# Package 'dataCompareR'

November 14, 2017

**Title** Compare Two Data Frames and Summarise the Difference

**Version** 0.1.1

**Description** Easy comparison of two tabular data
objects in R. Specifically designed to show differences between two sets of
data in a useful way that should make it easier to understand the differences,
and if necessary, help you work out how to remedy them. Aims
to offer a more useful output than all.equal() when your two data sets do not
match, but isn't intended to replace all.equal() as a way to test for equality.

**Depends** R (>= 3.2.3)

**Imports** dplyr (>= 0.5.0), knitr, stringi, markdown

**URL** https://github.com/capitalone/dataCompareR

**BugReports** https://github.com/capitalone/dataCompareR/issues

**License** Apache License 2.0 | file LICENSE

**LazyData** true

**RoxygenNote** 6.0.1

**Suggests** testthat, data.table, tibble, bit64, rmarkdown, titanic

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Rob Noble-Eddy [aut, cre],
Sarah Johnston [aut],
Sarah Pollicott [aut],
Merlijn van Horssen [aut],
Lukas Drapal [ctb],
Nikolaos Perrakis [ctb],
Nikhil Thomas Joy [ctb],
Shahriar Asta [ctb],
Karandeep Lidher [ctb],
Dan Kellett [ctb],
Kevin Chisholm [ctb],
Laura Joy [ctb],
Fergus Wadsley [ctb],

Heather Hackett [ctb],
David Robinson [ctb],
Cheryl Renton [ctb],
Matt Triggs [ctb],
Krishan Bhasin [ctb],
Carola Deppe [ctb]

**Maintainer** Rob Noble-Eddy <opensource@capitalone.com>

**Repository** CRAN

**Date/Publication** 2017-11-14 09:01:03 UTC

# R topics documented:

---

`allVarMatchMessage` *allVarMatchMessage*

---

## Description

Returns data about matching

## Usage

```
allVarMatchMessage(x)
```

## Arguments

| | |
|---|---|
| x | An dataCompareR object |

### Value

A string containing the required message

---

`checkEmpty` *checkEmpty*

---

### Description

checkEmpty

### Usage

```
checkEmpty(df)
```

### Arguments

| | |
|---|---|
| `df` | a data frame |

### Value

None. Stops if empty.

### Examples

```
## Not run: checkEmpty(iris)
```

---

`checkForRCompareCol` *checkForRcompareCol*

---

### Description

checkForRcompareCol

### Usage

```
checkForRCompareCol(df1)
```

### Arguments

| | |
|---|---|
| `df1` | a data frame |

### Value

None. Stops if error.

### Examples

```
## Not run: checkForRcompareCol(iris)
```

checkKeysExist *checkKeysExist*

## Description

checkKeysExist

## Usage

```
checkKeysExist(df, keys)
```

## Arguments

| | |
|---|---|
| df | a data frame |
| keys | a list of expected columns |

## Value

None. Stops if keys are not present as column names in df.

## Examples

```
## Not run: checkKeysExist(iris, 'columnName')
```

checkUniqueness *Checks that a list of indexes areunique*

## Description

Checks that a list of indexes areunique

## Usage

```
checkUniqueness(df_indices)
```

## Arguments

| | |
|---|---|
| df_indices | A vector of values |

## Value

Boolean - true if all values in vector are unique, false if not

## Examples

```
## Not run: checkUniqueness(c('car','van','van'))
## Not run: checkUniqueness(c('car','van','bus'))
```

## cleanColNames *cleanColNames : get colnames, remove leading and trailing whitespace and push to upper case*

### Description

cleanColNames : get colnames, remove leading and trailing whitespace and push to upper case

### Usage

```
cleanColNames(DF)
```

### Arguments

| | |
|---|---|
| DF | Input dataframe |

### Value

colInfo dataframe containing original and treated column names of DF

## coerceData *coerceData*

### Description

coerceData

### Usage

```
coerceData(doa, dob)
```

### Arguments

| | |
|---|---|
| doa | Data object A (any object that can be coerced to a data frame) |
| dob | Data object B (any object that can be coerced to a data frame) |

### Value

A list of 2 data frames, which is DOA and DOB coerced as data.frames

### Examples

```
## Not run: irisMatrix <- as.matrix(iris)
## Not run: coerceData(irisMatrix,iris)
```

`coerceFactorsToChar` *coerceFactorsToChar: convert all factor type fields to characters*

## Description

coerceFactorsToChar: convert all factor type fields to characters

## Usage

```
coerceFactorsToChar(DF)
```

## Arguments

| | |
|---|---|
| `DF` | Input dataframe |

## Value

`DF` with factor fields converted to character type

## Examples

```
## Not run: coerceFactorsToChar(iris)
```

`collapseClasses` *collapseClasses. Collapse the classes of an object to a single string*

