

Package: papersize (via r-universe)

October 16, 2026

Type Package

Title Sizing Plots and Files for Paper

Version 0.1.1

Maintainer Eli Pousson <eli.pousson@gmail.com>

Description A set of convenience functions extending grid, ggplot2, and patchwork to help you size plots and files for printing to paper, postcards, playing cards, and other physical media.

License MIT + file LICENSE

URL <https://github.com/elipousson/papersize>,
<https://elipousson.github.io/papersize/>

BugReports <https://github.com/elipousson/papersize/issues>

Depends R (>= 2.10)

Imports cli (>= 3.4.0), cliExtras (>= 0.1.0), glue, grDevices, grid, lifecycle, rlang, units, utils, vctrs

Suggests covr, filenamr, ggplot2 (>= 4.0.0), gridExtra, janitor, magick, patchwork, qpdf, roxygen2, sf, testthat (>= 3.0.0), withr

Remotes elipousson/cliExtras, elipousson/filenamr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libudunits2-dev

Repository <https://elipousson.r-universe.dev>

Date/Publication 2026-09-16 14:37:53 UTC

RemoteUrl <https://github.com/elipousson/papersize>

RemoteRef HEAD

RemoteSha 31165a41f69be0ca7aa5dc9a3b2ec03ebb64cf36

Contents

annotate_wave	2
area_unit_options	4
as_asp	4
as_page	6
as_thickness	7
as_unit	8
card_sizes	11
convert_dist_scale	11
convert_dist_units	13
dist_unit_options	14
dist_units	14
get_page_size	15
get_scale	17
get_social_size	18
ggsave_ext	18
grid_units	22
inset_page_element	22
is_dist_units	25
make_contact_sheets	26
make_page_size	28
margins	29
page_extras	30
page_layout	31
page_to_layout	35
page_to_viewport	36
paper_sizes	38
plot_band	39
plot_band_dims	41
plot_band_page	42
plot_cards	46
print_to_page	47
set_page_dims	49
set_page_margin	50
standard_scales	51
theme_page	52
Index	58

annotate_wave	<i>Annotate a plot with a wavy (sine) line</i>
---------------	--

Description

`annotate_wave()` draws a wavy line as a ggplot2 annotation layer, built with `ggplot2::annotate()` and `ggplot2::geom_line()`. Useful for decorative dividers, underlines, or water/wave motifs.

Usage

```
annotate_wave(
  xmin,
  xmax,
  y = 0,
  amplitude = 1,
  period = 2,
  phase = 0,
  n = 200,
  color = "black",
  linewidth = 1,
  lineend = "round",
  linestyle = c("wave", "zigzag"),
  ...
)
```

Arguments

<code>xmin, xmax</code>	Horizontal range the wave spans.
<code>y</code>	Baseline (vertical center) the wave oscillates around, Default: 0
<code>amplitude</code>	Height of the wave above and below <code>y</code> , Default: 1
<code>period</code>	Horizontal distance the wave takes to complete one full cycle, Default: 2
<code>phase</code>	Horizontal shift of the wave, in radians, Default: 0
<code>n</code>	Number of points used to draw the curve. Higher values produce a smoother "wave" curve. Ignored for "zigzag", Default: 200
<code>color, linewidth, lineend</code>	Aesthetics passed to <code>ggplot2::geom_line()</code> , Default: "black", 1, and "round"
<code>linestyle</code>	Either "wave" for a smooth sine curve or "zigzag" for an angular zigzag (triangle wave) between the same peaks and troughs, Default: "wave"
<code>...</code>	Additional arguments passed to <code>ggplot2::annotate()</code> .

Value

A ggplot2 annotation layer that can be added to a ggplot object.

Examples

```
## Not run:
if (interactive() && is_installed("ggplot2")) {
  ggplot2::ggplot() +
    annotate_wave(xmin = 0, xmax = 10, amplitude = 0.5, period = 2.2) +
    annotate_wave(
      xmin = 0,
      xmax = 10,
      y = -2,
      amplitude = 0.5,
```

```

        period = 2.2,
        linestyle = "wiggle"
    )
}

## End(Not run)

```

area_unit_options	<i>Area units (vector)</i>
-------------------	----------------------------

Description

A vector of supported area units derived from `dist_units` and `units::valid_udunits()`.

Usage

```
area_unit_options
```

Format

A character vector with 41 names, plural names, and aliases for area units.

as_asp	<i>Convert character string, page name, or page data.frame to numeric aspect ratio</i>
--------	--

Description

Convert a character vector with an aspect ratio to a numeric aspect ratio or get the aspect ratio for a given page that has an aspect ratio column.

Usage

```

as_asp(
  asp = NULL,
  page = NULL,
  orientation = NULL,
  flipped = FALSE,
  sep = ":",
  cols = c("width", "height"),
  ...
)

```

Arguments

asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or characters separated by a colon (e.g. "1:3"). If numeric, <code>as_esp()</code> returns the value of <code>asp</code> without modification.
page	If character, <code>page</code> is passed as the <code>name</code> parameter to <code>get_page_size()</code> .
orientation	Page orientation, Default: <code>NULL</code> . Supported options are "portrait", "landscape", or "square".
flipped	If <code>TRUE</code> , return aspect ratio of height to width (y / x), instead of width to height.
sep	Separator character to use if <code>asp</code> is a character vector. Defaults to ":".
cols	Length 2 character vector with column names for page dimensions. Defaults to <code>c("width", "height")</code> .
...	Arguments passed on to <code>get_page_size</code> name Page name, e.g. "letter", not case sensitive, Default: <code>NULL</code> width Page width in "in", "px" or "mm" units. Default: <code>NULL</code> height Page height in "in", "px" or "mm" units. Default: <code>NULL</code> reorient If <code>TRUE</code> and <code>orientation</code> is not <code>NULL</code>, flip width and height dimensions for any pages that do not match the provided orientation. Set <code>reorient = FALSE</code> to filter pages by orientation. type Page type, Options include "paper", "social", "postcard", "print", "card", or "screen". Default: <code>NULL</code> ignore.case If <code>FALSE</code>, filtering for page and type are case sensitive. Defaults to <code>TRUE</code>. arg.call Passed to <code>cli_abort()</code> to improve internal error messages. units Units to convert page dimensions to using <code>convert_unit_type()</code>. valueOnly Passed to <code>valueOnly</code> parameter of <code>grid::convertUnit()</code>. Defaults to <code>FALSE</code>.

Value

A numeric vector.

See Also

`isstatic::as_orientation()`

Examples

```
as_esp(8.5 / 11)

as_esp("11:17")

as_esp("3/2", sep = "/")

as_esp(page = "letter")
```

```
as_esp(page = "letter", orientation = "landscape")

as_esp(page = "letter", orientation = "portrait", flipped = TRUE)

as_esp(page = make_page_size(8.5, 11, "in"), flipped = TRUE)
```

as_page	<i>Coerce a character or named vector to a page data.frame using <code>get_page_size()</code> or <code>make_page_size()</code></i>
----------------	--

Description

Coerce a character or named vector to a page data.frame using `get_page_size()` or `make_page_size()`

Usage

```
as_page(x = NULL, ..., cols = c("width", "height"), class = "data.frame")
```

Arguments

x	A character vector passed to the name parameter of <code>get_page_size()</code> , a named list of vector passed to dims for <code>make_page_size()</code> , or a length 2 numeric vector passed as the width and height parameter to <code>make_page_size()</code> . If x is a character string that does not match any of the names in <code>paper_sizes</code> , x is passed as the name parameter to <code>make_page_size()</code>
...	Arguments passed on to <code>make_page_size</code>
width,height	Page width and height. Both are required, if asp is NULL. Default to NULL.
units	Units for width and height. Required unless units is included in dims or <code>require_units = FALSE</code> . Passed to <code>as_unit_type()</code> to validate.
asp	Aspect ratio. Required if only one of width or height are provided.
name	Optional name for paper size. Recommend avoiding duplication with existing names in <code>paper_sizes</code> .
dims	A list or data.frame that can be used to set width, height, units, and/or asp.
orientation	Page orientation, Default: NULL. Supported options are "portrait", "landscape", or "square". If width and height suggest a portrait orientation when orientation = "landscape", the dimensions are reversed so the page dimensions match the provided orientation.
require_units	If TRUE (default), units must be supplied (either directly or via dims) or the function errors. Set <code>require_units = FALSE</code> to allow units to be omitted, e.g. for use by <code>get_page_dims()</code> where a unitless width and height are valid input. If units is omitted and <code>require_units = FALSE</code> , the returned page has no units column.

valid_units	Character vector with name or symbols for valid units. Defaults to <code>NULL</code> but any other unit name or symbol, e.g. <code>"px"</code> , is permitted.
call	The execution environment of a currently running function, e.g. <code>call = caller_env()</code> . The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be <code>NULL</code> or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages .
cols	Column names to use for width and height columns. Defaults to <code>c("width", "height")</code> . Must be length 2 and the first value is always used as the width name and the second as the height.
class	Class of return object: <code>"data.frame"</code> (default) or <code>"list"</code> (only supported when input page size is a single row).

Value

A data.frame or list object with one or more page dimensions.

Examples

```
as_page(c(8.5, 11), units = "in")

as_page("letter")

as_page(
  list(name = "letter", w = 8.5, h = 11, units = "in"),
  cols = c("w", "h")
)
```

as_thickness

Calculate the thickness of a stack of cards or paper

Description

[as_thickness\(\)](#) estimates the thickness of a stack of `n` sheets of paper or cards, each `pt` in caliper thickness (in points, i.e. thousandths of an inch). Alternatively, supply a known thickness directly with `x`. Used by [plot_band\(\)](#) to size the folded sides of a band that wraps around a stack.

Usage

```
as_thickness(..., n = NULL, pt = 10, x = NULL, units = NULL)
```

Arguments

<code>...</code>	Not used.
<code>n</code>	Number of sheets or cards in the stack. Ignored if <code>x</code> is supplied. Either <code>n</code> or <code>x</code> must be supplied.
<code>pt</code>	Caliper thickness of a single sheet or card, in points (1/1000 in). Ignored if <code>x</code> is supplied. Default: 10, a typical thickness for a playing card.
<code>x</code>	Optional. A known stack thickness (numeric or unit object). If supplied, <code>n</code> and <code>pt</code> are ignored.
<code>units</code>	A character vector specifying the units for the corresponding numeric values.

Value

A **unit** object with the thickness of the stack.

Examples

```
as_thickness(n = 54)

as_thickness(x = 0.75, units = "in")
```

as_unit

Helper functions for grid units

Description

- `as_unit()`: Convert to a unit (allowing unit objects as units)
- `as_unit_type()`: Convert to unit type (or checking unit types)
- `convert_unit_type()`: Convert `x` from one unit type to another (preserving names for named vectors)
- `is_unit_type()`: Is `x` a character vector with a unit type supported by the grid package or a unit object with a supported type?
- `is_same_unit_type()`: Are `x` and `y` the same unit type?

Note, when `as_unit_type()` is used on a margin object, it returns the unique unit type as a length 1 character vector not a length 4 character vector as you could expect with other length 4 input objects.

