

# Package ‘cursr’

October 12, 2022

**Type** Package

**Title** Cursor and Terminal Manipulation

**Version** 0.1.0

**Author** Chris Mann

**Maintainer** Chris Mann <cmann3@unl.edu>

**Description** A toolbox for developing applications, games, simulations, or agent-based models in the R terminal. Included functions allow users to move the cursor around the terminal screen, change text colors and attributes, clear the screen, hide and show the cursor, map key presses to functions, draw shapes and curves, among others. Most functionalities require users to be in a terminal (not the R GUI).

**Imports** keypress

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.0.2

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-01-11 08:50:09 UTC

## R topics documented:

attr_off . . . . .	3
attr_on . . . . .	4
bg_off . . . . .	5
bg_on . . . . .	5
box_at . . . . .	6
clear . . . . .	7
color_off . . . . .	8
color_pair . . . . .	9

draw_arc	9
draw_bezier	10
draw_circle	11
draw_ellipse	12
draw_fn	13
draw_lerp	14
draw_path	14
draw_ray	15
draw_rect	16
draw_shape	17
erase	18
example_luckynumber	18
fg_off	19
fg_on	19
fill_circle	20
fill_ellipse	21
fill_rect	22
fill_shape	23
getkp	24
getkpl	24
grid_at	25
grid_mat	26
hide_cursor	27
hline	28
hline_at	28
in.term	29
load_cursor	30
make_bg	30
make_fg	31
make_style	32
mv	32
mv_col	33
mv_row	34
mv_to	35
path_arc	36
path_bezier	36
path_circle	37
path_ellipse	38
path_fill	39
path_fn	39
path_intersection	40
path_lerp	41
path_ray	41
path_rect	42
path_shape	43
repch	44
reset	44
save_cursor	45

*attr\_off* 3

show_cursor . . . . .	45
style . . . . .	46
Sym . . . . .	47
term_dim . . . . .	47
vline . . . . .	48
vline_at . . . . .	48
wr . . . . .	49
wrapup . . . . .	50
wrat . . . . .	50
wrch . . . . .	51
wrchat . . . . .	52
wrkp . . . . .	53
wrkpl . . . . .	54

**Index** 55

---

*attr\_off*                      *Attributes Off*

---

## Description

Turns off text attributes in the terminal, including bold text, italics, underline, etc.

## Usage

`attr_off(...)`

## Arguments

... characters indicating attributes to turn off. "bf" for bold face; "ft" for faint; "it" for italics; "ul" for underline; "sb" for slow blink; "fb" for fast blink; "rv" for reverse video (invert bg and fg colors); "st" for strike-through. All attributes are turned off if left blank.

## Details

Use [attr\\_on](#) to turn on attributes.

## Value

NULL

## See Also

Other style functions: [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

## Examples

```
cat("hello world!\n")
attr_on("bf", "ul")
cat("hello world!\n")
attr_off("bf")
cat("hello world!\n")
attr_off()
cat("hello world!\n")
```

---

attr\_on

*Attributes On*

---

## Description

Turns on text attributes in terminal, including bold text, italics, underline, etc. Note that not all terminals support each attribute.

## Usage

```
attr_on(...)
```

## Arguments

... characters indicating attributes to turn on. "bf" for bold face; "ft" for faint; "it" for italics; "ul" for underline; "sb" for slow blink; "fb" for fast blink; "rv" for reverse video (invert bg and fg colors); "st" for strike-through

## Details

Use [attr\\_off](#) to turn off the attributes.

## Value

NULL

## See Also

Other style functions: [attr\\_off\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

## Examples

```
cat("hello world!\n")
attr_on("bf", "ul")
cat("hello world!\n")
attr_off()
```

---

bg_off	<i>Turn Off Background Color</i>
--------	----------------------------------

---

**Description**

Return the background of future terminal text to the default color. Background color is turned on with [bg\\_on](#).

**Usage**

```
bg_off()
```

**Value**

NULL

**See Also**

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

**Examples**

```
# Different methods of specifying yellow
bg_on("yellow")
bg_on("#FFFF00")
bg_on(11)
bg_on(255, 255, 0)

# Turn off color
bg_off()
```

---

bg_on	<i>Turn On Background Color</i>
-------	---------------------------------

---

**Description**

Specifies the background color of all future text written in the terminal [bg\\_on](#) accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color. Not all terminals support each possible color.

**Usage**

```
bg_on(...)
```

**Arguments**

... character or numeric value

**Details**

Background color is turned off with `bg_off`.

**Value**

NULL

**See Also**

Other style functions: `attr_off()`, `attr_on()`, `bg_off()`, `color_off()`, `color_pair()`, `fg_off()`, `fg_on()`, `make_bg()`, `make_fg()`, `make_style()`, `reset()`, `style()`

**Examples**

```
# Different methods of specifying yellow
bg_on("yellow")
bg_on("#FFFF00")
bg_on(11)
bg_on(255, 255, 0)

# Turn off color
bg_off()
```

---

box\_at

*Draw Box*

---

**Description**

Draws a box of size `dim=c(height, width)` at `yx=c(row,col)`.

**Usage**

```
box_at(
  yx = c(1, 1),
  dim = NULL,
  text = c("|", "|", "-", "-", rep("+", 4)),
  fg = NA,
  bg = NA,
  attr = NA,
  fill = NA,
  fill.bg = NA,
  fill.fg = NA,
  fill.attr = NA
)
```

**Arguments**

<code>yx</code>	starting console row and column of top-left corner of box
<code>dim</code>	box dimensions in <code>c(height, width)</code> . If NA, defaults to the terminal's screen width.
<code>text</code>	repeated character used for box. <code>text</code> can either be a single character or a vector of 8 characters (left side, right side, top, bottom, 4 corners: upper left, upper right, lower left, lower right).
<code>fg</code>	foreground color. See <a href="#">fg_on</a> for more details.
<code>bg</code>	background color. See <a href="#">bg_on</a> for more details.
<code>attr</code>	border text attributes. See <a href="#">attr_on</a> for details.
<code>fill</code>	character object to fill box. Only the first character in the first element is used. If NA (the default), the box is not filled.
<code>fill.bg</code>	background color of the fill character.
<code>fill.fg</code>	foreground color of the fill character.
<code>fill.attr</code>	text attributes of the fill character.

