Package: minipdf (via r-universe)

May 8, 2026

Type Package

Title PDF Document Creator

Version 0.2.7

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

Description PDF is a standard file format for laying out text and images in documents. At its core, these documents are sequences of objects defined in plain text. This package allows for the creation of PDF documents at a very low level without any library or graphics device dependencies.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/coolbutuseless/minipdf

Depends R (>= 4.1.0)

Imports glue

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

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

Date/Publication 2025-08-31 09:42:48 UTC

RemoteUrl https://github.com/coolbutuseless/minipdf

RemoteRef HEAD

RemoteSha c6b57b2b168e0fa7393f3f91ddcd57b028197948

Contents

as.character.clip_rect 2
as.character.pdf_dict 3
as.character.pdf_stream 3

as.character.pdf_translate . . . 4
clip_polygon . . . 5
clip_rect . . . 5
create_pdf . . . 6
pdf_bezier . . . 7
pdf_circle . . . 8
pdf_clip_polygon . . . 9
pdf_clip_rect . . . 10
pdf_image . . . 11
pdf_line . . . 12
pdf_newpage . . . 13
pdf_polygon . . . 13
pdf_polyline . . . 14
pdf_rect . . . 15
pdf_rotate . . . 16
pdf_scale . . . 16
pdf_text . . . 17
pdf_translate . . . 19
pgpar . . . 19
print.pdf_doc . . . 20
tf_rotate . . . 21
tf_scale . . . 21
tf_translate . . . 22
write_pdf . . . 23

Index **24**

`as.character.clip_rect` *Convert clipping spec into PDF string*

Description

Convert clipping spec into PDF string

Usage

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

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

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

Arguments

x	clip object created with `clip_rect()` or `clip_polygon()`
...	ignored

Value

string representing a clipping specification

---

as.character.pdf_dict *Render* pdf_dict *as character string*

Description

Render pdf_dict as character string

Usage

```
## S3 method for class 'pdf_dict'
as.character(x, depth = 0, ...)
```

Arguments

x	pdf_dict object
depth	print depth. Default: 0. Used to control indentation
...	ignored

Value

Character representation

---

as.character.pdf_stream *Convert* pdf_stream *to character*

Description

Convert pdf_stream to character

Usage

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

Arguments

x	pdf_stream object
...	ignored

Value

character string representation of a pdf stream object

---

as.character.pdf_translate *Convert scale/rotate/translate specification to a PDF transformation string*

Description

Convert scale/rotate/translate specification to a PDF transformation string

Usage

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

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

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

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

Arguments

x	transform specification
...	ignored

Value

String representing a PDF transformation matrix 'cm' operation

clip_polygon *Define a clipping polygon for use as a* clip *argument*

Description

Define a clipping polygon for use as a clip argument

Usage

```
clip_polygon(xs, ys, id = NULL, rule = "winding")
```

Arguments

xs, ys	vertex coordinates. Note: polygon will automatically be closed
id	A numeric vector used to separate vertices into multiple polygons. All vertices with the same id belong to the same polygon. Default: NULL means that all vertices belong to a single polygon.
rule	fill rule. 'winding' or 'evenodd'. Default: 'winding'

Value

clipping polygon specification

See Also

Other clipping functions: clip_rect(), pdf_clip_polygon(), pdf_clip_rect()

Examples

```
doc <- create_pdf() |>
   pdf_rect(0, 0, 100, 100, clip = clip_polygon(xs = c(0, 100, 100),
   ys = c(0, 0, 100)))
```

clip_rect *Define a clipping rectangle for use as a* clip *argument*

Description

Define a clipping rectangle for use as a clip argument

Usage

```
clip_rect(x, y, width, height)
```

Arguments

x, y	position
width, height	size

Value

clipping rectangle specification

See Also

Other clipping functions: clip_polygon(), pdf_clip_polygon(), pdf_clip_rect()

Examples

```
doc <- create_pdf() |>
   pdf_rect(0, 0, 100, 100, clip = clip_rect(50, 50, 200, 200))
```

---

create_pdf *Create an new PDF*

Description

Create an new PDF

Usage

```
create_pdf(
  width = 400,
  height = 400,
  title = NULL,
  author = NULL,
  creator = "minipdf/R",
  creation_date = strftime(Sys.time(), format = "D:%Y%m%d%H%M")
)
```

Arguments

width, height	page size in pixels
title, author, creator, creation_date	Document-level metainformation about this file. Set value to NULL to exclude from PDF.