Usage

```
as_unit(
  x,
  units = NULL,
  data = NULL,
  recurse = FALSE,
  arg = caller_arg(x),
```

```

    call = parent.frame()
  )

as_unit_type(
  x,
  recurse = FALSE,
  data = NULL,
  valid_units = NULL,
  arg = caller_arg(x),
  call = parent.frame()
)

convert_unit_type(
  x,
  from = NULL,
  to = NULL,
  typeFrom = "dimension",
  valueOnly = FALSE,
  ...
)

is_unit_type(x, ...)

is_same_unit_type(x, y, recurse = FALSE, data = NULL, valid_units = NULL)

```

Arguments

x	A numeric vector. For <code>is.unit</code> , any R object.
units	A character vector specifying the units for the corresponding numeric values.
data	This argument is used to supply extra information for special <code>unit</code> types.
recurse	Whether to recurse into complex units.
arg	Passed to <code>cli_abort()</code> to improve internal error messages.
call	The execution environment of a currently running function, e.g. <code>call = caller_env()</code> . The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be <code>NULL</code> or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages .
valid_units	Character vector with name or symbols for valid units. Defaults to <code>NULL</code> but any other unit name or symbol, e.g. "px", is permitted.
from	Unit to convert from. If <code>NULL</code> and <code>x</code> is not a units object, convert to <code>to</code> units with a warning.

to	Unit to convert to. Passed to unitTo parameter of <code>grid::convertUnit()</code> . If NULL, return x as is.
typeFrom	Passed to typeFrom parameter of <code>grid::convertUnit()</code> . Defaults to "dimension".
valueOnly	Passed to valueOnly parameter of <code>grid::convertUnit()</code> . Defaults to FALSE.
...	Arguments passed on to <code>grid::convertUnit</code>
	axisFrom Either "x" or "y" to indicate whether the unit object represents a value in the x- or y-direction.
	axisTo Same as axisFrom , but applies to the unit object that is to be created.
	typeTo Same as typeFrom , but applies to the unit object that is to be created.
y	Object to compare to x.

Value

A `unit` class object, a character vector with a unit type, or a logical vector.

Examples

```

inch <- as_unit(1, "in")

inch

as_unit(10, inch)

as_unit(inch, "cm")

as_unit(inch)

convert_unit_type(inch, to = "cm")

convert_unit_type(c(10, 100), from = "mm", to = "cm")

is_unit_type("inch")

is_unit_type("inches")

is_same_unit_type(inch, "in")

is_same_unit_type("pt", "points")

```

card_sizes	<i>Standard card sizes</i>
------------	----------------------------

Description

Reference table of common playing card sizes for `get_card()`. Data is a subset of `paper_sizes` which is also included with `{sfext}` package.

Usage

```
card_sizes
```

Format

A data frame with 5 rows and 6 variables:

name Name of card
units Units ("in" or "mm") for dimensions
width Width in units
height Height in units
orientation Portrait (width less than height), landscape, or square

convert_dist_scale	<i>Convert distance from scale to actual units</i>
--------------------	--

Description

This function converts scale distances to actual units based on named `standard_scales`.

Usage

```
convert_dist_scale(
  dist = NULL,
  scale = NULL,
  standard = NULL,
  series = NULL,
  scale_unit = "in",
  scale_factor = NULL,
  actual_unit = NULL,
  dpi = 120,
  paper = NULL,
  orientation = NULL,
  ...
)
```

Arguments

dist	distance to convert. If paper is provided, dist is optional and paper width and height are used as dist.
scale	Scale name from <code>standard_scales[["scale"]]</code> .
standard	Scale standard. Options include "USGS", "Engineering", or "Architectural".
series	Map series from <code>standard_scales[["series"]]</code> . Series is only available for USGS scales.
scale_unit	"mm" (converted to cm by dividing by 10), "cm", "px" (converted to inches by dividing by dpi), or "in".
scale_factor	factor for converting from scale_unit to actual_unit, e.g. if 1" = 1', the scale factor is 12. optional if scale is provided; defaults to NULL.
actual_unit	any unit supported by <code>convert_dist_units()</code>
dpi	dots per square inch (used as conversion factor for "px" to "in")
paper	Name of paper passed to <code>get_paper()</code>
orientation	Page orientation, Default: NULL. Supported options are "portrait", "landscape", or "square".
...	Arguments passed on to <code>get_paper</code>
name	Page name, e.g. "letter", not case sensitive, Default: NULL
width	Page width in "in", "px" or "mm" units. Default: NULL
height	Page height in "in", "px" or "mm" units. Default: NULL
reorient	If TRUE and orientation is not NULL, flip width and height dimensions for any pages that do not match the provided orientation. Set <code>reorient = FALSE</code> to filter pages by orientation.
type	Page type, Options include "paper", "social", "postcard", "print", "card", or "screen". Default: NULL
ignore.case	If FALSE, filtering for page and type are case sensitive. Defaults to TRUE.
page	Used by <code>get_page_dims()</code> , page is either a character vector passed to the name parameter of <code>get_page_size()</code> , a data.frame with column names matching the cols parameter, or a length 2 numeric vector with the page width and height.
cols	Length 2 character vector with column names for page dimensions. Defaults to c("width", "height").
arg.call	Passed to <code>cli_abort()</code> to improve internal error messages.
units	Units to convert page dimensions to using <code>convert_unit_type()</code> .
valueOnly	Passed to valueOnly parameter of <code>grid::convertUnit()</code> . Defaults to FALSE.

Value

- If paper is not provided, return a vector of dist values converted from scale_unit to actual_unit based on scale_factor or information from `standard_scales` data.
- If paper is provided, return a data.frame with converted distances appended as columns named actual_width and actual_height.

See Also

Other dist: `convert_dist_units()`, `is_dist_units()`

<code>convert_dist_units</code>	<i>Convert distance (and area) values between different units</i>
---------------------------------	---

Description

Convert distance (and area) values between different units

Usage

```
convert_dist_units(
  dist,
  from = NULL,
  to = "meter",
  drop = FALSE,
  digits = NULL
)
```

Arguments

<code>dist</code>	Numeric or units object
<code>from</code>	Existing unit for dist, Default: NULL. If dist is a units object, the numerator is used as "from"
<code>to</code>	Unit to convert distance to, Default: 'meter'
<code>drop</code>	If TRUE, return numeric. If FALSE, return class units object.
<code>digits</code>	Number of digits to include in result; defaults to NULL.

Value

Object created by `units::set_units()`

See Also

`is_same_unit_type()`

Other dist: `convert_dist_scale()`, `is_dist_units()`

Examples

```
convert_dist_units(1, from = "mile", to = "km")

convert_dist_units(3, from = "ft", to = "yard")

mile <- units::set_units(1, "mi")

convert_dist_units(mile, to = "feet")
```

```
is_same_units(mile, "mile")
```

dist_unit_options	<i>Distance units (vector)</i>
-------------------	--------------------------------

Description

A vector of supported distance units pulled from `dist_units`.

Usage

```
dist_unit_options
```

Format

A character vector with 86 names, plural names, aliases, and symbols for distance units.

dist_units	<i>Distance units (data frame)</i>
------------	------------------------------------

Description

A subset of units supported by the units package accessible through the `units::valid_udunits()` function.

Usage

```
dist_units
```

Format

A data frame with 33 rows and 12 variables:

- `symbol` symbols
- `symbol_aliases` symbol aliases
- `name_singular` singular names
- `name_singular_aliases` singular name aliases
- `name_plural` character plural names
- `name_plural_aliases` plural name aliases
- `def` short definition
- `definition` definition
- `comment` comment
- `dimensionless` logical indicator for dimensionless units
- `source_xml` source XML
- `unit_opts` character vector with symbols, singular, and plural names for the unit

get_page_size	<i>Get a paper or card size based on name, dimensions, orientation, or type</i>
---------------	---

Description

- `get_page_size()` filters `paper_sizes` by one or more variables with option to reorient page dimensions or convert page units.
- `get_paper()` is equivalent to `get_page_size()` without the option to set units, type, or reorient parameters.
- `get_card()` is equivalent to `get_paper()` with `type = "card"` and the string "card" attached to the end of any provided name value supporting both "Poker card" and "Poker" as a valid name.
- `get_page_dims()` returns the width and height of a single page.
- `convert_page_units()` uses `convert_unit_type()` to convert the unit used for page dimensions.

Usage

```
get_page_size(
  name = NULL,
  width = NULL,
  height = NULL,
  orientation = NULL,
  reorient = TRUE,
  units = NULL,
  type = NULL,
  ignore.case = TRUE
)
```

```
get_paper(name = NULL, width = NULL, height = NULL, orientation = NULL)
```

```
get_card(name = NULL, width = NULL, height = NULL, orientation = NULL)
```

```
get_page_dims(
  page = NULL,
  width = NULL,
  height = NULL,
  orientation = NULL,
  cols = c("width", "height"),
  arg = caller_arg(page),
  call = parent.frame(),
  ...
)
```

```
convert_page_units(
```

```

    page,
    units = NULL,
    valueOnly = TRUE,
    cols = c("width", "height"),
    ...
)

```

Arguments

name	Page name, e.g. "letter", not case sensitive, Default: NULL
width	Page width in "in", "px" or "mm" units. Default: NULL
height	Page height in "in", "px" or "mm" units. Default: NULL
orientation	Page orientation, Default: NULL. Supported options are "portrait", "landscape", or "square".
reorient	If TRUE and orientation is not NULL, flip width and height dimensions for any pages that do not match the provided orientation. Set reorient = FALSE to filter pages by orientation.
units	Units to convert page dimensions to using convert_unit_type() .
type	Page type, Options include "paper", "social", "postcard", "print", "card", or "screen". Default: NULL
ignore.case	If FALSE, filtering for page and type are case sensitive. Defaults to TRUE.
page	Used by get_page_dims() , page is either a character vector passed to the name parameter of get_page_size() , a data.frame with column names matching the cols parameter, or a length 2 numeric vector with the page width and height.
cols	Length 2 character vector with column names for page dimensions. Defaults to c("width", "height").
arg, call	Passed to cli_abort() to improve internal error messages.
...	Additional parameters passed by get_page_dims() to get_page_size() if page is a character object.
valueOnly	Passed to valueOnly parameter of grid::convertUnit() . Defaults to FALSE.

Value

A data.frame with **page**, **paper**, or **card** name and dimensions.

[get_page_dims\(\)](#) returns a length 2 numeric vector with **page**'s width and height. If **page** resolved to a data.frame with a **units** column, that unit is also attached as a "**units**" attribute (NULL otherwise, e.g. when **page** was given only as bare **width/height** numbers) — check it before comparing dimensions from two different [get_page_dims\(\)](#) calls, which aren't guaranteed to share units.

See Also

```

make\_page\_size\(\)
get\_social\_size\(\)

```

Examples

```
get_paper("letter")

get_paper("letter", orientation = "landscape")

get_page_size(orientation = "square", reorient = FALSE)

get_page_size("ledger", units = "cm")

get_card("Tarot")

get_page_dims(get_paper("letter"))

convert_page_units(get_paper("letter"), units = "cm")
```

get_scale	<i>Get standard scales and convert to scale distances</i>
-----------	---

Description

This function returns a scale from [standard_scales](#) based on a provided name, standard, and/or series.

Usage

```
get_scale(scale = NULL, standard = NULL, series = NULL)
```

Arguments

scale	Scale name from <code>standard_scales[["scale"]]</code> .
standard	Scale standard. Options include "USGS", "Engineering", or "Architectural".
series	Map series from <code>standard_scales[["series"]]</code> . Series is only available for USGS scales.

Value

A tibble based on [standard_scales](#) with rows filtered to values that match parameters.

<code>get_social_size</code>	<i>Get social media image size to match platform and format</i>
------------------------------	---

Description

See `paper_sizes[paper_sizes$type == "social",]$name` for support image options.