## Description

collapseClasses. Collapse the classes of an object to a single string

## Usage

```
collapseClasses(x)
```

## Arguments

| | |
|---|---|
| `x` | any object |

## Value

a string listing the classes of x, seperated by commas

## Examples

```
## Not run: collapseClasses(iris)
## Not run: collapseClasses("hello")
```

## colsWithUnequalValues *colsWithUnequalValues: a dataframe summarising a column with unequal values*

### Description

colsWithUnequalValues: a dataframe summarising a column with unequal values

### Usage

```
colsWithUnequalValues(x, mismatches)
```

### Arguments

| | |
|---|---|
| x | the column to be considered |
| mismatches | - a mismatches object from an dataCompareR object |

### Value

data frame with a summary of the mismatching column

## compareData *Compare data. Wrapper for comparison functionality.*

### Description

Compare data. Wrapper for comparison functionality.

### Usage

```
compareData(DFA, DFB, keys = NULL, maxMismatches)
```

### Arguments

| | |
|---|---|
| DFA | dataframe as returned from prepareData |
| DFB | dataframe as returend from prepareData |
| keys | vector of chars - names of index variables |
| maxMismatches | Integer. The max number of mismatches to assess, after which dataCompareR will stop (without producing a dataCompareR object). Designed to improve performance for large datasets. |

### Value

mismatchObject containing mismatch data for each of the variables in the dataframes

## Examples

```
## Not run: compareData(iris, iris)

## Not run: iris2 <- iris
## Not run: iris2[1,1] <- 5.2
## Not run: iris2[2,1] <- 5.2
## Not run: compareData(iris, iris2)

## Not run: compareData(pressure, pressure, keys = 'temperature')
```

---

`compareNames` *compareNames : compare the intersect of colInfoA and colInfoB and return boolean of matched columns for each data frame*

---

## Description

compareNames : compare the intersect of colInfoA and colInfoB and return boolean of matched columns for each data frame

## Usage

```
compareNames(colInfoA, colInfoB)
```

## Arguments

| | |
|---|---|
| `colInfoA` | input data frames with original and treated column names |
| `colInfoB` | input data frames with original and treated column names |

---

`createAntiSubset` *Create a dataframe of the rows that don't match*

---

## Description

Create a dataframe of the rows that don't match

## Usage

```
createAntiSubset(index_antisubset, original_keys, index_key, df)
```

## Arguments

| | |
|---|---|
| `index_antisubset` | Vector of mismatching indices |
| `original_keys` | A character array |
| `index_key` | A character array |
| `df` | A data frame |

## Value

A dataframe containing the dropped rows

---

`createCleaningInfo` *Converts cleaning info into a format consumable by updateCompareObject.*

---

## Description

Converts cleaning info into a format consumable by updateCompareObject.

## Usage

```
createCleaningInfo(compObj, cleaningInfo)
```

## Arguments

| | |
|---|---|
| `compObj` | dataCompareRobject to be updated |
| `cleaningInfo` | list of cleaning information |

## Value

`compObj` updated dataCompareRobject

---

`createColMatching` *Converts the output of the column matching logic to something consumable by updateCompareObject.*

---

## Description

Converts the output of the column matching logic to something consumable by updateCompareObject.

## Usage

```
createColMatching(compObj, colMatchInfo)
```

## Arguments

| | |
|---|---|
| `compObj` | dataCompareRobject instance to be updated |
| `colMatchInfo` | List output from the column matching logic |

## Value

`compObj` updated with colMatching block

## createCompareObject *Generates an empty list of the correct class to store results*

### Description

Generates an empty list of the correct class to store results

### Usage

```
createCompareObject()
```

### Value

A list of class dataCompareRObject

## createMeta *Takes the raw info for the meta block of the output and puts it in a format usable by the updateCompareObject function*

### Description

Takes the raw info for the meta block of the output and puts it in a format usable by the updateCompareObject function

### Usage

```
createMeta(dataCompareRobject, DFA, DFB, arguments, timestamp, roundDigits)
```

### Arguments

| | |
|---|---|
| dataCompareRobject | Object of class dataCompareRobject |
| DFA | First data set passed in to the dataCompareR function |
| DFB | Second data set passed in to the dataCompareR function |
| arguments | Collection of arguments passed to compare object with labels that match the dataCompareR arg definitions |
| timestamp | Timestamp |
| roundDigits | The number of digits to round to, using round |