Value

pdf_doc object (i.e. a named list)

Examples

```
create_pdf()
```

pdf_bezier *Add a cubic bezier to a PDF doc*

Description

Add a cubic bezier to a PDF doc

Usage

```
pdf_bezier(
  doc,
  x0,
  y0,
  x1,
  y1,
  x2,
  y2,
  x3,
  y3,
  ...,
  gp = pgpar(),
  tf = NULL,
  clip = NULL
)
```

Arguments

doc	A pdf_doc object created by create_pdf()
x0, y0, x1, y1, x2, y2, x3, y3	start point, two control points and end point of bezier curve
...	further arguments to be added to gp
gp	A named list gp object created by pgpar()
tf	either a single transform (tf_translate(), tf_scale(), tf_rotate()), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
clip	either a single clip (clip_rect(), clip_polygon()), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

pdf_doc

See Also

Other object creation functions: pdf_circle(), pdf_image(), pdf_line(), pdf_polygon(), pdf_polyline(), pdf_rect(), pdf_text()

Examples

```
doc <- create_pdf() |>
  pdf_bezier(seq(0, 400, 6), 0, 250, 25, 25, 250, 400, 400, lwd = 1, alpha = 0.2)
```

`pdf_circle` *Add a circle to a PDF doc*

Description

Add a circle to a PDF doc

Usage

```
pdf_circle(doc, x, y, r, ..., gp = pgpar(), tf = NULL, clip = NULL)
```

Arguments

`doc`	A `pdf_doc` object created by `create_pdf()`
`x, y, r`	position of centre and radius of circle (Length = 1 or n)
`...`	further arguments to be added to gp
`gp`	A named list gp object created by `pgpar()`
`tf`	either a single transform (`tf_translate()`, `tf_scale()`, `tf_rotate()`), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
`clip`	either a single clip (`clip_rect()`, `clip_polygon()`), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

`pdf_doc`

See Also

Other object creation functions: `pdf_bezier()`, `pdf_image()`, `pdf_line()`, `pdf_polygon()`, `pdf_polyline()`, `pdf_rect()`, `pdf_text()`

Examples

```
doc <- create_pdf() |>
   pdf_circle(x = 200, y = 200, r = 50)
```

pdf_clip_polygon *Add a global clipping polygon to a PDF doc*

Description

Clipping regions are cumulative, and these is no operation to expand the global clipping region. use local clipping with the `clip` argument to individual objects.

Usage

```
pdf_clip_polygon(doc, xs, ys, id = NULL, rule = "winding", tf = NULL)
```

Arguments

doc	A pdf_doc object created by `create_pdf()`
xs, ys	vertex coordinates. Note: polygon will automatically be closed
id	A numeric vector used to separate vertices into multiple polygons. All vertices with the same id belong to the same polygon. Default: NULL means that all vertices belong to a single polygon.
rule	fill rule. 'winding' or 'evenodd'. Default: 'winding'
tf	either a single transform (`tf_translate()`, `tf_scale()`, `tf_rotate()`), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)

Value

pdf_doc

See Also

Other clipping functions: `clip_polygon()`, `clip_rect()`, `pdf_clip_rect()`

Other global clipping functions: `pdf_clip_rect()`

Examples

```
doc <- create_pdf() |>
   pdf_clip_polygon(xs = c(0, 100, 100), ys = c(0, 0, 100))
```

pdf_clip_rect *Add a global clipping rectangle to a PDF doc*

Description

Clipping regions are cumulative, and these is no operation to expand the global clipping region. use local clipping with the `clip` argument to individual objects.