Usage

```
get_social_size(
  name = NULL,
  platform = NULL,
  format = NULL,
  orientation = NULL
)
```

Arguments

<code>name</code>	Image size name, Default: <code>NULL</code>
<code>platform</code>	Social media platform, "Instagram", "Facebook", or "Twitter", Default: <code>NULL</code>
<code>format</code>	Image format, "post", "story", or "cover", Default: <code>NULL</code>
<code>orientation</code>	Image orientation, Default: <code>NULL</code> .

See Also

[`get\_page\_size\(\)`](#)

Examples

```
get_social_size("Instagram post")

get_social_size(platform = "Twitter")

get_social_size(format = "cover")
```

<code>ggsave_ext</code>	<i>Save a ggplot2 plot to file and update file EXIF metadata</i>
-------------------------	--

Description

Save a plot or map then update the EXIF metadata for the title, author, and create data. `ggsave_ext()` also supports creating a file name based on a sentence case name with spaces (e.g. "Baltimore city map") and appending a label (e.g. "baltcity") as a prefix to the output file name.

`map_ggsave_ext()` can take a list of {ggplot2} plots (e.g. a list of plot generated by `purrr::map()`) and create multiple files using the same parameters or use the {gridExtras} package to create a single combined PDF file.

Usage

```
ggsave_ext(
  plot = ggplot2::last_plot(),
  name = NULL,
  label = NULL,
  prefix = NULL,
  postfix = NULL,
  increment = NULL,
  filename = NULL,
  device = NULL,
  fileext = NULL,
  filetype = NULL,
  path = NULL,
  paper = NULL,
  orientation = NULL,
  width = NULL,
  height = NULL,
  asp = NULL,
  units = getOption("papersize.ggsave_units", "in"),
  scale = 1,
  dpi = 300,
  bgcolor = NULL,
  exif = FALSE,
  title = NULL,
  author = NULL,
  keywords = NULL,
  args = NULL,
  overwrite = TRUE,
  ask = FALSE,
  preview = FALSE,
  limitsize = TRUE,
  quiet = FALSE,
  ...
)

ggsave_social(
  plot = ggplot2::last_plot(),
  image = "Instagram post",
```

```

platform = NULL,
format = NULL,
orientation = NULL,
name = NULL,
filename = NULL,
fileext = "jpeg",
filetype = NULL,
dpi = 72,
width = 1080,
height = 1080,
units = "px",
...
)

map_ggsave_ext(
  plot,
  name = NULL,
  label = NULL,
  prefix = NULL,
  postfix = "pg_",
  filename = NULL,
  device = NULL,
  fileext = NULL,
  filetype = NULL,
  path = NULL,
  overwrite = TRUE,
  ...,
  single_file = TRUE,
  onefile = TRUE
)

```

Arguments

plot	Plot to save, defaults to last plot displayed. If plot is an "magick-image" class object, it is converted to a plot using <code>magick::image_ggplot()</code>
name	Plot name, used to create filename (if filename is NULL) using <code>filenamr::make_filename()</code>
label	Label to combine with name converted to snake case with <code>janitor::make_clean_names()</code> . The label is designed to identify the area or other shared characteristics across multiple data files, maps, or plots. label is ignored if name is NULL or if name includes a file extension.
prefix	File name prefix. "date" adds a date prefix, "time" adds a date/time prefix; defaults to NULL.
postfix	File name postfix; defaults to NULL.
increment	If TRUE, increment digits in string by 1. If numeric, increment digits in string by value. If NULL, 0, or if no digits are present in string, return string as is.
filename	File name to create on disk.

device	Device to use. Can either be a device function (e.g. png), or one of "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only). If NULL (default), the device is guessed based on the filename extension.
fileext	File type or extension. Optional if filename or path include a file extension.
filetype	File type (used if fileext is NULL).
path	Path of the directory to save plot to: path and filename are combined to create the fully qualified file name. Defaults to the working directory.
paper	Paper matching name from paper_sizes (e.g. "letter"). Not case sensitive.
orientation	Page orientation ("portrait", "landscape", or "square").
width, height	Plot size in units expressed by the units argument. If not supplied, uses the size of the current graphics device.
asp	Numeric aspect ratio used to determine width or height if only one of the two arguments is provided; defaults to NULL.
units	One of the following units in which the width and height arguments are expressed: "in", "cm", "mm" or "px".
scale	Multiplicative scaling factor.
dpi	Plot resolution. Also accepts a string input: "retina" (320), "print" (300), or "screen" (72). Only applies when converting pixel units, as is typical for raster output types.
bgcolor	Background color to optionally override plot.background theme element.
exif	If TRUE, the EXIF metadata for the exported file is updated with the exiftr package; defaults to FALSE.
title	Title to add to file metadata with exiftoolr, Default: NULL.
author	Author to add to file metadata to the "Author" and "XMP-dc:creator" tags. Default: NULL.
keywords	Keyword(s) added to file metadata to "IPTC:Keywords" and "XMP-dc:Subject" tags. Defaults to NULL.
args	Alternate arguments passed to exiftoolr::exif_call() . Other tag parameters are appended to args if they are not NULL.
overwrite	If TRUE (default), overwrite any existing file with the same name or ask to overwrite if ask = TRUE. Passed to filenamr::check_file_overwrite() .
ask	If TRUE, ask before overwriting file with the same name. Defaults to FALSE. Passed to filenamr::check_file_overwrite() .
preview	If TRUE, open saved file in default system application. Based on ggpreview from tjmisc package.
limitsize	When TRUE (the default), ggsave() will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.
quiet	If TRUE (default), suppress function messages.
...	Arguments passed on to ggplot2::ggsave

`create.dir` Whether to create new directories if a non-existing directory is specified in the `filename` or `path` (TRUE) or return an error (FALSE, default). If FALSE and run in an interactive session, a prompt will appear asking to create a new directory when necessary.

`image` Image name passed to name parameter of `get_social_size()`.

`platform` Social media platform, "Instagram", "Facebook", or "Twitter", Default: NULL

`format` Image format, "post", "story", or "cover", Default: NULL

`single_file, onefile` If TRUE, use `gridExtra::arrangeGrob()` to create an arrangelist class object that `ggplot2::ggsave()` can save as a single multi-page file. Note: this does not work with plots modified with patchwork including inset maps created with the `maplayer::layer_inset()` function.

See Also

`ggplot2::ggsave()`

<code>grid_units</code>	<i>Grid units (vector)</i>
-------------------------	----------------------------

Description

A vector of units supported by `grid::unit()`

Usage

`grid_units`

Format

A character vector with 34 abbreviated, singular, and plural unit names.

<code>inset_page_element</code>	<i>Create an inset with page size dimensions be added on top of the previous plot</i>
---------------------------------	---

Description

Create an inset with page size dimensions be added on top of the previous plot

Usage

```
inset_page_element(
  p,
  inset_page = NULL,
  left = NULL,
  bottom = NULL,
  right = NULL,
  top = NULL,
  align_to = "panel",
  on_top = TRUE,
  clip = TRUE,
  ignore_tag = FALSE,
  ...
)
```

Arguments

p	A grob, ggplot, patchwork, formula, raster, nativeRaster, or gt object to add as an inset
inset_page	Page size data.frame to use for inset, Default: NULL
left, bottom, right, top	numerics or units giving the location of the outer bounds. If given as numerics and inset_page is NULL, they will be converted to npc units. All four are required if inset_page is NULL. If inset_page is provided, top <i>or</i> bottom and left <i>or</i> right must be provided as the inset element is expected to be the width and height defined by inset_page.
align_to	Specifies what left, bottom, etc should be relative to. Either 'panel' (default), 'plot', or 'full'.
on_top	Logical. Should the inset be placed on top of the other plot or below (but above the background)?
clip	Logical. Should clipping be performed on the inset?
ignore_tag	Logical. Should autotagging ignore the inset?
...	Arguments passed on to <code>convert_unit_type</code>

arg Passed to `cli_abort()` to improve internal error messages.
valid_units Character vector with name or symbols for valid units. Defaults to NULL but any other unit name or symbol, e.g. "px", is permitted.
from Unit to convert from. If NULL and x is not a units object, convert to to units with a warning.
to Unit to convert to. Passed to unitTo parameter of `grid::convertUnit()`. If NULL, return x as is.
typeFrom Passed to typeFrom parameter of `grid::convertUnit()`. Defaults to "dimension".
valueOnly Passed to valueOnly parameter of `grid::convertUnit()`. Defaults to FALSE.

y Object to compare to **x**.

x A numeric vector.
For **is.unit**, any R object.

units A character vector specifying the units for the corresponding numeric values.

data This argument is used to supply extra information for special **unit** types.

call The execution environment of a currently running function, e.g. **call = caller_env()**. The corresponding function call is retrieved and mentioned in error messages as the source of the error.
You only need to supply **call** when throwing a condition from a helper function which wouldn't be relevant to mention in the message.
Can also be **NULL** or a [defused function call](#) to respectively not display any call or hard-code a code to display.
For more information about error calls, see [Including function calls in error messages](#).

recurse Whether to recurse into complex units.

Value

A **inset_path** object

See Also

[patchwork::inset_element\(\)](#)
[inset_page\(\)](#)

Examples

```
## Not run:
if (interactive() && is_installed("ggplot2")) {
  library(ggplot2)
  p <- ggplot(mpg, aes(displ, fill = class)) +
    geom_bar()

  ggplot(mpg, aes(displ, hwy, colour = class)) +
    geom_point() +
    inset_page_element(
      p = p,
      inset_page = get_page_size("Poker card", orientation = "landscape"),
      left = unit(1, "in"),
      bottom = unit(1, "in")
    )
}

## End(Not run)
```

is_dist_units

General utility functions for working with distance units objects

Description

- `is_dist_units()`: Is x a units object with a units attribute in `dist_unit_options` or `area_unit_options`?
- `get_dist_units()`: Get the distance units from x (if x is a sf or units objects or a character string from `dist_unit_options`)
- `as_dist_units()`: Convert x to units using `units::as_units`
- `is_same_units()`: Do x and y have the same distance units attribute? Names or symbols of valid distance units are allowed.

Usage

```
is_dist_units(x, arg = caller_arg(x))

get_dist_units(x, arg = caller_arg(x), call = parent.frame())

as_dist_units(x, units = NULL, arg = caller_arg(x), call = parent.frame())

is_same_units(x, y = NULL)
```

Arguments

<code>x, y</code>	objects to check
<code>arg</code>	Used internally and passed to <code>rlang::arg_match()</code> as error arg or used by <code>cli::cli_abort()</code> to improve error messages.
<code>call</code>	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.
<code>units</code>	Distance units to convert to. Must be one of <code>dist_unit_options</code> or <code>area_unit_options</code> .

See Also

Other dist: `convert_dist_scale()`, `convert_dist_units()`

Examples

```
mile <- units::set_units(1, "mi")

is_dist_units("mi")

is_dist_units(mile)
```

```

is_same_units(mile, "mile")

get_dist_units(mile)

as_dist_units(1, "mi")

as_dist_units(2, mile)

```

`make_contact_sheets` *Use `magick::image_ggplot()` to make contact sheets for images*

Description

[Experimental] Wraps `filenamr::read_exif()`, `magick::image_ggplot()`, and `page_layout()` to create contact sheets for a folder of images.

