Package ‘c14bazAAR’

January 12, 2020

Title Download and Prepare C14 Dates from Different Source Databases

Description
Query different C14 date databases and apply basic data cleaning, merging and calibration steps.

Version 1.2.0

URL https://docs.ropensci.org/c14bazAAR,
https://github.com/ropensci/c14bazAAR

BugReports https://github.com/ropensci/c14bazAAR/issues

Depends R (>= 3.4.0)

Language en_GB

License GPL-2 | file LICENSE

Encoding UTF-8

LazyData true

Imports crayon (>= 1.3.4), data.table (>= 1.11.4), dplyr (>= 0.7.2),
graphics, httr (>= 1.4.1), magrittr (>= 1.5), pbapply (>=
1.3-3), rlang (>= 0.1.1), stats (>= 3.4.0), tibble (>= 1.3.3),
tidyr (>= 0.6.3)

Suggests Bchron, countrycode, dataverse, ggplot2, ggridges, globe,
knitr, lwgeom, mapview, openxlsx, plyr, rgeos, rmarkdown,
rnaturalearth, rworldmap, rworldxtra, sf, stringdist, testthat

RoxygenNote 7.0.2

VignetteBuilder knitr

NeedsCompilation no

Author Clemens Schmid [aut, cre, cph]
(<https://orcid.org/0000-0003-3448-5715>),
Dirk Seidensticker [aut] (<https://orcid.org/0000-0002-8155-7702>),
Daniel Knitter [aut] (<https://orcid.org/0000-0003-3014-4497>),
Martin Hinz [aut] (<https://orcid.org/0000-0002-9904-6548>),
David Matzig [aut] (<https://orcid.org/0000-0001-7349-5401>),
Wolfgang Hamer [aut] (<https://orcid.org/0000-0002-5943-5020>),
Kay Schmütz [aut],

Nils Mueller-Scheessel [ctb] (<https://orcid.org/0000-0001-7992-8722>),
Ben Marwick [rev] (<https://orcid.org/0000-0001-7879-4531>),
Enrico R. Crema [rev] (<https://orcid.org/0000-0001-6727-5138>)

Maintainer Clemens Schmid <`clemens@nevrome.de`>

Repository CRAN

Date/Publication 2020-01-12 16:50:02 UTC

R topics documented:

`as.sf` *Convert a* **c14_date_list** *to a sf object*

Description

Most 14C dates have point position information in the coordinates columns **lat** and **lon**. This allows them to be converted to a spatial simple feature collection as provided by the `sf` package. This simplifies for example mapping of the dates.

Usage

```
as.sf(x, quiet = FALSE)

## Default S3 method:
as.sf(x, quiet = FALSE)
```

```
## S3 method for class 'c14_date_list'
as.sf(x, quiet = FALSE)
```

Arguments

`x`	an object of class c14_date_list
`quiet`	suppress warning about the removal of dates without coordinates

Value

an object of class sf

Examples

```
sf_c14 <- as.sf(example_c14_date_list)

## Not run:
library(mapview)
mapview(sf_c14$geom)

## End(Not run)
```

`c14_date_list` **c14_date_list**

Description

The **c14_date_list** is the central data structure of the `c14bazAAR` package. It's a tibble with set of custom methods and variables. Please see the variable_reference table for a description of the variables. Further available variables are ignored.
If an object is of class data.frame or tibble (tbl & tbl_df), it can be converted to an object of class **c14_date_list**. The only requirement is that it contains the essential columns **c14age** and **c14std**. The `as` function adds the string "c14_date_list" to the classes vector of the object and applies `order_variables()`, `enforce_types()` and the helper function `clean_latlon()` to it.

Usage

```
as.c14_date_list(x, ...)

is.c14_date_list(x, ...)

## S3 method for class 'c14_date_list'
format(x, ...)

## S3 method for class 'c14_date_list'
print(x, ...)

## S3 method for class 'c14_date_list'
plot(x, ...)
```

Arguments

x	an object
...	further arguments passed to or from other methods

Examples

```
as.c14_date_list(data.frame(c14age = c(2000, 2500), c14std = c(30, 35)))
is.c14_date_list(5) # FALSE
is.c14_date_list(example_c14_date_list) # TRUE

print(example_c14_date_list)
plot(example_c14_date_list)
```

