

# Package: ggrrr (via r-universe)

October 10, 2026

**Title** Addressing Annoyances and Irritations

**Version** 0.0.0.9028

**Description** Visualisation hacks, tabular data helpers, fonts, caching,  
tidy data functions. It is an swiss army knife, jack of all  
trades.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Biarch** false

**Imports** devtools, dplyr, glue, grid, gridExtra, gtable, huxtable,  
openxlsx, rlang, tibble, tidyr, rappdirs, fs, ggplot2,  
magrittr, flextable, htmltools, officer, patchwork, purrr,  
pillar, readr, rstudioapi, stringr, tidyselect, systemfonts,  
scales, colorspace, svglite, rsvg, utils, stats, graphics,  
grDevices, knitr, base64enc, ragg, lifecycle, pdftools,  
interfacer, PDQutils, fastmap, httr, pkgtools, whisker, withr

**Suggests** tidyverse, rmarkdown, gt, rJava, testthat (>= 3.0.0), Cairo

**Remotes** terminological/pkgtools

**VignetteBuilder** knitr

**RoxygenNote** 7.3.3.9007

**URL** <https://terminological.github.io/ggrrr/index.html>,  
<https://github.com/terminological/ggrrr>,  
<https://terminological.github.io/ggrrr/>

**BugReports** <https://github.com/terminological/ggrrr/issues>

**Config/testthat/edition** 3

**SystemRequirements** Google Chrome or other Chromium-based browser.  
chromium: chromium (rpm) or chromium-browser (deb)

**Depends** R (>= 4.4)

**Roxygen** list(markdown = TRUE)

**Config/pak/sysreqs** libcairo2-dev chromium cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev git make libharfbuzz-dev texlive libgit2-dev libicu-dev libjpeg-dev libpng-dev librsvg2-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libpoppler-cpp-dev poppler-data libx11-dev zlib1g-dev

**Repository** <https://terminological.r-universe.dev>

**Date/Publication** 2025-09-25 16:27:51 UTC

**RemoteUrl** <https://github.com/terminological/ggrrr>

**RemoteRef** 0.0.0.9028

**RemoteSha** 4e361832b453efd136ea48e6256da8bc05554cf3

## Contents

|                                       |    |
|---------------------------------------|----|
| \$.checked_list . . . . .             | 4  |
| as.character.rendered_plot . . . . .  | 5  |
| as.character.rendered_table . . . . . | 6  |
| as.long_format_table . . . . .        | 6  |
| as.struct_list . . . . .              | 7  |
| as_vars . . . . .                     | 8  |
| bind_rows_with_factors . . . . .      | 8  |
| breaks_log1p . . . . .                | 9  |
| cache_clear . . . . .                 | 9  |
| cache_delete_stale . . . . .          | 10 |
| cache_download . . . . .              | 11 |
| cached . . . . .                      | 12 |
| check_font . . . . .                  | 13 |
| code_snip . . . . .                   | 15 |
| code_snip_by_line . . . . .           | 16 |
| cran . . . . .                        | 17 |
| cut_date . . . . .                    | 17 |
| cut_integer . . . . .                 | 18 |
| cut_time . . . . .                    | 19 |
| data_supplement . . . . .             | 20 |
| date_to_time . . . . .                | 21 |
| dbeta2 . . . . .                      | 22 |
| dgamma2 . . . . .                     | 23 |
| dlnorm2 . . . . .                     | 24 |
| dnbinom2 . . . . .                    | 25 |
| drawDetails.watermark . . . . .       | 27 |
| dwedge . . . . .                      | 27 |
| fit_col_widths . . . . .              | 28 |
| fonts_available . . . . .             | 29 |
| full_seq_dates . . . . .              | 29 |
| full_seq_times . . . . .              | 30 |
| get_value_sets . . . . .              | 31 |
| gg_find_webfonts . . . . .            | 31 |
| gg_formatted_table . . . . .          | 32 |

|                                     |    |
|-------------------------------------|----|
| gg_hide_legend . . . . .            | 32 |
| gg_hide_X_axis . . . . .            | 33 |
| gg_hide_Y_axis . . . . .            | 33 |
| gg_label_size . . . . .             | 33 |
| gg_layer . . . . .                  | 34 |
| gg_narrow . . . . .                 | 35 |
| gg_pedantic . . . . .               | 35 |
| gg_resize_legend . . . . .          | 40 |
| gg_save_as . . . . .                | 41 |
| gg_set_size_defaults . . . . .      | 42 |
| gg_set_X_angle . . . . .            | 43 |
| gg_simple_table . . . . .           | 43 |
| gg_tiny_theme . . . . .             | 44 |
| gg_watermark . . . . .              | 44 |
| glyphmap . . . . .                  | 45 |
| here . . . . .                      | 45 |
| html_pdf_converter . . . . .        | 46 |
| hux_auto_widths . . . . .           | 46 |
| hux_bind_rows . . . . .             | 47 |
| hux_default_layout . . . . .        | 47 |
| hux_insert_start . . . . .          | 48 |
| hux_nest_group . . . . .            | 48 |
| hux_save_as . . . . .               | 49 |
| hux_set_caption . . . . .           | 49 |
| hux_set_font . . . . .              | 50 |
| hux_set_footer . . . . .            | 50 |
| hux_sprintf . . . . .               | 51 |
| hux_tidy . . . . .                  | 51 |
| hux_to_ggplot . . . . .             | 52 |
| intersecting_group_by . . . . .     | 53 |
| knit_print.rendered_plot . . . . .  | 53 |
| knit_print.rendered_table . . . . . | 54 |
| knit_versioned . . . . .            | 54 |
| logit_trans . . . . .               | 55 |
| map_struct . . . . .                | 56 |
| map2_struct . . . . .               | 57 |
| non_cran . . . . .                  | 57 |
| optional_fn . . . . .               | 58 |
| outputter . . . . .                 | 59 |
| pbeta2 . . . . .                    | 60 |
| percent_trans . . . . .             | 61 |
| pgamma2 . . . . .                   | 61 |
| plnorm2 . . . . .                   | 62 |
| pmap_struct . . . . .               | 63 |
| pmixnorm . . . . .                  | 64 |
| pnbinom2 . . . . .                  | 65 |
| print.rendered_plot . . . . .       | 66 |
| print.rendered_table . . . . .      | 67 |

|                                   |    |
|-----------------------------------|----|
| pwedge . . . . .                  | 67 |
| qbeta2 . . . . .                  | 68 |
| qgamma2 . . . . .                 | 69 |
| qlnorm2 . . . . .                 | 70 |
| qmixnorm . . . . .                | 71 |
| qnbinom2 . . . . .                | 72 |
| qwedge . . . . .                  | 74 |
| rbern . . . . .                   | 75 |
| rbeta2 . . . . .                  | 75 |
| rcategorical . . . . .            | 76 |
| rebuild_fonts . . . . .           | 77 |
| reset_fonts . . . . .             | 77 |
| rgamma2 . . . . .                 | 78 |
| rlnorm2 . . . . .                 | 79 |
| rbinom2 . . . . .                 | 80 |
| rowwise_mutate . . . . .          | 82 |
| rwedge . . . . .                  | 83 |
| scale_colour_consistent . . . . . | 84 |
| scale_colour_subtype . . . . .    | 86 |
| scale_fill_consistent . . . . .   | 87 |
| scale_fill_subtype . . . . .      | 89 |
| scale_x_log1p . . . . .           | 90 |
| scale_x_logit . . . . .           | 91 |
| scale_x_percent . . . . .         | 91 |
| scale_y_log1p . . . . .           | 92 |
| scale_y_logit . . . . .           | 92 |
| scale_y_percent . . . . .         | 93 |
| sprintf_list . . . . .            | 93 |
| std_size . . . . .                | 94 |
| struct_flatten . . . . .          | 94 |
| subset-struct-list . . . . .      | 95 |
| summarise_with_totals . . . . .   | 96 |
| time_to_date . . . . .            | 97 |
| unstable . . . . .                | 98 |
| wedge . . . . .                   | 98 |

**Index****100**


---

|                 |                              |
|-----------------|------------------------------|
| \$.checked_list | <i>Checked list accessor</i> |
|-----------------|------------------------------|

---

**Description**

A checked list is a sub class of list access operator that throws an error if you attempt to access a value that does not exist (rather than returning NULL) the point of this is to throw errors early if the data changes.

**Usage**

```
## S3 method for class 'checked_list'  
x$y
```

**Arguments**

|   |          |
|---|----------|
| x | the list |
| y | the item |

**Value**

the value of the list item or an error if it does not exist

---

```
as.character.rendered_plot
```

*Convert a rendered\_plot object to a character*

---

**Description**

Convert a rendered\_plot object to a character

**Usage**

```
## S3 method for class 'rendered_plot'  
as.character(x, ...)
```

**Arguments**

|     |                   |
|-----|-------------------|
| x   | the rendered_plot |
| ... | not used          |

**Value**

a named vector

---

```
as.character.rendered_table
```

*Convert a rendered\_table object to a character*

---

### Description

Convert a rendered\_table object to a character

### Usage

```
## S3 method for class 'rendered_table'
as.character(x, ...)
```

### Arguments

|     |                    |
|-----|--------------------|
| x   | the rendered_table |
| ... | not used           |

### Value

a named vector

---

```
as.long_format_table
```

*Convert a table to long format*

---

### Description

Converts a square display table format to a long format suitable for applying as a sequence of formatting operations in a google doc or as a ggplot. Currently only plain dataframes and huxtables are supported but flextables look very doable. Only a limited subset of formatting features is implemented at present as supported by roogledocs. The output format is a simple dataframe with the following columns:

### Usage

```
as.long_format_table(table, ...)
```

### Arguments

|       |                                   |
|-------|-----------------------------------|
| table | the input table (e.g. a huxtable) |
| ...   | passed onto subclass methods      |

## Details

- Character: label - non blank text (a single space is OK but not an empty string)
- Integer: row - must be an integer, 1-based from top left
- Integer: col - must be an integer, 1-based from top left
- Integer: rowSpan - must be an integer, minimum value 1
- Integer: colSpan - must be an integer, minimum value 1
- Character: fontName - font name as seen in font drop down of google docs e.g "Roboto", "Arial", "Times New Roman", unrecognised values will be displayed as Arial
- Character: fontFace - one of "bold", "bold.italic", "italic", "plain"
- Numeric: fontSize - in points
- Character: fillColour - as a hex string e.g. "#aaaaaa". N.b. British English spelling (sorry)
- Numeric: leftBorderWeight - border weight in points - minimum size that appears in google docs is 0.5
- Numeric: rightBorderWeight
- Numeric: topBorderWeight
- Numeric: bottomBorderWeight
- Character: alignment - one of "START", "CENTER", "END"
- Character: valignment - one of "TOP", "MIDDLE", "BOTTOM"

It also has an attribute colWidths which is a vector the same length as the width of the table containing the relative widths of the columns. The overall table width is decided on rendering.

So not supported at the moment are border line types, border colours, control of padding, row height control, alignment on a decimal point, complex content / markup in cells.

## Value

a format that is considered valid for roogledocs::Roogledocs\$updateTable()

---

|                |                                     |
|----------------|-------------------------------------|
| as.struct_list | <i>Cast to a list of structures</i> |
|----------------|-------------------------------------|

---

## Description

TODO: testing this empty input (must specify .class) struct\_list - identity simple struct - wrapped list of struct\_lists is flattened list of structs is wrapped nested plain lists / struct\_lists are flattened

## Usage

```
as.struct_list(x, .class = NULL)
```

## Arguments

|        |                                    |
|--------|------------------------------------|
| x      | a list                             |
| .class | the type of structures in the list |

**Value**

a structured list

---

as\_vars

*Reuse tidy-select syntax outside of a tidy-select function*

---

**Description**

Reuse tidy-select syntax outside of a tidy-select function

**Usage**

```
as_vars(tidyselect, data = NULL)
```

**Arguments**

|            |                                                                                                                                                |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| tidyselect | a tidyselect syntax which will be evaluated in context by looking for a call in the call stack that includes a dataframe as the first argument |
| data       | (optional) a specific dataframe with which to evaluate the tidyselect                                                                          |

**Value**

a list of symbols resulting from the evaluation of the tidyselect in the context of the current call stack (or a provided data frame)

---

bind\_rows\_with\_factors

*Bind rows for columns with factors*

---

**Description**

Bind\_rows works until there are factors with a set of different levels then it throws a wobbly. This handles that particular situation by combining factor levels.

**Usage**

```
bind_rows_with_factors(...)
```

**Arguments**

|     |                      |
|-----|----------------------|
| ... | a list of dataframes |
|-----|----------------------|

**Value**

the union of those dataframes. Factor levels are combined with a superset of all levels

**Examples**

```
library(tidyverse)
bind_rows_with_factors(iris,
  ggplot2::diamonds %>% dplyr::rename(Species = cut)) %>%
  dplyr::pull(Species) %>%
  levels()
```

---

|              |                                                   |
|--------------|---------------------------------------------------|
| breaks_log1p | <i>A scales breaks generator for log1p scales</i> |
|--------------|---------------------------------------------------|

---

**Description**

A scales breaks generator for log1p scales

**Usage**

```
breaks_log1p(n = 5, base = 10)
```

**Arguments**

|      |                         |
|------|-------------------------|
| n    | the number of breaks    |
| base | the base for the breaks |

**Value**

a function for ggplot scale breaks

**Examples**

```
library(tidyverse)
ggplot2::ggplot(diamonds, ggplot2::aes(x=price))+
  ggplot2::geom_density()+
  ggplot2::scale_x_continuous(trans="log1p", breaks=ggrrr::breaks_log1p())
```

---

|             |                                                                                     |
|-------------|-------------------------------------------------------------------------------------|
| cache_clear | <i>Clear data from the passthrough cache for complex or long running operations</i> |
|-------------|-------------------------------------------------------------------------------------|

---

**Description**

Clear data from the passthrough cache for complex or long running operations

**Usage**

```
cache_clear(
  .cache = getOption("cache.dir", default = rappdirs::user_cache_dir("ggrrr")),
  .prefix = ".*",
  interactive = TRUE
)
```

**Arguments**

|                          |                                                                                                                                                                                                 |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.cache</code>      | the location of the cache as a directory. May get its value from <code>getOption("cache.path")</code> or the default value which is <code>rappdirs::user_cache_dir(&lt;package_name&gt;)</code> |
| <code>.prefix</code>     | (optional) a name of the operation so that you can namespace the cached files and do selective clean up operations on them with <code>.cache_clear()</code>                                     |
| <code>interactive</code> | suppress are you sure? warning with a FALSE value (defaults to TRUE)                                                                                                                            |

**Value**

nothing. called for side effects

**Examples**

```
cache_clear(.prefix="example", .cache=tempdir(), interactive=FALSE)
```

---

|                    |                                      |
|--------------------|--------------------------------------|
| cache_delete_stale | <i>Delete stale files in a cache</i> |
|--------------------|--------------------------------------|

---

**Description**

Staleness is determined by the number of days from 2am on the current day in the current time-zone. A item cached for only one day becomes stale at 2am the day after it is cached. The time is configurable and `option(cache.time_day_starts = 0)` would be midnight. Automated analysis using caches and updated data should ensure that analysis does not cross this time point otherwise it may end up using old data.

**Usage**

```
cache_delete_stale(
  .cache = getOption("cache.dir", default = rappdirs::user_cache_dir("ggrrr")),
  .prefix = ".*",
  .stale = getOption("cache.stale", default = Inf)
)
```

**Arguments**

|                      |                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.cache</code>  | the location of the cache as a directory. May get its value from <code>getOption("cache.path")</code> or the default value which is <code>rappdirs::user_cache_dir(&lt;package_name&gt;)</code> |
| <code>.prefix</code> | (optional) a name of the operation so that you can namespace the cached files and do selective clean up operations on them with <code>.cache_clear()</code>                                     |
| <code>.stale</code>  | the length of time in days before considering cached data as stale.                                                                                                                             |

**Value**

nothing. called for side effects.

---

|                |                                            |
|----------------|--------------------------------------------|
| cache_download | <i>Download a file into a local cache.</i> |
|----------------|--------------------------------------------|

---

**Description**

This function copies a remote file to a local cache once and makes sure it is reused.

**Usage**

```
cache_download(
  url,
  ...,
  .nocache = getOption("cache.disable", default = FALSE),
  .cache = getOption("cache.download.dir", default =
    rappdirs::user_cache_dir("ggrrr-download")),
  .stale = getOption("cache.stale", default = Inf),
  .extn = NULL
)
```

**Arguments**

|                      |                                                                                                                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>url</code>     | the url to download                                                                                                                                                                                                                                                                                    |
| <code>...</code>     | Named arguments passed on to <a href="#">utils::download.file</a>                                                                                                                                                                                                                                      |
| <code>method</code>  | Method to be used for downloading files. Current download methods are "internal", "libcurl", "wget", "curl" and "wininet" (Windows only), and there is a value "auto": see 'Details' and 'Note'. The method can also be set through the option "download.file.method": see <a href="#">options()</a> . |
| <code>quiet</code>   | If TRUE, suppress status messages (if any), and the progress bar.                                                                                                                                                                                                                                      |
| <code>mode</code>    | character. The mode with which to write the file. Useful values are "w", "wb" (binary), "a" (append) and "ab". Not used for methods "wget" and "curl". See also 'Details', notably about using "wb" for Windows.                                                                                       |
| <code>cacheOK</code> | logical. Is a server-side cached value acceptable?                                                                                                                                                                                                                                                     |

- extra character vector of additional command-line arguments for the "wget" and "curl" methods.
- headers named character vector of additional HTTP headers to use in HTTP[S] requests. It is ignored for non-HTTP[S] URLs. The User-Agent header taken from the HTTPUserAgent option (see [options](#)) is automatically used as the first header.
- ... allow additional arguments to be passed, unused.
- .nocache an option to defeat the cacheing which can be set globally as options("cache.disable"=TRUE)
- .cache the location of the cache as a directory. May get its value from getOption("cache.path") or the default value which is rappdirs::user\_cache\_dir(<package\_name>)
- .stale the length of time in days before considering cached data as stale.
- .extn the file name extension

Value

the path to the downloaded file

---

|        |                                                                                      |
|--------|--------------------------------------------------------------------------------------|
| cached | <i>A simple pass-through filesystem cache for complex or long running operations</i> |
|--------|--------------------------------------------------------------------------------------|

---

Description

executes .expr and saves the output as an RDS file indexed by hash of code in .expr and the hash of specified input variables (in ..., which should contain any inputs that influence .expr). The expression is evaluated in the current environment and it is up to the user to ensure all significant environmental factors are accounted for in ...

