Package: pnpmisc (via r-universe)

December 18, 2024

Type Package

Title Utilities for Print-and-Play Board Games

Version 0.1.0-14

URL https://github.com/trevorld/pnpmisc,
https://trevorldavis.com/R/pnpmisc/dev/

BugReports https://github.com/trevorld/pnpmisc/issues

Description Utilities for print-and-play board games.

Imports grid, grDevices, pdftools, qpdf, stats, tools

Suggests bittermelon, gridpattern, piecepackr (>= 1.14.0-6), testthat, xmpdf

Remotes piecepackr/piecepackr

License MIT + file LICENSE

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Encoding UTF-8

Config/pak/sysreqs libjpeg-dev libssl-dev libpoppler-cpp-dev

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

RemoteUrl https://github.com/trevorld/pnpmisc

RemoteRef HEAD

RemoteSha 0287bc66e51c394562f7bcadff3c5de52fa3aef6

Contents

`bm_crop_layout` *Crop out a component from a print-and-play layout*

Description

`bm_crop_layout()` crops out a print-and-play component from a layout.

Usage

```
bm_crop_layout(
  page,
  ...,
  layout = "button_shy_cards",
  row = 1L,
  col = 1L,
  bleed = FALSE
)
```

Arguments

`page`	A `bittermelon::bm_pixmap()` object representing a print-and-play layout page. Often the output from `pdf_render_bm_pixmap()`.
`...`	Ignored for now.

layout	Either a layout preset name in layout_names() or a data frame with layout data (as returned by layout_grid()).
row, col	The row and col of the component in the layout (integers).
bleed	Include the bleed (if available).

Value

A bittermelon::bm_pixmap() object.

Examples

```
## Not run: # User not expected to have this PDF file
if (requireNamespace("bittermelon", quietly = TRUE)) {
  input <- "A Nice Cuppa - PNP.pdf"
  page <- pdf_render_bm_pixmap(input, page = 4L, dpi = 75)
  card <- bm_crop_layout(page, layout = "button_shy_cards", row = 1L, col = 1L)
  grid::grid.raster(card)
}

## End(Not run)
```

---

grid_add_cropmarks *Draw crop marks around components*

Description

grid_add_cropmarks() adds crop marks to the edges of components of a print-and-play layout.

Usage

```
grid_add_cropmarks(..., layout = "poker_3x3", bleed = NULL)
```

Arguments

...	Passed to piecepackr::grid.cropmark().
layout	Either a layout preset name in layout_names() or a data frame with layout data (as returned by layout_grid()).
bleed	Passed to piecepackr::grid.cropmark(). If NULL defaults to max(max(layout$bleed), 0.125).

Details

- This function draws in **inches** so make sure your graphics device is "big" enough.

Value

NULL invisibly. As a side effect draws crop marks to the active graphics device.

See Also

pdf_add_cropmarks(), piecepackr::grid.cropmark()

Examples

```
if (requireNamespace("piecepackr", quietly = TRUE) &&
    utils::packageVersion("piecepackr") >= "1.14.0-6") {
  grid::grid.newpage()
  vp <- grid::viewport(width=8.5, height=11, default.units="in",
                       x=0.5, y=0.5, just=c("left", "bottom"))
  grid::pushViewport(vp)
  grid_add_rects(layout = "poker_3x3")
  grid_add_cropmarks(layout = "poker_3x3")
  grid::popViewport()
}
```

---

grid_add_crosshairs *Draw (rounded) rectangles around components*

Description

grid_add_rects() draws (rounded) rectangles around components of a print-and-play layout.

Usage

```
grid_add_crosshairs(..., layout = "poker_3x3")
```

Arguments

...	Passed to piecepackr::grid.crosshair().
layout	Either a layout preset name in layout_names() or a data frame with layout data (as returned by layout_grid()).

Details

- This function draws in **inches** so make sure your graphics device is "big" enough.

Value

NULL invisibly. As a side effect draws crosshairs to the active graphics device.

Examples

```
if (requireNamespace("piecepackr", quietly = TRUE) &&
    utils::packageVersion("piecepackr") >= "1.14.0-5") {
  grid::grid.newpage()
  vp <- grid::viewport(width=8.5, height=11, default.units="in",
                       x=0.5, y=0.5, just=c("left", "bottom"))
  grid::pushViewport(vp)
  grid_add_crosshairs(layout = "poker_3x3")
  grid::popViewport()
}
```

---

grid_add_lines *Draw lines along component edges*

Description

`grid_add_lines()` draws lines along the components of a print-and-play layout.

