Package ‘daff’

October 13, 2022

Version 0.3.5

Title Diff, Patch and Merge for Data.frames

Description Diff, patch and merge for data frames. Document changes in data
sets and use them to apply patches. Changes to data can be made visible by using
render_diff. The V8 package is used to wrap the ‘daff.js’ JavaScript library
which is included in the package.

License MIT + file LICENSE

LazyData true

Imports V8 (>= 0.6), jsonlite, utils

URL http://github.com/edwindj/daff

Suggests testthat

RoxygenNote 6.1.1

NeedsCompilation no

Author Paul Fitzpatrick [aut] (JavaScript original,
http://paulfitz.github.io/daff/),
Edwin de Jonge [aut, cre] (R wrapper,
<https://orcid.org/0000-0002-6580-4718>),
Gregory R. Warnes [aut]

Maintainer Edwin de Jonge <edwindjonge@gmail.com>

Repository CRAN

Date/Publication 2019-05-15 09:20:03 UTC

R topics documented:

daff ................................................................. 2
differs_from .................................................. 2
diff_data ...................................................... 3
merge_data .................................................... 5
patch_data .................................................... 6
render_diff ................................................... 7
which_conflicts ............................................. 8
write_diff ................................................... 9
Description

Daff calculates differences between two data.frames. This difference can be stored and later used to patch the original data. Differences can also be made visual by using `render_diff` showing what changed.

Details

Storing the difference between data sets allows for tracking or incorporating manual changes to data sets. Ideally changes to data should be scripted to be reproducible, but there are situations or scenario’s where this is not possible or happens out of your control. daff can help track these changes.

actions

- **diff_data**: Find differences in values between data.frames
- **patch_data**: Apply a patch generated with `diff_data` to a data.frame
- **merge_data**: Merge two diverged data.frames originating from a same parent

daff.js

Daff wraps the daff.js library which offers more functionality.

differs_from

Description

This is the same function as `diff_data` but with arguments reversed. This is more useful when using dplyr and magrittr.

Usage

```r
differs_from(data, data_ref, ...)
```

Arguments

- **data**: data.frame to check for changes
- **data_ref**: data.frame reference data frame
- **...**: not further specified
Value

difference object

See Also

diff_data

---

diff_data  Do a data diff

Description

Find differences with a reference data set. The diff can be used to patch_data, to store the difference for documentation purposes using write_diff or to visualize the difference using render_diff

Usage

diff_data(data_ref, data, always_show_header = TRUE,
always_show_order = FALSE, columns_to_ignore = c(),
count_like_a_spreadsheet = TRUE, ids = c(),
ignore_whitespace = FALSE, never_show_order = FALSE,
ordered = TRUE, padding_strategy = c("auto", "smart", "dense",
"sparse"), show_meta = TRUE, show_unchanged = FALSE,
show_unchanged_columns = FALSE, show_unchanged_meta = FALSE,
unchanged_column_context = 1L, unchanged_context = 1L)

Arguments

data_ref data.frame reference data frame
data data.frame to check for changes
always_show_header
  logical Should we always give a table header in diffs? This defaults to TRUE,
  and - frankly - you should leave it at TRUE for now.
always_show_order
  logical Diffs for tables where row/column order has been permuted may include
  an extra row/column specifying the changes in row/column numbers. If
  you’d like that extra row/column to always be included, turn on this flag, and
  turn off never_show_order.
columns_to_ignore
  character List of columns to ignore in all calculations. Changes related to
  these columns should be discounted.
count_like_a_spreadsheet
  logical Should column numbers, if present, be rendered spreadsheet-style as
  A,B,C,...,AA,BB,CC? Defaults to TRUE.
ids
  character List of columns that make up a primary key, if known. Otherwise
  heuristics are used to find a decent key (or a set of decent keys).
ignore_whitespace
   logical  Should whitespace be omitted from comparisons. Defaults to FALSE.
never_show_order
   logical  Diffs for tables where row/column order has been permuted may include an extra row/column specifying the changes in row/column numbers. If you’d like to be sure that that row/column is never included, turn on this flag, and turn off always_show_order.
ordered
   logical  Is the order of rows and columns meaningful? Defaults to ‘TRUE’.
padding_strategy
   logical  Strategy to use when padding columns. Valid values are "auto", "smart", "dense", and "sparse". Leave null for a sensible default.
show_meta
   logical  Show changes in column properties, not just data, if available. Defaults to TRUE.
show_unchanged
   logical  Should we show all rows in diffs? We default to showing just rows that have changes (and some context rows around them, if row order is meaningful), but you can override this here.
show_unchanged_columns
   logical  Should we show all columns in diffs? We default to showing just columns that have changes (and some context columns around them, if column order is meaningful), but you can override this here. Irrespective of this flag, you can rely on index/key columns needed to identify rows to be included in the diff.
show_unchanged_meta
   logical  Show all column properties, if available, even if unchanged. Defaults to FALSE.
unchanged_column_context
   integer  When showing context columns around a changed column, what is the minimum number of such columns we should show?
unchanged_context
   integer  When showing context rows around a changed row, what is the minimum number of such rows we should show?

