Package: Ipaper (via r-universe)

May 8, 2026

Type Package

Title Collection of personal practical R functions

Version 0.1.11

Description Awesome functions in R.

License GPL-3

Encoding UTF-8

LazyData TRUE

Depends R (>= 3.4), magrittr

Imports jsonlite, magrittr, purrr, rlang, progress, lubridate, stringr, data.table, fst, qs2, openxlsx2, readxl, graphics, svglite, methods, parallel, doParallel, foreach, iterators, plyr, dplyr, clipr, glue, matrixStats, rstudioapi, remotes, usethis, crayon, showtext, sysfonts, zeallot

Suggests ggplot2, testthat (>= 3.1.0), knitr, rmarkdown, Cairo, sp, rtrend, Cairo, grid, gridExtra

VignetteBuilder knitr

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev git make libharfbuzz-dev libgit2-dev libicu-dev libpng-dev libuv1-dev libssl-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-05-07 14:58:50 UTC

RemoteUrl https://github.com/rpkgs/Ipaper

RemoteRef HEAD

RemoteSha 872c4005a0b1a227424f8094cb8cafdd9aefd3cc

Contents

add_dn 3
apply_3d 3

apply_col . . . 4
array_3dTo2d . . . 5
array2dt . . . 6
char2script . . . 7
chunk . . . 7
clamp . . . 8
cut_plevels . . . 8
date_ym . . . 9
dir.show . . . 9
dir2 . . . 10
dt_day2year . . . 10
dt_ddply . . . 11
dt_tools . . . 12
export . . . 13
file_ext . . . 13
file_size . . . 14
flipud . . . 14
fprintf . . . 14
fread_dir . . . 15
getwd_clip . . . 15
git_tools . . . 16
github . . . 16
group_map2 . . . 16
grouped_list . . . 17
ifelse2 . . . 17
import . . . 18
install_gitee . . . 18
key_blind . . . 19
killCluster . . . 19
label_tag . . . 19
listk . . . 20
llply . . . 20
melt_list . . . 21
mkdir . . . 22
obj_size . . . 22
parse_deg . . . 23
path.mnt . . . 23
pdf_view . . . 23
print.data.table . . . 24
pro_map . . . 24
progressively . . . 25
read_excel . . . 26
read_xlsx . . . 26
read_xlsx2list . . . 26
runningId . . . 27
set_dim . . . 27
set_jupyter . . . 28
slope_arr . . . 28

which.na 29
write_fig 29
write_sheet 31

Index **33**

`add_dn` *Add n-day flag*

Description

To aggregated data into n-day (e.g. 8-day, 16-day) like MODIS product, a n-day flag is need.

Usage

```
add_dn(d, days = 8)
```

Arguments

`d`	data.frame or data.table
`days`	Integer number or vector, can't have duplicated value.

Examples

```
date <- seq.Date(as.Date("2010-01-01"), as.Date("2010-12-31"), by = "day")
d <- data.frame(date)
dnew <- add_dn(d, days = c(8, 16))
```

`apply_3d` *apply function for 3d array*

Description

NA values will be removed automatically

Usage

```
apply_3d(
  array,
  dim = 3,
  FUN = rowMeans2,
  by = NULL,
  scale = 1,
  na.rm = TRUE,
  ...
)
```

Arguments

array	A 3d array
dim	giving the subscripts to split up data by.
FUN	function, should only be row applied function, e.g. matrixStats::rowMeans2, matrixStats::rowMins, matrixStats::rowRanges. Because 3d array will be convert to matrix first, with the aggregated dim in the last dimension.
by	• If not provided (NULL), the aggregated dim will be disappear. For example, daily precipitation [nrow, ncol, 31-days] aggregate into monthly [nrow, ncol]. • If provided, by should be equal to the aggregated dim. For example, daily precipitation [nrow, ncol, 365-days] aggregate into monthly [nrow, ncol, 12-months]. In that situation, by should be equal to 365, and be format(date, '%Y%m').
scale	in the same length of by, or a const value, value_returned = FUN(x)*scale. This parameter is designed for converting monthly to yearly, meanwhile multiply days in month. Currently, same group should have the same scale factor. Otherwise, only the first is used.

