Package 'ggcube'

May 27, 2026

Type Package

Title 3D Plotting with 'ggplot2'

Version 0.1.0

Description A 'ggplot2' extension for creating 3D figures. Provides 3D geoms, stats, and a coord_3d() coordinate system supporting rotation, perspective, and lighting.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/matthewkling/ggcube, https://matthewkling.github.io/ggcube/

BugReports https://github.com/matthewkling/ggcube/issues

Depends R (>= 3.5), ggplot2

Imports dplyr, tidyr, stringr, scales, rlang, grid, lifecycle, labeling, purrr, magrittr, polyclip, isoband, systemfonts

RoxygenNote 7.3.2

Suggests alphashape3d, av, base64enc, geometry, gifski, knitr, MASS, mgcv, patchwork, progress, ragg, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/Needs/website rmarkdown

NeedsCompilation no

Author Matthew Kling [aut, cre, cph]

Maintainer Matthew Kling <mattkling@berkeley.edu>

Repository CRAN

Date/Publication 2026-05-27 10:00:23 UTC

Contents

aes *Aesthetic mapping with positional z support*

Description

This function extends `ggplot2::aes()` to support positional mapping to the z aesthetic. It maintains full backward compatibility with the original `aes()` function while enabling the convenient `aes(x, y, z)` syntax for 3D plots.

Usage

```
aes(x, y, z, ...)
```

Arguments

`x`	Variable to map to x aesthetic (required)
`y`	Variable to map to y aesthetic (required)
`z`	Variable to map to z aesthetic (optional)
`...`	Other aesthetic mappings (color, size, etc.)

Details

This function is a lightweight wrapper around `ggplot2::aes()` that:

- Maintains full backward compatibility with existing 2D plots
- Enables positional z mapping: `aes(x_var, y_var, z_var)`
- Works with any geom that uses the z aesthetic (contour, raster, 3D plots)
- Passes through all other aesthetics unchanged

Value

An aesthetic mapping object, same as `ggplot2::aes()`

See Also

`aes` for the original aesthetic mapping function, `coord_3d` for 3D coordinate systems

Examples

```
library(ggplot2)

# 2D plots work like regular ggplot2
ggplot(mtcars, aes(mpg, wt)) + geom_point()

# 3D plots can use positional syntax, or explicitly map to z
# same as: ggplot(mtcars, aes(mpg, wt, z = qsec)) + geom_point() + coord_3d()
ggplot(mtcars, aes(mpg, wt, qsec)) + geom_point() + coord_3d()
```

```
# Also works with non-ggcube z-aesthetic geoms
ggplot(mountain, aes(x, y, z)) + geom_contour()
```

`animate_3d` *Animate a 3D plot*

Description

Renders an animated GIF or MP4 of a rotating ggplot with `coord_3d()`, smoothly interpolating rotation angles across frames.

Usage

```
animate_3d(
  plot,
  pitch = NULL,
  roll = NULL,
  yaw = NULL,
  nframes = NULL,
  fps = NULL,
  duration = NULL,
  width = 480,
  height = 480,
  res = 96,
  renderer = gifski_renderer_3d(),
  start_pause = 0,
  end_pause = 0,
  rewind = FALSE,
  cores = 1,
  device = "png",
  progress = interactive()
)
```

Arguments

`plot`	A ggplot object that uses `coord_3d()`.
`pitch`, `roll`, `yaw`	Rotation parameters to animate. Each can be: • `NULL` (default): Hold at the value specified in the plot's `coord_3d()` call. • A single numeric value: Hold at this value for all frames (overriding the plot's coord). • A numeric vector of length 2+: Keyframe values that are interpolated linearly across the animation. Keyframes are evenly spaced, so `c(0, 180, 180)` with 100 frames means: start at 0, reach 180 at frame 50, hold at 180 until frame 100.

`nframes`, `fps`, `duration`
: Control animation length and speed. Specify any two of three; the third is computed as `duration = nframes / fps`. Defaults: `nframes = 100`, `fps = 10`.

`width`, `height`
: Dimensions of the animation in pixels. Defaults to 480 x 480.

`res`
: Resolution in ppi for the rendered frames. Default is 96. Controls the size of point-based text/lines/point elements relative to the plot. See `?grDevices::png` for details.

`renderer`
: A renderer function that combines image frames into an animation file. Built-in options:

  - `gifski_renderer_3d()` (default): Renders a GIF using the **gifski** package.
  - `av_renderer_3d()`: Renders a video using the **av** package.
  - `file_renderer_3d()`: Returns the frame image files without combining them.

  Any function with signature `function(frames, fps)` can be used.

`start_pause`, `end_pause`
: Number of frames to hold at the beginning and end of the animation. Default is 0.

`rewind`
: Logical. If `TRUE`, the animation plays in reverse after completing, creating a seamless loop. Default is `FALSE`.

`cores`
: Number of CPU cores for parallel frame rendering. Default is 1 (sequential). Values greater than 1 use `parallel::parLapply()` to render frames in parallel, which can significantly speed up animation of complex plots. Note that progress reporting is not available during parallel rendering.

`device`
: The graphics device to use for rendering frames. Default is `"png"`. Other options include `"ragg_png"` (requires the **ragg** package).

`progress`
: Whether to print a progress bar. Ignored when `cores > 1`.

Value

The return value of the renderer function. For `gifski_renderer_3d()`, a `gif_3d` object (a file path with class attributes for display in RStudio and knitr). For `av_renderer_3d()`, a `video_3d` object (a file path with class attributes for display). For `file_renderer_3d()`, a character vector of file paths to the rendered frame images.

See Also

`anim_save_3d()` for saving animations to file. `gifski_renderer_3d()` and `file_renderer_3d()` for renderer options.

Examples

```
# Plot to animate. Note `anchor = "camera"` keeps light source from rotating.
p <- ggplot() +
  geom_function_3d(
    fun = function(x, y) sin(x) * cos(y),
```

```
    xlim = c(-pi, pi), ylim = c(-pi, pi),
    fill = "steelblue", color = "steelblue", linewidth = .5) +
  coord_3d(light = light(mode = "hsl", anchor = "camera",
                         direction = c(1, 1, 0))) +
  theme_void()

# Simple turntable rotation
animate_3d(p, yaw = c(-30, 330))

# Multi-segment orbit with pitch change
animate_3d(p, yaw = c(0, 720), roll = c(-90, 0, -90),
           nframes = 120, fps = 15)

# Save to file (writes to a temporary path; specify your own to keep it)
anim <- animate_3d(p, yaw = c(0, 360))
anim_save_3d(anim, file.path(tempdir(), "rotating_surface.gif"))
```

`anim_save_3d` *Save a 3D animation to a file*

Description

Saves an animation object produced by `animate_3d()` to a file. If no animation is provided, saves the most recently rendered animation.

Usage

```
anim_save_3d(animation = NULL, filename)
```

Arguments

`animation`	An animation object from `animate_3d()`, or `NULL` to use the last rendered animation.
`filename`	Output file path.

Value

Invisibly returns the output file path (a character string). Called primarily for its side effect of copying the animation to `filename`.

Examples

```
p <- ggplot() +
  geom_function_3d(
    fun = function(x, y) sin(x) * cos(y),
    xlim = c(-pi, pi), ylim = c(-pi, pi),
```

```
    fill = "steelblue", color = "steelblue") +
  coord_3d()

animate_3d(p, yaw = c(0, 360))
anim_save_3d(filename = file.path(tempdir(), "my_animation.gif"))
```

---

annotate_3d *Create a 3D annotation specification*

Description

Defines fixed-position annotations to be embedded within a 3D layer. Annotations are depth-sorted together with the primary geometry, unlike ggplot2's `annotate()` which creates a separate layer.

Usage

```
annotate_3d(type, ...)
```

Arguments

type	Character string specifying the annotation type. One of `"point"`, `"text"`, or `"segment"`.
...	Type-specific parameters. See Details.

Details

Parameters can be vectors to create multiple annotations of the same type in one call, with scalar values recycled as needed (matching `ggplot2::annotate()` behaviour).

Point annotations:

Requires `x`, `y`, `z`. Optional styling: `colour`/`color`, `fill`, `size`, `shape`, `alpha`, `stroke`.

Text annotations:

Requires `x`, `y`, `z`, `label`. Optional styling: `colour`/`color`, `size`, `alpha`, `family`, `fontface`, `hjust`, `vjust`, `angle`, `lineheight`.

Segment annotations:

Requires `x`, `y`, `z`, `xend`, `yend`, `zend`. Optional styling: `colour`/`color`, `linewidth`, `linetype`, `alpha`.

Value

An S3 object of class `"annotate_3d"`.

Examples

```
p <- ggplot(mountain, aes(x, y, z)) +
  coord_3d(ratio = c(2, 3, 1),
           light = light(mode = "hsl", direction = c(-1, 0, 0)))

# Basic point annotation
p + geom_surface_3d(
  annotate = annotate_3d("point", x = .5, y = .25, z = 100,
                         color = "red", size = 3))

# Vectorized: multiple points in one call
p + geom_surface_3d(
  annotate = annotate_3d("point", x = .5, y = .25, z = seq(50, 100, 5),
                         color = "red", size = 3))

# Multiple annotation types
p + geom_surface_3d(
  annotate = list(
    annotate_3d("point", x = .5, y = .25, z = 100,
                color = "red", size = 3),
    annotate_3d("text", x = .5, y = .25, z = 100,
                color = "red", label = "Look here!",
                vjust = -1, fontface = "bold"),
    annotate_3d("segment", x = .5, y = .25, z = 100,
                xend = .5, yend = .25, zend = 50,
                color = "red", size = 3)))
```

---

camera_facing *Compute camera-facing direction for 3D text*

Description

Creates a specification for camera-facing "billboard" text that will compute the appropriate facing direction for each label position.

Usage

```
camera_facing(
  coord = NULL,
  pitch = 0,
  roll = -60,
  yaw = -30,
  dist = 2,
  mode = c("plane", "point")
)
```

Arguments

coord	Optional objected created by coord_3d. If provided, coord parameters are drawn from this and the pitch, roll, yaw, dist arguments are ignored.
pitch, roll, yaw	Rotation angles in degrees, matching the values you'll use in coord_3d().
dist	Distance from viewer to center of the data cube, matching the value you'll use in coord_3d(). Default is 2. Only used when mode = "point".
mode	Character string specifying how labels should face the camera: • "plane" (default): All labels face parallel to the view plane, like billboards. This is typically easier to read. • "point": Each label faces toward the camera position. Labels at the edges will angle inward, giving perspective-correct orientation.

Details

With mode = "plane", all text labels face the same direction (parallel to the viewing plane), similar to traditional billboard rendering. This produces clean, readable labels.

With mode = "point", each label faces directly toward the camera's position in 3D space. Labels near the edges of the plot will angle inward, which is geometrically correct but can look less clean.

Since geom_text_3d() computes vertex positions before the view is known, you must specify the view parameters here and use the same values in coord_3d().

Value

A camera_facing_spec object to pass to the facing parameter of geom_text_3d().

See Also

geom_text_3d() for rendering 3D text labels

Examples

```
# Parallel billboard text (default) - all labels face same direction
df <- expand.grid(x = c("H", "B"), y = c("a", "o", "u"), z = c("g", "t"))
df$label <- paste0(df$x, df$y, df$z)

ggplot(df, aes(x, y, z = z, label = label)) +
  geom_text_3d(method = "polygon",
               facing = camera_facing(pitch = 20, roll = -60, yaw = -30)) +
  coord_3d(pitch = 20, roll = -60, yaw = -30)

# Point-facing text - labels angle toward camera position
ggplot(df, aes(x, y, z = z, label = label)) +
  geom_text_3d(method = "polygon",
               facing = camera_facing(pitch = 20, roll = -60, yaw = -30,
                                      mode = "point")) +
  coord_3d(pitch = 20, roll = -60, yaw = -30)
```

coord_3d *3D coordinate system*

Description

coord_3d is a 3D coordinate system that creates a 2D view of 3D data. This is the essential core component of any plot made with ggcube. It supports rotation, perspective projection, and options for controlling plot aspect ratios, panel selection, axis label placement, and lighting.