Usage

```
pdf_clip_rect(doc, x, y, width, height, tf = NULL)
```

Arguments

doc	A pdf_doc object created by `create_pdf()`
x, y	position
width, height	size
tf	either a single transform (`tf_translate()`, `tf_scale()`, `tf_rotate()`), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)

Value

pdf_doc

See Also

Other clipping functions: `clip_polygon()`, `clip_rect()`, `pdf_clip_polygon()`

Other global clipping functions: `pdf_clip_polygon()`

Examples

```
doc <- create_pdf() |>
   pdf_clip_rect(0, 0, 200, 200)
```

pdf_image *Add image to a PDF doc*

Description

Add image to a PDF doc

Usage

```
pdf_image(
  doc,
  im,
  x,
  y,
  scale = 1,
  interpolate = FALSE,
  ...,
  gp = pgpar(),
  tf = NULL,
  clip = NULL
)
```

Arguments

doc	A pdf_doc object created by create_pdf()
im	Image represented as a numeric matrix or array with all values in range [0, 255]. **matrix** A gray image **array with 2 planes** Gray image with an alpha channel **array with 3 planes** An RGB image **array with 4 planes** An RGB image with an alpha channel
x, y	position of bottom-left corner of image. (Length = 1)
scale	scale factor when rendering image Default: 1. (Length = 1)
interpolate	Should pixel values be interpolated? Default: FALSE. (Length = 1)
...	further arguments to be added to gp
gp	A named list gp object created by pgpar()
tf	either a single transform (tf_translate(), tf_scale(), tf_rotate()), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
clip	either a single clip (clip_rect(), clip_polygon()), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

pdf_doc

See Also

Other object creation functions: pdf_bezier(), pdf_circle(), pdf_line(), pdf_polygon(), pdf_polyline(), pdf_rect(), pdf_text()

Examples

```
im <- matrix(1:100, 10, 10)
doc <- create_pdf() |>
   pdf_image(im, 20, 20, scale = 2)
```

---

pdf_line *Add a line to a PDF doc*

Description

Add a line to a PDF doc

Usage

```
pdf_line(doc, x1, y1, x2, y2, ..., gp = pgpar(), tf = NULL, clip = NULL)
```

Arguments

doc	A pdf_doc object created by create_pdf()
x1, y1, x2, y2	endpoints (Length = 1 or n)
...	further arguments to be added to gp
gp	A named list gp object created by pgpar()
tf	either a single transform (tf_translate(), tf_scale(), tf_rotate()), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
clip	either a single clip (clip_rect(), clip_polygon()), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

pdf_doc

See Also

Other object creation functions: pdf_bezier(), pdf_circle(), pdf_image(), pdf_polygon(), pdf_polyline(), pdf_rect(), pdf_text()

Examples

```
doc <- create_pdf() |>
   pdf_line(10, 10, 100, 100, col = 'red')
```

pdf_newpage *Start a new page in a PDF odc*

Description

Start a new page in a PDF odc

Usage

```
pdf_newpage(doc)
```

Arguments

doc	A pdf_doc object created by create_pdf()

Value

doc with new page added (and made the current page)

Examples

```
create_pdf() |>
   pdf_newpage()
```

pdf_polygon *Add a polygon to a PDF doc*

Description

Add a polygon to a PDF doc

Usage

```
pdf_polygon(doc, xs, ys, id = NULL, ..., gp = pgpar(), tf = NULL, clip = NULL)
```

Arguments

doc	A pdf_doc object created by create_pdf()
xs, ys	vertex coordinates. Note: polygon will automatically be closed
id	A numeric vector used to separate vertices into multiple polygons. All vertices with the same id belong to the same polygon. Default: NULL means that all vertices belong to a single polygon.
...	further arguments to be added to gp
gp	A named list gp object created by pgpar()

tf	either a single transform (tf_translate(), tf_scale(), tf_rotate()), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
clip	either a single clip (clip_rect(), clip_polygon()), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

pdf_doc

See Also

Other object creation functions: pdf_bezier(), pdf_circle(), pdf_image(), pdf_line(), pdf_polyline(), pdf_rect(), pdf_text()