See Also

apply_row matrixStats::rowRanges

Examples

```
set.seed(1)
size <- c(10, 8, 31)
arr <- array(rnorm(10 * 8 * 31), dim = size)

by <- c(rep(1, 10), rep(2, 21))
r2 <- apply_3d(arr, 3, by = by, FUN = rowMeans)

## Not run:
arr_yearly <- apply_3d(arr, by = year(dates), scale = days_in_month(dates))

## End(Not run)
```

---

apply_col *apply_col*

Description

- apply_col: aggregate by col, return a [ngrp, ncol] matrix
- apply_row: aggregate by row, return a [nrow, ngrp] matrix

Usage

```
apply_col(mat, by, FUN = colMeans2, scale = 1, ...)

apply_row(mat, by, FUN = rowMeans2, scale = 1, ...)
```

Arguments

mat	matrix, [nrow, ncol]
by	integer vector, with the dim of [ntime]
scale	in the same length of by, or a const value, value_returned = FUN(x)*scale. This parameter is designed for converting monthly to yearly, meanwhile multiply days in month. Currently, same group should have the same scale factor. Otherwise, only the first is used.

Details

For example, setting the dimension of mat is [ngrid, ntime], if you want to aggregate by time, apply_row should be used here; if you want to aggregate by region (grids), apply_col should be used.

Note

This function also suits for big.matrix object.

Examples

```
mat <- matrix(rnorm(4 * 6), 4, 6)
mat_bycol <- apply_col(mat, c(1, 1, 2, 2), colMeans)
mat_byrow <- apply_row(mat, c(1, 1, 2, 2, 3, 3), rowMeans)
```

---

array_3dTo2d *array_3dTo2d*

Description

array_3dTo2d

Usage

```
array_3dTo2d(array, I_grid = NULL)

array_2dTo3d(array, I_grid = NULL, dim)
```

Arguments

array	array with the dimension of [nlon, nlat, ntime]
I_grid	subindex of [nrow, ncol]
dim	[nrow, ncol]

Value

[nlat*nlon, ntime]

`array2dt` *Convert array to data.table*

Description

Convert array to data.table

Usage

```
array2dt(arr, dimnames)

dt2array(dt, value_col = "value")
```

Arguments

`arr`	input array
`dimnames`	list of dimension names, e.g., `dimnames(arr)`

Value

data.table

Examples

```
library(Ipaper)

arr <- array(1:6,
  dim = c(2, 3),
  dimnames = list(
    site = c("A", "B"),
    date = c("d1", "d2", "d3")
    # var = c("v1", "v2", "v3", "v4")
  )
)

# array -> dt
dt <- array2dt(arr, dimnames(arr))
print(dt)

# dt -> array
arr2 <- dt2array(dt)
all.equal(arr, arr2)
```

char2script *generate R script of character vector*

Description

generate R script of character vector

Usage

```
char2script(x, collapse = "\"", verbose = TRUE)

code_ChrVec(x, collapse = "\"", verbose = TRUE)
```

Arguments

x	character vector, data.frame or list.
collapse	an optional character string to separate the results. Not NA_character_.

chunk *chunk*

Description

chunk

Usage

```
chunk(x, nchunk = 6)
```

References

https://stackoverflow.com/questions/3318333/split-a-vector-into-chunks-in-r

clamp *clamp*

Description

clamp values in the range of `lims`

Usage

```
clamp(x, lims = c(0, 1), fill.na = FALSE)

clamp_min(x, value = 0)

clamp_max(x, value = 0)
```

Arguments

`x`	Numeric vector
`lims`	limits
`fill.na`	If true, values of lims are set to NA; else, values are just constrained in the range of `lims`.

Examples

```
clamp(1:10, lims = c(4, 7), fill.na = TRUE)
```

cut_plevels *cut_plevels*

Description

cut_plevels

Usage

```
cut_plevels(x, pvalue = c(0.01, 0.05, 0.1), verbose = FALSE)
```

Arguments

`x`	numeric vector
`pvalue`	p <= x%, means its significant at x% level

Examples

```
x <- c(-0.09, -0.4, 0.04, 0.15)
cut_plevels(x, verbose = TRUE)
```

date_ym *The corresponding date of the first day of month*

Description

The corresponding date of the first day of month

Usage

```
date_ym(date)

date_y(date)

date_yj(year, doy)

date_ydn(year, dn, delta = 8)
```

dir.show *Open directory in Explorer*

Description

open assign path in windows explorer, and default path is current directory. This function is only designed for windows system.

