

Package: gog (via r-universe)

October 15, 2026

Title Grammar of Graphics

Version 0.6.1

Description Draw a plot by describing it. You name a table, a mark to draw, and which column goes to which part of the picture. This specification uses the same words in R, 'Python', 'Julia' and 'JavaScript', and one engine, written in 'Rust', draws it as an SVG in all four. On macOS and Windows the engine comes with the package. On Linux the install compiles it, which needs 'Rust' 1.75 or newer.

License Apache License (== 2.0)

URL <https://github.com/psychometrician/gog>,
<https://psychometrician.github.io/gog-book/>

BugReports <https://github.com/psychometrician/gog/issues>

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.75)

Encoding UTF-8

Depends R (>= 4.3)

Roxygen list(markdown = TRUE)

Imports jsonlite, utils

Suggests DBI, RSQLite, gapminder, IRdisplay, knitr, pkgload, repr,
testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libclang-dev

Repository <https://psychometrician.r-universe.dev>

Date/Publication 2026-10-15 11:17:05 UTC

RemoteUrl <https://github.com/psychometrician/gog>

RemoteRef release

RemoteSha 06c61bc7cf434709b8f2e9251af0a7830814a024

RemoteSubdir r-pkg/gog

Contents

*.gog_atom	3
+.gog_spec	4
bin	4
bounds	5
box	5
brush	6
cluster	7
color	7
colour	9
confidence	9
cumulative	10
data	11
density	12
deviation	13
facet	13
facet-operators	14
flow	14
globe	15
gog_table	16
group	17
jitter	18
knit_print.gog_arrangement	19
knit_print.gog_spec	19
label	20
layout	21
map	22
marks	23
nest	24
network	25
opacity	26
order	27
palette	28
partition	28
pattern	29
play	30
polar	32
quantile	32
query	33
range	33
render_svg	34
repel	34
repr_html.gog_arrangement	35
repr_html.gog_spec	35
repr_text.gog_arrangement	36
repr_text.gog_spec	36
save_gif	37

`*.gog_atom` 3

save_json	37
save_svg	38
shape	39
size	40
smooth_band	41
space	41
stack	42
style	43
theme	45
title	46
transforms	47
x	48
x_label	50
y	50
y_label	51
z	52
z_label	53

Index 54

<code>*.gog_atom</code>	<i>Derive a compound layer atom from a mark and a transform.</i>
-------------------------	--

Description

Derive a compound layer atom from a mark and a transform.

Usage

```
## S3 method for class 'gog_atom'
e1 * e2

## S3 method for class 'gog_callable'
e1 * e2
```

Arguments

<code>e1</code>	The mark, or a layer already carrying one transform.
<code>e2</code>	The transform being joined to it.

Value

A layer `gog_atom` carrying the mark and its transforms.

<code>+.gog_spec</code>	<i>Compose plot atoms with +.</i>
-------------------------	-----------------------------------

Description

Compose plot atoms with +.

Usage

```
## S3 method for class 'gog_spec'
lhs + rhs
```

Arguments

lhs	The plot so far, or the first atom of a sentence.
rhs	The atom, layer or table being added to it.

Value

A gog_spec carrying both sides.

bin	<i>Bin a continuous variable into equal-width buckets (the histogram transform).</i>
-----	--

Description

Bin a continuous variable into equal-width buckets (the histogram transform).

Usage

```
bin(bins = NULL, width = NULL, tiling = NULL, ...)
```

Arguments

bins	Number of bins — one positive whole number. Positional: <code>bin(30)</code> .
width	Bin width in the data's own units, e.g. <code>bin(width = 5)</code> . Mutually exclusive with bins.
tiling	How the plane is partitioned when both axes are cut: "rect" (the default) or "hex". Two-dimensional bins only — zone * <code>bin(tiling = "hex")</code> .
...	Not used. Any argument here is refused with direction.

bounds	<i>Pre-computed bounds.</i>
--------	-----------------------------

Description

Pre-computed bounds.

Usage

bounds(lower, upper, start, end)

Arguments

lower, upper	Columns bounding the measure axis — the span every band mark draws. Required by ribbon/interval/line/step.
start, end	Columns bounding the domain axis. zone only: they give a rectangle its other two sides. Omit a pair and a zone spans the panel on that axis, which is the whole reason the mark exists.

box	<i>Box-and-whisker mark.</i>
-----	------------------------------

Description

Box-and-whisker mark.

Usage

box(whiskers = NULL)

Arguments

whiskers	"tukey" (default) runs the whiskers to the most extreme point within 1.5*IQR of the box and draws the points beyond as outliers – the standard box plot. "range" runs them to the true minimum and maximum with no outliers (the plain five-number summary).
----------	--

brush

Let the reader select rows, and push back the rest.

Description

brush puts a bound on one column's values. Rows inside it keep the plot's colors; rows outside it are dimmed, so a selection is read against what it was taken from. Where the page can run the engine, dragging moves the bound; on paper it stays where the sentence put it.

Usage

```
brush(field, at = NULL)
```

Arguments

field	Column the bound is read on, as a bare name.
at	Where the selection opens: two numbers on a column that measures, or the names to select on a column of categories. Left out, nothing is selected and the plot draws exactly as it would with no brush at all.

Details

A brush highlights. It never removes rows. Removing rows before the statistics run is what limits does, on the binding, and it counts what it dropped. The two are the same shape and different operations: change a domain and a histogram re-bins the survivors, brush it and the same bars stay, with the selected part standing out.

One column per brush. Write two for a rectangle: `brush(gdp, at = c(1200, 45000)) + brush(life, at = c(55, 78))`.

A mark can be brushed when one row is one shape: point, text, rule and zone. A line draws one shape through many rows, so there is no single row to select, and gog says so rather than guessing. A summarized layer is drawn whole, because a selection of twelve of a bar's forty rows has no honest picture.

Value

A gog_atom added to a plot with +.

cluster

Join the closest leaves, one pair at a time – the cluster tree.

Description

The leaves are the levels of the bound categorical position, and each one is described by a profile: its value at every level of the over column. The engine measures the distance between every pair of profiles, joins the closest two, and keeps joining until one tree holds them all. The other position stays unbound and carries the merge distance, which the axis names itself.

Usage

```
cluster(value, over)
```

Arguments

value	The numeric column distances are measured on, bare. Omit it on a tile plot, where <code>color()</code> already names the values.
over	The profile column, bare: each leaf holds one value at every level of it. Omit it only when the table has one row per leaf.

Details

Two marks read the one tree. `path * cluster(amount, over = nutrient) + x(food)` draws it: each join stands at the height its two branches merged, so the reader can see which leaves sit close and where the groups part. `y(food)` instead lays the same tree on its side. `zone * cluster(over = nutrient)` is the other reading: the tile plot unchanged, with the leaf axis's slots reordered to the tree's leaf order, so alike rows sit together. On a tile plot the values are already on color, which is why value is omitted there.

Composed with `|` and `/`, a clustered panel decides the order of any axis it shares, which is what puts a tree flush against a reordered tile plot.

The statistic is fixed: distances are Euclidean on the values as given, and branches join at their average distance. Rescale the value column where the data lives if its units should not weigh in.

color

Map fill/stroke color to a column.

Description

Map fill/stroke color to a column.

Usage

```
color(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
  decimals = NULL,
  short = NULL
)
```

Arguments

field	Column to bind (bare name). Text picks colors from the palette; a number runs along a sequential ramp.
scale	How the number becomes a position: "linear" (the default), "log", "time", or "category". A log scale leaves the data in its own units and labels the ticks 1, 10, 100 — it is not the same as plotting $\log(x)$, which would run the axis 0, 1, 2 and leave the reader to exponentiate.
base	Base of a "log" scale; 10 unless given. Use 2 for doublings, octaves or bits, and $\exp(1)$ for e-foldings. The base is very nearly cosmetic — every base draws the same picture, because the axis is normalized by its own range — so what it really chooses is where the gridlines fall and how they are labeled.
limits	For a continuous column, the two ends of the color scale, as <code>c(low, high)</code> . NA on either end leaves that end fitted to the data.
tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	Not taken: free fits an axis to each panel's own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	FALSE leaves this channel's legend out of the plot. The column is still mapped; only the legend that decodes it is not drawn, for a plot that says the same thing another way, such as names written at the ends of the lines.
axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	Not drawn yet on this channel: only <code>label()</code> prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only <code>label()</code> writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

colour	<i>The British spelling of color(), refused with direction.</i>
--------	---

Description

gog writes American English throughout and accepts no second spelling, which is Law 2 applied to the vocabulary itself: two ways to write one word is a silent letter, and the reader pays for it. ggplot2 accepts colour and color both, so a reader arriving from there types colour and, unexported, would meet R's "could not find function" — a message that names no fix. Exported for the same reason JavaScript still exports facet (spec §13): a word a reader will plausibly type earns a refusal that says what to write instead.

