Package: ggdiagram (via r-universe)

February 3, 2025

Title Object-Oriented Diagram Plots with ggplot2

Version 0.0.0.9000

Description The ggdiagram package creates path diagrams with an object-oriented approach and plots diagrams with ggplot2.

License CC0

URL https://github.com/wjschne/ggdiagram, https://wjschne.github.io/ggdiagram/

BugReports https://github.com/wjschne/ggdiagram/issues

Imports arrowheadr, bezier, dplyr, farver, geomtextpath, ggarrow, ggforce, ggplot2, ggtext, grDevices, grid, janitor, magick, magrittr, pdftools, purrr, rlang, S7, scales, signs, stringr, tibble, tidyr, tinter, tinytex, utils

Suggests knitr, lavaan, quarto, rmarkdown, simstandard, spelling, testthat (>= 3.0.0), tidyverse, viridis

VignetteBuilder knitr, quarto

Config/Needs/website quarto

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Collate 'ggdiagram-package.R' 'utils-pipe.R' 'a_early.R' 'str.R' 'colors.R' 'angles.R' 'style.R' 'points.R' 'labels.R' 'lines.R' 'segments.R' 'paths.R' 'circles.R' 'ellipses.R' 'arcs.R' 'bezier.R' 'rectangles.R' 'polygons.R' 'equations.R' 'distances.R' 'intersections.R' 'inside.R' 'rotate.R' 'zzz.R'

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libfribidi-dev libharfbuzz-dev libmagick++-dev gsfonts libicu-dev libjpeg-dev libpng-dev libxml2-dev libssl-dev libpoppler-cpp-dev

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

RemoteUrl https://github.com/wjschne/ggdiagram

RemoteRef HEAD

RemoteSha 4617b8d4b730026912ac0af5bcc2dbaf31c3a6af

Contents

`as.geom` *as.geom function*

Description

Converts a ggdiagram shape to a ggplot2 geom

Usage

```
as.geom(x, ...)
```

Arguments

`x`	a shape
`...`	<dynamic-dots> Pass arguments to ggplot2::geom_point

Details

Usually the `as.geom` function is not necessary to call explicitly because it is called whenever a ggdiagram shape is added to a ggplot. However, in complex situations (e.g., making a function that assembles many objects), it is sometimes necessary to make the call explicitly.

Value

geom

Examples

```
library(ggplot2)
c1 <- ob_circle(radius = 3)
ggplot() +
  as.geom(c1, fill = "black") +
  coord_equal()
```

bind *bind method*

Description

bind method

Usage

```
bind(x, ...)
```

Arguments

x	list of objects to bind
...	<dynamic-dots> properties passed to style

Value

a bound object of same class as x (or list of objects if x contains objects of different types)

Examples

```
bind(c(ob_point(1,2), ob_point(3,4)))
bind(c(ob_circle(ob_point(0,0), radius = 1),
       ob_circle(ob_point(1,1), radius = 2)))
```

circle_from_3_points *Get a circle from 3 points*

Description

Get a circle from 3 points

Usage

```
circle_from_3_points(p1, p2 = NULL, p3 = NULL, ...)
```

Arguments

p1	an ob_point of length 1 or length 3
p2	an ob_point of length 1 or NULL
p3	an ob_point of length 1 or NULL

Value

ob_point object

Examples

```
p1 <- ob_point(1,1)
p2 <- ob_point(2,4)
p3 <- ob_point(5,3)
circle_from_3_points(p1,p2, p3)
```

`class_color` *color class*

Description

color class

Usage

```
class_color(
  color = character(0),
  hue = NULL,
  saturation = NULL,
  brightness = NULL,
  alpha = NULL
)
```

Arguments

`color`	character (R color or hex code)
`hue`	get or set the hue of a color (i.e., the h in the hsv model)
`saturation`	get or set the saturation of a color (i.e., the s in the hsv model)
`brightness`	get or set the brightness of a color (i.e., the v in the hsv model)
`alpha`	get or set the transparency of a color

Value

class_color object

Slots

`transparentize` function to return the color with a new transparency (i.e., alpha)

`lighten` function to return a lighter color

`darken` function to return a darker color

Examples

```
mycolor <- class_color("blue")
mycolor
# Display html hexcode
c(mycolor)
# Set transparency
mycolor@transparentize(.5)
# Lighten color
mycolor@lighten(.5)
# Darken color
mycolor@darken(.5)
```

---

connect *Arrow connect one shape to another*

Description

Arrow connect one shape to another

Usage

```
connect(x, y, ...)
```

Arguments

x	first shape object
y	second shape object
...	<dynamic-dots> Arguments passed to style

Value

ob_segment

---

distance *Calculate distance between 2 points*

Description

Calculate distance between 2 points

Usage

```
distance(x, y, ...)
```

Arguments

`x`	a point, line, segment, or circle object
`y`	a point, line, or circle object
`...`	<dynamic-dots> Not used

Value

numeric

Examples

```
# Distance between two objects
p1 <- ob_point(0, 0)
p2 <- ob_point(3, 4)
distance(p1, p2)

# Distance between the endpoints of a segment
s1 <- ob_segment(p1, p2)
distance(s1)

# Distance between a point and a line
l1 <- ob_line(slope = 0, intercept = 1)
distance(p1, l1)

# Shortest distance between the edges of 2 circles
c1 <- ob_circle(p1, radius = 1)
c2 <- ob_circle(p2, radius = 2)
distance(c1, c2)
```

`equation` *equation*

Description

Get equation for object

Usage

```
equation(x, type = c("y", "general", "parametric"), digits = 2)
```

Arguments

`x`	object
`type`	equation type. Can be `y`, `general`, or `parametric`
`digits`	rounding digits

Value

string

get_tibble *Get object data with styles in a tibble*

Description

Get object data with styles in a tibble

Get object data in a tibble, filling in any missing styles with defaults

Usage

```
get_tibble(x)

get_tibble_defaults(x)
```

Arguments

x	object

Value

a tibble

tibble

ggdiagram *ggdiagram function*

Description

This is a convenient way to specify geom defaults

Usage

```
ggdiagram(
  font_family = "sans",
  font_size = 11,
  linewidth = 0.5,
  point_size = 1.5,
  rect_linewidth = linewidth,
  theme_function = ggplot2::theme_void,
  ...
)
```

Arguments

`font_family`	font family
`font_size`	font size in points
`linewidth`	line width
`point_size`	point size
`rect_linewidth`	line with of rectangles
`theme_function`	ggplot2 theme
`...`	<dynamic-dots> Arguments sent to ggplot2::theme

Value

ggplot function

Examples

```
ggdiagram() + ob_circle()
```

`inside` *is an ob_point inside a shape ?*

Description

is an ob_point inside a shape ?