### Value

dataCompareRobject

## createMismatches *Create mismatch object*

### Description

Create mismatch object

### Usage

```
createMismatches(compObj, misObj, keys)
```

### Arguments

| | |
|---|---|
| compObj | RCompareObject, output from processFlow |
| misObj | MismatchObject, output from compareData (processFlow) |
| keys | Character vector, the keys matched on, to allow removal of any extra columns introudced by the compare process |

### Value

The mismatch object

## createMismatchObject *Create mismatch object*

### Description

Create mismatch object

### Usage

```
createMismatchObject(dat_a, dat_b, dat_eq, str_index)
```

### Arguments

| | |
|---|---|
| dat_a | dataframe, output from prepareData |
| dat_b | dataframe, output from prepareDate |
| dat_eq | dataframe, output from locateMismatches |
| str_index, | vector of index variables (could have length 1) |

### Value

An dataCompareR mismatch object

### Examples

```
## Not run: createMismatchObject(dataA, dataB, mism, idx)
```

createReportText *createReportText: prepares text which is used in the summary report Saves R markdown and HTML reports in the area specified by the user. Reports are called RcompareReport.Rmd (.html) Uses knitr package to create tables in the markdown (createReportText function) and HTML report.*

## Description

createReportText: prepares text which is used in the summary report Saves R markdown and HTML reports in the area specified by the user. Reports are called RcompareReport.Rmd (.html) Uses knitr package to create tables in the markdown (createReportText function) and HTML report.

## Usage

```
createReportText(x)
```

## Arguments

| | |
|---|---|
| x | input object which summary comparison information |

## Value

text in R markdown format

## Examples

```
## Not run: createReportText(x=MysummaryCompareObject)
```

createRowMatching *function for updating a compare object with information passed to it from the match rows function*

## Description

function for updating a compare object with information passed to it from the match rows function

## Usage

```
createRowMatching(compObj, x, matchKey)
```

### Arguments

| | |
|---|---|
| compObj | dataCompareRobject to be updated |
| x | Object of information with classes related to the relevant section of the dataCompareRobject |
| matchKey | the list of keys based on which the row matching was performed |

### Value

compObj Updated dataCompareRobject

---

## createTextSummary *createTextSummary: create a text based summary of an dataCompareR object*

---

### Description

createTextSummary: create a text based summary of an dataCompareR object

### Usage

```
createTextSummary(x, ...)
```

### Arguments

| | |
|---|---|
| x | an dataCompareR object |
| ... | Arguments passed on to other functions |

### Value

cat's lines to the screen (or to be captured) cat(newLine)

---

## currentObjVersion *Place to store and access the current object version.*

---

### Description

Place to store and access the current object version.

### Usage

```
currentObjVersion()
```

### Value

currentVersion int of the version number

executeCoercions *executeCoercions:*

## Description

executeCoercions:

## Usage

```
executeCoercions(DFA, DFB, WhitespaceTrim = TRUE)
```

## Arguments

| | |
|---|---|
| DFA | Input dataframe A |
| DFB | Input dataframe B |
| WhitespaceTrim | User defined boolean for whether leading/trailing white space is trimmed in strings (TRUE / FALSE) |

## Value

out list containing 3 data frames DFA, DFB and DataTypes

DFA Dataframe with factor fields converted to character type and white space trimming (if option is selected by the user)

DFB Dataframe with factor fields converted to character type and white space trimming (if option is selected by the user)

DataTypes Dataframe with field types before and after cleaning for both DFA and DFB

## Examples

```
## Not run: executeCoercions(DFA=iris,DFB=iris,WhitespaceTrim= TRUE)
```

---

generateMismatchData *Extract data from a dataCompareR comparison*

## Description

Produces a list of two data frames, containing the mismatched rows from the two input tables

Note that this function requires the user to pass in the two data frames used in the inital comparison. If this data does not match that used for the generation of the dataCompareR object the results produced will not be accurate.

## Usage

```
generateMismatchData(x, dfA, dfB, ...)
```

### Arguments

| | |
|---|---|
| x | A dataCompareRobject. |
| dfA | Data frame (or object coercable to a data frame). One of the two data frames used in the initial rCompare call. |
| dfB | Data frame (or object coercable to a data frame). One of the two data frames used in the initial rCompare call. |
| ... | Unused currently, may be used in future |

### Value

mismatchData A list containing two objects: mismatched rows in first data object and mismatched rows in second data object

### See Also

Other dataCompareR.functions: print.dataCompareRobject, rCompare, saveReport, summary.dataCompareRobject

---

## getCoercions *Subsets on the variables that have a coercion.*

---

### Description

Subsets on the variables that have a coercion.