**Value**

NULL

**See Also**

Other drawing functions: [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
box_at(yx=c(4,4), dim=c(5,10), text="X")
```

---

clear

*Clear Text*

---

**Description**

Clear text from the terminal. Passing values "start" or "end" allow the user to clear specific portions of the screen relative to the cursor.

**Usage**

```
clear(x = c("screen", "end", "start"), ...)
```

**Arguments**

x character describing console location to clear. The default, "screen", clears the entire screen; "start" clears all text from the beginning of the screen until the cursor's position; "end" clears all text from the cursor's position to the bottom of the screen.

... objects passed to/from methods

**Value**

NULL

**Examples**

```
clear()

cat(paste(LETTERS[1:10], collapse="\n"))

clear("start")
clear("end")
```

---

color\_off

*Turn Off Colors in Terminal*

---

**Description**

Return the background and foreground of future terminal text to the default colors.

**Usage**

```
color_off()
```

**Value**

NULL

**See Also**

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

**Examples**

```
bg_on("red")
fg_on("yellow")

# Turn off color
color_off()
```

---

`color_pair`*Create Background & Foreground Color Combination*

---

**Description**

Returns the ANSI codes for the specified colors. `color_pair` accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color.

**Usage**

```
color_pair(fg, bg)
```

**Arguments**

<code>fg</code>	character or numeric value for the foreground color
<code>bg</code>	character or numeric value for the background color

**Value**

ANSI character string

**See Also**

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

**Examples**

```
# Blue background with white text
color_pair("white", "blue")
color_pair("#FFFFFF", "#0000FF")
color_pair(0, 12)
color_pair(c(255, 255, 255), c(0,0,255))
```

---

`draw_arc`*Draw an Arc*

---

**Description**

Calculate the path of an arc within a grid and print to screen.

**Usage**

```
draw_arc(yx, start, end, r = 1, n = 50, text = "x", ...)
```

**Arguments**

yx	center (row, col) coordinate of circle
start	starting angle in radians
end	ending angle in radians
r	radius of circle
n	number of points along curve to calculate
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_arc(yx=c(10,10), start=pi/2, end=pi, r=6)
```

---

draw\_bezier

*Draw a Bezier Curve*

---

**Description**

Calculate the path of a Bezier Curve with up to two control points in a grid and draw to screen.

**Usage**

```
draw_bezier(start, end, c1, c2 = NULL, n = 50, text = "x", ...)
```

**Arguments**

start	starting (row, col) coordinate
end	ending (row, col) coordinate
c1	coordinate of first control point
c2	coordinate of second control point
n	number of points along curve to calculate
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_bezier(start=c(10,1), end=c(10,10), c1=c(1, 3))
```

---

draw\_circle

*Draw a Circle*


---

**Description**

Calculate the path of a circle in a grid and draw it to screen.

**Usage**

```
draw_circle(yx, r = 1, n = 50, text = "x", ...)
```

**Arguments**

yx	center (row, col) coordinate
r	radius of the circle in grid points
n	number of points along curve to calculate
text	character value drawn at coordinate
...	parameters that are passed to <a href="#">style()</a> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_circle(yx=c(10,10), r=5)
```

---

`draw_ellipse`*Draw Ellipse*

---

### Description

Calculate the path of an ellipse within a grid and draw to screen.

### Usage

```
draw_ellipse(yx = c(0, 0), rx = 1, ry = 1, n = 50, text = "x", ...)
```

### Arguments

<code>yx</code>	(row, col) coordinate of the center of the ellipse
<code>rx</code>	radius along the x-axis in grid points
<code>ry</code>	radius along the y-axis in grid points
<code>n</code>	number of points along curve to calculate
<code>text</code>	character value drawn at coordinate
<code>...</code>	parameters that are passed to <code>style()</code> , including the foreground color <code>fg</code> , background color <code>bg</code> , and attribute <code>attr</code>

### Value

NULL

### See Also

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

### Examples

```
draw_ellipse(yx=c(10,10), rx=8, ry = 4)
```

---

draw_fn	<i>Draw a Function</i>
---------	------------------------

---

## Description

Calculate the path within a grid of an user-supplied function and print to screen.

## Usage

```
draw_fn(x1, x2, fn, n = 50, text = "x", ...)
```

## Arguments

x1	starting column value of the path
x2	ending column value of the path
fn	function returning row value for a column input
n	number of points along curve to calculate
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color <code>fg</code> , background color <code>bg</code> , and attribute <code>attr</code>

## Value

NULL

## See Also

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

## Examples

```
draw_fn(x1=1, x2=10,  
        function(x){sqrt(x)}  
)
```

---

 draw\_lerp

*Draw a Line*


---

**Description**

Interpolate between two points in a grid and draw to screen.

**Usage**

```
draw_lerp(start, end, n = 50, text = "x", ...)
```

**Arguments**

start	starting (row, col) coordinate
end	ending (row, col) coordinate
n	number of points along curve to calculate
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_lerp(start=c(10,1), end=c(1,3))
```

---

 draw\_path

*Draw Path*


---

**Description**

Draws text at each supplied coordinate.

**Usage**

```
draw_path(coord, text = "x", ...)
```

**Arguments**

coord	matrix or list containing (row, col) coordinates.
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_lerp()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

**Examples**

```
c <- path_circle(yx = c(5,5), r=3)
draw_path(c, text="0")
```

---

draw\_ray

*Draw a Ray*

---

**Description**

Calculate the path of a ray extending and print to screen.

**Usage**

```
draw_ray(start, end, lim = c(64, 128), n = 200, text = "x", ...)
```

**Arguments**

start	start (row, col) coordinate of the ray
end	either an ending coording, an angle in radians, or a character direction (u, d, l, r, ul, ur, dl, dr)
lim	bounding box dimensions used to calculate ray
n	number of points along curve to calculate
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_ray(start=c(10,10), end=pi/6)
draw_ray(start=c(10,10), end=pi/6, lim=c(15,15))
draw_ray(start=c(10,10), end=c(4,2))
```

---

draw\_rect

*Draw a Rectangle*


---

**Description**

Calculate the path of a rectangle in a grid and draw to screen.

