Package 'compare'

May 8, 2026

Version 0.2-6

Title Comparing Objects for Differences

Author Paul Murrell

Maintainer Paul Murrell <p.murrell@auckland.ac.nz>

Description Functions to compare a model object to a comparison object. If the objects are not identical, the functions can be instructed to explore various modifications of the objects (e.g., sorting rows, dropping names) to see if the modified versions are identical.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-08-25 08:54:01

Contents

`commentQuestions` *Apply a Commenting Scheme*

Description

This function applies a commenting scheme to a set of comparisons to produce a set of comments.

Usage

```
commentQuestions(result, ...)
```

Arguments

`result`	A set of comparison results, as generated by the `compareFiles()` function.
`...`	One or more commenting schemes, as generated by the `questionComments()` function.

Value

A matrix of comments.

Author(s)

Paul Murrell

See Also

`questionComments` and `compareFiles`

Examples

```
modelNames <- c("id", "age",
                "edu", "class",
                "IndianMothers")
files <- list.files(system.file("example", package="compare"),
                    pattern="^student[0-9]+[.]R$",
                    full.names=TRUE)

results <- compareFiles(files,
                        modelNames,
                        system.file("example", "model.R", package="compare"),
                        allowAll=TRUE,
                        resultNames=gsub(".+[/]|[.]R", "", files))
q1comments <- questionComments(c("id", "age", "edu", "class"),
                               comments("class",
                                        transformComment("coerced",
                                                         "'class' is a factor!")))
commentQuestions(results, q1comments)
```

compare *Compare Two Objects*

Description

Compare two objects and, if they are not the same, attempt to transform them to see if they are the same after being transformed.

Usage

```
compare(model, comparison,
        equal = TRUE,
        coerce = allowAll,
        shorten = allowAll,
        ignoreOrder = allowAll,
        ignoreNameCase = allowAll,
        ignoreNames = allowAll,
        ignoreAttrs = allowAll,
        round = FALSE,
        ignoreCase = allowAll,
        trim = allowAll,
        dropLevels = allowAll,
        ignoreLevelOrder = allowAll,
        ignoreDimOrder = allowAll,
        ignoreColOrder = allowAll,
        ignoreComponentOrder = allowAll,
        colsOnly = !allowAll,
        allowAll = FALSE)

compareName(model, compName,
            ...,
            ignore.case = TRUE,
            compEnv = .GlobalEnv)
```

Arguments

model	The "correct" object.
comparison	The object to be compared with the model.
equal	Test for equality if test for identity fails.
coerce	If objects are not the same, allow coercion of comparsion to model class.
shorten	If the length of one object is less than the other, shorten the longer object.
ignoreOrder	Ignore the order of values when comparing.
ignoreNameCase	Ignore the case of names when comparing.
ignoreNames	Ignore names attributes altogether.
ignoreAttrs	Ignore attributes altogether.

round	If objects are not the same, allow numbers to be rounded.
ignoreCase	Ignore the case of string values.
trim	Ignore leading and trailing spaces in string values.
dropLevels	If factors are not the same, allow unused levels to be dropped.
ignoreLevelOrder	Ignore the order of factor levels.
ignoreDimOrder	Ignore the order of dimensions when comparing matrices, arrays, or tables.
ignoreColOrder	Ignore the order of columns when comparing data frames.
ignoreComponentOrder	Ignore the order of components when comparing lists.
colsOnly	Only transform columns (not rows) when comparing data frames.
allowAll	Allow any sort of transformation (almost; see Details).
compName	A string giving the *name* of the comparison object.
...	Arguments to be passed to `compare()`.
ignore.case	Ignore the case of the name when searching for the comparison object.
compEnv	An environment within which to search for the comparison object.

Details

The `compare()` function compares two objects for equality. The arguments allow for various transformations of the objects (e.g., type coercion) in order to try and achieve equality.

Specific transformations can be turned on via the appropriate argument, or the `allowAll` argument can be used to enable all transformations, with the exception of rounding of numeric values.