### Usage

```
getCoercions(typesDf)
```

### Arguments

| | |
|---|---|
| typesDf | Dataframe of type information from the executeCoercion function |

### Value

coercedT Subset version of typesDf where a coercion occurred

`getMismatchColNames` *Extracts the column names only in one data frame from a table of match information*

## Description

Extracts the column names only in one data frame from a table of match information

## Usage

```
getMismatchColNames(colMatchInfo, colNameCol, matchFlagCol)
```

## Arguments

| | |
|---|---|
| `colMatchInfo` | Dataframe with column names, match flag |
| `colNameCol` | Name of the column with the column names |
| `matchFlagCol` | Name of the column with the match flag |

## Value

Vector of column names that do not match

`is.dataCompareRobject` *Check object is of class dataCompareRobject*

## Description

Check object is of class dataCompareRobject

## Usage

```
is.dataCompareRobject(x)
```

## Arguments

| | |
|---|---|
| `x` | An object |

## Value

A boolean: TRUE if object is class dataCompareRobject and FALSE if not

`isNotNull` *isNotNull: is object not null*

## Description

isNotNull: is object not null

## Usage

```
isNotNull(x)
```

## Arguments

| | |
|---|---|
| `x` | an object |

## Value

`boolean` is object null T/F

## Examples

```
## Not run: isNotNull(NULL)
## Not run: isNotNull(5)
```

`listObsNotVerbose` *listObsNotVerbose*

## Description

Return a summary of mismatching data

## Usage

```
listObsNotVerbose(i, x, uniquevarlist, nObs)
```

## Arguments

| | |
|---|---|
| `i` | The position of the element we want to compare |
| `x` | An dataCompareR object |
| `uniquevarlist` | A list of the variables in the compare |
| `nObs` | How many obervations to return |

## Value

A list of mismatching observations from start/end of mismatches

## listObsVerbose *listObsVerbose*

### Description

Return all mismatching data

### Usage

```
listObsVerbose(i, x)
```

### Arguments

| | |
|---|---|
| i | The position of the element we want to compare |
| x | An dataCompareR object |

### Value

A list of mismatching observations

## locateMismatches *Checks whether elements in two input data frames are equal.*

### Description

Checks whether elements in two input data frames are equal.

### Usage

```
locateMismatches(DFA, DFB, keys = NULL, maxMismatches = NA)
```

### Arguments

| | |
|---|---|
| DFA | input data frame |
| DFB | input data frame |
| keys | character vector of index variables |
| maxMismatches | Integer. The max number of mismatches to assess, after which dataCompareR will stop (without producing a dataCompareR object). Designed to improve performance for large datasets. |

### Value

data frame containing keys and boolean logic of match/no match for each element If data types are not equal returns FALSE. Treats NA and NaN as unequal.

## makeValidKeys *makeValidKeys*

### Description

Correct syntactically invalid Keys

### Usage

```
makeValidKeys(keys)
```

### Arguments

| | |
|---|---|
| keys | A character vector |

### Value

A character vector with syntactically valid names

### Examples

```
## Not run: makeValidKeys(c(" hello", "__BAD NAME___")
```

## makeValidNames *makeValidNames*

### Description

Correct syntactically invalid names in a data frame

### Usage

```
makeValidNames(df)
```

### Arguments

| | |
|---|---|
| df | A data frame |

### Value

A data frame with syntactically valid names

### Examples

```
## Not run: makeValidNames(iris)
```

`matchColumns` *matchColumns : create subset of DFA and DFB to contain matching column names for both data frames*

## Description

matchColumns : create subset of DFA and DFB to contain matching column names for both data frames

## Usage

```
matchColumns(DFA, DFB)
```

## Arguments

| | |
|---|---|
| `DFA` | input data frame |
| `DFB` | input data frame |

## Value

matchColOut named list of data frames. subsetA,subsetB contain only columns common to both data frames. colInfoA,colInfoB contain mapping of column names from original to treated and boolean indicator of common columns.

`matchMultiIndex` *Generate two dataframes that contain the same rows based on a two-column index*

## Description

Generate two dataframes that contain the same rows based on a two-column index

## Usage

```
matchMultiIndex(df_a, df_b, indices)
```

## Arguments

| | |
|---|---|
| `df_a` | A dataframe |
| `df_b` | A dataframe |
| `indices` | A char vector |

## Value

A list containing the two dataframes, subsetted by shared indices, and a list which itself contains the vectors for the dropped rows

`matchNoIndex` *Generate two dataframes that contain the same rows based on a two-column index*

## Description

Generate two dataframes that contain the same rows based on a two-column index

## Usage

```
matchNoIndex(df_a, df_b)
```

## Arguments

| | |
|---|---|
| `df_a` | A dataframe |
| `df_b` | A dataframe |

## Value

A list containing the two dataframes, subsetted to the size of the smaller one, and a list containing vectors of the rows dropped.