Usage

```
grid_add_lines(..., layout = "poker_3x3", gp = gpar())
```

Arguments

`...`	Ignored for now.
`layout`	Either a layout preset name in `layout_names()` or a data frame with layout data (as returned by `layout_grid()`).
`gp`	Passed to `grid::grid.segments()`.

Details

- This function draws in **inches** so make sure your graphics device is "big" enough.

Value

`NULL` invisibly. As a side effect draws rectangles to the active graphics device.

See Also

`grid::grid.segments()`

Examples

```
grid::grid.newpage()
vp <- grid::viewport(width=8.5, height=11, default.units="in",
                     x=0.0, y=0.0, just=c("left", "bottom"))
grid::pushViewport(vp)
grid_add_lines(layout = "poker_3x3",
               gp = grid::gpar(lty = "dashed", col = "grey"))
grid::popViewport()
```

grid_add_rects *Draw (rounded) rectangles around components*

Description

`grid_add_rects()` draws (rounded) rectangles around components of a print-and-play layout.

Usage

```
grid_add_rects(
  ...,
  layout = "poker_3x3",
  r = unit(0, "in"),
  gp = gpar(col = "black", fill = NA, lwd = 1)
)
```

Arguments

`...`	Ignored for now.
`layout`	Either a layout preset name in `layout_names()` or a data frame with layout data (as returned by `layout_grid()`).
`r`, `gp`	Passed to `grid::grid.roundrect()`.

Details

- This function draws in **inches** so make sure your graphics device is "big" enough.
- Sometimes if you use the same color as a solid background color this can be used to effectively "remove" unwanted card outlines.

Value

`NULL` invisibly. As a side effect draws rectangles to the active graphics device.

See Also

`pdf_add_rects()`, `grid::grid.roundrect()`

Examples

```
grid::grid.newpage()
vp <- grid::viewport(width=8.5, height=11, default.units="in",
                     x=0.5, y=0.5, just=c("left", "bottom"))
grid::pushViewport(vp)
grid_add_rects(layout = "poker_3x3")
grid::popViewport()
```

`layout_grid` *Layout data frame for a grid of identically sized components*

Description

`layout_grid()` calculates a layout data frame for a grid of identically sized print-and-play components.

Usage

```
layout_grid(
  nrow = 2L,
  ncol = 1L,
  width = 2.5,
  height = 3.5,
  bleed = 0,
  paper = c("letter", "a4"),
  orientation = c("landscape", "portrait")
)
```

Arguments

`nrow`, `ncol`	Number of rows and columns of print-and-play components (e.g. cards)
`width`, `height`, `bleed`	Width, height, and bleed for each print-and-play component (in inches)
`paper`, `orientation`	print-and-play paper size and orientation

Value

A data frame with columns "row", "col", "x", "y", "width", "height", "bleed", "paper", and "orientation".

Examples

```
# Button Shy card layout
layout_grid(nrow = 2L, ncol = 3L, height = 3.447, width = 2.469, bleed = 0.125)
```

`layout_preset` *Layout data frame for a named preset*

Description

`layout_preset()` calculates a layout data frame for a named preset. `layout_names()` returns the supported layout presets.

Usage

```
layout_preset(name = "button_shy_cards")

layout_names()
```

Arguments

`name`	Preset name. Must be in `layout_names()`.

Details

Unless otherwise indicated assumes letter-sized paper. Supports the following presets:

button_shy_cards Button Shy Games PnP cards. **Caveat:** matches many but not **all** of their PnP files e.g. `ROVE-PNP-V2.pdf` better matches the `poker_3x2_bleed` layout.

button_shy_rules Button Shy Games PnP rule booklet pages.

poker_3x2_bleed Poker-sized cards (2.5" by 3.5") in 3 columns of 2 cards (landscape) with an 1/8" bleed around each card. Examples of PnP games using this layout include Galdor's Grip.

poker_3x3 Poker-sized cards (2.5" by 3.5") in 3 columns of 3 cards (portrait) with an zero bleed around each card. Examples of PnP games using this layout include the original Mini Rogue.

poker_4x2 Poker-sized cards (2.5" by 3.5") in 4 columns of 2 cards (landscape) with zero bleed around each card. Examples of PnP games using this layout include the Decktet.

