Package 'controller'

July 23, 2026

Title Controlled Vocabularies

Version 0.1.0

Description Work with controlled vocabularies; a predefined set of terms intended to be used in in a specific context. This package introduces the 'control' verb, which recodes values in a vector using a lookup table of preferred and variant terms (a 'thesaurus'). It also includes functions for reading controlled vocabularies from standard formats.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports curl, dplyr, fs, rlang, utils

Depends R (>= 4.1.0)

URL https://controller.joeroe.io, https://github.com/joeroe/controller

BugReports https://github.com/joeroe/controller/issues

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

VignetteBuilder knitr

NeedsCompilation no

Author Joe Roe [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1011-1244>)

Maintainer Joe Roe <joe@joeroe.io>

Repository CRAN

Date/Publication 2026-07-23 13:30:07 UTC

Contents

colour_thesaurus 2
control 2

control_matches 4
control_names 5
read_fish 6

Index **7**

`colour_thesaurus` *Colour thesaurus*

Description

A dataset of colour names based on Ingrid Sundberg's "colour thesaurus". Contains 240 names for shades of 12 colours, formatted for use as a thesaurus in `control()`.

Usage

```
colour_thesaurus
```

Format

A data frame with 240 rows and 2 variables:

colour preferred colour name: `"white"`, `"tan"`, `"yellow"`, `"orange"`, `"red"`, `"pink"`, `"purple"`, `"blue"`, `"green"`, `"brown"`, `"grey"`, or `"black"`

shade variant names for shades of `colour`

Source

Ingrid Sundberg (https://web.archive.org/web/20250619164626/https://ingridsundberg.com/2014/02/04/the-color-thesaurus/)

`control` *Recode values using a thesaurus*

Description

The `control()` verb replaces values in a vector with values looked up in a thesaurus. It is similar to `switch()` or `dplyr::recode()`, but the replacement values are specified as a data frame instead of as individual arguments.

By default `control()` replaces only values of `x` that exactly match terms in `thesaurus`. Additional arguments allow for case insensitive and fuzzy matching strategies (see details). `control_ci()` and `control_fuzzy()` are convenience aliases for case insensitive exact matching and full fuzzy matching respectively.

Usage

```
control(
  x,
  thesaurus,
  thesaurus_cols = c(1, 2),
  case_insensitive = FALSE,
  fuzzy_boundary = FALSE,
  fuzzy_encoding = FALSE,
  quiet = FALSE,
  warn_unmatched = TRUE
)

control_ci(x, thesaurus, thesaurus_cols = c(1, 2), ...)

control_fuzzy(x, thesaurus, thesaurus_cols = c(1, 2), ...)
```

Arguments

`x`	Vector to recode.
`thesaurus`	Data frame with a vector of preferred terms and a vector of variants.
`thesaurus_cols`	Vector of two column names or positions specifying which columns in `thesaurus` contain the preferred terms and variants respectively. Defaults to the first two columns.
`case_insensitive`	Set to `TRUE` to perform case insensitive matching.
`fuzzy_boundary`	Set to `TRUE` to perform fuzzy matching that ignores differences in the word boundaries used (e.g. `"foo bar"` matches `"foo-bar"`).
`fuzzy_encoding`	Set to `TRUE` to perform fuzzy matching that ignores non-ASCII characters that may have been encoded differently (e.g. `"foo"` matches `"foö"`).
`quiet`	Set to `TRUE` suppress messages about replaced values.
`warn_unmatched`	If `TRUE` (the default), issues a warning for values that couldn't be matched in `thesaurus`.
`...`	For `control_ci()` and `control_fuzzy()`, other arguments passed to `control()`. This includes fuzzy matching options (`fuzzy_boundary`, `fuzzy_encoding`) and output options (`quiet`, `warn_unmatched`).

Value

A vector the same length as `x` with values matching variants in `thesaurus` replaced with the preferred term. NAs in `x` are preserved as NAs.

By default gives a message listing replaced values and a warning listing any values not matched in the thesaurus. These can be suppressed with `quiet = TRUE` and `warn_unmatched = FALSE` respectively.

Examples

```
data(colour_thesaurus)

# Exact matching
x <- c("red", "lipstick", "green", "mint", "blue", "azure")
control(x, colour_thesaurus)

# Case insensitive matching
x <- toupper(x)
control_ci(x, colour_thesaurus)

# control_matches() returns a data frame showing which match type was used:
control_matches(x, colour_thesaurus, case_insensitive = TRUE)
```

`control_matches` *Show detailed match results from a thesaurus*

Description

`control_matches()` returns a data frame showing which type of match was used for each value in `x`. This is useful for debugging or inspecting how `control()` recodes values.