Usage

```
inside(x, y)
```

Arguments

`x`	object
`y`	object

Value

logical

intersection *intersection of 2 objects (e.g., lines)*

Description

intersection of 2 objects (e.g., lines)

Usage

```
intersection(x, y, ...)
```

Arguments

x	object
y	object
...	<dynamic-dots> properties passed to style

Value

shape object

intersection_angle *Compute the angle of the intersection of two objects*

Description

Compute the angle of the intersection of two objects

Usage

```
intersection_angle(x, y)
```

Arguments

x	an object (point, segment, line)
y	an object (point, segment, line)

Value

ob_angle object

label_object *Automatic label for objects*

Description

Automatic label for objects

Usage

```
label_object(object, ...)
```

Arguments

object	object
...	<dynamic-dots> additional arguments

Value

string

latex_color *Surround TeX expression with a color command*

Description

Surround TeX expression with a color command

Usage

```
latex_color(x, color)
```

Arguments

x	TeX expression
color	color

Value

string

Examples

```
latex_color("X^2", "red")
```

`map_ob` *map_ob*

Description

A wrapper for purrr::map. It takes a ggdiagram object with multiple elements, applies a function to each element within the object, and returns a ggdiagram object

Usage

```
map_ob(.x, .f, ..., .progress = FALSE)
```

Arguments

`.x`	a ggdiagram object
`.f`	a function that returns a ggdiagram object
`...`	<dynamic-dots> arguments passed to .f
`.progress`	display progress if TRUE

Value

a ggdiagram object

`mean_color` *Average across colors*

Description

Average across colors

Usage

```
mean_color(x)
```

Arguments

`x`	color

Value

string

Examples

```
mean_color(c("red", "violet"))
```

midpoint *Get one or more points at positions from 0 to 1*

Description

It is possible to get more than one midpoint by specifying a position vector with a length greater than 1. Position values outside 0 and 1 will usually work, but will be outside the object.

Usage

```
midpoint(x, y, position = 0.5, ...)
```

Arguments

`x`	object
`y`	object (can be omitted for segments and arcs)
`position`	numeric vector. 0 is start, 1 is end. Defaults to .5
`...`	<dynamic-dots> properties passed to style

Value

ob_point

nudge *Move an object*

Description

Move an object

Usage

```
nudge(object, x, y, ...)
```

Arguments

`object`	object
`x`	nudge right and left
`y`	nudge up and down
`...`	<dynamic-dots> properties passed to style

Value

object of same class as `object`

Examples

```
ob_circle() |> nudge(x = 2)
# Alternative to nudge:
ob_circle() + ob_point(2, 0)
```

`ob_angle` *ob_angle*

Description

Creates an angle in the metric of radians, degrees, and turns.

Usage

```
ob_angle(
  .data = numeric(0),
  degree = integer(0),
  radian = integer(0),
  turn = integer(0)
)

degree(degree = numeric(0))

radian(radian = numeric(0))

turn(turn = numeric(0))
```

Arguments

`.data`	a real number indicating the number of turns.
`degree`	degrees
`radian`	radians
`turn`	proportion of full turns of a circle (1 turn = 2 * pi radians)
`positive`	if angle is negative, adds a full turn to ensure the angle is positive
`negative`	if angle is positive, subtracts a full turn to ensure the angle is negative

Details

Angles turns can be any real number, but degrees are displayed as values between -360 and +360, and radians are between -2pi and +2pi.

Value

ob_angle

Examples

```
# Three Different ways to make a right angle
## 90 degrees
degree(90)

## half pi radians
radian(.5 * pi)

## A quarter turn
turn(.25)

# Operations
degree(30) + degree(20)
degree(350) + degree(20)
degree(30) - degree(30)
degree(30) - degree(50)

degree(30) * 2
degree(30) / 3

radian(1) + 1 # added or subtracted numbers are radians
degree(10) + 10 # added or subtracted numbers are degrees
turn(.25) + .25 # added or subtracted numbers are turns

# Trigonometric functions work as normal
sin(degree(30))
cos(degree(30))
tan(degree(30))
```

---

ob_arc *ob_arc class*

Description

Create arcs and wedges

Usage

```
ob_arc(
  center = ob_point(0, 0),
  radius = 1,
  start = 0,
  end = 0,
  label = character(0),
  start_point = S7::class_missing,
  end_point = S7::class_missing,
  n = 360,
  type = "arc",
  alpha = numeric(0),
```

```
  arrow_head = list(),
  arrow_fins = list(),
  arrowhead_length = numeric(0),
  length_head = numeric(0),
  length_fins = numeric(0),
  color = character(0),
  fill = character(0),
  lineend = numeric(0),
  linejoin = numeric(0),
  linewidth = numeric(0),
  linewidth_fins = numeric(0),
  linewidth_head = numeric(0),
  linetype = numeric(0),
  resect = numeric(0),
  resect_fins = numeric(0),
  resect_head = numeric(0),
  stroke_color = character(0),
  stroke_width = numeric(0),
  style = S7::class_missing,
  x0 = numeric(0),
  y0 = numeric(0),
  ...
)

ob_wedge(
  center = ob_point(0, 0),
  radius = 1,
  start = 0,
  end = 0,
  label = character(0),
  start_point = S7::class_missing,
  end_point = S7::class_missing,
  n = 360,
  type = "wedge",
  alpha = numeric(0),
  arrow_head = list(),
  arrow_fins = list(),
  arrowhead_length = numeric(0),
  length_head = numeric(0),
  length_fins = numeric(0),
  color = NA,
  fill = "black",
  lineend = numeric(0),
  linejoin = numeric(0),
  linewidth = numeric(0),
  linewidth_fins = numeric(0),
  linewidth_head = numeric(0),
  linetype = numeric(0),
```

```
  resect = numeric(0),
  resect_fins = numeric(0),
  resect_head = numeric(0),
  stroke_color = character(0),
  stroke_width = numeric(0),
  style = S7::class_missing,
  x0 = numeric(0),
  y0 = numeric(0),
  ...
)

ob_circular_segment(
  center = ob_point(0, 0),
  radius = 1,
  start = 0,
  end = 0,
  label = character(0),
  start_point = S7::class_missing,
  end_point = S7::class_missing,
  n = 360,
  type = "segment",
  alpha = numeric(0),
  arrow_head = list(),
  arrow_fins = list(),
  arrowhead_length = numeric(0),
  length_head = numeric(0),
  length_fins = numeric(0),
  color = NA,
  fill = "black",
  lineend = numeric(0),
  linejoin = numeric(0),
  linewidth = numeric(0),
  linewidth_fins = numeric(0),
  linewidth_head = numeric(0),
  linetype = numeric(0),
  resect = numeric(0),
  resect_fins = numeric(0),
  resect_head = numeric(0),
  stroke_color = character(0),
  stroke_width = numeric(0),
  style = S7::class_missing,
  x0 = numeric(0),
  y0 = numeric(0),
  ...
)
```