Value

A data frame with columns "row", "col", "x", "y", "width", "height", "bleed", "paper", and "orientation".

Examples

```
layout_preset("button_shy_cards")
layout_names()
```

ls_temp_pdfs *List and remove temporary pdfs*

Description

Using the functions in this package can lead to several pdf files cluttering your `tempdir()`. `ls_temp_pdfs()` lists them while `rm_temp_pdfs()` removes them.

Usage

```
ls_temp_pdfs(exclude = character())

rm_temp_pdfs(exclude = character())
```

Arguments

`exclude`	A character vector of filenames to exclude.

Value

`ls_temp_pdfs()` returns a character vector. `rm_temp_pdfs()` returns `invisible(NULL)`.

Examples

```
## Not run: # May delete user pdf files in `tempdir()`
f1 <- pdf_create_blank()
f2 <- pdf_compress(f1)
f3 <- pdf_subset(f2, pages = 1L)
ls_temp_pdfs()
rm_temp_pdfs()
ls_temp_pdfs()

## End(Not run)
```

pdf_add_cropmarks *Add crop marks to a pdf*

Description

`pdf_add_cropmarks()` adds crop marks to the edges of components of a print-and-play layout.

Usage

```
pdf_add_cropmarks(
  input,
  output = NULL,
  ...,
  layout = "poker_3x3",
  pages = "even",
  dpi = 300,
  bleed = NULL
)
```

Arguments

input	Input pdf filename.
output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Passed to piecepackr::grid.cropmark().
layout	Either a layout preset name in layout_names() or a data frame with layout data (as returned by layout_grid()).
pages	A positive numeric vector of pages to include, a negative numeric vector of pages to exclude, or a string: **all** Include all the pages. **even** Include just the even pages. **odd** Include just the odd pages. **2-up saddle stitch** The order of the pages to create a saddle-stitch booklet if printing 2-up.
dpi	Dots per inch. Passed to pdftools::pdf_render_page().
bleed	Passed to piecepackr::grid.cropmark(). If NULL defaults to max(max(layout$bleed), 0.125).

Details

- The original pdf document will be rasterized.

Value

output pdf file name invisibly. As a side effect adds crop marks to a pdf.

See Also

grid_add_cropmarks(), piecepackr::grid.cropmark()

Examples

```
if (requireNamespace("piecepackr", quietly = TRUE)) {
  input <- pdf_create_blank(length = 2L, width = 11, height = 8.5)
  output <- pdf_add_cropmarks(input, pages = "odd",
                              layout = "button_shy_cards", dpi = 75)
```

```
  unlink(input)
  unlink(output)
}
```

`pdf_add_crosshairs` *Add crosshairs to a pdf*

Description

`pdf_add_crosshairs()` adds crosshairs to the corners of components of a print-and-play layout.

Usage

```
pdf_add_crosshairs(
  input,
  output = NULL,
  ...,
  layout = "button_shy_cards",
  pages = "even",
  dpi = 300
)
```

Arguments

`input`	Input pdf filename.
`output`	Output pdf filename. `NULL` defaults to `tempfile(fileext = ".pdf")`.
`...`	Passed to `piecepackr::grid.crosshair()`.
`layout`	Either a layout preset name in `layout_names()` or a data frame with layout data (as returned by `layout_grid()`).
`pages`	A positive numeric vector of pages to include, a negative numeric vector of pages to exclude, or a string: **all** Include all the pages. **even** Include just the even pages. **odd** Include just the odd pages. **2-up saddle stitch** The order of the pages to create a saddle-stitch booklet if printing 2-up.
`dpi`	Dots per inch. Passed to `pdftools::pdf_render_page()`.

Details

- The default layout supports Button Shy games.
- The original pdf document will be rasterized.

Value

`output` pdf file name invisibly. As a side effect adds crosshairs to a pdf.

Examples

```
if (requireNamespace("piecepackr", quietly = TRUE) &&
    utils::packageVersion("piecepackr") >= "1.14.0-5") {
  input <- pdf_create_blank(length = 2L, width = 11, height = 8.5)
  output <- pdf_add_crosshairs(input, pages = "odd",
                               layout = "button_shy_cards", dpi = 75)
  unlink(input)
  unlink(output)
}
```

`pdf_add_origami` *Add origami symbols to a pdf*

Description

`pdf_add_origami()` adds origami symbols to the pdf. Currently only supports adding origami symbols to Boardgame Barrio's Small Board Game Jackets.