Usage

```
dir.show(path = getwd(), verbose = FALSE)
```

Arguments

path	the path you want to open

dir2 *dir2*

Description

dir2

Usage

```
dir2(path = ".", pattern = NULL, full.names = TRUE, ...)
```

Arguments

path	a character vector of full path names; the default corresponds to the working directory, getwd(). Tilde expansion (see path.expand) is performed. Missing values will be ignored. Elements with a marked encoding will be converted to the native encoding (and if that fails, considered non-existent).
pattern	an optional regular expression. Only file names which match the regular expression will be returned.
full.names	a logical value. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned.
...	other parameters to base::dir()

See Also

base::dir()

---

dt_day2year *dt_day2year*

Description

dt_day2year

Usage

```
dt_day2year(
  dat,
  nmiss_day_per_mon = 3,
  nmiss_MonPerYear = 0,
  nmin_year = 55,
  ...
)
```

Arguments

dat	A data.table, at least with the columns of c("site", "date", "value")

dt_ddply *Split* `data.table`*, apply function, and return results in a* `data.table`*.*

Description

For each subset of a `data.table`, apply function then combine results into a `data.table`.

Usage

```
dt_ddply(
  .data,
  .variables,
  .f = NULL,
  ...,
  .progress = "none",
  .drop = TRUE,
  .parallel = FALSE
)

dt_ldply(
  .data,
  .f = NULL,
  ...,
  .progress = "none",
  .parallel = FALSE,
  .id = NA
)

dt_dlply(
  .data,
  .variables,
  .f = NULL,
  ...,
  .progress = "none",
  .drop = TRUE,
  .parallel = FALSE
)
```

Arguments

`.data`	data frame to be processed
`.variables`	variables to split data frame by, as `as.quoted` variables, a formula or character vector
`.f`	A function, specified in one of the following ways: • A named function, e.g. `mean`. • An anonymous function, e.g. `\(x) x + 1` or `function(x) x + 1`.

- A formula, e.g. `~ .x + 1`. Use `.x` to refer to the first argument. No longer recommended.
- A string, integer, or list, e.g. `"idx"`, `1`, or `list("idx", 1)` which are shorthand for `\(x) pluck(x, "idx")`, `\(x) pluck(x, 1)`, and `\(x) pluck(x, "idx", 1)` respectively. Optionally supply `.default` to set a default value if the indexed element is `NULL` or does not exist.

[Experimental]

Wrap a function with `in_parallel()` to declare that it should be performed in parallel. See `in_parallel()` for more details. Use of `...` is not permitted in this context.

`...`	other arguments passed on to `.fun`
`.progress`	name of the progress bar to use, see `create_progress_bar`
`.drop`	should combinations of variables that do not appear in the input data be preserved (FALSE) or dropped (TRUE, default)
`.parallel`	if `TRUE`, apply function in parallel, using parallel backend provided by foreach

Examples

```
dt <- data.table(x = 1:10, y = 1:5)
dt_dlply(dt, .(y), ~.[which.max(x)])
dt_ddply(dt, .(y), ~ top_n(., 1, x))
```

`dt_tools` *data.frame manipulating function by* `dplyr::across`

Description

data.frame manipulating function by `dplyr::across`

Usage

```
dt_round(d, digits = 2)

dt_chr2num(d, fun = as.numeric)
```

export *export data*

Description

export data

Usage

```
export(x, path, ..., nthreads = 6)
```

Details

Support rda, rds, fst, csv, qs

file_ext *get file name and extension*

Description

get file name and extension

Usage

```
file_ext(file)

file_name(file)
```

Arguments

file	file path

Examples

```
file_name("./a.pdf")
file_ext("./a.pdf")
```

file_size *file_size*

Description

file_size

Usage

```
file_size(file)
```

Arguments

file	file path

flipud *flipud and fliplr*

Description

flipud and fliplr

Usage

```
flipud(x, ...)

fliplr(x)
```

fprintf *fprintf Print sprintf result into console, just like C style fprintf function*