---

calibrate *Calibrate all valid dates in a* **c14_date_list**

Description

Calibrate all dates in a **c14_date_list** with `Bchron::BchronCalibrate()`. The function provides two different kinds of output variables that are added as new list columns to the input **c14_date_list**: **calprobdistr** and **calrange**. **calrange** is accompanied by **sigma**. See `?Bchron::BchronCalibrate` and `?c14bazAAR:::hdr` for some more information.
calprobdistr: The probability distribution of the individual date for all ages with an individual probability >= 1e-06. For each date there's a data.frame with the columns **calage** and **density**.
calrange: The contiguous ranges which cover the probability interval requested for the individual date. For each date there's a data.frame with the columns **dens** and **from** and **to**.

Usage

```
calibrate(x, choices = c("calrange"), sigma = 2, ...)

## Default S3 method:
calibrate(x, choices = c("calrange"), sigma = 2, ...)

## S3 method for class 'c14_date_list'
calibrate(x, choices = c("calrange"), sigma = 2, ...)
```

Arguments

x	an object of class c14_date_list
choices	whether the result should include the full calibrated probability dataframe ('calprobdistr') or the sigma range ('calrange'). Both arguments may be given at the same time.
sigma	the desired sigma value (1,2,3) for the calibrated sigma ranges
...	passed to Bchron::BchronCalibrate()

Value

an object of class c14_date_list with the additional columns **calprobdistr** or **calrange** and **sigma**

Examples

```
calibrate(
  example_c14_date_list,
  choices = c("calprobdistr", "calrange"),
  sigma = 1
)
```

`classify_material` *Apply material classification on a* **c14_date_list**

Description

Add column **material_thes** with simplified and unified terms for material categories. The classification is manually curated and therefore maybe not up-to-date. It's stored in a material_thesaurus list, and downloaded directly from github with `c14bazAAR::get_material_thesaurus()`. With this setup you can also easily apply own thesaurus tables.

Usage

```
classify_material(
  x,
  material_thesaurus = c14bazAAR::get_material_thesaurus(),
  quiet = FALSE
)

## Default S3 method:
classify_material(
  x,
  material_thesaurus = c14bazAAR::get_material_thesaurus(),
  quiet = FALSE
)

## S3 method for class 'c14_date_list'
classify_material(
  x,
  material_thesaurus = c14bazAAR::get_material_thesaurus(),
  quiet = FALSE
)
```

Arguments

`x`	an object of class c14_date_list
`material_thesaurus`	a thesaurus table
`quiet`	suppress decision log output

Value

an object of class c14_date_list with the additional column **material_thes**

Examples

```
classify_material(
  example_c14_date_list,
  quiet = TRUE
)
```

`coordinate_precision` *Return coordinate precision according to number of digits in the columns* **lat** *and* **lon** *of a* **c14_date_list**

Description

The precision of the coordinates for each date vary greatly. `c14bazAAR::coordinate_precision()` calculates the mean of the possible deviation in meters and adds it to the **c14_date_list** with the column **coord_precision**.

Usage

```
coordinate_precision(x)

## Default S3 method:
coordinate_precision(x)

## S3 method for class 'c14_date_list'
coordinate_precision(x)
```

Arguments

`x`	an object of class c14_date_list

Value

an object of class c14_date_list with the additional column **coord_precision**

Examples

```
# calculate coordinate precision for all dates
ex <- coordinate_precision(example_c14_date_list)
ex[,c("lat", "lon", "coord_precision")]
```

`determine_country_by_coordinate`
Functions to improve the country attribution in a **c14_date_list**

Description

`c14bazAAR` provides several functions to check and improve the spatial attribution of the individual dates in a **c14_date_list** to a country.

`c14bazAAR::standardize_country_name()` adds column **country_thes** with standardized country names. Most source databases come with a column **country** that contains a character name of the origin country for each date. Unfortunately the different source databases don't rely on a unified naming convention and therefore use various terms to represent the same country (for example: United Kingdom, Great Britain, GB, etc.). This function aims to standardize the country naming scheme. To achieve this, it compares the names to values in an external (`countrycode::codelist`) and an internal country_thesaurus reference list. The latter needs manual curation to catch semantic and spelling errors in the source databases.

`c14bazAAR::determine_country_by_coordinate()` adds the column **country_coord** with standardized country attribution based on the coordinate information of the dates. Due to the inconsistencies in the **country** column in many c14 source databases it's often necessary to rely on the coordinate position (**lat** & **lon**) for reliable country attribution information.