Usage

```
coord_3d(
  pitch = 0,
  roll = -60,
  yaw = -30,
  persp = TRUE,
  dist = 2,
  expand = TRUE,
  clip = "off",
  panels = "background",
  xlabels = "auto",
  ylabels = "auto",
  zlabels = "auto",
  title_position = c("auto", "center"),
  rotate_labels = TRUE,
  scales = "free",
  ratio = c(1, 1, 1),
  zoom = 1,
  light = ggcube::light(),
  ...
)
```

Arguments

roll, pitch, yaw	Rotation around x, y, and z axes, respectively, in degrees. Positive values rotate the near face of the plot "downward", "rightward", and clockwise, respectively.
persp	Logical indicating whether to apply perspective projection. When TRUE (the default), objects farther from the viewer appear smaller. When FALSE, produces an orthographic projection in which lines that are parallel in 3D space render as parallel in the plot.
dist	Distance from viewer to center of the data cube. Only used when persp = TRUE. Larger values create less perspective distortion. Default is 2. Values less than 1 are allowed but can be problematic for rendering.
expand	Logical indicating whether to expand axis ranges beyond the data range, similar to standard ggplot2 behavior. If TRUE (the default), expansion behavior can be controlled using standard axis scaling functions, e.g. ... + scale_x_continuous(expand = expansion(.5)).

`clip` Character string indicating clipping behavior. Use `"off"` (the default, recommended for some 3D plots) to allow drawing outside the plot panel.

`panels` Character vector specifying which panels to render, including one or more of the following:

- `"background"` (the default), `"foreground"`: faces laying behind or in front of the cube's interior volume, respectively. These panels vary depending on plot rotation.
- `"xmin"`, `"ymax"`, etc.: names of specific cube faces.
- `"all"`, `"none"`: display the full cube or remove all faces.

See cube_theming for details on panel styling, including transparency of foreground panels.

`xlabels`, `ylabels`, `zlabels`

Character strings or length-2 character vectors specifying axis label (text and title) placement. Labels are placed inline with grid lines for the selected panel face. For each axis, there are four potential panels where labels could be placed, and two potential edges for each panel. Labels can only be placed on visible faces (see `panels` argument). Each parameter accepts:

- `"auto"` (default): Automatic edge selection based on an algorithm that prioritizes edges that are visible on the periphery of the plot and considers several attributes of face geometry for better readability.
- `c("face1", "face2")`: Manual edge specification using two adjacent face names (e.g., `c("xmin", "ymin")` selects the edge shared by the xmin and ymin faces). The **first face** in the vector determines which face the axis labels will be aligned with, while the second face identifies which edge of this face gets labelled. Available face names are: "xmin", "xmax", "ymin", "ymax", "zmin", "zmax".

See cube_theming for details on axis label styling.

`title_position` Character string controlling axis title placement. Currently only affects titles for internal axes (not on the plot periphery); titles for axes on the plot periphery are always centered along the axis edge.

- `"auto"` (default): Internal axis titles are placed at the near end of the axis, outside the plot area.
- `"center"`: Internal axis titles are centered along the axis edge.

`rotate_labels` Logical indicating whether axis labels (text and titles) should automatically rotate to align with the projected axis directions. When `FALSE`, uses theme text and title angle settings.

`scales` Character string specifying aspect ratio behavior:

- `"free"` (default): Each axis scales independently to fill cube space, then `ratio` applies to standardized coordinates. This gives maximum visual range for each dimension.
- `"fixed"`: Maintains proportional relationships in raw data values, as scaled by `ratio`. Similar to `coord_fixed()` but for 3D (visual ratios match the labeled axis ranges).

`ratio` Numeric vector of length 3 specifying relative axis lengths as `c(x, y, z)`. Defaults to `c(1, 1, 1)` for equal proportions.

- With `scales = "free"`: Ratios apply to scaled cube coordinates
- With `scales = "fixed"`: Ratios apply to original data coordinates

`zoom` Numeric value controlling the framing of the plot. Values greater than 1 zoom in (tighter framing, may crop edges), values less than 1 zoom out (more whitespace around the plot). Default is 1.

`light` A lighting specification object created by `light()`,`"none"` to disable lighting, or `NULL` to inherit plot-level lighting specs from the coord. Specify plot-level lighting in `coord_3d()` and layer-specific lighting in `geom_*3d()` functions.

`...` Additional arguments reserved for internal use.

Value

A `Coord` object that can be added to a ggplot.

See Also

`light()` for lighting specification, cube_theming for panel and text styling, polygon_params for 3D-related parameters for polygon layers.

Examples

```
# base plot used in examples
p <- ggplot() +
  geom_function_3d(
    aes(fill = after_stat(z), color = after_stat(z)),
    fun = function(x, y) sin(x) * cos(y),
    xlim = c(-pi, pi), ylim = c(-pi, pi),
    n = 50, light = light("direct", contrast = .7)) +
  scale_fill_viridis_c() +
  scale_color_viridis_c() +
  theme(legend.position = "none")

# 3D plot with default coord settings
p + coord_3d()



# Use `pitch`, `roll`, `yaw` to control plot rotation ----------------------

# zero rotation gives view from x-y face
p + coord_3d(pitch = 0, roll = 0, yaw = 0)

# pitch rotates plot around y axis
p + coord_3d(pitch = 30, roll = 0, yaw = 0)

# roll rotates plot around x axis
p + coord_3d(pitch = 0, roll = 30, yaw = 0)

# yaw rotates plot around z axis
p + coord_3d(pitch = 0, roll = 0, yaw = 30)
```

```
# combine them to achieve arbitrary rotations
p + coord_3d(pitch = 20, roll = 40, yaw = 60)


# Use `persp` and `dist` to control perspective effects --------------------

# strong perspective effect as if seen from very close
p + coord_3d(dist = 1)

# weaker perspective effects as if seen from far away
p + coord_3d(dist = 3)

# orthographic projection (`dist = Inf` would be equivalent but it errors)
p + coord_3d(persp = FALSE)


# Use `scales` and `ratio` to control aspect ratio -------------------------

# The default "free" scales shown above give cube with maximum visual range.
# Use "fixed" scales to make figure match data scales, like coord_fixed.
p + coord_3d(scales = "fixed")

# Custom aspect ratios: make y twice as long visually
p + coord_3d(scales = "free", ratio = c(1, 2, 1))

# Combine behaviors: fix scales and make y twice as long
p + coord_3d(scales = "fixed", ratio = c(1, 2, 1))


# Use `panels` to select which cube faces to render ------------------------

p + coord_3d(panels = c("xmin", "xmax", "zmax"))

# foreground panels default to 20% opaque (can be styled using `theme()`)
p + coord_3d(panels = "all")


# Use label params to control axis text placement and rotation -------------

p + coord_3d(xlabels = c("ymax", "zmax"),
             zlabels = c("xmax", "ymin"))

p + coord_3d(rotate_labels = FALSE)
```

---

cube_theming *Using themes to style 3D panels and axis labels*

Description

In ggcube, standard ggplot2 themes generally influence 3D plots as expected, including adding complete themes like `ggplot2::theme_dark()` and modifying theme elements like `theme(panel.background = element_rect`. However, ggcube also provides additional theme elements that control 3D-specific styling of panels and labels.

Text elements

- `axis.text.z`: Styling for z-axis tick labels (inherits from `axis.text`)
- `axis.title.z`: Styling for z-axis title (inherits from `axis.title`)
- `axis.text`, `axis.text`: Standard styling with `element_text()`.

Use `element_text(margin = margin(...))` to adjust text padding, with left/right margins affecting axis text and top/bottom margins affecting axis titles; since placement and justification of these elements varies dynamically, no distinction is made between left and right margins, or between top and bottom margins – you can set either, and the maximum of the two will be used.

Panel elements

- `panel.foreground`: Styling for cube faces rendered in front of data (inherits from `panel.background`). Uses `element_rect(alpha = .2)` by default, to prevent foreground panels from obscuring the data.
- `panel.border.foreground`: Styling for cube faces rendered in front of data (inherits from `panel.border`)
- `panel.grid.foreground`: Styling for grid lines on foreground faces (inherits from `panel.grid`)
- `panel.grid.major.foreground`: Major grid lines on foreground faces (inherits from `panel.grid.foreground`)

Background panels use standard `panel.background`, `panel.border`, `panel.grid`, etc., while foreground panels use the `*.foreground` variants listed above. Since the foreground elements inherit from the standard background and grid elements, you can use `panel.background`, etc. to style both background and foreground faces simultaneously.

Enhanced elements

- `element_rect()` extends `ggplot2::element_rect()` by adding an `alpha` parameter for transparency effects. This is particularly useful for `panel.foreground` components that sit in front of the data.

Examples

```
# example code
p <- ggplot(sphere_points, aes(x, y, z)) +
  geom_hull_3d() +
  coord_3d(panels = "all") +
  theme(panel.background = element_rect(color = "black"),
          panel.border = element_rect(color = "black"),
          panel.foreground = element_rect(alpha = .3),
          panel.grid.foreground = element_line(color = "gray", linewidth = .25),
          axis.text = element_text(color = "darkblue"),
```

```
            axis.text.z = element_text(color = "darkred"),
            axis.title = element_text(margin = margin(t = 30)), # add padding
            axis.title.x = element_text(color = "magenta"))
```

`element_rect` *Rectangle theme element with alpha support*

Description

This function extends `ggplot2::element_rect()` to support transparency via an alpha parameter. It maintains full backward compatibility with the original `element_rect()` function while enabling transparent panel styling, which is particularly useful for foreground panels in 3D plots.

Usage

```
element_rect(
  fill = NULL,
  colour = NULL,
  linewidth = NULL,
  linetype = NULL,
  color = NULL,
  inherit.blank = FALSE,
  size = lifecycle::deprecated(),
  alpha = NULL
)
```

Arguments

`fill`	Fill color for the rectangle. Use `NA` for no fill.
`colour, color`	Line color for the rectangle border. Use `NA` for no border.
`linewidth`	Line width for the rectangle border.
`linetype`	Line type for the rectangle border (e.g., "solid", "dashed").
`inherit.blank`	Should this element inherit from `element_blank`?
`size`	**[Deprecated]** Use `linewidth` instead.
`alpha`	Transparency level for the rectangle fill, ranging from 0 (completely transparent) to 1 (completely opaque). Particularly useful for styling foreground panels in 3D plots to create layered visual effects.

Value

A theme element object that can be used in `theme()` specifications.

See Also

`element_rect` for the original function, `coord_3d` for 3D coordinate systems that utilize foreground panels, cube_theming for details on panel/gridline/axis label styling.

Examples

```
# Basic 3D plot with semi-transparent foreground panels
ggplot(mountain, aes(x, y, z)) +
  stat_surface_3d(fill = "darkblue", color = "lightblue", linewidth = .1) +
  coord_3d(panels = c("background", "ymin")) +
  theme(panel.foreground = element_rect(alpha = 0.6))

# Completely transparent foreground panels
ggplot(mtcars, aes(mpg, wt, qsec)) +
  geom_point() +
  coord_3d(panels = "all") +
  theme(panel.border = element_rect(color = "black"),
        panel.foreground = element_rect(fill = "blue", alpha = 0))
```

---

geom_bar_3d *3D bar chart with automatic counting or binning*

Description

Creates 3D bar charts by automatically counting discrete data or binning continuous data. This is the 3D analogue of ggplot2::geom_bar() and ggplot2::geom_histogram().

