Package ‘waldo’

July 13, 2020

Title Find Differences Between R Objects

Version 0.2.0

Description Compare complex R objects and reveal the key differences. Designed particularly for use in testing packages where being able to quickly isolate key differences makes understanding test failures much easier.

License MIT + file LICENSE

URL https://github.com/r-lib/waldo

BugReports https://github.com/r-lib/waldo/issues

Imports cli, diffobj, fansi, glue, methods, rematch2, rlang, tibble

Suggests testthat, covr, R6

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Hadley Wickham [aut, cre],
RStudio [cph]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2020-07-13 04:10:02 UTC

R topics documented:

`compare` *Compare two objects*

Description

This compares two R objects, identifying the key differences. It:

- Orders the differences from most important to least important.
- Displays the values of atomic vectors that are actually different.
- Carefully uses colour to emphasise changes (while still being readable when colour isn't available).
- Uses R code (not a text description) to show where differences arise.
- Where possible, it compares elements by name, rather than by position.
- Errs on the side of producing too much output, rather than too little.

`compare()` is an alternative to `all.equal()`.

Usage

```
compare(
  x,
  y,
  ...,
  x_arg = "x",
  y_arg = "y",
  tolerance = NULL,
  ignore_srcref = TRUE,
  ignore_attr = FALSE,
  ignore_encoding = TRUE,
  ignore_function_env = FALSE,
  ignore_formula_env = FALSE
)
```

Arguments

`x, y`	Objects to compare. `y` is treated as the reference object so messages describe how `x` is different to `y`
`...`	A handful of other arguments are supported with a warning for backward compatability. These include: • `all.equal()` arguments `checkNames` and `check.attributes` • `testthat::compare()` argument `tol` All other arguments are ignored with a warning.
`x_arg, y_arg`	Name of `x` and `y` arguments, used when generated paths to internal components.

`tolerance`
: If non-`NULL`, used as threshold for ignoring small floating point difference when comparing numeric vectors. Setting to any non-`NULL` value will cause integer and double vectors to be compared based on their values, rather than their types.

  It uses the same algorithm as `all.equal()`, i.e., first we generate `x_diff` and `y_diff` by subsetting `x` and `y` to look only locations with differences. Then we check that `mean(abs(x_diff -y_diff)) / mean(abs(y_diff))` (or just `mean(abs(x_diff -y_diff))` if `y_diff` is small) is less than `tolerance`.

`ignore_srcref`
: Ignore differences in function `srcrefs`? `TRUE` by default since the `srcref` does not change the behaviour of a function, only its printed representation.

`ignore_attr`
: Ignore all differences in attributes? Only provided for backward compatibility with `all.equal()`. Using `TRUE` is not generally recommended because it will ignore many important functional differences.

`ignore_encoding`
: Ignore string encoding? `TRUE` by default, because this is R's default behaviour. Use `FALSE` when specifically concerned with the encoding, not just the value of the string.

`ignore_function_env, ignore_formula_env`
: Ignore the environments of functions and formulas, respectively? These are provided primarily for backward compatibility with `all.equal()` which always ignores these environments.

Value

A character vector with class "waldo_compare". If there are no differences it will have length 0; otherwise each element is contains the description of a single difference.

Examples

```
# Thanks to diffobj package comparison of atomic vectors shows differences
# with a little context
compare(letters, c("z", letters[-26]))
compare(c(1, 2, 3), c(1, 3))
compare(c(1, 2, 3), c(1, 3, 4, 5))
compare(c(1, 2, 3), c(1, 2, 5))

# More complex objects are traversed, stopping only when the types are
# different
compare(
  list(x = list(y = list(structure(1, z = 2)))),
  list(x = list(y = list(structure(1, z = "a"))))
)

# Where possible, recursive structures are compared by name
compare(iris, rev(iris))

compare(list(x = "x", y = "y"), list(y = "y", x = "x"))
# Otherwise they're compared by position
compare(list("x", "y"), list("x", "z"))
compare(list(x = "x", x = "y"), list(x = "x", y = "z"))
```

`compare_proxy` *Proxy for waldo comparison*

Description

Use this generic to override waldo's default comparison if you need to override the defaults (typically because your object contains an external pointer).

waldo comes with methods for two common cases:

- data.table: the `.internal.selfref` attribute is set to `NULL`. This is an external pointer that is used for performance optimisation, and doesn't affect the data.
- `xml2::xml_node`: the underlying XML data is stored in memory in C, behind an external pointer, so the we best can do is to convert the object to a string.

Usage

```
compare_proxy(x)
```

Arguments

`x`	An object.

Index