Arguments

`center`	point at center of the arc (default = ob_point(0,0))

radius	distance between center and edge arc (default = 1)
start	start angle (default = 0 degrees)
end	end angle (default = 0 degrees)
label	A character, angle, or label object
start_point	Specify where arc starts. Overrides @center
end_point	Specify where arc ends Overrides @center
n	number of points in arc (default = 360)
type	Type of object to drawn. Can be "arc", "wedge", or "segment"
alpha	numeric value for alpha transparency
arrow_head	A 2-column matrix of polygon points
arrow_fins	A 2-column matrix of polygon points
arrowhead_length	Determines the size of the arrow ornaments. This parameter becomes the length parameter in ggarrow functions. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner.
length_head	Determines the size of the arrow head. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
length_fins	Determines the size of the arrow fins. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
color	character string for color
fill	character string for fill color
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linewidth	Width of lines
linewidth_fins	Line width for arrow fins
linewidth_head	Line width for arrow fins
linetype	type of lines
resect	A numeric(1) denoting millimeters or to shorten the arrow head and fins.
resect_fins	A numeric(1) denoting millimeters or to shorten the arrow fins
resect_head	A numeric(1) denoting millimeters or to shorten the arrow head.
stroke_color	Color of point border line
stroke_width	Stroke width in arrows
style	a style object
x0	x-coordinate of center point. If specified, overrides x-coordinate of @center.
y0	x-coordinate of center point. If specified, overrides y-coordinate of @center.
...	<dynamic-dots> arguments passed to style object

Value

ob_arc object

Slots

`aesthetics` A list of information about the arc's aesthetic properties

`angle_at` A function that finds the angle of the specified point in relation to the arc's center

`apothem` Distance from center to the chord's midpoint

`arc_length` Distance along arc from `start_point` to `end_point`

`auto_label` Places a label at the arc's midpoint

`chord` `ob_segment` from `start_point` to `end_point`

`geom` A function that converts the object to a geom. Any additional parameters are passed to `ggarrow::geom_arrow`.

`hatch` A function that puts hatch (tally) marks on arcs. Often used to indicate which arcs have the same angle. The `k` parameter controls how many hatch marks to display. The `height` parameter controls how long the hatch mark segment is. The `sep` parameter controls the separation between hatch marks when `k > 2`. Additional parameters sent to `ob_segment`.

`length` The number of arcs in the arc object

`midpoint` A function that selects 1 or more midpoints of the ob_arc. The `position` argument can be between 0 and 1. Additional arguments are passed to `ob_point`.

`point_at` A function that finds a point on the arc at the specified angle.

`sagitta` `ob_segment` from `chord` midpoint to `ob_arc` midpoint

`tangent_at` A function that finds the tangent line at the specified angle.

`theta` interior angle (end - start)

`tibble` Gets a tibble (data.frame) containing parameters and styles used by `ggarrow::geom_arrow`.

Examples

```
library(ggplot2)

# center point
p_center <- ob_point(0,0)

# 90-degree arc
a_90 <- ob_arc(
 center = p_center,
 radius = 6,
 start = degree(0),
 end = degree(90)
 )

 # Print arc
 a_90
```

```
# Plot arc and its center point
ggplot() + coord_equal() + theme_minimal() +
 p_center +
 a_90
```

`ob_array` *make an array of shapes along a line*

Description

make an array of shapes along a line

Usage

```
ob_array(x, k = 2, sep = 1, where = "east", anchor = "center", ...)
```

Arguments

`x`	shape
`k`	number of duplicate shapes to make
`sep`	separation distance between shapes
`where`	angle or named direction (e.g.,northwest, east, below, left)
`anchor`	bounding box anchor
`...`	<dynamic-dots> properties passed to shape

Value

An array of shapes of the same class as object passed to x

`ob_bezier` *The ob_bezier (i.e., bezier curve) class*

Description

The ob_bezier is specified with an ob_point object that contains at least 2 points, the start and the end. Such a "curve" would actually be a straight line segment. If three points are specified, the middle point is a control point, and a quadratic bezier curve will result. Higher-order bezier curves can be created by having more control points in the middle.

Usage

```
ob_bezier(
  p = S7::class_missing,
  label = character(0),
  label_sloped = TRUE,
  n = 360,
  alpha = numeric(0),
  arrow_head = S7::class_missing,
  arrow_fins = S7::class_missing,
  arrowhead_length = numeric(0),
  length_head = numeric(0),
  length_fins = numeric(0),
  color = character(0),
  fill = character(0),
  lineend = numeric(0),
  linejoin = numeric(0),
  linewidth = numeric(0),
  linewidth_fins = numeric(0),
  linewidth_head = numeric(0),
  linetype = numeric(0),
  resect = numeric(0),
  resect_fins = numeric(0),
  resect_head = numeric(0),
  stroke_color = character(0),
  stroke_width = numeric(0),
  style = S7::class_missing,
  ...
)
```

Arguments

`p`	ob_point or list of ob_points
`label`	A character, angle, or label object
`label_sloped`	A logical value indicating whether the label should be sloped with the curve
`n`	Number of points in a polygon, circle, arc, or ellipse
`alpha`	numeric value for alpha transparency
`arrow_head`	A 2-column matrix of polygon points
`arrow_fins`	A 2-column matrix of polygon points
`arrowhead_length`	Determines the size of the arrow ornaments. This parameter becomes the `length` parameter in ggarrow functions. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner.
`length_head`	Determines the size of the arrow head. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.

length_fins	Determines the size of the arrow fins. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
color	character string for color
fill	character string for fill color
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linewidth	Width of lines
linewidth_fins	Line width for arrow fins
linewidth_head	Line width for arrow fins
linetype	type of lines
resect	A numeric(1) denoting millimeters or to shorten the arrow head and fins.
resect_fins	A numeric(1) denoting millimeters or to shorten the arrow fins
resect_head	A numeric(1) denoting millimeters or to shorten the arrow head.
stroke_color	Color of point border line
stroke_width	Stroke width in arrows
style	Gets and sets the styles associated with ob_beziers
...	<dynamic-dots> properties passed to style

Details

If you wish to specify multiple bezier curves, you must supply a list of ob_point objects. When plotted, the ob_bezier function uses the bezier::bezier function to create the point coordinates of the curve and the ggarrow::geom_arrow function to create the geom.

Value

ob_bezier object

Slots

`length` The number of curves in the ob_bezier object

`tibble` Gets a tibble (data.frame) containing parameters and styles used by `ggarrow::geom_arrow`.

`geom` A function that converts the object to a geom. Any additional parameters are passed to `ggarrow::geom_arrow`.

`midpoint` A function that selects 1 or more midpoints of the ob_bezier. The `position` argument can be between 0 and 1. Additional arguments are passed to `ob_point`.