Usage

```
geom_bar_3d(
  mapping = NULL,
  data = NULL,
  stat = StatBar3D,
  position = "identity",
  ...,
  bins = 10,
  binwidth = NULL,
  drop = TRUE,
  width = 1,
  faces = "all",
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_bar_3d(
```

```
  mapping = NULL,
  data = NULL,
  geom = GeomPolygon3D,
  position = "identity",
  ...,
  bins = 10,
  binwidth = NULL,
  drop = TRUE,
  width = 1,
  faces = "all",
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes().
data	The data to be displayed in this layer.
stat	The statistical transformation to use. Defaults to StatBar3D.
position	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use position_on_face().
...	Other arguments passed on to the layer function (typically GeomPolygon3D), such as aesthetics like colour, fill, linewidth, annotate = annotate_3d(...), etc.
bins	Number of bins in each dimension. Either a single value (used for both x and y) or a vector of length 2 giving c(bins_x, bins_y). Default is 10. Ignored for discrete variables.
binwidth	Bin width in each dimension. Either a single value or a vector of length 2 giving c(binwidth_x, binwidth_y). If provided, overrides bins. Ignored for discrete variables.
drop	If TRUE (the default), empty bins/combinations are not rendered. If FALSE, empty bins render as zero-height columns.
width	Column width as a fraction of bin spacing. Either a single value (used for both x and y) or a vector of length 2 giving c(width_x, width_y). Default is 1.0 (columns touch). Use values less than 1 for gaps between columns.
faces	Character vector specifying which faces to render. Options: • "all" (default): Render all 6 faces • "none": Render no faces • Vector of face names: c("zmax", "xmin", "ymax"), etc.

	Valid face names: "xmin", "xmax", "ymin", "ymax", "zmin", "zmax". Note that this setting acts jointly with backface culling, which removes faces whose interior faces the viewer – e.g., when `cull_backfaces = TRUE` and `faces = "all"` (the default), only front faces are rendered.
`light`	A lighting specification object created by `light()`,`"none"` to disable lighting, or `NULL` to inherit plot-level lighting specs from the coord. Specify plot-level lighting in `coord_3d()` and layer-specific lighting in `geom_*3d()` functions.
`cull_backfaces`	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
`sort_method`	Depth sorting algorithm. See sorting_methods for details.
`scale_depth`	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
`force_convex`	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.
`na.rm`	If `FALSE`, missing values are removed.
`show.legend`	Logical indicating whether this layer should be included in legends.
`inherit.aes`	If `FALSE`, overrides the default aesthetics.
`geom`	The geometric object used to display the data. Defaults to `GeomPolygon3D`.

Details

The stat automatically detects whether x and y are discrete or continuous:

- **Both discrete**: Counts occurrences of each (x, y) combination
- **Both continuous**: Performs 2D binning (like a 3D histogram)
- **Mixed**: Bins the continuous axis while preserving discrete groups

For pre-computed bar heights, use `geom_col_3d()` instead.

Value

A `Layer` object that can be added to a ggplot.

Aesthetics

`stat_bar_3d()` requires the following aesthetics:

- **x**: X coordinate

- **y**: Y coordinate

And optionally understands:

- **weight**: Observation weights for counting

Computed variables

These variables can be used with `after_stat()` to map to aesthetics:

- `count`: Number of observations in each bin (default for z)
- `proportion`: Count divided by total count (sums to 1)
- `ncount`: Count scaled to maximum of 1
- `density`: Count divided by (total count × bin area); integrates to 1 for continuous data
- `ndensity`: Density scaled to maximum of 1

See Also

`geom_col_3d()` for pre-computed heights, `coord_3d()` for 3D coordinate systems, `light()` for lighting specifications.

Examples

```
# Discrete x and y: count combinations
d_discrete <- data.frame(
  x = sample(letters[1:4], 200, replace = TRUE),
  y = sample(LETTERS[1:3], 200, replace = TRUE)
)
ggplot(d_discrete, aes(x, y)) +
  geom_bar_3d() +
  coord_3d()

# Continuous x and y: 2D histogram
d_cont <- data.frame(x = rnorm(1000), y = rnorm(1000))
ggplot(d_cont, aes(x, y)) +
  geom_bar_3d() +
  coord_3d()

# Mixed: one discrete, one continuous
d_mixed <- data.frame(
  group = rep(c("A", "B", "C"), each = 100),
  value = c(rnorm(100, 2), rnorm(100, 1, 2), rnorm(100, 0))
)
ggplot(d_mixed, aes(x = group, y = value, fill = group)) +
  geom_bar_3d(bins = 20, width = c(.5, 1)) +
  coord_3d(scales = "fixed", ratio = c(1, 1, .1))

# Use density instead of count for z
ggplot(d_mixed,
       aes(x = group, y = value, z = after_stat(density))) +
  geom_bar_3d(bins = 20, width = c(.5, 1)) +
```

```
  coord_3d()

# Show empty bins with drop = FALSE
ggplot(d_cont, aes(x, y)) +
  geom_bar_3d(drop = FALSE) +
  coord_3d()
```

---

geom_col_3d *3D columns from grid data*

Description

Creates 3D columns (rectangular prisms) from grid data in which x and y fall on a regular grid. Works with both complete and sparse grid data. Each data point becomes a rectangular 3D column extending from a base level to the data value.

Usage

```
geom_col_3d(
  mapping = NULL,
  data = NULL,
  stat = StatCol3D,
  position = "identity",
  ...,
  width = 1,
  faces = "all",
  zmin = NULL,
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_col_3d(
  mapping = NULL,
  data = NULL,
  geom = GeomPolygon3D,
  position = "identity",
  ...,
  width = 1,
  faces = "all",
  zmin = NULL,
```

```
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes().
data	The data to be displayed in this layer.
stat	The statistical transformation to use on the data. Defaults to StatCol3D.
position	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use position_on_face().
...	Other arguments passed on to the layer function (typically GeomPolygon3D), such as aesthetics like colour, fill, linewidth, annotate = annotate_3d(...), etc.
width	Numeric value controlling box width as a fraction of grid spacing. Default is 1.0 (volumes touch each other). Use 0.8 for small gaps, 1.2 for overlap. Grid spacing is determined automatically using ggplot2::resolution()
faces	Character vector specifying which faces to render. Options: • "all" (default): Render all 6 faces • "none": Render no faces • Vector of face names: c("zmax", "xmin", "ymax"), etc. Valid face names: "xmin", "xmax", "ymin", "ymax", "zmin", "zmax". Note that this setting acts jointly with backface culling, which removes faces whose interior faces the viewer – e.g., when cull_backfaces = TRUE and faces = "all" (the default), only front faces are rendered.
zmin	Base level for all columns. When provided as a parameter, overrides any zmin aesthetic mapping. If NULL (the default), uses the zmin aesthetic if mapped, otherwise defaults to 0.
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
sort_method	Depth sorting algorithm. See sorting_methods for details.

scale_depth	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
force_convex	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.
na.rm	If FALSE, missing values are removed.
show.legend	Logical indicating whether this layer should be included in legends.
inherit.aes	If FALSE, overrides the default aesthetics.
geom	The geometric object used to display the data. Defaults to GeomPolygon3D.

Details

This is analogous to ggplot2::geom_col() for 3D plots. For automatic counting or binning, see geom_bar_3d().

Note that column geometries often require pairwise depth sorting for correct rendering. This is the default for smaller data sets, but not for larger data sets due to compute speed; in those cases you may wish to manually specify sort_method = "pairwise" for good results.

Value

A Layer object that can be added to a ggplot.

Aesthetics

stat_col_3d() requires the following aesthetics:

- **x**: X coordinate (grid position)
- **y**: Y coordinate (grid position)
- **z**: Z coordinate (column top height)

And optionally understands:

- **zmin**: Base level for each column (can be overridden by the zmin parameter)

Computed variables

- normal_x, normal_y, normal_z: Face normal components
- col_id: Sequential column number
- face_type: Face name ("zmax", "xmin", etc.)

See Also

geom_bar_3d() for automatic counting/binning, stat_surface_3d() for smooth surface rendering, coord_3d() for 3D coordinate systems, light() for lighting specifications.

Examples

```
# Basic 3D bar chart from regular grid
# (columns extend from z=0 by default)
d <- expand.grid(x = 1:5, y = 1:5)
d$z <- d$x + d$y + rnorm(25, 0, 0.5)
ggplot(d, aes(x, y, z)) +
  geom_col_3d() +
  coord_3d()

# Set uniform base level using `zmin` parameter
ggplot(d, aes(x, y, z)) +
  geom_col_3d(aes(fill = z), color = "white",
              zmin = 5) +
  coord_3d()

# Set variable base levels using `zmin` aesthetic
d$base_level <- runif(nrow(d), -5, 1)
ggplot(d, aes(x, y, z = z, zmin = base_level)) +
  geom_col_3d(color = "black") +
  coord_3d()

# Show only a subset of column faces
ggplot(d, aes(x, y, z)) +
  geom_col_3d(faces = c("zmax", "ymin"),
    cull_backfaces = FALSE,
    fill = "steelblue", color = "black") +
  coord_3d()

# With gaps between columns
ggplot(d, aes(x, y, z)) +
  geom_col_3d(color = "black", width = 0.6) +
  coord_3d()
```

---

geom_contour_3d *Contours of a 3D surface*

Description

Renders a surface as stacked horizontal contour bands. Each contour band is a polygon placed at its corresponding z-level.

Usage

```
geom_contour_3d(
  mapping = NULL,
  data = NULL,
  stat = "surface_3d",
  position = "identity",
```

```
  ...,
  bins = 20,
  binwidth = NULL,
  breaks = NULL,
  cull_backfaces = FALSE,
  sort_method = "pairwise",
  scale_depth = TRUE,
  force_convex = FALSE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_contour_3d(
  mapping = NULL,
  data = NULL,
  geom = "contour_3d",
  position = "identity",
  ...,
  bins = 20,
  binwidth = NULL,
  breaks = NULL,
  cull_backfaces = FALSE,
  sort_method = "pairwise",
  scale_depth = TRUE,
  force_convex = FALSE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes().
data	Point grid data with x, y, z coordinates.
stat	Statistical transformation. Defaults to "surface_3d".
position	Position adjustment, defaults to "identity".
...	Other arguments passed to the layer.
bins	Number of contour levels. Default is 20. Ignored if breaks or binwidth is provided.
binwidth	Width of each contour band. Overrides bins if provided.
breaks	Numeric vector specifying exact contour break points. Overrides both bins and binwidth if provided.
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic

	reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
`sort_method`	Depth sorting algorithm. See sorting_methods for details.
`scale_depth`	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
`force_convex`	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.
`light`	A lighting specification object created by `light()`,`"none"` to disable lighting, or `NULL` to inherit plot-level lighting specs from the coord. Specify plot-level lighting in `coord_3d()` and layer-specific lighting in `geom_*3d()` functions.
`na.rm`	If `FALSE`, missing values are removed.
`show.legend`	Logical indicating whether this layer should be included in legends.
`inherit.aes`	If `FALSE`, overrides the default aesthetics.
`geom`	Geometry function used to display the data. Defaults to `"surface_3d"`.

Details

This geom takes point grid data (like that produced by `stat_surface_3d()`, `stat_function_3d()`, `stat_smooth_3d()`, or `stat_density_3d()`) and converts it to filled contour polygons using the `isoband` package.

Value

A `Layer` object that can be added to a ggplot.

Aesthetics

`geom_contour_3d()` requires:

x, y, z Point coordinates forming a regular grid

And understands these additional aesthetics:

fill Band fill color. For automatic coloring by elevation, use `aes(fill = after_stat(z))`. Default is "grey60".

colour Band border color (default: "grey30")

alpha Transparency

linewidth Border width (default: 0.1)

linetype Border line type

Computed variables

Each contour band is placed at its corresponding z-level (the upper boundary of the band). To color by elevation, use `aes(fill = after_stat(z))`.

See Also

`geom_surface_3d()` for continuous surface rendering, `geom_ridgeline_3d()` for cross-sectional ridgeline rendering, `stat_function_3d()` for mathematical function surfaces, `stat_smooth_3d()` for fitted model surfaces, `coord_3d()` for 3D coordinate systems, `ggplot2::geom_contour_filled()` for the 2D equivalent.

Examples

```
# Basic usage with volcano data
ggplot(mountain, aes(x, y, z)) +
      geom_contour_3d(color = "white", fill = "black") +
      coord_3d(light = "none", ratio = c(1.5, 2, 1))

# Map fill to elevation and customize number of levels
ggplot(mountain, aes(x, y, z, fill = after_stat(z))) +
      geom_contour_3d(bins = 12, color = "white") +
      scale_fill_viridis_c() +
      coord_3d(light = "none", ratio = c(1.5, 2, 1))

# Specify exact breaks
ggplot(mountain, aes(x, y, z, fill = after_stat(z))) +
  geom_contour_3d(breaks = seq(0, 200, by = 5)) +
  scale_fill_viridis_c() +
  coord_3d(light = "none")

# With stat_density_3d
ggplot(faithful, aes(eruptions, waiting)) +
  stat_density_3d(geom = "contour_3d",
    sort_method = "pairwise") +
  coord_3d()

# With stat_function_3d
ggplot() +
  stat_function_3d(
    fun = function(x, y) sin(x) * cos(y),
    xlim = c(-pi, pi), ylim = c(-pi, pi),
    geom = "contour_3d",
    bins = 50, color = "black"
  ) +
  scale_fill_viridis_c(option = "B") +
  coord_3d(light = "none")
```

`geom_density_3d` *3D surface from 2D density estimate*

Description

A 3D version of `ggplot2::stat_density_2d()`. Creates surfaces from 2D point data using kernel density estimation. The density values become the z-coordinates of the surface, allowing visualization of data concentration as peaks and valleys in 3D space.

Usage