**Usage**

```
draw_rect(yx1, yx2, text = "x", ...)
```

**Arguments**

yx1	upper-left (row, col) coordinate
yx2	lower-right (row, col) coordinate
text	character value drawn at coordinate
...	parameters that are passed to <a href="#">style()</a> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_rect(c(5,5), c(9,9))
```

---

draw_shape	<i>Draw a Shape</i>
------------	---------------------

---

### Description

Calculate the path of a shape given supplied vertices and draw to screen.

### Usage

```
draw_shape(mat, cycle = TRUE, n = 30, text = "x", ...)
```

### Arguments

mat	an Nx2 matrix of (row, col) coordinates
cycle	logical value determining whether to the first and last coordinates
n	number of points along each edge to calculate
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

### Value

NULL

### See Also

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

### Examples

```
# Right Triangle
draw_shape(rbind(
  c(10,1),
  c(10,10),
  c(1,1)
), cycle=TRUE)
```

erase

*Erase Text*

---

**Description**

Clear text from the cursor's row . Passing values "start" and "end" allow the user to erase specific portions of the row relative to the cursor.

**Usage**

```
erase(x = c("row", "start", "end"), ...)
```

**Arguments**

x	character describing location to clear. The default, "row", clears the entire row; "start" clears all text from the beginning of the row until the cursor's position; "end" clears all text from the cursor's position until the end of the row.
...	objects passed to/from methods

**Value**

NULL

**Examples**

```
cat('hello world!')  
erase('row')
```

---

example\_luckynumber*Example Program From Vignette*

---

**Description**

Simple program that asks for a letter and a number and returns another value to screen.

**Usage**

```
example_luckynumber()
```

**Value**

NULL

---

fg_off	<i>Turn Off Foreground Color</i>
--------	----------------------------------

---

**Description**

Return future terminal text to the default color. Foreground color is turned on with [fg\\_on](#).

**Usage**

```
fg_off()
```

**Value**

NULL

**See Also**

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

**Examples**

```
# Different methods of specifying red
fg_on("red")
fg_on("#FF0000")
fg_on(1)
fg_on(255, 0, 0)

# Turn off color
fg_off()
```

---

fg_on	<i>Turn On Foreground Color</i>
-------	---------------------------------

---

**Description**

Specifies the color of all future text written in the terminal [fg\\_on](#) accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color. Not all terminals support each possible color.

**Usage**

```
fg_on(...)
```

**Arguments**

... character or numeric value

**Details**

Foreground color is turned off with `fg_off`.

**Value**

NULL

**See Also**

Other style functions: `attr_off()`, `attr_on()`, `bg_off()`, `bg_on()`, `color_off()`, `color_pair()`, `fg_off()`, `make_bg()`, `make_fg()`, `make_style()`, `reset()`, `style()`

**Examples**

```
# Different methods of specifying red
fg_on("red")
fg_on("#FF0000")
fg_on(9)
fg_on(255, 0, 0)

# Turn off color
fg_off()
```

---

fill\_circle

*Draw a Filled-In Circle*

---

**Description**

Calculate the path of a circle in a grid and draw it to screen.

**Usage**

```
fill_circle(yx, r = 1, n = 50, text = "x", ...)
```

**Arguments**

<code>yx</code>	center (row, col) coordinate
<code>r</code>	radius of the circle in grid points
<code>n</code>	number of points along curve to calculate
<code>text</code>	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color <code>fg</code> , background color <code>bg</code> , and attribute <code>attr</code>

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_circle(yx=c(10,10), r=5)
```

---

fill\_ellipse

*Draw a Filled-In Ellipse*


---

**Description**

Calculate the path of an ellipse within a grid and draw to screen.

**Usage**

```
fill_ellipse(yx = c(0, 0), rx = 1, ry = 1, n = 50, text = "x", ...)
```

**Arguments**

yx	(row, col) coordinate of the center of the ellipse
rx	radius along the x-axis in grid points
ry	radius along the y-axis in grid points
n	number of points along curve to calculate
text	character value drawn at coordinate
...	parameters that are passed to <a href="#">style()</a> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_ellipse(yx=c(10,10), rx=8, ry = 4)
```

---

fill\_rect

*Draw a Filled-In Rectangle*


---

**Description**

Calculate the path of a rectangle in a grid and draw to screen.

**Usage**

```
fill_rect(yx1, yx2, text = "x", ...)
```

**Arguments**

yx1	upper-left (row, col) coordinate
yx2	lower-right (row, col) coordinate
text	character value drawn at coordinate
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
draw_rect(c(5,5), c(9,9))
```

---

`fill_shape`*Draw a Filled-In Shape*

---

**Description**

Calculate the path of a shape given supplied vertices and draw to screen.

**Usage**

```
fill_shape(mat, cycle = TRUE, n = 30, text = "x", ...)
```

**Arguments**

<code>mat</code>	an Nx2 matrix of (row, col) coordinates
<code>cycle</code>	logical value determining whether to the first and last coordinates
<code>n</code>	number of points along each edge to calculate
<code>text</code>	character value drawn at coordinate
<code>...</code>	parameters that are passed to <code>style()</code> , including the foreground color <code>fg</code> , background color <code>bg</code> , and attribute <code>attr</code>

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
# Right Triangle
fill_shape(rbind(
  c(10,1),
  c(10,10),
  c(1,1)
), cycle=TRUE)
```

getkp

*Get Keypress*

---

**Description**

Listen for a keypress, then apply keypress to a function or echo it to the terminal screen. The user must be in a terminal to use `getkp`; it will not work in RStudio or the R GUI. All actions within R are halted until the keypress is returned.

**Usage**

```
getkp(fn = list(), echo = FALSE)
```

**Arguments**

<code>fn</code>	list of named functions
<code>echo</code>	whether the keypress should be echoed to the screen if not found in list

**Value**

character naming the key that was pressed (*invisibly*).

**Examples**

```
f <- list(
  'up'    = function(){mv(row=-1)},
  'down'  = function(){mv(row=-1)},
  'left'  = function(){mv(col=-1)},
  'right' = function(){mv(col=1)}
)
## Not run:
getkp(fn=f, echo=FALSE)

## End(Not run)
```

---

getkpl*Loop a Keypress*

---

**Description**

Maintain a loop that listens for a keypress, then applies the keypress to a function or echoes it to the terminal screen. The user must be in a terminal to use `getkp`; it will not work in RStudio or the R GUI. All actions within R are halted until the keypress is returned.

**Usage**

```
getkpl(escape = "escape", fn = list(), echo = FALSE)
```

**Arguments**

escape	vector of character keypresses that escape the loop. The default is "escape" key.
fn	list of named functions
echo	whether the keypress should be echoed to the screen if not found in list

**Value**

NULL

**Examples**

```
f <- list(
  'up'    = function(){mv(row=-1)},
  'down'  = function(){mv(row=1)},
  'left'  = function(){mv(col=-1)},
  'right' = function(){mv(col=1)}
)
## Not run:
getkpl(escape = c("escape", "enter"), fn=f, echo=FALSE)

## End(Not run)
```

---

grid\_at

*Draw a Character Grid Matrix*

---

**Description**

Constructs a grid with given dimension, character values, and step parameter, and prints it to screen

**Usage**