Usage

```
colour(...)
```

Arguments

... Ignored. The call is always refused.

confidence	<i>Confidence interval of the mean per group, for the interval mark.</i>
------------	--

Description

Confidence interval of the mean per group, for the interval mark.

Usage

```
confidence(level = NULL)
```

Arguments

level Confidence level, one number strictly between 0 and 1; default 0.95. Positional: confidence(0.99).

cumulative

*The running total of a measure, along its axis.***Description**

cumulative adds each position's value to the values before it, in the order the axis draws its positions: left to right on a number, in the categories' own order on a category. It adds up what the layer already measures – the column named on y, or what a transform before it made – and every mark draws the total as it draws any measure. The total is the column cumulative, while the measure's own column keeps each row's value, so `label(count)` and `label(cumulative)` print the two.

Usage

```
cumulative(share = NULL)
```

Arguments

share TRUE divides each group's total by its last, so every curve ends at 1: the empirical distribution function of each group. Defaults to FALSE.

Details

Each color or group split runs its own total. A category axis is added up in its declared order, so sort it in the table rather than with `order()`.

Value

A `gog_atom` joined to a mark with `*`.

See Also

[transforms](#) for the measures it adds up; [stack\(\)](#), which adds up the groups at one position instead of the positions of one group.

Examples

```
# Each year's sales added to the years before it.
sales <- data.frame(year = 2019:2023, sales = c(120, 135, 128, 150, 162))
r <- data(sales) + line * cumulative + x(year) + y(sales)

# The cumulative histogram, and one distribution function per species.
h <- data(iris) + step * bin * cumulative + x(Sepal.Length)
e <- data(iris) + step * bin * cumulative(share = TRUE) + x(Sepal.Length) +
  color(Species)
```

data	<i>Start a plot with a data frame.</i>
------	--

Description

Every plot begins by naming its table. The columns are then written as bare names in the channels that follow, and the nearest table wins where a plot has more than one.

Arguments

df	An R data frame.
name	Optional table name (auto-derived from the variable name if omitted).

Details

Masks `utils::data()`, R's dataset loader. Use `utils::data()` to load a built-in dataset; this one binds a table you already have. Both readings of `data(mtcars)` are well formed, so that one case cannot be told apart and is the reason the collision is documented rather than caught.

Value

A `gog_spec` – the start of a sentence, extended with `+`.

Usage

```
data(df, name = NULL)
```

The Usage section is written out here rather than generated, because R CMD check reads a generated `\usage\{data(...)\}` as base R's dataset loader and then reports `df` as a missing dataset. Suppressing the generated one is the documented way out; the signature is not lost, only moved.

Examples

```
p <- data(mtcars) + point + x(wt) + y(mpg)

# Naming the table explicitly, which matters when a plot has more than one.
q <- data(mtcars, name = "cars") + point + x(wt) + y(mpg)
```

density	<i>Estimate the smooth probability density of a continuous variable (a KDE).</i>
---------	--

Description

Estimate the smooth probability density of a continuous variable (a KDE).

Usage

```
density(
  adjust = NULL,
  bandwidth = NULL,
  levels = NULL,
  compare = NULL,
  reach = NULL
)
```

Arguments

adjust	Multiplier on the automatically-chosen bandwidth: <code>density(2)</code> is twice as smooth, <code>density(0.5)</code> rougher. Positional: <code>density(2)</code> .
bandwidth	Absolute bandwidth in the data's own units, e.g. <code>density(bandwidth = 5)</code> . Mutually exclusive with <code>adjust</code> , and meaningless on the two-dimensional reading, where the two axes carry different quantities — refused there toward <code>adjust</code> .
levels	How many levels to cut a field into: <code>path * density(levels = 8)</code> traces their boundaries, <code>zone * density(levels = 8)</code> fills between them. Refused on a <i>density curve</i> , which is one line with nothing to cut into.
compare	What a violin 's width means from one slot to the next: "count" (the default) scales each violin by its group's row count, so a thin violin is a small group and the widths mean one thing across the panel; "shape" draws every violin to the same area, comparing shapes and nothing else. Meaningful only in the violin reading (<code>ribbon * density + x(<category>) + y(<number>)</code>), and refused on the curve and the field, which have no slots to compare.
reach	How far each violin reaches from the line its category sits on, in slots : 0.4 (the default) keeps every shape inside its own slot — two facing violins fill four fifths of the space between their categories, a categorical bar's own rule — and past 0.5 they run into their neighbors, which is the ridgeline plot (<code>area * density(reach = 2.5) + x(<number>)</code>). Measured one way, so a ribbon reaches it on each side and an area on one — which keeps the half violin exactly half of the violin at any value. Violin-only, like <code>compare</code> .

deviation	<i>Spread band per group, mean +/- k standard deviations.</i>
-----------	---

Description

Spread band per group, mean +/- k standard deviations.

Usage

```
deviation(multiplier = NULL)
```

Arguments

multiplier	How many standard deviations reach each side of the mean; default 1. Positional: deviation(2).
------------	--

facet	<i>Name the column that splits the plot into panels — small multiples.</i>
-------	--

Description

facet() joins the plot with the facet operators, not +:

Usage

```
facet(field, wrap = NULL)
```

Arguments

field	Column to facet by (bare name).
wrap	Panels to draw before the line of them turns. NULL (the default) leaves the panels in one line.

Details

plot facet(cyl)	one panel per category, side by side
plot / facet(drv)	one panel per category, stacked
plot / facet(drv) facet(cyl)	crossed into a grid (rows x columns)

The panels share one scale — that is what makes them comparable — and any transform in the plot runs within each panel's rows. The facet column must be a category column: a number is a position along an axis, not a name for a panel.

wrap folds a long line of panels into a rectangle: twelve countries side by side are twelve slivers, and wrap = 4 makes them a 4 x 3 grid. The number is **how many panels before the line turns**, and which way the line runs is the operator's to say, not wrap's:

```
plot | facet(country, wrap = 4) levels run across, 4 to a row
plot / facet(country, wrap = 4) levels run down, 4 to a column
```

That is one number where ggplot2 has `nrow` and `ncol`, and it cannot contradict itself. Wrapping a *crossed* facet is refused: a crossing already fills a rectangle.

facet-operators

Facet a plot into panel columns, or set a plot beside another.

Description

| facets into panel columns and sets plots side by side; / facets into panel rows and sets them one above the other.

Usage

```
## S3 method for class 'gog_spec'
e1 | e2

## S3 method for class 'gog_spec'
e1 / e2
```

Arguments

e1, e2 The plot on the left, and the facet or plot joined to it.

Value

A faceted `gog_spec`, or a `gog_arrangement` when two plots are composed.

flow

Lay a magnitude through its stages — the flow diagram.

Description

The stages arrive as **columns**, in reading order: one row of the table is one path through all of them, and `flow(class, sex, survived)` runs each row from its class to its survived. Rows sharing a path add together, which quietly sets aside any column the atom did not name.

Usage

```
flow(..., shared = FALSE, layered = FALSE)
```

Arguments

...	The stage columns, bare names, in reading order. At least two: one column has no between.
shared	TRUE when the two columns name one set of places, the place each amount leaves and the place it arrives at, laid on one axis. FALSE, the default, runs the columns as stages side by side.
layered	TRUE when each row is a link between two places, and the places stand in layers worked out from the links: the Sankey diagram. Not together with shared = TRUE.

Details

Three marks read the one layout. `ribbon * flow(...)` draws the **bands**, a path's magnitude carried between adjacent stages; `zone * flow(...)` draws each stage's **slots**, stacked with no padding so the measure axis stays honest; `text * flow(...) + label(name)` names each slot where it sits. Layer any two or all three.