The `compareName()` function is a wrapper for `compare()` that requires the *name* of the comparison object rather than the objects itself, plus it allows an environment to be supplied that contains the actual comparison object. This is useful for the situation where a particular object with a particular name should have been generated and allows the generated object to differ from the desired object in terms of the case of its name.

Value

An object of class `"comparison"`.

This is a list. The most important components are `result`, which gives the overall success/failure of the comparison, and `transform`, which describes the transformations attempted during the comparison (whether they were successful or not).

Author(s)

Paul Murrell

See Also

`comparison-class` and `compareEqual`

Examples

```
obj1 <- c("a", "a", "b", "c")
obj2 <- factor(obj1)
compare(obj1, obj2, allowAll=TRUE)
```

`compareCoerce` *Compare Two Objects of Different Class*

Description

Compare two objects for equality, coercing the comparison object to the same class as the model object if necessary beforehand.

Usage

```
compareCoerce(model, comparison, transform = character(),
              equal = TRUE, ...)

## S3 method for class 'data.frame'
compareCoerce(model, comparison,
              transform=character(),
              equal=TRUE,
              ignoreColOrder=FALSE,
              ignoreNameCase=FALSE,
              ...)
```

Arguments

`model`	The "correct" object.
`comparison`	The object to be compared with the `model`.
`transform`	A character vector containing any transformations that have been performed on the objects prior to this comparison.
`equal`	Whether to test for equality if the test for identity fails.
`ignoreColOrder`	For data frames, whether to reorder columns by name first.
`ignoreNameCase`	For data frames and lists, whether to ignore the case of names when reordering components by name.
`...`	Arguments passed to `compareEqual()`.

Details

This function is generic, with methods for logical, integer, numeric, and character vectors, factors, arrays, matrices, tables, data frames, and lists.

The integer and numeric methods use the appropriate special-case coercion for factors.

Value

An object of class "comparison". Use isTRUE() to determine whether the comparison has succeeded.

Author(s)

Paul Murrell

See Also

compare and compareEqual

Examples

```
compareCoerce(letters, factor(letters))
```

---

compareEqual *Compare Two Objects for Equality*

Description

Compare two objects and allow for various minor differences between them.

Usage

```
compareEqual(model, comparison, transform = character(), ...)

## S3 method for class 'logical'
compareEqual(model, comparison, transform = character(), ...)

## S3 method for class 'numeric'
compareEqual(model, comparison, transform = character(),
             round=FALSE,
             ...)

## S3 method for class 'character'
compareEqual(model, comparison, transform=character(),
             ignoreCase=FALSE,
             trim=FALSE,
             ...)

## S3 method for class 'factor'
compareEqual(model, comparison, transform=character(),
             dropLevels=FALSE,
             ignoreLevelOrder=FALSE,
             ...)
```

```
## S3 method for class 'matrix'
compareEqual(model, comparison, transform=character(),
             ignoreDimOrder=FALSE,
             ...)

## S3 method for class 'array'
compareEqual(model, comparison, transform=character(),
             ignoreDimOrder=FALSE,
             ...)

## S3 method for class 'table'
compareEqual(model, comparison, transform=character(),
             ignoreDimOrder=FALSE,
             ...)

## S3 method for class 'data.frame'
compareEqual(model, comparison, transform=character(),
             ignoreColOrder=FALSE,
             ignoreNameCase=FALSE,
             ...,
             recurseFun=compareEqual)

## S3 method for class 'list'
compareEqual(model, comparison, transform=character(),
             ignoreComponentOrder=FALSE,
             ignoreNameCase=FALSE,
             ...,
             recurseFun=compareEqual)
```

Arguments

`model`	The "correct" object.
`comparison`	The object to be compared with the `model`.
`transform`	A character vector containing any transformations that have been performed on the objects prior to this comparison.
`round`	For numeric vectors, whether the objects should be rounded before comparison. If `FALSE` there is no rounding, if `TRUE` both objects are rounded to zero decimal places, if an integer value, then both objects are rounded to the specified number of decimal places. May also be a function (of one argument).
`ignoreCase`	For character vectors, whether to ignore the case of the strings.
`trim`	For character vectors, whether to ignore leading and trailing white space.
`dropLevels`	For factors, whether to drop unused levels before the comparison.
`ignoreLevelOrder`	For factors, whether to ignore the order of levels.
`ignoreDimOrder`	For matrices, arrays, and tables, whether to reorder of the dimensions by name before the comparison.