Usage

```
cached(  
  .expr,  
  ...,  
  .nocache = getOption("cache.disable", default = FALSE),  
  .cache = getOption("cache.dir", default = rappdirs::user_cache_dir("ggrrr")),  
  .prefix = getOption("cache.item.prefix", default = "cached"),  
  .stale = getOption("cache.stale", default = Inf)  
)
```

Arguments

- .expr the code the output of which requires caching. Other than a return value this should not create side effects or change global variables.
- ... inputs that the code in .expr depends on and changes in which require the code re-running.

|                       |                                                                                                                                                                                                 |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.nocache</code> | an option to defeat the cacheing which can be set globally as <code>options("cache.disable"=TRUE)</code>                                                                                        |
| <code>.cache</code>   | the location of the cache as a directory. May get its value from <code>getOption("cache.path")</code> or the default value which is <code>rappdirs::user_cache_dir(&lt;package_name&gt;)</code> |
| <code>.prefix</code>  | (optional) a name of the operation so that you can namespace the cached files and do selective clean up operations on them with <code>.cache_clear()</code>                                     |
| <code>.stale</code>   | the length of time in days before considering cached data as stale.                                                                                                                             |

**Value**

the output of `.expr` which will usually be a value

**Examples**

```
colName = "Petal.Width"
{
  iris[[colName]]
} %>% cached(iris, colName, .prefix="example", .cache=tempdir())
```

---

|            |                                     |
|------------|-------------------------------------|
| check_font | <i>Ensures a font is available.</i> |
|------------|-------------------------------------|

---

**Description**

This checks to see if a font exists. If missing it will try and install from google fonts or brick.io. If nothing can be done it will suggest alternatives from `fonts_available()`. In all cases this will make the font available to `systemfonts` (for ragg and svg devices), and `extrafonts` (for pdf etc). Webfonts are automatically downloaded into the users font directory and from there will be picked up by cairo devices in theory, and system pdf/svg viewers. In practice this is a bit hit and miss.

**Usage**

```
check_font(family, .legacy = FALSE)
```

**Arguments**

|                      |                                  |
|----------------------|----------------------------------|
| <code>family</code>  | a font family name or names      |
| <code>.legacy</code> | should we try and register fonts |

**Value**

the font family name if it can be located or an alternative if not.

**Examples**

```

check_font(c("Roboto","Arial","Kings","EB Garamond"))
fonts_available(c("Roboto","Arial","Kings","EB Garamond"))

plot = ggplot2::ggplot()+
  ggplot2::theme_void(base_family="Roboto")+
  ggplot2::geom_point()+ggplot2::theme(margins = ggplot2::margin(14,0,14,0))+
  ggplot2::annotate("label",x=0,y=0,label="Kings: Em dash: \u2014 hello world", family="Kings")+
  ggplot2::annotate("text",x=0,y=1,label="Roboto: UTF-8 subscript 2: \u2082", family="Roboto")+
  ggplot2::annotate("text",x=0,y=2,label="EB Garamond: UTF-8 gte: \u2265", family="EB Garamond")

if (FALSE) {

  # Does not work - "invalid font type" & "font family 'Kings'
  # not found in PostScript font database"
  # font but no unicode support
  check_font(c("Roboto","Kings","EB Garamond"), .legacy=TRUE)
  tmp = tempfile(fileext = ".pdf")
  grDevices::pdf(tmp, width=3, height = 1, units="in")
  plot
  grDevices::dev.off()
  utils::browseURL(tmp)

  # font and unicode support
  tmp = tempfile(fileext = ".pdf")
  grDevices::cairo_pdf(tmp, width=3, height = 1, units="in")
  plot
  grDevices::dev.off()
  utils::browseURL(tmp)

  # font and unicode support
  tmp = tempfile(fileext = ".png")
  grDevices::png(tmp, width=3, height = 1, units="in", res=300)
  plot
  grDevices::dev.off()
  utils::browseURL(tmp)

  # font and unicode support
  tmp = tempfile(fileext = ".png")
  ragg::agg_png(tmp, width=3, height = 1, units="in", res=300)
  plot
  grDevices::dev.off()
  utils::browseURL(tmp)

  # font and unicode support
  tmp = tempfile(fileext = ".svg")
  svglite::svglite(tmp, width=3, height = 1)
  plot
  grDevices::dev.off()
  utils::browseURL(tmp)
}

```

```

# Does not work - "family 'Roboto'
# not included in postscript() device"
# however: names(grDevices::postscriptFonts()) includes Roboto
tmp = tempfile(fileext = ".eps")
grDevices::postscript(tmp, width=3, height = 1)
plot
grDevices::dev.off()
utils::browseURL(tmp)

# font and unicode support
tmp = tempfile(fileext = ".eps")
grDevices::cairo_ps(tmp, width=3, height = 1)
plot
grDevices::dev.off()
utils::browseURL(tmp)

# font and partial unicode support
tmp = tempfile(fileext = ".ps")
Cairo::CairoPS(tmp, width=3, height = 1)
plot
grDevices::dev.off()
utils::browseURL(tmp)

}

```

code\_snip

*Display a code snippet***Description**

Pulls out a code snippet based on a start and end tags as comments within the code

**Usage**

```

code_snip(
  type,
  filename,
  startMatches = "START",
  endMatches = "END",
  includeStart = FALSE,
  includeEnd = FALSE,
  sep = "\n...\n"
)

```

**Arguments**

|          |                      |
|----------|----------------------|
| type     | the code type        |
| filename | the source code file |

|              |                                                 |
|--------------|-------------------------------------------------|
| startMatches | a regex that matched start lines                |
| endMatches   | a regex that matches end lines                  |
| includeStart | is the regex inclusive of the start line or not |
| includeEnd   | is the regex inclusive of the end line or not   |
| sep          | a separator                                     |

**Value**

a text string of the selected code

---

|                   |                               |
|-------------------|-------------------------------|
| code_snip_by_line | <i>Display a code snippet</i> |
|-------------------|-------------------------------|

---

**Description**

Pulls out a code snippet based on a vector of start and end lines.

**Usage**

```
code_snip_by_line(type, filename, starts = 1, ends = Inf, sep = "\n...\n")
```

**Arguments**

|          |                                                          |
|----------|----------------------------------------------------------|
| type     | The code type (as understood by the minted latex plugin) |
| filename | The source code filename                                 |
| starts   | a vector of start indices ( as line numbers )            |
| ends     | a vector of end indices ( as line numbers )              |
| sep      | a separator                                              |

**Value**

a formatted string based on the file

---

|      |                                                           |
|------|-----------------------------------------------------------|
| cran | <i>Make sure packages available on CRAN are installed</i> |
|------|-----------------------------------------------------------|

---

**Description****[Deprecated]****Usage**

```
cran(cran_deps)
```

**Arguments**

cran\_deps      a vector of package names

**Value**

nothing

**Examples**

```
# cran("tidyverse")
```

---

|          |                                                            |
|----------|------------------------------------------------------------|
| cut_date | <i>Places a set of dates withing a regular time series</i> |
|----------|------------------------------------------------------------|

---

**Description**

where the periodicity of the time series is expressed as numbers of days, weeks, months quarters, or years. **[Deprecated]**

**Usage**

```
cut_date(dates, full_seq = dates, factor = FALSE, ...)
```

**Arguments**

|          |                                                                                                                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dates    | a set of dates                                                                                                                                                                                                                                  |
| full_seq | a full sequence of allowable dates as created by 'full_seq_dates()'. Alternatively a vector of dates will some regular periodicity, that will be used as an input for 'full_seq_dates()', if missing this will be derived from the data itself. |
| factor   | return the result as an ordered factor with the date ranges as a label. if false this returns a date vector where the date is                                                                                                                   |
| ...      | if full_seq is not give, or a plain vector of dates, other options for 'full_seq_dates()' can be set here. E.g. (fmt="%d/%m/%Y", period="1 week")                                                                                               |

**Value**

a set of dates, representing the start (or end) of the period the date falls into, where the period is defined by 'full\_seq' - which is usually defined by 'full\_seq\_dates()'

**Examples**

```
# dates = as.Date(c("2020-01-01", "2020-02-01", "2020-01-15", "2020-02-03", NA))
# fs = full_seq_dates(dates, "2 days")
# dates - cut_date(dates, fs)
# cut_date(dates, fs, TRUE)

# A weekly set of dates:
# dates2 = Sys.Date() + floor(stats::runif(50, max=10))*7

# in this specific situation the final date is not truncated because the
# input data is seen as an exact match for the whole output period.
# cut_date(dates2, fmt = "%d/%b", factor = TRUE)

# if the input dates don't line up with the output dates
# there may be incomplete coverage of the first and last category.
# where the cutting results in short periods. In this
# instance the first and last periods are truncated to prevent them
# being counted as complete when they are in fact potentially missing a few days worth of data:
# cut_date(dates2, fmt = "%d/%b", factor = TRUE, period = "-2 weeks", anchor="sun")
```

---

cut\_integer

---

*Cut and label an integer valued quantity*


---

**Description**

Deals with some annoying issues classifying integer data sets, such as ages, into groups. where you want to specify just the change over points as integers and clearly label the resulting ordered factor.

**Usage**

```
cut_integer(
  x,
  cut_points,
  glue = "{label}",
  lower_limit = -Inf,
  upper_limit = Inf,
  ...
)
```

**Arguments**

|             |                                                                                                               |
|-------------|---------------------------------------------------------------------------------------------------------------|
| x           | a vector of integer valued numbers, e.g. ages, counts                                                         |
| cut_points  | a vector of integer valued cut points which define the lower boundaries of conditions                         |
| glue        | a glue spec that may be used to generate a label. It can use {low}, {high}, {next_low}, or {label} as values. |
| lower_limit | the minimum value we should include (this is inclusive for the bottom category) (default -Inf)                |
| upper_limit | the maximum value we should include (this is also inclusive for the top category) (default Inf)               |
| ...         | not used                                                                                                      |

**Value**

an ordered factor of the integer

**Examples**

```
cut_integer(stats::rbinom(20,20,0.5), c(5,10,15))
cut_integer(floor(stats::runif(100,-10,10)), cut_points = c(2,3,4,6), lower_limit=0, upper_limit=10)
```

---

|          |                                                           |
|----------|-----------------------------------------------------------|
| cut_time | <i>Places a set of dates within a regular time series</i> |
|----------|-----------------------------------------------------------|

---

**Description**

where the periodicity of the time series is expressed as numbers of days, weeks, months quarters, or years. **[Deprecated]**

**Usage**

```
cut_time(
  timepoints,
  full_seq = timepoints,
  unit = attr(timepoints, "unit"),
  day_zero = attr(timepoints, "day_zero"),
  factor = FALSE,
  ...
)
```

**Arguments**

|            |                                                                                                                                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| timepoints | a set of times (defined by count of periods from a zero day - see 'date_to_time()')                                                                                                                                                             |
| full_seq   | a full sequence of allowable dates as created by 'full_seq_dates()'. Alternatively a vector of dates with some regular periodicity, that will be used as an input for 'full_seq_dates()', if missing this will be derived from the data itself. |
| unit       | the unit of the timepoints in terms of "1 week"                                                                                                                                                                                                 |
| day_zero   | the origin of the timepoints                                                                                                                                                                                                                    |
| factor     | return the result as an ordered factor with the date ranges as a label. if false this returns a date vector where the date is                                                                                                                   |
| ...        | if full_seq is not give, or a plain vector of dates, other options for 'full_seq_dates()' can be set here. E.g. (fmt="%d/%m/%Y", period="1 week)                                                                                                |

**Value**

a set of dates, representing the start (or end) of the period the date falls into, where the period is defined by 'full\_seq' - which is usually defined by 'full\_seq\_dates()'

**Examples**

```
#dates = as.Date(c("2020-01-01", "2020-02-01", "2020-01-15", "2020-02-03", NA))
#fs = full_seq_dates(dates, "2 days")
#dates = cut_date(dates, fs)
#cut_date(dates, fs, TRUE)

# A weekly set of dates:
# dates2 = Sys.Date() + floor(stats::runif(50, max=10)) * 7
# times2 = date_to_time(dates2)

# in this specific situation the final date is not truncated because the
# input data is seen as an exact match for the whole output period.
#cut_time(times2, fmt = "%d/%b", factor = TRUE)

# if the input dates don't line up with the output dates
# there may be incomplete coverage of the first and last category.
# where the cutting results in short periods. In this instance
# the first and last periods are truncated to prevent them
# being counted as complete when they are in fact potentially missing a few days worth of data:
#cut_time(times2, fmt = "%d/%b", factor = TRUE, period = "-2 weeks", anchor="sun")
#times2 = cut_time(times2, fmt = "%d/%b", factor = FALSE, period = "-2 weeks", anchor="sun")
```

---

data\_supplement

---

*Create a function list that allows for supplementary tables (as huxtables) to be added to a XLSX output file.*

---

**Description**

This function encapsulates a excel output file as a destination for data tables. With the output of this function you can add extra data to the supplement as a new sheet, or you can write the spreadsheet to disk. When each data table is written either the table can be written silently or returned so that it is included in a knitr document. This is controlled by option("hide.supplementary.tables"=TRUE).

**Usage**

```
data_supplement(  
  ...,  
  filename = "supplementary-material.xlsx",  
  out = ggrrr::outputter(...),  
  nameGlue = "Supplementary Table {index}"  
)
```

**Arguments**

- ... Named arguments passed on to [.outputter](#)
- directory the root of the output
- ... not used must be empty
- datedFile do you want the filename to have the date appended?
- datedSubdirectory do you want the files to be placed in a dated subdirectory?
- filename the xlsx filename
- out an outputter (defaults to a default outputter )
- nameGlue What will the tables be named

**Value**

a list of 2 functions. \$add\_table(hux, caption, footnote, index), which takes a huxtable, caption, and index a writes the huxtable into a supplementary. \$write() which writes the collection of tables to the excel file.

---

|              |                                                     |
|--------------|-----------------------------------------------------|
| date_to_time | <i>Convert a set of dates to numeric timepoints</i> |
|--------------|-----------------------------------------------------|

---

**Description**

Using a day\_zero and a unit specification or a full sequence of dates (see 'full\_seq\_dates()') **[Deprecated]**

**Usage**

```
date_to_time(  
  dates,  
  unit = .day_interval(dates),  
  day_zero = getOption("day_zero", "2019-12-29")  
)
```

**Arguments**

|          |                                                                                                                         |
|----------|-------------------------------------------------------------------------------------------------------------------------|
| dates    | a vector of dates to convert                                                                                            |
| unit     | a specification of the unit of the resulting time series. Will be determined from periodicity of dates if not specified |
| day_zero | the origin of the conversion. Defaults to the beginning of the COVID pandemic                                           |

**Value**

a sequence of numeric time points as the number of periods since day zero

**Examples**

```
# DEPRECATED
# times = date_to_time(as.Date("2019-12-29")+0:100, "1 week")
# dates = time_to_date(times)
```

---

|        |                              |
|--------|------------------------------|
| dbeta2 | <i>The Beta Distribution</i> |
|--------|------------------------------|

---

**Description**

Density, distribution function, quantile function and random generation for the Beta distribution with parameters shape1 and shape2 (and optional non-centrality parameter ncp).

**Usage**

```
dbeta2(x, prob, kappa, log = FALSE)
```

**Arguments**

|       |                                                                                                   |
|-------|---------------------------------------------------------------------------------------------------|
| x     | vector of quantiles                                                                               |
| prob  | the mean probability (vectorised)                                                                 |
| kappa | a coefficient of variation. where 0 is no variability and 1 is maximally variability (vectorised) |
| log   | logical; if TRUE, probabilities p are given as log(p).                                            |

**Value**

dbeta gives the density, pbeta the distribution function, qbeta the quantile function, and rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**See Also**

[stats::dbeta\(\)](#)

**Examples**

```
dbeta2(c(0.25,0.5,0.75), 0.5, 0.25)
```

---

dgamma2

*The Gamma Distribution*


---

**Description**

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

**Usage**

```
dgamma2(x, mean, sd = sqrt(mean), log = FALSE, convex = TRUE)
```

**Arguments**

|        |                                                                      |
|--------|----------------------------------------------------------------------|
| x      | vector of quantiles                                                  |
| mean   | the mean value on the true scale (vectorised)                        |
| sd     | the standard deviation on the true scale (vectorised)                |
| log    | logical; if TRUE, probabilities p are given as log(p).               |
| convex | Show a warning if the distribution selected is not a convex function |

**Value**

dgamma gives the density, pgamma gives the distribution function, qgamma gives the quantile function, and rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**See Also**

[stats::dgamma\(\)](#)

**Examples**

```
dgamma2(seq(0,4,0.25), 2, 1)
```

dlnorm2

*The Log Normal Distribution***Description**

Density, distribution function, quantile function and random generation for the log normal distribution whose logarithm has mean equal to meanlog and standard deviation equal to sdlog.

**Usage**

```
dlnorm2(x, mean = 1, sd = sqrt(exp(1) - 1), log = FALSE)
```

**Arguments**

|      |                                                        |
|------|--------------------------------------------------------|
| x    | vector of quantiles                                    |
| mean | the mean value on the true scale (vectorised)          |
| sd   | the standard deviation on the true scale (vectorised)  |
| log  | logical; if TRUE, probabilities p are given as log(p). |

**Details**

The log normal distribution has density

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\log(x)-\mu)^2/2\sigma^2}$$

where  $\mu$  and  $\sigma$  are the mean and standard deviation of the logarithm. The mean is  $E(X) = \exp(\mu + 1/2\sigma^2)$ , the median is  $med(X) = \exp(\mu)$ , and the variance  $Var(X) = \exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)$  and hence the coefficient of variation is  $\sqrt{\exp(\sigma^2) - 1}$  which is approximately  $\sigma$  when that is small (e.g.,  $\sigma < 1/2$ ).

**Value**

dlnorm gives the density, plnorm gives the distribution function, qlnorm gives the quantile function, and rlnorm generates random deviates.

