# Package 'csmtools'

November 16, 2016

**Type** Package

**Title** R code in use in CSM solutions

**Version** 1.3.0

**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,covr

## 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

---

`dread` *Fread all files in a directory*

---

## Description

Fread all files in a directory

## Usage

```
dread(dir_path, ...)
```

## Arguments

| | |
|---|---|
| `dir_path` | A valid directory path |
| `...` | Additional arguments to `fread` |

## Value

A data table

dreads *A function to do multiple directory reads*

## Description

A function to do multiple directory reads

## Usage

```
dreads(envs, pattern, colnames = NULL, filters = NULL, combine_dir = TRUE,
  combine_env = TRUE, ...)
```

## Arguments

| | |
|---|---|
| envs | A named list<br>• dir_path A character string sepecifying the path of the directory<br>• hive A boolean, is this a hive table or a regular file<br>• ext Optional, if hive is FALSE, specify if only files with a certain extension should be read, e.g. .dat or .csv. |
| pattern | A regex character string. Only file names which match the regular expression will be returned. |
| colnames | Any column names for the tables being read in. Note these must be universal so if any of the tables differ, leave this NULL and turn off the combine_* actions as appropriate |
| filters | Any regex filters to apply, no negation works at this time. Can be passed as a list or vector |
| combine_dir | A Boolean, collapse the list of data.tables read in from each env into a data.table, keeping the file/table names in the table_name column? |
| combine_env | A Boolean, collapse the list of data.tables from the different envs into a data.table, keeping the names of the envs in the env_name column? |
| ... | Additional arguments to fread |

## Value

A list of data.tables or a data.table

## See Also

dread

## Examples

```
## Not run:
# read all walmart trip types from three different environments
# some are hive, some are regular files

wlm_01_envs <- list(
  prd = list(dir_path = file.path(home, "csm_synd_hive_schemas/csm_syndicated")
             , hive = TRUE)
  , dev = list(dir_path = file.path(home, "csm_synd_hive_schemas/csm_syndicated_dev")
```

```
                , hive = TRUE)
  , leg = list(dir_path = file.path(home, "dev/trip_typing/legacy_walmart/artifacts")
                , hive = FALSE
                , ext = ".dat")
)

ft <- paste0(1929:1940, collapse = "|")

dreads(wlm_01_envs, "wm_triptypes", filters = ft)

## End(Not run)
```

---

dt_compare *Merge and compare columns of data.frames (data.tables)*

---

## Description

Merge and compare columns of data.frames (data.tables)

## Usage

```
dt_compare(x, y, compare = NULL, func = `-`, precision = 6, ...)
```

## Arguments

| | |
|---|---|
| x | A data.frame |
| y | A data.frame |
| compare | A character string or vector of shared column names |
| func | A binary function to compare the columns with, should be appropriate for the datatypes of the columns |
| precision | The precision of the comparison, is the `digits` argument to the `round` function |
| ... | Any arguments to the `merge` function |

## Value

A data.frame

## Examples

```
x <- iris[1:50,]
y <- iris[1:60,]
x$id <- seq(nrow(x))
y$id <- seq(nrow(y))
y$Sepal.Width = y$Sepal.Width + rnorm(n = nrow(y))

# can specify any arguments to 'merge'
res <- dt_compare(x, y, compare = c("Sepal.Width", "Sepal.Length"), by = "id", all.y = TRUE)
```

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$min_file_size <- factor(res$min_file_size, levels = paste(size, units))
table(res$min_file_size)
```

---

## 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)
```

---

## gg_color_hue *Emulate the default ggplot2 color palette*

---

### Description

Emulate the default ggplot2 color palette

### Usage

```
gg_color_hue(n, rgb = TRUE)
```

### Arguments

| | |
|---|---|
| n | The number of colors to produce |
| rgb | A Boolean, should a rgb(<r>, <g>, <b>) character string be returned instead of the hex values? |

### Details

An adapted function from John Colby's http://stackoverflow.com/a/8197703/3034614 on how to emulate the ggplot2 default color palette, which is just equal spacing on the color wheel.