```
grid_at(
  yx = c(1, 1),
  dim = NULL,
  step = c(2, 2),
  text = c(".", ".", "+", "|", "|", "-", "-", rep("+", 8)),
  border = TRUE
)
```

**Arguments**

<code>yx</code>	(row, column) on screen or window where the upper-left corner of the grid is to be printed
<code>dim</code>	(row, column) vector for size of grid.
<code>step</code>	numeric vector describing grid step across (rows, columns)
<code>text</code>	character vector of values for the grid, in order: horizontal grid line, vertical grid line, grid intersection, left border, right border, top border, bottom border, corners (upper-left, upper-right, lower-left, lower-right), ticks (right, bottom, left, top)
<code>border</code>	logical value for whether a border should be included.

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
grid_at(yx=c(2,2), dim=c(11,13), step=c(2,4), border=TRUE)
```

---

grid\_mat

*Create a Character Grid Matrix*

---

**Description**

Constructs a grid with provided dimensions (row, col), character values for gridlines, and a step parameter noting the number of rows and columns between each gridline.

**Usage**

```
grid_mat(
  dim,
  step = c(2, 2),
  text = c(".", ".", "+", "|", "|", "-", "-", rep("+", 8)),
  border = TRUE
)
```

**Arguments**

dim	(row, column) vector for size of grid.
step	numeric vector describing grid step across (rows, columns)
text	character vector of values for the grid, in order: horizontal grid line, vertical grid line, grid intersection, left border, right border, top border, bottom border, corners (upper-left, upper-right, lower-left, lower-right), ticks (right, bottom, left, top)
border	logical value for whether a border should be included.

**Value**

rowxcol matrix

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
grid_mat(dim=c(11,13), step=c(2,4), border=TRUE)
```

---

hide\_cursor

*Hide Cursor*

---

**Description**

Make the cursor invisible. The cursor can be revealed with [show\\_cursor](#)

**Usage**

```
hide_cursor()
```

**Value**

NULL

**Examples**

```
hide_cursor()
cat("\n\nHello World!")
show_cursor()
```

---

hline	<i>Horizontal Line</i>
-------	------------------------

---

**Description**

Horizontal Line

**Usage**

```
hline(n, text = "-")
```

**Arguments**

n	integer describing the character length of the line
text	character to be repeated

**Value**

character string of length n

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
hline(10, "*") # *****  
hline(5, "$") # $$$$$
```

---

hline_at	<i>Draw Horizontal Line</i>
----------	-----------------------------

---

**Description**

Draws a horizontal line of length n at (row, col)

**Usage**

```
hline_at(yx, n, text = "-", ...)
```

**Arguments**

<code>yx</code>	(row, col) coordinates where line should be drawn.
<code>n</code>	integer describing the character length of the line
<code>text</code>	character to be repeated
<code>...</code>	parameters that are passed to <code>style()</code> , including the foreground color <code>fg</code> , background color <code>bg</code> , and attribute <code>attr</code>

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#), [vline\(\)](#)

**Examples**

```
hline_at(c(3,4),6,"-") # print "-----" at (3,4)
```

---

in.term

*Determine whether in Terminal*


---

**Description**

Tests whether the session is in terminal and returns TRUE or FALSE. Many of the `cursor` functions require being in terminal and will not work with RStudio or the R GUI application.

**Usage**

```
in.term()
```

**Value**

logical value; TRUE or FALSE

**Examples**

```
in.term()
```

---

load_cursor	<i>Load Cursor</i>
-------------	--------------------

---

**Description**

Restore cursor to its previously saved location from [save\\_cursor](#).

**Usage**

```
load_cursor()
```

**Value**

NULL

**Examples**

```
save_cursor()
cat("\n\nHello World!")
load_cursor()
```

---

make_bg	<i>Create Background Color</i>
---------	--------------------------------

---

**Description**

Returns the ANSI code for the specified background color. `make_bg` accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color.

**Usage**

```
make_bg(...)
```

**Arguments**

... character or numeric value

**Value**

ANSI character string

**See Also**

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

## Examples

```
# Different methods of specifying cyan
make_bg("cyan")
make_bg("#00FFFF")
make_bg(14)
make_bg(0, 255, 255)
```

---

make_fg	<i>Create Foreground Color</i>
---------	--------------------------------

---

## Description

Returns the ANSI code for the specified foreground color. `make_fg` accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color.

## Usage

```
make_fg(...)
```

## Arguments

```
...          character or numeric value
```

## Value

ANSI character string

## See Also

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#), [style\(\)](#)

## Examples

```
# Different methods of specifying red
make_fg("red")
make_fg("#FF0000")
make_fg(9)
make_fg(255, 0, 0)
```

---

make_style	<i>Create Color &amp; Attribute Style</i>
------------	---

---

**Description**

Returns the ANSI codes for the specified colors and text attributes.

**Usage**

```
make_style(fg = NA, bg = NA, attr = NA)
```

**Arguments**

fg	character or numeric value for the foreground color. See <a href="#">fg_on</a> for more details.
bg	character or numeric value for the background color. See <a href="#">bg_on</a> for more details.
attr	character vector describing attributes to turn on. See <a href="#">attr_on</a> for more details.

**Value**

ANSI character string

**See Also**

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [reset\(\)](#), [style\(\)](#)

**Examples**

```
cat(make_style(fg="blue", bg=c(192,192,192), attr=c("ul", "st")))
cat("Hello World!\n")
reset()
```

---

mv	<i>Move Cursor</i>
----	--------------------

---

**Description**

Move cursor relative to its current position on the screen. Screen coordinates are given by (row, column) with the position of the screen being (1, 1).

**Usage**

```
mv(row = 0L, col = 0L)
```

**Arguments**

row	number of rows in which to move the cursor. Positive values move the cursor down; negative values move the cursor up. If row has two or more values, the second value replaces col.
col	number of columns in which to move the cursor. Positive values move the cursor forward; negative values move the cursor backwards.

**Details**

The user must be in a terminal to use the functionality; it will not work in RStudio or the R GUI.

**Value**

NULL

**See Also**

[mv\\_to](#) to move to a specific location on the screen.

Other moving functions: [mv\\_col\(\)](#), [mv\\_row\(\)](#), [mv\\_to\(\)](#)

**Examples**

```
# move the cursor down one and forward two
mv(1, 2)

# Alternatively, you can specify the coordinates as a single vector.
loc <- c(1, 2)
mv(loc)

# to move to the left one unit (only works if the current column is > 1)
mv(, -1)
```

---

mv\_col

*Move Cursor to Column*

---

**Description**

Move the cursor to the specified column, while maintaining the same row.