`aesthetics` A list of information about the ob_bezier's aesthetic properties

Examples

```
library(ggplot2)
control_points <- ob_point(c(0,1,2,4), c(0,4,0,0))
ggplot() +
  coord_equal() +
  ob_bezier(control_points, color = "blue") +
  ob_path(control_points, linetype = "dashed", linewidth = .5) +
  control_points
```

---

ob_circle *ob_circle class*

Description

ob_circle class

Usage

```
ob_circle(
  center = ob_point(0, 0),
  radius = 1,
  label = character(0),
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  n = numeric(0),
  style = S7::class_missing,
  x0 = numeric(0),
  y0 = numeric(0),
  ...
)
```

Arguments

center	point at center of the circle
radius	distance between center and edge circle
label	A character, angle, or label object
alpha	numeric value for alpha transparency
color	character string for color
fill	character string for fill color
linewidth	Width of lines
linetype	type of lines
n	number of points in circle (default = 360)

style	an ob_style object
x0	x-coordinate of center point. If specified, overrides x-coordinate of @center.
y0	x-coordinate of center point. If specified, overrides y-coordinate of @center.
...	<dynamic-dots> arguments passed to style object

Value

ob_circle object

Slots

aesthetics A list of information about the circle's aesthetic properties

angle_at A function that finds the angle of the specified point in relation to the circle's center

area area of the circle

bounding_box a rectangle that contains all the circles

circumference circumference of the circle

geom A function that converts the object to a geom. Any additional parameters are passed to ggforce::geom_circle.

length The number of circles in the circle object

normal_at A function that finds a point that is perpendicular from the circle and at a specified distance

point_at A function that finds a point on the circle at the specified angle.

polygon a tibble containing information to create all the polygon points in a circle.

tangent_at A function that finds the tangent line at the specified angle.

tibble Gets a tibble (data.frame) containing parameters and styles used by ggforce::geom_cirlce.

Examples

```
# specify center point and radius
p <- ob_point(0,0)
ob_circle(p, radius = 6)
```

---

ob_covariance *create double-headed arrow paths indicating variance*

Description

create double-headed arrow paths indicating variance

Usage

```
ob_covariance(
  x,
  y,
  where = NULL,
  bend = 0,
  looseness = 1,
  arrow_head = arrowheadr::arrow_head_deltoid(d = 2.3, n = 100),
  resect = 2,
  ...
)
```

Arguments

x	object
y	object
where	exit angle
bend	Angle by which the control points are rotated
looseness	distance of control points as a ratio of the distance to the object's center (e.g., in a circle of radius 1, looseness = 1.5 means that that the control points will be 1.5 units from the start and end points.)
arrow_head	A 2-column matrix of polygon points
resect	A numeric(1) denoting millimeters or to shorten the arrow head and fins.
...	<dynamic-dots> properties passed to style

Value

An ob_bezier object

---

ob_ellipse *ob_ellipse class*

Description

Makes ellipses and superellipses

Usage

```
ob_ellipse(
  center = ob_point(0, 0),
  a = 1,
  b = a,
  angle = 0,
  m1 = numeric(0),
  m2 = numeric(0),
```

```
  label = character(0),
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  n = numeric(0),
  style = S7::class_missing,
  x0 = numeric(0),
  y0 = numeric(0),
  ...
)
```

Arguments

`center`	point at center of ellipse. *Settable.*
`a`	distance of semi-major axis. *Settable.*
`b`	distance of semi-minor axis. *Settable.*
`angle`	ellipse rotation. *Settable.*
`m1`	exponent of semi-major axis. *Settable.* Controls roundedness of superellipse
`m2`	exponent of semi-minor axis. *Settable.* By default equal to `m1`. If different, some functions may not work as expected (e.g., `point_at`).
`label`	A character, angle, or label object
`alpha`	numeric value for alpha transparency
`color`	character string for color
`fill`	character string for fill color
`linewidth`	Width of lines
`linetype`	type of lines
`n`	number of points in ellipse (default = 360). *Settable.*
`style`	gets and sets style parameters
`x0`	x-coordinate of center point. If specified, overrides x-coordinate of `@center`.
`y0`	x-coordinate of center point. If specified, overrides y-coordinate of `@center`.
`...`	<dynamic-dots> arguments passed to style object

Value

ob_ellipse object

Slots

`length` Gets the number of ellipses

`tibble` Gets a tibble (data.frame) containing parameters and styles used by `ggforce::geom_ellipse`.

`geom` A function that converts the object to a geom. Any additional parameters are passed to `ggforce::geom_ellipse`.

`normal_at` A function that finds a point perpendicular to the ellipse at angle `theta` at the specified `distance`. The `definitional` parameter is passed to the `point_at` function. If a point is supplied instead of an angle, the point is projected onto the ellipse and then the normal is calculated found from the projected point.

`point_at` A function that finds a point on the ellipse at an angle `theta`. If `definitional` is `FALSE` (default), then `theta` is interpreted as an angle. If `TRUE`, then `theta` is the parameter in the definition of the ellipse in polar coordinates.

`tangent_at` A function that finds a tangent line on the ellipse. Uses `point_at` to find the tangent point at angle `theta` and then returns the tangent line at that point. If a point is supplied instead of an angle, the point is projected onto the ellipse and then the tangent line is found from there.

Examples

```
# specify center point and semi-major axes
p <- ob_point(0,0)
ob_ellipse(p, a = 2, b = 3)
```

`ob_intercept` *ob_intercept*

Description

Triangle polygons used in path diagrams.

Usage

```
ob_intercept(
  center = ob_point(0, 0),
  width = 1,
  label = character(0),
  top = S7::class_missing,
  left = S7::class_missing,
  right = S7::class_missing,
  vertex_radius = numeric(0),
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  x0 = numeric(0),
  y0 = numeric(0),
  style = S7::class_missing,
  ...
)
```

Arguments

center	point at center
width	length of side
label	A character, angle, or label object
top	Top vertex of triangle
left	Left vertex of triangle
right	Right vertex of triangle
vertex_radius	A numeric or unit vector of length one, specifying the vertex radius
alpha	numeric value for alpha transparency
color	character string for color
fill	character string for fill color
linewidth	Width of lines
linetype	type of lines
x0	overrides x-coordinate in center@x
y0	overrides x-coordinate in center@y
style	Gets and sets the styles associated with polygons
...	<dynamic-dots> properties passed to style

Value

ob_polygon object

Slots

length The number of polygons in the ob_polygon object

tibble Gets a tibble (data.frame) containing parameters and styles used by ggplot2::geom_polygon.