Usage

```
pdf_add_origami(input, output = NULL, ..., dpi = 300)
```

Arguments

`input`	Input pdf filename.
`output`	Output pdf filename. `NULL` defaults to `tempfile(fileext = ".pdf")`.
`...`	Ignored.
`dpi`	Dots per inch. Passed to `pdftools::pdf_render_page()`.

Value

`output` pdf file name invisibly. As a side effect creates pdf file with added origami symbols.

Examples

```
f1 <- pnpmisc:::pdf_create_mock_sbgj()
f2 <- pdf_add_origami(f1)

unlink(f1)
unlink(f2)
```

pdf_add_rects *Add (rounded) rectangles to a pdf*

Description

`pdf_add_rects()` adds (rounded) rectangles around components of a print-and-play layout.

Usage

```
pdf_add_rects(
  input,
  output = NULL,
  ...,
  layout = "poker_3x3",
  pages = "all",
  dpi = 300,
  r = unit(0, "in"),
  gp = gpar(col = "black", fill = NA, lwd = 1)
)
```

Arguments

`input`	Input pdf filename.
`output`	Output pdf filename. `NULL` defaults to `tempfile(fileext = ".pdf")`.
`...`	Ignored for now.
`layout`	Either a layout preset name in `layout_names()` or a data frame with layout data (as returned by `layout_grid()`).
`pages`	A positive numeric vector of pages to include, a negative numeric vector of pages to exclude, or a string: **all** Include all the pages. **even** Include just the even pages. **odd** Include just the odd pages. **2-up saddle stitch** The order of the pages to create a saddle-stitch booklet if printing 2-up.
`dpi`	Dots per inch. Passed to `pdftools::pdf_render_page()`.
`r, gp`	Passed to `grid::grid.roundrect()`.

Details

- Sometimes if you use the same color as a solid background color this can be used to effectively "remove" unwanted card outlines.

Value

`output` pdf file name invisibly. As a side effect creates pdf file with added origami symbols.

See Also

grid_add_rects(), grid::grid.roundrect()

Examples

```
f1 <- pdf_create_blank(length = 2L, paper = "letter")
f2 <- pdf_add_rects(f1, layout = "poker_3x3", dpi = 75)
# "Remove" unwanted card border lines by covering up with white
f3 <- pdf_add_rects(f2, layout = "poker_3x3", dpi = 75,
                    gp = grid::gpar(col = "white", fill = NA, lwd = 2))
unlink(f1)
unlink(f2)
unlink(f3)
```

---

pdf_append_blank *Append blank pages to a pdf*

Description

pdf_append_blank() appends blank pages to a pdf.

Usage

```
pdf_append_blank(input, output = NULL, ..., minimum = 1L, multiples_of = 1L)
```

Arguments

input	Input pdf filename.
output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Ignored.
minimum	Final number of pages should be at least this integer.
multiples_of	Final number of pages should be a multiple of this integer.

Examples

```
f1 <- pdf_create_blank(length = 1L)
f2 <- pdf_append_blank(f1, multiples_of = 4L)
qpdf::pdf_length(f2)

# Clean up
unlink(f1)
unlink(f2)
```

pdf_clean *Copies pdf file while removing temporary pdf files*

Description

pdf_clean() copies input pdf file to output and removes temporary pdf files with rm_temp_pdfs(exclude = output).

Usage

```
pdf_clean(input, output = NULL, ...)
```

Arguments

input	Input pdf filename.
output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Ignored.

Value

output pdf file name invisibly. As a side effect copies input to output and removes temporary pdf files.

Examples

```
## Not run: # May delete user files in `tempdir()`
f1 <- pdf_create_blank()
f2 <- pdf_compress(f1)
f3 <- pdf_subset(f2, pages = 1L)
ls_temp_pdfs()
pdf_clean(f3, "output.pdf")
ls_temp_pdfs()
unlink("output.pdf")

## End(Not run)
```

pdf_compress *Wrappers around qpdf functions*

Description

These functions wrap around the utilities in the qpdf package but alters the arguments to match this package's conventions: the first two arguments must be input and output, by default output = tempfile(fileext = ".pdf"), and all other arguments must be named.