`ignoreColOrder` For data frames, whether to reorder the columns by name before the comparison.

`ignoreComponentOrder`
For lists, whether to reorder the components by name before the comparison.

`ignoreNameCase` For data frames and lists, whether to ignore the case of names when reordering components by name.

`...` Other arguments controlling the comparison.

`recurseFun` For data frames, the function to use to compare the columns of the data frames. For lists, the function used to compare the components of the list.

Details

This function compares two objects for identity (using `identical()`), then if that fails and `equal=TRUE`, compares the objects for equality. The arguments allow for various relaxations on what "equal" means.

For numeric vectors, the comparison uses `all.equal()` to allow for differences in floating-point representation. The `round` argument also allows for much more lenient comparisons. The `round` argument can also be a function, e.g., `floor()` (see other examples below).

Value

An object of class `"comparison"`. Use `isTRUE()` to determine whether the comparison has succeeded.

Author(s)

Paul Murrell

See Also

`compare`

Examples

```
compareEqual(letters, paste(" ", letters, " "), trim=TRUE)
compareEqual(c(.1, 1, 10), c(.13, 1.3, 13),
             round=function(x) { signif(x, 1) })
```

`compareFile` *Compare Several Objects*

Description

Generate objects by running code from one or more files then compare the resulting objects to a set of model objects, allowing for and reporting on minor differences.

Usage

```
compareFile(filename, modelNames,
            modelCode = NULL, modelSave = NULL, modelAnswers = NULL,
            round = FALSE, ...)

compareFiles(filenames, modelNames,
             modelCode=NULL, modelSave=NULL,
             resultNames=filenames, ...)
```

Arguments

filename	The name of a file containing R code.
filenames	A vector of names of files containing R code.
modelNames	A vector of names of objects of interest that should be generated by the code in the file(s).
modelCode	The name of a file that contains R code to generate model objects.
modelSave	The name of a binary file that can be used to load model objects.
modelAnswers	A list containing model objects.
round	A logical indicating whether to round, or an integer indicating how many digits to round to, or a function of one argument (that performs something like rounding), or a named list of any of those.
resultNames	A vector of names to be used to identify the results from different files.
...	For compareFile(), arguments to the compare() function; for compareFiles(), arguments to the compareFile() function.

Details

The compareFile() function is useful for comparing several pairs of objects at once, where the comparison objects are generated by running R code from a file and the model objects can be generated in a number of ways.

The compareFiles() function extends this to comparing the same set of model objects to several different sets of comparison objects (where each set of comparison objects is generated from a separate file of R code).

For generating model objects, if a binary file is specified, that is used in preference to source code; if a list of objects is provided, that is used; and if no model objects are specified, they are taken from the global workspace (using modelNames).

The round argument may be a named list so that different rounding is applied to different comparisons.

Value

compareFile() returns an object of class "comparisonList", which is just a list of "comparison" objects.

compareFiles() returns an object of class "comparisonListList". This is just a list of "comparisonList" objects, but a print method is defined to make the display of this object a little more sane.

Author(s)

Paul Murrell

See Also

compare

Examples

```
modelNames <- c("id", "age",
                "edu", "class",
                "IndianMothers")
compareFile(system.file("example", "student1.R", package="compare"),
            modelNames,
            system.file("example", "model.R", package="compare"))
```

---

compareIdentical *Compare Two Objects for Identity*

Description

This function is just a wrapper for the `identical()` function to determine whether two objects are identical. Its usefulness is in being able to be combined with other functions in the **compare** package that perform much more relaxed comparisons.

Usage

```
compareIdentical(model, comparison, transform = character(), ...)
```

Arguments

`model`	The "correct" object.
`comparison`	The object to be compared with the `model`.
`transform`	A character vector containing any transformations that have been performed on the objects prior to this comparison.
`...`	Allows arguments that are only relevant to other comparison functions.