---

ob_label *ob_label class*

Description

ob_label class

Usage

```
ob_label(
  label = character(0),
  center = S7::class_missing,
  angle = numeric(0),
  alpha = numeric(0),
  color = character(0),
  family = character(0),
  fill = character(0),
  fontface = character(0),
  hjust = numeric(0),
  label.color = character(0),
  label.margin = class_margin(ggplot2::margin(1, 1, 1, 1, "pt")),
  label.padding = class_margin(ggplot2::margin(2, 2, 2, 2, "pt")),
  label.r = numeric(0),
  label.size = numeric(0),
  lineheight = numeric(0),
  polar_just = numeric(0),
  nudge_x = numeric(0),
  nudge_y = numeric(0),
  size = numeric(0),
  straight = logical(0),
  text.color = character(0),
  vjust = numeric(0),
  style = S7::class_missing,
  plot_point = FALSE,
  position = 0.5,
  spacing = numeric(0),
  x = S7::class_missing,
  y = S7::class_missing,
  ...
)
```

Arguments

`label`	text label
`center`	ob_point indicating the center of the label
`angle`	angle of text
`alpha`	numeric value for alpha transparency
`color`	character string for color
`family`	font family
`fill`	character string for fill color
`fontface`	Can be plain, bold, italic, or bold.italic
`hjust`	horizontal justification. 0 means left justified, 1 means right justified, 0.5 means horizontally centered
`label.color`	Color of label outline.

label.margin	Amount of distance around label. Unit vector of length four. Usually created with ggplot2::margin.
label.padding	Amount of padding around label. Unit vector of length four. Usually created with ggplot2::margin.
label.r	Radius of rounded corners. Defaults to 0.15 lines.
label.size	Width of label outline.
lineheight	Height of line of text
polar_just	an angle, polar point, or point that alters hjust and vjust (polar polar_just not stored in style)
nudge_x	Horizontal adjustment to nudge labels by.
nudge_y	Vertical adjustment to nudge labels by.
size	numeric size
straight	logical. If TRUE, make bzpath label text straight instead of curved.
text.color	Color of label text.
vjust	vertical justification. 0 means bottom aligned, 1 means top aligned, 0.5 means vertically centered
style	a style list
plot_point	plot ob_point (default = FALSE)
position	position (used in conjunction with the place function)
spacing	letter spacing for labels used with ob_path and ob_bezier
x	x-coordinate of center point. If specified, overrides x-coordinate of @center.
y	x-coordinate of center point. If specified, overrides y-coordinate of @center.
...	<dynamic-dots> properties passed to style

Value

ob_label object

---

ob_latex *ob_latex class*

Description

make a latex equation

Usage

```
ob_latex(
  tex = character(0),
  center = ob_point(0, 0),
  width = numeric(0),
  height = numeric(0),
  hjust = 0.5,
  vjust = 0.5,
  angle = 0,
  aspect_ratio = 1,
  border = numeric(0),
  family = character(0),
  math_mode = TRUE,
  filename = character(0),
  color = character(0),
  fill = "white",
  density = 300,
  latex_packages = character(0),
  preamble = character(0),
  force_recompile = TRUE,
  delete_files = TRUE
)
```

Arguments

`tex`	LaTeX equation
`center`	ob_point
`width`	width (specify width or height but not both)
`height`	height (specify width or height but not both)
`hjust`	horizontal adjustment. 0 means left justified, 1 means right justified, 0.5 means centered
`vjust`	vertical justification. 0 means bottom aligned, 1 means top aligned, 0.5 means vertically centered
`angle`	angle of text
`aspect_ratio`	alters the aspect ratio of the image
`border`	border space (in points) around image
`family`	font family (installed on system) of plain text
`math_mode`	include dollar signs automatically. Set to `FALSE` when the latex command is not in math mode
`filename`	bare file name without extension (e.g., `myequation`)
`color`	set color of equation text
`fill`	set color of background rectangle
`density`	image quality (dots per inch)

`latex_packages`
: load latex packages

`preamble`
: additional latex commands to load in preamble

`force_recompile`
: Will re-run xelatex even if .pdf file exists already

Value

ob_latex object

Slots

`rectangle` gets or sets rectangle that contains the image

`image` raster bitmap

`ob_line` *ob_line class*

Description

Creates a line

Usage

```
ob_line(
  slope = numeric(0),
  intercept = numeric(0),
  xintercept = numeric(0),
  a = numeric(0),
  b = numeric(0),
  c = numeric(0),
  alpha = numeric(0),
  color = character(0),
  lineend = numeric(0),
  linejoin = numeric(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  style = S7::class_missing,
  ...
)
```

Arguments

`slope`
: coefficient in y = slope * x + intercept

`intercept`
: value of y when x is 0

`xintercept`
: value of x when y is 0

a	coefficient in general form: a * x + b * y + c = 0
b	coefficient in general form: a * x + b * y + c = 0
c	constant in general form: a * x + b * y + c = 0
alpha	numeric value for alpha transparency
color	character string for color
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linewidth	Width of lines
linetype	type of lines
style	a style list
...	<dynamic-dots> properties passed to style

Value

ob_line object

`ob_ngon` *The ob_ngon (regular polygon) class*

Description

An ngon is a regular polygon, meaning that each side is of equal length. The `ob_ngon` object can be specified with a center, n (number of sides), radius, and angle. Instead of specifying a radius, one can specify either the `side_length` or the length of the `apothem` (i.e., the distance from the center to a side's midpoint.

Usage

```
ob_ngon(
  center = ob_point(0, 0),
  n = 3L,
  radius = numeric(0),
  angle = 0,
  label = character(0),
  side_length = numeric(0),
  apothem = numeric(0),
  vertex_radius = numeric(0),
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  style = S7::class_missing,
  x0 = numeric(0),
  y0 = numeric(0),
  ...
)
```

Arguments

center	point at center of the ngon
n	Number of sides
radius	Distance from center to a vertex
angle	description
label	A character, angle, or label object
side_length	Distance of each side
apothem	Distance from center to a side's midpoint
vertex_radius	A numeric or unit vector of length one, specifying the corner radius
alpha	numeric value for alpha transparency
color	character string for color
fill	character string for fill color
linewidth	Width of lines
linetype	type of lines
style	Gets and sets the styles associated with ob_ngon
...	<dynamic-dots> properties passed to style

Value

ob_ngon object

Slots

area The area of the ngons in the ob_ngon object

length The number of ngons in the ob_ngon object

normal_at A function that finds a point that is perpendicular from the ngon and at a specified distance

perimeter The length of all the side segments

point_at A function that finds a point on the ngon at the specified angle.

segments side segments of the regular polygon

tangent_at A function that finds the tangent line at the specified angle.

tibble Gets a tibble (data.frame) containing parameters and styles used by `ggforce::geom_shape`.

vertices points on the regular polygon

`ob_path` *The ob_path class*

Description

A `ob_path` is specified with an `ob_point` object that contains at least 2 points, the start and the end. Any number of intermediate points are possible.