Description

fprintf Print sprintf result into console, just like C style fprintf function

Usage

```
fprintf(fmt, ...)
```

Arguments

fmt	a character vector of format strings, each of up to 8192 bytes.
...	other parameters will be passed to sprintf

Examples

```
cat(fprintf("%s\n", "Hello phenofit!"))
```

fread_dir *fread_dir*

Description

fread_dir

Usage

```
fread_dir(indir, pattern = "*.csv", ..., .progress = "text", list2df = FALSE)
```

Arguments

...	others to data.table::fread()

getwd_clip *GET or SET working directory*

Description

- getwd_clip: get directory path in clipboard, same as getwd function
- setwd_clip: set directory path in clipboard, same as setwd function

Usage

```
getwd_clip()

setwd_clip()
```

Note

Only works in windows

References

1. https://stackoverflow.com/questions/10959521/how-to-write-to-clipboard-on-ubuntu-linux-in-r

Examples

```
## Not run:
getwd_clip()
setwd_clip()
dir.show()

## End(Not run)
```

git_tools *git tools*

Description

git tools

Usage

```
git_push(f = FALSE)

git_commit_amend()

git_commit(title = Sys.time())

git_set_remote(url)
```

github *github*

Description

github

Usage

```
github(path = getwd())
```

group_map2 *grouped_list*

Description

grouped_list

Usage

```
group_map2(df, .f, result.name = "data", ..., .keep = FALSE, .progress = FALSE)
```

grouped_list *grouped_list*

Description

grouped_list

Usage

```
grouped_list(data, group)
```

Arguments

data	A list object, length(data) = nrow(group), with elements in the same order as group rows
group	A data.frame or data.table object

ifelse2 *ifelse2*

Description

ternary operator just like java test ? yes : no

Usage

```
ifelse2(test, yes, no)
```

Arguments

test	an object which can be coerced to logical mode.
yes	return values for true elements of test.
no	return values for false elements of test.

Examples

```
x <- ifelse2(TRUE, 1:4, 1:10)
```

import *import data to R*

Description

import data to R

Usage

```
import(path, ..., nthreads = 6)
```

Arguments

nthreads	The number of threads to use when reading data (the initial value is 1L). When TBB is not available, values greater than 1 emit a warning and fall back to 1.

Details

Support rda, rds, fst, csv, qs2

install_gitee *Attempts to install a package directly from gitee.*

Description

Attempts to install a package directly from gitee.

Usage

```
install_gitee(repo)
```

Arguments

Repository	address in the format username/repo[/subdir][@ref|#pull]. Alternatively, you can specify subdir and/or ref using the respective parameters (see below); if both is specified, the values in repo take precedence.

Examples

```
## Not run:
install_gitee("adv-r/Ipaper")

## End(Not run)
```

key_blind *blind shortcuts to rstudio addin*

Description

blind shortcuts to rstudio addin

Usage

```
key_blind()
```

killCluster *killCluster*

Description

killCluster

Usage

```
killCluster()
```

label_tag *label_tag*

Description

label_tag

Usage

```
label_tag(labels, tag = TRUE, expression = TRUE, letter_begin = 1)

char2expr(labels)
```

Arguments

labels	character vector or expression vector
tag	boolean

Examples

```
label_tag(1:5)
char2expr(1:5)
```

listk *listk*

Description

listk

Usage

```
listk(...)
```

Arguments

...	objects, possibly named.