`matchRows` *Generate two dataframes and returns subsets of these dataframes that have shared rows.*

## Description

Generate two dataframes and returns subsets of these dataframes that have shared rows.

## Usage

```
matchRows(df_a, df_b, indices = NA)
```

## Arguments

| | |
|---|---|
| `df_a` | A dataframe |
| `df_b` | A dataframe |
| `indices` | The indices to match rows between `df_a` and `df_b`. Can be NA, single character, or a vector of characters |

## Value

A list containing the two dataframes, subsetted by shared indices, and a list which itself contains dataframes for the dropped rows

`matchSingleIndex` *Generate two dataframes that contain the same rows based on a single index*

## Description

Generate two dataframes that contain the same rows based on a single index

## Usage

```
matchSingleIndex(df_a, df_b, index_key, original_keys)
```

## Arguments

| | |
|---|---|
| `df_a` | A dataframe |
| `df_b` | A dataframe |
| `index_key` | A character vector |
| `original_keys` | A character vector |

## Value

A list containing the two dataframes, subsetted by shared indices, and a list which itself contains the vectors for the dropped rows

`metaDataInfo` *Creates a list of info about the dataframe.*

## Description

Creates a list of info about the dataframe.

## Usage

```
metaDataInfo(name, df)
```

## Arguments

| | |
|---|---|
| `name` | The variable name of the df from the dataCompareR function call |
| `df` | A data frame |

## Value

`dfInfo` A list of info about the data frame

`mismatchHighStop` *mismatchHighStop Checks if we've exceeded threshold of mismatches*

## Description

mismatchHighStop Checks if we've exceeded threshold of mismatches

## Usage

```
mismatchHighStop(trueFalseMatrix, maxMismatches)
```

## Arguments

| | |
|---|---|
| `trueFalseMatrix` | a matrix of true/false |
| `maxMismatches` | number of mismatches at which the routine stopes |

## Value

Nothing. Stops if threshold exceeded

`orderColumns` *orderColumns: order columns by treated column names*

## Description

orderColumns: order columns by treated column names

## Usage

```
orderColumns(colInfo)
```

## Arguments

| | |
|---|---|
| `colInfo` | dataframe containing original and treated column names of DF |

## Value

ordered colInfo dataframe containing original and treated column names of DF

`outputSectionHeader` *outputSectionHeader: creates an outputSectionHeader*

## Description

outputSectionHeader: creates an outputSectionHeader

## Usage

```
outputSectionHeader(headerName)
```

## Arguments

| | |
|---|---|
| `headerName` | a header name |

## Value

`character` a character based section headers

---

`prepareData` *prepareData Prepares data for comparison in 3 stages. 1. Match columns - filter dataframes to those columns that match and summarise differences 2. Match rows - filter dataframes to those rows that match and summarise differences 3. Coerce data*

## Description

prepareData Prepares data for comparison in 3 stages. 1. Match columns - filter dataframes to those columns that match and summarise differences 2. Match rows - filter dataframes to those rows that match and summarise differences 3. Coerce data

## Usage

```
prepareData(dfA, dfB, keys = NA, trimChars = TRUE)
```

## Arguments

| | |
|---|---|
| `dfA` | data frame. The first data object. dataCompareR will attempt to coerce all data objects to data frames. |
| `dfB` | data frame. The second data object. dataCompareR will attempt to coerce all data objects to data frames. |
| `keys` | String. Name of identifier column(s) used to compare dfA and dfB. NA if no identifier (row order will be used instead), a character for a single column name, or a vector of column names to match of multiple columns |
| `trimChars` | Boolean. If true, strings and factors have whitespace trimmed before comparison. |

## Value

`dataCompareRObject` containing details of the comparison

## Examples

```
## Not run: dfA <- iris
## Not run: dfB <- iris
## Not run: keys <- NA
## Not run: prepareData(dfA,dfB,keys, trimChars = TRUE)
```

---

`print.dataCompareRobject`

*Printing RCompare Output*

---

## Description

Prints a brief report of an dataCompareR object to the screen.