`finalize_country_name()` picks the country name in a hierarchical order from the results of `c14bazAAR::determine_country_by_coordinate()` and `c14bazAAR::standardize_country_name()` functions, followed by the original input of the database. The result is added to the input date list with the column **country_final**.

`finalize_country_name()` also calls the other functions `c14bazAAR::determine_country_by_coordinate()` and `c14bazAAR::standardize_country_name()` if the necessary columns are missing yet.

Usage

```
determine_country_by_coordinate(x, suppress_spatial_warnings = TRUE)

## Default S3 method:
determine_country_by_coordinate(x, suppress_spatial_warnings = TRUE)

## S3 method for class 'c14_date_list'
determine_country_by_coordinate(x, suppress_spatial_warnings = TRUE)
```

```
finalize_country_name(x, quiet = FALSE)

## Default S3 method:
finalize_country_name(x, quiet = FALSE)

## S3 method for class 'c14_date_list'
finalize_country_name(x, quiet = FALSE)

standardize_country_name(
  x,
  country_thesaurus = get_country_thesaurus(),
  codesets = c("country.name.de", "iso3c"),
  quiet = FALSE,
  ...
)

## Default S3 method:
standardize_country_name(
  x,
  country_thesaurus = get_country_thesaurus(),
  codesets = c("country.name.de", "iso3c"),
  quiet = FALSE,
  ...
)

## S3 method for class 'c14_date_list'
standardize_country_name(
  x,
  country_thesaurus = get_country_thesaurus(),
  codesets = c("country.name.de", "iso3c"),
  quiet = FALSE,
  ...
)
```

Arguments

`x`	an object of class c14_date_list
`suppress_spatial_warnings`	suppress some spatial data messages and warnings
`quiet`	suppress suppress decision log output
`country_thesaurus`	data.frame with correct and variants of country names
`codesets`	which country codesets should be searched for in `countrycode::codelist` beyond **country.name.en**? See `?countrycode::codelist` for more information
`...`	additional arguments are passed to `stringdist::stringdist()`. `stringdist()` is used for fuzzy string matching of the country names in `countrycode::codelist`

Value

an object of class c14_date_list with the additional columns **country_thes**, **country_coord** and/or **country_final**

Examples

```
library(magrittr)
example_c14_date_list %>%
  determine_country_by_coordinate() %>%
  standardize_country_name() %>%
  finalize_country_name()
```

`duplicates` *Mark and remove duplicates in a* **c14_date_list**

Description

Duplicates are found in `c14bazAAR::mark_duplicates()` by comparison of **labnr**s. Only dates with exactly equal **labnr**s are considered duplicates. Duplicate groups are numbered (from 0) and these numbers linked to the individual dates in the new column **duplicate_group**. While `c14bazAAR::mark_duplicates()` only finds duplicates, `c14bazAAR::remove_duplicates()` removes them with three different strategies according to the value of the arguments `preferences` and `supermerge`:

1. Option 1: By merging all dates in a **duplicate_group**. All non-equal variables in the duplicate group are turned to `NA`. This is the default option.
2. Option 2: By selecting individual database entries in a **duplicate_group** according to a trust hierarchy as defined by the parameter `preferences`. In case of duplicates within one database the first occurrence in the table (top down) is selected. All databases not mentioned in `preferences` are dropped.
3. Option 3: Like option 2, but in this case the different datasets in a **duplicate_group** are merged column by column to create a superdataset with a maximum of information. The column **sourcedb** is dropped in this case to indicate that multiple databases have been merged. Data citation is a lot more difficult with this option. It can be activated with `supermerge`.

The option `log` allows to add a new column **duplicate_remove_log** that documents the variety of values provided by all databases for this duplicated date. `c14bazAAR::remove_duplicates()` needs the column **duplicate_group** and calls `c14bazAAR::mark_duplicates()` if it is missing.