Examples

```
doc <- create_pdf() |>
   pdf_polygon(xs = c(100, 200, 200), ys = c(100, 100, 200))
```

---

pdf_polyline *Add a polyline to a PDF doc*

Description

Add a polyline to a PDF doc

Usage

```
pdf_polyline(doc, xs, ys, ..., gp = pgpar(), tf = NULL, clip = NULL)
```

Arguments

doc	A pdf_doc object created by create_pdf()
xs, ys	vertex coordinates
...	further arguments to be added to gp
gp	A named list gp object created by pgpar()
tf	either a single transform (tf_translate(), tf_scale(), tf_rotate()), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
clip	either a single clip (clip_rect(), clip_polygon()), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

pdf_doc

See Also

Other object creation functions: pdf_bezier(), pdf_circle(), pdf_image(), pdf_line(), pdf_polygon(), pdf_rect(), pdf_text()

Examples

```
doc <- create_pdf() |>
   pdf_polyline(xs = c(100, 200, 200), ys = c(100, 100, 200))
```

---

pdf_rect *Add a rectangle to a PDF doc*

Description

Add a rectangle to a PDF doc

Usage

```
pdf_rect(doc, x, y, width, height, ..., gp = pgpar(), tf = NULL, clip = NULL)
```

Arguments

doc	A pdf_doc object created by create_pdf()
x, y	position of lower left of rectangle (Length = 1 or n)
width, height	width of height of rectangle (Length = 1 or n)
...	further arguments to be added to gp
gp	A named list gp object created by pgpar()
tf	either a single transform (tf_translate(), tf_scale(), tf_rotate()), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
clip	either a single clip (clip_rect(), clip_polygon()), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

pdf_doc

See Also

Other object creation functions: pdf_bezier(), pdf_circle(), pdf_image(), pdf_line(), pdf_polygon(), pdf_polyline(), pdf_text()

Examples

```
doc <- create_pdf() |>
   pdf_rect(10, 10, 100, 100, gp = pgpar(fill = 'red'))
```

pdf_rotate *Modify global transformation matrix with additional rotation*

Description

Global transformations are cumulative, and these is no operation to reset the global transformation. For local transformations use the `tf` argument for individual objects.

Usage

```
pdf_rotate(doc, rads, x = 0, y = 0)
```

Arguments

`doc`	A `pdf_doc` object created by `create_pdf()`
`rads`	rotation angle in radians
`x, y`	location to rotate around

Value

`pdf_doc`

See Also

Other transform functions: `pdf_scale()`, `pdf_translate()`, `tf_rotate()`, `tf_scale()`, `tf_translate()`

Other global transform functions: `pdf_scale()`, `pdf_translate()`

Examples

```
doc <- create_pdf() |>
   pdf_rotate(rads = pi)
```

pdf_scale *Modify global transformation matrix with additional scaling*

Description

Global transformations are cumulative, and these is no operation to reset the global transformation. For local transformations use the `tf` argument for individual objects.

Usage

```
pdf_scale(doc, x, y = x)
```

Arguments

doc	A pdf_doc object created by create_pdf()
x, y	scale amount in each direction. If 'y' value is not specified it is made the same as the 'x' value

Value

pdf_doc

See Also

Other transform functions: pdf_rotate(), pdf_translate(), tf_rotate(), tf_scale(), tf_translate()

Other global transform functions: pdf_rotate(), pdf_translate()

Examples

```
doc <- create_pdf() |>
   pdf_scale(x = 10)
```

---

pdf_text *Add text to a PDF doc*

Description

Add text to a PDF doc

Usage

```
pdf_text(
  doc,
  text,
  x,
  y,
  fontfamily = "Helvetica",
  fontface = "plain",
  fontsize = 12,
  mode = 0,
  ...,
  gp = pgpar(),
  tf = NULL,
  clip = NULL
)
```