Usage

```

make_contact_sheets(
  images,
  dims = NULL,
  ncol = NULL,
  nrow = NULL,
  captions = "{file_name}\n{date_created}",
  caption_size = 12,
  caption_position = "panel",
  image_margin = margins(0.1, unit = "in"),
  page = "letter",
  orientation = "portrait",
  image_max = NULL,
  image_fileext = NULL,
  tags = NULL,
  tz = NULL,
  save = FALSE,
  filename = NULL,
  ...
)

```

Arguments

<code>images</code>	File path or data.frame from <code>filenamr::read_exif()</code>
<code>dims</code>	Image dimensions in same dimensions as page. Required.
<code>ncol, nrow</code>	The dimensions of the grid to create. If both are <code>NULL</code> , <code>dims</code> will be used or <code>dims</code> will be determined based on the plot dimensions.

<code>captions</code>	Template for caption, passed to <code>glue::glue_data()</code> using the images data.frame as <code>.x</code> . Note that this template may vary if you are using a custom tags parameter or modify the "filenamr.exif_xwalk" option. See <code>filenamr::read_exif()</code> for more details. Default: "{file_name}\n{date_created}"
<code>caption_size</code>	Caption size, passed to <code>ggplot2::element_text()</code> for plot.caption for theme, Default: 12
<code>caption_position</code>	Caption position, passed to plot.caption.position for theme, Default: 'panel'
<code>image_margin</code>	Image margin passed Default: <code>margins(0.1, unit = "in")</code>
<code>page</code>	Paper name or a data.frame with width and height columns. Optional if width and height are both provided, Default: NULL. If <code>ncol</code> and <code>nrow</code> are also supplied, <code>page</code> is not used to determine the grid size — but if <code>margin</code> or <code>marks</code> are used, <code>page</code> must still be set to the exact final output size (the same width/height you plan to pass to <code>ggplot2::ggsave()</code>), not just the combined size of the grid of plots. See <code>margin</code> and <code>marks</code> for why this matters.
<code>orientation</code>	Paper orientation, Optional if width and height are both provided, Default: 'landscape'
<code>image_max</code>	Maximum number of images to use for contact sheets.
<code>image_fileext</code>	Passed to fileext parameter of <code>filenamr::read_exif()</code> , Default: NULL
<code>tags</code>	List of EXIF tags to read from files. If NULL (default), set to option "filenamr.exif_tags" or default <code>default_exif_tags</code> .
<code>tz</code>	Time zone to pass to <code>lubridate::ymd_hms()</code> if <code>format_exif</code> is TRUE. Typically set to <code>Sys.timezone()</code> to convert date/time columns.
<code>save</code>	If TRUE, save contact sheet to a file. filename may be required if save is TRUE. Default: FALSE
<code>filename</code>	File name to create on disk.
<code>...</code>	Additional parameters passed to <code>map_ggsave_ext()</code> excluding width, height, and units.

Value

A list of patchwork object or (if `save = TRUE`) invisibly return the list and save a file.

See Also

`filenamr::read_exif()` `glue::glue()` `magick::editing()`, `magick::image_ggplot()` `ggplot2::labs()`, `ggplot2::theme()`, `ggplot2::margin()`

make_page_size	<i>Make a page size data frame</i>
----------------	------------------------------------

Description

Width and height or aspect ratio and either width or height are required. Units are also required. If orientation is provided, width and height may be reversed to match the provided orientation.

Usage

```
make_page_size(
  width = NULL,
  height = NULL,
  units = NULL,
  asp = NULL,
  orientation = NULL,
  name = NULL,
  dims = NULL,
  valid_units = NULL,
  cols = c("width", "height"),
  class = "data.frame",
  require_units = TRUE,
  call = caller_env()
)
```

Arguments

width, height	Page width and height. Both are required, if asp is NULL. Default to NULL.
units	Units for width and height. Required unless units is included in dims or <code>require_units = FALSE</code> . Passed to <code>as_unit_type()</code> to validate.
asp	Aspect ratio. Required if only one of width or height are provided.
orientation	Page orientation, Default: NULL. Supported options are "portrait", "landscape", or "square". If width and height suggest a portrait orientation when orientation = "landscape", the dimensions are reversed so the page dimensions match the provided orientation.
name	Optional name for paper size. Recommend avoiding duplication with existing names in <code>paper_sizes</code> .
dims	A list or data.frame that can be used to set width, height, units, and/or asp.
valid_units	Character vector with name or symbols for valid units. Defaults to NULL but any other unit name or symbol, e.g. "px", is permitted.
cols	Column names to use for width and height columns. Defaults to c("width", "height"). Must be length 2 and the first value is always used as the width name and the second as the height.

<code>class</code>	Class of return object: "data.frame" (default) or "list" (only supported when input page size is a single row).
<code>require_units</code>	If TRUE (default), <code>units</code> must be supplied (either directly or via <code>dims</code>) or the function errors. Set <code>require_units = FALSE</code> to allow <code>units</code> to be omitted, e.g. for use by <code>get_page_dims()</code> where a unitless width and height are valid input. If <code>units</code> is omitted and <code>require_units = FALSE</code> , the returned page has no units column.
<code>call</code>	The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be NULL or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages.

Value

A data.frame with columns named (by default) width, height, units, orientation, and asp or a list with those same names.

See Also

[get_page_size\(\)](#)

Examples

```
make_page_size(48, 24, "in", name = "Tabletop map")

make_page_size(8.5, 8.5, "in", name = "Trimmed letter")

make_page_size(5, asp = 1.25, units = "cm", class = "list")
```

margins

Specify the margins of a page or element

Description

An extended version of `ggplot2::margin()` with more flexibility in specification. If margin is a margin class object it is returned as is. If margin is length 1, the t, r, b, and l values are all set to that value. If margin is length 2, the t and b are set to half the first value and the r and l are set to half the second value. The ... parameters also allow you to use the same syntax as margin.

Usage

```

margins(margin = NULL, ..., unit = "in")

is_margin(x)

get_margin(margin = NULL, ..., unit = "in")

```

Arguments

margin	A numeric list or vector or a margin class object. For <code>get_margin()</code> only, margin can be a margin name from <code>page_extras[["margins"]]</code> . Defaults to NULL.
...	Additional numeric values combined with margin if provided.
unit	Default units for margins (ignored if margin is a margin class object). Passed to <code>as_unit_type()</code> so units class objects as well as unit names supported <code>grid::unit()</code> are allowed. Defaults to "in" except for <code>margin()</code> where unit defaults to "pt" to match <code>ggplot2::margin()</code> .
x	Object to check for class margin and unit

See Also

`set_page_margin()`

Examples

```

margins(1, unit = "in")

margins(c(2, 2), unit = "in")

margins(list(1, 1, 1.5, 1), unit = "cm")

margins(t = 1, r = 3, b = 1.5, l = 3, unit = "in")

standard_margin <- get_margin("standard", unit = "in")

is_margin(standard_margin)

standard_margin

```

Description

A named list of additional reference data that currently includes only one data.frame: a reference table of margin sizes in "in" and "cm".

Usage

```
page_extras
```

Format

A length 1 list.

page_layout	<i>Use patchwork to lay out a list of fixed aspect plots on a larger page</i>
-------------	---

Description

Use patchwork to lay out a list of fixed aspect plots on a larger page

Usage

```
page_layout(
  plots = NULL,
  page = NULL,
  width = NULL,
  height = NULL,
  orientation = "landscape",
  byrow = FALSE,
  guides = NULL,
  tag_level = NULL,
  design = NULL,
  paginate = TRUE,
  ncol = NULL,
  nrow = NULL,
  dims = NULL,
  gutter = NULL,
  margin = NULL,
  position = "top-left",
  unit = "in",
  marks = FALSE,
  images = FALSE,
  dpi = 120,
  widths = NULL,
  heights = NULL,
  call = caller_env()
)
```

Arguments

plots	Page name, a data.frame with width and height columns, or a list of ggplot2 objects with card plots. Default: NULL
-------	--

<code>page</code>	Paper name or a <code>data.frame</code> with width and height columns. Optional if width and height are both provided, Default: <code>NULL</code> . If <code>ncol</code> and <code>nrow</code> are also supplied, <code>page</code> is not used to determine the grid size — but if <code>margin</code> or <code>marks</code> are used, <code>page</code> must still be set to the exact final output size (the same width/height you plan to pass to <code>ggplot2::ggsave()</code>), not just the combined size of the grid of plots. See <code>margin</code> and <code>marks</code> for why this matters.
<code>width, height</code>	Paper width and height, Default: <code>NULL</code>
<code>orientation</code>	Paper orientation, Optional if width and height are both provided, Default: <code>'landscape'</code>
<code>byrow</code>	Analogous to <code>byrow</code> in <code>matrix()</code> . If <code>FALSE</code> the plots will be filled in in column-major order
<code>guides</code>	A string specifying how guides should be treated in the layout. <code>'collect'</code> will collect guides below to the given nesting level, removing duplicates. <code>'keep'</code> will stop collection at this level and let guides be placed alongside their plot. <code>auto</code> will allow guides to be collected if a upper level tries, but place them alongside the plot if not. If you modify default guide "position" with <code>theme(legend.position=...)</code> while also collecting guides you must apply that change to the overall patchwork (see example).
<code>tag_level</code>	A string (<code>'keep'</code> or <code>'new'</code>) to indicate how auto-tagging should behave. See <code>plot_annotation()</code> .
<code>design</code>	Specification of the location of areas in the layout. Can either be specified as a text string or by concatenating calls to <code>area()</code> together. See the examples for further information on use.
<code>paginate</code>	If <code>TRUE</code> , create a list of <code>patchwork</code> objects when the number of plots is greater than the number of spaces in the plot layout. Default to <code>TRUE</code> .
<code>ncol, nrow</code>	The dimensions of the grid to create. If both are <code>NULL</code> , dims will be used or dims will be determined based on the plot dimensions.
<code>dims</code>	Optional. Plot dimensions. Ignored if <code>ncol</code> and <code>nrow</code> are supplied. Otherwise, if <code>NULL</code> (default), dims are inferred based on the dimensions of the first plot in plots.
<code>gutter</code>	Optional. Spacing to add between plots in the grid, as a single number (used for both row and column spacing), a length-2 numeric vector <code>c(row, col)</code> , or a named vector or list with <code>row</code> and <code>col</code> elements. Interpreted in <code>unit</code> . Implemented by adding half of <code>gutter</code> to the interior-facing sides of each plot's own <code>plot.margin</code> (so two adjacent plots each contribute half, summing to the full gutter between them) based on its row/column position in the <code>ncol</code> x <code>nrow</code> grid — this replaces each plot's existing <code>plot.margin</code> . Plots on the outer edge of the grid are not padded on their outward-facing side; use <code>margin</code> for space around the outside of the whole grid. Default: <code>NULL</code> (no extra spacing). <code>gutter</code> changes the total size of the combined grid (adding $(ncol - 1) * col_gutter$ and $(nrow - 1) * row_gutter$), which <code>marks</code> accounts for automatically — the automatic <code>ncol/nrow</code> calculation from <code>page/dims</code> also reserves room for it, so fewer plots may fit per page than with <code>gutter = 0</code> .