Usage

```
pdf_compress(input, output = NULL, ...)

pdf_rotate_pages(input, output = NULL, ..., pages = "all")

pdf_subset(input, output = NULL, ..., pages = 1L)
```

Arguments

input	Input pdf filename.
output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Passed to the underlying qpdf functions.
pages	A positive numeric vector of pages to include, a negative numeric vector of pages to exclude, or a string: **all** Include all the pages. **even** Include just the even pages. **odd** Include just the odd pages. **2-up saddle stitch** The order of the pages to create a saddle-stitch booklet if printing 2-up.

Value

output filename of new pdf file invisibly.

See Also

qpdf::pdf_compress(), qpdf::pdf_rotate_pages(), qpdf::pdf_subset()

Examples

```
f1 <- pdf_create_blank(width = 6, height = 4)
f2 <- pdf_compress(f1)
f3 <- pdf_subset(f2, pages = 1L)
f4 <- pdf_rotate_pages(f3, angle = 90)

unlink(f1)
unlink(f2)
unlink(f3)
unlink(f4)
```

pdf_create_blank *Create pdf of blank pages*

Description

pdf_create_blank() creates blank pdf pages.

Usage

```
pdf_create_blank(
  output = NULL,
  ...,
  length = 1L,
  paper = c("special", "letter", "a4"),
  orientation = c("portrait", "landscape"),
  width = 8.5,
  height = 11,
  bg = "white"
)
```

Arguments

output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Ignored.
length	Number of pages to create.
paper	Paper size. Either "letter", "a4", or "special".
orientation	Either "portrait" or "landscape". Ignored if paper = "special".
width, height	Paper size in inches if paper = "special".
bg	output pdf background color.

Value

output pdf file name invisibly. As a side effect creates a blank pdf file.

Examples

```
f1 <- pdf_create_blank(paper = "a4", orientation = "landscape")
f2 <- pdf_create_blank(length = 4L)
unlink(f1)
unlink(f2)
```

pdf_create_jacket *Create printable 4x6 photo box jacket pdf*

Description

`pdf_create_jacket()` creates a printable 4x6 photo box jacket.

Usage

```
pdf_create_jacket(
  output = NULL,
  ...,
  front = NULL,
  back = NULL,
  spine = NULL,
  paper = c("letter", "a4")
)
```

Arguments

`output`	Output pdf filename. `NULL` defaults to `tempfile(fileext = ".pdf")`.
`...`	Ignored.
`front`	Fill color/pattern/gradient or grob for front cover section of wallet. Will be drawn in a viewport with a width of 4.139 inches and a height of 6.141 inches. If `NULL` (default) we draw a template.
`back`	Fill color/pattern/gradient or grob for back cover section of wallet. Will be drawn in a viewport with a width of 4.139 inches and a height of 6.141 inches. If `NULL` (default) we draw a template.
`spine`	Fill color/pattern/gradient or grob for spine section of wallet. Will be drawn in a rotated viewport with a width of 6.141 inches and a height of 1.052 inches. If `NULL` (default) we draw a template.
`paper`	Paper size. Either "letter", "a4", or "special".

Details

To make the 4x6 photo jacket from the pdf file:

1. Print it out
2. Use the crop marks to trim off the left and right edge
3. Make the two indicated mountain folds on both sides of the spine
4. Trim off the top and bottom edge
5. Insert into the 4x6 photo box

See Also

pdf_add_origami() to add origami symbols to pre-existing Boardgame Barrio's Small Board Game Jackets.

Examples

```
# Template `front`, `back`, and `spine`
f1 <- pdf_create_jacket()
unlink(f1)

# Fill `front`, `back`, and `spine`
f2 <- pdf_create_jacket(front = "red", back = "blue", spine = "purple")
unlink(f2)

# Grob `front`, `back`, and `spine`
if (require("gridpattern", quietly = TRUE)) {
  pal <- grDevices::palette()
  herringbone <- patternGrob("polygon_tiling", type = "herringbone",
                             fill = pal[2], spacing = 0.1)
  rhombille <- patternGrob("polygon_tiling", type = "rhombille",
                           fill = pal[3], spacing = 0.3)
  pythagorean <- patternGrob("polygon_tiling", type = "pythagorean",
                             fill = pal[4], spacing = 0.1)
  f3 <- pdf_create_jacket(front = herringbone, back = pythagorean,
                          spine = rhombille, bleed = TRUE)
  unlink(f3)
}
```

---

pdf_create_wallet *Create print-and-play card wallet pdf*

Description

pdf_create_wallet() creates print-and-play card origami wallets pdfs.