Arguments

doc	A pdf_doc object created by create_pdf()
text	string
x, y	position (Length = 1 or N)
fontfamily	Font name. Default: 'Helvetica'. One of: "Helvetica", "Courier", "Times", "Symbol", "ZapfDingbats". 'sans', 'mono' and 'serif' also accepted for 'Helvetica', 'Courier' and 'Times', respectively. (Length = 1)
fontface	Font styling. Default: 'plain'. One of: 'plain', 'bold', 'italic', 'bold.italic' (Length = 1)
fontsize	Default: 12 (Length = 1)
mode	Default: 0 (Length = 1) • 0 - Fill text. Normal. Default • 1 - Stroke text • 2 - Fill then stroke • 3 - NO fill or stroke. Invisible • 4 - Fill text and add to path for clipping • 5 - Stroke text and add to path for clipping • 6 - Fill, then stroke text and add to path for clipping • 7 - Add text to path for clipping
...	further arguments to be added to gp
gp	A named list gp object created by pgpar()
tf	either a single transform (tf_translate(), tf_scale(), tf_rotate()), or a list of these transforms. Default: NULL, no local transformation applied (global transformations still apply)
clip	either a single clip (clip_rect(), clip_polygon()), or a list of these clips. Default: NULL, no local clipping applied (global clipping still applicable)

Value

pdf_doc

See Also

Other object creation functions: pdf_bezier(), pdf_circle(), pdf_image(), pdf_line(), pdf_polygon(), pdf_polyline(), pdf_rect()

Examples

```
doc <- create_pdf() |>
   pdf_text("Hello", x = 20, y = 20, fontsize = 50)
```

pdf_translate *Modify global transformation matrix with additional translation*

Description

Global transformations are cumulative, and these is no operation to reset the global transformation. For local transformations use the `tf` argument for individual objects.

Usage

```
pdf_translate(doc, x, y)
```

Arguments

doc	A pdf_doc object created by `create_pdf()`
x, y	translation

Value

pdf_doc

See Also

Other transform functions: `pdf_rotate()`, `pdf_scale()`, `tf_rotate()`, `tf_scale()`, `tf_translate()`

Other global transform functions: `pdf_rotate()`, `pdf_scale()`

Examples

```
doc <- create_pdf() |>
   pdf_translate(x = 10, y = 10)
```

pgpar *Create graphical parameters for PDF objects*

Description

This is similar to `grid::gpar()` except that values can only be scalars (i.e. length = 1)

Usage

```
pgpar(
  col = "black",
  fill = "black",
  alpha = 1,
  lty,
  lwd,
  lineend,
  linejoin,
  linemitre,
  rule
)
```

Arguments

col, fill	set graphics parameters for this object
alpha	additional alpha applied to col, fill
lty, lwd, lineend, linejoin, linemitre	line optins
rule	fill rule. 'winding' (default) or 'evenodd'

Value

a graphics parameter object

Examples

```
pgpar()
```

---

print.pdf_doc *Print a 'pdf' object to the console*

Description

Print a 'pdf' object to the console

Usage

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

Arguments

x	pdf object
...	ignored

Value

None

---

tf_rotate *Create a rotation specification (for use as* tf *argument)*

Description

Create a rotation specification (for use as tf argument)

Usage

```
tf_rotate(rads, x = 0, y = 0)
```

Arguments

rads	rotation angle in radians
x, y	location to rotate around

Value

rotation specification

See Also

Other transform functions: pdf_rotate(), pdf_scale(), pdf_translate(), tf_scale(), tf_translate()

Examples

```
doc <- create_pdf() |>
   pdf_text(text = "hello", x = 0, y = 0, tf = tf_rotate(rads = pi))
```

---

tf_scale *Create a scaling specification (for use as* tf *argument)*

Description

Create a scaling specification (for use as tf argument)

Usage

```
tf_scale(x, y = x)
```

Arguments

x, y	scale amount in each direction. If 'y' value is not specified it is made the same as the 'x' value