**Usage**

```
mv_col(n = 1L)
```

**Arguments**

n positive integer specifying the column

**Details**

The user must be in a terminal to use the functionality; it will not work in RStudio or the R GUI.

**Value**

NULL

**See Also**

Other moving functions: [mv\\_row\(\)](#), [mv\\_to\(\)](#), [mv\(\)](#)

**Examples**

```
# Position cursor at the beginning of the row
mv_col(1)

# Move cursor to the 10th column in the row
mv_col(10)
```

---

mv\_row

*Move Cursor to Row*

---

**Description**

Moves cursor to the beginning of the row relative to its current location.

**Usage**

```
mv_row(n = 1L)
```

**Arguments**

n                    number of rows to move. Positive values indicate the next rows; negative values indicate the previous rows

**Details**

The user must be in a terminal to use the functionality; it will not work in RStudio or the R GUI.

**Value**

NULL

**See Also**

Other moving functions: [mv\\_col\(\)](#), [mv\\_to\(\)](#), [mv\(\)](#)

## Examples

```
# move the cursor to the beginning of the previous line
mv_row(-1)
```

---

mv_to	<i>Move Cursor to Specified Location</i>
-------	--

---

## Description

Move cursor relative to its current position on the screen. Screen coordinates are given by (row, column) with the position of the screen being (1, 1).

## Usage

```
mv_to(row = 1L, col = 1L)
```

## Arguments

row	positive integer specifying the console row. If row has two or more values, the second value replaces col.
col	positive integer specifying the console column.

## Details

The user must be in a terminal to use the functionality; it will not work in RStudio or the R GUI.

## Value

NULL

## See Also

[mv](#) to move relative to the current location on the screen.

Other moving functions: [mv\\_col\(\)](#), [mv\\_row\(\)](#), [mv\(\)](#)

## Examples

```
# move the cursor to the 2nd row, 4th column
mv_to(2, 4)

# alternatively, you can specify the coordinates as a vector.
loc <- c(2, 4)
mv_to(loc)
```

---

path_arc	<i>Arc Path</i>
----------	-----------------

---

**Description**

Calculate the path of an arc within a grid.

**Usage**

```
path_arc(yx, start, end, r = 1, n = 50)
```

**Arguments**

yx	center (row, col) coordinate of circle
start	starting angle in radians
end	ending angle in radians
r	radius of circle
n	number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_arc(yx=c(10,10), start=pi/2, end=pi, r=6)
```

---

path_bezier	<i>Bezier Curve Path</i>
-------------	--------------------------

---

**Description**

Calculate the path of a Bezier Curve with up to two control points in a grid.

**Usage**

```
path_bezier(start, end, c1, c2 = NULL, n = 50)
```

**Arguments**

start	starting (row, col) coordinate
end	ending (row, col) coordinate
c1	coordinate of first control point
c2	coordinate of second control point
n	number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_bezier(start=c(10,1), end=c(10,10), c1=c(1, 3))
```

---

path_circle	<i>Path of a Circle</i>
-------------	-------------------------

---

**Description**

Calculate the path of a circle in a grid.

**Usage**

```
path_circle(yx, r = 1, n = 50)
```

**Arguments**

yx	center (row, col) coordinate
r	radius of the circle in grid points
n	number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_circle(yx=c(10,10), r=5)
```

---

path\_ellipse

*Ellipse Path*

---

**Description**

Calculate the path of an ellipse within a grid.

**Usage**

```
path_ellipse(yx = c(0, 0), rx = 1, ry = 1, n = 50)
```

**Arguments**

yx	(row, col) coordinate of the center of the ellipse
rx	radius along the x-axis in grid points
ry	radius along the y-axis in grid points
n	number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_ellipse(yx=c(10,10), rx=8, ry = 4)
```

---

path_fill	<i>Fill In Path</i>
-----------	---------------------

---

**Description**

Calculate the coordinates of all points inside of a path.

**Usage**

```
path_fill(mat)
```

**Arguments**

mat	Nx2 matrix of (row, column) path coordinates
-----	--

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
c0 <- path_circle(c(10,10), r=5)
path_fill(c0)
```

---

path_fn	<i>Function Path</i>
---------	----------------------

---

**Description**

Calculate the path within a grid of an user-supplied function.

**Usage**

```
path_fn(x1, x2, fn, n = 50)
```

**Arguments**

x1	starting column value of the path
x2	ending column value of the path
fn	function returning row value for a column input
n	number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_fn(x1=1, x2=10,  
        function(x){sqrt(x)}  
)
```

---

path_intersection	<i>Intersection between Two Paths</i>
-------------------	---------------------------------------

---

**Description**

Calculate the points of intersection between two paths.

**Usage**

```
path_intersection(path)
```

**Arguments**

path            list containing two coordinate (row, column) matrices.

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
c1 <- path_circle(c(4,4), r=3)  
c2 <- path_circle(c(6,6), r=3)  
path_intersection(list(c1, c2))
```

---

path_lerp	<i>Linear Interpolation Path</i>
-----------	----------------------------------

---

**Description**

Interpolate between two points in a grid.

**Usage**

```
path_lerp(start, end, n = 50)
```

**Arguments**

start	starting (row, col) coordinate
end	ending (row, col) coordinate
n	number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_lerp(start=c(10,1), end=c(1,3))
```

---

path_ray	<i>Ray Path</i>
----------	-----------------

---

**Description**

Calculate the path of a ray extending

**Usage**

```
path_ray(start, end, lim = c(64, 128), n = 200)
```

**Arguments**

start	start (row, col) coordinate of the ray
end	either an ending coording, an angle in radians, or a character direction (u, d, l, r, ul, ur, dl, dr)
lim	bounding box dimensions used to calculate ray
n	number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_rect\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_ray(start=c(10,10), end=pi/6)
path_ray(start=c(10,10), end=pi/6, lim=c(15,15))
path_ray(start=c(10,10), end=c(4,2))
```

---

path\_rect

*Rectangle Path*

---

**Description**

Calculate the path of a rectangle in a grid.

**Usage**

```
path_rect(yx1, yx2)
```

**Arguments**

yx1	upper-left (row, col) coordinate
yx2	lower-right (row, col) coordinate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_shape\(\)](#)

**Examples**

```
path_rect(c(5,5), c(9,9))
```

---

path_shape	<i>Path along a Shape</i>
------------	---------------------------

---

**Description**

Calculate the path of a shape given supplied vertices.

**Usage**