## Usage

```
## S3 method for class 'dataCompareRobject'
print(x, nVars = 5, nObs = 5,
  verbose = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | an object of class "dataCompareR", usually a result of a call to `rCompare`. |
| `nVars` | the number of mismatched columns to print and extract rows for |
| `nObs` | the number of rows to print from the top and bottom of the mismatched list for each selected column |
| `verbose` | logical; if TRUE will print out the full list of columns and rows that do not match |
| `...` | Passes additional arguments to print |

## See Also

Other dataCompareR.functions: `generateMismatchData`, `rCompare`, `saveReport`, `summary.dataCompareRobject`

## Examples

```
rc1 <- rCompare(iris,iris)
print(rc1)
```

## print.summary.dataCompareRobject *Printing summaryRCompare Output*

### Description

Printing summaryRCompare Output

### Usage

```
## S3 method for class 'summary.dataCompareRobject'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | an object of class "summary.dataCompareRobject", usually a result of a call to summary.dataCompareRobject. |
| ... | Additional arguments passsed on to createTextSummary |

### Examples

```
rc1 <- rCompare(iris,iris)
summary(rc1)
```

## processFlow *processFlow Handles the process flow for the whole package*

### Description

processFlow Handles the process flow for the whole package

### Usage

```
processFlow(dfa, dfb, roundDigits, keys, mismatches, trimChars, argsIn)
```

### Arguments

| | |
|---|---|
| dfa | Dataframe. One of the two data frames to be compared |
| dfb | Dataframe. One of the two data frames to be compared |
| roundDigits | Integer. If NA, numerics are not rounded before comparison. If /coderoundDigits is specified, numerics are rounded to /coderoundDigits decimal places using round. |
| keys | The keys used to match rows between dfa and dfb |

| | |
|---|---|
| mismatches | Integer. The max number of mismatches to assess, after which dataCompareR will stop (without producing a dataCompareR object). Designed to improve performance for large datasets. |
| trimChars | Boolean. Do we trim characers before comparing? |
| argsIn | The arguments that were passed to the main dataCompareR function |

## Value

dataCompareRObject containing details of the comparison

---

rCompare *Compare two data frames*

---

## Description

Compare two data frames (or objects coercible to data frames) and produce a dataCompareR object containing details of the matching and mismatching elements of the data. See vignette("dataCompareR") for more details.

## Usage

```
rCompare(dfA, dfB, keys = NA, roundDigits = NA, mismatches = NA,
  trimChars = FALSE)
```

## Arguments

| | |
|---|---|
| dfA | data frame. The first data object. dataCompareR will attempt to coerce all data objects to data frames. |
| dfB | data frame. The second data object. dataCompareR will attempt to coerce all data objects to data frames. |
| keys | String. Name of identifier column(s) used to compare dfA and dfB. NA if no identifier (row order will be used instead), a character for a single column name, or a vector of column names to match of multiple columns |
| roundDigits | Integer. If NA, numerics are not rounded before comparison. If specified, numerics are rounded to the specified number of decimal places using round. |
| mismatches | Integer. The max number of mismatches to assess, after which dataCompareR will stop (without producing an dataCompareR object). Designed to improve performance for large data sets. |
| trimChars | Boolean. If true, strings and factors have whitespace trimmed before comparison. |

## Value

An dataCompareR object. An S3 object containing details of the comparison between the two data objects. Can be used with summary, print, saveReport and generateMismatchData

## See Also

Other dataCompareR.functions: generateMismatchData, print.dataCompareRobject, saveReport, summary.dataCompareRobject

## Examples

```
iris2 <- iris
iris2 <- iris2[1:130,]
iris2[1,1] <- 5.2
iris2[2,1] <- 5.2
rCompare(iris,iris2,key=NA)
compDetails <- rCompare(iris,iris2,key=NA, trimChars = TRUE)
print(compDetails)
summary(compDetails)

pressure2 <- pressure
pressure2[2,2] <- pressure2[2,2] + 0.01
rCompare(pressure2,pressure2,key='temperature')
rCompare(pressure2,pressure2,key='temperature', mismatches = 10)
```

---

# rcompObjItemLength *rcompObjItemLength: return length of an item, returning 0 if null, and handling the fact that we might have a data frames or a vector*

---

## Description

rcompObjItemLength: return length of an item, returning 0 if null, and handling the fact that we might have a data frames or a vector

## Usage

```
rcompObjItemLength(x)
```

## Arguments

| | |
|---|---|
| x | an object |

## Value

length, numeric

## rounddf *Round all numeric fields in a data frame*

### Description

Round all numeric fields in a data frame

### Usage

```
rounddf(df, roundDigits)
```

### Arguments

| | |
|---|---|
| df | A data frame to round |
| roundDigits | Number of digits to round to |

### Value

A rounded data frame

## saveReport *Save a report based on a dataCompareR object*

### Description

Saves R markdown and HTML reports in the area specified by the user.

