Package: papersize (via r-universe)

August 30, 2024

Type Package

Title Sizing Plots and Files for Paper

Version 0.1.0.9001

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 (>= 1.1.0), units, utils, vctrs

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

Remotes elipousson/cliExtras, elipousson/filenamr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

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

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

RemoteRef HEAD

RemoteSha 12db7c6beed734472e9b6ec0222dd2378dc66e62

Contents

`area_unit_options` *Area units (vector)*

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` *Convert character string, page name, or page data.frame to numeric aspect ratio*

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, `as_asp()` returns the value of asp without modification.
`page`	If character, page is passed as the name parameter to `get_page_size()`.
`orientation`	Page orientation, Default: `NULL`. Supported options are "portrait", "landscape", or "square".
`flipped`	If `TRUE`, return aspect ratio of height to width (y / x), instead of width to height.
`sep`	Separator character to use if asp is a character vector. Defaults to ":".
`cols`	Length 2 character vector with column names for page dimensions. Defaults to c("width", "height").

`...`	Arguments passed on to `get_page_size`
	`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 `reorient = FALSE` 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`.
	`units` Units to convert page dimensions to using `convert_unit_type()`.

Value

A numeric vector.

See Also

isstatic::as_orientation()

Examples

```
as_asp(8.5 / 11)

as_asp("11:17")

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

as_asp(page = "letter")

as_asp(page = "letter", orientation = "landscape")

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

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

---

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

Description

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

Usage

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

Arguments

x	A character vector passed to the name parameter of get_page_size(), a named list of vector passed to dims for make_page_size(), or a length 2 numeric vector passed as the width and height parameter to make_page_size(). If x is a character string that does not match any of the names in paper_sizes, x is passed as the name parameter to make_page_size()
...	Arguments passed on to make_page_size
	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. Passed to as_unit_type() 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 `paper_sizes`.

`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.

`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.

`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.

`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 as the width name and the second as the height.

`class` Class of return object: "data.frame" (default) or "list" (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_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 `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.
recurse	Whether to recurse into complex units.
arg	Passed to `cli_abort()` to improve internal error messages.
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.
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`.
...	Arguments passed on to `grid::convertUnit` `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("inchs")

is_same_unit_type(inch, "in")

is_same_unit_type("pt", "points")
```

---

card_sizes *Standard card sizes*

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` *Convert distance from scale to actual units*

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 `standard_scales[["scale"]]`.
`standard`	Scale standard. Options include "USGS", "Engineering", or "Architectural".
`series`	Map series from `standard_scales[["series"]]`. 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 if provided; defaults to `NULL`.
`actual_unit`	any unit supported by `convert_dist_units()`
`dpi`	dots per square inch (used as conversion factor for "px" to "in")
`paper`	Name of paper passed to `get_paper()`
`orientation`	Page orientation, Default: `NULL`. Supported options are "portrait", "landscape", or "square".

`...`	Arguments passed on to `get_paper`
	`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`

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 appends as columns named actual_width and actual_height.

See Also

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

`convert_dist_units` *Convert distance (and area) values between different units*

Description

Convert distance (and area) values between different units

Usage

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

Arguments

`dist`	Numeric or units object
`from`	Existing unit for dist, Default: `NULL`. If dist is a units object, the numerator is used as "from"
`to`	Unit to convert distance to, Default: 'meter'
`drop`	If `TRUE`, return numeric. If `FALSE`, return class units object.
`digits`	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_units *Distance units (data frame)*

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

dist_unit_options *Distance units (vector)*

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.

get_page_size *Get a paper or card size based on name, dimensions, orientation, or type*

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.

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 *Get standard scales and convert to scale distances*

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

Value

A tibble based on standard_scales with rows filtered to values that match parameters.

`get_social_size` *Get social media image size to match platform and format*

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

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

See Also

`get_page_size()`

Examples

```
get_social_size("Instagram post")

get_social_size(platform = "Twitter")

get_social_size(format = "cover")
```

ggsave_ext *Save a ggplot2 plot to file and update file EXIF metadata*

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 `magick::image_ggplot()`
`name`	Plot name, used to create filename (if filename is `NULL`) using `filenamr::make_filename()`
`label`	Label to combine with name converted to snake case with `janitor::make_clean_names()`. 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). Applies only to 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 exifr 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()`

---

grid_units *Grid units (vector)*

Description

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

Usage

```
grid_units
```

Format

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

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

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, or nativeRaster 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 *or* bottom and left *or* 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 `convert_unit_type` `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`.

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

Value

A `inset_path` object

See Also

`patchwork::inset_element()`

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

x, y	objects to check
arg	Used internally and passed to rlang::arg_match() as error arg or used by cli::cli_abort() to improve error messages.
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.
units	Distance units to convert to. Must be one of dist_unit_options or area_unit_options.