<code>margin</code>	Optional. A margin to add around the outside of the combined grid of plots, e.g. so the grid can be centered on a larger sheet of paper when saved with <code>ggplot2::ggsave()</code> . Passed to <code>get_margin()</code> with <code>unit</code> . The margin pads the composed page rather than the individual plots in <code>plots</code> , and does not affect the number of rows and columns in the grid. Default: <code>NULL</code> , which computes a margin from <code>position</code> whenever plot dimensions are known (supplied via <code>dims</code> , or auto-detected from the first plot) — see <code>position</code> . Set <code>margin</code> explicitly to override that (or to add a margin when dimensions aren't known, e.g. <code>ncol/nrow</code> supplied without <code>dims</code>). <code>margin</code> itself always renders correctly regardless of <code>page</code> , because it is applied as a fixed absolute-unit margin around whatever canvas size <code>ggplot2::ggsave()</code> is eventually called with. <code>marks</code> , below, is the one that depends on <code>page</code> being set correctly — see <code>marks</code> .
<code>position</code>	Where to place plots on <code>page</code> when they don't fill it completely — e.g. fewer plots than the page has room for, or a remainder page when <code>paginate</code> splits a longer list. One of "top-left" (default), "top", "top-right", "left", "center", "right", "bottom-left", "bottom", or "bottom-right". Only takes effect when plot dimensions are known (see <code>margin</code>) and <code>margin</code> isn't supplied directly; the grid is first shrunk to just the rows/columns needed for the plots being placed (rather than the full page capacity), then the leftover page space is split into a <code>margin</code> that pushes that grid toward the requested anchor — "center" splits leftover space evenly on both axes, "top-left" (matching the pre-existing default behavior) assigns it all to the bottom/right, and so on.
<code>unit</code>	Unit used for <code>gutter</code> , and for <code>margin</code> if <code>margin</code> is a bare numeric vector or list (ignored for <code>margin</code> if it is a <code>unit</code> class object; <code>gutter</code> does not support <code>unit</code> class objects). Default: "in".
<code>marks</code>	If <code>TRUE</code> , add crop marks in the <code>margin</code> area showing where to cut the page into individual plots. Requires <code>margin</code> . Marks are placed assuming the grid of plots exactly fills the page after subtracting <code>margin</code> , i.e. the same assumption <code>margin</code> itself relies on. Default: <code>FALSE</code> . Unlike <code>margin</code> , <code>marks</code> reads <code>page</code> (via <code>get_page_dims()</code>) to work out where the margin area is, so <code>page</code> must equal the exact final output size — the same width/height you pass to <code>ggsave()</code> — even if <code>ncol/nrow</code> are supplied directly and <code>page</code> would otherwise be unused. Passing just the combined size of the grid of plots (i.e. <code>page</code> without <code>margin</code> added on) will place the marks in the wrong location, because <code>ggplot2::ggsave()</code> will render a larger canvas than <code>marks</code> was calculated for.
<code>images</code>	Not yet implemented. If <code>TRUE</code> and <code>dims</code> is <code>NULL</code> , the input plots are assumed to be plots created with <code>magick::image_ggplot()</code> and <code>dpi</code> is used to infer dimensions.
<code>dpi</code>	Not yet implemented. Resolution.
<code>widths, heights</code>	Optional. Column widths and row heights passed to <code>patchwork::wrap_plots()</code> . By default (<code>NULL</code>), and whenever plot dimensions are known (supplied via <code>dims</code> , or auto-detected from the first plot), each column/row is pinned to that exact size — so plots keep their true physical size instead of

`patchwork`'s own proportional division of space, which only happens to line up with each plot's true size when `plots` exactly fills every cell of the grid with same-sized content. Pass `widths/heights` explicitly (as `unit` objects, one per column/row) to override this.

call The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

A `patchwork` object or a list of `patchwork` objects.

See Also

`ggplot2::ggplot_build()` `patchwork::wrap_plots()`, `patchwork::plot_layout()`

Examples

```
page_layout(
  plots = plot_cards("Poker", 6),
  page = "letter"
)

# `page` must be the final output size (grid of 3x2 Poker cards, 7.5x7in,
# plus the 0.5/0.75in margin on each side = 8.5x8.5in), not just the size
# of the grid of plots - this is what ggsave(width, height) should match
page_layout(
  plots = plot_cards("Poker", 6),
  page = make_page_size(width = 8.5, height = 8.5, units = "in"),
  ncol = 3,
  nrow = 2,
  margin = margins(t = 0.75, r = 0.5, b = 0.75, l = 0.5, unit = "in"),
  marks = TRUE
)

# `gutter` adds spacing between plots, which grows the grid (3x2 Poker
# cards with a 0.1in gutter = 7.7x7.1in) - `page` (and `marks`) account
# for it automatically, so it still needs the full 8.7x8.6in page size
page_layout(
  plots = plot_cards("Poker", 6),
  page = make_page_size(width = 8.7, height = 8.6, units = "in"),
  ncol = 3,
  nrow = 2,
  gutter = 0.1,
  margin = margins(t = 0.75, r = 0.5, b = 0.75, l = 0.5, unit = "in"),
  marks = TRUE
)

# Fewer plots than the page holds a `position` (default "top-left") to
# decide where the shrunk-to-fit grid lands on the page
page_layout(
  plots = plot_cards("Poker", 1),
```

```

    page = "letter",
    position = "center"
  )

```

page_to_layout

Convert a page size data.frame to a Grid layout object

Description

Convert a page size data.frame to a Grid layout object

Usage

```

page_to_layout(
  page,
  margins = NULL,
  region = NULL,
  ncol = 1,
  nrow = 1,
  gutter = NULL,
  widths = NULL,
  heights = NULL,
  units = "in",
  respect = TRUE,
  just = "center",
  cols = c("width", "height")
)

```

Arguments

page	A page size data.frame from <code>get_page_size()</code> or a data.frame or list from <code>make_page_size()</code> .
margins	A numeric list or vector or a margin class object. Defaults to NULL. margins are removed from the overall layout width and height.
region	Optional. An additional page data.frame where the region width and height are used as the column width and row height.
ncol	An integer describing the number of columns in the layout.
nrow	An integer describing the number of rows in the layout.
gutter	Gutter width/height. Not yet implemented.
widths	A numeric vector or unit object describing the widths of the columns in the layout.
heights	A numeric vector or unit object describing the heights of the rows in the layout.
units	Passed to default.units parameter of <code>grid::grid.layout()</code> .

respect	A logical value or a numeric matrix. If a logical, this indicates whether row heights and column widths should respect each other. If a matrix, non-zero values indicate that the corresponding row and column should be respected (see examples below).
just	A string or numeric vector specifying how the layout should be justified if it is not the same size as its parent viewport. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: "left", "right", "centre", "center", "bottom", and "top". For numeric values, 0 means left alignment and 1 means right alignment. NOTE that in this context, "left", for example, means align the left edge of the left-most layout column with the left edge of the parent viewport.
cols	Column names to use for width and height columns. Defaults to c("width", "height"). Must be length 2 and the first value is always used as the width name and the second as the height.

Value

A Grid layout object with the same width and height and default units as the input page size.

Examples

```
a8_layout <- page_to_layout(
  get_paper("A8", orientation = "landscape"),
  ncol = 2,
  nrow = 2
)

grid::grid.show.layout(a8_layout)
```

page_to_viewport	<i>Convert a page data.frame to a viewport class object</i>
-------------------------	---

Description

Create a **viewport** class object with a width and height matching the dimensions of a page data.frame and default.units that match the page units.

Usage

```
page_to_viewport(page, name = NULL, cols = c("width", "height"), ...)
```

Arguments

page	A page data.frame from <code>get_page_size()</code> or <code>make_page_size()</code> .
name	A character value to uniquely identify the viewport once it has been pushed onto the viewport tree.
cols	Column names to use for width and height columns. Defaults to <code>c("width", "height")</code> . Must be length 2 and the first value is always used as the width name and the second as the height.
...	Arguments passed on to <code>grid::viewport</code>
x	A numeric vector or unit object specifying x-location.
y	A numeric vector or unit object specifying y-location.
just	A string or numeric vector specifying the justification of the viewport relative to its (x, y) location. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: <code>"left"</code> , <code>"right"</code> , <code>"centre"</code> , <code>"center"</code> , <code>"bottom"</code> , and <code>"top"</code> . For numeric values, 0 means left alignment and 1 means right alignment.
gp	An object of class <code>"gpar"</code> , typically the output from a call to the function <code>gpar</code> . This is basically a list of graphical parameter settings.
clip	One of <code>"on"</code> , <code>"inherit"</code> , or <code>"off"</code> , indicating whether to clip to the extent of this viewport, inherit the clipping region from the parent viewport, or turn clipping off altogether. For back-compatibility, a logical value of <code>TRUE</code> corresponds to <code>"on"</code> and <code>FALSE</code> corresponds to <code>"inherit"</code> . May also be a grob (or a <code>gTree</code>) that describes a clipping path or the result of a call to <code>as.path</code> .
mask	One of <code>"none"</code> (or <code>FALSE</code>) or <code>"inherit"</code> (or <code>TRUE</code>) or a grob (or a <code>gTree</code>) or the result of call to <code>as.mask</code> . This specifies that the viewport should have no mask, or it should inherit the mask of its parent, or it should have its own mask, as described by the grob.
xscale	A numeric vector of length two indicating the minimum and maximum on the x-scale. The limits may not be identical.
yscale	A numeric vector of length two indicating the minimum and maximum on the y-scale. The limits may not be identical.
angle	A numeric value indicating the angle of rotation of the viewport. Positive values indicate the amount of rotation, in degrees, anticlockwise from the positive x-axis.
layout	A Grid layout object which splits the viewport into subregions.
layout.pos.row	A numeric vector giving the rows occupied by this viewport in its parent's layout.
layout.pos.col	A numeric vector giving the columns occupied by this viewport in its parent's layout.
parent	A grid viewport object.
children	A <code>vpList</code> object.

Value

A viewport class object with the same width and height as the input page size.

Examples

```
vp <- page_to_viewport(get_paper("Poker card"))
grid::grid.show.viewport(vp)
```

paper_sizes

Standard paper and image sizes

Description

Reference table of standard paper, postcard, photo print, social media image sizes, and playing card sizes for `get_page_size()`. Derived from [visioguy/PaperSizes](#) repo, [Adobe UK guide to photo sizes](#) and other sources. Data is identical to data included with `{sfext}` package.

Usage

```
paper_sizes
```

Format

A data frame with 125 rows and 9 variables:

name Name of paper

series Series

standard Standard

size Size in series

units Units ("in", "mm", or "px") for dimensions

width Width in units

height Height in units

orientation Portrait (width less than height), landscape, or square

type Type (paper, postcard, print, or social)

plot_band

Plot a folded paper band to bundle a stack of cards or paper

Description

`plot_band()` creates a ggplot2 plot of a flat, foldable paper band sized to wrap around a stack of cards or paper and hold it together, similar to a currency or belly band. `paper` sets the width and height of the item being bundled, `orientation` sets which of those two dimensions the band wraps around, and the folded sides of the band (that wrap around the two edges of the stack) are sized to match the stack thickness calculated by `as_thickness()`. The band extends past the far edge of the stack with an additional segment so it can be glued or taped closed after wrapping.