Value
difference object

See Also
differs_from

Examples
library(daff)
x <- iris
x[1,1] <- 10
diff_data(x, iris)

dd <- diff_data(x, iris)
merge_data

```r
#write_diff(dd, "diff.csv")
summary(dd)
```

---

merge_data

Merge two tables based on a parent version

Description

merge_data provides a three-way merge: suppose two versions are based on a common version, this function will merge tables a and b.

Usage

```r
merge_data(parent, a, b)
```

Arguments

- `parent` data.frame
- `a` data.frame changed version of parent
- `b` data.frame other changed version of parent

Details

If both a and b change the same table cell with a different value, this results in a conflict. In that case a warning will be generated with the number of conflicts. In the returned data.frame of a conflicting merge columns with conflicting values are of type character and contain all three values coded as

```
(parent) a /// b
```

Value

merged data.frame. When a merge has conflicts the columns of conflicting changes are of type character and contain all three values.

See Also

`which_conflicts`

Examples

```r
parent <- a <- b <- iris[1:3,]
a[1,1] <- 10
b[2,1] <- 11
# succesful merge
merge_data(parent, a, b)

parent <- a <- b <- iris[1:3,]
a[1,1] <- 10
```
b[1,1] <- 11
# conflicting merge (both a and b change same cell)
merged <- merge_data(parent, a, b)
merged # note the conflict

# find out which rows contain a conflict
which_conflicts(merged)

patch_data

Description

Patch data with a diff generated by diff_data

Usage

patch_data(data, patch)

Arguments

data data.frame that should be patched
patch generated with diff_data

Value

data.frame that has been patched.

Examples

library(daff)
x <- iris
# change a value
x[1,1] <- 1000

patch <- diff_data(iris, x)
print(patch)
# apply patch
iris_patched <- patch_data(iris, patch)

iris_patched[1,1] == 1000
render_diff

**Render a data_diff to html**

**Description**

Converts a diff_data object to HTML code, and opens the resulting HTML code in a browser window if `view==TRUE` and R is running interactively.

**Usage**

```r
render_diff(diff, file = tempfile(fileext = "html"),
view = interactive(), fragment = FALSE, pretty = TRUE, title,
summary = !fragment, use.DataTables = !fragment)
```

**Arguments**

- `diff`: diff_data object generated with `diff_data`
- `file`: character target file (optional)
- `view`: logical Open the generated HTML in a browser if R is being used interactively
- `fragment`: logical If TRUE generate (just) an HTML table, otherwise generate a valid HTML document.
- `pretty`: logical Use HTML arrow characters instead of '–>'.
- `title`: character title text. Defaults to the quoted names of the data objects compared, separated by 'vs.'
- `summary`: logical Should a summary of changes be shown above the HTML table?
- `use.DataTables`: logical Include jQuery DataTables plugin and enable: - pagination (10,25,50,100,All) - searching - filtering - column visibility (individually enable/disable) - copy/csv/excel/pdf export buttons - column reorder (drag and drop) - row reorder (drag and drop) - row/multirow select

**Value**

generated html

**See Also**

data_diff

**Examples**

```r
y <- iris[1:3,]
x <- y

x <- head(x,2) # remove a row
x[1,1] <- 10 # change a value
x$hello <- "world" # add a column
```
which_conflicts <- function(merged)
{
  which_conflicts <- which(merged)  # return which rows of a merged data.frame contain conflicts
  return(which_conflicts)
}

Description

return which rows of a merged data.frame contain conflicts.

Usage

which_conflicts(merged)

Arguments

merged data.frame merged data.frame with possible conflicts.

Value

integer vector with row positions containing conflicts.

See Also
merge_data

Examples

parent <- a <- b <- iris[1:3,]
a[1,1] <- 10
b[2,1] <- 11
# succesful merge
merge_data(parent, a, b)

parent <- a <- b <- iris[1:3,]
a[1,1] <- 10
b[1,1] <- 11
# conflicting merge (both a and b change same cell)
merged <- merge_data(parent, a, b)
merged #note the conflict

#find out which rows contain a conflict
which_conflicts(merged)
**write_diff**

Write or read a diff to or from a file

**Description**

The diff information is stored in the Coopy highlighter diff format: [https://paulfitz.github.io/daff-doc/spec.html](https://paulfitz.github.io/daff-doc/spec.html)

**Usage**

```r
write_diff(diff, file = "diff.csv")
read_diff(file)
```

**Arguments**

- `diff`: generated with `diff_data`
- `file`: filename or connection

**Details**

Note that type information of the target data.frame is lost when writing a patch to disk. Using a stored diff to patch a data.frame will use the column types of the source data.frame to determine the target column types. New introduced columns may become characters.

Names of the reference and comparison dataset are also lost when writing a data_diff object to disk.

**Value**

`diff` object that can be used in `patch_data`
Index

daff, 2
 daff-package (daff), 2
 diff_data, 2, 3, 6, 7
 differs_from, 2

merge_data, 2, 5, 8

patch_data, 2, 3, 6, 9

read_diff (write_diff), 9
 render_diff, 3, 7

which_conflicts, 5, 8
 write_diff, 3, 9