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 shets 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

images	File path or data.frame from `filenamr::read_exif()`
dims	Image dimensions in same dimensions as page. Required.
ncol, nrow	The dimensions of the grid to create. If both are NULL, dims will be used or dims will be determined based on the plot dimensions.
captions	Template for caption, passed to `glue::glue_data()` using the images data.frame as .x. Note that this template may vary if you are using a custom tags parameter or modify the "filenamr.exif_xwalk" option. See `filenamr::read_exif()` for more details. Default: "file_name\ndate_created"
caption_size	Caption size, passed to `ggplot2::element_text()` for plot.caption for theme, Default: 12
caption_position	Caption position, passed to plot.caption.position for theme, Default: 'panel'
image_margin	Image margin passed Default: `margins(0.1, unit = "in")`
page	Paper name or a data.frame with width and height columns. Optional if width and height are both provided, Default: NULL
orientation	Paper orientation, Optional if width and height are both provided, Default: 'landscape'
image_max	Maximum number of images to use for contact sheets.
image_fileext	Passed to fileext parameter of `filenamr::read_exif()`, Default: NULL
tags	List of EXIF tags to read from files. If NULL (default), set to option "filenamr.exif_tags" or default `default_exif_tags`.

tz	Time zone to pass to lubridate::ymd_hms() if format_exif is TRUE. Typically set to Sys.timezone() to convert date/time columns.
save	If TRUE, save contact sheet to a file. filename may be required if save is TRUE. Default: FALSE
filename	File name to create on disk.
...	Additional parameters passed to map_ggsave_ext() 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 *Make a page size data frame*

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,
  asp = NULL,
  orientation = NULL,
  name = NULL,
  dims = NULL,
  valid_units = NULL,
  cols = c("width", "height"),
  class = "data.frame",
  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. Passed to `as_unit_type()` 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 `paper_sizes`.
`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 as the width name and the second as the height.
`class`	Class of return object: "data.frame" (default) or "list" (only supported when input page size is a single row).
`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.

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)

margin(t = 0, r = 0, b = 0, l = 0, unit = "pt")

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

Arguments

`margin`	A numeric list or vector or a margin class object. For `get_margin()` only, margin can be a margin name from `page_extras[["margins"]]`. 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 `as_unit_type()` so units class objects as well as unit names supported `grid::unit()` are allowed. Defaults to "in" except for `margin()` where unit defaults to "pt" to match `ggplot2::margin()`.
`x`	Object to check for class margin and unit
`t`, `r`, `b`, `l`	Dimensions of each margin: top, right, bottom, and left. (To remember order, think trouble).