```
path_shape(mat, cycle = TRUE, n = 30)
```

**Arguments**

mat	an Nx2 matrix of (row, col) coordinates
cycle	logical value determining whether to the first and last coordinates
n	number of points along each edge to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: [path\\_arc\(\)](#), [path\\_bezier\(\)](#), [path\\_circle\(\)](#), [path\\_ellipse\(\)](#), [path\\_fill\(\)](#), [path\\_fn\(\)](#), [path\\_intersection\(\)](#), [path\\_lerp\(\)](#), [path\\_ray\(\)](#), [path\\_rect\(\)](#)

**Examples**

```
# Right Triangle
path_shape(rbind(
  c(10,1),
  c(10,10),
  c(1,1)
), cycle=TRUE)
```

---

`repch`*Repeat a Character*

---

**Description**

Repeat a character n times and concatenate into a single value.

**Usage**

```
repch(x, n)
```

**Arguments**

x	character to be repeated
n	number of times to be repeated

**Value**

character vector

**Examples**

```
repch("abc", 5)
```

---

`reset`*Reset Console Style*

---

**Description**

Turns off all text attributes and colors in the terminal.

**Usage**

```
reset()
```

**Value**

NULL

**See Also**

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [style\(\)](#)

**Examples**

```
attr_on("ul")
fg_on("red")
bg_on(c(10,60,205))
cat("Hello World!\n")

reset()
cat("Hello World!\n")
```

---

save_cursor	<i>Save Cursor Position</i>
-------------	-----------------------------

---

**Description**

Save the position of the cursor so that the position can be restored for later with [load\\_cursor](#).

**Usage**

```
save_cursor()
```

**Value**

NULL

**Examples**

```
save_cursor()
cat("\n\nHello World!")
load_cursor()
```

---

show_cursor	<i>Show Cursor</i>
-------------	--------------------

---

**Description**

Reveal the cursor after it has been hidden by [hide\\_cursor](#).

**Usage**

```
show_cursor()
```

**Value**

NULL

## Examples

```
hide_cursor()
cat("\n\nHello World!")
show_cursor()
```

---

style

*Add Color & Attributes to a Character*

---

## Description

Add color and other text attributes to a character vector. Attributes can be seen after text is passed to `cat`, though it may only show up in a terminal. Note that terminal attributes and colors are automatically reset to default after text is printed.

## Usage

```
style(x, fg = NA, bg = NA, attr = NA)
```

## Arguments

<code>x</code>	character vector to be styled
<code>fg</code>	character or numeric value for the foreground color. See <a href="#">fg_on</a> for more details.
<code>bg</code>	character or numeric value for the background color. See <a href="#">bg_on</a> for more details.
<code>attr</code>	character vector describing attributes to turn on. See <a href="#">attr_on</a> for more details.

## Value

character vector

## See Also

Other style functions: [attr\\_off\(\)](#), [attr\\_on\(\)](#), [bg\\_off\(\)](#), [bg\\_on\(\)](#), [color\\_off\(\)](#), [color\\_pair\(\)](#), [fg\\_off\(\)](#), [fg\\_on\(\)](#), [make\\_bg\(\)](#), [make\\_fg\(\)](#), [make\\_style\(\)](#), [reset\(\)](#)

## Examples

```
x <- style("Hello World!\n", fg="blue", bg=c(192,192,192), attr=c("ul", "st"))
cat(paste(x, "It is nice to meet you!"))
```

---

Sym	<i>Unicode Symbols</i>
-----	------------------------

---

**Description**

A named list containing the unicode character for various box drawing, mathematical, currency, astrological, and other symbols.

**Usage**

Sym

**Format**

A named list of characters

---

term_dim	<i>Determine Terminal Size</i>
----------	--------------------------------

---

**Description**

Function determines the size of the terminal in number of rows and columns. The value may not be accurate in RStudio or the R GUI.

**Usage**

```
term_dim()
```

**Value**

numeric vector (# of rows, # of columns)

**Examples**

```
term_dim()
```

---

vline	<i>Vertical Line</i>
-------	----------------------

---

**Description**

Vertical Line

**Usage**

```
vline(n, text = "|")
```

**Arguments**

n	integer describing the character length of the line
text	character to be repeated

**Value**

character string of length n, separated by "\n"

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\\_at\(\)](#)

**Examples**

```
vline(4, "*") # "*\n*\n*\n*"
```

---

vline_at	<i>Draw Vertical Line</i>
----------	---------------------------

---

**Description**

Draws a vertical line of length n at (row, col)

**Usage**

```
vline_at(yx, n, text = "|", ...)
```

**Arguments**

<code>yx</code>	(row, col) coordinates where top of the line should be drawn.
<code>n</code>	integer describing the character length of the line
<code>text</code>	character to be repeated
<code>...</code>	parameters that are passed to <code>style()</code> , including the foreground color <code>fg</code> , background color <code>bg</code> , and attribute <code>attr</code>

**Value**

NULL

**See Also**

Other drawing functions: [box\\_at\(\)](#), [draw\\_arc\(\)](#), [draw\\_bezier\(\)](#), [draw\\_circle\(\)](#), [draw\\_ellipse\(\)](#), [draw\\_fn\(\)](#), [draw\\_lerp\(\)](#), [draw\\_path\(\)](#), [draw\\_ray\(\)](#), [draw\\_rect\(\)](#), [draw\\_shape\(\)](#), [fill\\_circle\(\)](#), [fill\\_ellipse\(\)](#), [fill\\_rect\(\)](#), [fill\\_shape\(\)](#), [grid\\_at\(\)](#), [grid\\_mat\(\)](#), [hline\\_at\(\)](#), [hline\(\)](#), [vline\(\)](#)

**Examples**

```
vline_at(c(3,4),6,"|") # print "|" at (3,4), ..., (8,4)
```

---

`wr` *Write String to Terminal*

---

**Description**

Writes a string of characters to the terminal at the current cursor position. `wr` accepts text colors and attributes, but these are reset to default afterwards if used.

**Usage**

```
wr(text, fg = NA, bg = NA, attr = NA)
```

**Arguments**

<code>text</code>	string to be printed to the Console
<code>fg</code>	foreground color. See <a href="#">fg_on</a> for more details.
<code>bg</code>	background color. See <a href="#">bg_on</a> for more details.
<code>attr</code>	character attribute. See <a href="#">attr_on</a> for more details.

**Value**

NULL

**See Also**

Other writing functions: [wrat\(\)](#), [wrch\(\)](#), [wrkpl\(\)](#), [wrkp\(\)](#)

**Examples**