The length of the result is determined by n for rlnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**Note**

The cumulative hazard  $H(t) = -\log(1 - F(t))$  is -plnorm(t, r, lower = FALSE, log = TRUE).

## Source

dlnorm is calculated from the definition (in ‘Details’). [pqr]lnorm are based on the relationship to the normal.

Consequently, they model a single point mass at  $\exp(\text{meanlog})$  for the boundary case  $\text{sdlog} = 0$ .

## References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 14. Wiley, New York.

## See Also

[Distributions](#) for other standard distributions, including [dnorm](#) for the normal distribution.

## Examples

```
dlnorm2(seq(0,4,0.25), 2, 1)
```

---

dnbinom2

---

*The Negative Binomial Distribution*


---

## Description

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters size and prob.

## Usage

```
dnbinom2(x, mean, sd = sqrt(mean), log = FALSE)
```

## Arguments

|      |                                                        |
|------|--------------------------------------------------------|
| x    | vector of (non-negative integer) quantiles.            |
| mean | the mean value on the true scale (vectorised)          |
| sd   | the standard deviation on the true scale (vectorised)  |
| log  | logical; if TRUE, probabilities p are given as log(p). |

## Details

The negative binomial distribution with  $\text{size} = n$  and  $\text{prob} = p$  has density

$$p(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!} p^n (1-p)^x$$

for  $x = 0, 1, 2, \dots, n > 0$  and  $0 < p \leq 1$ .

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is  $\mu = n(1-p)/p$  and variance  $n(1-p)/p^2$ .

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see [pgamma](#)) with scale parameter  $(1 - \text{prob})/\text{prob}$  and shape parameter  $\text{size}$ . (This definition allows non-integer values of  $\text{size}$ .)

An alternative parametrization (often used in ecology) is by the *mean*  $\mu$  (see above), and *size*, the *dispersion parameter*, where  $\text{prob} = \text{size}/(\text{size} + \mu)$ . The variance is  $\mu + \mu^2/\text{size}$  in this parametrization.

If an element of  $x$  is not integer, the result of `dnbinom` is zero, with a warning.

The case  $\text{size} == 0$  is the distribution concentrated at zero. This is the limiting distribution for  $\text{size}$  approaching zero, even if  $\mu$  rather than  $\text{prob}$  is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of  $\mu$ .

The quantile is defined as the smallest value  $x$  such that  $F(x) \geq p$ , where  $F$  is the distribution function.

## Value

`dnbinom` gives the density, `pnbinom` gives the distribution function, `qnbino` gives the quantile function, and `rnbinom` generates random deviates.

Invalid  $\text{size}$  or  $\text{prob}$  will result in return value `NaN`, with a warning.

The length of the result is determined by  $n$  for `rnbinom`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than  $n$  are recycled to the length of the result. Only the first elements of the logical arguments are used.

`rnbinom` returns a vector of type [integer](#) unless generated values exceed the maximum representable integer when [double](#) values are returned.

## Source

`dnbinom` computes via binomial probabilities, using code contributed by Catherine Loader (see [dbinom](#)).

`pnbinom` uses [pbeta](#).

`qnbino` uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

`rnbinom` uses the derivation as a gamma mixture of Poisson distributions, see

Devroye, L. (1986) *Non-Uniform Random Variate Generation*. Springer-Verlag, New York. Page 480.

**See Also**

[Distributions](#) for standard distributions, including [dbinom](#) for the binomial, [dpois](#) for the Poisson and [dgeom](#) for the geometric distribution, which is a special case of the negative binomial.

**Examples**

```
dnbinom2(0:5, 2, sqrt(2))
```

---

drawDetails.watermark    *Internal function for drawing watermark on ggplots*

---

**Description**

Internal function for drawing watermark on ggplots

**Usage**

```
## S3 method for class 'watermark'
drawDetails(x, recording)
```

**Arguments**

|           |                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------|
| x         | A grid grob.                                                                                                   |
| recording | A logical value indicating whether a grob is being added to the display list or redrawn from the display list. |

**Value**

a grid object

---

dwedge                      *Wedge distribution*

---

**Description**

The wedge distribution has a domain of 0 to 1 and has a linear probability density function over that domain.

**Usage**

```
dwedge(x, a, lower.tail = TRUE, log = FALSE)
```

**Arguments**

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <code>x</code>          | vector of quantiles                                                                |
| <code>a</code>          | a gradient from -2 (left skewed) to 2 (right skewed)                               |
| <code>lower.tail</code> | logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise $P[X > x]$ . |
| <code>log</code>        | logical; if TRUE, probabilities $p$ are given as $\log(p)$ .                       |

**Details**

The `rwedge` can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

**Value**

a vector of probabilities, quantiles, densities or samples.

**Examples**

```
pwedge(seq(0,1,0.1), a=1)
dwedge(seq(0,1,0.1), a=1)
qwedge(c(0.25,0.5,0.75), a=-1)

stats::cor(
  stats::qnorm(rwedge(1000, a=2)),
  stats::qnorm(rwedge(1000, a=-2))
)
```

---

fit\_col\_widths

*Estimate column content widths*


---

**Description**

Widths are based on dataframe or huxtable content ignoring rowspans and potential for wrapping.

**Usage**

```
fit_col_widths(table)
```

**Arguments**

|                    |                                           |
|--------------------|-------------------------------------------|
| <code>table</code> | a table to get column content widths for. |
|--------------------|-------------------------------------------|

**Value**

a vector of column widths

**Examples**

```
library(tidyverse)
iris %>% fit_col_widths()
```

---

|                 |                                                                           |
|-----------------|---------------------------------------------------------------------------|
| fonts_available | <i>Which fonts are available on this system without hitting webfonts.</i> |
|-----------------|---------------------------------------------------------------------------|

---

### Description

Which fonts are available on this system without hitting webfonts.

### Usage

```
fonts_available(family)
```

### Arguments

|        |                             |
|--------|-----------------------------|
| family | a font family name or names |
|--------|-----------------------------|

### Value

the font family name if it can be located or an empty list otherwise

### Examples

```
fonts_available(c("Arial", "sdfdsfsd"))
```

---

|                |                                               |
|----------------|-----------------------------------------------|
| full_seq_dates | <i>Expand a data vector to the full range</i> |
|----------------|-----------------------------------------------|

---

### Description

Convert a vector of observation dates to a ordered sequence of every day in the time series **[Deprecated]**

### Usage

```
full_seq_dates(dates, period = "1 day", anchor = "start", fmt = "%d %b")
```

### Arguments

|        |                                                                                                                                                                                                                                                  |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dates  | a vector of dates, possibly including NA values                                                                                                                                                                                                  |
| period | the gap between observations as a number or, a negative number means the resulting sequence defines a end of time periods, a positive defines the beginning. may be an integer number of days, or a text string like '2 weeks', '-1 month', etc. |
| anchor | defines the day of week the periods start or end. either "start", "end", a day of the week, or a date                                                                                                                                            |
| fmt    | a strptime formatting string for date range labels.                                                                                                                                                                                              |

**Value**

a vector of dates for complete between the minimum and maximum of dates, on the day of week of the anchoring date

**Examples**

```
# full_seq_dates(c("2020-01-01", "2020-02-01", "2020-01-15", "2020-02-01", NA), "2 days")
```

---

|                |                                                   |
|----------------|---------------------------------------------------|
| full_seq_times | <i>Generate a full regular timepoint sequence</i> |
|----------------|---------------------------------------------------|

---

**Description**

**[Deprecated]**

**Usage**

```
full_seq_times(
  timepoints,
  period = unit,
  unit = attr(timepoints, "unit"),
  day_zero = attr(timepoints, "day_zero"),
  ...
)
```

**Arguments**

|            |                                                                                                                                         |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| timepoints | a set of timepoints relating to data                                                                                                    |
| period     | the desired interval between time points, e.g. "1 day". negative periods define the intervals as closed on the left                     |
| unit       | the unit of the timepoints in terms of "1 week"                                                                                         |
| day_zero   | the origin of the timepoints                                                                                                            |
| ...        | passed to 'full_seq_dates()', particularly anchor, and fmt, to define the day of week of the new sequence and the format of the labels. |

**Value**

a complete set of timepoints on the same scale as the original but with potentially different frequency. This will probably involve non integer times

**Examples**

```
# DEPRECATED
# times = date_to_time(as.Date("2019-12-29")+0:100, "1 week")
# tmp = full_seq_times(times)
```

---

|                |                                            |
|----------------|--------------------------------------------|
| get_value_sets | <i>Get a value set list of a dataframe</i> |
|----------------|--------------------------------------------|

---

**Description**

This function examines a dataframe and returns a list of the columns with sub-lists as all the options for factors. This provides programmatic access (and autocomplete) to the values available in a dataframe, and throws an early error if we try to access data by a variable that does not exist.

**Usage**

```
get_value_sets(df)
```

**Arguments**

|    |                        |
|----|------------------------|
| df | a dataframe to examine |
|----|------------------------|

**Value**

a list of lists with the column name and the factor levels as list, as a checked list.

---

|                  |                                   |
|------------------|-----------------------------------|
| gg_find_webfonts | <i>Find webfonts for a ggplot</i> |
|------------------|-----------------------------------|

---

**Description**

Resolves any missing fonts in a ggplot and tries to resolve them against webfont providers (Brick and Google Fonts) locally caching them. This function is the default for gg\_save\_as

**Usage**

```
gg_find_webfonts(plot)
```

**Arguments**

|      |          |
|------|----------|
| plot | a ggplot |
|------|----------|

**Value**

a list of css webfont specifications

---

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| gg_formatted_table | <i>Display a long format table as a ggplot object.</i> |
|--------------------|--------------------------------------------------------|

---

### Description

This is useful if you want to combine a formatted table with a plot in a multi-panel patchwork.

### Usage

```
gg_formatted_table(
  longFormatTable,
  colWidths = NULL,
  tableWidthInches = 5.9,
  font = "Roboto",
  ...
)
```

### Arguments

|                  |                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------|
| longFormatTable  | a table - usually converted using <code>as.long_format_table()</code>                               |
| colWidths        | (optional) the relative widths of the columns.                                                      |
| tableWidthInches | the maximum desired width of the plot. Text will be scaled to fit this width.                       |
| font             | the default font family                                                                             |
| ...              | passed to <code>as.long_format_table</code> if and only if the input is not already in that format. |

### Value

a ggplot object containing the table as a ggplot.

---

|                |                                  |
|----------------|----------------------------------|
| gg_hide_legend | <i>Hide the legend of a plot</i> |
|----------------|----------------------------------|

---

### Description

Hide the legend of a plot

### Usage

```
gg_hide_legend()
```

### Value

a theme

---

|                |                                  |
|----------------|----------------------------------|
| gg_hide_X_axis | <i>Hide the x axis of a plot</i> |
|----------------|----------------------------------|

---

**Description**

Hide the x axis of a plot

**Usage**

```
gg_hide_X_axis()
```

**Value**

a theme

---

|                |                                  |
|----------------|----------------------------------|
| gg_hide_Y_axis | <i>Hide the y axis of a plot</i> |
|----------------|----------------------------------|

---

**Description**

Hide the y axis of a plot

**Usage**

```
gg_hide_Y_axis()
```

**Value**

a theme

---

|               |                                                         |
|---------------|---------------------------------------------------------|
| gg_label_size | <i>Convert a label size from points to ggplot units</i> |
|---------------|---------------------------------------------------------|

---

**Description**

Labels like `geom_text` are in a random unit size which is only mysteriously connected to the size of text on axes

**Usage**

```
gg_label_size(pts)
```

**Arguments**

|     |                      |
|-----|----------------------|
| pts | label size in points |
|-----|----------------------|

**Value**

a ggplot size aesthetic for labels

---

gg\_layer

---

*Support for ggplot compositional functions*


---

**Description**

I often want to make a function that does both data manipulation or summarisation and plotting of the summary, or combines data from multiple sources. I often want to reuse this and apply similar options to several layers at once, without causing a whole lot of grief. This function allows you to pass on data, augment a aesthetic mapping, and pass on other styling to a set of layers filtering them so they are relevant to the geom. I almost never want `inherit.aes` to be true.

**Usage**

```
gg_layer(
  geom,
  data = NULL,
  mapping,
  ...,
  .default = list(),
  .switch_fill = inherits(geom, "GeomRibbon")
)
```

**Arguments**

|                           |                                                                                                                                                                                      |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>geom</code>         | The ggplot2 layer e.g. <code>ggplot2::GeomPoint</code>                                                                                                                               |
| <code>data</code>         | the data you are going to plot - optional and will be inherited                                                                                                                      |
| <code>mapping</code>      | a mapping typically inherit from the higher level function call or augmented (see example)                                                                                           |
| <code>...</code>          | inherit from the higher level function to enable user customisation these are checked for their relevance to the geom before being passed on                                         |
| <code>.default</code>     | a list containing default aesthetics e.g. <code>list(colour='blue')</code> that can be overridden by the user if they supply a <code>...</code> or mapping aesthetic that overrides. |
| <code>.switch_fill</code> | do you want to use the colour aesthetic for fill for this layer. Most commonly I want to do this with ribbons.                                                                       |

**Value**

a ggplot layer.

**Examples**

```
# top level function contains `...` and `mapping` extensions points:
myPlot = function(data, formula, ..., mapping = .gg_check_for_aes(...)) {
  xCol = rlang::f_lhs(formula)
  yCol = rlang::f_rhs(formula)
  ggplot2::ggplot(data)+
    gg_layer(
      ggplot2::GeomPoint,
      data = data,
      mapping=ggplot2::aes(x=!!xCol, y=!!yCol, !!!mapping),
      ...,
      .default = list(size=10)
    )
}
myPlot(iris, Sepal.Length~Sepal.Width, mapping = ggplot2::aes(colour=Species))
myPlot(iris, Sepal.Length~Petal.Length, mapping = ggplot2::aes(colour=Species), shape="+", size=5)
myPlot(mtcars, mpg~wt, mapping = ggplot2::aes(colour=as.factor(cyl), size=hp))
```

---

|           |                             |
|-----------|-----------------------------|
| gg_narrow | <i>Make a plot narrower</i> |
|-----------|-----------------------------|

---

**Description**

Make a plot narrower

**Usage**

```
gg_narrow(ang = 90)
```

**Arguments**

ang                      the angle for the x labels

**Value**

a theme

---

|             |                                                 |
|-------------|-------------------------------------------------|
| gg_pedantic | <i>An opinionated set of defaults for plots</i> |
|-------------|-------------------------------------------------|

---

**Description**

This is a set of styles with a focus on making plots compact, and minimally fussy, and ensuring fonts are consistent between axes and labels. It sets default sizes for line widths and point sizes. It also switched the default png renderer in knitr to `ragg::ragg_png` to allow for modern font support.

**Usage**

```
gg_pedantic(
  lineSize = 0.25,
  fontSize = 8,
  font = "Roboto",
  size = lineSize * 2,
  ...
)
```

**Arguments**

|          |                                                          |
|----------|----------------------------------------------------------|
| lineSize | the default line and shape size in ggplot units          |
| fontSize | the base font size                                       |
| font     | the default font name.                                   |
| size     | the size of points (the default size aesthetic)          |
| ...      | Named arguments passed on to <code>ggplot2::theme</code> |

... additional element specifications not part of base ggplot2. In general, these should also be defined in the element tree argument. [Splicing](#) a list is also supported.

line all line elements ([element\\_line\(\)](#))

rect all rectangular elements ([element\\_rect\(\)](#))

text all text elements ([element\\_text\(\)](#))

title all title elements: plot, axes, legends ([element\\_text\(\)](#)); inherits from text)

point all point elements ([element\\_point\(\)](#))

polygon all polygon elements ([element\\_polygon\(\)](#))

geom defaults for geoms ([element\\_geom\(\)](#))

spacing all spacings ([unit\(\)](#))

margins all margins ([margin\(\)](#))

aspect.ratio aspect ratio of the panel

axis.title, axis.title.x, axis.title.y, axis.title.x.top, axis.title.x.bottom, axis.title.y.left, axis.title.y.right labels of axes ([element\\_text\(\)](#)). Specify all axes' labels (axis.title), labels by plane (using axis.title.x or axis.title.y), or individually for each axis (using axis.title.x.bottom, axis.title.x.top, axis.title.y.left, axis.title.y.right). axis.title.\*.\* inherits from axis.title.\* which inherits from axis.title, which in turn inherits from text

axis.text, axis.text.x, axis.text.y, axis.text.x.top, axis.text.x.bottom, axis.text.y.left, axis.text.y.right tick labels along axes ([element\\_text\(\)](#)). Specify all axis tick labels (axis.text), tick labels by plane (using axis.text.x or axis.text.y), or individually for each axis (using axis.text.x.bottom, axis.text.x.top, axis.text.y.left, axis.text.y.right). axis.text.\*.\* inherits from axis.text.\* which inherits from axis.text, which in turn inherits from text

axis.ticks, axis.ticks.x, axis.ticks.y, axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y.left, axis.ticks.y.right tick marks along axes ([element\\_line\(\)](#)). Specify all tick marks (axis.ticks), ticks by plane (using axis.ticks.x or axis.ticks.y), or individually for

each axis (using `axis.ticks.x.bottom`, `axis.ticks.x.top`, `axis.ticks.y.left`, `axis.ticks.y.right`). `axis.ticks.*` inherits from `axis.ticks` which inherits from `axis.ticks`, which in turn inherits from `line`

`axis.minor.ticks.x.top`, `axis.minor.ticks.x.bottom`, `axis.minor.ticks.y.left`, `axis.minor.ticks.y.right` minor tick marks along axes (`element_line()`). `axis.minor.ticks.*` inherit from the corresponding major ticks `axis.ticks.*`.

`axis.ticks.length`, `axis.ticks.length.x`, `axis.ticks.length.x.top`, `axis.ticks.length.x.bottom` length of tick marks (unit). `axis.ticks.length` inherits from `spacing`.

`axis.minor.ticks.length`, `axis.minor.ticks.length.x`, `axis.minor.ticks.length.x.top`, `axis.minor.ticks.length.x.bottom` length of minor tick marks (unit), or relative to `axis.ticks.length` when provided with `rel()`.