Uses knitr and markdown to create reports. Reports have the extensions .Rmd or .html. By default the `table.css` style sheet is used for format the html output.

### Usage

```
saveReport(compareObject, reportName, reportLocation = ".",
  HTMLReport = TRUE, showInViewer = TRUE, stylesheet = NA)
```

### Arguments

| | |
|---|---|
| compareObject | a dataCompareR object. |
| reportName | String. The name of the report. Reports will be saved as reportName.Rmd and (optionally) reportName.html in `reportLocation` |
| reportLocation | String. Location to save reports specified by the user. The R markdown and (optionally) HTML reports will be saved in this area |
| HTMLReport | Boolean. Option to output html report. |
| showInViewer | Boolean. Does the html report open automatically in the viewer? |
| stylesheet | String. Optional link to customised css stylesheet |

## See Also

Other dataCompareR.functions: generateMismatchData, print.dataCompareRobject, rCompare, summary.dataCompareRobject

## Examples

```
## Not run: saveReport(rcObj, reportName = 'testReport')
```

---

`subsetDataColumns` *subsetDataColumns : create subset of DFA and DFB to contain matching column names for both data frames*

---

## Description

subsetDataColumns : create subset of DFA and DFB to contain matching column names for both data frames

## Usage

```
subsetDataColumns(DFA, DFB, colInfoList)
```

## Arguments

| | |
|---|---|
| `DFA` | input data frame |
| `DFB` | input data frame |
| `colInfoList` | named list containing the column mapping data frames and the list of common column names |

## Value

matchColOut named list of data frames. subsetA,subsetB contain only columns common to both data frames. colInfoA,colInfoB contain mapping of column names from original to treated and boolean indicator of common columns.

summary.dataCompareRobject *Summarizing RCompare Output*

## Description

Summarizing RCompare Output

## Usage

```
## S3 method for class 'dataCompareRobject'
summary(object, mismatchCount = 5, ...)
```

## Arguments

| | |
|---|---|
| object | an dataCompareR object, usually a result of a call to rCompare. |
| mismatchCount | Integer. How many mismatches to include in tables |
| ... | Passes any additional arguments (not used in current version) |

## Value

The function summary.dataCompareR computes and returns a list of summary details from the dataCompareR output given in object containing

| | |
|---|---|
| datanameA | name of the first dataframe in the compare call |
| datanameB | name of the second dataframe in the compare call |
| nrowA | the number of rows in datanameA |
| nrowB | the number of rows in datanameB |
| version | the version of rCompare used to generate the dataCompareR object object |
| runtime | the date and time the dataCompareR object object was created |
| rversion | the version of R used |
| datasetSummary | a data frame containing the meta data information on datanameA and datanameB |
| ncolCommon | the number of columns of the same name contained in both datanameA and datanameB |
| ncolInAOnly | the number of columns only in datanameA |
| ncolInBOnly | the number of columns only in datanameB |
| ncolID | the number of columns used to match rows in datanameA and datanameB |
| typeMismatch | a data frame detailing which columns in both datanameA and datanameB have different class types |
| typeMismatchN | the number of columns with different variable types |
| nrowCommon | the number of rows with matching ID columns in both datanameA and datanameB |
| nrowInAOnly | the number of rows with non matching ID columns in datanameA |

| | |
|---|---|
| nrowInBOnly | the number of rows with non matching ID columns in datanameB |
| nrowSomeUnequal | the number of matched rows where at least one value is unequal |
| nrowAllEqual | the number of matched rows where all values are equal |
| ncolsAllEqual | the number of matched columns where all values are equal |
| ncolsSomeUnequal | the number of matched columns where at least one value is unequal |
| colsWithUnequalValues | a data frame detailing the mismatches for each matched column |
| nrowNAmisMatch | the number of matched numeric rows that contain a NA |
| maxDifference | the maximum difference between numeric columns from all matched columns |

## See Also

Other dataCompareR.functions: generateMismatchData, print.dataCompareRobject, rCompare, saveReport

## Examples

```
rc1 <- rCompare(iris,iris)
summary(rc1)
```

---

# trimCharVars *trimCharVars: trim white spaces in character variables from an input dataframe*

---

## Description

trimCharVars: trim white spaces in character variables from an input dataframe

## Usage

```
trimCharVars(DF)
```

## Arguments

| | |
|---|---|
| DF | Input dataframe |

## Value

DF with preceding and trailing white spaces removed from character fields

## Examples

```
## Not run: trimCharVars(iris)
```

`updateCompareObject` *Generic function for updating a compare object with information passed to it, that has methods based on the class of the info argument.*

## Description

Generic function for updating a compare object with information passed to it, that has methods based on the class of the info argument.

