Package: y3628 (via r-universe)

February 13, 2025

Title Base package for the y3628 data analysis suite

Version 0.0.3

Description Base functionalities used by other packages in the y3628 suite of data analysis tools. These were initially written for various projects during my years at Michigan.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

biocViews CellBiology, Genetics

Imports dplyr, methods, readr, readxl, rlang, stringr, tools, vctrs

Depends R (>= 4.1)

Suggests tibble, datasets

Config/pak/sysreqs libicu-dev libx11-dev

Repository https://yeyuan98.r-universe.dev

RemoteUrl https://github.com/yeyuan98/y3628

RemoteRef HEAD

RemoteSha dd92c11e2b1087b4549f346ffd06f234cbf3c0cd

Contents

flexTableReader *Read 'table' file with flexible file extension*

Description

Read 'table' file with flexible file extension

Usage

```
flexTableReader(col_names, col_types, skip, ...)
```

Arguments

`col_names`	Column names, character vector
`col_types`	Column types, same as `readr::read_csv()`. See details.
`skip`	Number of lines to skip before reading data
`...`	These dots are for future extensions and must be empty.

Details

Depending on the actual file format provided by the user, column types may not be respected. For example, if the file is excel spreadsheet, then the column types provided is mapped to that supported by the `readxl` package.

Value

Reader function that accepts a file path

Examples

```
# TODO
```

getNormalizedParams *Get specific parameters for a table reader*

Description

Get specific parameters for a table reader

Usage

```
getNormalizedParams(ext, ...)
```

Arguments

`ext`	File extension.
`...`	Parameters, see details.

Details

Currently the following parameters are supported: col_names, col_types, skip. For excel files, col_types is not supported yet.

Value

Named list of normalized parameters.

Examples

```
# Not exported
```

`getTableReader` *Get filetype-specific table reader*

Description

Get filetype-specific table reader

Usage

```
getTableReader(ext)
```

Arguments

`ext`	File extension.

Value

A suitable reader function.

Examples

```
# Not exported.
```

grouper *Grouping of data frame without taking up the ellipsis*

Description

Motivation: the original `dplyr::group_by()` uses ellipsis to allow flexibility in specifying grouping variables. However, sometimes a function might need the ellipsis for other purposes. In those cases, it is desirable to "save" the ellipsis by allowing the user to provide all grouping variables in a single parameter.

Usage

```
grouper(.data, var.group, ...)
```

Arguments

`.data`	Data frame to perform `dplyr::group_by()`.
`var.group`	**Quoted** group variable(s). Multiple variables must be inside `gvars()`. To quote group variables use `rlang::enexpr()` in the caller function.
`...`	Must be empty.

Details

This function is not intended for end users.

Value

Grouped data frame.

Examples

```
# Correct use
f <- function(df, var.group){
    var.group <- rlang::enexpr(var.group)
    return(grouper(df, var.group))
}
# Single group variable
f(mtcars, cyl)
# Multiple group variable
f(mtcars, gvars(cyl,vs))

# Wrong use
f <- function(df, var.group){ return(grouper(df, var.group)) }
# f(mtcars, cyl) will error
```

`groupThenSummarize` *Customizable summary of data frame variables*

Description

Customizable summary of data frame variables

Usage

```
groupThenSummarize(.data, var.group, .fns, ...)
```

Arguments

`.data`	a data frame (extension)
`var.group`	variables to group, either single name or gvars()
`.fns`	named list of summary functions
`...`	<`tidy-select`> Selection of variables to summarize on

Value

a data frame (extension) of summary

Examples

```
# Single group variable
groupThenSummarize(mtcars, cyl, list(m=mean, s=sd), disp:wt)
# Multiple group variable
groupThenSummarize(mtcars, gvars(cyl,vs), list(m=mean, s=sd), disp:wt)
#   cyl == 4 & vs == 0 group will have NA sd values.
#   This is because there is only one row in this group.
mtcars[mtcars$cyl == 4 & mtcars$vs == 0,]
```

`metaRcrd` `metaRcrd` *record-style vector*

Description

This creates a vector where each item is a record of data and metadata fields. Data fields are used for record equality/comparison operations. Metadata fields are conceptually "attributes" attached to the data fields.

Usage

```
metaRcrd(fields, meta.fields, ...)
```

Arguments

`fields`	A named list or data.frame where each row is a record. Names of this list are the field names for the record vector.
`meta.fields`	A character vector giving fields that should be considered as "metadata" fields.
`...`	For future extensions. Must be empty.

Value

An S3 vector of class `y3628_metaRcrd`.

Note

Set operations must use `vctrs` methods (e.g., `vctrs::vec_set_union()`). Base set operations are not generic and hence invalid.

See Also

`vctrs::new_rcrd()`

Examples

```
require(vctrs)
require(tibble)

## Representing metadata of experimental samples

today <- Sys.Date()
dates <- c(today, today, today, today+1) # when
conds <- factor(c("L", "M", "H", "L"), levels = c("L", "M", "H")) # condition
who <- c("me", "me", "me", "Ahri") # personnel
eg1 <- metaRcrd(list(date=dates, condition=conds, personnel=who), "personnel")

dates <- c(today) # when
conds <- factor("L", levels = c("L", "M", "H")) # condition
who <- "Ahri" # personnel
eg2 <- metaRcrd(list(date=dates, condition=conds, personnel=who), "personnel")

### concatenate
eg_full <- vec_c(eg1, eg2) # c() works just fine too
eg_full

### equality/set operation
eg1 == eg2
vec_set_difference(eg1, eg2) # use vec_set_* methods

### comparison/sort
eg1 <= eg2
sort(eg_full)

### use in a tibble
```

```
proj <- c("fancy project 1", "ok project 2") # what project
rec <- list(eg1, eg2) # experiment record for each project
project_record <- tibble(project = proj, record = rec)
project_record
```

`mR_filter` *Filtering metaRcrd objects*

Description

Filtering metaRcrd objects

Usage

```
mR_filter(mRcrd, what)
```

Arguments

`mRcrd`	An S3 vector of class `y3628_metaRcrd`.
`what`	Filtering criteria. Quoted and evaluated to subset the record.

Value

Subsetted `y3628_metaRcrd`.

Examples

```
require(datasets)
my_mtcars <- datasets::mtcars[c("cyl", "hp", "mpg")]
my_mtcars <- metaRcrd(my_mtcars, meta.fields = "mpg")
mR_filter(my_mtcars, mpg > 20)
```

`str_split_summary` *Applying summary function to string splits*

Description

Applying summary function to string splits

Usage

```
str_split_summary(string, pattern, summary)
```

Arguments

`string`	Input vector, passed to `stringr::str_split()`.
`pattern`	Pattern to look for, passed to `stringr::str_split()`.
`summary`	Summary function that will be applied to each split.

Value

Vector same length as input vector

Examples

```
t <- c("0.1,0.3,0.7", "NA", "0.2", "")
str_split_summary(t, ",", \(x) mean(as.numeric(x)))
```

Index