`axis.line`, `axis.line.x`, `axis.line.x.top`, `axis.line.x.bottom`, `axis.line.y`, `axis.line.y.left`, `axis.line.y.right` lines along axes (`element_line()`). Specify lines along all axes (`axis.line`), lines for each plane (using `axis.line.x` or `axis.line.y`), or individually for each axis (using `axis.line.x.bottom`, `axis.line.x.top`, `axis.line.y.left`, `axis.line.y.right`). `axis.line.*` inherits from `axis.line` which inherits from `axis.line`, which in turn inherits from `line`

`legend.background` background of legend (`element_rect()`; inherits from `rect`)

`legend.margin` the margin around each legend (`margin()`; inherits from `margins`).

`legend.spacing`, `legend.spacing.x`, `legend.spacing.y` the spacing between legends (unit). `legend.spacing.x` & `legend.spacing.y` inherit from `legend.spacing` or can be specified separately. `legend.spacing` inherits from `spacing`.

`legend.key` background underneath legend keys (`element_rect()`; inherits from `rect`)

`legend.key.size`, `legend.key.height`, `legend.key.width` size of legend keys (unit); key background height & width inherit from `legend.key.size` or can be specified separately. In turn `legend.key.size` inherits from `spacing`.

`legend.key.spacing`, `legend.key.spacing.x`, `legend.key.spacing.y` spacing between legend keys given as a unit. Spacing in the horizontal (x) and vertical (y) direction inherit from `legend.key.spacing` or can be specified separately. `legend.key.spacing` inherits from `spacing`.

`legend.key.justification` Justification for positioning legend keys when more space is available than needed for display. The default, `NULL`, stretches keys into the available space. Can be a location like "center" or "top", or a two-element numeric vector.

`legend.frame` frame drawn around the bar (`element_rect()`).

`legend.ticks` tick marks shown along bars or axes (`element_line()`)

`legend.ticks.length` length of tick marks in legend (`unit()`); inherits from `legend.key.size`.

`legend.axis.line` lines along axes in legends (`element_line()`)

`legend.text` legend item labels (`element_text()`; inherits from `text`)

`legend.text.position` placement of legend text relative to legend keys or bars ("top", "right", "bottom" or "left"). The legend text placement might be incompatible with the legend's direction for some guides.

`legend.title` title of legend (`element_text()`; inherits from `title`)  
`legend.title.position` placement of legend title relative to the main legend ("top", "right", "bottom" or "left").  
`legend.position` the default position of legends ("none", "left", "right", "bottom", "top", "inside")  
`legend.position.inside` A numeric vector of length two setting the placement of legends that have the "inside" position.  
`legend.direction` layout of items in legends ("horizontal" or "vertical")  
`legend.byrow` whether the legend-matrix is filled by columns (FALSE, the default) or by rows (TRUE).  
`legend.justification` anchor point for positioning legend inside plot ("center" or two-element numeric vector) or the justification according to the plot area when positioned outside the plot  
`legend.justification.top`, `legend.justification.bottom`, `legend.justification.left`, `legend.justification.right` Same as `legend.justification` but specified per `legend.position` option.  
`legend.location` Relative placement of legends outside the plot as a string. Can be "panel" (default) to align legends to the panels or "plot" to align legends to the plot as a whole.  
`legend.box` arrangement of multiple legends ("horizontal" or "vertical")  
`legend.box.just` justification of each legend within the overall bounding box, when there are multiple legends ("top", "bottom", "left", "right", "center" or "centre")  
`legend.box.margin` margins around the full legend area, as specified using `margin()`; inherits from `margins`.  
`legend.box.background` background of legend area (`element_rect()`; inherits from `rect`)  
`legend.box.spacing` The spacing between the plotting area and the legend box (unit); inherits from `spacing`.  
`panel.background` background of plotting area, drawn underneath plot (`element_rect()`; inherits from `rect`)  
`panel.border` border around plotting area, drawn on top of plot so that it covers tick marks and grid lines. This should be used with `fill = NA` (`element_rect()`; inherits from `rect`)  
`panel.spacing`, `panel.spacing.x`, `panel.spacing.y` spacing between facet panels (unit). `panel.spacing.x` & `panel.spacing.y` inherit from `panel.spacing` or can be specified separately. `panel.spacing` inherits from `spacing`.  
`panel.grid`, `panel.grid.major`, `panel.grid.minor`, `panel.grid.major.x`, `panel.grid.major.y`, `panel.grid.minor.x`, `panel.grid.minor.y` grid lines (`element_line()`). Specify major grid lines, or minor grid lines separately (using `panel.grid.major` or `panel.grid.minor`) or individually for each axis (using `panel.grid.major.x`, `panel.grid.minor.x`, `panel.grid.major.y`, `panel.grid.minor.y`). Y axis grid lines are horizontal and x axis grid lines are vertical. `panel.grid.*.*` inherits from `panel.grid.*` which inherits from `panel.grid`, which in turn inherits from `line`

`panel.ontop` option to place the panel (background, gridlines) over the data layers (logical). Usually used with a transparent or blank `panel.background`.

`panel.widths, panel.heights` Sizes for panels (units). Can be a single unit to set the total size for the panel area, or a unit vector to set the size of individual panels.

`plot.background` background of the entire plot (`element_rect()`; inherits from `rect`)

`plot.title` plot title (text appearance) (`element_text()`; inherits from `title`) left-aligned by default

`plot.title.position, plot.caption.position` Alignment of the plot title/subtitle and caption. The setting for `plot.title.position` applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).

`plot.subtitle` plot subtitle (text appearance) (`element_text()`; inherits from `title`) left-aligned by default

`plot.caption` caption below the plot (text appearance) (`element_text()`; inherits from `title`) right-aligned by default

`plot.tag` upper-left label to identify a plot (text appearance) (`element_text()`; inherits from `title`) left-aligned by default

`plot.tag.position` The position of the tag as a string ("topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright") or a coordinate. If a coordinate, can be a numeric vector of length 2 to set the x,y-coordinate relative to the whole plot. The coordinate option is unavailable for `plot.tag.location = "margin"`.

`plot.tag.location` The placement of the tag as a string, one of "panel", "plot" or "margin". Respectively, these will place the tag inside the panel space, anywhere in the plot as a whole, or in the margin around the panel space.

`plot.margin` margin around entire plot (unit with the sizes of the top, right, bottom, and left margins); inherits from `margin`.

`strip.background, strip.background.x, strip.background.y` background of facet labels (`element_rect()`; inherits from `rect`). Horizontal facet background (`strip.background.x`) & vertical facet background (`strip.background.y`) inherit from `strip.background` or can be specified separately

`strip.clip` should strip background edges and strip labels be clipped to the extend of the strip background? Options are "on" to clip, "off" to disable clipping or "inherit" (default) to take the clipping setting from the parent viewport.

`strip.placement` placement of strip with respect to axes, either "inside" or "outside". Only important when axes and strips are on the same side of the plot.

`strip.text, strip.text.x, strip.text.y, strip.text.x.top, strip.text.x.bottom, strip.text` facet labels (`element_text()`; inherits from `text`). Horizontal facet labels (`strip.text.x`) & vertical facet labels (`strip.text.y`) inherit from `strip.text` or can be specified separately. Facet strips have dedicated

position-dependent theme elements (`strip.text.x.top`, `strip.text.x.bottom`, `strip.text.y.left`, `strip.text.y.right`) that inherit from `strip.text.x` and `strip.text.y`, respectively. As a consequence, some theme stylings need to be applied to the position-dependent elements rather than to the parent elements

`strip.switch.pad.grid`, `strip.switch.pad.wrap` space between strips and axes when strips are switched (unit); inherits from `spacing`.

`complete` set this to `TRUE` if this is a complete theme, such as the one returned by `theme_grey()`. Complete themes behave differently when added to a ggplot object. Also, when setting `complete = TRUE` all elements will be set to inherit from blank elements.

`validate` `TRUE` to run `check_element()`, `FALSE` to bypass checks.

## Value

nothing

---

|                               |                                |
|-------------------------------|--------------------------------|
| <code>gg_resize_legend</code> | <i>Make the legend smaller</i> |
|-------------------------------|--------------------------------|

---

## Description

Make the legend smaller

## Usage

```
gg_resize_legend(pointSize = 0.75, textSize = 6, spaceLegend = 0.75)
```

## Arguments

- |                          |                                                                       |
|--------------------------|-----------------------------------------------------------------------|
| <code>pointSize</code>   | • the ggplot size of lines or points                                  |
| <code>textSize</code>    | • the size in pts of the text                                         |
| <code>spaceLegend</code> | • degree of spacing between items in the scale (defines overall size) |

## Value

a theme

gg\_save\_as

*Save a plot to multiple formats***Description**

Saves a ggplot object to disk at a set physical size. Allows specific maximum dimensions with an optional target aspect ratio to fit into specific configurations for publication. e.g. a half page plot or a third of a 2 column page. Allows output in pdf for journal publication or png for inclusion in documents, and makes sure that the outputs are near identical.

**Usage**

```
gg_save_as(...)
```

**Arguments**

```
...      Named arguments passed on to .gg_save_as
filename base of target filename (excluding extension).
plot     a ggplot
size     a standard size see std_size
maxWidth maximum width in inches
maxHeight maximum height in inches
aspectRatio defaults to maxWidth/maxHeight
formats  some of svg, png, pdf, Rdata, eps, ...
...      Named arguments passed on to svglite::svglite
pointsize Default point size.
standalone Produce a standalone svg file? If FALSE, omits xml header
and default namespace.
id        A character vector of ids to assign to the generated SVG's. If creating
more SVG files than supplied ids the exceeding SVG's will not have an
id tag and a warning will be thrown.
fix_text_size Should the width of strings be fixed so that it doesn't
change between svg renderers depending on their font rendering? Defaults to TRUE. If TRUE each string will have the textLength CSS property set to the width calculated by systemfonts and lengthAdjust='spacingAndGlyphs'. Setting this to FALSE can be beneficial for heavy post-processing that may change content or style of strings, but may lead to inconsistencies between strings and graphic elements that depend on the dimensions of the string (e.g. label borders and background).
scaling   A scaling factor to apply to the rendered line width and text size. Useful for getting the right sizing at the dimension that you need.
always_valid Should the svgfile be a valid svg file while it is being written to? Setting this to TRUE will incur a considerable performance hit (>50% additional rendering time) so this should only be set to TRUE if the file is being parsed while it is still being written to.
file      Identical to filename. Provided for backward compatibility.
```

## Details

For maximum cross platform reproducibility we are using the combination of systemfonts for font management, svglite to render the canonical output rsvg to convert that to pdf, and ragg for bitmap formats. In some situations rsvg fails in which case we fall back to rendering in a headless chrome instance. This rather complicated pipeline ensures modern webfont support, and editable SVG or PDF.

## Value

the output is an sensible default object that can be displayed given the context it is called in, for example if knitting an RMarkdown document a link to the png file for embedding, if latex a link to the pdf file.

---

`gg_set_size_defaults`    *Set sizes in ggplot uniformly*

---

## Description

Set the default sizes of lines, points and fonts in ggplot geoms, and text labels in ggplot axes to get a single consistent look and feel.

## Usage

```
gg_set_size_defaults(  
  lineSize = 0.5,  
  fontSizePts = 4 + lineSize * 8,  
  font = "Roboto",  
  size = lineSize * 2  
)
```

## Arguments

|                          |                                                   |
|--------------------------|---------------------------------------------------|
| <code>lineSize</code>    | the width of lines                                |
| <code>fontSizePts</code> | the size of labels and other on plot text in pts. |
| <code>font</code>        | the font family name                              |
| <code>size</code>        | the size of points (the default size aesthetic)   |

## Value

nothing

## Examples

```
library(tidyverse)  
gg_set_size_defaults(lineSize = 0.25)
```

---

|                |                                                     |
|----------------|-----------------------------------------------------|
| gg_set_X_angle | <i>Set the angle of the x axis labels of a plot</i> |
|----------------|-----------------------------------------------------|

---

**Description**

Also sets horizontal and vertical alignment correctly, and does top and bottom axes.

**Usage**

```
gg_set_X_angle(ang = 60)
```

**Arguments**

|     |                            |
|-----|----------------------------|
| ang | the angle for the x labels |
|-----|----------------------------|

**Value**

a theme

---

|                 |                                                                              |
|-----------------|------------------------------------------------------------------------------|
| gg_simple_table | <i>A simple table as a ggplot patchwork object, no customisation allowed</i> |
|-----------------|------------------------------------------------------------------------------|

---

**Description**

A simple table as a ggplot patchwork object, no customisation allowed

**Usage**

```
gg_simple_table(df, pts = 8, font = "sans", unwrapped = FALSE)
```

**Arguments**

|           |                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| df        | the dataframe with the table data. Column names will become headings                                                                              |
| pts       | text size in points                                                                                                                               |
| font      | the font family                                                                                                                                   |
| unwrapped | <ul style="list-style-type: none"> <li>set this to TRUE if you want to add to a patchwork and use patchwork::wrap_plots(p,list(table))</li> </ul> |

**Value**

A gtable object (i.e. a grob) optionally wrapped as a patchwork plot.

**Examples**

```
if (FALSE) {
  gg_simple_table(tibble::tibble(x=c(1,2,3),y=c(5,4,3)),pts=10)
}
```

---

|               |                                    |
|---------------|------------------------------------|
| gg_tiny_theme | <i>A space saving ggplot theme</i> |
|---------------|------------------------------------|

---

**Description**

A ggplot theme with minimal fluff and with the defaults set small.

**Usage**

```
gg_tiny_theme(baseSize = 8, font = "Roboto")
```

**Arguments**

|          |                            |
|----------|----------------------------|
| baseSize | the size of the base font. |
| font     | the font family name       |

**Value**

a ggplot theme

**Examples**

```
if (interactive()) {
  ggplot2::ggplot(ggplot2::diamonds,
    ggplot2::aes(x=carat,y=price,color=color))+
  ggplot2::geom_point()+
  gg_tiny_theme()
}
```

---

|              |                                    |
|--------------|------------------------------------|
| gg_watermark | <i>Add in a watermark to plots</i> |
|--------------|------------------------------------|

---

**Description**

Add in a watermark to plots

**Usage**

```
gg_watermark(
  lab = "DRAFT",
  disable = getOption("ggrrr.disable.watermark", default = FALSE)
)
```

**Arguments**

|         |                                                                                                                                   |
|---------|-----------------------------------------------------------------------------------------------------------------------------------|
| lab     | the watermark label (DRAFT)                                                                                                       |
| disable | <ul style="list-style-type: none"> <li>global option to disable all watermarks options("ggrrr.disable.watermark"=TRUE)</li> </ul> |

**Value**

a watermark layer

---

|          |                             |
|----------|-----------------------------|
| glyphmap | <i>The glyphmap dataset</i> |
|----------|-----------------------------|

---

**Description**

The adobe glyph name to UTF cross map.

**Usage**

data(glyphmap)

**Format**

A dataframe containing the following columns:

- agl (character) - the agl: adobe glyph label
- unicode (character) - the unicode code

4281 rows and 2 columns

**Details**

from <https://raw.githubusercontent.com/adobe-type-tools/agl-aglfn/refs/heads/master/glyphlist.txt>

---

|      |                                                |
|------|------------------------------------------------|
| here | <i>Drop in replacement for here (here pkg)</i> |
|------|------------------------------------------------|

---

**Description**

Drop in replacement for here (here pkg)

**Usage**

here(..., projRoot = .locate\_project())

**Arguments**

|          |                                                  |
|----------|--------------------------------------------------|
| ...      | the relative path within the project             |
| projRoot | the project root - defaults to .locate_project() |

**Value**

a path

---

|                    |                                                                           |
|--------------------|---------------------------------------------------------------------------|
| html_pdf_converter | <i>Convert html to pdf depending on what is available on the platform</i> |
|--------------------|---------------------------------------------------------------------------|

---

**Description**

If html2pdf is installed it will use that by default, if not it will fall back to a headless chrome instance.

**Usage**

```
html_pdf_converter(html, filename, maxWidth, maxHeight)
```

**Arguments**

|           |                                   |
|-----------|-----------------------------------|
| html      | the html fragment                 |
| filename  | the pdf filename                  |
| maxWidth  | the maximum page width in inches  |
| maxHeight | the maximum page height in inches |

**Value**

nothing called for side effects

---

|                 |                                                                                         |
|-----------------|-----------------------------------------------------------------------------------------|
| hux_auto_widths | <i>Calculate a sensible column and table width for a huxtable based on its content.</i> |
|-----------------|-----------------------------------------------------------------------------------------|

---

**Description**

Calculate a sensible column and table width for a huxtable based on its content.

**Usage**

```
hux_auto_widths(hux, target = "html", including_headers = FALSE)
```

**Arguments**

|                   |                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------|
| hux               | the huxtable                                                                               |
| target            | the expected output (could be "docx"/"odt", "xlsx") which are the only options that matter |
| including_headers | Should we try and fit the header contents as well (TRUE) or let those wrap (FALSE).        |

**Value**

the huxtable with the width options set.

---

|               |                                |
|---------------|--------------------------------|
| hux_bind_rows | <i>Bind rows for huxtables</i> |
|---------------|--------------------------------|

---

**Description**

Sometimes vanilla bind\_rows gets confused.

**Usage**

```
hux_bind_rows(...)
```

**Arguments**

... a list of huxtables

**Value**

a single huxtable

---

|                    |                                           |
|--------------------|-------------------------------------------|
| hux_default_layout | <i>A tidy article theme for huxtables</i> |
|--------------------|-------------------------------------------|

---

**Description**

The main aim is to get something that works with google docs when you copy and paste.

**Usage**

```
hux_default_layout(
  hux,
  defaultFontSize = 8,
  defaultFont = "Roboto",
  headerRows = 1
)
```

**Arguments**

|                 |                                     |
|-----------------|-------------------------------------|
| hux             | a huxtable object                   |
| defaultFontSize | default size of font in points (8)  |
| defaultFont     | the font family name                |
| headerRows      | the number of rows that are headers |

**Value**

the formatted huxtable.

**Examples**

```
library(tidyverse)
hux = iris %>% hux_default_layout()
```

---

|                               |                                               |
|-------------------------------|-----------------------------------------------|
| <code>hux_insert_start</code> | <i>Insert row at start maintaining format</i> |
|-------------------------------|-----------------------------------------------|

---

**Description**

Insert row at start maintaining format

**Usage**

```
hux_insert_start(hux, ..., fill = "", colspan = 1)
```

**Arguments**

- `hux` a huxtable
- `...` stuff to insert into cells
- `fill` padding for empty cells.
- `colspan` how far to span first inserted cell?