Examples

```
a <- 1
b <- 1:2
c <- 1:3
l1 <- listk(a, b, c)
l2 <- listk(a, b, c = 1:3)
l3 <- listk(a = 1, b = 1:2, c = 1:3)
l4 <- listk(1, 1:2, c)
```

llply *plyr function in purrr style*

Description

plyr function in purrr style

Usage

```
llply(.data, .f = NULL, .progress = "none", .parallel = FALSE, ...)

ldply(.data, .f = NULL, ...)

laply(.data, .f = NULL, ...)

map_simplify(.data, .f = NULL, ...)
```

Arguments

.data	list to be processed
.progress	name of the progress bar to use, see create_progress_bar
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
...	other arguments passed on to .f

References

1. https://github.com/TylerGrantSmith/purrrgress

Examples

```
x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE, FALSE, FALSE, TRUE))
llply(x, mean, .progress = "text")
llply(x, ~ mean(.x), .progress = TRUE)
llply(x, quantile, probs = 1:3 / 4)
```

`melt_list` *melt_list*

Description

melt_list

Usage

```
melt_list(list, ..., na.rm = TRUE, str2num = TRUE, str2factor = TRUE)

melt_tree(x, names, ...)
```

Arguments

`list`	A list object, with the same colnames data.frame in every element.
`...`	other parameters to melt
`na.rm`	Boolean
`var.name`	vector of id variables. Can be integer (variable position) or string (variable name). If blank, will use all non-measured variables.

References

1. https://stackoverflow.com/questions/15673550/why-is-rbindlist-better-than-rbind

Examples

```
# data.frame
df <- data.frame(year = 2010, day = 1:3, month = 1, site = "A")
l <- list(a = df, b = df)

melt_list(l, 'id') %>% as_tibble()
melt_list(l, 'id' = 1) %>% as_tibble()
melt_list(set_names(l, NULL), 'id') %>% as_tibble()

# data.table
df <- data.table::data.table(year = 2010, day = 1:3, month = 1, site = "A")
```

```
l <- list(a = df, b = df)
melt_list(l, "id")
melt_list(l, id = c("a", "b"))
# int `id` is much more efficient
melt_list(l, id = c(1, 2))
```

`mkdir` *mkdir*

Description

mkdir

Usage

```
mkdir(path)

file_mv(files, outdir)

file_cp(files, outdir)

check_dir(path)
```

Arguments

`path`	character vectors

`obj_size` *obj_size*

Description

Get object size in `unit`

Usage

```
obj_size(obj, unit = "Mb")

obj.size(obj, unit = "Mb")
```

Arguments

`obj`	Object
`unit`	"Kb", "Mb" or "Gb"

Examples

```
obj_size(1:100)
```

parse_deg *parse_deg*

Description

- *: zero or more
- +: one or more
- ?: zero or one

Usage

```
parse_deg(xs)
```

path.mnt *convert windows path to wsl*

Description

In windows system, conversion will not be applied.

Usage

```
path.mnt(path)

path_mnt(path)
```

Arguments

path	character vector of file paths.

pdf_view *pdf_view*

Description

pdf_view

Usage

```
pdf_view(file, ...)

SumatraPDF(path = getwd(), verbose = FALSE)

evince(path = getwd(), verbose = FALSE)
```

Arguments

file	the path of pdf file

Note

not work in wsl

---

print.data.table *print.data.table*

Description

print.data.table

Usage

```
## S3 method for class 'data.table'
print(d, n = NULL, ..., maxrows = 1e+06)
```

Examples

```
d <- data.table(1:100)
options(datatable.print.nrow = 100)
print(d)
```

---

pro_map *Modified purrr functions with progress bar*

Description

Modified purrr functions with progress bar

Usage

```
pro_map(.x, .f, ..., .progress = FALSE)
```

Arguments

.x	A list or atomic vector.
.f	A function, specified in one of the following ways: • A named function, e.g. mean. • An anonymous function, e.g. \(x) x + 1 or function(x) x + 1. • A formula, e.g. ~ .x + 1. Use .x to refer to the first argument. No longer recommended.

- A string, integer, or list, e.g. `"idx"`, `1`, or `list("idx", 1)` which are shorthand for `\(x) pluck(x, "idx")`, `\(x) pluck(x, 1)`, and `\(x) pluck(x, "idx", 1)` respectively. Optionally supply `.default` to set a default value if the indexed element is `NULL` or does not exist.

[Experimental]

Wrap a function with `in_parallel()` to declare that it should be performed in parallel. See `in_parallel()` for more details. Use of `...` is not permitted in this context.

`...` Additional arguments passed on to the mapped function.