```
mv_to(5,4)
wrch("h")
wrch("e", fg="red")
wr("llo World")
```

---

wrapup

*Return Screen to Blank State*


---

**Description**

Function to be used at the end of a terminal function. It resets the colors and attributes to their default values, clears the screen, and reveals the cursor.

**Usage**

```
wrapup()
```

**Value**

NULL

---

wrat

*Write At a Specific Location*


---

**Description**

Move cursor to specified location in the terminal screen, then print the supplied text. This function will only work in terminal, not the RStudio Console or R GUI.

**Usage**

```
wrat(yx, text, ...)
```

**Arguments**

<code>yx</code>	numeric vector specifying the (row, col) coordinates to print at
<code>text</code>	text to be written at <code>yx</code>
<code>...</code>	colors and attributes added to text. See <a href="#">wr</a> , <a href="#">fg_on</a> , <a href="#">bg_on</a> , and <a href="#">attr_on</a> for more details.

**Details**

The coordinates are given in matrix notation: (row, column), with the top-left corner of the screen being (1,1).

**Value**

NULL

**See Also**

Other writing functions: [wrch\(\)](#), [wrkpl\(\)](#), [wrkp\(\)](#), [wr\(\)](#)

**Examples**

```
wrat(c(10,6), "CURSR")
wrat(c(4,1), "Hello World!", fg="red", attr=c("bf", "ul"))

mat <- rbind(c(5,2), c(10,5), c(1,19))
wrat(mat, "HI", fg="yellow")
```

---

wrch

*Write Character to Terminal*

---

**Description**

Writes a single character to the terminal at the current cursor position. `wr` accepts text colors and attributes, but these are reset to default afterwards if used.

**Usage**

```
wrch(chr, fg = NA, bg = NA, attr = NA)
```

**Arguments**

<code>chr</code>	character to be printed to the Console
<code>fg</code>	foreground color. See <a href="#">fg_on</a> for more details.
<code>bg</code>	background color. See <a href="#">bg_on</a> for more details.
<code>attr</code>	character attribute. See <a href="#">attr_on</a> for more details.

**Value**

NULL

**See Also**

Other writing functions: [wrat\(\)](#), [wrkpl\(\)](#), [wrkp\(\)](#), [wr\(\)](#)

### Examples

```
mv_to(5,4)
wrch("h")
wrch("e", fg="red")
wr("llo World")
```

---

wrchat

*Write Character to Terminal at Specified Location*

---

### Description

Move cursor to specified location in the terminal screen, then print the supplied character. This function will only work in terminal, not the RStudio Console or R GUI.

### Usage

```
wrchat(row, col, chr, ...)
```

### Arguments

row	row in which character is printed. If length of row is greater than one, the second value replaces col.
col	column in which character is printed
chr	character to be printed to the Console
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

### Details

The coordinates are given in matrix notation: (row, column), with the top-left corner of the screen being (1,1).

### Value

NULL

### Examples

```
wrchat(5, 4, "h")
```

**Description**

Detect keypress and print it to the terminal screen, while invisibly returning the keypress. The user can specify which characters to ignore, and can also map keys to a list of functions. Any keypress mapped to a function will not be echoed to the screen.

**Usage**

```
wrkp(ignore = "escape", fn = list(), ...)
```

**Arguments**

ignore	vector of keypresses to ignore.
fn	list of functions, named by key, to be called when key is pressed.
...	parameters that are passed to <code>style()</code> , including the foreground color <code>fg</code> , background color <code>bg</code> , and attribute <code>attr</code>

**Value**

NULL

**See Also**

Other writing functions: [wrat\(\)](#), [wrch\(\)](#), [wrkpl\(\)](#), [wr\(\)](#)

**Examples**

```
## Not run:
wrkp(
  ignore="escape",
  fn = list(
    enter = function(){mv_row(1)},
    left = function(){mv(0, -1)},
    right = function(){mv(0, 1)},
    up = function(){mv(-1,0)},
    down = function(){mv(1,0)},
    space = function(){cat(" ")}
  )
)
## End(Not run)
```

---

 wrkpl

*Echo Keypress to Screen in a Loop*


---

## Description

Detect keypress and print it to the terminal screen, while invisibly returning the keypress. The user can specify which characters to ignore, and can also map keys to a list of functions. Any keypress mapped to a function will not be echoed to the screen.

## Usage

```
wrkpl(escape = c("escape"), ignore = NA_character_, fn = list(), ...)
```

## Arguments

escape	vector of keypresses to escape the loop.
ignore	vector of keypresses to ignore.
fn	list of functions, named by key, to be called when key is pressed.
...	parameters that are passed to <code>style()</code> , including the foreground color fg, background color bg, and attribute attr

## Value

NULL

## See Also

Other writing functions: [wrat\(\)](#), [wrch\(\)](#), [wrkp\(\)](#), [wr\(\)](#)

## Examples