Value

An object of class `"comparison"`. Use `isTRUE()` to determine whether the comparison has succeeded.

Author(s)

Paul Murrell

See Also

compare and compareEqual

Examples

```
compareIdentical(1:10/10, 1:10/10)
compareIdentical(1:10/10, 3:12/10 - 2/10)
```

---

compareIgnoreAttrs *Compare Two Objects with Different Attributes*

Description

Compare two objects for equality, ignoring any attributes if necessary beforehand.

Usage

```
compareIgnoreAttrs(model, comparison, transform = character(),
                   equal = TRUE, ...)
```

Arguments

model	The "correct" object.
comparison	The object to be compared with the model.
transform	A character vector containing any transformations that have been performed on the objects prior to this comparison.
equal	Whether to test for equality if the test for identity fails.
...	Arguments passed to compareEqual().

Value

An object of class "comparison". Use isTRUE() to determine whether the comparison has succeeded.

Author(s)

Paul Murrell

See Also

compare and compareEqual

Examples

```
model <- list(a=1:26, b=letters)
comparison <- model
attr(comparison, "test") <- "test"
compareIgnoreAttrs(model, comparison)
```

compareIgnoreNameCase *Compare Two Objects with Different Names*

Description

Compare two objects for equality, ignoring the case of name attributes, or ignoring name attributes altogether, if necessary beforehand.

Usage

```
compareIgnoreNameCase(model, comparison, transform = character(),
                      equal = TRUE, ...)

## S3 method for class 'data.frame'
compareIgnoreNameCase(model, comparison, transform=character(),
                      equal=TRUE,
                      colsOnly=TRUE,
                      ignoreColOrder=FALSE,
                      ignoreNameCase=FALSE,
                      ...)

compareIgnoreNames(model, comparison, transform=character(),
                   equal=TRUE, ...)

## S3 method for class 'data.frame'
compareIgnoreNames(model, comparison, transform=character(),
                   equal=TRUE,
                   colsOnly=TRUE,
                   ignoreColOrder=FALSE,
                   ignoreNameCase=FALSE,
                   ...)
```

Arguments

model	The "correct" object.
comparison	The object to be compared with the model.
transform	A character vector containing any transformations that have been performed on the objects prior to this comparison.
equal	Whether to test for equality if the test for identity fails.
colsOnly	Only ignore (case of) column names (NOT row names).
ignoreColOrder	For data frames and lists, sort the columns or components by name before ignoring the case of names.
ignoreNameCase	When reordering the columns or components by name (i.e., when ignoreColOrder=TRUE), whether to ignore the case of the names.
...	Arguments passed to compareEqual().

Details

These functions are generic, with specific methods for data frames and lists.

Value

An object of class "comparison". Use isTRUE() to determine whether the comparison has succeeded.

Author(s)

Paul Murrell

See Also

compare and compareEqual

Examples

```
model <- data.frame(x=1:26, y=letters, z=factor(letters),
                    stringsAsFactors=FALSE)
comparison <- data.frame(a=1:26, b=letters, c=factor(letters),
                         stringsAsFactors=FALSE)
compareIgnoreNames(model, comparison)
```

---

compareIgnoreOrder *Compare Two Objects with Different Order*

Description

Compare two objects for equality, sorting them if necessary beforehand.

Usage

```
compareIgnoreOrder(model, comparison, transform = character(),
                   equal = TRUE, ...)

## S3 method for class 'data.frame'
compareIgnoreOrder(model, comparison,
                   transform=character(),
                   equal=TRUE,
                   ignoreColOrder=FALSE,
                   ignoreNameCase=FALSE,
                   ...)
```

Arguments

model	The "correct" object.
comparison	The object to be compared with the model.
transform	A character vector containing any transformations that have been performed on the objects prior to this comparison.
equal	Whether to test for equality if the test for identity fails.
ignoreColOrder	For data frames, whether to reorder columns by name first.
ignoreNameCase	For data frames and lists, whether to ignore the case of names when reordering components by name.
...	Arguments passed to compareEqual().

Details

This function is generic, with specific methods for arrays, matrices, tables, and data frames.