Usage

```
plot_band(
  paper,
  orientation = c("horizontal", "vertical"),
  n = NULL,
  pt = 10,
  x = NULL,
  thickness = NULL,
  overlap = NULL,
  band_width = NULL,
  fill = "white",
  color = "black",
  linewidth = 0.5,
  glue_fill = "grey85",
  fold_linetype = "dashed",
  fold_linewidth = linewidth/2
)
```

Arguments

<code>paper</code>	Paper, page, or card name, or a data.frame with width and height columns (as returned by <code>get_page_size()</code> or <code>make_page_size()</code>), passed to <code>as_page()</code> . Sets the width and height of the stack of cards or paper the band wraps around.
<code>orientation</code>	Band orientation, either "horizontal" or "vertical". "horizontal" wraps the band left-to-right, crossing the middle of the stack, so the band's panels use the width of <code>paper</code> and the band's own height is set by <code>band_width</code> . "vertical" wraps the band top-to-bottom instead, so the panels use height and the band's own width is set by <code>band_width</code> . Default: "horizontal"
<code>n</code> , <code>pt</code> , <code>x</code>	Passed to <code>as_thickness()</code> to calculate the thickness of the stack. Ignored if <code>thickness</code> is supplied. <code>pt</code> is also used to add a small ease allowance

	(the caliper of a single sheet or card) to the folded side segments of the band, since a flat band can't wrap tightly around a stack at its exact thickness. Ignored if thickness is supplied.
thickness	Optional. A unit object or number (in the units of paper) with the thickness of the stack, overriding n , pt , and x . If supplied, no ease allowance is added and thickness sets the folded side segments of the band exactly. Default: NULL
overlap	Distance the band extends past the far edge of the stack, forming a segment that can be glued or taped closed. A unit object or a number (in the units of paper). Default: NULL , which uses 40% of the width of paper (if orientation is "horizontal") or 40% of the height of paper (if orientation is "vertical") — i.e. 40% of whichever dimension of paper the band wraps around.
band_width	Height of the band if orientation is "horizontal", or width of the band if orientation is "vertical" — the dimension of the band perpendicular to the direction it wraps around the stack. A unit object or a number (in the units of paper). Default: NULL , which uses 40% of the height of paper (if orientation is "horizontal") or 40% of the width of paper (if orientation is "vertical").
fill, color, linewidth	Fill, outline color, and outline linewidth for the band, Default: "white", "black", and 0.5
glue_fill	Fill color used to distinguish the overlapping segment of the band set by overlap , Default: "grey85"
fold_linetype, fold_linewidth	Linetype and linewidth for the lines marking where the band should be folded, Default: "dashed" and linewidth / 2

Value

A ggplot2 object.

See Also

[as_thickness\(\)](#)
[plot_band_dims\(\)](#)

Examples

```
## Not run:
if (interactive() && is_installed("ggplot2")) {
  plot_band(get_card("Poker"), n = 54)

  plot_band(get_card("Tarot"), n = 78, orientation = "vertical")
}

## End(Not run)
```

plot_band_dims	<i>Calculate the printed dimensions of a folded paper band</i>
----------------	--

Description

`plot_band_dims()` calculates the overall printed width and height of the band `plot_band()` draws for a given stack of cards or paper, including the fold-ease `plot_band()` adds to the stack thickness (see `plot_band()`'s `pt` parameter) and the `overlap` segment. Use it to get the band's actual size — e.g. to center `plot_band()`'s plot on a larger page with `print_to_page()` — without re-deriving the layout by hand.

Usage

```
plot_band_dims(
  paper,
  orientation = c("horizontal", "vertical"),
  n = NULL,
  pt = 10,
  x = NULL,
  thickness = NULL,
  overlap = NULL,
  band_width = NULL
)
```

Arguments

paper	Paper, page, or card name, or a data.frame with width and height columns (as returned by <code>get_page_size()</code> or <code>make_page_size()</code>), passed to <code>as_page()</code> . Sets the width and height of the stack of cards or paper the band wraps around.
orientation	Band orientation, either "horizontal" or "vertical". "horizontal" wraps the band left-to-right, crossing the middle of the stack, so the band's panels use the width of <code>paper</code> and the band's own height is set by <code>band_width</code> . "vertical" wraps the band top-to-bottom instead, so the panels use height and the band's own width is set by <code>band_width</code> . Default: "horizontal"
n, pt, x	Passed to <code>as_thickness()</code> to calculate the thickness of the stack. Ignored if <code>thickness</code> is supplied. <code>pt</code> is also used to add a small ease allowance (the caliper of a single sheet or card) to the folded side segments of the band, since a flat band can't wrap tightly around a stack at its exact thickness. Ignored if <code>thickness</code> is supplied.
thickness	Optional. A unit object or number (in the units of <code>paper</code>) with the thickness of the stack, overriding <code>n</code> , <code>pt</code> , and <code>x</code> . If supplied, no ease allowance is added and <code>thickness</code> sets the folded side segments of the band exactly. Default: NULL

<code>overlap</code>	Distance the band extends past the far edge of the stack, forming a segment that can be glued or taped closed. A <code>unit</code> object or a number (in the units of <code>paper</code>). Default: <code>NULL</code> , which uses 40% of the <code>width</code> of <code>paper</code> (if <code>orientation</code> is <code>"horizontal"</code>) or 40% of the <code>height</code> of <code>paper</code> (if <code>orientation</code> is <code>"vertical"</code>) — i.e. 40% of whichever dimension of <code>paper</code> the band wraps around.
<code>band_width</code>	Height of the band if <code>orientation</code> is <code>"horizontal"</code> , or width of the band if <code>orientation</code> is <code>"vertical"</code> — the dimension of the band perpendicular to the direction it wraps around the stack. A <code>unit</code> object or a number (in the units of <code>paper</code>). Default: <code>NULL</code> , which uses 40% of the <code>height</code> of <code>paper</code> (if <code>orientation</code> is <code>"horizontal"</code>) or 40% of the <code>width</code> of <code>paper</code> (if <code>orientation</code> is <code>"vertical"</code>).

Value

A `data.frame` with the band's overall `width`, `height`, and `units`, as returned by `make_page_size()`.

See Also

`plot_band()`

Examples

```
plot_band_dims(get_card("Poker"), n = 54)

plot_band_dims(get_card("Tarot"), n = 78, orientation = "vertical")
```

<code>plot_band_page</code>	<i>Arrange one or more paper bands on a page for printing</i>
-----------------------------	---

Description

`plot_band_page()` creates `n_bands` copies of a `plot_band()` band and arranges them, at their true printed size, in a grid on `page` — a thin wrapper around `page_layout()` (the same layout function used for a grid of cards), which handles sizing the grid to the band's true dimensions, shrinking it to fit if `n_bands` is fewer than `page` holds, positioning that grid via `position`, gutter spacing, margins, crop marks, and pagination when `n_bands` doesn't fit on one page. The result can be saved directly with `ggplot2::ggsave()`.

Usage

```
plot_band_page(
  paper,
  orientation = c("horizontal", "vertical"),
  n = NULL,
  pt = 10,
  x = NULL,
  thickness = NULL,
```

```

overlap = NULL,
band_width = NULL,
fill = "white",
color = "black",
linewidth = 0.5,
glue_fill = "grey85",
fold_linetype = "dashed",
fold_linewidth = linewidth/2,
page = "letter",
page_orientation = NULL,
n_bands = 1,
ncol = NULL,
nrow = NULL,
byrow = FALSE,
gutter = NULL,
margin = NULL,
position = "center",
unit = "in",
marks = FALSE,
paginate = TRUE
)

```

Arguments

paper	Paper, page, or card name, or a data.frame with width and height columns (as returned by <code>get_page_size()</code> or <code>make_page_size()</code>), passed to <code>as_page()</code> . Sets the width and height of the stack of cards or paper the band wraps around.
orientation	Band orientation, either "horizontal" or "vertical". "horizontal" wraps the band left-to-right, crossing the middle of the stack, so the band's panels use the <code>width</code> of <code>paper</code> and the band's own height is set by <code>band_width</code> . "vertical" wraps the band top-to-bottom instead, so the panels use <code>height</code> and the band's own width is set by <code>band_width</code> . Default: "horizontal"
n, pt, x	Passed to <code>as_thickness()</code> to calculate the thickness of the stack. Ignored if <code>thickness</code> is supplied. <code>pt</code> is also used to add a small ease allowance (the caliper of a single sheet or card) to the folded side segments of the band, since a flat band can't wrap tightly around a stack at its exact thickness. Ignored if <code>thickness</code> is supplied.
thickness	Optional. A <code>unit</code> object or number (in the units of <code>paper</code>) with the thickness of the stack, overriding <code>n</code> , <code>pt</code> , and <code>x</code> . If supplied, no ease allowance is added and <code>thickness</code> sets the folded side segments of the band exactly. Default: <code>NULL</code>
overlap	Distance the band extends past the far edge of the stack, forming a segment that can be glued or taped closed. A <code>unit</code> object or a number (in the units of <code>paper</code>). Default: <code>NULL</code> , which uses 40% of the <code>width</code> of <code>paper</code> (if <code>orientation</code> is "horizontal") or 40% of the <code>height</code> of <code>paper</code>

	(if <code>orientation</code> is "vertical") — i.e. 40% of whichever dimension of <code>paper</code> the band wraps around.
<code>band_width</code>	Height of the band if <code>orientation</code> is "horizontal", or width of the band if <code>orientation</code> is "vertical" — the dimension of the band perpendicular to the direction it wraps around the stack. A <code>unit</code> object or a number (in the units of <code>paper</code>). Default: <code>NULL</code> , which uses 40% of the <code>height</code> of <code>paper</code> (if <code>orientation</code> is "horizontal") or 40% of the <code>width</code> of <code>paper</code> (if <code>orientation</code> is "vertical").
<code>fill, color, linewidth</code>	Fill, outline color, and outline linewidth for the band, Default: "white", "black", and 0.5
<code>glue_fill</code>	Fill color used to distinguish the overlapping segment of the band set by <code>overlap</code> , Default: "grey85"
<code>fold_linetype, fold_linewidth</code>	Linetype and linewidth for the lines marking where the band should be folded, Default: "dashed" and <code>linewidth / 2</code>
<code>page</code>	Paper name or a data.frame with width and height columns, passed to <code>page_layout()</code> . Default: "letter"
<code>page_orientation</code>	Page orientation, passed to <code>page_layout()</code> 's <code>orientation</code> parameter (kept as a separate parameter here to avoid a name clash with the band's own <code>orientation</code>). Default: <code>NULL</code> , which uses <code>page</code> as supplied without reorienting it (unlike <code>page_layout()</code> itself, which defaults to "landscape").
<code>n_bands</code>	Number of copies of the band to arrange on <code>page</code> . Default: 1
<code>ncol, nrow</code>	The dimensions of the grid to create. If both are <code>NULL</code> , <code>dims</code> will be used or <code>dims</code> will be determined based on the plot dimensions.
<code>byrow</code>	Analogous to <code>byrow</code> in <code>matrix()</code> . If <code>FALSE</code> the plots will be filled in in column-major order
<code>gutter</code>	Optional. Spacing to add between plots in the grid, as a single number (used for both row and column spacing), a length-2 numeric vector <code>c(row, col)</code> , or a named vector or list with <code>row</code> and <code>col</code> elements. Interpreted in <code>unit</code> . Implemented by adding half of <code>gutter</code> to the interior-facing sides of each plot's own <code>plot.margin</code> (so two adjacent plots each contribute half, summing to the full gutter between them) based on its row/column position in the <code>ncol</code> x <code>nrow</code> grid — this replaces each plot's existing <code>plot.margin</code> . Plots on the outer edge of the grid are not padded on their outward-facing side; use <code>margin</code> for space around the outside of the whole grid. Default: <code>NULL</code> (no extra spacing). <code>gutter</code> changes the total size of the combined grid (adding $(ncol - 1) * col_gutter$ and $(nrow - 1) * row_gutter$), which <code>marks</code> accounts for automatically — the automatic <code>ncol/nrow</code> calculation from <code>page/dims</code> also reserves room for it, so fewer plots may fit per page than with <code>gutter = 0</code> .
<code>margin</code>	Optional. A margin to add around the outside of the combined grid of plots, e.g. so the grid can be centered on a larger sheet of paper when

saved with `ggplot2::ggsave()`. Passed to `get_margin()` with `unit`. The margin pads the composed page rather than the individual plots in `plots`, and does not affect the number of rows and columns in the grid. Default: `NULL`, which computes a margin from `position` whenever plot dimensions are known (supplied via `dims`, or auto-detected from the first plot) — see `position`. Set `margin` explicitly to override that (or to add a margin when dimensions aren't known, e.g. `ncol/nrow` supplied without `dims`). `margin` itself always renders correctly regardless of `page`, because it is applied as a fixed absolute-unit margin around whatever canvas size `ggplot2::ggsave()` is eventually called with. `marks`, below, is the one that depends on `page` being set correctly — see `marks`.