```
geom_density_3d(
  mapping = NULL,
  data = NULL,
  stat = "density_3d",
  position = "identity",
  ...,
  grid = "rectangle",
  n = 40,
  direction = "x",
  trim = TRUE,
  h = NULL,
  adjust = 1,
  pad = 0.1,
  min_ndensity = 0,
  light = NULL,
  cull_backfaces = FALSE,
  sort_method = NULL,
  force_convex = FALSE,
  scale_depth = TRUE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_density_3d(
  mapping = NULL,
  data = NULL,
  geom = "surface_3d",
  position = "identity",
  ...,
  grid = "rectangle",
  n = 40,
  direction = "x",
  trim = TRUE,
  h = NULL,
  adjust = 1,
```

```
  pad = 0.1,
  min_ndensity = 0,
  light = NULL,
  cull_backfaces = FALSE,
  sort_method = NULL,
  force_convex = FALSE,
  scale_depth = TRUE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). This stat requires x and y aesthetics. By default, fill is mapped to after_stat(density) and z is mapped to after_stat(density).
data	The data to be displayed in this layer. Must contain x and y columns with point coordinates.
stat	The statistical transformation to use on the data. Defaults to StatDensity3D.
position	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use position_on_face().
...	Other arguments passed on to the layer function (typically GeomPolygon3D), such as aesthetics like colour, fill, linewidth, annotate = annotate_3d(...), etc.
grid, n, direction, trim	Parameters determining the geometry, resolution, and orientation of the surface grid. See grid_generation for details.
h	Bandwidth vector. If NULL (default), uses automatic bandwidth selection via MASS::bandwidth.nrd(). Can be a single number (used for both dimensions) or a vector of length 2 for different bandwidths in x and y directions.
adjust	Multiplicative bandwidth adjustment factor. Values greater than 1 produce smoother surfaces; values less than 1 produce more detailed surfaces. Default is 1.
pad	Proportional range expansion factor. The computed density grid extends this proportion of the raw data range beyond each data limit. Default is 0.1.
min_ndensity	Lower cutoff for normalized density (computed variable ndensity described below), below which to filter out results. This is particularly useful for removing low-density corners of rectangular density grids when density surfaces are shown for multiple groups, as in the example below. Default is 0 (no filtering).
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for

	open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
`sort_method`	Depth sorting algorithm. See sorting_methods for details.
`force_convex`	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.
`scale_depth`	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
`na.rm`	If `FALSE`, missing values are removed.
`show.legend`	Logical indicating whether this layer should be included in legends.
`inherit.aes`	If `FALSE`, overrides the default aesthetics.
`geom`	The geometric object used to display the data. Defaults to `GeomPolygon3D`.

Value

A `Layer` object that can be added to a ggplot.

Aesthetics

`stat_density_3d()` requires the following aesthetics from input data:

- **x**: X coordinate of data points
- **y**: Y coordinate of data points

And optionally understands:

- **group**: Grouping variable for computing separate density surfaces
- Additional aesthetics are passed through for surface styling

Computed variables specific to StatDensity3D

- `density`: The kernel density estimate at each grid point
- `ndensity`: Density estimate scaled to maximum of 1 within each group
- `count`: Density estimate × number of observations in group (expected count)
- `n`: Number of observations in each group

Grouping

When aesthetics like `colour` or `fill` are mapped to categorical variables, `stat_density_3d()` computes separate density surfaces for each group, just like `stat_density_2d()`. Each group gets its own density calculation with proper `count` and `n` values.

Computed variables

The following computed variables are available via `after_stat()`:

- `x`, `y`, `z`: Grid coordinates and function values
- `normal_x`, `normal_y`, `normal_z`: Surface normal components
- `slope`: Gradient magnitude from surface calculations
- `aspect`: Direction of steepest slope from surface calculations
- `dzdx`, `dzdy`: Partial derivatives from surface calculation

See Also

`ggplot2::stat_density_2d()` for 2D density contours, `stat_surface_3d()` for surfaces from existing grid data, `light()` for lighting specifications, `coord_3d()` for 3D coordinate systems.

Examples

```
# Basic density surface from scattered points
p <- ggplot(faithful, aes(eruptions, waiting)) +
  coord_3d() +
  scale_fill_viridis_c()
p + geom_density_3d() + guides(fill = guide_colorbar_3d())



# Specify alternative grid geometry and light model
p + geom_density_3d(grid = "equilateral", n = 30, direction = "y",
                    light = light("direct"),
                    color = "white", linewidth = .1) +
  guides(fill = guide_colorbar_3d())

# Color by alternative density metric
p + geom_density_3d(aes(fill = after_stat(count)))

# Adjust bandwidth for smoother or more detailed surfaces
p + geom_density_3d(adjust = 0.5, n = 100, color = "white")  # More detail
p + geom_density_3d(adjust = 2, color = "white")   # Smoother

# As contour plot instead of default surface plot
p + stat_density_3d(geom = "contour_3d", light = "none",
                    color = "black", bins = 25,
                    sort_method = "pairwise")

# Multiple density surfaces by group,
# using normalized density to equalize peak heights
ggplot(iris, aes(Petal.Length, Sepal.Length, fill = Species)) +
  geom_density_3d(aes(z = after_stat(ndensity), group = Species),
                  color = "black", alpha = .7, light = NULL) +
  coord_3d()

# Same, but with extra padding to remove edge effects and
```

```
# with density filtering to remove rectangular artifacts
ggplot(iris, aes(Petal.Length, Sepal.Length, fill = Species)) +
  geom_density_3d(aes(z = after_stat(ndensity)),
                  pad = .3, min_ndensity = .001,
                  color = "black", alpha = .7, light = NULL) +
  coord_3d(ratio = c(3, 3, 1))
```

---

geom_hull_3d *3D convex and alpha hulls*

Description

Turns 3D point clouds into surface hulls consisting of triangular polygons, using either convex hull or alpha shape algorithms.

Usage

```
geom_hull_3d(
  mapping = NULL,
  data = NULL,
  stat = StatHull3D,
  position = "identity",
  ...,
  method = "convex",
  radius = NULL,
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
  inherit.aes = TRUE,
  show.legend = TRUE
)

stat_hull_3d(
  mapping = NULL,
  data = NULL,
  geom = GeomPolygon3D,
  position = "identity",
  ...,
  method = "convex",
  radius = NULL,
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
```

```
  inherit.aes = TRUE,
  show.legend = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). The required aesthetics are x, y, and z. Additional aesthetics can use computed variables with ggplot2::after_stat().
data	The data to be displayed in this layer.
stat	The statistical transformation to use on the data. Defaults to StatHull3D.
position	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use position_on_face().
...	Other arguments passed on to the layer function (typically GeomPolygon3D), such as aesthetics like colour, fill, linewidth, annotate = annotate_3d(...), etc.
method	Triangulation method. Either: • "convex": Convex hull triangulation (default). Requires the **geometry** package. • "alpha": Alpha shape triangulation (can capture non-convex topologies)
radius	Square root of "alpha" parameter when alpha method is used. A face is included in the resulting alpha shape if it can be "exposed" by a sphere of this radius. If NULL (the default), a simple heuristic based on the data scale is used to calculate a radius value. Note that alpha shapes are quite sensitive to the coordinate scales of your data. See Details section.
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
sort_method	Depth sorting algorithm. See sorting_methods for details.
scale_depth	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
inherit.aes	If FALSE, overrides the default aesthetics.
show.legend	Logical indicating whether this layer should be included in legends.
geom	The geometric object used to display the data. Defaults to GeomPolygon3D.

Value

A Layer object that can be added to a ggplot.

Grouping

Hulls respect ggplot2 grouping aesthetics. To create separate hulls for different subsets of your data, use `aes(group = category_variable)` or similar grouping aesthetics. Each group will get its own independent hull.

Alpha scale sensitivity

Alpha shape method is highly sensitive to coordinate scales. The `alpha` parameter that works for data scaled 0-1 will likely fail for data scaled 0-1000. Guidelines for choosing radius:

- Start with `alpha = 1.0` and adjust based on results
- For data with mixed scales (e.g., x: 0-1, y: 0-1000), consider rescaling your data first
- Larger alpha values → smoother, more connected surfaces
- Smaller alpha values → more detailed surfaces, but may fragment
- If you get no triangles, try increasing alpha by 10x
- If surface fills unwanted holes, try decreasing alpha by 10x

Aesthetics

`stat_hull_3d()` requires the following aesthetics:

- **x**: X coordinate
- **y**: Y coordinate
- **z**: Z coordinate

Computed variables

- `normal_x`, `normal_y`, `normal_z`: Surface normal components

See Also

`coord_3d()` for 3D coordinate systems, geom_polygon_3d for the default geometry with depth sorting, `light()` for lighting specifications.

Examples

```
# Convex hull
ggplot(sphere_points, aes(x, y, z)) +
  geom_hull_3d(method = "convex", fill = "gray40") +
  coord_3d()

# Alpha shape (for sphere data, gives similar result to convex)

ggplot(sphere_points, aes(x, y, z)) +
  geom_hull_3d(method = "alpha", radius = 2, fill = "gray40") +
  coord_3d()


# Use `cull_backfaces = FALSE` to render far side of hull
```

```
ggplot(sphere_points, aes(x, y, z)) +
  geom_hull_3d( # default culling for comparison
    method = "convex", light = NULL,
    fill = "steelblue", color = "darkred", linewidth = .5, alpha = .5) +
  geom_hull_3d( # culling disabled
    aes(x = x + 2.5), cull_backfaces = FALSE,
    method = "convex", light = NULL,
    fill = "steelblue", color = "darkred", linewidth = .5, alpha = .5) +
  coord_3d(scales = "fixed")

# Use grouping to build separate hulls for data subsets
ggplot(iris, aes(Petal.Length, Sepal.Length, Sepal.Width,
                 color = Species, fill = Species)) +
      geom_hull_3d() +
      coord_3d(scales = "fixed")
```

---

geom_path_3d *3D paths connecting observations*

Description

Connects observations in 3D space in the order they appear in the data. It converts path data into individual segments for proper depth sorting while maintaining the appearance of connected paths. Each path is divided into segments that can be depth-sorted independently.

Usage

```
geom_path_3d(
  mapping = NULL,
  data = NULL,
  stat = StatPath3D,
  position = "identity",
  ...,
  sort_method = "painter",
  scale_depth = TRUE,
  arrow = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_path_3d(
  mapping = NULL,
  data = NULL,
  geom = GeomSegment3D,
  position = "identity",
```

```
  ...,
  sort_method = "painter",
  scale_depth = TRUE,
  arrow = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). Requires x, y, z coordinates. Grouping aesthetics determine separate paths.
data	The data to be displayed in this layer.
stat	The statistical transformation to use on the data. Defaults to "path_3d".
position	Position adjustment, defaults to "identity".
...	Other arguments passed on to ggplot2::layer().
sort_method	Character indicating algorithm used to determine the order in which elements are rendered. This controls depth sorting for all geometry types within a layer, including polygons, points, segments, and text. Default varies by geometry type. • "painter": Elements are sorted by the mean depth (distance from viewer after rotation) of their vertices. This is fast, but can give incorrect results when primitives overlap in screen space at different depths. • "pairwise": A more intensive sorting algorithm that compares every pair of elements to determine occlusion order. Uses type-specific geometric tests: polygon overlap detection for polygon-polygon pairs, point-in-polygon tests for polygon-point pairs, line clipping for polygon-segment pairs, and line intersection for segment-segment pairs. When elements are coplanar, smaller primitives (points, segments) render on top of larger ones (polygons). Slower but more accurate. • "auto": Uses pairwise if the data has fewer than 500 rows, and painter otherwise.
scale_depth	Logical indicating whether to apply depth-based scaling to linewidth. When TRUE (default), path segments closer to the viewer appear thicker, and segments farther away appear thinner.
arrow	Specification for arrow heads, created by arrow().
lineend	Line end style, one of "round", "butt", "square".
na.rm	If FALSE, missing values are removed with a warning.
show.legend	Logical indicating whether this layer should be included in legends.
inherit.aes	If FALSE, overrides the default aesthetics.
geom	The geometric object used to display the data. Defaults to "segment_3d".

Value

A Layer object that can be added to a ggplot.