Usage

```
mark_duplicates(x)

## Default S3 method:
mark_duplicates(x)
```

```
## S3 method for class 'c14_date_list'
mark_duplicates(x)

remove_duplicates(x, preferences = NULL, supermerge = FALSE, log = TRUE)

## Default S3 method:
remove_duplicates(x, preferences = NULL, supermerge = FALSE, log = TRUE)

## S3 method for class 'c14_date_list'
remove_duplicates(x, preferences = NULL, supermerge = FALSE, log = TRUE)
```

Arguments

`x`	an object of class c14_date_list
`preferences`	character vector with the order of source databases by which the deduping should be executed. If e.g. preferences = c("radon", "calpal") and a certain date appears in radon and euroevol, then only the radon entry remains. Default: NULL. With preferences = NULL all overlapping, conflicting information in individual columns of one duplicated date is removed. See Option 2 and 3.
`supermerge`	boolean. Should the duplicated datasets be merged on the column level? Default: FALSE. See Option 3.
`log`	logical. If log = TRUE, an additional column is added that contains a string documentation of all variants of the information for one date from all conflicting databases. Default = TRUE.

Value

an object of class c14_date_list with the additional columns **duplicate_group** or **duplicate_remove_log**

Examples

```
library(magrittr)

test_data <- tibble::tribble(
  ~sourcedb, ~labnr,  ~c14age, ~c14std,
 "A",       "lab-1", 1100,    10,
 "A",       "lab-1", 2100,    20,
 "B",       "lab-1", 3100,    30,
 "A",       "lab-2", NA,      10,
 "B",       "lab-2", 2200,    20,
 "C",       "lab-3", 1300,    10
) %>% as.c14_date_list()

# mark duplicates
test_data %>% mark_duplicates()

# remove duplicates with option 1:
test_data %>% remove_duplicates()

# remove duplicates with option 2:
```

```
test_data %>% remove_duplicates(
  preferences = c("A", "B")
)

# remove duplicates with option 3:
test_data %>% remove_duplicates(
  preferences = c("A", "B"),
  supermerge = TRUE
)
```

enforce_types — *Enforce variable types in a* **c14_date_list**

Description

Enforce variable types in a **c14_date_list** and remove everything that doesn't fit (e.g. text in a number field). See the variable_reference table for a documentation of the variable types. `enforce_types()` is called in `c14bazAAR::as.c14_date_list()`.

Usage

```
enforce_types(x, suppress_na_introduced_warnings = TRUE)

## Default S3 method:
enforce_types(x, suppress_na_introduced_warnings = TRUE)

## S3 method for class 'c14_date_list'
enforce_types(x, suppress_na_introduced_warnings = TRUE)
```

Arguments

`x`	an object of class c14_date_list
`suppress_na_introduced_warnings`	suppress warnings caused by data removal in type transformation due to wrong database entries (such as text in a number column)

Value

an object of class c14_date_list

Examples

```
# initial situation
ex <- example_c14_date_list
class(ex$c14age)

# modify variable/column type
ex$c14age <- as.character(ex$c14age)
```

```
class(ex$c14age)

# fix type with enforce_types()
ex <- enforce_types(ex)
class(ex$c14age)
```

`example_c14_date_list` *Example c14_date_list*

Description

c14_date_list with 1000 random dates for tests and example code.

Format

A c14_date_list. See variable_reference for an explanation of the variable meaning.

`fuse` *Fuse multiple* **c14_date_list***s*

Description

This function combines **c14_date_list**s with `dplyr::bind_rows()`.
This is not a joining operation and it therefore might introduce duplicates. See `c14bazAAR::mark_duplicates()` and `c14bazAAR::remove_duplicates()` for a way to find and remove them.

Usage

```
fuse(...)

## Default S3 method:
fuse(...)

## S3 method for class 'c14_date_list'
fuse(...)
```

Arguments

`...`	objects of class c14_date_list

Value

an object of class c14_date_list

Examples

```
# fuse three identical example c14_date_lists
fuse(example_c14_date_list, example_c14_date_list, example_c14_date_list)
```

---

get_c14data — *Download radiocarbon source databases and convert them to a* **c14_date_list**

Description

`get_c14data()` allows to download source databases and adjust their variables to conform to the definition in the variable_reference table. That includes renaming and arranging the variables (with `c14bazAAR::order_variables()`) as well as type conversion (with `c14bazAAR::enforce_types()`) – so all the steps undertaken by `as.c14_date_list()`.
All databases require different downloading and data wrangling steps. Therefore there's a custom getter function for each of them (see `?get_all_dates`).

`get_c14data()` is a wrapper to download all dates from multiple databases and `c14bazAAR::fuse()` the results.