### Value

A character vector

### Examples

```
gg_color_hue(5)
gg_color_hue(5, rgb = FALSE)
```

---

## 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)
```

## make_color *A function to convert colors to their hex values*

### Description

A function to convert colors to their hex values

### Usage

```
make_color(...)
```

### Arguments

| | |
|---|---|
| x | A (possibly) mixed-type vector |

### Details

A list is returned instead of a vector to avoid the coercion of a Boolean value to a character one

### Value

A list of hex values or Boolean FALSE when the element cannot be interpreted as a color

### Examples

```
make_color(NA, "black", "blackk", 5, "#00", "#000000", "rgb(1, 1, 1, 0.5)")
```

---

## make_compare_names *Make names for comparing two data sets*

---

### Description

Make names for comparing two data sets

### Usage

```
make_compare_names(compare, suffixes = c(".x", ".y"), sep = "_")
```

### Arguments

| | |
|---|---|
| compare | The column names to compare |
| suffixes | The suffixes for each set to use |
| sep | The separator between the names, the sep argument to the paste function |

### Value

A character vector

### Examples

```
make_compare_names(compare = c("dollars", "units"), suffixes = c(".hive", ".sas"))
```

---

## 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)
```

---

paman *Element-wise paste of a matrix using the column names*

---

## Description

Element-wise paste of a matrix using the column names

## Usage

```
paman(x, ...)
```

## Arguments

| | |
|---|---|
| x | A matrix, or object able to be coerced to a matric with non-null dimnames |
| ... | Arguments to paste |

## Details

Works just like a normal paste0 function except the input is expected to be a matrix and the output will likewise be a matrix. Its a "paste" for a "matrix" "element" = "pa" + "m" + "el" = "pamel". Helpful for parsing hovertext for 3D plotly objects.

## Value

A matrix

## See Also

pamat pamel

## Examples

```
m <- matrix(runif(9), nrow = 3, dimnames = list(LETTERS[1:3], LETTERS[4:6]))

paman(m, "From: ", y, " to: ", x)
paman(m, "From: ", x, " to: ", y)
```

## pamat *Element-wise paste of two (or more) matrices with a separator*

### Description

Element-wise paste of two (or more) matrices with a separator

### Usage

```
pamat(..., sep = " ")
```

### Arguments

| | |
|---|---|
| ... | Matrix objects |
| sep | A character string, the separator between the element-wise paste |

### Details

The matrices will be pasted together in the order in which they are specified and one separator will be shared used. Its a "paste" for a "matrix" "element" = "pa" + "m" + "el" = "pamel". Helpful for parsing hovertext for 3D plotly objects.

### Value

A matrix

### See Also

pamel paman

### Examples

```
caps <- matrix(LETTERS[1:9], nrow = 3)
lows <- matrix(letters[1:9], nrow = 3)
pamat(caps, lows, sep = " -> ")
```

## pamel *Element-wise paste of a matrix*

### Description

Element-wise paste of a matrix

### Usage

```
pamel(n, ...)
```

### Arguments

| | |
|---|---|
| n | The number of rows of the incoming/outgoing matrix |
| ... | One or more R objects, to be converted to character vectors. Expecting that this contains the matrix object, but that is not strictly enforced. |

### Details

Works just like a normal paste0 function except the input is expected to be a matrix and the output will likewise be a matrix. Its a "paste" for a "matrix" "element" = "pa" + "m" + "el" = "pamel". Helpful for parsing hovertext for 3D plotly objects.

### Value

A matrix

### See Also

pamat paman

### Examples

```
m <- matrix(runif(9), nrow = 3)
pamel(nrow(m), "Value: ", round(m, 4), " units")
```

---

## preview_palatte *Preview your hex color palatte*

---

### Description

Preview your hex color palatte

### Usage

```
preview_palatte(x)
```

### Arguments

| | |
|---|---|
| x | A vector of hex colors |

### Details

Plots a simple image with swaths of the colors specified in the palatte, in the order in which they are specified

### Examples

```
colfunc <- colorRampPalette(c("white", "dodgerblue"))
my_cols <- colfunc(20)
preview_palatte(my_cols)
```

# Index