**Value**

a huxtable with row inserted at start in the same format

---

|                             |                                 |
|-----------------------------|---------------------------------|
| <code>hux_nest_group</code> | <i>Make a huxtable narrower</i> |
|-----------------------------|---------------------------------|

---

**Description**

Converts row spanning columns into column spanning header rows making a table narrower but longer. The column that is being moved is retained to allow for the appearance of indentation.

**Usage**

```
hux_nest_group(t, col = 1)
```

**Arguments**

- `t` the huxtable
- `col` the column index you want to nest into the row above

**Value**

a narrower huxtable

---

hux\_save\_as

*Save a table to a variety of formats*


---

### Description

depending on the context return the correct format for a document. The basic output here is to use HTML as an output if possible and convert it to an image or a PDF that can then be included into a latex document for example.

### Usage

```
hux_save_as(...)
```

### Arguments

...      Named arguments passed on to [.hux\\_save\\_as](#)

hux   the huxtable to save

filename   the filename, which may omit the extension

size   a `std_size` list entry

maxWidth   or the maximum width in inches

maxHeight   and either the maximum height in inches

aspectRatio   or the minimum allowed aspect ratio

formats   if the extension is omitted, all the formats described here will be saved.  
             Currently supported outputs are "html", "png", "pdf", "docx", "xlsx"

defaultFontSize   the default font size

sheetname   if saving as an xlsx file.

### Value

the output depends on if the function is called in a knitr session. It maybe the HTML or a link to the pdf output for example.

---

hux\_set\_caption

*Set a huxtable caption as a first row*


---

### Description

Keeps the same formatting as the rest of the table

### Usage

```
hux_set_caption(hux, caption)
```

**Arguments**

|         |              |
|---------|--------------|
| hux     | a huxtable   |
| caption | caption text |

**Value**

a huxtable with first row caption

---

|              |                                                            |
|--------------|------------------------------------------------------------|
| hux_set_font | <i>Set the font family and size in a huxtable globally</i> |
|--------------|------------------------------------------------------------|

---

**Description**

Set the font family and size in a huxtable globally

**Usage**

```
hux_set_font(hux, defaultFontSize = 8, defaultFont = "Roboto")
```

**Arguments**

|                 |                       |
|-----------------|-----------------------|
| hux             | a huxtable table      |
| defaultFontSize | the desired font size |
| defaultFont     | the desired font      |

**Value**

the altered huxtable

---

|                |                                                      |
|----------------|------------------------------------------------------|
| hux_set_footer | <i>Add a footer row as a final row in a huxtable</i> |
|----------------|------------------------------------------------------|

---

**Description**

Keeps the same formatting as the rest of the table except for borders

**Usage**

```
hux_set_footer(hux, footer)
```

**Arguments**

|        |             |
|--------|-------------|
| hux    | a huxtable  |
| footer | footer text |

**Value**

a huxtable with last row footer

---

|             |                                                                      |
|-------------|----------------------------------------------------------------------|
| hux_sprintf | <i>A sprintf alternative that handles NA values gracefully (ish)</i> |
|-------------|----------------------------------------------------------------------|

---

**Description**

A sprintf alternative that handles NA values gracefully (ish)

**Usage**

```
hux_sprintf(fmt, ..., na.text = "-")
```

**Arguments**

|         |                                      |
|---------|--------------------------------------|
| fmt     | sprintf format string                |
| ...     | sprintf inputs                       |
| na.text | an string to replace NA values with. |

**Value**

a string value

---

|          |                                                                        |
|----------|------------------------------------------------------------------------|
| hux_tidy | <i>Convert a dataframe to a huxtable with nested rows and columns.</i> |
|----------|------------------------------------------------------------------------|

---

**Description**

The assumption here is that the input data is a long format tidy dataframe with both rows and columns specified by values of the rowGroupVars and colGroupVars columns. The long format (sparse) table is translated into a nested tree of rows (using rowGroupVars) and a nested tree of columns (from colGroupVars). Individual data items are placed in the cell intersecting these two trees. If there are multiple matches an additional layer of grouping is added to the columns.

**Usage**

```
hux_tidy(
  tidyDf,
  rowGroupVars,
  colGroupVars,
  missing = "-",
  na = "-",
  displayRedundantColumnNames = FALSE,
  ...
)
```

**Arguments**

|                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tidyDf                      | A dataframe with row groupings (as a set of columns) and column groupings (as a set of columns) and data, where the data is in a tidy format with a row per "cell" or cell group.                                                                                                                                                                                                                                                                                   |
| rowGroupVars                | A dplyr::vars(...) column specification which will define how rows are grouped                                                                                                                                                                                                                                                                                                                                                                                      |
| colGroupVars                | A dplyr::vars(...) column specification with defines how columns will be grouped                                                                                                                                                                                                                                                                                                                                                                                    |
| missing                     | If there is no content for a given rowGroup / colGroup combination then this character will be used as a placeholder                                                                                                                                                                                                                                                                                                                                                |
| na                          | If there are NA contents then this character will be used.                                                                                                                                                                                                                                                                                                                                                                                                          |
| displayRedundantColumnNames | if there is one column per column group the name of that column may be irrelevant (e.g. if there is a col_name, value fully tidy format) and col_name is in the colGroupVars list then the name of the column value is redundant and not displayed by default. However sometimes you want to display this if you have named it as something specific e.g. including the units. If there is more than one column per colGroup the column titles are needed and kept. |
| ...                         | passed to hux_default_layout()                                                                                                                                                                                                                                                                                                                                                                                                                                      |

**Value**

a huxtable table

---

|               |                                              |
|---------------|----------------------------------------------|
| hux_to_ggplot | <i>Convert a huxtable to a ggplot object</i> |
|---------------|----------------------------------------------|

---

**Description**

Useful if you need to include a formatted table in a figure with a plot

**Usage**

```
hux_to_ggplot(hux, width = 5.9)
```

**Arguments**

|       |                          |
|-------|--------------------------|
| hux   | the huxtable             |
| width | the desired ggplot width |

**Value**

a ggplot object of the right width

---

`intersecting_group_by` *Create a dataframe with groups mathing a range of predicates*

---

### Description

Create a new data frame including duplicate rows where the rows fulfil a potentially overlapping set of conditions specified as named predicates (as formulae)

### Usage

```
intersecting_group_by(.data, ..., .colname)
```

### Arguments

|                       |                                                                                                                      |
|-----------------------|----------------------------------------------------------------------------------------------------------------------|
| <code>.data</code>    | a data frame                                                                                                         |
| <code>...</code>      | a set of predicates specified like <code>case_when</code> s syntax, such as <code>mpg &lt; 5 ~ "gas guzzlers"</code> |
| <code>.colname</code> | the name of the new group                                                                                            |

### Value

a new dataframe containing the overlapping groups which may create duplicates of individual rows.

### Examples

```
library(tidyverse)
iris %>% dplyr::group_by(Species) %>% intersecting_group_by(
  Sepal.Length > mean(Sepal.Length) ~ "Long",
  Sepal.Width > mean(Sepal.Width) ~ "Wide"
)
```

---

`knit_print.rendered_plot`  
*Knit a rendered\_plot object*

---

### Description

Knit a `rendered_plot` object

### Usage

```
## S3 method for class 'rendered_plot'
knit_print(x, options, ...)
```

**Arguments**

|         |                   |
|---------|-------------------|
| x       | the rendered_plot |
| options | the chunk options |
| ...     | not used          |

**Value**

nothing - used for side effects

---

```
knit_print.rendered_table
```

*Knit a rendered\_table object*

---

**Description**

Knit a rendered\_table object

**Usage**

```
## S3 method for class 'rendered_table'
knit_print(x, ...)
```

**Arguments**

|     |                    |
|-----|--------------------|
| x   | the rendered_table |
| ... | not used           |

**Value**

nothing - used for side effects

---

```
knit_versioned
```

*Knit to a versioned file in a sub-directory of the project*

---

**Description**

used in a knitr preamble to direct the output to a subdirectory of the project

**Usage**

```
knit_versioned(
  directory = NULL,
  ...,
  datedFile = !datedSubdirectory,
  datedSubdirectory = FALSE
)
```

**Arguments**

|                   |                                                                                                                         |
|-------------------|-------------------------------------------------------------------------------------------------------------------------|
| directory         | the root of the output - can be an absolute path or a relative path interpreted as relative to the root of the project. |
| ...               | ignored                                                                                                                 |
| datedFile         | do you want the filename to have the date appended (defaults TRUE)?                                                     |
| datedSubdirectory | do you want the files to be placed in a dated subdirectory (defaults FALSE)?                                            |

**Details**

```

---
title: "Analysis 1"
output: html_document
knit: ggrrr::knit_versioned("output/analysis-1")
---
```

This can only work when deployed as a library and hence no standalone version of it exists, because the fully qualified packagename has to be used.

**Value**

nothing. called for side effects

---

|             |                    |
|-------------|--------------------|
| logit_trans | <i>logit scale</i> |
|-------------|--------------------|

---

**Description**

Perform logit scaling with right axis formatting. To not be used directly but with ggplot (e.g. `ggplot2::scale_y_continuous(trans = "logit")`)

**Usage**

```
logit_trans(n = 5, ...)
```

**Arguments**

|     |                 |
|-----|-----------------|
| n   | number of breas |
| ... | not used        |

**Value**

A scales object

## Examples

```
tibble::tibble(pvalue = c(0.001, 0.05, 0.1), fold_change = 1:3) %>%
  ggplot2::ggplot(ggplot2::aes(fold_change , pvalue)) +
  ggplot2::geom_point() +
  ggplot2::scale_y_continuous(trans = "logit")
```

---

map\_struct

*Apply a function to each element of a vector*


---

## Description

The map functions transform their input by applying a function to each element of a list or atomic vector and returning an object of the same length as the input.

- map() always returns a list. See the [modify\(\)](#) family for versions that return an object of the same type as the input.
- map\_lgl(), map\_int(), map\_dbl() and map\_chr() return an atomic vector of the indicated type (or die trying). For these functions, .f must return a length-1 vector of the appropriate type.
- map\_vec() simplifies to the common type of the output. It works with most types of simple vectors like Date, POSIXct, factors, etc.
- walk() calls .f for its side-effect and returns the input .x.

## Usage

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

## Arguments

|           |                                                              |
|-----------|--------------------------------------------------------------|
| .x        | a struct_list                                                |
| .f        | a function to apply to each structure                        |
| ...       | additional parameters to .f but anonymous function preferred |
| .progress | display a progress bar (logical or string name)              |

## Value

a struct\_list

## See Also

[purrr::map\(\)](#)

---

|             |                            |
|-------------|----------------------------|
| map2_struct | <i>Map over two inputs</i> |
|-------------|----------------------------|

---

### Description

These functions are variants of `map()` that iterate over two arguments at a time.

### Usage

```
map2_struct(.x, .y, .f, ..., .progress = FALSE)
```

### Arguments

|                        |                                                                           |
|------------------------|---------------------------------------------------------------------------|
| <code>.x</code>        | a <code>struct_list</code>                                                |
| <code>.y</code>        | a list or vector the same length as <code>.x</code>                       |
| <code>.f</code>        | a function to apply to each <code>.x</code> , <code>.y</code> pair        |
| <code>...</code>       | additional parameters to <code>.f</code> but anonymous function preferred |
| <code>.progress</code> | display a progress bar (logical or string name)                           |

### Value

a `struct_list`

---

|          |                                                 |
|----------|-------------------------------------------------|
| non_cran | <i>Make sure github packages are installed.</i> |
|----------|-------------------------------------------------|

---

### Description

Use a locally checked out version if available. **[Deprecated]**

### Usage

```
non_cran(name, github, force = FALSE, subdir = "", ...)
```

### Arguments

|                     |                                                               |
|---------------------|---------------------------------------------------------------|
| <code>name</code>   | the name of the package                                       |
| <code>github</code> | something like "github-repo/project-name"                     |
| <code>force</code>  | will only update a loaded package if TRUE (defaults to FALSE) |
| <code>subdir</code> | if the package is in a subdirectory of the github repo        |
| <code>...</code>    | passed to <code>devtools::install_github</code>               |

**Value**

nothing

**Examples**

```
# non_cran("patchwork", "thomasp85/patchwork")
```

---

optional\_fn

---

*Get an optional function without triggering a CRAN warning*


---

**Description**

You want to use a function if it is installed but don't want it to be installed as part of your package and you don't want to reference it as part of the Imports or Suggests fields in a package DESCRIPTION.

**Usage**

```
optional_fn(
  pkg,
  name,
  alt = function(...) {
    stop("function `", pkg, "::", name, "(`...)` not available")
  }
)
```

**Arguments**

|      |                                     |
|------|-------------------------------------|
| pkg  | the package name                    |
| name | the function you wish to use        |
| alt  | a function that can be used instead |

**Value**

the function you want if available or the alternative

**Examples**

```
fn = .optional_fn("openssl:md5", rlang::hash)
as.character(fn(as.raw(c(1,2,3))))
```

---

|           |                                                          |
|-----------|----------------------------------------------------------|
| outputter | <i>Generate a versioned file name in a subdirectory.</i> |
|-----------|----------------------------------------------------------|

---

## Description

This function generates a function that resolves a file path fragment to a specific file location, accounting for a versioning strategy involving the current date. The defaults create a naming strategy that places an file in the "output" sub-directory of the current project with a filename suffix including the date.

## Usage

```
outputter(  
  directory = .here("output"),  
  ...,  
  datedFile = !datedSubdirectory,  
  datedSubdirectory = FALSE  
)
```

## Arguments

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| directory         | the root of the output                                      |
| ...               | not used must be empty                                      |
| datedFile         | do you want the filename to have the date appended?         |
| datedSubdirectory | do you want the files to be placed in a dated subdirectory? |

## Value

a function that takes a filename and boolean delete parameter. When called with a filename component this function will return the absolute path of a file which is versioned with date. If the file exists and delete=TRUE it is deleted. (allowing for libraries that refuse to overwrite existing files)

## Examples

```
out = outputter("~/output",datedSubdirectory=TRUE)  
out("file.png")
```

pbeta2

*The Beta Distribution*

---

**Description**

Density, distribution function, quantile function and random generation for the Beta distribution with parameters shape1 and shape2 (and optional non-centrality parameter ncp).

**Usage**

```
pbeta2(q, prob, kappa, lower.tail = TRUE, log.p = FALSE)
```

**Arguments**

|            |                                                                                                   |
|------------|---------------------------------------------------------------------------------------------------|
| q          | vector of quantiles                                                                               |
| prob       | the mean probability (vectorised)                                                                 |
| kappa      | a coefficient of variation. where 0 is no variability and 1 is maximally variability (vectorised) |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise $P[X > x]$ .                |
| log.p      | logical; if TRUE, probabilities p are given as $\log(p)$ .                                        |

**Value**

dbeta gives the density, pbeta the distribution function, qbeta the quantile function, and rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**See Also**

[stats::pbeta\(\)](#)

**Examples**

```
pbeta2(c(0.25, 0.5, 0.75), 0.5, 0.25)
```

---

|               |                         |
|---------------|-------------------------|
| percent_trans | <i>percentage scale</i> |
|---------------|-------------------------|

---

**Description**

display a 0-1 scale as 0-100%

**Usage**

```
percent_trans()
```

**Details**

**[Deprecated]**

**Value**

A scales object

---

|         |                               |
|---------|-------------------------------|
| pgamma2 | <i>The Gamma Distribution</i> |
|---------|-------------------------------|

---

**Description**

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

**Usage**

```
pgamma2(
  q,
  mean,
  sd = sqrt(mean),
  lower.tail = TRUE,
  log.p = FALSE,
  convex = TRUE
)
```

**Arguments**

|            |                                                                                    |
|------------|------------------------------------------------------------------------------------|
| q          | vector of quantiles                                                                |
| mean       | the mean value on the true scale (vectorised)                                      |
| sd         | the standard deviation on the true scale (vectorised)                              |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise $P[X > x]$ . |
| log.p      | logical; if TRUE, probabilities p are given as $\log(p)$ .                         |
| convex     | Show a warning if the distribution selected is not a convex function               |

**Value**

dgamma gives the density, pgamma gives the distribution function, qgamma gives the quantile function, and rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**See Also**

`stats::pgamma()`

**Examples**

```
pgamma2(seq(0,4,0.25), 2, 1)
```

---

plnorm2

*The Log Normal Distribution*

---

**Description**

Density, distribution function, quantile function and random generation for the log normal distribution whose logarithm has mean equal to meanlog and standard deviation equal to sdlog.

**Usage**

```
plnorm2(q, mean = 1, sd = sqrt(exp(1) - 1), lower.tail = TRUE, log.p = FALSE)
```

**Arguments**

|            |                                                                                       |
|------------|---------------------------------------------------------------------------------------|
| q          | vector of quantiles                                                                   |
| mean       | the mean value on the true scale (vectorised)                                         |
| sd         | the standard deviation on the true scale (vectorised)                                 |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ . |
| log.p      | logical; if TRUE, probabilities p are given as log(p).                                |

**Details**

The log normal distribution has density

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\log(x)-\mu)^2/2\sigma^2}$$

where  $\mu$  and  $\sigma$  are the mean and standard deviation of the logarithm. The mean is  $E(X) = \exp(\mu + 1/2\sigma^2)$ , the median is  $med(X) = \exp(\mu)$ , and the variance  $Var(X) = \exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)$  and hence the coefficient of variation is  $\sqrt{\exp(\sigma^2) - 1}$  which is approximately  $\sigma$  when that is small (e.g.,  $\sigma < 1/2$ ).

**Value**

dlnorm gives the density, plnorm gives the distribution function, qlnorm gives the quantile function, and rlnorm generates random deviates.

The length of the result is determined by n for rlnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**Note**

The cumulative hazard  $H(t) = -\log(1 - F(t))$  is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

**Source**

dlnorm is calculated from the definition (in ‘Details’). [pqr]lnorm are based on the relationship to the normal.

Consequently, they model a single point mass at  $\exp(\text{meanlog})$  for the boundary case  $\text{sdlog} = 0$ .

**References**

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 14. Wiley, New York.

**See Also**

[Distributions](#) for other standard distributions, including [dnorm](#) for the normal distribution.