Usage

```
ob_path(
  p = S7::class_missing,
  label = character(0),
  alpha = numeric(0),
  arrow_head = S7::class_missing,
  arrow_fins = S7::class_missing,
  arrowhead_length = numeric(0),
  length_head = numeric(0),
  length_fins = numeric(0),
  color = character(0),
  fill = character(0),
  lineend = numeric(0),
  linejoin = numeric(0),
  linewidth = numeric(0),
  linewidth_fins = numeric(0),
  linewidth_head = numeric(0),
  linetype = numeric(0),
  resect = numeric(0),
  resect_fins = numeric(0),
  resect_head = numeric(0),
  stroke_color = character(0),
  stroke_width = numeric(0),
  style = S7::class_missing,
  ...
)
```

Arguments

`p`	`ob_point` or list of `ob_point`s
`label`	A character, angle, or label object
`alpha`	numeric value for alpha transparency
`arrow_head`	A 2-column matrix of polygon points
`arrow_fins`	A 2-column matrix of polygon points
`arrowhead_length`	Determines the size of the arrow ornaments. This parameter becomes the `length` parameter in ggarrow functions. Numeric values set the ornament

	size relative to the linewidth. A unit value sets the ornament size in an absolute manner.
`length_head`	Determines the size of the arrow head. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
`length_fins`	Determines the size of the arrow fins. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
`color`	character string for color
`fill`	character string for fill color
`lineend`	Line end style (round, butt, square).
`linejoin`	Line join style (round, mitre, bevel).
`linewidth`	Width of lines
`linewidth_fins`	Line width for arrow fins
`linewidth_head`	Line width for arrow fins
`linetype`	type of lines
`resect`	A numeric(1) denoting millimeters or to shorten the arrow head and fins.
`resect_fins`	A numeric(1) denoting millimeters or to shorten the arrow fins
`resect_head`	A numeric(1) denoting millimeters or to shorten the arrow head.
`stroke_color`	Color of point border line
`stroke_width`	Stroke width in arrows
`style`	Gets and sets the styles associated with paths
`...`	<dynamic-dots> properties passed to style

Details

If you wish to specify multiple paths, you must supply a list of `ob_point` objects. When plotted, the `ob_path` function uses the ggarrow::geom_arrow function to create the geom.

Value

ob_path object

Slots

`length` The number of paths in the ob_path object

`tibble` Gets a tibble (data.frame) containing parameters and styles used by `ggarrow::geom_arrow`.

ob_point *ob_point*

Description

Points are specified with x and y coordinates.

Polar points are ordinary points but are specified with an angle (theta) and a radial distance (r)

Usage

```
ob_point(
  x = 0,
  y = 0,
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  shape = numeric(0),
  size = numeric(0),
  stroke = numeric(0),
  style = S7::class_missing,
  ...
)

ob_polar(
  theta = S7::class_missing,
  r = numeric(0),
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  shape = numeric(0),
  size = numeric(0),
  stroke = numeric(0),
  style = S7::class_missing
)
```

Arguments

x	Vector of coordinates on the x-axis (also can take a tibble/data.frame or 2-column matrix as input.)
y	Vector of coordinates on the y-axis
alpha	numeric value for alpha transparency
color	character string for color
fill	character string for fill color

`shape`	Point shape type. Can be specified with an integer (between 0 and 25), a single character (which uses that character as the plotting symbol), a . to draw the smallest rectangle that is visible (i.e., about one pixel), an NA to draw nothing, or a mapping to a discrete variable.
`size`	numeric size
`stroke`	Width of point border line
`style`	Gets and sets the styles associated with points
`...`	<dynamic-dots> properties passed to style
`theta`	Angle of the vector from the origin to the ob_point
`r`	Radius = Distance from the origin to the ob_point

Value

ob_point object

Slots

`auto_label` Gets x and y coordinates and makes a label `"(x,y)"`

`length` The number of points in the ob_point object

`tibble` Gets a tibble (data.frame) containing parameters and styles used by `ggplot2::geom_point`.

`xy` Gets a 2-column matrix of the x and y coordinates of the ob_point object.

`geom` A function that converts the object to a geom. Any additional parameters are passed to `ggplot2::geom_point`.

`ob_polygon` *The ob_polygon class*

Description

A polygon is specified with an ob_point that contains at least 3 points, the start and the end. Any number of intermediate points are possible.

Usage

```
ob_polygon(
  p = S7::class_missing,
  label = character(0),
  vertex_radius = numeric(0),
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  style = S7::class_missing,
  ...
)
```

Arguments

p	ob_point or list of ob_point objects
label	A character, angle, or label object
vertex_radius	A numeric or unit vector of length one, specifying the corner radius
alpha	numeric value for alpha transparency
color	character string for color
fill	character string for fill color
linewidth	Width of lines
linetype	type of lines
style	Gets and sets the styles associated with polygons
...	<dynamic-dots> properties passed to style

Details

If you wish to specify multiple polygons, you must supply a list of ob_points. When plotted, the ob_polygon function uses the ggforce::geom_shape function to create the geom.

Value

ob_polygon object

Slots

`length` The number of polygons in the ob_polygon object

`tibble` Gets a tibble (data.frame) containing parameters and styles used by `ggforce::geom_shape`.

---

ob_rectangle *ob_rectangle class*

Description

ob_rectangle class

Usage

```
ob_rectangle(
  center = S7::class_missing,
  width = numeric(0),
  height = numeric(0),
  east = S7::class_missing,
  north = S7::class_missing,
  west = S7::class_missing,
  south = S7::class_missing,
  northeast = S7::class_missing,
```

```
  northwest = S7::class_missing,
  southwest = S7::class_missing,
  southeast = S7::class_missing,
  angle = numeric(0),
  vertex_radius = numeric(0),
  label = character(0),
  alpha = numeric(0),
  color = character(0),
  fill = character(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  style = S7::class_missing,
  x0 = numeric(0),
  y0 = numeric(0),
  ...
)
```

Arguments

`center`	point at center of the rectangle
`width`	width
`height`	height
`east`	right middle point
`north`	top middle point
`west`	left middle point
`south`	top middle point
`northeast`	upper right point
`northwest`	upper left point
`southwest`	lower left point
`southeast`	lower right point
`angle`	angle of text
`vertex_radius`	A numeric or unit vector of length one, specifying the corner radius for rounded corners
`label`	A character, angle, or label object
`alpha`	numeric value for alpha transparency
`color`	character string for color
`fill`	character string for fill color
`linewidth`	Width of lines
`linetype`	type of lines
`style`	a style object
`x0`	overrides x-coordinate in `center@x`
`y0`	overrides y-coordinate in `center@x`
`...`	<dynamic-dots> arguments passed to style object

Value

ob_rectangle object

Examples

```
# specify center point
p <- ob_point(0,0)
ob_rectangle(p, width = 2, height = 2)
```

