Package ‘OuhscMunge’

March 18, 2017

Title Data Manipulation Operations

Description Data manipulation operations frequently used in OUHSC BBMC projects.

Version 0.1.7.9000

Date 2017-03-17

Author Will Beasley [aut, cre]

Maintainer Will Beasley <wibeasley@hotmail.com>

URL https://github.com/OuhscBbmc/OuhscMunge, http://ouhsc.edu/bbmc/

BugReports https://github.com/OuhscBbmc/OuhscMunge/issues

License GPL-2

LazyData TRUE

Depends R(>= 3.1.0)

Imports devtools (>= 1.8.0),
dplyr,
lubridate

Suggests RODBC,
testthat

Roxygen list(markdown = TRUE)

RoxygenNote 6.0.1

R topics documented:

clump_month_date *Assign date for a given year & month*

Description

This accepts a date, but changes the day. Set/degrade/clump all the days within a month to the same day.

Usage

```
clump_month_date(date_detailed, day_of_month = 15L)
```

Arguments

date_detailed	The Date value containing the desired year and month. The day will be overwritten. Required
day_of_month	The factor label assigned to the missing value. Defaults to 15.

Details

We use this frequently to set/degrade/clump all the days to the middle of their respective month (ie, the 15th day). The midpoint of a month is usually the most appropriate summary location. It makes graphs more intuitive. Using the midpoint of month can also avoid problems with timezones. A date won't get nudged to a neighboring month accidentally.

Value

An array of Date values.

Note

A stop error will be thrown if date_detailed is not a Date, or if day_of_month is not bounded by [1, 31]. Be careful that if you set a November date the 31st day, the result will be December 1st. Consequently, we recommend not setting the day to a value after the 28.

Author(s)

Will Beasley

Examples

```
library(OuhscMunge)
detailed <- seq.Date(from=as.Date("2011-04-21"), to=as.Date("2011-07-14"), by="day")
clumped <- clump_month_date(detailed)
table(clumped)
# 2011-04-15 2011-05-15 2011-06-15 2011-07-15
#         10         31         30         14
```

deterge *Convert (and possibly clean) a vector*

Description

Cast values to desired data type.

Usage

```
deterge_to_double(x, bound_lower, bound_upper)
deterge_to_integer(x, bound_lower, bound_upper)
```

Arguments

`x`	The input vector that needs to be cast. Required
`bound_lower`	Elements below this inclusive threshold will be set to `NA`.
`bound_upper`	Elements above this inclusive threshold will be set to `NA`.

Details

–write something here–

Value

An array of values.

Author(s)

Will Beasley

Examples

```
library(OuhscMunge)
deterge_to_double(c(NA, 1:10), 4, 8)
deterge_to_integer(c(NA, 1:10), 4, 8)
```

headstart_utilities *Utilities for outputting characteristics of a datatset used it code.*

Description

These functions are used during the execution of a program. Rather they produce snippets that can be pasted into code, and help the developer avoid some typing.

Usage

```
column_rename_headstart( d, try_snake_case=TRUE )
column_class_headstart( d )
column_value_headstart( x )
```

Arguments

`d`	A `data.frame` to describe.
`try_snake_case`	If `TRUE` column names are attempted to be converted to snake_case.
`x`	A vector to describe.

Value

Prints formatted code to the console.

Author(s)

Will Beasley

Examples

```
column_rename_headstart(datasets::OrchardSprays)
column_rename_headstart(datasets::iris)
column_class_headstart(datasets::OrchardSprays)
column_value_headstart(datasets::OrchardSprays$treatment)
```

match_statistics *Create explicit factor level for missing values.*

Description

Missing values are converted to a factor level. This explicit assignment can reduce the chances that missing values are inadvertantly ignored. It also allows the presence of a missing to become a predictor in models.

Usage

```
match_statistics(d_parent, d_child, join_columns)
```

Arguments

`d_parent` A `data.frame` of the parent table.

`d_child` A `data.frame` of the child table.

`join_columns` The `character` vector of the column names used to join to parent and child tables.

Details

If a nonexistent column is passed to `join_columns`, an error will be thrown naming the violating column name.

More information about the 'parent' and 'child' terminology and concepts can be found in the Hierarchical Database Model Wikipedia entry, among many other sources.

Value

A `numeric` array of the following elements:

- `parent_in_child` The count of parent records found in the child table.
- `parent_not_in_child` The count of parent records *not* found in the child table.
- `parent_na_any` The count of parent records with a `NA` in at least one of the join columns.
- `deadbeat_proportion` The proportion of parent records *not* found in the child table.
- `child_in_parent` The count of child records found in the parent table.
- `child_not_in_parent` The count of child records *not* found in the parent table.
- `child_na_any` The proportion of child records *not* found in the parent table.
- `orphan_proportion` The count of child records with a `NA` in at least one of the join columns.

Note

The `join_columns` parameter is passed directly to `dplyr::semi_join()` and `dplyr::anti_join()`.