<code>position</code>	Where to place the bands on <code>page</code> when <code>n_bands</code> doesn't fill it completely. Passed to <code>page_layout()</code> 's <code>position</code> parameter, but defaults to "center" here rather than <code>page_layout()</code> 's own "top-left" default.
<code>unit</code>	Unit used for <code>gutter</code> , and for <code>margin</code> if margin is a bare numeric vector or list (ignored for <code>margin</code> if it is a <code>unit</code> class object; <code>gutter</code> does not support <code>unit</code> class objects). Default: "in".
<code>marks</code>	If <code>TRUE</code>, add crop marks in the <code>margin</code> area showing where to cut the page into individual plots. Requires <code>margin</code>. Marks are placed assuming the grid of plots exactly fills the page after subtracting <code>margin</code>, i.e. the same assumption <code>margin</code> itself relies on. Default: <code>FALSE</code>. Unlike <code>margin</code>, <code>marks</code> reads <code>page</code> (via <code>get_page_dims()</code>) to work out where the margin area is, so <code>page</code> must equal the exact final output size — the same width/height you pass to <code>ggsave()</code> — even if <code>ncol/nrow</code> are supplied directly and <code>page</code> would otherwise be unused. Passing just the combined size of the grid of plots (i.e. <code>page</code> without <code>margin</code> added on) will place the marks in the wrong location, because <code>ggplot2::ggsave()</code> will render a larger canvas than <code>marks</code> was calculated for.
<code>paginate</code>	If <code>TRUE</code> , create a list of <code>patchwork</code> objects when the number of plots is greater than the number of spaces in the plot layout. Default to <code>TRUE</code> .

Value

A `patchwork` object, or (if `n_bands` doesn't fit on a single page and `paginate = TRUE`) a list of `patchwork` objects — one per page. Save with `ggplot2::ggsave()`.

See Also

[plot_band\(\)](#)
[page_layout\(\)](#)

Examples

```
## Not run:
if (interactive() && is_installed(c("ggplot2", "patchwork"))) {
  plot_band_page(get_card("Poker"), n = 54, n_bands = 4)

  plot_band_page(
```

```

    get_card("Poker"),
    n = 54,
    n_bands = 1,
    position = "bottom-right"
  )
}

## End(Not run)

```

plot_cards

Use ggplot to plot for one or more cards

Description

`plot_cards()` takes a card that defines the dimensions of a card and creates a list of plots using the specified aesthetics.

Usage

```

plot_cards(
  card,
  n = 1,
  orientation = "portrait",
  number = FALSE,
  color = "white",
  size = 5,
  family = NULL,
  fill = "gray20",
  border = FALSE,
  inset = grid::unit(c(5, 5), "mm"),
  linetype = "dashed",
  linewidth = 1,
  text = NULL,
  center = NULL
)

```

Arguments

<code>card</code>	Card name or data.frame with width and height columns.
<code>n</code>	Number of cards to plot, Default: 1
<code>orientation</code>	Card orientation, Default: 'portrait'
<code>number</code>	If TRUE, add a number to each card, Default: FALSE
<code>color</code>	Color for number, text, and border, Default: 'white'
<code>size</code>	Font size for number and text, Default: 5
<code>family</code>	Font family for number and text, Default: 'Georgia'

fill	Length 1 or 2 character vector with color name. If length 2, the first value is assumed to be the card fill and the second value is assumed to be the inset border fill. Default: 'gray20'
border	If TRUE, add a border to the card. Default: FALSE
inset	Unit or numeric vector with inset distance for card border, Default: <code>grid::unit(c(5, 5), "mm")</code> . If inset is a numeric vector, it is expected to be a percent relative to the card width and height.
linetype	linetype for card border, Default: 'dashed'
linewidth	linewidth for card border, Default: 2
text	Character vector with card text, Default: NULL
center	Position of card center, Default: <code>c(0, 0)</code>

Value

A list of plot objects where each item on the list is a card.

Examples

```
#'
## Not run:
if (interactive() && is_installed("ggplot2")) {
  plot_cards("Tarot", n = 2, number = TRUE)[[2]]

  plot_cards("Poker", n = 1, number = TRUE, text = " ")
}

## End(Not run)
```

print_to_page	<i>Explicitly draw plot using dimensions from page data.frame or list</i>
----------------------	---

Description

[Experimental]

Usage

```
print_to_page(plot, page, newpage = TRUE, vp = NULL, ...)

print_to_page_layout(
  plot,
  page,
  row_position = 1,
  col_position = 1,
  row_height = NULL,
  col_width = NULL,
```

```

    nrow = 1,
    ncol = 1,
    layout = NULL,
    parent = NULL,
    children = NULL,
    filename = NULL,
    ...
)

```

Arguments

plot	Plot to display
page	A page data.frame from <code>get_page_size()</code> or <code>make_page_size()</code> .
newpage	draw new (empty) page first?
vp	viewport to draw plot in
...	Arguments passed on to <code>page_to_viewport</code> , <code>page_to_layout</code>
name	A character value to uniquely identify the viewport once it has been pushed onto the viewport tree.
cols	Column names to use for width and height columns. Defaults to <code>c("width", "height")</code> . Must be length 2 and the first value is always used as the width name and the second as the height.
margins	A numeric list or vector or a margin class object. Defaults to <code>NULL</code> . margins are removed from the overall layout width and height.
region	Optional. An additional page data.frame where the region width and height are used as the column width and row height.
gutter	Gutter width/height. Not yet implemented.
units	Passed to default.units parameter of <code>grid::grid.layout()</code> .
widths	A numeric vector or unit object describing the widths of the columns in the layout.
heights	A numeric vector or unit object describing the heights of the rows in the layout.
respect	A logical value or a numeric matrix. If a logical, this indicates whether row heights and column widths should respect each other. If a matrix, non-zero values indicate that the corresponding row and column should be respected (see examples below).
just	A string or numeric vector specifying how the layout should be justified if it is not the same size as its parent viewport. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: <code>"left"</code> , <code>"right"</code> , <code>"centre"</code> , <code>"center"</code> , <code>"bottom"</code> , and <code>"top"</code> . For numeric values, 0 means left alignment and 1 means right alignment. NOTE that in this context, <code>"left"</code> , for example, means align the left edge of the left-most layout column with the left edge of the parent viewport.

row_position, col_position	Row and column position. If nrow is smaller than row_position or ncol is smaller than col_position, row_position is used for nrow or col_position is used for ncol instead.
row_height, col_width	Row height and column width.
nrow	An integer describing the number of rows in the layout.
ncol	An integer describing the number of columns in the layout.
layout	Passed to page_to_viewport() with layout_position as layout.pos.row and layout.pos.col if provided. Defaults to NULL where layout is defined by page_to_layout using ncol, nrow, page and any additional parameters passed to ...
parent	A grid viewport object.
children	A vpList object.
filename	File name to create on disk.

Value

Invisibly returns the original plot using a viewport from [page_to_viewport\(\)](#).

See Also

[ggplot2::print.ggplot\(\)](#)

Examples

```
## Not run:
if (interactive() && is_installed("ggplot2")) {
  library(ggplot2)

  p <- ggplot(mpg, aes(displ, hwy, colour = class)) +
    geom_point()

  print_to_page(
    p,
    page = get_page_size("Tarot card")
  )
}

## End(Not run)
```

set_page_dims

Set page data.frame dimensions, orientation, or aspect ratio

Description

Set page data.frame dimensions, orientation, or aspect ratio

set_page_asp: appends an aspect ratio column a page size data.frame

Usage

```

set_page_dims(
  page,
  dims = NULL,
  width = NULL,
  height = NULL,
  units = NULL,
  cols = c("width", "height")
)

set_page_orientation(
  page,
  orientation = NULL,
  tolerance = 0.1,
  cols = c("width", "height")
)

set_page_asp(page, flipped = FALSE, cols = c("width", "height"))

```

Arguments

page	If character, page is passed as the name parameter to <code>get_page_size()</code> .
dims	A vector or list with values in the same order as cols (defaults to <code>c(<width>, <height>)</code> to match cols).
width, height	Page width and height. Both are required, if asp is NULL. Default to NULL.
units	Units for width and height. Required unless units is included in dims or <code>require_units = FALSE</code> . Passed to <code>as_unit_type()</code> to validate.
cols	Column names to use for width and height columns. Defaults to <code>c("width", "height")</code> . Must be length 2 and the first value is always used as the width name and the second as the height.
orientation	Page orientation, Default: NULL. Supported options are "portrait", "landscape", or "square". If width and height suggest a portrait orientation when orientation = "landscape", the dimensions are reversed so the page dimensions match the provided orientation.
tolerance	Positive numeric value above or below 1 used to determine if an aspect ratio is square, landscape, or portrait.
flipped	If TRUE, return aspect ratio of height to width (y / x), instead of width to height.

set_page_margin	<i>Set margins for page data.frame (adding body width, height, and asp)</i>
------------------------	---

Description

Set margins for page data.frame (adding body width, height, and asp)

Usage

```
set_page_margin(
  page = NULL,
  margins,
  unit = "in",
  cols = c("width", "height"),
  ...
)
```

Arguments

page	A character vector with a page size name or a data.frame. Passed to x parameter of <code>as_page()</code> .
margins	Passed to <code>get_margin()</code> with unit value.
unit	Unit used for the margin. If margin is a unit object, unit is ignored. If page uses different units, the margins are converted into the page units for consistency.
cols	Column names to use for width and height columns. Defaults to c("width", "height"). Must be length 2 and the first value is always used as the width name and the second as the height.
...	Passed to <code>as_page()</code> with page and cols.

Value

A data.frame with the page dimensions and additional columns for body dimensions, body aspect ratio, and margins.

See Also

`ggplot2::margin()`; `set_page_dims()`; `set_page_orientation()`; `set_page_asp()`

standard_scales

Standard map, architectural, and engineering scales

Description

Standard map scales derived from USGS 2002 report on map scales <https://pubs.usgs.gov/fs/2002/0015/report.pdf>

Usage

```
standard_scales
```

Format

A data frame with 36 rows and 16 variables:

`scale` Scale name
`standard` Standard (USGS, architectural, or engineering)
`series` Series name (USGS map scales only)
`actual_ft` Scale distance for 1 ft actual.
`actual_ft_unit` Unit of scale for 1 ft actual.
`scale_in` Actual distance for 1 in scale.
`scale_in_unit` Unit of actual distance for 1 in scale.
`scale_in_accuracy` Accuracy of 1 in scale (approximate or exact)
`scale_cm` Actual distance for 1 cm scale.
`scale_cm_unit` Unit of actual distance for 1 cm scale.
`scale_cm_accuracy` Accuracy of 1 cm scale (approximate or exact)
`size_latlon` Standard size in latitude/longitude
`size_latlon_unit` Unit of latitude/longitude size (minutes or degrees)
`area_approx` Approximate actual area
`area_approx_unit` Approximate area unit
`series_status` Series status (select USGS map series are "abandoned")

Details

Common architectural and engineering scales derived from FEMA guide to using scales
<https://www.usfa.fema.gov/downloads/pdf/nfa/engineer-architect-scales.pdf>

theme_page

Modify plot aspect ratio to match a page size

Description

Wrapper for `ggplot2::theme()` that passes `as_asp()` with `flip = TRUE` to the `aspect.ratio` argument to force a plot to match the aspect ratio of the provided page.