Usage

```
pdf_create_wallet(
  output = NULL,
  ...,
  front = NULL,
  back = NULL,
  spine = NULL,
  bleed = unit(0, "in"),
  paper = c("letter", "a4")
)
```

Arguments

output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Ignored.
front	Fill color/pattern/gradient or grob for front cover section of wallet. Will be drawn in a rotated masked viewport with a width of 2 + 2 * bleed inches and a height of 8 + 2 * bleed inches. If NULL (default) we draw a template.
back	Fill color/pattern/gradient or grob for back cover section of wallet. Will be drawn in a rotated masked viewport with a width of 2 * 2 * bleed inches and a height of 8 + 2 * bleed inches. If NULL (default) we draw a template.
spine	Fill color/pattern/gradient or grob for spine section of wallet. Will be drawn in a masked viewport with a width of 8 + 2 * bleed inches and a height of 6 + 2 * bleed inches. If NULL (default) we draw a template.
bleed	Bleed zone size to assume: • If bleed is a grid::unit() simply use it • If bleed is numeric then convert via grid::unit(bleed, "in") • If bleed is TRUE assume 1/8 inch bleed zone size • If bleed is FALSE assume 0 inch bleed zone size
paper	Paper size. Either "letter", "a4", or "special".

Details

To make the wallets from the pdf files:

1. Print out two-sided flipping on the **long** edge
2. Using the crop marks on the first page (the page showing the "spine") trim the edges with a cutting tool (the four dots should now be at the corner of each page).
3. Make the four corner valley folds each labeled "Folds 1"
4. Make the two valley folds each labeled "Folds 2"
5. Flip over the paper and make the two valley folds each labeled "Folds 3"
6. Fold in half (valley fold)

Value

output pdf file name invisibly. As a side effect creates a print-and-play card wallet pdf file.

Examples

```
# Template `front`, `back`, and `spine`
f1 <- pdf_create_wallet()
unlink(f1)

# Fill `front`, `back`, and `spine`
f2 <- pdf_create_wallet(front = "red", back = "blue", spine = "purple")
unlink(f2)
```

```
# Grob `front`, `back`, and `spine` with bleed
if (require("gridpattern", quietly = TRUE)) {
  pal <- grDevices::palette()
  herringbone <- patternGrob("polygon_tiling", type = "herringbone",
                             fill = pal[2], spacing = 0.1)
  rhombille <- patternGrob("polygon_tiling", type = "rhombille",
                           fill = pal[3], spacing = 0.2)
  pythagorean <- patternGrob("polygon_tiling", type = "pythagorean",
                             fill = pal[4], spacing = 0.1)
  f3 <- pdf_create_wallet(front = herringbone, back = rhombille,
                          spine = pythagorean, bleed = TRUE)
  unlink(f3)
}
```

---

pdf_gs *Process the pdf file with ghostscript*

Description

pdf_gs() processes the pdf file with ghostscript. This may prevent issues with other pdf processing functions like pdftools::pdf_pagesize().

Usage

```
pdf_gs(input, output = NULL, ..., args = character(0L))
```

Arguments

input	Input pdf filename.
output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Ignored.
args	Arguments to pass to ghostscript. Automatically adds -dBATCH -dNOPAUSE -sDEVICE=pdfwrite -sAutoRotatePages=None -sOutputFile={output}.

Examples

```
if (tools::find_gs_cmd()[[1L]] != "") {
  f1 <- pdf_create_blank()
  f2 <- pdf_gs(f1)

  unlink(f1)
  unlink(f2)
}
```

pdf_orientation *Tell whether pdf is in portrait or landscape mode*

Description

pdf_orientation() tells whether a pdf is in portrait or landscape mode.

Usage

```
pdf_orientation(input, ...)
```

Arguments

input	Input pdf filename.
...	Ignored.

Value

A character vector with a length equal to the number of pages in input.

Examples

```
f1 <- pdf_create_blank(width = 8.5, height = 11)
pdf_orientation(f1)

f2 <- pdf_create_blank(width = 11, height = 8.5, length = 2L)
pdf_orientation(f2)

unlink(f1)
unlink(f2)
```

pdf_pad_paper *Pad pdf file (to a larger paper size)*

Description

pdf_pad_paper() makes a pdf file larger by padding it (i.e. adding space to the outside margins). The original images are **not** rescaled.