What each path is weighed by rides on `y: + y(count)` weighs that column, and binding nothing at all makes every path weigh 1. Bound to `x` instead, the count runs the stages from top to bottom. On the band layer, `color(<stage>)` colors every band by the category its path holds at that stage, and `color()` of any other categorical column splits each band by its values; the slots take their paint from `style()`.

The stage axis is drawn from the atom's own columns, so there is nothing for `x()` to say — to reorder the stages, reorder the arguments.

Two columns that name one set of places take `shared = TRUE`: `flow(exporter, importer, shared = TRUE)` reads each row as an amount leaving one place and arriving at another, gives each place one slot whichever column names it, and lays every slot on the one count axis. Flat, that draws the arc diagram, each band arching from one place to the other; in `polar()` the count goes around the circle and it draws the chord diagram. Each slot is in two parts with no line between them: the ends where the place is named in the first column, in the slot's color, then the ends where it is named in the second, a shade darker. So `flow(importer, exporter, shared = TRUE)` shades the exports instead of the imports. The panel is white and has no frame unless `theme()` says otherwise, since the plot has no numbers to read.

A list of links whose places stand in layers takes `layered = TRUE`: `flow(source, target, layered = TRUE)` reads each row as an amount moving from one place to another, and puts each place in the first layer after every place that sends to it. That draws the Sankey diagram: a band may skip a layer, and a flow may stop before the last layer. Each layer is sorted by where its places' partners are, the places of a layer stand a gap apart, and each name stands beside its slot. Links that form a cycle have no layers and are refused.

Description

The globe coordinate space. x is **longitude** and y is **latitude**, both in degrees, and each mark stands at its place on the sphere's surface. The marks that draw on a `map()` draw here: `point` is a place, `path` is a route bending along great circles, `text` is a name at a place, `rule` holds one coordinate whole (a meridian pole to pole, or a parallel all the way round), and `zone` with `group()` fills each region of a boundary. The graticule is the panel grid, turned off like any other with `theme(grid = "none")`.

Usage

```
globe(turn = 0, tilt = 0)
```

Arguments

<code>turn</code>	Degrees of longitude the view faces. Default 0.
<code>tilt</code>	Degrees of latitude the view faces, -90 to 90. Default 0.

Details

A globe shows one half of the earth. `turn` and `tilt` name the place the view faces, in the words `space()` already uses: `globe(turn = 178, tilt = -18)` puts Fiji at the center of the disk. A bearing wraps, so any `turn` is a view; `tilt` is a latitude and stops at the poles. Rows on the far half are hidden behind the sphere, and `gog` says how many rather than dropping them in silence. In the web edition, dragging the globe turns it.

Both positions are spent on the place, so a mark that measures along an axis has none left; carry a quantity on `size()` or `color()` instead, as on a map. A binned field on the sphere is designed and not drawn yet: its correct tiling is hexagonal, because rectangular bins do not cover equal area on a sphere, and that equal-area grid is not built.

Examples

```
## Not run:
data(quakes_fiji) + point + x(east) + y(north) + size(magnitude) +
  globe(turn = 178, tilt = -18)
data(borders) + zone + x(lon) + y(lat) + group(country) +
  color(continent) + globe(turn = 20, tilt = 5)

## End(Not run)
```

gog_table

Read one of the book's example tables

Description

Fetches a table published beside the manual and returns it ready to plot. The full list of names is in the book's data chapter.

Usage

```
gog_table(name, text = character())
```

Arguments

name	The table's name without the extension, such as "gapminder_2007".
text	Columns that must stay text. A CSV records what a value is and never what kind of thing it is, so a column of 01, 02, 03 comes back as the numbers 1, 2, 3 unless it is named here.

Value

A data frame.

Examples

```
## Not run:
gapminder_2007 <- gog_table("gapminder_2007")
data(gapminder_2007) + point + x(gdp) + y(life)

# `session` holds labels that look like numbers, so it stays text.
sessions <- gog_table("sessions", text = "session")

## End(Not run)
```

group	<i>Group line/path marks by a column (connects points within the same group). Use color(field) when you also want color distinction — it implies grouping.</i>
-------	--

Description

Group line/path marks by a column (connects points within the same group). Use color(field) when you also want color distinction — it implies grouping.

Usage

```
group(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
```

```
    decimals = NULL,  
    short = NULL  
)
```

Arguments

field	Column to group by, as a bare name.
scale, base, limits	Not taken: this channel distinguishes categories rather than measuring them, so there is no scale or range along it. gog refuses each of them here.
tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	Not taken: free fits an axis to each panel’s own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	Not taken: group splits without encoding anything, so it draws no legend. gog refuses it here and names the five channels that do.
axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	Not drawn yet on this channel: only label() prints a column’s numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only label() writes a column’s numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

Value

A gog_atom added to a plot with +.

jitter	<i>Spread a strip plot’s coincident points apart, along the categorical axis only.</i>
--------	--

Description

Spread a strip plot’s coincident points apart, along the categorical axis only.

Usage

```
jitter(amount = NULL)
```

Arguments

amount Spread as a multiple of the default, one number from 0 to 1.25; default 1. Positional: `jitter(0.5)` for half the spread. At 1.25 a point reaches the edge of its category's slot, and more is refused.

`knit_print.gog_arrangement`

knitr print method for an arrangement.

Description

knitr print method for an arrangement.

Usage

`knit_print.gog_arrangement(x, ...)`

Arguments

x The plot or arrangement being displayed.
 ... Passed on by the display host; unused.

`knit_print.gog_spec` *knitr print method: the plot inline as SVG in HTML, and as a converted*

Description

knitr print method: the plot inline as SVG in HTML, and as a converted

Usage

`knit_print.gog_spec(x, ...)`

Arguments

x The plot or arrangement being displayed.
 ... Passed on by the display host; unused. vector figure on a LaTeX page. Called automatically by knitr when a `gog_spec` is the last expression in a code chunk.

label	<i>Draw a column's values as text labels — the text mark's content.</i>
-------	---

Description

label supplies the string a text mark draws, the way x and y supply its position: `text + x(a) + y(b) + label(name)`. A string column is drawn as-is, a numeric one is formatted. It is text's required channel — no other mark accepts it, so a labeled scatter is the superposition point + text. A scale is refused: a label is content, like shape, not a magnitude to run a scale along.

Usage

```
label(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
  decimals = NULL,
  short = NULL
)
```

Arguments

field	Column whose values are drawn as the text, as a bare name.
scale, base, limits	Not taken: this channel distinguishes categories rather than measuring them, so there is no scale or range along it. gog refuses each of them here.
tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	Not taken: free fits an axis to each panel's own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	Not taken: a label is read where it is written, so it draws no legend. gog refuses it here and names the five channels that do.
axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	How many decimal places each number is printed with, a whole number from 0 to 10: <code>label(life, decimals = 1)</code> prints 70.72848 as 70.7. The places are fixed, so 77 prints as 77.0 and a column of labels reads as one column of figures. A column of text has nothing to round.

short TRUE writes a large number short, with K for thousands, M for millions, B for billions and T for trillions: `label(population, short = TRUE)` prints 3811953827 as 3.81B. It keeps three significant digits unless `decimals` gives the places after the point, so `decimals = 1` prints 3.8B. A column of text has no number to write short.

Value

A `gog_atom` added to a plot with `+`.

layout	<i>Place a graph – node positions computed from an edge table.</i>
--------	--

Description

The two columns name each row’s endpoints, and one row of the table is one edge between its two values. The nodes derive: their names from the columns’ union, their degree by counting the rows that name them. The engine computes every position, the same way in every binding, so one table is one picture wherever it is drawn.

Usage

`layout(from, to)`

Arguments

from The column naming each edge’s first endpoint, bare.
to The column naming each edge’s second endpoint, bare.

Details

Three marks read the one placement, inside `network()`. `edge * layout(from, to)` draws the connections, `point * layout(...)` the nodes, and `text * layout(...) + label(name)` names them. On the node layers, `size(degree)` and `color(degree)` read that count.

The positions mean nothing as quantities – the picture can be mirrored without changing what the graph says – so nothing binds to `x`, `y` or `z`, and the third dimension, when you want it, is the space’s: `network(turn = 30, tilt = 25)`.

map

Flatten the sphere onto the page.