Aesthetics

`geom_path_3d()` understands the following aesthetics:

- **x, y, z**: Coordinates (required)
- `group`: Grouping variable to create separate paths
- `colour`: Line color
- `linewidth`: Line width (gets depth-scaled when `scale_depth = TRUE`)
- `linetype`: Line type
- `alpha`: Transparency

Computed variables for StatSegment3D

- `x`, `y`, `z`: Start coordinates of each segment
- `xend`, `yend`, `zend`: End coordinates of each segment
- `group`: Hierarchical group identifier preserving original grouping

Grouping

Multiple paths are created based on grouping aesthetics (group, colour, etc.). Each group forms a separate path, and segments from different paths can be interleaved during depth sorting for proper 3D rendering.

See Also

`geom_segment_3d()` for individual segments.

Examples

```
library(ggplot2)

x <- seq(0, 20*pi, pi/16)
spiral <- data.frame(
  x = x,
  y = sin(x),
  z = cos(x))

# Basic path
ggplot(spiral, aes(x, y, z)) +
  geom_path_3d() +
  coord_3d()

# With aesthetic coloring
ggplot(spiral, aes(x, y, z, color = y)) +
  geom_path_3d(linewidth = 1, lineend = "round") +
  coord_3d() +
  scale_color_gradientn(colors = c("red", "purple", "blue"))

# With grouping
ggplot(spiral, aes(x, y, z, color = x > 30)) +
```

```
  geom_path_3d(linewidth = 1, lineend = "round") +
  coord_3d()
```

`geom_point_3d` *3D scatter plot with 2D reference elements*

Description

`geom_point_3d()` creates scatter plots in 3D space with automatic depth-based size scaling. Points closer to the viewer appear larger, while points farther away appear smaller, creating realistic perspective effects. Optionally adds reference lines and points projecting to cube faces, to illustrate point locations in 2D.

Usage

```
geom_point_3d(
  mapping = NULL,
  data = NULL,
  stat = StatPoint3D,
  position = "identity",
  ...,
  na.rm = FALSE,
  sort_method = "painter",
  scale_depth = TRUE,
  raw_points = TRUE,
  ref_lines = FALSE,
  ref_points = FALSE,
  ref_faces = "zmin",
  ref_circle_radius = 1.5,
  ref_circle_vertices = 16,
  ref_line_color = NULL,
  ref_line_colour = NULL,
  ref_line_linewidth = 0.25,
  ref_line_linetype = NULL,
  ref_line_alpha = NULL,
  ref_point_color = NULL,
  ref_point_colour = NULL,
  ref_point_fill = NULL,
  ref_point_size = NULL,
  ref_point_alpha = NULL,
  ref_point_stroke = NULL,
  ref_point_shape = NULL,
  inherit.aes = TRUE,
  show.legend = NA
)
```

```
stat_point_3d(
  mapping = NULL,
  data = NULL,
  geom = GeomPoint3D,
  position = "identity",
  ...,
  na.rm = FALSE,
  sort_method = "painter",
  scale_depth = TRUE,
  raw_points = TRUE,
  ref_lines = FALSE,
  ref_points = FALSE,
  ref_faces = "zmin",
  ref_circle_radius = 1.5,
  ref_circle_vertices = 16,
  ref_line_color = NULL,
  ref_line_colour = NULL,
  ref_line_linewidth = 0.25,
  ref_line_linetype = NULL,
  ref_line_alpha = NULL,
  ref_point_color = NULL,
  ref_point_colour = NULL,
  ref_point_fill = NULL,
  ref_point_size = NULL,
  ref_point_alpha = NULL,
  ref_point_stroke = NULL,
  ref_point_shape = NULL,
  inherit.aes = TRUE,
  show.legend = NA
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). In addition to the standard point aesthetics, geom_point_3d() requires x, y, and z coordinates.
data	The data to be displayed in this layer.
stat	The statistical transformation to use on the data. Defaults to "point_3d".
position	Position adjustment, defaults to "identity".
...	Other arguments passed on to ggplot2::layer().
na.rm	If FALSE, missing values are removed with a warning.
sort_method	Character indicating algorithm used to determine the order in which elements are rendered. This controls depth sorting for all geometry types within a layer, including polygons, points, segments, and text. Default varies by geometry type. • "painter": Elements are sorted by the mean depth (distance from viewer after rotation) of their vertices. This is fast, but can give incorrect results when primitives overlap in screen space at different depths.

- "pairwise": A more intensive sorting algorithm that compares every pair of elements to determine occlusion order. Uses type-specific geometric tests: polygon overlap detection for polygon-polygon pairs, point-in-polygon tests for polygon-point pairs, line clipping for polygon-segment pairs, and line intersection for segment-segment pairs. When elements are coplanar, smaller primitives (points, segments) render on top of larger ones (polygons). Slower but more accurate.
- "auto": Uses pairwise if the data has fewer than 500 rows, and painter otherwise.

scale_depth Logical indicating whether to apply depth-based scaling to point sizes, point stroke widths, and reference line widths. When TRUE (default), points/lines closer to the viewer appear larger/wider, and points farther away appear smaller. When FALSE, all points/lines have uniform size/width.

raw_points Logical indicating whether to show the original 3D points. Default is TRUE.

ref_lines Logical indicating whether to show reference lines projecting from points to cube faces. Default is FALSE.

ref_points Type of reference points to create. Options:

- FALSE: No reference points (default)
- TRUE or "circles": Circular reference points that project properly
- "points": Single-point references (renders faster and supports non-circular shapes, but point shape is not 3D-transformed)

ref_faces Character vector specifying which cube faces to project to. Valid face names are: "xmin", "xmax", "ymin", "ymax", "zmin", "zmax". Default is "zmin" (bottom face). Multiple faces can be specified.

ref_circle_radius
Radius for circular reference points as percentage of standardized coordinate space.

ref_circle_vertices
Number of vertices for circular reference points (higher = smoother).

ref_line_color, ref_line_colour, ref_line_linewidth, ref_line_linetype, ref_line_alpha
Styling overrides for reference lines. When NULL, inherits from point aesthetics or uses defaults. Both American (ref_line_color) and British (ref_line_colour) spellings are accepted.

ref_point_color, ref_point_colour, ref_point_fill, ref_point_size, ref_point_alpha, ref_point_stroke, ref_point_shape
Styling overrides for reference points and circles. When NULL, inherits from raw point aesthetics with shape-aware logic (complex shapes 21-25 vs simple shapes 0-20/characters). Both American (ref_point_color) and British (ref_point_colour) spellings are accepted.

inherit.aes If FALSE, overrides the default aesthetics.

show.legend Logical indicating whether this layer should be included in legends.

geom The geometric object used to display the data. Defaults to "point_3d".

Details

StatPoint3D performs identity transformation (passes data through unchanged) while properly handling discrete scales for 3D coordinate systems. It can optionally generate reference lines and points projecting to cube faces.

Value

A `Layer` object that can be added to a ggplot.

Aesthetics

`geom_point_3d()` supports all the same aesthetics as `ggplot2::geom_point()`, plus z:

- **x**: X coordinate (required)
- **y**: Y coordinate (required)
- **z**: Z coordinate (required)
- `alpha`: Transparency
- `colour`: Point border color
- `fill`: Point fill color (for certain shapes)
- `shape`: Point shape
- `size`: Point size (gets depth-scaled when `scale_depth = TRUE`)
- `stroke`: Border width for shapes with borders

StatPoint3D returns the following computed variables

- `x_raw`, `y_raw`, `z_raw`: Original values before discrete-to-numeric conversion
- `element_type`: Type of element ("raw_point", "ref_point", "ref_line", "ref_circle")
- `segment_id`: ID linking the two endpoints of reference lines
- `ref_face`: Which face reference elements project to
- `ref_circle_radius`: Radius for circular reference points
- `ref_circle_vertices`: Number of vertices for circular reference points

Depth Scaling

The depth scaling uses an inverse relationship with distance, following the mathematical relationship: `apparent_size = base_size * reference_distance / actual_distance`

This creates realistic perspective where:

- Objects twice as far appear half as big
- Objects twice as close appear twice as big
- The center of the plot volume renders at exactly the user-specified size

Point Rendering

ggcube uses shape-aware rendering for improved stroke behavior:

- **Simple shapes (0-20, characters):** Use `size` for fontsize, `stroke` for border width, no fill
- **Complex shapes (21-25):** Use ggplot2's approach: `size + 0.5*stroke` for fontsize to prevent gaps
- Both size and stroke are depth-scaled when `scale_depth = TRUE`
- All shapes preserve stroke depth scaling and parameter control

Reference Features

Reference lines and points help visualize the 3D relationships by projecting data points onto cube faces:

- **Reference lines**: Connect each 3D point to its projection on specified faces
- **Reference points**: Show the projected location on the faces
- **Reference circles**: Circular projections that appear as realistic 3D shadows
- **Depth sorting**: All elements (original points, reference lines, reference points) are automatically depth-sorted for proper 3D rendering

Aesthetic Inheritance

Reference elements intelligently inherit styling from raw points:

- **Shape-aware**: Complex shapes (21-25) with fill/colour are handled differently from simple shapes (0-20/characters)
- **Alpha detection**: If all points have alpha=1, uses 0.5 default for ref elements; otherwise inherits mapped alpha
- **Priority**: Explicit ref_* parameters > mapped aesthetics > smart defaults

See Also

`geom_segment_3d()` for 3D segments, `geom_path_3d()` for 3D paths, `coord_3d()` for 3D coordinate systems.

Examples

```
library(ggplot2)

# Basic 3D scatter plot with depth scaling
ggplot(expand.grid(x = 1:5, y = 1:5, z = 1:5),
       aes(x, y, z, fill = z)) +
  geom_point_3d(size = 10, shape = 21, color = "white", stroke = .1) +
  coord_3d(pitch = 40, roll = 5, yaw = 0, dist = 1.5) +
  scale_fill_viridis_c()

# Add circular reference points on 2D face panel
ggplot(mtcars, aes(mpg, wt, qsec)) +
```

```
  geom_point_3d(size = 3,
    ref_points = TRUE, ref_faces = c("ymax", "xmax")) +
  coord_3d()
```

---

geom_polygon_3d *3D polygon geometry with depth sorting*

Description

geom_polygon_3d() renders 3D polygons with proper depth sorting for realistic 3D surface visualization. It's designed to work with surface data from stat_hull_3d() and stat_surface_3d(), as well as regular polygon data like maps.

Usage

```
geom_polygon_3d(
  mapping = NULL,
  data = NULL,
  stat = StatIdentity3D,
  position = "identity",
  ...,
  rule = "evenodd",
  sort_method = "auto",
  scale_depth = TRUE,
  force_convex = FALSE,
  cull_backfaces = FALSE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes().
data	The data to be displayed in this layer. Note that if you specify light or cull_backfaces, behavior will depend on the "winding order" of polygon vertices, with the counter-clockwise face considered the "front".
stat	The statistical transformation to use on the data. Defaults to "identity_3d".
position	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use position_on_face().
...	Other arguments passed on to the layer function (typically GeomPolygon3D), such as aesthetics like colour, fill, linewidth, annotate = annotate_3d(...), etc.

rule	Either "evenodd" or "winding". If polygons with holes are being drawn (using the subgroup aesthetic) this argument defines how the hole coordinates are interpreted. See ggplot2::geom_polygon() for reference.
sort_method	Depth sorting algorithm. See sorting_methods for details.
scale_depth	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
force_convex	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
na.rm	If FALSE, missing values are removed.
show.legend	Logical indicating whether this layer should be included in legends.
inherit.aes	If FALSE, overrides the default aesthetics.

Value

A Layer object that can be added to a ggplot.

Aesthetics

geom_polygon_3d() requires:

- **x**: X coordinate
- **y**: Y coordinate
- **z**: Z coordinate (for depth sorting)
- **group**: Polygon grouping variable

And understands these additional aesthetics:

- fill: Polygon fill color
- colour: Border color
- linewidth: Border line width
- linetype: Border line type
- alpha: Transparency
- subgroup: Secondary grouping for polygons with holes
- order: Vertex order within polygons (for proper polygon construction)

Examples

```
# Typically used via stats such as stat_surface_3d() or stat_hull_3d()
ggplot(sphere_points, aes(x, y, z)) +
  stat_hull_3d(method = "convex", fill = "dodgerblue",
            light = light(fill = TRUE, mode = "hsl")) +
  coord_3d()

# Can be used directly with properly structured data
triangles <- data.frame(x = rep(c(3, 2, 1), 3),
                            y = rep(c(1, 3, 1), 3),
                            z = rep(1:3, each = 3),
                            shape = rep(letters[1:3], each = 3))
ggplot(triangles, aes(x, y, z, fill = shape)) +
  geom_polygon_3d(color = "black") +
  coord_3d()

# Use `sort_method` to choose between depth sorting algorithms
d <- data.frame(group = rep(letters[1:3], each = 4),
                x = c(1, 1, 2, 2,   1, 1, 3, 3,   2, 2, 3, 3),
                y = rep(c(1, 2, 2, 1), 3),
                z = rep(c(1, 1.5, 2), each = 4))
p <- ggplot(d, aes(x, y, z, group = group, fill = group)) +
      coord_3d(pitch = 50, roll = 20, yaw = 0,
               scales = "fixed", light = "none") +
      theme_light()

# fast, but rendering order is incorrect in this particular example
p + geom_polygon_3d(color = "black", linewidth = 1, alpha = .75,
      sort_method = "painter")

# correct rendering order (but slower for large data sets)
p + geom_polygon_3d(color = "black", linewidth = 1, alpha = .75,
      sort_method = "pairwise")
```

`geom_ridgeline_3d` *3D ridgeline plot from point grid*

Description

Renders a point grid as ridgeline polygons (Joy Division style). Each slice along one axis becomes a filled polygon, creating a cross-sectional view of the surface.