Usage

```
pdf_pad_paper(
  input,
  output = NULL,
  ...,
  bg = "white",
  dpi = 300,
  paper = c("letter", "a4")
)
```

Arguments

input	Input pdf filename.
output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Ignored.
bg	output pdf background color.
dpi	Dots per inch. Passed to pdftools::pdf_render_page().
paper	Paper size. Either "letter", "a4", or "special".

Value

output pdf file name invisibly. As a side effect creates padded pdf file.

Examples

```
# Some PnP files' size is the intersection of A4/letter page sizes
# i.e. shorter than A4 and narrower than letter.
# We usually want pad to full A4 or letter page size.
input <- tempfile(fileext = ".pdf")
grDevices::pdf(input, width = 8.3, height = 11, bg = "blue")
grid::grid.text("")
invisible(grDevices::dev.off())

pdf_width(input)
pdf_height(input)

output <- pdf_pad_paper(input, dpi = 75)
pdf_width(output)
pdf_height(output)
unlink(output)

output_a4 <- pdf_pad_paper(input, dpi = 75, paper = "a4")
pdf_width(output_a4)
pdf_height(output_a4)
unlink(output_a4)

unlink(input)
```

pdf_pages *Get integer vector of subset of pdf pages*

Description

`pdf_pages()` calculates an integer vector of subset of pdf pages.

Usage

```
pdf_pages(input, ..., pages = c("all", "even", "odd", "2-up saddle stitch"))
```

Arguments

`input`	Input pdf filename.
`...`	Ignored.
`pages`	A positive numeric vector of pages to include, a negative numeric vector of pages to exclude, or a string: **all** Include all the pages. **even** Include just the even pages. **odd** Include just the odd pages. **2-up saddle stitch** The order of the pages to create a saddle-stitch booklet if printing 2-up.

Value

An integer vector.

Examples

```
f <- pdf_create_blank(length = 8L)
pdf_pages(f, pages = 1:4)
pdf_pages(f, pages = -(1:4))
pdf_pages(f, pages = "all")
pdf_pages(f, pages = "even")
pdf_pages(f, pages = "odd")

unlink(f) # Clean up
```

pdf_render_bm_pixmap *Render a pdf page into a bittermelon pixmap object*

Description

`pdf_render_bm_pixmap()` renders a pdf page into a bittermelon pixmap object.

Usage

```
pdf_render_bm_pixmap(input, ..., page = 1L, dpi = 300)
```

Arguments

`input`	Input pdf filename.
`...`	Ignored.
`page`	Integer of page to render.
`dpi`	Dots per inch. Passed to `pdftools::pdf_render_page()`.

Value

A `bittermelon::bm_pixmap()` object.

See Also

`pdftools::pdf_render_page()`, `pdf_render_raster()`

Examples

```
if (requireNamespace("bittermelon", quietly = TRUE)) {
  f <- pdf_create_wallet()
  bm <- pdf_render_bm_pixmap(f, page = 1L, dpi = 75)
  grid::grid.raster(bm)
  unlink(f)
}
```

pdf_render_raster *Render a pdf page into a raster object*

Description

`pdf_render_raster()` renders a pdf page into a raster object.

Usage

```
pdf_render_raster(input, ..., page = 1L, dpi = 300, native = FALSE)
```

Arguments

input	Input pdf filename.
...	Ignored.
page	Integer of page to render.
dpi	Dots per inch. Passed to pdftools::pdf_render_page().
native	If TRUE return a nativeRaster object else a raster object.

Value

If native = TRUE returns a nativeRaster object else a raster object.

See Also

pdftools::pdf_render_page(), pdf_render_bm_pixmap()

Examples

```
if (requireNamespace("bittermelon", quietly = TRUE)) {
  f <- pdf_create_wallet()
  r <- pdf_render_raster(f, page = 1L, dpi = 75)
  grid::grid.raster(r)
  unlink(f)
}
```

---

pdf_rm_crosshairs *Remove crosshairs*

Description

pdf_rm_crosshairs() removes unwanted crosshairs.