We now generally recommend against using `...` to pass additional (constant) arguments to `.f`. Instead use a shorthand anonymous function:

```
# Instead of
x |> map(f, 1, 2, collapse = ",")
# do:
x |> map(\(x) f(x, 1, 2, collapse = ","))
```

This makes it easier to understand which arguments belong to which function and will tend to yield better error messages.

`.progress` Whether to show a progress bar. Use `TRUE` to turn on a basic progress bar, use a string to give it a name, or see progress_bars for more details.

`progressively` *Helper function for generating progress bar functions*

Description

Helper function for generating progress bar functions

Usage

```
progressively(
  .mapper,
  .farg = NULL,
  .xarg = NULL,
  .yarg = NULL,
  .larg = NULL,
  .atarg = NULL,
  .parg = NULL
)
```

read_excel *read csv or xls/xlsx file*

Description

read csv or xls/xlsx file

Usage

```
read_excel(f, ...)
```

Arguments

...	others to fun, one of data.table::fread(), readxl::read_xls(), read_xlsx()

read_xlsx *read_xlsx*

Description

read_xlsx

Usage

```
read_xlsx(file, sheet = 1, ...)
```

read_xlsx2list *read_xlsx2list*

Description

If excel file hava many sheets, this function also works.

Usage

```
read_xlsx2list(file, ...)
```

Arguments

file	xlsx or xls file path
...	other parameters to readxl::read_excel()

runningId *print the running ID in the console*

Description

print the running ID in the console

Usage

```
runningId(i, step = 1, N, prefix = "")
```

Arguments

i	the running Id.
step	how long of print step.
N	The number of total iteration
prefix	prefix string

Examples

```
for (i in 1:10) {
  runningId(i, prefix = "phenofit")
}
```

set_dim *Set dimensions of an Object*

Description

Set dimensions of an Object

Usage

```
set_dim(x, dim)
```

Arguments

x	an R object, for example a matrix, array or data frame.

See Also

base::dim

Examples

```
x <- 1:12
set_dim(x, c(3, 4))
```

set_jupyter *set_jupyter*

Description

set_jupyter

Usage

```
set_jupyter(width = 10, height = 6, res = 200)

set_dirRoot(verbose = TRUE, dir = getwd())
```

References

1. https://stackoverflow.com/questions/42729049/how-to-change-the-size-of-r-plots-in-jupyter

slope_arr *slope_arr*

Description

slope_arr

Usage

```
slope_arr(arr, FUN = rtrend::slope_mk, return.list = FALSE)
```

Arguments

arr	a matrix ([ngrid, ntime]) or array ([nlon, nlat, ntime]).
FUN	slope functions, see rtrend::slope_mk().
return.list	boolean, • TRUE: list(slope, pvalue) will be return • FALSE: a array, with the dim of [nx, ny, 2].

Value

t, A 3d array, with the dim of [nx, ny, 2].

- t[,,1]: slope
- t[,,2]: pvalue

which.na *which functions*

Description

which functions

Usage

```
which.na(x)

which.notna(x)

which.isnull(x)

which.null(x)

which.notnull(x)

which.notempty(x)

which.empty(x)

which.dup(x)
```

Arguments

x	numeric vector

write_fig *write figures into disk*

Description

write figure to pdf, tif, png, jpg, svg, or emf, according to file suffix.

Usage

```
write_fig(
  p,
  file = "Rplot.pdf",
  width = 10,
  height = 5,
  devices = NULL,
  res = 300,
```

```
  show = TRUE,
  use.cairo_pdf = TRUE,
  use.file_show = FALSE
)
```

Arguments

p	could be one of grid, ggplot or plot expression
file	file path of output figure
width	the width of the device in inches.
height	the height of the device in inches.
devices	can be c("pdf", "tif", "tiff", "png", "jpg", "svg", "emf"). If not specified, devices will be determined according to the postpix of file. The default type is pdf.
res	The nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for units other than the default. If not specified, taken as 72 ppi to set the size of text and line widths.
show	Boolean. Whether show file after finished writing?
use.cairo_pdf	This parameter is for pdf type. whether to use grDevices::cairo_pdf? cairo_pdf supports self defined font, but can not create multiple page pdf.
use.file_show	boolean. If true, file.show will be used mandatorily.