Usage

```
geom_ridgeline_3d(
  mapping = NULL,
  data = NULL,
  stat = "identity_3d",
```

```
  position = "identity",
  ...,
  direction = "x",
  base = NULL,
  cull_backfaces = FALSE,
  sort_method = "pairwise",
  scale_depth = TRUE,
  force_convex = FALSE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes().
data	Point grid data with x, y, z coordinates.
stat	Statistical transformation. Defaults to "identity".
position	Position adjustment, defaults to "identity".
...	Other arguments passed to the layer.
direction	Direction of ridges: **"x"** One ridge per unique x value; ridge varies in y (default) **"y"** One ridge per unique y value; ridge varies in x
base	Z-value for ridge polygon bottoms. If NULL, uses min(z).
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
sort_method	Depth sorting algorithm. See sorting_methods for details.
scale_depth	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
force_convex	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
na.rm	If FALSE, missing values are removed.

show.legend	Logical indicating whether this layer should be included in legends.
inherit.aes	If FALSE, overrides the default aesthetics.

Value

A Layer object that can be added to a ggplot.

Aesthetics

x, y, z Point coordinates (required)

fill Polygon fill color (default: "grey70")

colour Polygon border color (default: "black")

alpha Transparency

linewidth Border width

linetype Border type

See Also

geom_surface_3d(), stat_function_3d(), coord_3d()

Examples

```
# From point grid
grid_data <- expand.grid(x = seq(-3, 3, 0.5), y = seq(-3, 3, 0.1))
grid_data$z <- with(grid_data, dnorm(y) * dnorm(x) * 10)

ggplot(grid_data, aes(x, y, z = z)) +
  geom_ridgeline_3d() +
  coord_3d()

# With fill
ggplot(grid_data, aes(x, y, z = z, fill = x)) +
  geom_ridgeline_3d(colour = "white", linewidth = 0.3) +
  scale_fill_viridis_c() +
  coord_3d()

# Ridges in y direction
ggplot(grid_data, aes(x, y, z = z)) +
  geom_ridgeline_3d(direction = "y", fill = "steelblue") +
  coord_3d()

# With stat_function_3d
ggplot() +
  stat_function_3d(fun = function(x, y) sin(x) * cos(y),
                   xlim = c(-pi, pi), ylim = c(-pi, pi),
                   n = 30,
                   geom = "ridgeline_3d") +
  coord_3d()
```

geom_segment_3d *3D line segments*

Description

geom_segment_3d() and stat_segment_3d() draw line segments in 3D space with automatic depth-based linewidth scaling and proper depth sorting. Each segment is defined by start coordinates (x, y, z) and end coordinates (xend, yend, zend).

Usage

```
geom_segment_3d(
  mapping = NULL,
  data = NULL,
  stat = StatSegment3D,
  position = "identity",
  ...,
  sort_method = "painter",
  scale_depth = TRUE,
  arrow = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_segment_3d(
  mapping = NULL,
  data = NULL,
  geom = GeomSegment3D,
  position = "identity",
  ...,
  sort_method = "painter",
  scale_depth = TRUE,
  arrow = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). Requires x, y, z for start coordinates and xend, yend, zend for end coordinates.
data	The data to be displayed in this layer.

stat	The statistical transformation to use on the data. Defaults to "segment_3d".
position	Position adjustment, defaults to "identity".
...	Other arguments passed on to ggplot2::layer().
sort_method	Character indicating algorithm used to determine the order in which elements are rendered. This controls depth sorting for all geometry types within a layer, including polygons, points, segments, and text. Default varies by geometry type. • "painter": Elements are sorted by the mean depth (distance from viewer after rotation) of their vertices. This is fast, but can give incorrect results when primitives overlap in screen space at different depths. • "pairwise": A more intensive sorting algorithm that compares every pair of elements to determine occlusion order. Uses type-specific geometric tests: polygon overlap detection for polygon-polygon pairs, point-in-polygon tests for polygon-point pairs, line clipping for polygon-segment pairs, and line intersection for segment-segment pairs. When elements are coplanar, smaller primitives (points, segments) render on top of larger ones (polygons). Slower but more accurate. • "auto": Uses pairwise if the data has fewer than 500 rows, and painter otherwise.
scale_depth	Logical indicating whether to apply depth-based scaling to linewidth. When TRUE (default), segments closer to the viewer appear thicker, and segments farther away appear thinner.
arrow	Specification for arrow heads, created by arrow().
lineend	Line end style, one of "round", "butt", "square".
na.rm	If FALSE, missing values are removed with a warning.
show.legend	Logical indicating whether this layer should be included in legends.
inherit.aes	If FALSE, overrides the default aesthetics.
geom	The geometric object used to display the data. Defaults to "segment_3d".

Value

A Layer object that can be added to a ggplot.

Aesthetics

geom_segment_3d() understands the following aesthetics:

- **x, y, z**: Start coordinates (required)
- **xend, yend, zend**: End coordinates (required)
- colour: Line color
- linewidth: Line width (gets depth-scaled when scale_depth = TRUE)
- linetype: Line type
- alpha: Transparency

See Also

geom_path_3d() for connected paths.

Examples

```
# Basic 3D segments
ggplot(sphere_points,
       aes(x, y, z, xend = 0, yend = 0, zend = 0)) +
  geom_segment_3d() +
  coord_3d()

# 3D vector field
data <- expand.grid(x = -1:2, y = -1:2, z = -1:2)
data2 <- data + seq(-.5, .5, length.out = length(as.matrix(data)))
data <- cbind(data, setNames(data2, c("x2", "y2", "z2")))
ggplot(data, aes(x, y, z,
       xend = x2, yend = y2, zend = z2, color = x)) +
  geom_segment_3d(arrow = arrow(length = unit(0.1, "inches"),
                  type = "closed", angle = 15),
                  linewidth = .5) +
  coord_3d()
```

---

geom_smooth_3d *3D surface from smoothed conditional means*

Description

A 3D version of ggplot2::geom_smooth(). Creates surfaces by fitting smoothing models to scattered (x,y,z) data points. The fitted statistical model is evaluated on a regular grid and rendered as a 3D surface with optional standard error surfaces.

Usage

```
geom_smooth_3d(
  mapping = NULL,
  data = NULL,
  stat = StatSmooth3D,
  position = "identity",
  ...,
  method = "loess",
  formula = NULL,
  method.args = list(),
  xlim = NULL,
  ylim = NULL,
  n = NULL,
  grid = NULL,
  direction = NULL,
```

```
  trim = NULL,
  domain = c("bbox", "chull"),
  se = FALSE,
  level = 0.95,
  se_fill = NULL,
  se_colour = NULL,
  se_color = NULL,
  se_alpha = 0.5,
  se_linewidth = NULL,
  points = FALSE,
  point_colour = "black",
  point_color = NULL,
  point_fill = NA,
  point_size = 1.5,
  point_shape = 19,
  point_alpha = 1,
  point_stroke = 0.5,
  residuals = FALSE,
  residual_colour = "black",
  residual_color = NULL,
  residual_linewidth = 0.5,
  residual_linetype = 1,
  residual_alpha = 1,
  light = NULL,
  cull_backfaces = FALSE,
  sort_method = NULL,
  force_convex = TRUE,
  scale_depth = TRUE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_smooth_3d(
  mapping = NULL,
  data = NULL,
  geom = GeomPolygon3D,
  position = "identity",
  ...,
  method = "loess",
  formula = NULL,
  method.args = list(),
  xlim = NULL,
  ylim = NULL,
  n = NULL,
  grid = NULL,
  direction = NULL,
  trim = NULL,
```

```
  domain = c("bbox", "chull"),
  se = FALSE,
  level = 0.95,
  se_fill = NULL,
  se_colour = NULL,
  se_color = NULL,
  se_alpha = 0.5,
  se_linewidth = NULL,
  points = FALSE,
  point_colour = "black",
  point_color = NULL,
  point_fill = NA,
  point_size = 1.5,
  point_shape = 19,
  point_alpha = 1,
  point_stroke = 0.5,
  residuals = FALSE,
  residual_colour = "black",
  residual_color = NULL,
  residual_linewidth = 0.5,
  residual_linetype = 1,
  residual_alpha = 1,
  light = NULL,
  cull_backfaces = FALSE,
  sort_method = NULL,
  force_convex = TRUE,
  scale_depth = TRUE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. This stat requires `x`, `y`, and `z` aesthetics from the input data. By default, fill is mapped to `after_stat(fitted)`.
`data`	The data to be displayed in this layer. Must contain x, y, z columns.
`stat`	The statistical transformation to use on the data. Defaults to `StatSmooth3D`.
`position`	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use `position_on_face()`.
`...`	Other arguments passed on to the layer function (typically GeomPolygon3D), such as aesthetics like `colour`, `fill`, `linewidth`, `annotate` = `annotate_3d(...)`, etc.
`method`	Smoothing method to use. Currently supported: • `"loess"` (default): Local polynomial regression • `"lm"`: Linear model • `"glm"`: Generalized linear model

- "gam": Generalized additive model (requires mgcv package)

formula — Model formula. If NULL (default), uses method-appropriate defaults: z ~ x + y for lm and glm, z ~ s(x) + s(y) for gam, auto for loess.

method.args — List of additional arguments passed to the fitting function. For loess, this might include span or degree. For lm, this might include weights. For glm, this might include family (defaults to gaussian()). For gam, this might include smoothing parameters or basis specifications.

xlim, ylim — Numeric vectors of length 2 giving the range for prediction grid. If NULL (default), uses the exact data range with no extrapolation.

grid, n, direction, trim — Parameters determining the geometry, resolution, and orientation of the surface grid. See grid_generation for details.

domain — Character indicating the x-y domain over which to visualize the surface. The default, "bbox", shows predictions over the full rectangular bounding box of the predictors.The alternative, "chull", shows predictions only within the convex hull of the input data, which prevents extrapolation into unoccupied corners of predictor space.

se — Logical indicating whether to display confidence interval bands around the smooth; if TRUE, these are rendered as additional surfaces; they inherit aesthetics from the primary smooth layer unless otherwise specified. Defaults to FALSE.

level — Level of confidence interval to use (0.95 by default).

se_fill — Fill colour for confidence interval bands. If NULL, inherits from the main surface fill aesthetic.

se_colour, se_color — Color for confidence interval band borders. If NULL, inherits from the main surface color aesthetic.

se_alpha — Alpha transparency for confidence interval bands. Defaults to 0.5.

se_linewidth — Line width for confidence interval band borders. If NULL, inherits from the main surface linewidth aesthetic.

points — Logical indicating whether to overlay the original data points on the fitted surface. Points are depth-sorted together with the surface polygons for proper 3D rendering. Point styling is controlled via the point_* parameters; mapped aesthetics from the layer are not inherited by annotation points. Default is FALSE.

point_colour, point_color — Color for data points. Defaults to "black".

point_fill — Fill color for data points (only relevant for shapes 21-25). Defaults to NA (transparent).

point_size — Size of data points.

point_shape — Shape of data points.

point_alpha — Alpha transparency for data points.

point_stroke — Stroke width for data points.

`residuals` Logical indicating whether to draw residual lines connecting data points to the fitted surface. Default is `FALSE`. Residual styling is controlled via the `residual_*` parameters. Note: residual lines may render incorrectly when combined with `se = TRUE`, as the lines intersect the confidence interval surfaces and cannot be split at intersection points.

`residual_colour`, `residual_color`
Color for residual lines.

`residual_linewidth`
Line width for residual lines.

`residual_linetype`
Line type for residual lines.

`residual_alpha` Alpha transparency for residual lines.

`light` A lighting specification object created by `light()`,`"none"` to disable lighting, or `NULL` to inherit plot-level lighting specs from the coord. Specify plot-level lighting in `coord_3d()` and layer-specific lighting in `geom_*3d()` functions.

`cull_backfaces` Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).

`sort_method` Depth sorting algorithm. See sorting_methods for details.

`force_convex` Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.

`scale_depth` Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.

`na.rm` If `FALSE`, missing values are removed.

`show.legend` Logical indicating whether this layer should be included in legends.

`inherit.aes` If `FALSE`, overrides the default aesthetics.

`geom` The geometric object used to display the data. Defaults to `GeomPolygon3D`.

Value

A `Layer` object that can be added to a ggplot.

Aesthetics

`stat_smooth_3d()` requires the following aesthetics from input data:

- **x**: X coordinate
- **y**: Y coordinate
- **z**: Z coordinate (response variable to be smoothed)

Computed variables specific to StatSmooth3D

- `level`: Type of surface ("fitted", "upper CI", or "lower CI" for confidence bands)
- `fitted`: Smoothed predictions (same as `z` when `level == "fitted"`)
- `se`: Standard errors of the fitted values (available when `se = TRUE`)

Computed variables

The following computed variables are available via `after_stat()`:

- `x`, `y`, `z`: Grid coordinates and function values
- `normal_x`, `normal_y`, `normal_z`: Surface normal components
- `slope`: Gradient magnitude from surface calculations
- `aspect`: Direction of steepest slope from surface calculations
- `dzdx`, `dzdy`: Partial derivatives from surface calculation

See Also

`stat_surface_3d()` for surfaces from existing grid data, `stat_function_3d()` for mathematical function surfaces.

Examples

