prefix BJ)
sru <- '($dewey="17*" or $lc_call="bj*") and $year="18*"'
worldcat_api_search(sru, max_records=Inf)
```

```
# Music (Dewey 780s) materials that are only held by The New York Public
# Library (a "cg" code of 11 means there is only one holding)
# [searching with debugging]
sru <- '$dewey="78*" and $holding_library=NYP
          and $library_holdings_group=11'
worldcat_api_search(sru, debug=TRUE)

Keyword search for "danger music" from year 2010 to present
worldcat_api_search('$keyword="danger music" and $year="2010-"')


## End(Not run)
```

`worldcat_permalink_from_isbn` *Get WorldCat catalog permalinks from ISBNs*

Description

Takes a string representation of ISBNs. Returns permalinks to the WorldCat catalog entries using those ISBNs.

Usage

```
worldcat_permalink_from_isbn(x, normalize = TRUE)
```

Arguments

`x`	A string (or vector of strings) of ISBNs
`normalize`	a logical indicating whether the ISBNs should be normalized prior to creating the permalink (default is `TRUE`)

Details

If normalize=TRUE and the ISBN is invalid, the permalink is NA. If normalize=FALSE, the permalink may be invalid. No validity check on the URL is performed

Value

Worldcat permalinks using ISBNs.

Examples

```
worldcat_permalink_from_isbn("1788393724")
# http://www.worldcat.org/isbn/1788393724

worldcat_permalink_from_isbn("0-124-91540-X")
# http://www.worldcat.org/isbn/012491540X
```

```
worldcat_permalink_from_isbn("0-124-91540-X", normalize=FALSE)
# http://www.worldcat.org/isbn/0-124-91540-X

# vectorized
worldcat_permalink_from_isbn(c("1788393724", NA, "0-124-91540-X"))
```

`worldcat_permalink_from_issn` *Get WorldCat catalog permalinks from ISSNs*

Description

Takes a string representation of ISSNs. Returns permalinks to the WorldCat catalog entries using those ISSNs.

Usage

```
worldcat_permalink_from_issn(x, normalize = TRUE)
```

Arguments

`x`	A string (or vector of strings) of ISSNs
`normalize`	a logical indicating whether the ISSNs should be normalized prior to creating the permalink (default is `TRUE`)

Details

If normalize=TRUE and the ISSN is invalid, the permalink is NA. If normalize=FALSE, the permalink may be invalid. No validity check on the URL is performed

Value

Worldcat permalinks using ISSNs.

Examples

```
worldcat_permalink_from_issn("0968-1221")   # http://www.worldcat.org/issn/0968-1221

worldcat_permalink_from_issn("2434-561X")   # http://www.worldcat.org/issn/2434561X

# vectorized
worldcat_permalink_from_issn(c("0968-1221", NA, "2434-561X"))
```

`worldcat_permalink_from_oclc_number` *Get WorldCat catalog permalinks from OCLC numbers*

Description

Takes a string representation of OCLC numbers. Returns permalinks to the WorldCat catalog entries using those OCLC numbers

Usage

```
worldcat_permalink_from_oclc_number(x)
```

Arguments

x	A string (or vector of strings) of OCLC numbers

Details

No validity check on the URL is performed

Value

Worldcat permalinks using the OCLC numbers

Examples

```
worldcat_permalink_from_oclc_number("1005106045")
# http://www.worldcat.org/oclc/1005106045

# vectorized
worldcat_permalink_from_oclc_number(c("1049727704", NA,
                                      "1005106045"))
```

Index