Description

The cartographic coordinate space. *x* is **longitude** and *y* is **latitude**, both in degrees, and the projection decides where each place lands. Nothing else about the sentence changes: *point* is a place, *path* is a route through places in the table's order, *text* is a name at a place, and *rule* spans the axis it does not name, which on a map is a meridian or a parallel.

Usage

```
map(preserve = "area")
```

Arguments

preserve What the flattening keeps: "area" or "angle".

Details

Both positions are spent on the place, so a mark that measures along an axis has none left. Carry a quantity on a channel instead. `size(<column>)` gives the proportional-symbol map and `color(<column>)` shades each place, which is what cartography does once the two axes are gone.

A sphere cannot be laid flat without giving something up, and area and angle cannot both survive. *preserve* names which one does:

- "area" – every region gets ink in proportion to its true size. The default, because a map is usually read by area: a projection that inflates Greenland says something false about the number inside it. Uses the Equal Earth projection, which reaches both poles.
- "angle" – every small shape keeps its true form, and area is what pays. Uses Mercator, so Greenland arrives the size of Africa while being fourteen times smaller. Mercator sends the poles infinitely far away, so it stops at 85.05 degrees and says how many rows it could not reach.

The panel is shaped by the projection rather than by the page, so `theme(ratio = )` is refused here: an equal-area map stretched to fit a box is no longer equal-area, though it still looks like a map.

Examples

```
## Not run:
data(quakes_fiji) + point + x(east) + y(north) + size(magnitude) + map()
data(borders) + path + x(lon) + y(lat) + group(country) + map()
data(borders) + path + x(lon) + y(lat) + group(country) +
  map(preserve = "angle")

## End(Not run)
```

marks

Marks – the shapes a plot draws

Description

A **mark** is the shape a layer draws, and every plot needs exactly one. A mark carries no data of its own: it is combined with **channels** that say which column goes to which part of the picture, and the engine decides every stroke from there.

Usage

point

line

path

rule

zone

area

bar

step

interval

ribbon

text

surface

edge

Format

Each is a `gog_atom` object, used bare rather than called.

Details

Marks are the grammar's consonants. They are plain objects rather than functions, so they are written bare in a sentence:

```
data(table) + point + x(column) + y(column)
```

A mark and its required positions together are the smallest thing that draws; neither renders alone. Which channels a mark accepts is a property of the mark, and asking for one it cannot carry is refused with a direction rather than silently ignored.

The marks

point One glyph per row. How two measurements move together.

line Connects rows in **x order**. How one value changed over time.

path Connects rows in **data order** rather than x order, so a series may double back on itself.

area Fills the region between the line and the baseline.

bar A rectangle from the baseline to the value. How large each category is.

step The line family's right-angled member: a value holds steady until it jumps.

interval A segment from a low value to a high one. Needs a pair, so it is usually written with range or confidence.

ribbon A band between two values across a span.

text Draws the string given by `label` at each position.

rule A reference line across the panel – the threshold everything else is compared against.

zone Shades a rectangular region of the panel.

surface A sheet over a grid of samples, for three-dimensional data.

box is a mark too, but takes an argument, so it is documented separately.

See Also

`box()` for the box-and-whisker mark; `x()` and `y()` for positions; `style()` for settings that do not vary with the data.

Examples

```
# A mark plus its positions is the smallest sentence that draws.
p <- data(mtcars) + point + x(wt) + y(mpg)

# The same data, a different shape: one atom is all that changes.
q <- data(mtcars) + line + x(wt) + y(mpg)
```

nest

Pack the panel with nested regions.

Description

The nested coordinate space, and the treemap is what it draws. Every row's measure becomes an **area** rather than a length, and the areas partition the panel: a region's share of the picture is its share of the total.

Usage

```
nest()
```

Details

Nothing else about the sentence changes, which is what makes this a space and not a chart type. `bar * sum + y(revenue) + color(region)` is a stacked column flat, a pie in `polar()`, and a treemap here — three answers to the one question of what carries a share, and the words are the same in all three.

A packed panel has **no axes**: its two directions carry no variable, and two neighboring regions are not near each other in the data. So there is nothing to label and nothing to tick, and the color legend is what decodes the regions. Bind a position to split the packing one level further — each category gets a region of its own and its rows are packed inside it.

`dodge`, `stack` and `jitter` are refused here: they decide what happens when two marks land in the same place, and in a packing none can.

Examples

```
## Not run:
data(sales) + bar * sum + y(revenue) + color(region) + nest()
data(sales) + bar * sum + x(region) + y(revenue) + color(product) + nest()

## End(Not run)
```

network

The graph-theoretic space – where a `layout()`'s placement is drawn.

Description

A layout's positions are the graph's, not the data's, so this space draws no axes, no ticks and no grid: there is nothing there for them to number. It is the one space a layout draws in, and a layout is the one thing that stands in it.

Usage

```
network(turn = NULL, tilt = NULL)
```

Arguments

<code>turn</code>	Degrees around the upright axis, stated only for the cube.
<code>tilt</code>	Degrees of elevation, stated only for the cube.

Details

Stating a viewing angle states the third dimension. Bare `network()` draws the graph flat; `network(turn = 30, tilt = 25)` computes the placement in a cube and draws it from that angle, with each unstated angle taking the standard view. In the web edition the cube turns with a drag, the way the cube in `space()` turns.

opacity	<i>Map opacity to a numeric column.</i>
---------	---

Description

Map opacity to a numeric column.

Usage

```
opacity(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
  decimals = NULL,
  short = NULL
)
```

Arguments

field	Column to bind (bare name).
scale	How the number becomes a position: "linear" (the default), "log", "time", or "category". A log scale leaves the data in its own units and labels the ticks 1, 10, 100 — it is not the same as plotting $\log(x)$, which would run the axis 0, 1, 2 and leave the reader to exponentiate.
base	Base of a "log" scale; 10 unless given. Use 2 for doublings, octaves or bits, and $\exp(1)$ for e-foldings. The base is very nearly cosmetic — every base draws the same picture, because the axis is normalized by its own range — so what it really chooses is where the gridlines fall and how they are labeled.
limits	The two ends of the scale, as <code>c(low, high)</code> . NA on either end leaves that end fitted to the data.
tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	Not taken: free fits an axis to each panel's own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	FALSE leaves this channel's legend out of the plot. The column is still mapped; only the legend that decodes it is not drawn, for a plot that says the same thing another way, such as names written at the ends of the lines.

axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	Not drawn yet on this channel: only <code>label()</code> prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only <code>label()</code> writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

order	<i>Order the categorical axis by a column.</i>
-------	--

Description

Every atom that takes a column is a noun naming a property, and its argument is the column that drives it — `color(species)`, `size(population)`, `order(gold)`. This one was called `sort_by` until it became the last survivor of a `_by` convention that `color_by` → `color` had already retired: the suffix marked the argument as a key, which says nothing when every atom in the group takes a key.

Usage

```
order(field, desc = FALSE)
```

Arguments

field	Column to order by (bare name). Use a numeric column (e.g. <code>y</code>) to order by value, which overrides a factor's levels. Use the category column itself to ask for that column's own order: alphabetical for plain text, and the declared levels for a factor.
desc	Logical. TRUE = descending (largest first / $Z \rightarrow A$).

Details

Masks `base::order`, as `data`, `title`, and `palette` mask theirs. Use `base::order()` for the base function.

Examples

```
## Not run:
data(medals) + bar + x(gold) + y(country) + order(gold, desc = TRUE)

## End(Not run)
```

palette	<i>Set the categorical color palette.</i>
---------	---

Description

Set the categorical color palette.

Usage

```
palette(pal)
```

Arguments

pal	Either a palette name ("gog", "okabe") or a character vector of hex colors. The palette is used in first-appearance order; if there are more categories than palette entries, additional colors are generated automatically via the HSL color wheel so no two categories share a color. Name the colors to bind each one to a level instead, as in <code>palette(c(Asia = "tomato", Europe = "steelblue"))</code> : a level then keeps its color whatever order the rows arrive in. Every name must be a level of the column mapped to color, and every level drawn must have a name.
-----	---

partition	<i>Divide a whole among nested parts — one ring per level of a hierarchy.</i>
-----------	---

Description

The hierarchy arrives as **columns**, outermost first: one row of the table is one leaf, and `partition(group, item, detail)` says which columns spell the path down to it. A blank level ends that branch early, which is what gives a real hierarchy its ragged rim.