Usage

```
theme_page(page, orientation = NULL, ...)
```

Arguments

<code>page</code>	If character, page is passed as the name parameter to <code>get_page_size()</code> .
<code>orientation</code>	Page orientation, Default: NULL. Supported options are "portrait", "landscape", or "square".
<code>...</code>	Arguments passed on to <code>ggplot2::theme</code>
<code>line</code>	all line elements (<code>element_line()</code>)
<code>rect</code>	all rectangular elements (<code>element_rect()</code>)
<code>text</code>	all text elements (<code>element_text()</code>)
<code>title</code>	all title elements: plot, axes, legends (<code>element_text()</code> ; inherits from <code>text</code>)
<code>point</code>	all point elements (<code>element_point()</code>)
<code>polygon</code>	all polygon elements (<code>element_polygon()</code>)
<code>geom</code>	defaults for geoms (<code>element_geom()</code>)
<code>spacing</code>	all spacings (<code>unit()</code>)
<code>margins</code>	all margins (<code>margin()</code>)
<code>aspect.ratio</code>	aspect ratio of the panel
<code>axis.title</code> , <code>axis.title.x</code> , <code>axis.title.y</code> , <code>axis.title.x.top</code> , <code>axis.title.x.bottom</code> , <code>axis.title.x.left</code> , <code>axis.title.x.right</code> , <code>axis.title.y.left</code> , <code>axis.title.y.right</code>	labels of axes (<code>element_text()</code>). Specify all axes' labels (<code>axis.title</code>), labels by plane (using <code>axis.title.x</code> or <code>axis.title.y</code>), or individually for each axis (using <code>axis.title.x.bottom</code> , <code>axis.title.x.top</code> , <code>axis.title.y.left</code> , <code>axis.title.y.right</code>). <code>axis.title.*</code> inherits from <code>axis.title</code> , which in turn inherits from <code>text</code>
<code>axis.text</code> , <code>axis.text.x</code> , <code>axis.text.y</code> , <code>axis.text.x.top</code> , <code>axis.text.x.bottom</code> , <code>axis.text.x.left</code> , <code>axis.text.x.right</code> , <code>axis.text.y.left</code> , <code>axis.text.y.right</code>	tick labels along axes (<code>element_text()</code>). Specify all axis tick labels (<code>axis.text</code>), tick labels by plane (using <code>axis.text.x</code> or <code>axis.text.y</code>), or individually for each axis (using <code>axis.text.x.bottom</code> , <code>axis.text.x.top</code> , <code>axis.text.y.left</code> , <code>axis.text.y.right</code>). <code>axis.text.*</code> inherits from <code>axis.text</code> , which in turn inherits from <code>text</code>
<code>axis.ticks</code> , <code>axis.ticks.x</code> , <code>axis.ticks.x.top</code> , <code>axis.ticks.x.bottom</code> , <code>axis.ticks.y</code> , <code>axis.ticks.x.left</code> , <code>axis.ticks.x.right</code> , <code>axis.ticks.y.left</code> , <code>axis.ticks.y.right</code>	tick marks along axes (<code>element_line()</code>). Specify all tick marks (<code>axis.ticks</code>), ticks by plane (using <code>axis.ticks.x</code> or <code>axis.ticks.y</code>), or individually for each axis (using <code>axis.ticks.x.bottom</code> , <code>axis.ticks.x.top</code> , <code>axis.ticks.y.left</code> , <code>axis.ticks.y.right</code>). <code>axis.ticks.*</code> inherits from <code>axis.ticks</code> , which in turn inherits from <code>line</code>
<code>axis.minor.ticks.x.top</code> , <code>axis.minor.ticks.x.bottom</code> , <code>axis.minor.ticks.y.left</code> , <code>axis.minor.ticks.y.right</code>	minor tick marks along axes (<code>element_line()</code>). <code>axis.minor.ticks.*</code> inherit from the corresponding major ticks <code>axis.ticks.*</code> .
<code>axis.ticks.length</code> , <code>axis.ticks.length.x</code> , <code>axis.ticks.length.x.top</code> , <code>axis.ticks.length.x.bottom</code> , <code>axis.ticks.length.y</code>	length of tick marks (<code>unit</code>). <code>axis.ticks.length</code> inherits from <code>spacing</code> .
<code>axis.minor.ticks.length</code> , <code>axis.minor.ticks.length.x</code> , <code>axis.minor.ticks.length.x.top</code> , <code>axis.minor.ticks.length.x.bottom</code> , <code>axis.minor.ticks.length.y</code>	length of minor tick marks (<code>unit</code>), or relative to <code>axis.ticks.length</code> when provided with <code>rel()</code> .

axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y, axis.line.y 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 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. Using this setting overrides the aspect ratio set by the theme, coord or facets. An exception is made when

the plot has a single panel and exactly one of the width *or* height is set, in which case an attempt is made to preserve the aspect ratio.

`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.x.left, strip.text.x.right` 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

A ggplot `theme` class object with the `aspect.ratio` argument set to match the page height / width.

See Also

`ggplot2::theme()`

Index

- * **annotate**
 - annotate_wave, 2
- * **card**
 - plot_band, 39
 - plot_band_dims, 41
 - plot_band_page, 42
 - plot_cards, 46
- * **datasets**
 - area_unit_options, 4
 - card_sizes, 11
 - dist_unit_options, 14
 - dist_units, 14
 - grid_units, 22
 - page_extras, 30
 - paper_sizes, 38
 - standard_scales, 51
- * **dist**
 - convert_dist_scale, 11
 - convert_dist_units, 13
 - is_dist_units, 25

abort(), 25, 34
annotate_wave, 2
annotate_wave(), 2
area(), 32
area_unit_options, 4
as.mask, 37
as.path, 37
as_asp, 4
as_asp(), 5
as_dist_units (*is_dist_units*), 25
as_dist_units(), 25
as_page, 6
as_page(), 39, 41, 43, 51
as_thickness, 7
as_thickness(), 7, 39–41, 43
as_unit, 8
as_unit(), 8
as_unit_type (*as_unit*), 8
as_unit_type(), 6, 8, 28, 30, 50

card_sizes, 11
cli::cli_abort(), 25
cli_abort(), 5, 9, 12, 16, 23
convert_dist_scale, 11
convert_dist_scale(), 13, 25
convert_dist_units, 13
convert_dist_units(), 12, 13, 25
convert_page_units (*get_page_size*), 15
convert_page_units(), 15
convert_unit_type, 23
convert_unit_type (*as_unit*), 8
convert_unit_type(), 5, 8, 12, 15, 16

defused function call, 7, 9, 24, 29
dist_unit_options, 14, 25
dist_units, 14

element_geom(), 53
element_line(), 53–55
element_point(), 53
element_polygon(), 53
element_rect(), 53–56
element_text(), 53, 54, 56
exiftoolr::exif_call(), 21

filenamr::check_file_overwrite(), 21
filenamr::make_filename(), 20
filenamr::read_exif(), 26, 27

get_card (*get_page_size*), 15
get_card(), 11, 15
get_dist_units (*is_dist_units*), 25
get_dist_units(), 25
get_margin (*margins*), 29
get_margin(), 30, 33, 45, 51
get_page_dims (*get_page_size*), 15
get_page_dims(), 6, 12, 15, 16, 29, 33, 45
get_page_size, 5, 15
get_page_size(), 5, 6, 12, 15, 16, 18, 29, 35, 37–39, 41, 43, 48, 50, 53

get_paper, 12
 get_paper(*get_page_size*), 15
 get_paper(), 12, 15
 get_scale, 17
 get_social_size, 18
 get_social_size(), 16, 22
 get_standard_scale(*get_scale*), 17
 ggplot2::annotate(), 2, 3
 ggplot2::element_text(), 27
 ggplot2::geom_line(), 2, 3
 ggplot2::ggplot_build(), 34
 ggplot2::ggsave, 21
 ggplot2::ggsave(), 22, 27, 32, 33, 42, 45
 ggplot2::labs(), 27
 ggplot2::margin(), 27, 29, 30, 51
 ggplot2::print.ggplot(), 49
 ggplot2::theme, 53
 ggplot2::theme(), 27, 57
 ggsave_ext, 18
 ggsave_ext(), 19
 ggsave_social(*ggsave_ext*), 18
 glue::glue(), 27
 glue::glue_data(), 27
 gpar, 37
 grid::convertUnit, 10
 grid::convertUnit(), 5, 10, 12, 16, 23
 grid::grid.layout(), 35, 48
 grid::unit(), 22, 30
 grid::viewport, 37
 grid_units, 22
 gridExtra::arrangeGrob(), 22

 Including function calls in error
 messages, 7, 9, 24, 29
 inset_page(), 24
 inset_page_element, 22
 is_dist_units, 25
 is_dist_units(), 13, 25
 is_margin(*margins*), 29
 is_same_unit_type(*as_unit*), 8
 is_same_unit_type(), 8, 13
 is_same_units(*is_dist_units*), 25
 is_same_units(), 25
 is_unit_type(*as_unit*), 8
 is_unit_type(), 8
 isstatic::as_orientation(), 5

 janitor::make_clean_names(), 20

 layout_cards(*page_layout*), 31
 lubridate::ymd_hms(), 27

 magick::editing(), 27
 magick::image_ggplot(), 20, 26, 27, 33
 make_contact_sheets, 26
 make_page_size, 6, 28
 make_page_size(), 6, 16, 35, 37, 39,
 41–43, 48
 map_ggsave_ext(*ggsave_ext*), 18
 map_ggsave_ext(), 19, 27
 maplayer::layer_inset(), 22
 margin(), 30, 53–55
 margins, 29
 matrix(), 32, 44

 page_extras, 30
 page_layout, 31
 page_layout(), 26, 42, 44, 45
 page_to_layout, 35, 48
 page_to_viewport, 36, 48
 page_to_viewport(), 49
 paper_sizes, 38
 patchwork::inset_element(), 24
 patchwork::plot_layout(), 34
 patchwork::wrap_plots(), 33, 34
 plot_annotation(), 32
 plot_band, 39
 plot_band(), 7, 39, 41, 42, 45
 plot_band_dims, 41
 plot_band_dims(), 40, 41
 plot_band_page, 42
 plot_band_page(), 42
 plot_cards, 46
 plot_cards(), 46
 png, 21
 print_to_page, 47
 print_to_page(), 41
 print_to_page_layout(*print_to_page*),
 47
 purrr::map(), 19

 rlang::arg_match(), 25

 set_page_asp(*set_page_dims*), 49
 set_page_asp(), 51
 set_page_dims, 49
 set_page_dims(), 51
 set_page_margin, 50

`set_page_margin()`, [30](#)
`set_page_orientation` (*set_page_dims*),
[49](#)
`set_page_orientation()`, [51](#)
`standard_scales`, [11](#), [12](#), [17](#), [51](#)

`theme(legend.position=...)`, [32](#)
`theme_grey()`, [57](#)
`theme_page`, [52](#)

`unit()`, [53](#), [54](#)
`units::as_units`, [25](#)
`units::set_units()`, [13](#)
`units::valid_udunits()`, [4](#), [14](#)