## Usage

```
updateCompareObject(x, compObj)
```

## Arguments

| | |
|---|---|
| `x` | Object of information with classes related to the relevant section of the dataCompareRobject |
| `compObj` | dataCompareRobject to be updated |

## Value

compObj Updated dataCompareRobject

`updateCompareObject.cleaninginfo` *Updates cleaning info in the compare object*

## Description

Updates cleaning info in the compare object

## Usage

```
## S3 method for class 'cleaninginfo'
updateCompareObject(x, compObj)
```

## Arguments

| | |
|---|---|
| `x` | list of type cleaninginfo with data types |
| `compObj` | dataCompareRobject to be updated |

## Value

`compObj` updated dataCompareRobject

## updateCompareObject.colmatching

*Adds a colMatching block to the output*

### Description

Adds a colMatching block to the output

### Usage

```
## S3 method for class 'colmatching'
updateCompareObject(x, compObj)
```

### Arguments

| | |
|---|---|
| x | List of class colmatching with column matching info |
| compObj | dataCompareRobject instance to be updated |

### Value

compObj Updated dataCompareRobject

## updateCompareObject.matches

*Adds a colMatching block to the output*

### Description

Adds a colMatching block to the output

### Usage

```
## S3 method for class 'matches'
updateCompareObject(x, compObj)
```

### Arguments

| | |
|---|---|
| x | List of class 'matches' with column matching info |
| compObj | dataCompareRobject instance to be updated |

### Value

compObj Updated dataCompareRobject

---

`updateCompareObject.meta`
*Takes raw info for meta and adds it to the compare object*

---

## Description

Takes raw info for meta and adds it to the compare object

## Usage

```
## S3 method for class 'meta'
updateCompareObject(x, compObj)
```

## Arguments

| | |
|---|---|
| `x` | List of class 'meta' with data related to meta |
| `compObj` | dataCompareRobject to be appended |

## Value

`compObj` dataCompareRobject updated with meta block

---

`updateCompareObject.mismatches`
*Adds a colMatching block to the output*

---

## Description

Adds a colMatching block to the output

## Usage

```
## S3 method for class 'mismatches'
updateCompareObject(x, compObj)
```

## Arguments

| | |
|---|---|
| `x` | List of class 'mismatches' with column matching info |
| `compObj` | dataCompareRobject instance to be updated |

## Value

`compObj` Updated dataCompareRobject

## updateCompareObject.rowmatching *Adds a rowMatching block to the output*

### Description

Adds a rowMatching block to the output

### Usage

```
## S3 method for class 'rowmatching'
updateCompareObject(x, compObj)
```

### Arguments

| | |
|---|---|
| x | List of class rowMatching with row matching info |
| compObj | dataCompareRobject instance to be updated |

### Value

compObj Updated dataCompareRobject

## validateArguments *validateArguments*

### Description

validateArguments

### Usage

```
validateArguments(matchKey = NA, roundDigits = NA, coerceCols = TRUE,
  maxMismatch = NA)
```

### Arguments

| | |
|---|---|
| matchKey | A character or character vector of column names to match on |
| roundDigits | Integer. If NA, numerics are not rounded before comparison. If specified, numerics are rounded to the specified number of decimal places using round. |
| coerceCols | Boolean - do we coerce columns names? |
| maxMismatch | Cap for number of mismatches |

### Value

Nothing. Errors if any parameters are invalid.

## Examples

```
## Not run: validateArguments('plantName',1E-8,T,1000)
## Not run: validateArguments('colorName',1E-9,F,10)
```

---

`validateData` *validateData : routine to validate the input data*

---

## Description

validateData : routine to validate the input data

## Usage

```
validateData(df1, df2, keys = NA)
```

## Arguments

| | |
|---|---|
| `df1` | a data frame |
| `df2` | a data frame |
| `keys` | Keys used |

## Value

None. Stops if error.

## Examples

```
## Not run: validateData(iris,iris)
```

---

`variableDetails` *Create variable mismatch details*

---

## Description

Create variable mismatch details

## Usage

```
variableDetails(dat)
```

## Arguments

| | |
|---|---|
| `dat` | The mismatch data |

## Value

mismatch details

variableMismatches *Create variable mismatch table*

## Description

Create variable mismatch table

## Usage

```
variableMismatches(varname, vals_a, vals_b, vector_eq)
```

## Arguments

| | |
|---|---|
| varname, | variable to create mismatch table for |
| vals_a, | variables from dfA |
| vals_b, | variables from dfB |
| vector_eq, | a list of columns which are equal |

## Value

Mismatch table

# Index