Usage

```
control_matches(
  x,
  thesaurus,
  thesaurus_cols = c(1, 2),
  case_insensitive = FALSE,
  fuzzy_boundary = FALSE,
  fuzzy_encoding = FALSE
)
```

Arguments

`x`	Vector to recode.
`thesaurus`	Data frame with a vector of preferred terms and a vector of variants.
`thesaurus_cols`	Vector of two column names or positions specifying which columns in `thesaurus` contain the preferred terms and variants respectively. Defaults to the first two columns.
`case_insensitive`	Set to `TRUE` to perform case insensitive matching.
`fuzzy_boundary`	Set to `TRUE` to perform fuzzy matching that ignores differences in the word boundaries used (e.g. `"foo bar"` matches `"foo-bar"`).
`fuzzy_encoding`	Set to `TRUE` to perform fuzzy matching that ignores non-ASCII characters that may have been encoded differently (e.g. `"foo"` matches `"foö"`).

Value

A data frame with the same number of rows as `x`. The first column (`term`) contains the original values. Subsequent columns contain the match result for each match type (e.g. `exact_match`, `case_insensitive_match`, `fuzzy_boundary_match`, `fuzzy_encoding_match`). Rows for NA values in `x` are all NAs.

Examples

```
data(colour_thesaurus)

x <- c("red", "lipstick", "green", "mint", "blue", "azure")
control_matches(x, colour_thesaurus)
```

`control_names` *Control the names of an object*

Description

`control_names()` controls the names of an object using a thesaurus. It extracts the names of `x`, passes them to `control()`, and reassigns the result to `names(x)`.

`control_names_ci()` and `control_names_fuzzy()` are convenience aliases for case insensitive and full fuzzy matching respectively.

Usage

```
control_names(x, thesaurus, thesaurus_cols = c(1, 2), ...)

control_names_ci(x, thesaurus, thesaurus_cols = c(1, 2), ...)

control_names_fuzzy(x, thesaurus, thesaurus_cols = c(1, 2), ...)
```

Arguments

`x`	Object with names to control.
`thesaurus`	Data frame with a vector of preferred terms and a vector of variants.
`thesaurus_cols`	Vector of two column names or positions specifying which columns in `thesaurus` contain the preferred terms and variants respectively. Defaults to the first two columns.
`...`	Other arguments passed to `control()`. This includes fuzzy matching options (`case_insensitive`, `fuzzy_boundary`, `fuzzy_encoding`) and output options (`quiet`, `warn_unmatched`).

Value

The object `x` with its names replaced by controlled values.

Examples

```
df <- data.frame(temp = 20, humid = 65, `wind speed` = 10, date = "2024-01-01")
thesaurus <- data.frame(
  preferred = c("temperature", "humidity", "wind_speed"),
  variant = c("temp", "humid", "wind speed")
)
control_names(df, thesaurus)
```

`read_fish` *Read a FISH controlled vocabulary*

Description

Reads controlled vocabularies from Historic England's FISH (Forum on Information Standards in Heritage) group. These can be downloaded from: https://heritage-standards.org.uk/fish-vocabularies/.

Usage

```
read_fish(path)
```

Arguments

`path`	Path or URL to a vocabulary in FISH's CSV format. Can be either a `.zip` archive or a directory containing already uncompressed files, or a URL.

Value

A data frame with two columns: `preferred` and `term`.

References

- Forum on Information Standards in Heritage, "FISH Thesaurus Table Structure", Available from: https://heritage-standards.org.uk/fish-vocabularies/.

Examples

```
# Read a FISH vocabulary from a local zip file
nationality_zip <- system.file("extdata", "fish-nationality.zip",
                               package = "controller")
read_fish(nationality_zip)
```

Index

∗ **datasets**
colour_thesaurus, 2

colour_thesaurus, 2
control, 2
control(), 2, *5*
control_ci *(*control*)*, 2
control_fuzzy *(*control*)*, 2
control_matches, 4
control_names, 5
control_names_ci *(*control_names*)*, 5
control_names_fuzzy *(*control_names*)*, 5

dplyr::recode(), *2*

read_fish, 6

switch(), *2*