Package ‘unjoin’
October 12, 2022

Title Separate a Data Frame by Normalization

Version 0.1.0

Description Separate a data frame in two based on key columns. The function
unjoin() provides an inside-out version of a nested data frame. This is used
to identify duplication and normalize it (in the database sense) by linking
two tables with the redundancy removed. This is a basic requirement for
detecting topology within spatial structures that has motivated the need for
this package as a building block for workflows within more applied projects.

Depends R (>= 3.3.2)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports dplyr (>= 0.5.0), rlang, tibble

Suggests gapminder, tidyr, testthat, covr, spelling

URL https://github.com/hypertidy/unjoin

BugReports https://github.com/hypertidy/unjoin/issues

Language en-US

NeedsCompilation no

Author Michael D. Sumner [aut, cre],
Simon Wotherspoon [ctb],
Hadley Wickham [ctb] (named the concept, provided excellent guidance
via tidyr source code)

Maintainer Michael D. Sumner <mdsumner@gmail.com>

Repository CRAN

Date/Publication 2020-05-13 05:20:02 UTC

R topics documented:

unjoin ................................................................. 2
Description
Split a table in two and remove repeated values.

Usage
unjoin(data, ..., key_col = "idx0")

## S3 method for class 'data.frame'
unjoin(data, ..., key_col = ".idx0")

## S3 method for class 'unjoin'
unjoin(data, ..., key_col = ".idx0")

Arguments
data A data frame.
... Specification of columns to unjoin by. For full details, see the 'dplyr::select' documentation.
key_col The name of the new column to key the two output data frames.

Details
The data frame on input is treated as "data", the new data frame is treated as the normalized key. This means that the split-off and de-duplicated table has the name given via the 'key_col' argument (defaults to ".idx0") and shares this name with the common key.

It's not yet clear if this flexibility around naming is a good idea, but it enables a simple scheme for chaining unjoins, though you'd better not use the same 'key_col' again.

This is a subset of the tasks done by nest.

See Also
'dplyr::inner_join' for the inverse operation.
'tidyr::nest' for the complementary operation resulting in one nested data frame

Examples
library(dplyr)
data("Seatbelts", package= "datasets")
x <- unjoin(as.data.frame(Seatbelts), front, law)
y <- inner_join(x$.idx0, x$data) %>% select(-.idx0)
all.equal(y[colnames(Seatbelts)], as.data.frame(Seatbelts))
iris %>% unjoin(-Species)
chickwts %>% unjoin(weight)

if (require("gapminder")) {
  gapminder %>%
    group_by(country, continent) %>%
    unjoin()

  gapminder %>%
    unjoin(-country, -continent)
  unjoin(gapminder)
}

unjoin(iris, Petal.Width) %>% unjoin(Species, key_col = ".idx1")
Index

nest, 2
unjoin, 2