Author(s)

Will Beasley

Examples

```
ds_parent <- data.frame(
  parent_id         = 1L:10L,
  letter            = rep(letters[1:5], each=2),
  index             = rep(1:2, times=5),
  dv                = runif(10),
  stringsAsFactors  = FALSE
)
```

```
ds_child <- data.frame(
  child_id         = 101:140,
  parent_id        = c(4, 5, rep(6L:14L, each=4), 15, 16),
  letter           = rep(letters[3:12], each=4),
  index            = rep(1:2, each=2, length.out=40),
  dv               = runif(40),
  stringsAsFactors  = FALSE
)

#Match on one column:
match_statistics(ds_parent, ds_child, join_columns="parent_id")

#Match on two columns:
match_statistics(ds_parent, ds_child, join_columns=c("letter", "index"))
```

`OuhscMunge` *Data manipulation operations frequently used in OUHSC BBMC projects.* `http://www.ouhsc.edu/bbmc/`

Description

Thanks to Funders, including HRSA/ACF D89MC23154

OUHSC CCAN Independent Evaluation of the State of Oklahoma Competitive Maternal, Infant, and Early Childhood Home Visiting (MIECHV) Project., which evaluates MIECHV expansion and enhancement of Evidence-based Home Visitation programs in four Oklahoma counties.

Details

OuhscMunge.

Note

The release version will eventually be available through CRAN by running `install.packages('OuhscMunge')`. The most recent development version is available through GitHub by running `devtools::install_github (repo = 'OuhscBbmc/OuhscMunge')` (make sure devtools is already installed). If you're having trouble with the package, please install the development version. If this doesn't solve your problem, please create a new issue, or email Will.

Author(s)

William Howard Beasley, University of Oklahoma Health Sciences Center, College of Medicine, Dept of Pediatrics, BBMC.

Maintainer: Will Beasley wibeasley@hotmail.com

Examples

```
## Not run:
# Install/update REDCapR with the release version from CRAN.
install.packages('OuhscMunge') #But it's not on CRAN yet.

# Install/update REDCapR with the development version from GitHub
#install.packages('devtools') #Uncomment if `devtools` isn't installed already.
devtools::install_github('OuhscBbmc/OuhscMunge')

## End(Not run)
```

---

replace_nas_with_explicit

Create explicit factor level for missing values.

Description

Missing values are converted to a factor level. This explicit assignment can reduce the chances that missing values are inadvertantly ignored. It also allows the presence of a missing to become a predictor in models.

The function is retained so existing code doesn't break. For new code, consider using `dplyr::coalesce()`. if you don't need to convert the missing code to a factor level.

Usage

```
replace_nas_with_explicit(scores, new_na_label = "Unknown",
  create_factor = FALSE, add_unknown_level = FALSE)
```

Arguments

`scores` An array of values, ideally either factor or character. Required

`new_na_label` The factor label assigned to the missing value. Defaults to `Unknown`.

`create_factor` Converts `scores` into a factor, if it isn't one already. Defaults to `FALSE`.

`add_unknown_level` Should a new factor level be created? (Specify `TRUE` if it already exists.) Defaults to `FALSE`.

Value

An array of values, where the `NA` values are now a factor level, with the label specified by the `new_na_label` value.

Note

The `create_factor` parameter is respected only if `scores` isn't already a factor. Otherwise, levels without any values would be lost.

A `stop` error will be thrown if the operation fails to convert all the `NA` values.

Author(s)

Will Beasley

Examples

```
library(OuhscMunge) #Load the package into the current R session.
missing_indices <- c(3, 6, 8, 25)
# With a character variable:
a <- letters
a[missing_indices] <- NA_character_
a <- OuhscMunge::replace_nas_with_explicit(a)

# With a factor variable:
b <- factor(letters, levels=letters)
b[missing_indices] <- NA_character_
b <- OuhscMunge::replace_nas_with_explicit(b, add_unknown_level=TRUE)
```

replace_with_nas *Convert blank, zero-length values to NAs for a variety of data types.*

Description

Elements of zero-length are converted to NAs. Can force cohersion to an optionally-specified data type.

The function is retained so existing code doesn't break. For new code, consider using dplyr::na_if().

Usage

```
replace_with_nas(x, return_type = NULL)
```

Arguments

x	An array of values. Required
return_type	Data type of returned vector. Optional

Details

If return_type is missing, returned data type will match input. Supports cohersion to integer, numeric, character, logical, and Date vectors.

If return_type=logical, a logical vector will be returned if x contains only blanks and the characters "0" and "1".

Value

An array of values with NAs.

Note

Contact the package author if you'd like the function generalized so that additional values (other that `""`) are converted to `NA`s.

Author(s)

Will Beasley

Examples

```
library(OuhscMunge) #Load the package into the current R session.
replace_with_nas(c("a", "b", "", "d", ""))
replace_with_nas(c("a", "b", "", "d", ""), return_type="character")

replace_with_nas(c(1, 2, "", "", 5), return_type="character")
replace_with_nas(c(1, 2, "", "", 5)) #Equivalent to previous line.
replace_with_nas(c(1, 2, "", "", 5), return_type="integer")
replace_with_nas(c(1, 2, "", "", 5), return_type="numeric")

replace_with_nas(c("2011-02-03", "", "", "2011-02-24"), return_type="Date")
replace_with_nas(c("T", "", "", "F", "FALSE", "", "TRUE"), return_type="logical")
replace_with_nas(c("1", "", "", "0", "0"    , "", "1")   , return_type="logical")
```

`snake_case` *Convert variable names to snake_case*

Description

This function attempts to convert variables to snake_case, even if it's already in snake_case. The important regex lines were posted by Stack Overflow user epost in "Elegant Python function to convert CamelCase to snake_case?".

Usage

```
snake_case(x)
```

Arguments

x	A vector to of names to convert.

Value

A vector of converted names.

Author(s)

Will Beasley

Examples

```
snake_case(colnames(datasets::OrchardSprays))
snake_case(colnames(datasets::iris))
```

Index