Value

scale transform specification

See Also

Other transform functions: pdf_rotate(), pdf_scale(), pdf_translate(), tf_rotate(), tf_translate()

Examples

```
doc <- create_pdf() |>
   pdf_text(text = "hello", x = 0, y = 0, tf = tf_scale(x = 10))
```

---

tf_translate *Create a translation specification (for use as* tf *argument)*

Description

Create a translation specification (for use as tf argument)

Usage

```
tf_translate(x, y)
```

Arguments

x, y	translation

Value

translation specification

See Also

Other transform functions: pdf_rotate(), pdf_scale(), pdf_translate(), tf_rotate(), tf_scale()

Examples

```
doc <- create_pdf() |>
   pdf_text(text = "hello", x = 0, y = 0, tf = tf_translate(x = 10, y = 10))
```

`write_pdf` *Write pdf to file or string*

Description

Write pdf to file or string

Usage

```
write_pdf(doc, filename = NULL)
```

Arguments

`doc`	A `pdf_doc` object created by `create_pdf()`
`filename`	Output filename. Default: NULL means no output to file but return a string representation of the PDF

Value

string or None

Examples

```
create_pdf() |>
   pdf_circle(200, 200, 50, lwd = 5, fill = 'hotpink') |>
   write_pdf() |>
   cat()
```

Index

∗ **clipping functions**
clip_polygon, 5
clip_rect, 5
pdf_clip_polygon, 9
pdf_clip_rect, 10
∗ **global clipping functions**
pdf_clip_polygon, 9
pdf_clip_rect, 10
∗ **global transform functions**
pdf_rotate, 16
pdf_scale, 16
pdf_translate, 19
∗ **object creation functions**
pdf_bezier, 7
pdf_circle, 8
pdf_image, 11
pdf_line, 12
pdf_polygon, 13
pdf_polyline, 14
pdf_rect, 15
pdf_text, 17
∗ **transform functions**
pdf_rotate, 16
pdf_scale, 16
pdf_translate, 19
tf_rotate, 21
tf_scale, 21
tf_translate, 22

as.character.clip_list
*(*as.character.clip_rect*)*, 2
as.character.clip_polygon
*(*as.character.clip_rect*)*, 2
as.character.clip_rect, 2
as.character.pdf_dict, 3
as.character.pdf_rotate
*(*as.character.pdf_translate*)*, 4
as.character.pdf_scale
*(*as.character.pdf_translate*)*, 4
as.character.pdf_stream, 3
as.character.pdf_transform_list
*(*as.character.pdf_translate*)*, 4
as.character.pdf_translate, 4

clip_polygon, 5, *6*, *9, 10*
clip_rect, 5, *5*, *9, 10*
create_pdf, 6, *7–19*, *23*

pdf_bezier, 7, *8*, *12*, *14, 15*, *18*
pdf_circle, *7*, 8, *12*, *14, 15*, *18*
pdf_clip_polygon, *5, 6*, 9, *10*
pdf_clip_rect, *5, 6*, *9*, 10
pdf_image, *7, 8*, 11, *12*, *14, 15*, *18*
pdf_line, *7, 8*, 12, *12*, *14, 15*, *18*
pdf_newpage, 13
pdf_polygon, *7, 8*, *12*, 13, *15*, *18*
pdf_polyline, *7, 8*, *12*, 14, *14, 15*, *18*
pdf_rect, *7, 8*, *12*, *14*, 15, *15*, *18*
pdf_rotate, 16, *17*, *19*, *21, 22*
pdf_scale, 16, *16*, *19*, *21, 22*
pdf_text, *7, 8*, *12*, *14, 15*, 17
pdf_translate, *16, 17*, 19, *21, 22*
pgpar, *7, 8*, *11–15*, *18*, 19
print.pdf_doc, 20

tf_rotate, *16, 17*, *19*, 21, *22*
tf_scale, *16, 17*, *19*, 21, *21, 22*
tf_translate, *16, 17*, *19*, *21*, 22, *22*

write_pdf, 23