For arrays, matrices, and tables, the dimensions are sorted (by dimnames). For data frames, the rows are sorted, not the columns.

Value

An object of class "comparison". Use isTRUE() to determine whether the comparison has succeeded.

Author(s)

Paul Murrell

See Also

compare and compareEqual

Examples

```
compareIgnoreOrder(1:10, 10:1)
```

---

compareShorten *Compare Two Objects with Different Lengths*

Description

Compare two objects for equality, shortening either one if necessary beforehand.

Usage

```
compareShorten(model, comparison, transform = character(),
               equal = TRUE, ...)

## S3 method for class 'data.frame'
compareShorten(model, comparison, transform=character(),
               equal=TRUE, colsOnly=TRUE,
               ignoreColOrder=FALSE, ignoreNameCase=FALSE, ...)
```

Arguments

`model`	The "correct" object.
`comparison`	The object to be compared with the `model`.
`transform`	A character vector containing any transformations that have been performed on the objects prior to this comparison.
`equal`	Whether to test for equality if the test for identity fails.
`colsOnly`	Whether to only drop extra columns (not rows) when comparing data frames.
`ignoreColOrder`	For data frames, whether to reorder columns by name first.
`ignoreNameCase`	For data frames and lists, whether to ignore the case of names when reordering components by name.
`...`	Arguments passed to `compareEqual()`.

Details

This function checks whether the two objects being compared are of the same size and, if they are not, it shrinks the larger one. Then the two objects are compared using `compareIdentical()` and, if that fails and `equal=TRUE`, using `compareEqual()`.

This function is generic, with specific methods for arrays, matrices, tables, and data frames.

For vectors, extra elements are dropped from the longer object. For data frames, extra columns, (and, if `colsOnly=FALSE`, extra rows) are dropped. For lists, extra components are dropped. For arrays, extra dimensions are dropped. For matrices, the `comparison` is forced to be two-dimensional. For tables, extra dimensions are collapsed (using `sum()`).

Value

An object of class `"comparison"`. Use `isTRUE()` to determine whether the comparison has succeeded.

Author(s)

Paul Murrell

See Also

`compare` and `compareEqual`

Examples

```
compareShorten(1:5, 1:10)
compareShorten(matrix(1:10, ncol=2),
               array(1:100, dim=c(5, 2, 10)))
```

---

comparison-class *Comparison Between Two Objects*

Description

This class is used to represent the result of a comparison between two objects.

It is just a list, with the following components:

result Logical value indicating whether the comparison succeeded (the objects were the same).

transform Character vector of descriptions of any transformations that have been applied to the objects being compared.

tM The transformed model object.

tC The transformed comparison object.

tMpartial The partially transformed model object.

tCpartial The partially transformed comparison object.

partialTransform The transformations that have been applied to the objects being compared, *except for* any transformations that were applied during the test for equality (i.e., by `compareEqual()`).

Usage

```
isTRUE(x)

transforms(comp)
```

Arguments

`x`, `comp`	A comparison object.

Details

Partial values in the result of a comparison are useful for when a comparison fails, but a subsequent comparison will be attempted, i.e., for daisy-chaining the various `compareSomething()` functions, i.e., what `compare()` does.

Value

`isTRUE()` returns a logical value, based on the comparison `result`.

`transforms()` returns a character vector of transformation descriptions.

Author(s)

Paul Murrell

See Also

compare and compareEqual

Examples

```
isTRUE(compareIgnoreOrder(1:10, 10:1))
transforms(compareIgnoreOrder(1:10, 10:1))
```

---

markQuestions *Apply a Marking Scheme*

Description

This function applies a marking scheme to a set of comparisons to produce a set of marks.

Usage

```
markQuestions(result, ...)
```

Arguments

result	A set of comparison results, as generated by the compareFiles() function.
...	One or more marking schemes, as generated by the questionMarks() function.

Value

A matrix of marks.