```
# Generate scattered 3D data
set.seed(123)
d <- data.frame(
  x = runif(100, -1, 3),
  y = runif(100, -3, 3)
)
d$z <- abs(1 + d$x^2 - d$y^2 + rnorm(100, 0, 1))

# Base plot
p <- ggplot(d, aes(x, y, z)) +
  coord_3d(light = NULL) +
  scale_fill_viridis_c()

# Basic smooth surface with default loess model
p + geom_smooth_3d()



# Show data points
p + geom_smooth_3d(points = TRUE)

# Show data points with residual lines
p + geom_smooth_3d(points = TRUE, residuals = TRUE,
      point_color = "red", alpha = .8)

# Linear model surface with 90% confidence intervals
p + geom_smooth_3d(aes(fill = after_stat(level)),
```

```
      method = "lm", color = "black", se = TRUE,
      level = 0.99, se_alpha = .7, n = 10) +
      scale_fill_manual(values = c("red", "darkorchid4", "steelblue"))

# Linear model surface with custom model formula
p + geom_smooth_3d(method = "lm", n = 10,
      formula = z ~ poly(x, 2) + poly(y, 2) + x:y)

# Loess with custom span parameter, and lighting effects
p + geom_smooth_3d(
      method = "loess", method.args = list(span = 0.3),
      fill = "steelblue", color = "white", n = 20,
      light = light(direction = c(0, -1, 0), color = FALSE))

# GLM with gamma family and log link
p + geom_smooth_3d(
      method = "glm", n = 10,
      method.args = list(family = Gamma(link = "log")),
      formula = z ~ poly(x, 2) + poly(y, 2))

# Visualize uncertainty with computed "standard error" variable
p + geom_smooth_3d(aes(fill = after_stat(se * 2))) +
  scale_fill_viridis_c()

# Extend surface beyond training data range (explicit extrapolation)
p + geom_smooth_3d(method = "lm", xlim = c(-5, 5), ylim = c(-5, 5))

# Clip surface to predictor convex hull
# to prevent extrapolation into corner areas
p + geom_smooth_3d(method = "lm", domain = "chull")

# Specify alternative grid geometry
p + geom_smooth_3d(grid = "right1", n = 30, direction = "y")

# Separate fits for data subgroups
ggplot(mtcars, aes(wt, mpg, qsec, fill = factor(cyl))) +
  geom_smooth_3d(method = "lm", alpha = .7,
    xlim = c(0, 5), ylim = c(0, 40)) + # specify shared domain
  coord_3d() + theme_light()
```

`geom_text_3d` *3D text labels*

Description

`geom_text_3d()` renders text labels in 3D space. Two rendering methods are available: `"billboard"` (default) renders text as flat labels that always face the camera, while `"polygon"` renders text as filled polygons that can be oriented in any direction.

Usage

```
geom_text_3d(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  method = c("billboard", "polygon"),
  facing = "zmax",
  coord = NULL,
  aspect_adjust = 1,
  light = "none",
  angle = 0,
  nudge_x = 0,
  nudge_y = 0,
  nudge_z = 0,
  size = 3.88,
  hjust = 0.5,
  vjust = 0.5,
  lineheight = 1.2,
  scale_depth = TRUE,
  family = "sans",
  weight = "normal",
  italic = FALSE,
  fontface = "plain",
  spacing = 0,
  tolerance = 0.01,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_text_3d(
  mapping = NULL,
  data = NULL,
  geom = NULL,
  position = "identity",
  ...,
  method = c("billboard", "polygon"),
  facing = "zmax",
  coord = NULL,
  aspect_adjust = 1,
  light = "none",
  angle = 0,
  nudge_x = 0,
  nudge_y = 0,
  nudge_z = 0,
  size = 3.88,
  hjust = 0.5,
```

```
  vjust = 0.5,
  lineheight = 1.2,
  scale_depth = TRUE,
  family = "sans",
  weight = "normal",
  italic = FALSE,
  fontface = "plain",
  spacing = 0,
  tolerance = 0.01,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings. See ggplot2::aes().
data	A data frame containing the variables to plot.
position	Position adjustment. Defaults to "identity".
...	Other arguments passed to the layer.
method	Rendering method: "billboard" (default) for flat camera-facing text, or "polygon" for 3D-oriented filled text.
facing	(Polygon method only) Direction the text should face. One of "xmin", "xmax", "ymin", "ymax", "zmin", "zmax" (default), "camera", a numeric vector of length 3, or a camera_facing() specification. Use "camera" with the coord parameter to automatically face the camera. Ignored with a warning for billboard method.
coord	(Polygon method only) A coord_3d() object. When provided: (1) if facing = "camera", extracts rotation parameters automatically; (2) uses the coord's ratio and scales parameters to correct text aspect ratio so text doesn't appear stretched. Ignored for billboard method.
aspect_adjust	(Polygon method only) Scalar multiplier for text height relative to width. Default is 1 (normal proportions). Values > 1 make text taller, values < 1 make text wider. When coord is provided, this is multiplied by the computed aspect correction. Ignored for billboard method.
light	(Polygon method only) Either a lighting specification created by light(), "none" (the default) for no lighting, or NULL to inherit plot-level lighting from coord_3d(). When provided, text polygons are shaded based on their facing direction. Ignored for billboard method.
angle	Rotation angle in degrees. For polygon method, rotates around the facing axis. For billboard method, rotates in the view plane.
nudge_x, nudge_y, nudge_z	Offset to apply to text position in data units.
size	Font size in mm (same units as ggplot2::geom_text()). Default is 3.88, which corresponds to approximately 11pt text.

hjust, vjust	Horizontal and vertical justification of text (0-1).
lineheight	Line height multiplier for multi-line text.
scale_depth	Logical; if TRUE (default), scale by depth for perspective effect. For billboard method, this scales the font size. For polygon method, this scales the outline linewidth (the polygon fill is always depth-scaled by the 3D projection).
family	Font family.
weight	(Polygon method only) Font weight: "normal", "bold", "thin", "light", etc.
italic	(Polygon method only) Logical; use italic font variant.
fontface	Font face (for billboard method): "plain", "bold", "italic", or "bold.italic".
spacing	(Polygon method only) Additional letter spacing in em units.
tolerance	(Polygon method only) Bezier curve tolerance for text outlines. Lower values give smoother curves but more vertices.
na.rm	If TRUE, silently remove missing values.
show.legend	Logical. Should this layer be included in the legends?
inherit.aes	If TRUE, inherit aesthetics from the plot's default.
geom	The geometric object to use. Defaults based on method.

Value

A ggplot2 layer

Methods

- `"billboard"` (default): Fast, simple text rendering using native fonts. Text is always parallel to the view plane (like a billboard). Only uses the `colour` aesthetic for text color.
- `"polygon"`: Renders text as filled polygon outlines. Text can be oriented to face any direction using the `facing` parameter. Uses both `fill` (text fill) and `colour` (text outline) aesthetics.

Orientation (polygon method only)

The `facing` parameter controls which direction the text faces:

- `"zmax"` (default): Text faces upward (readable from above)
- `"xmin"`, `"xmax"`, `"ymin"`, `"ymax"`, `"zmin"`: Text faces toward the specified cube face
- A numeric vector `c(x, y, z)`: Text faces the specified direction
- `camera_facing(...)`: Text faces the camera

The `angle` parameter rotates the text around its facing axis (in degrees). With `angle = 0`, the text baseline is oriented as horizontally as possible (parallel to the x-y plane when the facing direction allows).

Camera-facing text

For billboard method, text automatically faces the camera - no configuration needed.

For polygon method, there are two ways to make text face the camera:

1. Use `facing = "camera"` with the `coord` parameter (recommended):

```
my_coord <- coord_3d(pitch = 20, roll = -60, yaw = -30)
ggplot(df, aes(x, y, z = z, label = label)) +
  geom_text_3d(method = "polygon", facing = "camera", coord = my_coord) +
  my_coord
```

1. Pass `camera_facing()` directly to `facing`:

```
ggplot(df, aes(x, y, z = z, label = label)) +
  geom_text_3d(method = "polygon",
               facing = camera_facing(pitch = 20, roll = -60, yaw = -30)) +
  coord_3d(pitch = 20, roll = -60, yaw = -30)
```

Aspect Ratio Correction

For polygon method, text can appear stretched if the x, y, and z axes have different data ranges. Providing a `coord_3d()` object via the `coord` parameter enables automatic aspect ratio correction based on the coord's `ratio` and `scales` settings.

If other layers in your plot have different data ranges that affect the final scale limits, you can fine-tune the correction with `aspect_adjust`:

- `aspect_adjust = 1` (default): Use the computed correction
- `aspect_adjust > 1`: Make text taller
- `aspect_adjust < 1`: Make text wider

Size and Scaling

Text size is specified in mm via the `size` parameter, matching the units used by `ggplot2::geom_text()`. The default size of 3.88 corresponds to approximately 11pt text.

Both methods use the same size units, so switching between `"billboard"` and `"polygon"` methods with the same `size` value will produce similarly sized text (assuming a typical 6 inch plot).

When `scale_depth = TRUE` (the default), text farther from the viewer appears smaller, creating a natural perspective effect. For billboard method, this scales the font size. For polygon method, this scales the outline linewidth (the polygon fill is always depth-scaled by the 3D projection).

Aesthetics

`geom_text_3d()` understands the following aesthetics (required in bold):

Both methods:

- **x**, **y**, **z**, **label**
- `alpha`

Billboard method only:

- `colour` - text color

Polygon method only:

- `fill` - text fill color
- `colour` - text outline color
- `linewidth` - outline thickness
- `linetype` - outline line type
- `group`

Other text properties (`size`, `hjust`, `vjust`, `angle`, `family`, `fontface`, `lineheight`, and for the polygon method `weight`, `italic`, `spacing`, `tolerance`) are currently accepted only as fixed values passed as arguments to `geom_text_3d()`, not as mappable aesthetics in `aes()`.

See Also

`camera_facing()` for camera-facing specifications, `text_outlines()` for text-to-polygon conversion

Examples

```
df <- expand.grid(x = c("B", "H", "N"), y = c("a", "o", "u"), z = c("g", "t"))
df$label <- paste0(df$x, df$y, df$z)

# Billboard text (default) - automatically faces camera
ggplot(df, aes(x, y, z, label = label)) +
  geom_text_3d() +
  coord_3d(scales = "fixed")

# Polygon text - can face any direction
ggplot(df, aes(x, y, z, label = label, fill = z)) +
  geom_text_3d(method = "polygon", facing = "zmax") +
  coord_3d(scales = "fixed")

# Polygon text facing camera (using coord parameter)
my_coord <- coord_3d(pitch = 0, roll = -60, yaw = -30, scales = "fixed")
ggplot(df, aes(x, y, z, label = label)) +
  geom_text_3d(method = "polygon", facing = "camera", coord = my_coord,
               color = "red", linewidth = .1) +
  my_coord

# Text with styling
ggplot(df, aes(x, y, z, label = label, fill = factor(x))) +
  geom_text_3d(method = "polygon", facing = "xmin",
               family = "serif", weight = "bold", italic = TRUE,
               size = 6, aspect_adjust = 2) +
  coord_3d(scales = "fixed")
```

geom_voxel_3d *3D voxel visualization from sparse 3D data*

Description

Creates 3D voxel visualizations from sparse 3D point data. Each data point becomes a fixed-size cube centered on its coordinates. Useful for volumetric data and 3D pixel art.