Usage

```
get_c14data(databases = c())
```

Arguments

`databases`	Character vector. Names of databases to be downloaded. "all" causes the download of all databases. `get_c14data()` prints a list of the currently available databases

Examples

```

## Not run:
 get_c14data(databases = c("adrac", "palmisano"))
  get_all_dates()
## End(Not run)
```

get_country_thesaurus *get_country_thesaurus*

Description

Download thesaurus and provide it as tibble.

Usage

```
get_country_thesaurus(
  ref_url = paste(c("https://raw.githubusercontent.com", "ropensci", "c14bazAAR",
     "master", "data-raw", "country_thesaurus.csv"), collapse = "/")
)
```

Arguments

`ref_url`	url of the relevant reference table

get_dates *Backend functions for data download*

Description

Backend functions to download data. See ?`get_c14data` for a more simple interface and further information.

Usage

```
get_14sea(db_url = get_db_url("14sea"))

get_adrac(db_url = get_db_url("adrac"))

get_austarch(db_url = get_db_url("austarch"))

get_all_dates()

get_calpal(db_url = get_db_url("calpal"))

get_context(db_url = get_db_url("context"))

get_eubar(db_url = get_db_url("eubar"))

get_euroevol(db_url = get_db_url("euroevol"))

get_irdd(db_url = get_db_url("irdd"))
```

```
get_kiteeastafrica(db_url = get_db_url("kiteeastafrica"))

get_palmisano(db_url = get_db_url("palmisano"))

get_radon(db_url = get_db_url("radon"))

get_radonb(db_url = get_db_url("radonb"))
```

Arguments

`db_url`	Character. URL that points to the c14 archive file. `c14bazAAR::get_db_url()` fetches the URL from a reference list on github

---

`get_db_url` *get db url*

Description

Downloads information for c14 source databases from a reference table on github.

Usage

```
get_db_url(
  db_name,
  ref_url = paste(c("https://raw.githubusercontent.com", "ropensci", "c14bazAAR",
    "master", "data-raw", "url_reference.csv"), collapse = "/")
)
```

Arguments

`db_name`	name of the database
`ref_url`	url of the relevant reference table

`get_db_version` *get db version*

Description

Downloads information for c14 source databases from a reference table on github.

Usage

```
get_db_version(
  db_name,
  ref_url = paste(c("https://raw.githubusercontent.com", "ropensci", "c14bazAAR",
    "master", "data-raw", "url_reference.csv"), collapse = "/")
)
```

Arguments

`db_name`	name of the database
`ref_url`	url of the relevant reference table

`get_material_thesaurus` *get_material_thesaurus*

Description

Download thesaurus and provide it as tibble.

Usage

```
get_material_thesaurus(
  ref_url = paste(c("https://raw.githubusercontent.com", "ropensci", "c14bazAAR",
     "master", "data-raw", "material_thesaurus.csv"), collapse = "/")
)
```

Arguments

`ref_url`	url of the relevant reference table

`order_variables` *Order the variables in a* **c14_date_list**

Description

Arrange variables according to a defined order. This makes sure that a **c14_date_list** always appears with the same outline.
A **c14_date_list** has at least the columns **c14age** and **c14std**. Beyond that there's a selection of additional variables depending on the input from the source databases, as a result of the `c14bazAAR` functions or added by other data analysis steps. This function arranges the expected variables in a distinct, predefined order. Undefined variables are added at the end.

Usage

```
order_variables(x)

## Default S3 method:
order_variables(x)

## S3 method for class 'c14_date_list'
order_variables(x)
```

Arguments

`x`	an object of class c14_date_list

Value

an object of class c14_date_list

`write_c14` *write* **c14_date_list***s to files*

Description

write **c14_date_list**s to files

Usage

```
write_c14(x, format = c("csv"), ...)

## Default S3 method:
write_c14(x, format = c("csv"), ...)

## S3 method for class 'c14_date_list'
write_c14(x, format = c("csv"), ...)
```

Arguments

`x`	an object of class c14_date_list
`format`	the output format: 'csv' (default) or 'xlsx'. 'csv' calls `utils::write.csv()`, 'xlsx' calls `openxlsx::write.xlsx()`
`...`	passed to the actual writing functions

Examples

```
write_c14(
  example_c14_date_list,
  file = tempfile(),
  format = "csv"
)
```

Index