---

ob_reuleaux *Reuleaux polygon*

Description

Reuleaux polygon

Usage

```
ob_reuleaux(
  center = ob_point(0, 0),
  n = 5,
  radius = 1,
  angle = 90,
  label = character(0),
  vertex_radius = numeric(0),
  alpha = numeric(0),
  color = "black",
  fill = character(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  style = S7::class_missing,
  ...
)
```

Arguments

center	point at center of the rectangle
n	Number of sides. True Reuleaux polygons have an odd number of sides, but Reauleaux-like shapes with an even number of sides are possible.
radius	Distance from center to a vertex
angle	angle of text
alpha	numeric value for alpha transparency
color	character string for color
fill	character string for fill color
linewidth	Width of lines
linetype	type of lines
...	<dynamic-dots> unused

Value

ob_reuleaux object

`ob_segment` *ob_segment class*

Description

ob_segment class

Usage

```
ob_segment(
  p1 = S7::class_missing,
  p2 = S7::class_missing,
  label = character(0),
  alpha = numeric(0),
  arrow_head = ggarrow::arrow_head_minimal(90),
  arrow_fins = list(),
  arrowhead_length = 4,
  length_head = numeric(0),
  length_fins = numeric(0),
  color = character(0),
  lineend = numeric(0),
  linejoin = numeric(0),
  linewidth = numeric(0),
  linewidth_fins = numeric(0),
  linewidth_head = numeric(0),
  linetype = numeric(0),
  resect = numeric(0),
  resect_fins = numeric(0),
  resect_head = numeric(0),
  stroke_color = character(0),
  stroke_width = numeric(0),
  style = S7::class_missing,
  x = S7::class_missing,
  xend = S7::class_missing,
  y = S7::class_missing,
  yend = S7::class_missing,
  ...
)
```

Arguments

`p1`	starting point
`p2`	end point

label	A character, angle, or label object
alpha	numeric value for alpha transparency
arrow_head	A 2-column matrix of polygon points
arrow_fins	A 2-column matrix of polygon points
arrowhead_length	Determines the size of the arrow ornaments. This parameter becomes the length parameter in ggarrow functions. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner.
length_head	Determines the size of the arrow head. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
length_fins	Determines the size of the arrow fins. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
color	character string for color
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linewidth	Width of lines
linewidth_fins	Line width for arrow fins
linewidth_head	Line width for arrow fins
linetype	type of lines
resect	A numeric(1) denoting millimeters or to shorten the arrow head and fins.
resect_fins	A numeric(1) denoting millimeters or to shorten the arrow fins
resect_head	A numeric(1) denoting millimeters or to shorten the arrow head.
stroke_color	Color of point border line
stroke_width	Stroke width in arrows
style	a style list
x	overrides the x-coordinate of p1
xend	overrides the y-coordinate of p1
y	overrides the x-coordinate of p2
yend	overrides the y-coordinate of p2
...	<dynamic-dots> properties passed to style

Value

ob_segment object

Slots

`geom` A function that converts the object to a geom. Any additional parameters are passed to `ggarrow::geom_arrow_segment`.

`hatch` A function that puts hatch (tally) marks on segments. Often used to indicate which segments have the same length. The `k` parameter controls how many hatch marks to display. The `height` parameter controls how long the hatch mark segment is. The `sep` parameter controls the separation between hatch marks when `k > 2`. Additional parameters sent to `ob_segment`.

`midpoint` A function that selects 1 or more midpoints of the ob_segment. The `position` argument can be between 0 and 1. Additional arguments are passed to `ob_point`.

`nudge` A function to move the segment by x and y units.

`ob_shape_list` *ob_shape_list*

Description

makes a heterogeneous list of different ggdiagram objects

Usage

```
ob_shape_list(.data = list())
```

Arguments

`.data`	a list of objects

Value

An object of `ob_shape_list` class. List of objects that can be converted to geoms

---

`ob_style` *ob_style class*

Description

ob_style class

Usage

```
ob_style(
  alpha = numeric(0),
  angle = numeric(0),
  arrow_head = list(),
  arrow_fins = list(),
  arrow_mid = list(),
  color = character(0),
  family = character(0),
  fill = character(0),
  fontface = character(0),
  hjust = numeric(0),
  justify = numeric(0),
  label.color = character(0),
  label.margin = list(),
  label.padding = list(),
  label.r = numeric(0),
  label.size = numeric(0),
  arrowhead_length = numeric(0),
  length_head = numeric(0),
  length_fins = numeric(0),
  length_mid = numeric(0),
  lineend = numeric(0),
  lineheight = numeric(0),
  linejoin = numeric(0),
  linewidth_fins = numeric(0),
  linewidth_head = numeric(0),
  linewidth = numeric(0),
  linetype = numeric(0),
  n = numeric(0),
  nudge_x = numeric(0),
  nudge_y = numeric(0),
  polar_just = numeric(0),
  resect = numeric(0),
  resect_fins = numeric(0),
  resect_head = numeric(0),
  shape = numeric(0),
  size = numeric(0),
  size.unit = numeric(0),
  straight = logical(0),
  stroke = numeric(0),
  stroke_color = character(0),
  stroke_width = numeric(0),
  text.color = character(0),
  vjust = numeric(0),
  ...
)
```

Arguments

`alpha`	numeric value for alpha transparency
`angle`	angle of text
`arrow_head`	A 2-column matrix of polygon points
`arrow_fins`	A 2-column matrix of polygon points
`arrow_mid`	A 2-column matrix of polygon points
`color`	character string for color
`family`	font family
`fill`	character string for fill color
`fontface`	Can be plain, bold, italic, or bold.italic
`hjust`	horizontal justification. 0 means left justified, 1 means right justified, 0.5 means horizontally centered
`justify`	A numeric(1) between 0 and 1 to control where the arrows should be drawn relative to the path's endpoints. A value of 0 sets the arrow's tips at the path's end, whereas a value of 1 sets the arrow's base at the path's end. From ggarrow.
`label.color`	Color of label outline.
`label.margin`	Amount of distance around label. Unit vector of length four. Usually created with `ggplot2::margin`.
`label.padding`	Amount of padding around label. Unit vector of length four. Usually created with `ggplot2::margin`.
`label.r`	Radius of rounded corners. Defaults to 0.15 lines.
`label.size`	Width of label outline.
`arrowhead_length`	Determines the size of the arrow ornaments. This parameter becomes the `length` parameter in ggarrow functions. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner.
`length_head`	Determines the size of the arrow head. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
`length_fins`	Determines the size of the arrow fins. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
`length_mid`	Determines the size of the middle arrows. Numeric values set the ornament size relative to the linewidth. A unit value sets the ornament size in an absolute manner. From ggarrow.
`lineend`	Line end style (round, butt, square).
`lineheight`	Height of line of text
`linejoin`	Line join style (round, mitre, bevel).
`linewidth_fins`	Line width for arrow fins