See Also

grDevices::cairo_pdf(), grDevices::png(), Cairo::Cairo()

Examples

```
## Not run:
library(ggplot2)
p <- ggplot(mpg, aes(class, hwy))
p1 <- p + geom_boxplot2()

## ggplot version
write_fig(p1, "Fig. 1. ggplot.pdf", show = TRUE)
# pdf_view("Fig. 1. ggplot.pdf")
write_fig(p1, "Fig. 1. ggplot", show = TRUE)
write_fig(p1, "Fig. 1. ggplot.pdf", show = TRUE,
    devices = c("jpg", "png", "svg", "pdf", "tif", "emf"))

## lattice
x <- seq(pi/4, 5 * pi, length.out = 100)
y <- seq(pi/4, 5 * pi, length.out = 100)
r <- as.vector(sqrt(outer(x^2, y^2, "+")))
grid <- expand.grid(x=x, y=y)
grid$z <- cos(r^2) * exp(-r/(pi^3))
p <- levelplot(z~x*y, grid, cuts = 50, scales=list(log="e"), xlab="",
          ylab="", main="Weird Function", sub="with log scales",
          colorkey = FALSE, region = TRUE)
```

```
write_fig(p, "fig_lattice", show = TRUE)

## grid expression
g <- grid::circleGrob()
write_fig(g, "fig_grid", show = TRUE)

## R expression
write_fig({
    rx <- range(x <- 10*1:nrow(volcano))
    ry <- range(y <- 10*1:ncol(volcano))
    ry <- ry + c(-1, 1) * (diff(rx) - diff(ry))/2
    tcol <- terrain.colors(12)
    # par(opar);
    # opar <- par(pty = "s", bg = "lightcyan")
    plot(x = 0, y = 0, type = "n", xlim = rx, ylim = ry, xlab = "", ylab = "")
    u <- par("usr")
    rect(u[1], u[3], u[2], u[4], col = tcol[8], border = "red")
    contour(x, y, volcano, col = tcol[2], lty = "solid", add = TRUE,
            vfont = c("sans serif", "plain"))
    title("A Topographic Map of Maunga Whau", font = 4)
    abline(h = 200*0:4, v = 200*0:4, col = "lightgray", lty = 2, lwd = 0.1)
}, "fig_expr.pdf")

# quote expression also works
expr = quote({
    rx <- range(x <- 10*1:nrow(volcano))
    ry <- range(y <- 10*1:ncol(volcano))
    ry <- ry + c(-1, 1) * (diff(rx) - diff(ry))/2
    tcol <- terrain.colors(12)
    # par(opar);
    # opar <- par(pty = "s", bg = "lightcyan")
    plot(x = 0, y = 0, type = "n", xlim = rx, ylim = ry, xlab = "", ylab = "")
    u <- par("usr")
    rect(u[1], u[3], u[2], u[4], col = tcol[8], border = "red")
    contour(x, y, volcano, col = tcol[2], lty = "solid", add = TRUE,
            vfont = c("sans serif", "plain"))
    title("A Topographic Map of Maunga Whau", font = 4)
    abline(h = 200*0:4, v = 200*0:4, col = "lightgray", lty = 2, lwd = 0.1)
})
write_fig(expr, "fig_expr2.pdf")

## End(Not run)
```

`write_sheet` *write_sheet*

Description

Write a single data frame into one sheet of a Workbook. `file` can be a `wbWorkbook` object or a file path string. When a file path is given, the workbook is loaded (or created if the file does not exist), the sheet is written, and the file is saved back automatically.

Write a list of data frames into a Workbook, one sheet per element. `file` can be a `wbWorkbook` object or a file path string. When a file path is given, the workbook is loaded (or created if absent), written, and saved back automatically. The file extension is normalised to `.xlsx`.

Usage

```
write_sheet(d, file, sheetName, overwrite = FALSE)

write_sheets(lst, file, .progress = "none", show = FALSE, overwrite_wb = TRUE)

write_list2xlsx(
  lst,
  file,
  .progress = "none",
  show = FALSE,
  overwrite_wb = TRUE
)
```

Arguments

`d`	A data frame.
`file`	A file path string or an existing `wbWorkbook` object.
`sheetName`	Sheet name string.
`overwrite`	if TRUE and `sheetName` already exists, replace it.
`lst`	List of data frames. Names are used as sheet names.
`.progress`	name of the progress bar to use, see create_progress_bar.
`show`	open the file after saving (only applies when `file` is a path).
`overwrite_wb`	if TRUE (default) always create a fresh workbook when `file` is a path, even if the file already exists. Set to FALSE to load and append to an existing file.