Usage

```
partition(..., cross = FALSE)
```

Arguments

...	The hierarchy's columns, bare names, outermost first.
cross	Cross the levels rather than nesting them down one axis — <code>partition(decade, theme, cross = TRUE)</code> is the mosaic. Must be written out in full; it sits after ..., so R will not match it by prefix.

Details

zone * partition(...) flat is the **icicle**; the same sentence + polar() is the **sunburst**, and that one atom is the whole difference. text * partition(...) + label(name) names each node where it sits, reading the center the same computation published.

What each branch is weighed by rides on x: + x(amount) apportions that column, and binding nothing at all makes every leaf weigh 1. Compose with proportion for shares. y(depth, limits = c(0, 4)) puts the hole in the middle, depth being the ring the transform synthesizes.

Every number belongs to a **leaf**. An interior node carrying one of its own is ambiguous — is its arc the children's total, or its own beside them? — so it is refused rather than guessed.

cross = TRUE turns the levels across each other instead of down one axis: the first divides the width, the second divides the height *within* each of those columns. That is the **mosaic**, and because both directions are then spent on the hierarchy there is no ring left to step and only the leaves are drawn.

pattern

Map paint texture to a categorical column — shape's twin.

Description

pattern tells categories apart by *texture* rather than hue: on a fill (bar/box/area/ribbon) each category draws as a hatch, on a stroke (line/step/interval) as a dash. It is the color-free way to separate series, so it survives grayscale printing and color-blindness; pairing it with color on the same column is the redundant encoding that is the accessibility best practice. A scale is refused: like shape it answers "which one?", not "how much?".

Usage

```
pattern(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
  decimals = NULL,
  short = NULL
)
```

Arguments

field Column to bind to the texture, as a bare name.

scale, base, limits

Not taken: this channel distinguishes categories rather than measuring them, so there is no scale or range along it. gog refuses each of them here.

tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	Not taken: free fits an axis to each panel's own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	FALSE leaves this channel's legend out of the plot. The column is still mapped; only the legend that decodes it is not drawn, for a plot that says the same thing another way, such as names written at the ends of the lines.
axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	Not drawn yet on this channel: only label() prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only label() writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

Details

Distinct from the style(pattern =) *setting*, which fixes one texture for the whole layer; pattern() *maps* a column to several.

Value

A gog_atom added to a plot with +.

play	<i>Cut the plot into frames and play them — the time dimension.</i>
------	---

Description

play is facet read in time. Both split the rows by a column's distinct values; | facet(continent) lays the pieces out across the page and play(year) lays them out in sequence, so point + x(gdp) + y(life) + play(year) is the time-lapse of the same data the faceted grid shows all at once.

Usage

```
play(  
  field,  
  speed = NULL,  
  scale = NULL,  
  base = NULL,  
  limits = NULL,  
  tick_count = NULL,
```

```

    free = FALSE,
    legend = NULL,
    axis = NULL,
    decimals = NULL,
    short = NULL
)

```

Arguments

field	Column whose distinct values become the frames, as a bare name.
speed	A multiple of the normal pace: 2 is twice as fast.
scale, base, limits	Not taken: this channel distinguishes categories rather than measuring them, so there is no scale or range along it. gog refuses each of them here.
tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.
free	Not taken: free fits an axis to each panel's own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	Not taken: each frame is named in the strip above the panel as it is shown, so play draws no legend. gog refuses it here and names the five channels that do.
axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	Not drawn yet on this channel: only label() prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only label() writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

Details

Every scale, the color map and every legend are fitted across the whole sequence, never per frame — a scale that re-fitted would move the axis under the data and make a still point look like a moving one. A layer that does not bind play is drawn in every frame, so a reference line stands still behind the marks that move.

Unlike facet, a **number** is welcome: panels compete for page area, where a hundred of them are unreadable at any size, but frames compete for time, where a hundred is a longer loop rather than a smaller picture.

speed is how many times faster than normal the frames run — 2 is twice as fast, 0.5 half. Where a static image is made from the plot (the PDF of this book, or any SVG converter) the first frame is what it shows.

Value

A gog_atom added to a plot with +.

polar	<i>Bend the plane into a circle.</i>
-------	--------------------------------------

Description

The polar coordinate space. x becomes the **angle** and y the **radius**, so the axis a bar chart stands its categories on is the axis a rose wraps around the circle. Nothing else about the sentence changes: `bar * count + x(month)` is a bar chart flat and a rose in polar, from the same words.

Usage

```
polar(start = 0)
```

Arguments

start	Degrees clockwise from the top where the circle begins. 0 (the default) starts the axis at twelve o'clock; -22.5 centers the first of eight categories there instead of starting it there.
-------	--

Details

The angular axis is *periodic* — one turn spans exactly the fitted range of x , so the circle closes with no seam. Angles run clockwise from twelve o'clock.

Examples

```
## Not run:
data(winds) + bar * count + x(direction) + polar()
data(winds) + bar * count + x(direction) + polar(start = -22.5)

## End(Not run)
```

quantile	<i>The p-th quantile of y per group.</i>
----------	--

Description

The p -th quantile of y per group.

Usage

```
quantile(p = NULL)
```

Arguments

p	The quantile probability, one number between 0 and 1. Positional: <code>quantile(0.9)</code> is the 90th percentile. There is no default; at 0, 0.5 and 1 the plot draws and says that min, median and max are the plain names for the same numbers.
---	--

Arguments

- low, high The band's two ends, as quantile probabilities between 0 and 1.
1. Positional: `range(0.25, 0.75)` is the middle half. An unset end is that side's extreme, so bare `range` is the whole group and `range(high = 0.9)` runs from the minimum to the 90th percentile.

<code>render_svg</code>	<i>Render a <code>gog_spec</code> to an SVG string.</i>
-------------------------	---

Description

Render a `gog_spec` to an SVG string.

Usage

```
render_svg(gog)
```

Arguments

`gog` A `gog_spec` object built with `data()` + ...

Value

A character string containing the SVG.

<code>repel</code>	<i>Move overlapping labels apart until they can be read.</i>
--------------------	--

Description

`text *repel` reads where the labels and the points actually sit and pushes each label a different way until they stop overlapping, drawing a thin leader line back to the point when a label has moved clear of it. Where there is no arrangement that fits, every label is still drawn and the plot reports how many are still crowded.

Usage

```
repel
```

Value

A `gog_atom` joined to a mark with `*`.

See Also

`style()`'s nudge, the constant offset this is the data-derived counterpart to; `jitter()`, the same idea for a scatter's points.

Examples

```
## Not run:
# Every country named, with the names moved off one another.
gapminder_2007 <- gog_table("gapminder_2007")
p <- data(gapminder_2007) + point + text * repel +
  x(gdp) + y(life) + label(country)

## End(Not run)
```

```
repr_html.gog_arrangement
```

repr method: an arrangement as inline SVG in a Jupyter cell.

Description

repr method: an arrangement as inline SVG in a Jupyter cell.

Usage

```
repr_html.gog_arrangement(obj, ...)
```

Arguments

<code>obj</code>	The plot or arrangement being displayed.
<code>...</code>	Passed on by the display host; unused.

```
repr_html.gog_spec
```

repr method: the plot as inline SVG in a Jupyter cell.

Description

repr method: the plot as inline SVG in a Jupyter cell.

Usage

```
repr_html.gog_spec(obj, ...)
```

Arguments

<code>obj</code>	The plot or arrangement being displayed.
<code>...</code>	Passed on by the display host; unused.

```
repr_text.gog_arrangement
```

repr method: a one-line description of an arrangement.

Description

repr method: a one-line description of an arrangement.

Usage

```
repr_text.gog_arrangement(obj, ...)
```

Arguments

obj	The plot or arrangement being displayed.
...	Passed on by the display host; unused.

```
repr_text.gog_spec
```

repr method: a one-line description for the text/plain fallback.

Description

repr method: a one-line description for the text/plain fallback.