linewidth_head	Line width for arrow fins
linewidth	Width of lines
linetype	type of lines
n	Number of points in a polygon, circle, arc, or ellipse
nudge_x	Horizontal adjustment to nudge labels by.
nudge_y	Vertical adjustment to nudge labels by.
polar_just	an angle, polar point, or point that alters hjust and vjust (polar polar_just not stored in style)
resect	A numeric(1) denoting millimeters or to shorten the arrow head and fins.
resect_fins	A numeric(1) denoting millimeters or to shorten the arrow fins
resect_head	A numeric(1) denoting millimeters or to shorten the arrow head.
shape	Point shape type. Can be specified with an integer (between 0 and 25), a single character (which uses that character as the plotting symbol), a . to draw the smallest rectangle that is visible (i.e., about one pixel), an NA to draw nothing, or a mapping to a discrete variable.
size	numeric size
size.unit	How the size aesthetic is interpreted: as points ("pt"), millimeters ("mm"), centimeters ("cm"), inches ("in"), or picas ("pc").
straight	logical. If TRUE, make bzpath label text straight instead of curved.
stroke	Width of point border line
stroke_color	Color of point border line
stroke_width	Stroke width in arrows
text.color	Color of label text.
vjust	vertical justification. 0 means bottom aligned, 1 means top aligned, 0.5 means vertically centered
...	<dynamic-dots> unused

Value

ob_style object

`ob_variance` *create double-headed arrow paths indicating variance*

Description

create double-headed arrow paths indicating variance

Usage

```
ob_variance(
  x,
  where = "north",
  theta = 50,
  bend = 0,
  looseness = 1,
  arrow_head = arrowheadr::arrow_head_deltoid(d = 2.3, n = 100),
  resect = 2,
  ...
)
```

Arguments

`x`	object
`where`	angle or named direction (e.g.,northwest, east, below, left)
`theta`	angle width
`bend`	Angle by which the control points are rotated
`looseness`	distance of control points as a ratio of the distance to the object's center (e.g., in a circle of radius 1, looseness = 1.5 means that that the control points will be 1.5 units from the start and end points.)
`arrow_head`	A 2-column matrix of polygon points
`resect`	A numeric(1) denoting millimeters or to shorten the arrow head and fins.
`...`	<dynamic-dots> properties passed to style

Value

Returns an object of type `ob_bezier`

`perpendicular_point` *Find point perpendicular to 2 points*

Description

Find point perpendicular to 2 points

Usage

```
e1 %|-% e2

e1 %-|% e2
```

Arguments

`e1`	first ob_point
`e2`	second ob_point

Value

ob_point object

ob_point object

Examples

```
x <- ob_point(0,0)
y <- ob_point(1,1)
# Find point perpendicular to x and y going vertically first
x %|-% y
# Find point perpendicular to x and y going horizontally first
x %-|% y
```

`place` *Place an object a specified distance from another object*

Description

Place an object a specified distance from another object

Usage

```
place(x, from, where = "right", sep = 1, ...)
```

Arguments

`x`	shape object
`from`	shape that x is placed in relation to
`where`	named direction, angle, or number (degrees)
`sep`	separation distance
`...`	<dynamic-dots> Arguments passed to style

Value

object of same class as `x`

polar2just *polar2just*

Description

Convert hjust and vjust parameters from polar coordinates

Usage

```
polar2just(x, multiplier = 1.2, axis = c("h", "v"))
```

Arguments

x	angle
multiplier	distance
axis	vertical (v) or horizontal (h)

Value

ob_angle object

projection *Find projection of a point on an object (e.g., line or segment)*

Description

Find projection of a point on an object (e.g., line or segment)

Usage

```
projection(p, object, ...)
```

Arguments

p	ob_point
object	object (e.g., line or segment)
...	<dynamic-dots> arguments passed to style object

Value

ob_point

`redefault` *Make a variant of a function with alternate defaults*

Description

Makes a copy of a function with new defaults. Similar to `purrr::partial` except that arguments with new defaults still accept input.

Usage

```
redefault(.f, ...)
```

Arguments

`.f`	function
`...`	<dynamic-dots> new defaults

Value

function

Examples

```
squircle <- redefault(ob_ellipse, m1 = 4)
squircle(a = 3)
```

`resect` *resect*

Description

Shorten segments

Usage

```
resect(x, distance, ...)
```

Arguments

`x`	object
`distance`	resect distance
`...`	<dynamic-dots> properties passed to style
`resect`	a numeric distance

Value

object of same class as x

rotate *Rotate an object in 2 dimensions*

Description

Rotate an object in 2 dimensions

Usage

```
rotate(x, theta, ..., origin = ob_point(0, 0))
```

Arguments

x	object
theta	angle
...	<dynamic-dots> properties passed to style
origin	length 2 vector or point about which rotation occurs

Value

shape object

round_probability *Probability rounding*

Description

Rounds to significant digits, removing leading zeros.

Usage

```
round_probability(
  p,
  accuracy = 0.01,
  digits = NULL,
  max_digits = NULL,
  remove_leading_zero = TRUE,
  round_zero_one = TRUE,
  phantom_text = NULL,
  phantom_color = NULL
)
```

Arguments

p	probability
accuracy	smallest increment
digits	significant digits
max_digits	maximum rounding digits
remove_leading_zero	remove leading zero
round_zero_one	round 0 and 1
phantom_text	invisible text inserted on the right
phantom_color	color of phantom text

Value

a character vector

Examples

```
round_probability(c(0, .0012, .012, .12, .99, .992, .9997, 1), digits = 2)
```

---

signs_centered *Centering signed numbers*

Description

A wrapper function for the signs::signs function. It adds a space to the right side of negative numbers so that it appear as if the minus sign does not affect the number's centering.

Usage

```
signs_centered(x, space = " ", encoding = "UTF-8", ...)
```

Arguments

x	a numeric vector
space	a character to be added to negative numbers (defaults to a UTF-8 figure space)
encoding	type of encoding (defaults to UTF-8)
...	parameters passed to signs:signs

Value

a vector of numbers converted to characters

Examples

```
library(ggplot2)
d <- data.frame(x = -4:0, y = -4:0)
# In these 2 plots, Compare the centering of the negative numbers on the x-axis
ggplot(d, aes(x,y))
ggplot(d, aes(x,y)) +
  scale_x_continuous(labels = signs_centered)
```

---

subscript *Create subscripts*

Description

Create subscripts

Create superscript

Usage

```
subscript(x, subscript = seq(length(x)))

superscript(x, superscript = seq(length(x)))
```

Arguments

x	string
subscript	subscript
superscript	superscript

Value

text

string

Examples

```
subscript("X", 1:3)
superscript(c("A", "B"), 2)
```

`unbind` *unbind*

Description

Converts an object with k elements into a list of k objects

Usage

```
unbind(x, ...)
```

Arguments

`x`	object

Value

a list of objects, each of length 1

Index