**Examples**

```
plnorm2(seq(0,4,0.25), 2, 1)
```

---

pmap\_struct

*Map over multiple input simultaneously (in "parallel")*


---

**Description**

These functions are variants of [map\(\)](#) that iterate over multiple arguments simultaneously. They are parallel in the sense that each input is processed in parallel with the others, not in the sense of multi-core computing, i.e. they share the same notion of "parallel" as [base::pmax\(\)](#) and [base::pmin\(\)](#).

**Usage**

```
pmap_struct(.l, .f, ..., .progress = FALSE)
```

**Arguments**

|                        |                                                                           |
|------------------------|---------------------------------------------------------------------------|
| <code>.l</code>        | A <code>struct_list</code> .                                              |
| <code>.f</code>        | a function to apply to each <code>.l</code> item                          |
| <code>...</code>       | additional parameters to <code>.f</code> but anonymous function preferred |
| <code>.progress</code> | display a progress bar (logical or string name)                           |

**Value**

a `struct_list`

---

pmixnorm

*Mixture of normal distribution PDFs*

---

**Description**

**[Deprecated]**

**Usage**

```
pmixnorm(q, means, sds, weights = 1, na.rm = FALSE)
```

**Arguments**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>q</code>       | vector of quantiles.                              |
| <code>means</code>   | a vector of normal distribution means             |
| <code>sds</code>     | a vector of normal distribution sds               |
| <code>weights</code> | a vector of weights                               |
| <code>na.rm</code>   | remove distributions which have NA for mean or sd |

**Value**

the pdf of the mixture distribution.

**Examples**

```
pmixnorm(q=c(2,20), means=c(10,13,14), sds=c(1,1,2), weights=c(2,2,3))
```

## Description

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters size and prob.

## Usage

```
pnbinom2(q, mean, sd = sqrt(mean), lower.tail = TRUE, log.p = FALSE)
```

## Arguments

|            |                                                                                       |
|------------|---------------------------------------------------------------------------------------|
| q          | vector of quantiles.                                                                  |
| mean       | the mean value on the true scale (vectorised)                                         |
| sd         | the standard deviation on the true scale (vectorised)                                 |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ . |
| log.p      | logical; if TRUE, probabilities p are given as log(p).                                |

## Details

The negative binomial distribution with size =  $n$  and prob =  $p$  has density

$$p(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!} p^n (1-p)^x$$

for  $x = 0, 1, 2, \dots, n > 0$  and  $0 < p \leq 1$ .

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is  $\mu = n(1-p)/p$  and variance  $n(1-p)/p^2$ .

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see [pgamma](#)) with scale parameter  $(1 - \text{prob})/\text{prob}$  and shape parameter size. (This definition allows non-integer values of size.)

An alternative parametrization (often used in ecology) is by the *mean* mu (see above), and size, the *dispersion parameter*, where  $\text{prob} = \text{size}/(\text{size} + \text{mu})$ . The variance is  $\text{mu} + \text{mu}^2/\text{size}$  in this parametrization.

If an element of x is not integer, the result of dnbinom is zero, with a warning.

The case size == 0 is the distribution concentrated at zero. This is the limiting distribution for size approaching zero, even if mu rather than prob is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of mu.

The quantile is defined as the smallest value  $x$  such that  $F(x) \geq p$ , where  $F$  is the distribution function.

**Value**

dnbinom gives the density, pnbinom gives the distribution function, qnbinom gives the quantile function, and rnbinom generates random deviates.

Invalid size or prob will result in return value NaN, with a warning.

The length of the result is determined by n for rnbinom, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

rnbinom returns a vector of type [integer](#) unless generated values exceed the maximum representable integer when [double](#) values are returned.

**Source**

dnbinom computes via binomial probabilities, using code contributed by Catherine Loader (see [dbinom](#)).

pnbinom uses [pbeta](#).

qnbinom uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

rnbinom uses the derivation as a gamma mixture of Poisson distributions, see

Devroye, L. (1986) *Non-Uniform Random Variate Generation*. Springer-Verlag, New York. Page 480.

**See Also**

[Distributions](#) for standard distributions, including [dbinom](#) for the binomial, [dpois](#) for the Poisson and [dgeom](#) for the geometric distribution, which is a special case of the negative binomial.

**Examples**

```
pnbinom2(0:5, 2, sqrt(2))
```

---

```
print.rendered_plot
```

*Print a rendered\_plot object*

---

**Description**

Print a rendered\_plot object

**Usage**

```
## S3 method for class 'rendered_plot'
print(x, ...)
```

**Arguments**

|     |                   |
|-----|-------------------|
| x   | the rendered_plot |
| ... | not used          |

**Value**

nothing - used for side effects

---

print.rendered\_table    *Print a rendered\_table object*

---

**Description**

Print a rendered\_table object

**Usage**

```
## S3 method for class 'rendered_table'
print(x, ...)
```

**Arguments**

|     |                    |
|-----|--------------------|
| x   | the rendered_table |
| ... | not used           |

**Value**

nothing - used for side effects

---

pwedge                      *Wedge distribution*

---

**Description**

The wedge distribution has a domain of 0 to 1 and has a linear probability density function over that domain.

**Usage**

```
pwedge(q, a, log.p = FALSE)
```

**Arguments**

|       |                                                        |
|-------|--------------------------------------------------------|
| q     | vector of quantiles                                    |
| a     | a gradient from -2 (left skewed) to 2 (right skewed)   |
| log.p | logical; if TRUE, probabilities p are given as log(p). |

**Details**

The `rwedge` can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

**Value**

a vector of probabilities, quantiles, densities or samples.

**Examples**

```
pwedge(seq(0,1,0.1), a=1)
dwedge(seq(0,1,0.1), a=1)
qwedge(c(0.25,0.5,0.75), a=-1)

stats::cor(
  stats::qnorm(rwedge(1000, a=2)),
  stats::qnorm(rwedge(1000, a=-2))
)
```

---

qbeta2

---

*The Beta Distribution*


---

**Description**

Density, distribution function, quantile function and random generation for the Beta distribution with parameters `shape1` and `shape2` (and optional non-centrality parameter `ncp`).

**Usage**

```
qbeta2(p, prob, kappa, lower.tail = TRUE, log.p = FALSE)
```

**Arguments**

|                         |                                                                                                   |
|-------------------------|---------------------------------------------------------------------------------------------------|
| <code>p</code>          | vector of probabilities                                                                           |
| <code>prob</code>       | the mean probability (vectorised)                                                                 |
| <code>kappa</code>      | a coefficient of variation. where 0 is no variability and 1 is maximally variability (vectorised) |
| <code>lower.tail</code> | logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise $P[X > x]$ .                |
| <code>log.p</code>      | logical; if TRUE, probabilities <code>p</code> are given as $\log(p)$ .                           |

**Value**

dbeta gives the density, pbeta the distribution function, qbeta the quantile function, and rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**See Also**

[stats::qbeta\(\)](#)

**Examples**

```
qbeta2(c(0.25,0.5,0.75), 0.5, 0.25)
```

---

qgamma2

*The Gamma Distribution*


---

**Description**

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

**Usage**

```
qgamma2(
  p,
  mean,
  sd = sqrt(mean),
  lower.tail = TRUE,
  log.p = FALSE,
  convex = TRUE
)
```

**Arguments**

|            |                                                                                    |
|------------|------------------------------------------------------------------------------------|
| p          | vector of probabilities                                                            |
| mean       | the mean value on the true scale (vectorised)                                      |
| sd         | the standard deviation on the true scale (vectorised)                              |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise $P[X > x]$ . |
| log.p      | logical; if TRUE, probabilities p are given as $\log(p)$ .                         |
| convex     | Show a warning if the distribution selected is not a convex function               |

**Value**

dgamma gives the density, pgamma gives the distribution function, qgamma gives the quantile function, and rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**See Also**

`stats::qgamma()`

**Examples**

```
qgamma2(c(0.25,0.5,0.75), 2, 1)
```

---

qlnorm2

*The Log Normal Distribution*


---

**Description**

Density, distribution function, quantile function and random generation for the log normal distribution whose logarithm has mean equal to meanlog and standard deviation equal to sdlog.

**Usage**

```
qlnorm2(p, mean = 1, sd = sqrt(exp(1) - 1), lower.tail = TRUE, log.p = FALSE)
```

**Arguments**

|            |                                                                                       |
|------------|---------------------------------------------------------------------------------------|
| p          | vector of probabilities.                                                              |
| mean       | the mean value on the true scale (vectorised)                                         |
| sd         | the standard deviation on the true scale (vectorised)                                 |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ . |
| log.p      | logical; if TRUE, probabilities p are given as log(p).                                |

**Details**

The log normal distribution has density

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\log(x)-\mu)^2/2\sigma^2}$$

where  $\mu$  and  $\sigma$  are the mean and standard deviation of the logarithm. The mean is  $E(X) = \exp(\mu + 1/2\sigma^2)$ , the median is  $med(X) = \exp(\mu)$ , and the variance  $Var(X) = \exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)$  and hence the coefficient of variation is  $\sqrt{\exp(\sigma^2) - 1}$  which is approximately  $\sigma$  when that is small (e.g.,  $\sigma < 1/2$ ).

**Value**

dlnorm gives the density, plnorm gives the distribution function, qlnorm gives the quantile function, and rlnorm generates random deviates.

The length of the result is determined by n for rlnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**Note**

The cumulative hazard  $H(t) = -\log(1 - F(t))$  is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

**Source**

dlnorm is calculated from the definition (in ‘Details’). [pqr]lnorm are based on the relationship to the normal.

Consequently, they model a single point mass at  $\exp(\text{meanlog})$  for the boundary case  $\text{sdlog} = 0$ .

**References**

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 14. Wiley, New York.

**See Also**

[Distributions](#) for other standard distributions, including [dnorm](#) for the normal distribution.

**Examples**

```
qlnorm2(c(0.25, 0.5, 0.72), 2, 1)
```

---

qmixnorm

---

*Mixture of normal distribution quantiles*


---

**Description**

**[Deprecated]**

**Usage**

```
qmixnorm(
  p,
  means,
  sds,
  weights = 1,
  na.rm = FALSE,
  method = c("exact", "samples", "moments"),
  ...
)
```

**Arguments**

|         |                                                                                                        |
|---------|--------------------------------------------------------------------------------------------------------|
| p       | vector of probabilities.                                                                               |
| means   | a vector of normal distribution means                                                                  |
| sds     | a vector of normal distribution sds                                                                    |
| weights | a vector of weights                                                                                    |
| na.rm   | remove distributions with NA values for mean or sd                                                     |
| method  | one of exact (solve with uniroot), samples (random resampling), moments (Cornish Fisher approximation) |
| ...     | passed on to internal function, samples=TRUE to force random sampling                                  |

**Value**

the value of the pth quantile

**Examples**

```
qmixnorm(p=c(0.025,0.5,0.975), means=c(10,13,14), sds=c(1,1,2))
```

---

qnbinom2

*The Negative Binomial Distribution*


---

**Description**

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters size and prob.

**Usage**

```
qnbinom2(p, mean, sd = sqrt(mean), lower.tail = TRUE, log.p = FALSE)
```

## Arguments

|                         |                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------|
| <code>p</code>          | vector of probabilities.                                                              |
| <code>mean</code>       | the mean value on the true scale (vectorised)                                         |
| <code>sd</code>         | the standard deviation on the true scale (vectorised)                                 |
| <code>lower.tail</code> | logical; if TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ . |
| <code>log.p</code>      | logical; if TRUE, probabilities <code>p</code> are given as $\log(p)$ .               |

## Details

The negative binomial distribution with `size = n` and `prob = p` has density

$$p(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!} p^n (1-p)^x$$

for  $x = 0, 1, 2, \dots, n > 0$  and  $0 < p \leq 1$ .

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is  $\mu = n(1-p)/p$  and variance  $n(1-p)/p^2$ .

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see [pgamma](#)) with scale parameter  $(1 - \text{prob})/\text{prob}$  and shape parameter `size`. (This definition allows non-integer values of `size`.)

An alternative parametrization (often used in ecology) is by the *mean* `mu` (see above), and `size`, the *dispersion parameter*, where `prob = size/(size+mu)`. The variance is  $\mu + \mu^2/\text{size}$  in this parametrization.

If an element of `x` is not integer, the result of `dnbinom` is zero, with a warning.

The case `size == 0` is the distribution concentrated at zero. This is the limiting distribution for `size` approaching zero, even if `mu` rather than `prob` is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of `mu`.

The quantile is defined as the smallest value  $x$  such that  $F(x) \geq p$ , where  $F$  is the distribution function.

## Value

`dnbinom` gives the density, `pnbinom` gives the distribution function, `qnbinom` gives the quantile function, and `rnbinom` generates random deviates.

Invalid `size` or `prob` will result in return value NaN, with a warning.

The length of the result is determined by `n` for `rnbinom`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

`rnbinom` returns a vector of type [integer](#) unless generated values exceed the maximum representable integer when [double](#) values are returned.

**Source**

dnbinom computes via binomial probabilities, using code contributed by Catherine Loader (see [dbinom](#)).

pnbinom uses [pbeta](#).

qnbinom uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

rnbinom uses the derivation as a gamma mixture of Poisson distributions, see

Devroye, L. (1986) *Non-Uniform Random Variate Generation*. Springer-Verlag, New York. Page 480.

**See Also**

[Distributions](#) for standard distributions, including [dbinom](#) for the binomial, [dpois](#) for the Poisson and [dgeom](#) for the geometric distribution, which is a special case of the negative binomial.

**Examples**

```
qnbinom2(c(0.25,0.5,0.75), 5, sqrt(5))
```

---

qwedge

*Wedge distribution*


---

**Description**

The wedge distribution has a domain of 0 to 1 and has a linear probability density function over that domain.

**Usage**

```
qwedge(p, a, lower.tail = TRUE, log.p = FALSE)
```

**Arguments**

|            |                                                                                    |
|------------|------------------------------------------------------------------------------------|
| p          | vector of probabilities                                                            |
| a          | a gradient from -2 (left skewed) to 2 (right skewed)                               |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise $P[X > x]$ . |
| log.p      | logical; if TRUE, probabilities p are given as $\log(p)$ .                         |

**Details**

The *rwedge* can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

**Value**

a vector of probabilities, quantiles, densities or samples.

**Examples**

```

pwedge(seq(0,1,0.1), a=1)
dwedge(seq(0,1,0.1), a=1)
qwedge(c(0.25,0.5,0.75), a=-1)

stats::cor(
  stats::qnorm(rwedge(1000, a=2)),
  stats::qnorm(rwedge(1000, a=-2))
)
```

rbern

*A random Bernoulli sample as a logical value***Description**

A random Bernoulli sample as a logical value

**Usage**

```
rbern(n, prob)
```

**Arguments**

|      |                                   |
|------|-----------------------------------|
| n    | number of observations            |
| prob | the mean probability (vectorised) |

**Value**

a vector of logical values of size n

**Examples**

```
table(rbern(100, 0.25))
```

rbeta2

*The Beta Distribution***Description**

Density, distribution function, quantile function and random generation for the Beta distribution with parameters shape1 and shape2 (and optional non-centrality parameter ncp).

**Usage**

```
rbeta2(n, prob, kappa)
```

**Arguments**

|       |                                                                                                   |
|-------|---------------------------------------------------------------------------------------------------|
| n     | number of observations                                                                            |
| prob  | the mean probability (vectorised)                                                                 |
| kappa | a coefficient of variation. where 0 is no variability and 1 is maximally variability (vectorised) |

**Value**

dbeta gives the density, pbeta the distribution function, qbeta the quantile function, and rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**See Also**

[stats::rbeta\(\)](#)

**Examples**

```
rbeta2(3, c(0.1,0.5,0.9),0.1)
```

---

rcategorical

*Sampling from the multinomial equivalent of the Bernoulli distribution*


---

**Description**

Sampling from the multinomial equivalent of the Bernoulli distribution

**Usage**

```
rcategorical(n, prob, factor = FALSE)
```

**Arguments**

|        |                                                                                                                                |
|--------|--------------------------------------------------------------------------------------------------------------------------------|
| n      | a sample size                                                                                                                  |
| prob   | a (optionally named) vector of probabilities that will be normalised to sum to 1                                               |
| factor | if not FALSE then factor levels will either be taken from the names of prob first, or if this is a character vector from this. |

**Value**

a vector of random class labels of length n. Labels come from names of prob or from a character vector in factor.

**Examples**

```

prob = c("one"=0.1,"two"=0.2,"seven"=0.7)
table(rcategorical(1000,prob))
rcategorical(10,prob,factor=TRUE)
rcategorical(10,rep(1,26),factor=letters)

```

rebuild\_fonts

*Rebuild font caches***Description**

This repopulates systemfonts from the webfont cache and then extrafont from systemfonts. This does a full rebuild and will be slow (depending a bit )

**Usage**

```
rebuild_fonts()
```

**Value**

nothing

reset\_fonts

*Reset any custom fonts***Description**

This wipes a lot of cached font data.

**Usage**

```

reset_fonts(
  confirm = utils::askYesNo(msg = "Are you sure?", default = FALSE),
  web = FALSE,
  fonts = FALSE
)

```

**Arguments**

|         |                                                                         |
|---------|-------------------------------------------------------------------------|
| confirm | set to TRUE to automatically confirm                                    |
| web     | also clear webfont cache? (default FALSE)                               |
| fonts   | also clear any downloaded fonts or converted afm files? (default FALSE) |

**Value**

nothing

---

rgamma2*The Gamma Distribution*

---

### Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

### Usage

```
rgamma2(n, mean, sd = sqrt(mean), convex = TRUE)
```

### Arguments

|        |                                                                      |
|--------|----------------------------------------------------------------------|
| n      | number of observations                                               |
| mean   | the mean value on the true scale (vectorised)                        |
| sd     | the standard deviation on the true scale (vectorised)                |
| convex | Show a warning if the distribution selected is not a convex function |

### Value

dgamma gives the density, pgamma gives the distribution function, qgamma gives the quantile function, and rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

### See Also

[stats::rgamma\(\)](#)

### Examples

```
rgamma2(10, 2, 1)
```

rlnorm2

*The Log Normal Distribution***Description**

Density, distribution function, quantile function and random generation for the log normal distribution whose logarithm has mean equal to meanlog and standard deviation equal to sdlog.