Usage

```
repr_text.gog_spec(obj, ...)
```

Arguments

obj	The plot or arrangement being displayed.
...	Passed on by the display host; unused.

save_gif	<i>Write a played plot to an animated GIF.</i>
----------	--

Description

A plot that binds `play()` moves in a browser, because the SVG carries its own timing. Most other places do not read that: a message to a friend, a slide, a post. This writes the same sequence as a GIF, which they do read.

Usage

```
save_gif(gog, path, scale = 1)
```

Arguments

<code>gog</code>	A played <code>gog_spec</code> — one that binds <code>play()</code> .
<code>path</code>	Where to write, ending in <code>.gif</code> .
<code>scale</code>	Multiplier on the plot's canvas. A plot is 800 by 600 unless its theme says otherwise, which is small for a post; <code>scale = 2</code> doubles it.

Details

The frames come out of the one renderer, so the file cannot disagree with the plot. Every scale, the color map and each legend are fitted across the whole sequence at once, and the frames are cut from that single drawing rather than drawn again one at a time.

Nothing needs to be installed. The engine converts and encodes on its own.

Value

The path, invisibly.

save_json	<i>Save what the engine reads to draw a plot, as a JSON file.</i>
-----------	---

Description

Every binding turns a plot into one request, the sentence and its tables, and the engine draws from that request alone. This writes the request to a file. Handed to the engine, the file draws the same plot byte for byte, in any language and on any machine, so it is the file to attach when a plot draws wrong.

Usage

```
save_json(gog, path)
```

Arguments

gog	A gog_spec, or an arrangement of them.
path	Where to write, ending in .json.

Details

Only the columns the sentence names are written, but every value in each of them is, including values the drawing does not show. Read the file before you share it.

The file is written even when the engine refuses the plot, because a refusal you believe is wrong is worth reporting too.

Value

The path, invisibly.

save_svg	<i>Write a plot's SVG to a file.</i>
----------	--------------------------------------

Description

The drawing `render_svg()` returns, written to path byte for byte, so the file is the same in every binding and on every platform.

Usage

```
save_svg(gog, path)
```

Arguments

gog	A gog_spec, or an arrangement of them.
path	Where to write, ending in .svg.

Details

The plot is drawn first and written second, so a plot gog refuses leaves a file already at path as it was.

Value

The path, invisibly.

shape

Map glyph shape to a categorical column.

Description

A scale is refused: shape answers "which one?", and there is no distance between circle and square for a scale to run along.

Usage

```
shape(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
  decimals = NULL,
  short = NULL
)
```

Arguments

field	Column to bind to the glyph shape, as a bare name.
scale, base, limits	Not taken: this channel distinguishes categories rather than measuring them, so there is no scale or range along it. gog refuses each of them here.
tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	Not taken: free fits an axis to each panel's own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	FALSE leaves this channel's legend out of the plot. The column is still mapped; only the legend that decodes it is not drawn, for a plot that says the same thing another way, such as names written at the ends of the lines.
axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	Not drawn yet on this channel: only label() prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

short Not drawn yet on this channel: only label() writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

Value

A gog_atom added to a plot with +.

size	<i>Map point radius to a numeric column.</i>
------	--

Description

Map point radius to a numeric column.

Usage

```
size(  
  field,  
  scale = NULL,  
  base = NULL,  
  limits = NULL,  
  tick_count = NULL,  
  speed = NULL,  
  free = FALSE,  
  legend = NULL,  
  axis = NULL,  
  decimals = NULL,  
  short = NULL  
)
```

Arguments

field	Column to bind (bare name).
scale	How the number becomes a position: "linear" (the default), "log", "time", or "category". A log scale leaves the data in its own units and labels the ticks 1, 10, 100 — it is not the same as plotting log(x), which would run the axis 0, 1, 2 and leave the reader to exponentiate.
base	Base of a "log" scale; 10 unless given. Use 2 for doublings, octaves or bits, and exp(1) for e-foldings. The base is very nearly cosmetic — every base draws the same picture, because the axis is normalized by its own range — so what it really chooses is where the gridlines fall and how they are labeled.
limits	The two ends of the size scale, as c(low, high). NA on either end leaves that end fitted to the data.
tick_count	Not taken: only a position, x, y or z, has an axis with ticks to count. gog refuses it here.

speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	Not taken: free fits an axis to each panel's own rows, and only a position, x, y or z, has an axis. gog refuses it here and says to write it on a position.
legend	FALSE leaves this channel's legend out of the plot. The column is still mapped; only the legend that decodes it is not drawn, for a plot that says the same thing another way, such as names written at the ends of the lines.
axis	Not taken: only a position, x, y or z, draws an axis. gog refuses it here and names the three positions.
decimals	Not drawn yet on this channel: only label() prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only label() writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

smooth_band	<i>The band around a smooth line: the fit's confidence band.</i>
-------------	--

Description

smooth's pair form, as confidence is mean's. At each point of the LOESS curve it gives the fit's low and high ends, so ribbon * smooth_band + line * smooth draws the trend with its band.

Usage

```
smooth_band(level = NULL)
```

Arguments

level	Confidence level, one number strictly between 0 and 1; default 0.95. Positional: smooth_band(0.99).
-------	---

space	<i>View a 3-D plot from an angle.</i>
-------	---------------------------------------

Description

Binding z is what makes a plot three-dimensional — one more vowel, not a chart type. space() sets the angle that plot is *viewed* from. That angle is a view parameter of the coordinate space — the way a polar plot needs a start angle — not a channel (it maps no column) and not a mark. Omit it and a 3-D plot takes a default three-quarter view that shows all three axes.

Usage

```
space(turn = 30, tilt = 25)
```

Arguments

turn	Degrees swung around the upright axis — which side you view the scene from. Default 30.
tilt	Degrees the eye is lifted above the floor — how steeply you look down. Default 25.

Examples

```
## Not run:
data(clusters) + point + x(a) + y(b) + z(c) +
  space(turn = -45, tilt = 30)

## End(Not run)
```

stack

Resolving overlap – stack, dodge and jitter

Description

When several marks land in the same place, something has to decide where they go. These are **collision modifiers**: they do not compute a statistic, they move marks that would otherwise cover each other. Like transforms they ride the * slot, so they compose with a statistic rather than replacing it – bar * count * stack bins, counts, and then piles the result.

Usage

```
dodge
```

```
stack(share = NULL, baseline = NULL)
```

Arguments

share	TRUE fills every pile to the same height, so the plot shows each group's share rather than its amount. Defaults to FALSE.
baseline	Where the pile starts: "zero" (the default), "center" for a stream centered on its own middle, or "wiggle" to minimize the movement of the bands.

Details

stack Piles the marks of each group end to end, so the total is the height of the pile.

dodge Places them side by side instead, so each is measured from the same baseline and they can be compared directly. On point it moves each point across its category only as far as it must to clear its neighbors, so no two touch: the beeswarm.

On bars, boxes and whiskers both need something to separate: a color or group binding that says which mark belongs to which group. Without one there is nothing to stack or dodge, and the engine says so. `point * dodge` needs none, since the points in one category already collide.

Value

A `gog_atom` joined to a mark with `*`.

See Also

[jitter\(\)](#), which resolves overlap by displacing marks slightly rather than by arranging them; [transforms](#) for the statistics these compose with.

Examples

```
# Piled: the height of the pile is the total.
s <- data(mtcars) + bar * count * stack + x(cyl) + color(gear)

# Side by side: each bar measured from the same baseline.
d <- data(mtcars) + bar * count * dodge + x(cyl) + color(gear)

# Every pile the same height, so the plot shows shares.
f <- data(mtcars) + bar * count * stack(share = TRUE) + x(cyl) + color(gear)
```

style

Set constant visual properties on a layer.

Description

Channels *map*: `color(species)` asks the reader "which species?" and earns a legend to answer it. `style()` *sets*: it fixes a visual property at one value for the whole layer, maps no column, consumes no scale, and produces no legend — there is nothing to decode.