```
## Not run:
wrkpl(
  escape = "escape",
  fn = list(
    enter = function(){mv_row(1)},
    left = function(){mv(0, -1)},
    right = function(){mv(0, 1)},
    up = function(){mv(-1,0)},
    down = function(){mv(1,0)},
    space = function(){cat(" ")}
  )
)
## End(Not run)
```

# Index

- \* **datasets**
    - Sym, 47
  - \* **drawing functions**
    - box\_at, 6
    - draw\_arc, 9
    - draw\_bezier, 10
    - draw\_circle, 11
    - draw\_ellipse, 12
    - draw\_fn, 13
    - draw\_lerp, 14
    - draw\_path, 14
    - draw\_ray, 15
    - draw\_rect, 16
    - draw\_shape, 17
    - fill\_circle, 20
    - fill\_ellipse, 21
    - fill\_rect, 22
    - fill\_shape, 23
    - grid\_at, 25
    - grid\_mat, 26
    - hline, 28
    - hline\_at, 28
    - vline, 48
    - vline\_at, 48
  - \* **moving functions**
    - mv, 32
    - mv\_col, 33
    - mv\_row, 34
    - mv\_to, 35
  - \* **path-fitting functions**
    - path\_arc, 36
    - path\_bezier, 36
    - path\_circle, 37
    - path\_ellipse, 38
    - path\_fill, 39
    - path\_fn, 39
    - path\_intersection, 40
    - path\_lerp, 41
    - path\_ray, 41
    - path\_rect, 42
    - path\_shape, 43
  - \* **style functions**
    - attr\_off, 3
    - attr\_on, 4
    - bg\_off, 5
    - bg\_on, 5
    - color\_off, 8
    - color\_pair, 9
    - fg\_off, 19
    - fg\_on, 19
    - make\_bg, 30
    - make\_fg, 31
    - make\_style, 32
    - reset, 44
    - style, 46
  - \* **writing functions**
    - wr, 49
    - wrat, 50
    - wrch, 51
    - wrkp, 53
    - wrkpl, 54
- attr\_off, 3, 4–6, 8, 9, 19, 20, 30–32, 44, 46
- attr\_on, 3, 4, 5–9, 19, 20, 30–32, 44, 46, 49–51
- bg\_off, 3, 4, 5, 6, 8, 9, 19, 20, 30–32, 44, 46
- bg\_on, 3–5, 5, 7–9, 19, 20, 30–32, 44, 46, 49–51
- box\_at, 6, 10–17, 21–23, 26–29, 48, 49
- clear, 7
- color\_off, 3–6, 8, 9, 19, 20, 30–32, 44, 46
- color\_pair, 3–6, 8, 9, 19, 20, 30–32, 44, 46
- draw\_arc, 7, 9, 11–17, 21–23, 26–29, 48, 49
- draw\_bezier, 7, 10, 10, 11–17, 21–23, 26–29, 48, 49
- draw\_circle, 7, 10, 11, 11, 12–17, 21–23, 26–29, 48, 49

- draw\_ellipse, [7](#), [10](#), [11](#), [12](#), [13–17](#), [21–23](#), [26–29](#), [48](#), [49](#)
- draw\_fn, [7](#), [10–12](#), [13](#), [14–17](#), [21–23](#), [26–29](#), [48](#), [49](#)
- draw\_lerp, [7](#), [10–13](#), [14](#), [15–17](#), [21–23](#), [26–29](#), [48](#), [49](#)
- draw\_path, [7](#), [10–14](#), [14](#), [16](#), [17](#), [21–23](#), [26–29](#), [48](#), [49](#)
- draw\_ray, [7](#), [10–15](#), [15](#), [16](#), [17](#), [21–23](#), [26–29](#), [48](#), [49](#)
- draw\_rect, [7](#), [10–16](#), [16](#), [17](#), [21–23](#), [26–29](#), [48](#), [49](#)
- draw\_shape, [7](#), [10–16](#), [17](#), [21–23](#), [26–29](#), [48](#), [49](#)
  
- erase, [18](#)
- example\_luckynumber, [18](#)
  
- fg\_off, [3–6](#), [8](#), [9](#), [19](#), [20](#), [30–32](#), [44](#), [46](#)
- fg\_on, [3–9](#), [19](#), [19](#), [30–32](#), [44](#), [46](#), [49–51](#)
- fill\_circle, [7](#), [10–17](#), [20](#), [21–23](#), [26–29](#), [48](#), [49](#)
- fill\_ellipse, [7](#), [10–17](#), [21](#), [21](#), [22](#), [23](#), [26–29](#), [48](#), [49](#)
- fill\_rect, [7](#), [10–17](#), [21](#), [22](#), [23](#), [26–29](#), [48](#), [49](#)
- fill\_shape, [7](#), [10–17](#), [21](#), [22](#), [23](#), [26–29](#), [48](#), [49](#)
  
- getkp, [24](#)
- getkpl, [24](#)
- grid\_at, [7](#), [10–17](#), [21–23](#), [25](#), [27–29](#), [48](#), [49](#)
- grid\_mat, [7](#), [10–17](#), [21–23](#), [26](#), [26](#), [28](#), [29](#), [48](#), [49](#)
  
- hide\_cursor, [27](#), [45](#)
- hline, [7](#), [10–17](#), [21–23](#), [26](#), [27](#), [28](#), [29](#), [48](#), [49](#)
- hline\_at, [7](#), [10–17](#), [21–23](#), [26–28](#), [28](#), [48](#), [49](#)
  
- in.term, [29](#)
  
- load\_cursor, [30](#), [45](#)
  
- make\_bg, [3–6](#), [8](#), [9](#), [19](#), [20](#), [30](#), [31](#), [32](#), [44](#), [46](#)
- make\_fg, [3–6](#), [8](#), [9](#), [19](#), [20](#), [30](#), [31](#), [32](#), [44](#), [46](#)
- make\_style, [3–6](#), [8](#), [9](#), [19](#), [20](#), [30](#), [31](#), [32](#), [44](#), [46](#)
- mv, [32](#), [34](#), [35](#)
- mv\_col, [33](#), [33](#), [34](#), [35](#)
- mv\_row, [33](#), [34](#), [34](#), [35](#)
- mv\_to, [33](#), [34](#), [35](#)
  
- path\_arc, [36](#), [37–43](#)
- path\_bezier, [36](#), [36](#), [37–43](#)
- path\_circle, [36](#), [37](#), [37](#), [38–43](#)
- path\_ellipse, [36](#), [37](#), [38](#), [39–43](#)
- path\_fill, [36–38](#), [39](#), [40–43](#)
- path\_fn, [36–39](#), [39](#), [40–43](#)
- path\_intersection, [36–40](#), [40](#), [41–43](#)
- path\_lerp, [36–40](#), [41](#), [42](#), [43](#)
- path\_ray, [36–41](#), [41](#), [42](#), [43](#)
- path\_rect, [36–42](#), [42](#), [43](#)
- path\_shape, [36–42](#), [43](#)
  
- repch, [44](#)
- reset, [3–6](#), [8](#), [9](#), [19](#), [20](#), [30–32](#), [44](#), [46](#)
  
- save\_cursor, [30](#), [45](#)
- show\_cursor, [27](#), [45](#)
- style, [3–6](#), [8](#), [9](#), [19](#), [20](#), [30–32](#), [44](#), [46](#)
- Sym, [47](#)
  
- term\_dim, [47](#)
  
- vline, [7](#), [10–17](#), [21–23](#), [26–29](#), [48](#), [49](#)
- vline\_at, [7](#), [10–17](#), [21–23](#), [26–29](#), [48](#), [48](#)
  
- wr, [49](#), [50](#), [51](#), [53](#), [54](#)
- wrapup, [50](#)
- wrat, [50](#), [50](#), [51](#), [53](#), [54](#)
- wrch, [50](#), [51](#), [51](#), [53](#), [54](#)
- wrchat, [52](#)
- wrkp, [50](#), [51](#), [53](#), [54](#)
- wrkpl, [50](#), [51](#), [53](#), [54](#)