Package ‘csmtools’

October 26, 2016

Type Package

Title R code in use in CSM solutions

Version 1.0.0

Author McClelland Legge <McClelland.Kemp@iriworldwide.com> [aut, cre]

Maintainer <McClelland.Kemp@iriworldwide.com>

Description A collection of helpful functions that are widely or commonly used by the organization.

License file LICENSE

LazyData TRUE

RoxygenNote 5.0.1

Imports data.table,magrittr

Suggests testthat

R topics documented:

`csmtools-package` *csmtools: A collection of useful code*

Description

csmtools is a collection of code that can be used in different projects to help prevent team members from reinventing the wheel for some things that others have already figured out.

Details

Bug reports, additions and enhancement requests are welcomed at: `https://github.com/McClellandLegge/csmtools`

`dt_reduce` *Apply a row-wise Reduce*

Description

Apply a row-wise Reduce

Usage

```
dt_reduce(DT, FUN, ...)
```

Arguments

`DT`	A `data.table`
`FUN`	Any binary function
`...`	Quoted column names from DT

Details

Apply a row-wise reduce for a given function on a set of a data.table's columns. The main advantage of this function is that names can be passed to the function as vectors, eliminating the need to hard code differencing, etc. based on column names. Additionally, the output is specified by the user – often we want to perform a calculation and have vector output, something usaully implemented with an ugly `unlist`.

Value

A vector

Class will vary

Examples

```
library("data.table")
DT <- as.data.table(head(iris))
# basic differencing
dt_reduce(DT, `-`, "Sepal.Length", "Sepal.Width")

# paste columns together row-wise
dt_reduce(DT, paste, colnames(DT))

# calculate the mean
dt_reduce(DT, `+`, "Sepal.Length", "Sepal.Width", "Petal.Length") / nrow(DT)
```

`file_size_filter` *Filter files based on their size*

Description

Filter files based on their size

Usage

```
file_size_filter(x, size = 0, units = "B", include = FALSE)
```

Arguments

`x`	A character vector of filenames
`size`	A numeric vector, the size of the file
`units`	A character vector specifying the units. Options are `B` , `KB`, `MB` and `GB`. Must match the length of `size` if specifying more than one unit.
`include`	A boolean, include files of size `size`?

Details

Filters out the files that are less than (or less than or equal to) the size with units specified. If all files are filtered out, then a character vector of length 0 is returned.

Value

A character vector

Examples

```
x <- list.files(path = Sys.getenv("TEMP"), full.names = TRUE)[1]
file_size_filter(x, size = 1, units = "KB")
```

filter_files *Filter out file names based on criteria*

Description

Filter out file names based on criteria

Usage

```
filter_files(x, size = 0, units = "B", include = FALSE, simplify = TRUE)
```

Arguments

`x`	A character vector of filenames
`size`	A numeric vector, the size of the file
`units`	A character vector specifying the units. Options are `B` , `KB`, `MB` and `GB`. Must match the length of `size` if specifying more than one unit.
`include`	A boolean, include files of size `size`?
`simplify`	A boolean, should we create a data.table from the list of output?

Details

Automatically excludes any non-existant items

Value

A list or a `data.table`

Examples

```
# single file size and unit specification
x <- list.files(path = Sys.getenv("TEMP"), full.names = TRUE)[1:100]
filter_files(x, size = 1, units = "KB")

# multiple specifications
size <- c(0, rep(1, 4))
units <- c("B", "B", "KB", "MB", "GB")
res <- filter_files(x, size, units)
res$filesize <- factor(res$filesize, levels = paste(size, units))
table(res$filesize)
```

floor *A function to extend the functionaliy of the base::floor function*

Description

A function to extend the functionaliy of the base::floor function

Usage

```
floor(x, digits = 0)
```

Arguments

`x`	A numeric
`digits`	The number of digits to the left of the decimal place to round to

Details

Usually used for purchase data when you need to floor the cents

Value

A numeric

Examples

```
x <- 99.9999
floor(x, 2)
# is equivalent to:
base::floor(100 * x)
```

has_hive *Check if the system has hive capabilities*

Description

Check if the system has hive capabilities

Usage

```
has_hive()
```

Details

Checks to see if the hive binaries are in the PATH variable

Value

A boolean

Examples

```
if(has_hive()) {
    print("yes")
} else {
    print("no")
}
```

`hive_datatypes` *Extract the R-datatypes for a hive table*

Description

Extract the R-datatypes for a hive table

Usage

```
hive_datatypes(schema, table_name)
```

Arguments

`schema`	A character string, the name of the hive schema
`table_name`	A character string, the name of the hive table

Details

For now the functions converts all number-y datatypes like integer, float, decimal to `numeric` and both date and string types to `character`.

Value

A `data.table` with columns:

- `name` The column name (`character`)
- `type` The R-datatype (`character`)

`hive_read` *Read a Hive table that is stored as a text file*

Description

Read a Hive table that is stored as a text file

Usage

```
hive_read(x, ...)
```

Arguments

`x`	A character string, the directory path of the hive table to read
`...`	Additional arguments to `fread`

Details

The function allows you to assign your own additional arguments to fread, but it defaults the separator to pipe ("|") and adds to the na.string to recognize the hive default.

Value

A `data.table`

`hread` *A function to perform a read of a hive table*

Description

A function to perform a read of a hive table

Usage

```
hread(table_name, schema, schema_loc, ...)
```

Arguments

`table_name`	A character string, the name of the hive table
`schema`	A character string, the name of the hive schema
`schema_loc`	A character string, the directory path of where the schema is located on the HDFS
`...`	Additional arguments to `hive_read`

Details

Will automatically read all files under the directory after finding the datatypes and column names from the hive metastore. Note that the `schema` can have a different physical location instead of being forced to have the `schema_loc/schema` naming convention.

Value

A `data.table`

Examples

```
## Not run:
schema_loc <- "/mapr/mapr03r/analytic_users/msmck/csm_synd_hive_schemas/csm_syndicated/"
hread("dictionary", "csm_syndicated", schema_loc)

## End(Not run)
```

iri_week *A function to derive the IRI week for the date specified*

Description

A function to derive the IRI week for the date specified

Usage

```
iri_week(x, fmt = "%Y-%m-%d", ...)
```

Arguments

`x`	A date or character string
`fmt`	A date format
`...`	Additional arguments to `as.Date` including further arguments to be passed from or to other methods, including `format` for `as.character` and `as.Date` methods.

Value

A numeric

Examples

```
iri_week(Sys.Date())
iri_week("Dec. 12, 2016", "%b. %d, %Y")
```

is_defined *A function to perform a read of a hive table*

Description

A function to perform a read of a hive table

Usage

```
is_defined(..., .all = FALSE)
```

Arguments

`...`	A list of objects to test if they are null
`.all`	Boolean, should we return the tests for each individual element?

Details

Good for testing an input(s) to a function when it might be NULL

Examples

```
foo <- function(x = NULL, y = NULL) {
  if (is_defined(x, y)) {
    return(paste0(x, y))
  } else if (is_defined(x)) {
    return(x)
  } else if (any(is_defined(x, y, .all = TRUE))) {
    return("one is not null")
  }
}
foo(x = 1)
foo(x = 1, y = 2)
foo(y = 2)
```

`ninja_load` *Load packages silently*

Description

Load packages silently

Usage

```
ninja_load(...)
```

Arguments

`...`	The quoted names of the packages you wish to load with deadly silence

Examples

```
# load some notoriously loud packages
## Not run:
ninja_load("data.table", "bit64")

## End(Not run)
```

Index