Usage

```
style(
  color = NULL,
  opacity = NULL,
  size = NULL,
  shape = NULL,
  border_color = NULL,
```

```

border_size = NULL,
caps = NULL,
center = NULL,
nudge = NULL,
pattern = NULL,
arrow = NULL,
reach = NULL,
angle = NULL,
gap = NULL,
...
)

```

Arguments

color	CSS color name ("steelblue") or hex ("4e79a7").
opacity	Number in 0–1. Set literally — unlike the opacity channel there is no data range to rescale from.
size	Pixels: point radius (default 4.5), or line stroke width (default 2).
shape	One of "circle", "square", "triangle", "diamond", "cross", "star", "wye".
border_color	Outline color for a closed glyph; on text, the color of the halo around each label (the panel's background color when unset).
border_size	Outline width for a closed glyph; on text, the width of the halo (3 when unset).
caps	TRUE (default) draws the end caps on an interval; FALSE leaves it bare.
center	TRUE (default) draws a box plot's median line.
nudge	Offset a text label from its point, in pixels.
pattern	A stroke's dash pattern, or a fill's hatch tile.
arrow	Draw the stroke as an arrow.
reach	How far a density curve runs past the data.
angle	Degrees to turn a text label, counterclockwise from horizontal: 90 reads upward. Each label turns about the point it names.
gap	The share of a bar's or a box's slot left empty beside it, from 0 (neighbors touch) to just under 1. On a dodged mark it is a share of the mark's part of the slot. Unset, a category bar leaves 0.2, a binned bar 0, and a box keeps its usual width.
...	Further settings; an unknown one is refused by name rather than ignored.

Details

A constant is therefore not a channel, by the same reasoning that keeps rotation out of the channel set: a channel maps a data column to a visual feature, and a constant maps nothing.

Examples

```

## Not run:
data(df) + x(a) + y(b) + point + style(color = "tomato", opacity = 0.3)
data(df) + x(a) + y(b) + line + style(color = "gray70", size = 1)

## End(Not run)

```

 theme

Set the plot's furniture — the page rather than the ink.

Description

Everything here maps no column, so each is a *setting*; but none of it belongs to a mark either, which is why it is not `style()`. A layer has no gridlines and a plot has no fill, so the two property sets are disjoint and telling them apart by where they were written would make a sub-expression mean different things in different places (Law 6). Spec §7 is the ruling.

Usage

```
theme(
  preset = NULL,
  grid = NULL,
  ratio = NULL,
  tick_angle = NULL,
  font_size = NULL,
  background = NULL,
  strip = NULL,
  strip_text = NULL,
  frame = NULL,
  axis_label = NULL,
  legend = NULL,
  width = NULL,
  height = NULL,
  gap = NULL
)
```

Arguments

preset	One of "gog" (the default look), "minimal" (no gridlines) or "bw" (a white panel inside a full rectangle — the journal look, and note that what goes black and white is the furniture, never the data).
grid	Which gridlines are drawn: "both", "x", "y" or "none", named by the <i>axis</i> whose ticks they mark — which is what lets the setting survive <code>polar()</code> , where the y axis's gridlines are rings.
ratio	The panel's width divided by its height. 1 is a square.
tick_angle	Degrees to turn the x tick labels through, between -90 and 90. Category names that do not fit side by side are turned to 90 without it, and thinned when even that is not enough, so state an angle only to choose one.
font_size	How many pixels a tick label is — and, through it, the size of every other piece of text the plot draws. One number rather than three, because the axis names and the title are a fixed step above it: 11 (the default) gives 11, 13 and 16, and 16 gives 16, 19 and 23. It is a measurement, not a multiplier, so <code>font_size = 1.5</code> is refused. It names no typeface — the engine measures text with its own width table and has none to choose.

background	The panel’s fill — any color the rest of the grammar accepts, including "transparent".
strip	The facet strip’s fill: the band above a panel that names the level it holds. Same colors as background. <code>theme("bw")</code> sets it white, because a gray band reproduces poorly in print, which is the one place that preset is for.
strip_text	The ink of the strip’s label. Leave it out and gog picks whichever of its two defaults reads on the band, so <code>theme(strip = "black")</code> already gives you white type. Name it when the ink is a real choice rather than a legibility one, such as a navy strip with gold type.
frame	How the panel is bounded: "full" (a rectangle, which is what <code>theme("bw")</code> sets and what a journal usually asks for), "axes" (the default, bottom and left only) or "none".
axis_label	Where each axis’s name sits: "beside" (the default, centered along its own axis, which turns the y name through 90 degrees) or "end" (at the axis’s far end, horizontal, so nothing is read sideways).
legend	Which side of the plot its legends sit on: "right" (the default), "left", "top" or "bottom". Above or below the plot the keys are laid in a row, which leaves the width beside the panel to the panel. To leave one legend out, write <code>legend = FALSE</code> on its channel instead, as in <code>color(continent, legend = FALSE)</code> .
width, height	How many pixels the plot asks for. On its own that is the image; composed into an arrangement with <code> </code> or <code>/</code> it is the plot’s <i>cell</i> , and the plots that ask for nothing split what is left — which is how a marginal histogram says it is thin. One meaning in both places (Law 6). Not to be confused with <code>ratio</code> , which shapes the <i>panel</i> inside whatever room the plot was given and never resizes the image.
gap	The space between an arrangement’s plots, in pixels, 0 or more; 20 when unset. An arrangement’s own setting, like its size: write it on the arrangement, <code>(a b) + theme(gap = 8)</code> . A single plot refuses it, and the engine spaces a plot’s own panels, a facet’s included.

Details

A named preset comes first and anything named adjusts it, because a preset you cannot adjust sends you straight back to asking for knobs.

title	<i>Set the plot title.</i>
-------	----------------------------

Description

Set the plot title.

Usage

`title(text)`

Arguments

text The label, as a character string.

transforms

Transforms – statistics computed before the mark is drawn

Description

A **transform** replaces the rows with a summary of them, and the mark then draws that summary. It is joined to a mark with *, which reads as derivation: bar * count is a bar of counts, and it is where a histogram comes from rather than being a chart type to look up.

Usage

smooth

count

sum

mean

median

max

min

proportion

Format

Each is a `gog_atom` object, used bare rather than called.

Details

The same transform behaves identically on every mark it is legal on. Keep the statistic and change the shape and the same numbers draw a line, an area or a step – that is the point of separating the two.

The transforms

count The number of rows for each value of the grouping position.

proportion Each group's share of the total, rather than its count.

sum, mean, median, max, min The aggregate of the measure within each group.

range The lowest and highest value in each group, as a pair – which is what interval and ribbon need.

smooth A fitted trend through the points rather than the points.

Transforms that take an argument are documented on their own pages: [bin\(\)](#), [density\(\)](#), [confidence\(\)](#), [bounds\(\)](#), [partition\(\)](#), [cumulative\(\)](#). [stack\(\)](#), [dodge](#) and [jitter\(\)](#) are a different kind of atom – they resolve the *overlap* of marks rather than computing a statistic.

See Also

[marks](#) for the shapes these are drawn with; [stack\(\)](#) for resolving overlap.

Examples

```
# A histogram is a bar of binned counts -- derived, not a chart type.
h <- data(mtcars) + bar * count + x(cyl)

# The same statistic, a different shape.
l <- data(mtcars) + line * count + x(cyl)

# An aggregate of a measure within each group.
m <- data(mtcars) + bar * mean + x(cyl) + y(mpg)
```

x

Bind the x-axis to a column.

Description

Bind the x-axis to a column.

Usage

```
x(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
  decimals = NULL,
  short = NULL
)
```

Arguments

field	Column to bind (bare name).
scale	How the number becomes a position: "linear" (the default), "log", "time", or "category". A log scale leaves the data in its own units and labels the ticks 1, 10, 100 — it is not the same as plotting $\log(x)$, which would run the axis 0, 1, 2 and leave the reader to exponentiate.
base	Base of a "log" scale; 10 unless given. Use 2 for doublings, octaves or bits, and $\exp(1)$ for e-foldings. The base is very nearly cosmetic — every base draws the same picture, because the axis is normalized by its own range — so what it really chooses is where the gridlines fall and how they are labeled.
limits	The two ends of the axis, as <code>c(low, high)</code> . NA on either end leaves that end fitted to the data.
tick_count	How many ticks the axis should aim for. A target, not a promise: the chosen values are rounded to readable numbers.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	When the plot is faceted, TRUE lets each panel fit this axis to its own rows instead of sharing one scale across all of them.
legend	Not taken by a position, which is read off its axis rather than from a legend; gog refuses it here and names the five channels that draw a legend: color, size, shape, pattern and opacity.
axis	FALSE leaves this axis out of the plot: its tick marks, its numbers and its name, and the margin they stood in. The scale is unchanged, so the gridlines and the panel's lines stay; <code>theme(grid = "none")</code> and <code>theme(frame = "none")</code> remove those.
decimals	Not drawn yet on this channel: only <code>label()</code> prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only <code>label()</code> writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

Examples

```
## Not run:
data(gapminder_2007) + point + x(gdp, scale = "log") + y(life)
data(audio) + line + x(freq, scale = "log", base = 2) + y(level)

## End(Not run)
```

x_label	<i>Override the x-axis label.</i>
---------	-----------------------------------

Description

Override the x-axis label.