**Usage**

```
rlnorm2(n, mean = 1, sd = sqrt(exp(1) - 1))
```

**Arguments**

|      |                                                                                          |
|------|------------------------------------------------------------------------------------------|
| n    | number of observations. If length(n) > 1, the length is taken to be the number required. |
| mean | the mean value on the true scale (vectorised)                                            |
| sd   | the standard deviation on the true scale (vectorised)                                    |

**Details**

The log normal distribution has density

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\log(x)-\mu)^2/2\sigma^2}$$

where  $\mu$  and  $\sigma$  are the mean and standard deviation of the logarithm. The mean is  $E(X) = \exp(\mu + 1/2\sigma^2)$ , the median is  $med(X) = \exp(\mu)$ , and the variance  $Var(X) = \exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)$  and hence the coefficient of variation is  $\sqrt{\exp(\sigma^2) - 1}$  which is approximately  $\sigma$  when that is small (e.g.,  $\sigma < 1/2$ ).

**Value**

dlnorm gives the density, plnorm gives the distribution function, qlnorm gives the quantile function, and rlnorm generates random deviates.

The length of the result is determined by n for rlnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

**Note**

The cumulative hazard  $H(t) = -\log(1 - F(t))$  is -plnorm(t, r, lower = FALSE, log = TRUE).

## Source

dlnorm is calculated from the definition (in ‘Details’). [pqr]lnorm are based on the relationship to the normal.

Consequently, they model a single point mass at  $\exp(\text{meanlog})$  for the boundary case  $\text{sdlog} = 0$ .

## References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 14. Wiley, New York.

## See Also

[Distributions](#) for other standard distributions, including [dnorm](#) for the normal distribution.

## Examples

```
rlnorm2(10, 2, 1)
```

---

rnbinom2

*The Negative Binomial Distribution*


---

## Description

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters size and prob.

## Usage

```
rnbinom2(n, mean, sd = sqrt(mean))
```

## Arguments

|      |                                                                                                    |
|------|----------------------------------------------------------------------------------------------------|
| n    | number of observations. If $\text{length}(n) > 1$ , the length is taken to be the number required. |
| mean | the mean value on the true scale (vectorised)                                                      |
| sd   | the standard deviation on the true scale (vectorised)                                              |

## Details

The negative binomial distribution with  $\text{size} = n$  and  $\text{prob} = p$  has density

$$p(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!} p^n (1-p)^x$$

for  $x = 0, 1, 2, \dots, n > 0$  and  $0 < p \leq 1$ .

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is  $\mu = n(1-p)/p$  and variance  $n(1-p)/p^2$ .

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see [pgamma](#)) with scale parameter  $(1 - \text{prob})/\text{prob}$  and shape parameter  $\text{size}$ . (This definition allows non-integer values of  $\text{size}$ .)

An alternative parametrization (often used in ecology) is by the *mean*  $\mu$  (see above), and *size*, the *dispersion parameter*, where  $\text{prob} = \text{size}/(\text{size} + \mu)$ . The variance is  $\mu + \mu^2/\text{size}$  in this parametrization.

If an element of  $x$  is not integer, the result of `dnbinom` is zero, with a warning.

The case  $\text{size} == 0$  is the distribution concentrated at zero. This is the limiting distribution for  $\text{size}$  approaching zero, even if  $\mu$  rather than  $\text{prob}$  is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of  $\mu$ .

The quantile is defined as the smallest value  $x$  such that  $F(x) \geq p$ , where  $F$  is the distribution function.

## Value

`dnbinom` gives the density, `pnbinom` gives the distribution function, `qnbinom` gives the quantile function, and `rnbinom` generates random deviates.

Invalid  $\text{size}$  or  $\text{prob}$  will result in return value `NaN`, with a warning.

The length of the result is determined by  $n$  for `rnbinom`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than  $n$  are recycled to the length of the result. Only the first elements of the logical arguments are used.

`rnbinom` returns a vector of type [integer](#) unless generated values exceed the maximum representable integer when [double](#) values are returned.

## Source

`dnbinom` computes via binomial probabilities, using code contributed by Catherine Loader (see [dbinom](#)).

`pnbinom` uses [pbeta](#).

`qnbinom` uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

`rnbinom` uses the derivation as a gamma mixture of Poisson distributions, see

Devroye, L. (1986) *Non-Uniform Random Variate Generation*. Springer-Verlag, New York. Page 480.

**See Also**

[Distributions](#) for standard distributions, including [dbinom](#) for the binomial, [dpois](#) for the Poisson and [dgeom](#) for the geometric distribution, which is a special case of the negative binomial.

**Examples**

```
rnbinom2(10, 5, sqrt(5))
```

---

rowwise\_mutate

---

*Create new data in a strictly row-wise fashion without vectorisation*


---

**Description**

Applies an expression to each row and assigns it to a new column. Per-row failures are handled with default values (NAs) or can be intercepted by the user with a `tryCatch(...)` expression. There are many other ways to do a similar thing in `dplyr` and `purrr` but they are all more complicated than I expect them to be.

**Usage**

```
rowwise_mutate(.data, ..., .onerror = function(e, ...) NA)
```

**Arguments**

|                       |                                                                                                                                                                                                                                                    |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.data</code>    | a dataframe. grouping is ignored                                                                                                                                                                                                                   |
| <code>...</code>      | a named list of expressions similar to <code>mutate</code> but where the expressions to be evaluated are evaluated in only in the context of the current row - and are not vectorised. This does not support <a href="#">dplyr::across</a> syntax. |
| <code>.onerror</code> | a function that is called for                                                                                                                                                                                                                      |

**Value**

a dataframe the same length as input with additional or altered columns

**Examples**

```
# calculations are scoped only to current row. Hence max(x) == x always:
iris %>% rowwise_mutate(
  widths = Sepal.Width+max(Petal.Width),
  lengths = Sepal.Length+max(Petal.Length),
  tmp = tibble::tibble(a=1, b=2)) %>%
dplyr::glimpse()

# This is different to standard dplyr behaviour when the additional tibble
# column is considered. standard dplyr rowwise does something unexpected:
iris %>% dplyr::rowwise() %>% dplyr::mutate(
  widths = Sepal.Width+max(Petal.Width),
  lengths = Sepal.Length+max(Petal.Length),
```

```

  tmp = tibble::tibble(a=1, b=2)) %>%
dplyr::glimpse()

withCallingHandlers({

  # As expressions are not vectorised we can use normal if ... else ... statements
  # and errors can be handled and default values provided.
  iris %>% rowwise_mutate(
    tmp = if (Petal.Width > 2.0) stop("error message: ",Petal.Width) else Petal.Width,
    .onerror = function(e) -Petal.Width
  ) %>%
  dplyr::glimpse()

  # The default values
  # are evaluated in the same context as the original expression, but only are
  # defaults for all the columns so makes most sense when a default value is given

  iris %>% rowwise_mutate(
    tmp = if (Petal.Width > 2.0) stop("too wide petals: ",Petal.Width) else Petal.Width,
    tmp2 = if (Sepal.Width > 4) stop("too wide sepals: ",Sepal.Width) else Sepal.Width,
    .onerror = function(e) Inf
  ) %>%
  dplyr::glimpse()

}, warning = function(e) {message(e); invokeRestart("muffleWarning")})

```

---

rwedge

*Wedge distribution*


---

## Description

The wedge distribution has a domain of 0 to 1 and has a linear probability density function over that domain.

## Usage

```
rwedge(n, a)
```

## Arguments

|   |                                                      |
|---|------------------------------------------------------|
| n | number of observations                               |
| a | a gradient from -2 (left skewed) to 2 (right skewed) |

## Details

The `rwedge` can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

**Value**

a vector of probabilities, quantiles, densities or samples.

**Examples**

```
pwedge(seq(0,1,0.1), a=1)
dwedge(seq(0,1,0.1), a=1)
qwedge(c(0.25,0.5,0.75), a=-1)

stats::cor(
  stats::qnorm(rwedge(1000, a=2)),
  stats::qnorm(rwedge(1000, a=-2))
)
```

---

scale\_colour\_consistent

*Use a colour scale from one plot in another*

---

**Description**

Multiple factor levels with custom labels are difficult to sync between graphs if some levels are missing in the second plot. This copies the palette from one plot to another. It is sometimes the case that we want to reuse the fill from one plot as the color for another.

**Usage**

```
scale_colour_consistent(
  plot,
  original_aesthetic = "colour",
  target_aesthetic = "colour",
  ...
)
```

**Arguments**

|                    |                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| plot               | a ggplot with a colour scale to clone                                                                                                                                                           |
| original_aesthetic | the original aesthetic we are cloning (fill or color)                                                                                                                                           |
| target_aesthetic   | the aesthetic in the new plot we want to match.                                                                                                                                                 |
| ...                | Named arguments passed on to <a href="#">ggplot2::scale_fill_manual</a>                                                                                                                         |
|                    | ... Arguments passed on to <a href="#">discrete_scale</a>                                                                                                                                       |
|                    | limits One of: <ul style="list-style-type: none"> <li>• NULL to use the default scale values</li> <li>• A character vector that defines possible values of the scale and their order</li> </ul> |

- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang [lambda](#) function notation.
- drop** Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.
- na.translate** Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.
- name** The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If NULL, the legend title will be omitted.
- minor\_breaks** One of:
- NULL for no minor breaks
  - `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
  - A numeric vector of positions
  - A function that given the limits returns a vector of minor breaks. Also accepts rlang [lambda](#) function notation. When the function has two arguments, it will be given the limits and major break positions.
- labels** One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.
- NULL for no labels
  - `waiver()` for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as breaks)
  - An expression vector (must be the same length as breaks). See `?plot-math` for details.
  - A function that takes the breaks as input and returns labels as output. Also accepts rlang [lambda](#) function notation.
- guide** A function used to create a guide or its name. See [guides\(\)](#) for more information.
- call** The call used to construct the scale for reporting messages.
- super** The super class to use for the constructed scale
- breaks** One of:
- NULL for no breaks
  - `waiver()` for the default breaks (the scale limits)
  - A character vector of breaks
  - A function that takes the limits as input and returns breaks as output
- na.value** The aesthetic value to use for missing (NA) values

---

scale\_colour\_subtype    *A discrete colour scale for dividing where there is a natural ordered subgrouping into groups and subgroups*

---

## Description

This is intended to combine with 'scale\_fill\_subtype' when we want to divide major groupings differently to minor groups

## Usage

```
scale_colour_subtype(
  subclasses,
  class_colour = "black",
  subclass_colour = "grey50",
  na.value = "grey50",
  aesthetics = "color",
  ...
)
```

## Arguments

|                 |                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------|
| subclasses      | a vector containing the count of the subcategories, e.g. c(2,3,4) defines 3 major categories and a total of 9 sub-categories |
| class_colour    | the colour for major group divisions                                                                                         |
| subclass_colour | the colour for sub group divisions                                                                                           |
| na.value        | missing value colour                                                                                                         |
| aesthetics      | this only really makes sense for color scales.                                                                               |
| ...             | passed on to ggplot2::discrete_scale()                                                                                       |

## Value

a ggplot scale

## Examples

```
library(tidyverse)

# prep some data:
data = ggplot2::diamonds %>%
  dplyr::mutate(color_cut = sprintf("%s (%s)", color, cut)) %>%
  dplyr::group_by(color, cut, color_cut) %>%
  dplyr::count() %>%
  dplyr::ungroup() %>%
  dplyr::mutate(color_cut = ordered(color_cut))
```

```
# work out the number of subgroups for each group:
subgroups = data %>%
  dplyr::select(color,cut) %>%
  dplyr::distinct() %>%
  dplyr::group_by(color) %>%
  dplyr::count() %>%
  dplyr::pull(n)

# plot as a horizontal stacked bar chart using color brewer as the main
# colour axis. N.b. having enough different colours here is important
ggplot2::ggplot(data, ggplot2::aes(y=1,x=n, fill=color_cut, color=color_cut))+
  ggplot2::geom_bar(stat="identity",orientation = "y")+
  ggrrr::scale_fill_subtype(.palette = scales::brewer_pal,
    palette="Accent", subclasses = subgroups)+
  ggrrr::scale_colour_subtype(subclasses=subgroups)+
  ggrrr::gg_hide_Y_axis()+
  ggrrr::gg_narrow()
```

---

scale\_fill\_consistent *Use a colour scale from one plot in another*

---

## Description

Multiple factor levels with custom labels are difficult to sync between graphs if some levels are missing in the second plot. This copies the palette from one plot to another. It is sometimes the case that we want to reuse the fill from one plot as the color for another.

## Usage

```
scale_fill_consistent(
  plot,
  original_aesthetic = "fill",
  target_aesthetic = "fill",
  ...
)
```

## Arguments

|                    |                                                                                                         |
|--------------------|---------------------------------------------------------------------------------------------------------|
| plot               | a ggplot with a colour scale to clone                                                                   |
| original_aesthetic | the original aesthetic we are cloning (fill or color)                                                   |
| target_aesthetic   | the aesthetic in the new plot we want to match.                                                         |
| ...                | Named arguments passed on to <a href="#">ggplot2::scale_fill_manual</a>                                 |
|                    | ... Arguments passed on to <a href="#">discrete_scale</a>                                               |
|                    | limits One of: <ul style="list-style-type: none"> <li>• NULL to use the default scale values</li> </ul> |

- A character vector that defines possible values of the scale and their order
  - A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang [lambda](#) function notation.
- drop** Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.
- na.translate** Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.
- name** The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If NULL, the legend title will be omitted.
- minor\_breaks** One of:
- NULL for no minor breaks
  - `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
  - A numeric vector of positions
  - A function that given the limits returns a vector of minor breaks. Also accepts rlang [lambda](#) function notation. When the function has two arguments, it will be given the limits and major break positions.
- labels** One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.
- NULL for no labels
  - `waiver()` for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as `breaks`)
  - An expression vector (must be the same length as `breaks`). See `?plot-math` for details.
  - A function that takes the `breaks` as input and returns labels as output. Also accepts rlang [lambda](#) function notation.
- guide** A function used to create a guide or its name. See [guides\(\)](#) for more information.
- call** The call used to construct the scale for reporting messages.
- super** The super class to use for the constructed scale
- breaks** One of:
- NULL for no breaks
  - `waiver()` for the default breaks (the scale limits)
  - A character vector of breaks
  - A function that takes the limits as input and returns breaks as output
- na.value** The aesthetic value to use for missing (NA) values

---

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| scale_fill_subtype | <i>Discrete fill or colour scale where there is a natural ordered subgrouping</i> |
|--------------------|-----------------------------------------------------------------------------------|

---

## Description

If you have a categorical variable defining colour or fill and it has a natural grouping you can use this to have a colour scale involving major colors defining the major groupings, and these are progressively lightened for each of the subcategories.

## Usage

```
scale_fill_subtype(
  .palette,
  subclasses,
  ...,
  undefined = "#606060",
  lighten = NA,
  na.value = "grey50",
  aesthetics = "fill"
)
```

## Arguments

|            |                                                                                                                                                                                                                                                                                                                                     |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .palette   | the palette for the major groupings, either as a function e.g. 'scales::viridis_pal', or as a manual set of colors e.g. 'c("#FF0000", "#00FF00", "#0000FF")'. if a function can be either discrete or continuous palette.                                                                                                           |
| subclasses | a vector containing the count of the subcategories, e.g. c(2,3,4) defines 3 major categories and a total of 9 sub-categories                                                                                                                                                                                                        |
| ...        | additional options to be passed to the major palette function, e.g. 'option="magma"', or to 'ggplot2::discrete_scale()', e.g. 'alpha=0.5'                                                                                                                                                                                           |
| undefined  | If the number of sub-categories in the data is longer than defined in 'subclasses', the extra categories are assumed to be an set of "other" categories, which will be coloured using this base colour                                                                                                                              |
| lighten    | The factor by which to lighten the colour at each step of the subgrouping. If left blank this will calculate a fraction based on the number of levels of the subgroup. Otherwise if, e.g. 0.5 the first sub category will be the full saturation, the second 0.5 saturation, the third 0.25 saturation, the fourth 0.125 and so on. |
| na.value   | what colour for NA values.                                                                                                                                                                                                                                                                                                          |
| aesthetics | this is a fill scale by default but can be used for colour by setting this to "color" or both as c("fill", "color")                                                                                                                                                                                                                 |

## Value

a ggplot scale

**Examples**

```
library(tidyverse)

# prep some data:
data = ggplot2::diamonds %>%
  dplyr::mutate(color_cut = sprintf("%s (%s)",color,cut)) %>%
  dplyr::group_by(color,cut,color_cut) %>%
  dplyr::count() %>%
  dplyr::ungroup() %>%
  dplyr::mutate(color_cut = ordered(color_cut))

# work out the number of subgroups for each group:
subgroups = data %>%
  dplyr::select(color,cut) %>%
  dplyr::distinct() %>%
  dplyr::group_by(color) %>%
  dplyr::count() %>%
  dplyr::pull(n)

# plot as a horizontal stacked bar chart using color brewer as the main
# colour axis. N.b. having enough different colours here is important
ggplot2::ggplot(data, ggplot2::aes(y=1,x=n, fill=color_cut, color=color_cut))+
  ggplot2::geom_bar(stat="identity",orientation = "y")+
  scale_fill_subtype(.palette = scales::brewer_pal,
    palette="Accent", subclasses = subgroups)+
  scale_colour_subtype(subclasses=subgroups)+
  gg_hide_Y_axis()+
  gg_narrow()
```

---

scale\_x\_log1p

*A log1p x scale*


---

**Description**

A log1p x scale

**Usage**

```
scale_x_log1p(..., n = 5, base = 10, sf = 2)
```

**Arguments**

|      |                                                                    |
|------|--------------------------------------------------------------------|
| ...  | Other arguments passed on to <code>scale_(x y)_continuous()</code> |
| n    | the number of major breaks                                         |
| base | the base for the logarithm                                         |
| sf   | significant figures                                                |

**Value**

a ggplot scale

---

|               |                        |
|---------------|------------------------|
| scale_x_logit | <i>A logit x scale</i> |
|---------------|------------------------|

---

**Description**

A logit x scale

**Usage**

```
scale_x_logit(..., n = 5, sf = 2)
```

**Arguments**

- ... Other arguments passed on to scale\_(x|y)\_continuous()
- n the number of major breaks
- sf significant figures

**Value**

a ggplot scale

---

|                 |                             |
|-----------------|-----------------------------|
| scale_x_percent | <i>A percentage x scale</i> |
|-----------------|-----------------------------|

---

**Description**

A percentage x scale

**Usage**

```
scale_x_percent(..., sf = 2)
```

**Arguments**

- ... Other arguments passed on to scale\_(x|y)\_continuous()
- sf significant figures

**Value**

a ggplot scale

---

|               |                        |
|---------------|------------------------|
| scale_y_log1p | <i>A log1p y scale</i> |
|---------------|------------------------|

---

**Description**

A log1p y scale

**Usage**

```
scale_y_log1p(..., n = 5, base = 10, sf = 2)
```

**Arguments**

- ... Other arguments passed on to scale\_(x|y)\_continuous()
- n the number of major breaks
- base the base for the logarithm
- sf significant figures

**Value**

a ggplot scale

---

|               |                        |
|---------------|------------------------|
| scale_y_logit | <i>A logit y scale</i> |
|---------------|------------------------|

---

**Description**

A logit y scale

**Usage**

```
scale_y_logit(..., n = 5, sf = 2)
```

**Arguments**

- ... Other arguments passed on to scale\_(x|y)\_continuous()
- n the number of major breaks
- sf significant figures

**Value**

a ggplot scale

---

|                 |                             |
|-----------------|-----------------------------|
| scale_y_percent | <i>A percentage y scale</i> |
|-----------------|-----------------------------|

---

**Description**

A percentage y scale

**Usage**

```
scale_y_percent(..., sf = 2)
```

**Arguments**

|     |                                                       |
|-----|-------------------------------------------------------|
| ... | Other arguments passed on to scale_(x y)_continuous() |
| sf  | significant figures                                   |

**Value**

a ggplot scale

---



---

|              |                                  |
|--------------|----------------------------------|
| sprintf_list | <i>Sprintf with a list input</i> |
|--------------|----------------------------------|

---

**Description**

A variant of sprintf that work well with inputs that are in the format of a list. Good examples of which are the quantile functions

**Usage**

```
sprintf_list(format, params, na.replace = "-")
```

**Arguments**

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| format     | the format string                                                 |
| params     | the inputs as a list (rather than as a set of individual numbers) |
| na.replace | a value to replace NA values with.                                |

**Value**

the formatted string

**Examples**

```
# generate a mixture confidence interval from a set of distributions
sprintf_list("%1.2f [%1.2f\u2013%1.2f]",
  qmixnorm(p=c(0.5,0.025,0.975),
    means=c(10,13,14), sds=c(1,1,2)))
```

---

|          |                                       |
|----------|---------------------------------------|
| std_size | <i>Standard image and paper sizes</i> |
|----------|---------------------------------------|

---

**Description**

The width and height of images to fit scientific publication standards.