Value

The (modified) `wbWorkbook` object, invisibly.

The (modified) `wbWorkbook` object, invisibly.

Index

add_dn, 3
apply_3d, 3
apply_col, 4
apply_row, *4*
apply_row *(*apply_col*)*, 4
array2dt, 6
array_2dTo3d *(*array_3dTo2d*)*, 5
array_3dTo2d, 5
as.quoted, *11*

base::dim, *27*
base::dir(), *10*

Cairo::Cairo(), *30*
char2expr *(*label_tag*)*, 19
char2script, 7
check_dir *(*mkdir*)*, 22
chunk, 7
clamp, 8
clamp_max *(*clamp*)*, 8
clamp_min *(*clamp*)*, 8
code_ChrVec *(*char2script*)*, 7
create_progress_bar, *12*, *20*
cut_plevels, 8

data.table::fread(), *15*, *26*
date_y *(*date_ym*)*, 9
date_ydn *(*date_ym*)*, 9
date_yj *(*date_ym*)*, 9
date_ym, 9
dir.show, 9
dir2, 10
dt2array *(*array2dt*)*, 6
dt_chr2num *(*dt_tools*)*, 12
dt_day2year, 10
dt_ddply, 11
dt_dlply *(*dt_ddply*)*, 11
dt_ldply *(*dt_ddply*)*, 11
dt_round *(*dt_tools*)*, 12
dt_tools, 12

evince *(*pdf_view*)*, 23
export, 13

file_cp *(*mkdir*)*, 22
file_ext, 13
file_mv *(*mkdir*)*, 22
file_name *(*file_ext*)*, 13
file_size, 14
fliplr *(*flipud*)*, 14
flipud, 14
fprintf, 14
fread_dir, 15

getwd, *10*
getwd_clip, 15
git_commit *(*git_tools*)*, 16
git_commit_amend *(*git_tools*)*, 16
git_push *(*git_tools*)*, 16
git_set_remote *(*git_tools*)*, 16
git_tools, 16
github, 16
grDevices::cairo_pdf(), *30*
grDevices::png(), *30*
group_map2, 16
grouped_list, 17

ifelse2, 17
import, 18
in_parallel(), *12*, *25*
install_gitee, 18

key_blind, 19
killCluster, 19

label_tag, 19
laply *(*llply*)*, 20
ldply *(*llply*)*, 20
listk, 20
llply, 20

map_simplify *(*llply*)*, 20

matrixStats::rowRanges, *4*
melt_list, 21
melt_tree *(*melt_list*)*, 21
mkdir, 22

obj.size *(*obj_size*)*, 22
obj_size, 22

parse_deg, 23
path.expand, *10*
path.mnt, 23
path_mnt *(*path.mnt*)*, 23
pdf_view, 23
print.data.table, 24
pro_map, 24
progress_bars, *25*
progressively, 25

read_excel, 26
read_xlsx, 26
read_xlsx(), *26*
read_xlsx2list, 26
readxl::read_excel(), *26*
readxl::read_xls(), *26*
regular expression, *10*
rtrend::slope_mk(), *28*
runningId, 27

set_dim, 27
set_dirRoot *(*set_jupyter*)*, 28
set_jupyter, 28
setwd_clip *(*getwd_clip*)*, 15
slope_arr, 28
SumatraPDF *(*pdf_view*)*, 23

which.dup *(*which.na*)*, 29
which.empty *(*which.na*)*, 29
which.isnull *(*which.na*)*, 29
which.na, 29
which.notempty *(*which.na*)*, 29
which.notna *(*which.na*)*, 29
which.notnull *(*which.na*)*, 29
which.null *(*which.na*)*, 29
write_fig, 29
write_list2xlsx *(*write_sheet*)*, 31
write_sheet, 31
write_sheets *(*write_sheet*)*, 31