Usage

```
x_label(text)
```

Arguments

text	The label, as a character string.
------	-----------------------------------

y	<i>Bind the y-axis to a column.</i>
---	-------------------------------------

Description

Bind the y-axis to a column.

Usage

```
y(  
  field,  
  scale = NULL,  
  base = NULL,  
  limits = NULL,  
  tick_count = NULL,  
  speed = NULL,  
  free = FALSE,  
  legend = NULL,  
  axis = NULL,  
  decimals = NULL,  
  short = NULL  
)
```

Arguments

field	Column to bind (bare name).
scale	How the number becomes a position: "linear" (the default), "log", "time", or "category". A log scale leaves the data in its own units and labels the ticks 1, 10, 100 — it is not the same as plotting log(x), which would run the axis 0, 1, 2 and leave the reader to exponentiate.

base	Base of a "log" scale; 10 unless given. Use 2 for doublings, octaves or bits, and exp(1) for e-foldings. The base is very nearly cosmetic — every base draws the same picture, because the axis is normalized by its own range — so what it really chooses is where the gridlines fall and how they are labeled.
limits	The two ends of the axis, as c(low, high). NA on either end leaves that end fitted to the data.
tick_count	How many ticks the axis should aim for. A target, not a promise: the chosen values are rounded to readable numbers.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	When the plot is faceted, TRUE lets each panel fit this axis to its own rows instead of sharing one scale across all of them.
legend	Not taken by a position, which is read off its axis rather than from a legend; gog refuses it here and names the five channels that draw a legend: color, size, shape, pattern and opacity.
axis	FALSE leaves this axis out of the plot: its tick marks, its numbers and its name, and the margin they stood in. The scale is unchanged, so the gridlines and the panel's lines stay; theme(grid = "none") and theme(frame = "none") remove those.
decimals	Not drawn yet on this channel: only label() prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only label() writes a column's numbers short today, and an axis or a legend chooses its own numbers. gog refuses it when the plot is drawn, and says why.

y_label	<i>Override the y-axis label.</i>
---------	-----------------------------------

Description

Override the y-axis label.

Usage

```
y_label(text)
```

Arguments

text	The label, as a character string.
------	-----------------------------------

z

*Bind the z-axis to a column.***Description**

Bind the z-axis to a column.

Usage

```
z(
  field,
  scale = NULL,
  base = NULL,
  limits = NULL,
  tick_count = NULL,
  speed = NULL,
  free = FALSE,
  legend = NULL,
  axis = NULL,
  decimals = NULL,
  short = NULL
)
```

Arguments

field	Column to bind (bare name).
scale	How the number becomes a position: "linear" (the default), "log", "time", or "category". A log scale leaves the data in its own units and labels the ticks 1, 10, 100 — it is not the same as plotting $\log(x)$, which would run the axis 0, 1, 2 and leave the reader to exponentiate.
base	Base of a "log" scale; 10 unless given. Use 2 for doublings, octaves or bits, and $\exp(1)$ for e-foldings. The base is very nearly cosmetic — every base draws the same picture, because the axis is normalized by its own range — so what it really chooses is where the gridlines fall and how they are labeled.
limits	The two ends of the axis, as <code>c(low, high)</code> . NA on either end leaves that end fitted to the data.
tick_count	How many ticks the axis should aim for. A target, not a promise: the chosen values are rounded to readable numbers.
speed	Not taken: only play spends time, so there is no pace to set here. gog refuses it and says to put it on the frames.
free	When the plot is faceted, TRUE lets each panel fit this axis to its own rows instead of sharing one scale across all of them.
legend	Not taken by a position, which is read off its axis rather than from a legend; gog refuses it here and names the five channels that draw a legend: color, size, shape, pattern and opacity.

axis	FALSE leaves this axis out of the plot: its tick marks, its numbers and its name, and the margin they stood in. The scale is unchanged, so the gridlines and the panel's lines stay; <code>theme(grid = "none")</code> and <code>theme(frame = "none")</code> remove those.
decimals	Not drawn yet on this channel: only <code>label()</code> prints a column's numbers rounded today, and an axis or a legend chooses its own numbers. <code>gog</code> refuses it when the plot is drawn, and says why.
short	Not drawn yet on this channel: only <code>label()</code> writes a column's numbers short today, and an axis or a legend chooses its own numbers. <code>gog</code> refuses it when the plot is drawn, and says why.

z_label	<i>Override the z-axis label.</i>
---------	-----------------------------------

Description

The third position takes a label override for the same reason it takes a scale and a tick: the positions are a family of three, and two of them having one is the per-channel exception the orthogonality law catches. Read only in 3-D, where the axis names sit on the projected cube's edges.

Usage

```
z_label(text)
```

Arguments

text	The label, as a character string.
------	-----------------------------------

Index

`*.gog_atom`, 3
`*.gog_callable(*.gog_atom)`, 3
`+.gog_spec`, 4
`/.gog_spec` (facet-operators), 14

`area` (marks), 23

`bar` (marks), 23
`bin`, 4
`bin()`, 48
`bounds`, 5
`bounds()`, 48
`box`, 5
`box()`, 24
`brush`, 6

`cluster`, 7
`color`, 7
`colour`, 9
`confidence`, 9
`confidence()`, 48
`count` (transforms), 47
`cumulative`, 10
`cumulative()`, 48

`data`, 11
`data()`, 33
`density`, 12
`density()`, 48
`deviation`, 13
`dodge`, 48
`dodge` (stack), 42

`edge` (marks), 23

`facet`, 13
`facet-operators`, 14
`flow`, 14

`globe`, 15
`gog_table`, 16

`group`, 17

`interval` (marks), 23

`jitter`, 18
`jitter()`, 35, 43, 48

`knit_print.gog_arrangement`, 19
`knit_print.gog_spec`, 19

`label`, 20
`layout`, 21
`layout()`, 25
`line` (marks), 23

`map`, 22
`map()`, 16
`marks`, 23, 48
`max` (transforms), 47
`mean` (transforms), 47
`median` (transforms), 47
`min` (transforms), 47

`nest`, 24
`network`, 25
`network()`, 21

`opacity`, 26
`order`, 27

`palette`, 28
`partition`, 28
`partition()`, 48
`path` (marks), 23
`pattern`, 29
`play`, 30
`point` (marks), 23
`polar`, 32
`polar()`, 15
`proportion` (transforms), 47

`quantile`, 32

query, [33](#)

range, [33](#)

render_svg, [34](#)

repel, [34](#)

repr_html.gog_arrangement, [35](#)

repr_html.gog_spec, [35](#)

repr_text.gog_arrangement, [36](#)

repr_text.gog_spec, [36](#)

ribbon (marks), [23](#)

rule (marks), [23](#)

save_gif, [37](#)

save_json, [37](#)

save_svg, [38](#)

shape, [39](#)

size, [40](#)

smooth (transforms), [47](#)

smooth_band, [41](#)

space, [41](#)

stack, [42](#)

stack(), [10](#), [48](#)

step (marks), [23](#)

style, [43](#)

style(), [24](#), [35](#)

sum (transforms), [47](#)

surface (marks), [23](#)

text (marks), [23](#)

theme, [45](#)

theme(), [15](#)

title, [46](#)

transforms, [10](#), [43](#), [47](#)

x, [48](#)

x(), [24](#)

x_label, [50](#)

y, [50](#)

y(), [24](#)

y_label, [51](#)

z, [52](#)

z_label, [53](#)

zone (marks), [23](#)