**Usage**

```
std_size
```

**Format**

A list with width and height in inches

---

|                |                                                     |
|----------------|-----------------------------------------------------|
| struct_flatten | <i>Ensure struct_list is a flat list of structs</i> |
|----------------|-----------------------------------------------------|

---

**Description**

Unlike `purrr::list_flatten` this is recursive for one reason which is that a `struct_list` must only contain structs.

**Usage**

```
struct_flatten(x)
```

**Arguments**

`x` a potentially nested list of `struct_lists`.

**Value**

a flat `struct_list` of structs

---

|                    |                                    |
|--------------------|------------------------------------|
| subset-struct-list | <i>Manipulate structured lists</i> |
|--------------------|------------------------------------|

---

## Description

These functions allow generic list behaviour.

## Usage

```
## S3 method for class 'struct_list'
c(...)

## S3 method for class 'struct_list'
rep(x, ...)

## S3 method for class 'struct_list'
x$y

## S3 method for class 'struct_list'
x[...]

## S3 replacement method for class 'struct_list'
x[...] <- value

## S3 method for class 'struct_list'
x[[...]]

## S3 replacement method for class 'struct_list'
x[[...]] <- value
```

## Arguments

|       |                  |
|-------|------------------|
| ...   | generic support  |
| x     | a struct_list    |
| y     | item to retrieve |
| value | the value        |

## Value

a struct\_list with all the items

## Functions

- `c(struct_list)`: Repeat a struct\_list
- `rep(struct_list)`: Repeat a struct\_list
- `$`: Subset a struct\_list

- `[`: Subset a `struct_list`
- ``[`(struct_list) <- value`: Assign a subset to a `struct_list`
- `[[`: get a value from a `struct_list`
- ``[[`(struct_list) <- value`: set a single value in a `struct_list`

### Examples

```
x = struct(a=1,b=2,c=1:3,.class="test")
y = struct(a=4,b=5,c=1:3,.class="test")
z= tibble::tibble(a= 1:10, b=rep(c(x,y),5))

z$b

c(x,y)
c(rep(x,5),y)
class(c(rep(x,5),rep(y,5))[[1]])

as.struct_list(list(x,y))

#' x = struct(a=1,b=2,c=1:3,.class="test")
class(rep(c(x,y),5)[[1]]) == "test"
class(rep(x,5))

a = (rep(c(x,y),5))
a[[1]] = y
a
```

---

`summarise_with_totals` *Summarise a subgroup and create a summary row*

---

### Description

Summarise and include a total row, or a row including the summary for the whole group, into a factor list. This looks and feels like a natural summarisation step, but applies the summarisation both to the subgroups and to the data ungrouped by one level. The additional group result is included as a new row. allows for a natural grouped and ungrouped summarisation

### Usage

```
summarise_with_totals(
  .data,
  ...,
  .groups = NULL,
  .total = "Total",
  .total_first = FALSE
)
```

**Arguments**

|                           |                                                                                           |
|---------------------------|-------------------------------------------------------------------------------------------|
| <code>.data</code>        | a dataframe                                                                               |
| <code>...</code>          | the summarisation specification                                                           |
| <code>.groups</code>      | what to do with the grouping after summarisation (same as <code>dplyr::summarise</code> ) |
| <code>.total</code>       | name of the total row which will be added into a factor list.                             |
| <code>.total_first</code> | should the total be before or after the groups                                            |

**Value**

a summarised dataframe with the additional totals or group row

**Examples**

```
library(tidyverse)
diamonds %>%
  dplyr::group_by(color, cut) %>%
  summarise_with_totals(
    mpg = sprintf("%.1f \u00B1 %.1f", mean(price), stats::sd(price)),
    .total = "Overall"
  )
```

---

time\_to\_date

---

*Convert a set of timepoints to dates*


---

**Description**

**[Deprecated]**

**Usage**

```
time_to_date(
  timepoints,
  unit = attr(timepoints, "unit"),
  day_zero = getOption("day_zero", "2019-12-29")
)
```

**Arguments**

|                         |                                                                                           |
|-------------------------|-------------------------------------------------------------------------------------------|
| <code>timepoints</code> | a set of numeric time points                                                              |
| <code>unit</code>       | the period / unit of the time points, which will be extracted from timepoints if possible |
| <code>day_zero</code>   | the zero day of the time series, will be extracted from timepoints if possible            |

**Value**

a vector of dates

**Examples**

```
# DEPRECATED
# times = date_to_time(as.Date("2019-12-29")+0:100, "1 week")
# dates = time_to_date(times)
```

---

|          |                                        |
|----------|----------------------------------------|
| unstable | <i>Get unstable version of package</i> |
|----------|----------------------------------------|

---

**Description**

reboot a library reloading from a local development copy if it exists locally alternatively get the most up to date github package. **[Deprecated]**

**Usage**

```
unstable(pkg = "ggrrr", org = "terminological")
```

**Arguments**

|     |                                             |
|-----|---------------------------------------------|
| pkg | the package to load the unstable version of |
| org | the github organisation                     |

**Value**

nothing

**Examples**

```
# ggrrr::unstable()
```

---

|       |                           |
|-------|---------------------------|
| wedge | <i>Wedge distribution</i> |
|-------|---------------------------|

---

**Description**

The wedge distribution has a domain of 0 to 1 and has a linear probability density function over that domain.

**Arguments**

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <code>n</code>          | number of observations                                                             |
| <code>x</code>          | vector of quantiles                                                                |
| <code>q</code>          | vector of quantiles                                                                |
| <code>p</code>          | vector of probabilities                                                            |
| <code>log</code>        | logical; if TRUE, probabilities <code>p</code> are given as $\log(p)$ .            |
| <code>log.p</code>      | logical; if TRUE, probabilities <code>p</code> are given as $\log(p)$ .            |
| <code>lower.tail</code> | logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise $P[X > x]$ . |
| <code>a</code>          | a gradient from -2 (left skewed) to 2 (right skewed)                               |

**Details**

The `rwedge` can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

**Value**

a vector of probabilities, quantiles, densities or samples.

**Examples**

```
pwedge(seq(0,1,0.1), a=1)
dwedge(seq(0,1,0.1), a=1)
qwedge(c(0.25,0.5,0.75), a=-1)

stats::cor(
  stats::qnorm(rwedge(1000, a=2)),
  stats::qnorm(rwedge(1000, a=-2))
)
```

# Index

## \* **cache**

- cache\_clear, 9
- cache\_delete\_stale, 10
- cache\_download, 11
- cached, 12

## \* **datasets**

- glyphmap, 45
- std\_size, 94

## \* **distributions**

- dbeta2, 22
- dgamma2, 23
- dlnorm2, 24
- dnbinom2, 25
- dwedge, 27
- pbeta2, 60
- pgamma2, 61
- plnorm2, 62
- pnbinom2, 65
- pwedge, 67
- qbeta2, 68
- qgamma2, 69
- qlnorm2, 70
- qnbinom2, 72
- qwedge, 74
- rbern, 75
- rbeta2, 75
- rcategorical, 76
- rgamma2, 78
- rlnorm2, 79
- rnbinom2, 80
- rwedge, 83
- wedge, 98

## \* **fonts**

- check\_font, 13
- fonts\_available, 29
- gg\_find\_webfonts, 31
- glyphmap, 45
- rebuild\_fonts, 77
- reset\_fonts, 77

## \* **ggplot**

- breaks\_log1p, 9
- drawDetails.watermark, 27
- gg\_formatted\_table, 32
- gg\_hide\_legend, 32
- gg\_hide\_X\_axis, 33
- gg\_hide\_Y\_axis, 33
- gg\_label\_size, 33
- gg\_layer, 34
- gg\_narrow, 35
- gg\_pedantic, 35
- gg\_resize\_legend, 40
- gg\_set\_size\_defaults, 42
- gg\_set\_X\_angle, 43
- gg\_simple\_table, 43
- gg\_tiny\_theme, 44
- gg\_watermark, 44
- logit\_trans, 55
- percent\_trans, 61
- scale\_colour\_consistent, 84
- scale\_colour\_subtype, 86
- scale\_fill\_consistent, 87
- scale\_fill\_subtype, 89
- scale\_x\_log1p, 90
- scale\_x\_logit, 91
- scale\_x\_percent, 91
- scale\_y\_log1p, 92
- scale\_y\_logit, 92
- scale\_y\_percent, 93

## \* **graph**

- gg\_simple\_table, 43

## \* **huxtable**

- as.long\_format\_table, 6
- fit\_col\_widths, 28
- hux\_auto\_widths, 46
- hux\_bind\_rows, 47
- hux\_default\_layout, 47
- hux\_insert\_start, 48
- hux\_nest\_group, 48

- hux\_set\_caption, 49
- hux\_set\_font, 50
- hux\_set\_footer, 50
- hux\_sprintf, 51
- hux\_tidy, 51
- hux\_to\_ggplot, 52
- \* **layout**
  - gg\_simple\_table, 43
- \* **output**
  - as.character.rendered\_plot, 5
  - as.character.rendered\_table, 6
  - code\_snip, 15
  - code\_snip\_by\_line, 16
  - data\_supplement, 20
  - gg\_save\_as, 41
  - here, 45
  - html\_pdf\_converter, 46
  - hux\_save\_as, 49
  - knit\_print.rendered\_plot, 53
  - knit\_print.rendered\_table, 54
  - knit\_versioned, 54
  - outputter, 59
  - print.rendered\_plot, 66
  - print.rendered\_table, 67
  - std\_size, 94
- \* **structures**
  - as.struct\_list, 7
  - map2\_struct, 57
  - map\_struct, 56
  - pmap\_struct, 63
  - struct\_flatten, 94
  - subset-struct-list, 95
- \* **tidydata**
  - \$.checked\_list, 4
  - bind\_rows\_with\_factors, 8
  - cut\_integer, 18
  - get\_value\_sets, 31
  - intersecting\_group\_by, 53
  - rowwise\_mutate, 82
  - summarise\_with\_totals, 96
  - .gg\_save\_as, 41
  - .hux\_save\_as, 49
  - .outputter, 21
  - [.struct\_list(subset-struct-list), 95
  - [<-.struct\_list(subset-struct-list), 95
  - [[.struct\_list(subset-struct-list), 95
  - [[<-.struct\_list(subset-struct-list), 95
- \$.checked\_list, 4
- \$.struct\_list(subset-struct-list), 95
- as.character.rendered\_plot, 5
- as.character.rendered\_table, 6
- as.long\_format\_table, 6
- as.struct\_list, 7
- as\_vars, 8
- base::pmax(), 63
- base::pmin(), 63
- bind\_rows\_with\_factors, 8
- breaks\_log1p, 9
- c.struct\_list(subset-struct-list), 95
- cache\_clear, 9
- cache\_delete\_stale, 10
- cache\_download, 11
- cached, 12
- check\_font, 13
- code\_snip, 15
- code\_snip\_by\_line, 16
- cran, 17
- cut\_date, 17
- cut\_integer, 18
- cut\_time, 19
- data\_supplement, 20
- date\_to\_time, 21
- dbeta2, 22
- dbinom, 26, 27, 66, 74, 81, 82
- dgamma2, 23
- dgeom, 27, 66, 74, 82
- discrete\_scale, 84, 87
- Distributions, 25, 27, 63, 66, 71, 74, 80, 82
- dlnorm2, 24
- dnbinom2, 25
- dnorm, 25, 63, 71, 80
- double, 26, 66, 73, 81
- dplyr::across, 82
- dpois, 27, 66, 74, 82
- drawDetails.watermark, 27
- dwedge, 27
- element\_geom(), 36
- element\_line(), 36–38
- element\_point(), 36
- element\_polygon(), 36
- element\_rect(), 36–39

- `element_text()`, 36–39
- `fit_col_widths`, 28
- `fonts_available`, 29
- `full_seq_dates`, 29
- `full_seq_times`, 30
- `get_value_sets`, 31
- `gg_find_webfonts`, 31
- `gg_formatted_table`, 32
- `gg_hide_legend`, 32
- `gg_hide_X_axis`, 33
- `gg_hide_Y_axis`, 33
- `gg_label_size`, 33
- `gg_layer`, 34
- `gg_narrow`, 35
- `gg_pedantic`, 35
- `gg_resize_legend`, 40
- `gg_save_as`, 41
- `gg_set_size_defaults`, 42
- `gg_set_X_angle`, 43
- `gg_simple_table`, 43
- `gg_tiny_theme`, 44
- `gg_watermark`, 44
- `ggplot2::scale_fill_manual`, 84, 87
- `ggplot2::theme`, 36
- `glyphmap`, 45
- `guides()`, 85, 88
- `here`, 45
- `html_pdf_converter`, 46
- `hux_auto_widths`, 46
- `hux_bind_rows`, 47
- `hux_default_layout`, 47
- `hux_insert_start`, 48
- `hux_nest_group`, 48
- `hux_save_as`, 49
- `hux_set_caption`, 49
- `hux_set_font`, 50
- `hux_set_footer`, 50
- `hux_sprintf`, 51
- `hux_tidy`, 51
- `hux_to_ggplot`, 52
- `integer`, 26, 66, 73, 81
- `intersecting_group_by`, 53
- `knit_print.rendered_plot`, 53
- `knit_print.rendered_table`, 54
- `knit_versioned`, 54
- `lambda`, 85, 88
- `logit_trans`, 55
- `map()`, 57, 63
- `map2_struct`, 57
- `map_struct`, 56
- `margin()`, 36–38
- `modify()`, 56
- `non_cran`, 57
- `optional_fn`, 58
- `options`, 11, 12
- `outputter`, 59
- `pbeta`, 26, 66, 74, 81
- `pbeta2`, 60
- `percent_trans`, 61
- `pgamma`, 26, 65, 73, 81
- `pgamma2`, 61
- `plnorm2`, 62
- `pmap_struct`, 63
- `pmixnorm`, 64
- `pnbinom2`, 65
- `print.rendered_plot`, 66
- `print.rendered_table`, 67
- `purrr::map()`, 56
- `pwedge`, 67
- `qbeta2`, 68
- `qgamma2`, 69
- `qlnorm2`, 70
- `qmixnorm`, 71
- `qnbinom2`, 72
- `qwedge`, 74
- `rbern`, 75
- `rbeta2`, 75
- `rcategorical`, 76
- `rebuild_fonts`, 77
- `rep.struct_list(subset-struct-list)`, 95
- `reset_fonts`, 77
- `rgamma2`, 78
- `rlnorm2`, 79
- `rnbinom2`, 80
- `rowwise_mutate`, 82
- `rwedge`, 83

scale\_colour\_consistent, [84](#)  
scale\_colour\_subtype, [86](#)  
scale\_fill\_consistent, [87](#)  
scale\_fill\_subtype, [89](#)  
scale\_x\_log1p, [90](#)  
scale\_x\_logit, [91](#)  
scale\_x\_percent, [91](#)  
scale\_y\_log1p, [92](#)  
scale\_y\_logit, [92](#)  
scale\_y\_percent, [93](#)  
Splicing, [36](#)  
sprintf\_list, [93](#)  
stats::dbeta(), [23](#)  
stats::dgamma(), [23](#)  
stats::pbeta(), [60](#)  
stats::pgamma(), [62](#)  
stats::qbeta(), [69](#)  
stats::qgamma(), [70](#)  
stats::rbeta(), [76](#)  
stats::rgamma(), [78](#)  
std\_size, [94](#)  
struct\_flatten, [94](#)  
subset-struct-list, [95](#)  
summarise\_with\_totals, [96](#)  
svglite::svglite, [41](#)  
  
theme\_grey(), [40](#)  
time\_to\_date, [97](#)  
  
unit(), [36](#), [37](#)  
unstable, [98](#)  
utils::download.file, [11](#)  
  
wedge, [98](#)