Author(s)

Paul Murrell

See Also

questionMarks and compareFiles

Examples

```
modelNames <- c("id", "age",
                "edu", "class",
                "IndianMothers")
files <- list.files(system.file("example", package="compare"),
                    pattern="^student[0-9]+[.]R$",
                    full.names=TRUE)

results <- compareFiles(files,
                        modelNames,
                        system.file("example", "model.R", package="compare"),
                        allowAll=TRUE,
                        resultNames=gsub(".+[/]|[.]R", "", files))
q1 <- questionMarks(c("id", "age", "edu", "class"),
                    maxMark=2,
                    rule("id", 1),
                    rule("age", 1),
                    rule("edu", 1),
                    rule("class", 1,
                         transformRule("coerced", 1)))

q2 <- questionMarks("IndianMothers",
                    maxMark=1,
                    rule("IndianMothers", 1))

markQuestions(results, q1, q2)
```

---

questionComments *Define a Commenting Scheme*

Description

These functions are used to specify how the result of a comparison between two objects should be converted to a set of comments for feedback.

Usage

```
questionComments(answerNames, ...)

comments(answerName, ...)

transformComment(pattern, comment)
```

Arguments

answerNames	The names of objects that have been compared.
answerName	The name of one object that has been compared.
pattern	A regular expression to search for within the comparison transformations.

comment	A comment to record if the regular expression is matched.
...	For questionComments, zero or more comments (as generated by the the comments() function); for comments(), zero or more transformation comments (as generated by the transformComment() function.

Details

These functions are used to define a commenting scheme. The function commentQuestions() can then be used to apply the results to a set of comparisons, as generated by compareFiles().

Value

An object of class "questionComments".

Author(s)

Paul Murrell

See Also

commentQuestions and compareFiles

Examples

```
# Comment a comparison involving several objects
# Start with a comment of 1 and deduct 1 if the comparison failed
questionComments(c("id", "age", "edu", "class"),
                 comments("class",
                          transformComment("coerced",
                                           "'class' is a factor!")))
```

---

questionMarks *Define a Marking Scheme*

Description

These functions are used to specify how the result of a comparison between two objects should be converted to a numeric mark.

Usage

```
questionMarks(answerNames, maxMark, ...)

rule(answerName, falseMark, ...)

transformRule(pattern, mark)
```

Arguments

answerNames	The names of objects that have been compared.
maxMark	The maximum mark for the question.
answerName	The name of one object that has been compared.
falseMark	How many marks to deduct if the comparison result was FALSE.
pattern	A regular expression to search for within the comparison transformations.
mark	How many marks to deduct if the regular expression is matched.
...	For questionMarks, zero or more marking rules (as generated by the the rule() function); for rule(), zero or more transformation rules (as generated by the transformRule() function.

Details

These functions are used to define a marking scheme. The function markQuestions() can then be used to apply the results to a set of comparisons, as generated by compareFiles().

Value

An object of class "questionMarks".

Author(s)

Paul Murrell

See Also

markQuestions and compareFiles

Examples

```
# Check a comparison involving an object called 'IndianMothers'
# Start with a mark of 1 and deduct 1 if the comparison failed
questionMarks("IndianMothers",
              maxMark=1,
              rule("IndianMothers", 1))

# Check comparisons involving several objects
# Start with a mark of 2, for each unsuccessful comparison
# deduct 1 mark, and if a comparison involving the object
# 'class' includes a coercion transformation, deduct 1 mark.
questionMarks(c("id", "age", "edu", "class"),
              maxMark=2,
              rule("id", 1),
              rule("age", 1),
              rule("edu", 1),
              rule("class", 1,
                   transformRule("coerced", 1)))
```

`sourceFile` *Evaluate R Code from a File*

Description

Evaluate R code from a file within a local environment and return a list of objects of interest.

Usage

```
sourceFile(filename, modelNames)
```

Arguments

`filename`	The name of a file containing R code.
`modelNames`	The names of objects of interest that should be generated by the R code in the file and that we want returned.

Value

A list containing one component for each name in `modelNames`.

Author(s)

Paul Murrell

See Also

`compareFile`

Examples

```
modelNames <- c("id", "age",
                "edu", "class",
                "IndianMothers")
sourceFile(system.file("example", "student1.R", package="compare"),
           modelNames)
```

Index