Source

ggplot2 package

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
```

---

page_extras *Extra reference data for page layouts*

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

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,
  images = FALSE,
  dpi = 120,
  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`
`page`	Paper name or a data.frame with width and height columns. Optional if width and height are both provided, Default: `NULL`
`width`, `height`	Paper width and height, Default: `NULL`
`orientation`	Paper orientation, Optional if width and height are both provided, Default: 'landscape'
`byrow`	Analogous to `byrow` in matrix(). If `FALSE` the plots will be filled in in column-major order
`guides`	A string specifying how guides should be treated in the layout. `'collect'` will collect guides below to the given nesting level, removing duplicates. `'keep'` will stop collection at this level and let guides be placed alongside their plot. `auto` 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 theme(legend.position=...) while also collecting guides you must apply that change to the overall patchwork (see example).
`tag_level`	A string (`'keep'` or `'new'`) to indicate how auto-tagging should behave. See `plot_annotation()`.
`design`	Specification of the location of areas in the layout. Can either be specified as a text string or by concatenating calls to `area()` together. See the examples for further information on use.
`paginate`	If `TRUE`, create a list of `patchwork` objects when the number of plots is greater than the number of spaces in the plot layout. Default to `TRUE`.

ncol, nrow	The dimensions of the grid to create. If both are NULL, dims will be used or dims will be determined based on the plot dimensions.
dims	Optional. Plot dimensions. Ignored if ncol and nrow are supplied. Otherwise, if NULL (default), dims are inferred based on the dimensions of the first plot in plots.
images	Not yet implemented. If TRUE and dims is NULL, the input plots are assumed to be plots created with magick::image_ggplot() and dpi is used to infer dimensions.
dpi	Not yet implemented. Resolution.
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_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 get_page_size() or a data.frame or list from make_page_size().
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 grid::grid.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: "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 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` *Convert a page data.frame to a* `viewport` *class object*

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 `get_page_size()` or `make_page_size()`.
`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 c("width", "height"). Must be length 2 and the first value is always used as as the width name and the second as the height.
`...`	Arguments passed on to `grid::viewport` `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: `"left"`, `"right"`, `"centre"`, `"center"`, `"bottom"`, and `"top"`. For numeric values, 0 means left alignment and 1 means right alignment. `gp` An object of class `"gpar"`, typically the output from a call to the function `gpar`. This is basically a list of graphical parameter settings. `clip` One of `"on"`, `"inherit"`, or `"off"`, 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 `TRUE` corresponds to `"on"` and `FALSE` corresponds to `"inherit"`. May also be a grob (or a gTree) that describes a clipping path or the result of a call to `as.path`.

`mask` One of `"none"` (or `FALSE`) or `"inherit"` (or `TRUE`) or a grob (or a gTree) or the result of call to `as.mask`. 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.

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_cards` *Use ggplot to plot for one or more cards*

Description

Make a plot of cards.

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

`card`	Card name or data.frame with width and height columns.
`n`	Number of cards to plot, Default: 1
`orientation`	Card orientation, Default: 'portrait'
`number`	If `TRUE`, add a number to each card, Default: `FALSE`
`color`	Color for number, text, and border, Default: 'white'
`size`	Font size for number and text, Default: 5
`family`	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: `grid::unit(c(5, 5), "mm")`. 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: c(0, 0)

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 *Explicitly draw plot using dimensions from page data.frame or list*

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 get_page_size() or make_page_size().
newpage	draw new (empty) page first?
vp	viewport to draw plot in
...	Arguments passed on to page_to_viewport, page_to_layout

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 c("width", "height"). Must be length 2 and the first value is always used as as the width name and the second as the height.

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.

gutter Gutter width/height. Not yet implemented.

units Passed to default.units parameter of grid::grid.layout().

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: `"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.

`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)

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

  print_to_page(
    plot,
    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 `get_page_size()`.
`dims`	A vector or list with values in the same order as cols (defaults to `c(<width>, <height>)` 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. Passed to `as_unit_type()` to validate.
`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 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` *Set margins for page data.frame (adding body width, height, and asp)*

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 `as_page()`.
margins	Passed to `get_margin()` 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 as the width name and the second as the height.
...	Passed to `as_page()` 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

`page`	If character, page is passed as the name parameter to `get_page_size()`.
`orientation`	Page orientation, Default: `NULL`. Supported options are "portrait", "landscape", or "square".

`...`	Arguments passed on to `ggplot2::theme`
	`line` all line elements (`element_line()`)
	`rect` all rectangular elements (`element_rect()`)
	`text` all text elements (`element_text()`)
	`title` all title elements: plot, axes, legends (`element_text()`; inherits from `text`)
	`aspect.ratio` aspect ratio of the panel
	`axis.title,axis.title.x,axis.title.y,axis.title.x.top,axis.title.x.bottom,axis.t` labels of axes (`element_text()`). Specify all axes' labels (`axis.title`), labels by plane (using `axis.title.x` or `axis.title.y`), or individually for each axis (using `axis.title.x.bottom`, `axis.title.x.top`, `axis.title.y.left`, `axis.title.y.right`). `axis.title.*.*` inherits from `axis.title.*` which inherits from `axis.title`, which in turn inherits from `text`
	`axis.text,axis.text.x,axis.text.y,axis.text.x.top,axis.text.x.bottom,axis.text.y` tick labels along axes (`element_text()`). Specify all axis tick labels (`axis.text`), tick labels by plane (using `axis.text.x` or `axis.text.y`), or individually for each axis (using `axis.text.x.bottom`, `axis.text.x.top`, `axis.text.y.left`, `axis.text.y.right`). `axis.text.*.*` inherits from `axis.text.*` which inherits from `axis.text`, which in turn inherits from `text`
	`axis.ticks,axis.ticks.x,axis.ticks.x.top,axis.ticks.x.bottom,axis.ticks.y,axis.t` tick marks along axes (`element_line()`). Specify all tick marks (`axis.ticks`), ticks by plane (using `axis.ticks.x` or `axis.ticks.y`), or individually for each axis (using `axis.ticks.x.bottom`, `axis.ticks.x.top`, `axis.ticks.y.left`, `axis.ticks.y.right`). `axis.ticks.*.*` inherits from `axis.ticks.*` which inherits from `axis.ticks`, which in turn inherits from `line`

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

`axis.ticks.length,axis.ticks.length.x,axis.ticks.length.x.top,axis.ticks.length.` length of tick marks (`unit`)

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

`axis.line,axis.line.x,axis.line.x.top,axis.line.x.bottom,axis.line.y,axis.line.y` 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()`)

`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.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

`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.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`)

`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,l` 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", or "right")

`legend.box.margin` margins around the full legend area, as specified using `margin()`

`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`)

`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.grid,panel.grid.major,panel.grid.minor,panel.grid.major.x,panel.grid.major` grid lines (`element_line()`). Specify major grid lines, or minor grid lines separately (using `panel.grid.major` or `panel.grid.minor`) or individually for each axis (using `panel.grid.major.x`, `panel.grid.minor.x`, `panel.grid.major.y`, `panel.grid.minor.y`). Y axis grid lines are horizontal and x axis grid lines are vertical. `panel.grid.*.*` inherits from `panel.grid.*` which inherits from `panel.grid`, which in turn inherits from `line`

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

`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)

`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.` 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` space between strips and axes when strips are switched (`unit`)

`strip.switch.pad.wrap` space between strips and axes when strips are switched (`unit`)

`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 `validate_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