Usage

```
pdf_rm_crosshairs(
  input,
  output = NULL,
  ...,
  layout = "poker_3x2_bleed",
  pages = "odd",
  dpi = 300
)
```

Arguments

input	Input pdf filename.
output	Output pdf filename. NULL defaults to tempfile(fileext = ".pdf").
...	Ignored.
layout	Either a layout preset name in layout_names() or a data frame with layout data (as returned by layout_grid()).
pages	A positive numeric vector of pages to include, a negative numeric vector of pages to exclude, or a string: **all** Include all the pages. **even** Include just the even pages. **odd** Include just the odd pages. **2-up saddle stitch** The order of the pages to create a saddle-stitch booklet if printing 2-up.
dpi	Dots per inch. Passed to pdftools::pdf_render_page().

Details

- In order to work the PnP layout needs a solid color bleed.
- The default layout supports Galdor's Grip (PnP letter size v1).
- The original pdf document will be rasterized.

Value

output pdf file name invisibly. As a side effect removes from crosshairs from a pdf.

Examples

```
if (requireNamespace("bittermelon", quietly = TRUE) &&
    requireNamespace("piecepackr", quietly = TRUE) &&
    utils::packageVersion("piecepackr") >= "1.14.0-5") {
  f1 <- pdf_create_blank(length = 2L, width = 11, height = 8.5)
  f2 <- pdf_add_crosshairs(f1, pages = "all",
                           layout = "poker_3x2_bleed", dpi = 75)
  f3 <- pdf_rm_crosshairs(f2, pages = "odd",
                          layout = "poker_3x2_bleed", dpi = 75)
  unlink(f1)
  unlink(f2)
  unlink(f3)
}
```

pdf_set_bookmarks *Wrappers around xmpdf functions*

Description

These functions wrap around the utilities in the xmpdf package but alters the arguments to match this package's conventions: the first two arguments must be `input` and `output`, by default `output = tempfile(fileext = ".pdf")`, and all other arguments must be named.

Usage

```
pdf_set_bookmarks(input, output = NULL, ..., bookmarks)

pdf_set_docinfo(input, output = NULL, ..., docinfo)

pdf_set_xmp(input, output = NULL, ..., xmp)
```

Arguments

`input`	Input pdf filename.
`output`	Output pdf filename. `NULL` defaults to `tempfile(fileext = ".pdf")`.
`...`	Currently ignored.
`bookmarks`	See `xmpdf::set_bookmarks()`.
`docinfo`	See `xmpdf::set_docinfo()`.
`xmp`	See `xmpdf::set_xmp()`.

Value

`output` filename of new pdf file invisibly.

See Also

`xmpdf::set_bookmarks()`, `xmpdf::set_docinfo()`, `xmpdf::set_xmp()`

Examples

```
f1 <- pdf_create_blank(length = 2L)

if (requireNamespace("xmpdf", quietly = TRUE) &&
    xmpdf::supports_set_bookmarks()) {
  bm <- data.frame(title = c("Page 1", "Page 2"), page = c(1, 2))
  f2 <- pdf_set_bookmarks(f1, bookmarks = bm)
  unlink(f2)
}

if (requireNamespace("xmpdf", quietly = TRUE) &&
    xmpdf::supports_set_docinfo()) {
```

```
  di <- xmpdf::docinfo(title = "A Title", author = "The Author")
  f3 <- pdf_set_docinfo(f1, docinfo = di)
  unlink(f3)
}

if (requireNamespace("xmpdf", quietly = TRUE) &&
    xmpdf::supports_set_xmp()) {
  xmp <- xmpdf::xmp(title = "A Title", creator = "The Author")
  f4 <- pdf_set_xmp(f1, xmp = xmp)
  unlink(f4)
}
unlink(f1)
```

---

pdf_width *Get dimensions of pdf pages*

Description

pdf_width() and pdf_height() get the dimensions of the pdf file pages.

Usage

```
pdf_width(input, ..., units = "inches", numeric = FALSE)

pdf_height(input, ..., units = "inches", numeric = FALSE)
```

Arguments

input	Input pdf filename.
...	Ignored.
units	Units to use. See grid::unit().
numeric	If TRUE return numeric else a grid::unit() object.

Value

If numeric = TRUE a numeric vector else a grid::unit() object.

Examples

```
f <- pdf_create_blank(width = 8.5, height = 11)
pdf_width(f)
pdf_height(f, units = "cm")
pdf_height(f, units = "mm", numeric = TRUE)
unlink(f)
```

Index