Usage

```
geom_voxel_3d(
  mapping = NULL,
  data = NULL,
  stat = StatVoxel3D,
  position = "identity",
  ...,
  width = 1,
  faces = "all",
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_voxel_3d(
  mapping = NULL,
  data = NULL,
  geom = GeomPolygon3D,
  position = "identity",
  ...,
  width = 1,
  faces = "all",
  light = NULL,
  cull_backfaces = TRUE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes().
data	The data to be displayed in this layer.
stat	The statistical transformation to use on the data. Defaults to StatVoxel3D.
position	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use position_on_face().
...	Other arguments passed on to the layer function (typically GeomPolygon3D), such as aesthetics like colour, fill, linewidth, annotate = annotate_3d(...), etc.
width	Numeric value controlling box width as a fraction of grid spacing. Default is 1.0 (volumes touch each other). Use 0.8 for small gaps, 1.2 for overlap. Grid spacing is determined automatically using ggplot2::resolution()
faces	Character vector specifying which faces to render. Options: • "all" (default): Render all 6 faces • "none": Render no faces • Vector of face names: c("zmax", "xmin", "ymax"), etc. Valid face names: "xmin", "xmax", "ymin", "ymax", "zmin", "zmax". Note that this setting acts jointly with backface culling, which removes faces whose interior faces the viewer – e.g., when cull_backfaces = TRUE and faces = "all" (the default), only front faces are rendered.
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
sort_method	Depth sorting algorithm. See sorting_methods for details.
scale_depth	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
force_convex	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.
na.rm	If FALSE, missing values are removed.
show.legend	Logical indicating whether this layer should be included in legends.
inherit.aes	If FALSE, overrides the default aesthetics.
geom	The geometric object used to display the data. Defaults to GeomPolygon3D.

Details

Note that voxel geometries sometimes require pairwise depth sorting for correct rendering. This is the default for smaller data sets, but not for larger data sets due to compute speed; in those cases you may wish to manually specify `sort_method = "pairwise"`.

Value

A `Layer` object that can be added to a ggplot.

Aesthetics

Voxel 3D requires the following aesthetics:

- **x**: X coordinate (voxel center position)
- **y**: Y coordinate (voxel center position)
- **z**: Z coordinate (voxel center position)

Computed variables

- `normal_x`, `normal_y`, `normal_z`: Face normal components
- `voxel_id`: Sequential voxel number
- `face_type`: Face name ("zmax", "xmin", etc.)

See Also

`stat_col_3d()` for variable-height columns, `stat_surface_3d()` for smooth surfaces, `coord_3d()` for 3D coordinate systems, `light()` for lighting specifications.

Examples

```
# Sparse 3D voxel data
voxel_data <- data.frame(
  x = round(rnorm(100, 0, 2)),
  y = round(rnorm(100, 0, 2)),
  z = round(rnorm(100, 0, 2))
)

p <- ggplot(voxel_data, aes(x, y, z)) + coord_3d()

# Basic 3D voxel plot
p + geom_voxel_3d(fill = "steelblue")

# With aesthetic fill
p + stat_voxel_3d(aes(fill = z)) +
  scale_fill_viridis_c() +
  guides(fill = guide_colorbar_3d())

# Show only some voxel faces
p + geom_voxel_3d(fill = "steelblue",
                  faces = c("zmax", "ymin", "xmin"))
```

grid_generation *Grid generation*

Description

Parameters defining the geometry, resolution, and orientation of a regular grid of surface tiles, as used by various ggcube layer functions including `geom_smooth_3d()`, `geom_function_3d()`, `geom_density_3d()`, and `geom_surface_3d()`.

Arguments

`grid`	Character specifying tile geometry. Options: `"rectangle"` Rectangular grid (the default). `"right1"` Rectangular grid with each quad split into two right triangles along the bottom-left to top-right diagonal. `"right2"` Like `"right1"` but split along the bottom-right to top-left diagonal. `"equilateral"` Equilateral triangular lattice (tessellated via Delaunay triangulation). Can prevent lighting artifacts where a surface curves past parallel with the sight line.
`n`	Either a single integer specifying grid resolution in both dimensions, or a vector of length 2 specifying `c(nx, ny)` for separate x and y resolutions. Default is `40`. Higher values create smoother surfaces but slower rendering.
`direction`	Either `"x"` (the default) or `"y"`, specifying the orientation of tile rows. Ignored for rectangular grids.
`trim`	Logical. Only relevant for `grid = "equilateral"`. If `TRUE` (default), trims edge vertices so that grid boundaries are straight lines. If `FALSE`, preserves the full lattice, resulting in a grid with irregular edges.

guide_3d *Color guides showing lighting effects*

Description

Creates color guides that show shading variation as gradients within each color. Shows the full range of colors visible when shading is enabled in 3D plots.

Usage

```
guide_colorbar_3d(reverse_shade = FALSE, shade_range = c(-0.5, 0.5), ...)

guide_legend_3d(reverse_shade = FALSE, shade_range = c(-0.5, 0.5), ...)
```

Arguments

reverse_shade	Logical. If TRUE, reverses the lighting gradient direction. By default, shadows are placed on the left, or on the bottom for horizontal colorbars.
shade_range	Length-2 numeric vector in the range -1 to 1, giving the limits of the shading gradient. -1 is full shade, and 1 is full highlight. Default is `c(-.5, 5)`.
...	Additional arguments passed to `guide_colorbar()` or `guide_legend()`.

Details

When fill and color aesthetics map to the same variable (e.g., `aes(fill = z, color = z)`), ggplot2 creates a shared scale with a single guide. In this case, use `guides(fill = guide_*_3d())` to apply shading, **not** `guides(color = guide_*_3d())`, even if your layer uses the color aesthetic. Only use the color guide when color and fill map to different variables and you want separate guides for each.

Value

A guide object that displays shading effects

Examples

```
# continuous `colorbar` guide
ggplot(mountain, aes(x, y, z, fill = z)) +
   stat_surface_3d(light = light(mode = "hsl", direction = c(1, 0, 0))) +
   guides(fill = guide_colorbar_3d()) +
   scale_fill_gradientn(colors = c("tomato", "dodgerblue")) +
   coord_3d()

# discrete `legend` guide
ggplot(mountain, aes(x, y, z, fill = x > .5, group = 1)) +
   stat_surface_3d(light = light(mode = "hsl", direction = c(1, 0, 0))) +
   guides(fill = guide_legend_3d()) +
   coord_3d()
```

---

light *Lighting specification for 3D surface rendering*

Description

Creates a lighting specification object for use with 3D polygon layers. Lighting modifies the brightness of fill and/or base color aesthetics based on surface orientation (i.e., it implements form shadows but not cast shadows). Various options are available to control light qualities and light source location. The result can be passed `coord_3d()` to for application to the whole plot, or to individual `geom_*_3d()` layer functions (which take precedence over plot-level lighting).

Usage

```
light(
  method = "diffuse",
  direction = c(-0.5, 0, 1),
  anchor = "scene",
  position = NULL,
  distance_falloff = FALSE,
  fill = TRUE,
  color = TRUE,
  mode = "hsv",
  contrast = 1,
  backface_scale = -1,
  backface_offset = 0
)
```

Arguments

method	Character string specifying lighting model: • "diffuse": The default. Atmospheric lighting with soft shadows (only surfaces pointing directly away from light source are fully dark; base color occurs on surfaces perpendicular to light) • "direct": Direct lighting with hard shadows (all surfaces angled beyond 90 degrees from light source are fully dark; base color occurs on surfaces angled 45 degrees toward light) • "rgb": Map surface orientation to three dimensional color space. Rather than the light model darkening or brightening base colors as in the other models, this model generates a wholly new set of colors. If this option is selected, parameters like mode and contrast are ignored, but fill, color, and direction still apply.
direction	Numeric vector of length 3 specifying direction in 3D space (x, y, z) that light comes from for directional lighting. Depending on anchor, direction is relative to either the plot axes or the viewing pane. The default is c(-.5, 0, 1), giving diagonal lighting from the upper right edge with default rotation and anchor. At least one value must be non-zero. Values are automatically normalized, so magnitude doesn't matter, only sign and relative magnitude. Direction is specified in visual space (relative to the rendered cube's bounding box), not data space. This argument is ignored if position is provided.
anchor	Character string specifying the reference frame for light direction: • "scene" (default): Light direction is fixed relative to the data/scene. Light rotates with the plot, so regardless of rotation, direction = c(1, 0, 0) lights surfaces facing the "xmax" cube face, for example. • "camera": Light direction is fixed relative to the camera/viewer. Light direction stays in place regardless of rotation, so direction = c(1, 0, 0) lights surfaces facing the right side of the plot, for example. When all rotation parameters are zero, these options give the same result.

`position` Numeric vector of length 3 specifying light source position in data coordinate space for positional lighting. When specified, each face gets its own light direction calculated from the light position to the face center. Mutually exclusive with `direction`. Default is NULL (use directional lighting).

`distance_falloff` Logical indicating whether to apply distance-based intensity falloff for positional lighting using inverse square law (intensity proportional to 1/distance²). Only used when `position` is specified. Default is FALSE.

`fill` Logical indicating whether to apply lighting to fill colors. Default is TRUE.

`color` Logical indicating whether to apply lighting to border/line colors. Default is TRUE.

`mode` Character string specifying color lighting mode:

- `"hsv"`: The default. Modifies *value* component of *HSV* color (fades to bright colors at high end, black at low end).
- `"hsl"`: Modifies *lightness* component of *HSL* color (fades to white at high end, black at low end).

These two options give identical results for grayscale colors.

`contrast` Numeric value greater than zero controlling the intensity of lighting effects. 1.0 (the default) gives full black-to-white range. Values less than 1 give subtler effects, while values greater than 1 give more dramatic effects.

`backface_scale`, `backface_offset` Numeric values that determine how "frontface" light values get modified (scaled and then offset) to derive "backface" light values. A backface is the side of a polygon that faces the underside of a surface or the inside of a volume. Frontface light values are typically in the range `[-1, 1]` (unless `contrast` is boosted). The default scale of -1 gives backfaces highly contrasting lighting to frontfaces. To light backfaces the same as frontfaces, set scale to 1. To uniformly darken (brighten) all backfaces, use a negative (positive) offset.

Details

Note that light-like effects can also be achieved in some stats by mapping color aesthetics to computed variables such as `after_stat(dzdx)`; see `geom_surface_3d()` for examples.

Value

A `lighting` object that can be passed to polygon-based 3D layers or to `coord_3d()`.

Examples

```
# base plot used in examples
p <- ggplot(sphere_points, aes(x, y, z)) +
  geom_hull_3d(fill = "#9e2602", color = "#5e1600")

# Light qualities ------------------------------------------------------------

# default `"diffuse"` lighting
```

```
p + coord_3d()



# use `"direct"` lighting to apply full shade to unlit surfaces
p + coord_3d(light = light(method = "direct"))

# use `"hsl"` mode to fade highlights to white
p + coord_3d(light = light(mode = "hsl"))

# adjust lighting intensity with `contrast`
p + coord_3d(light = light(mode = "hsl", contrast = 1.5))

# use `"rgb"` lighting to map face orientation to 3D color space
p + coord_3d(light = light(method = "rgb"))


# Lighting targets ----------------------------------------------------------

# use `fill` and `color` to select which aesthetics get modified by light
p + coord_3d(light = light(fill = TRUE, color = FALSE))

# apply lighting to aesthetically mapped colors, with shaded guide to match
p + geom_hull_3d(aes(fill = x, color = x)) +
      scale_fill_viridis_c() +
      scale_color_viridis_c() +
      guides(fill = guide_colorbar_3d()) +
      coord_3d(light = light(mode = "hsl", contrast = .9))

# disable lighting entirely
# (equivalent to specifying `light(fill = FALSE, color = FALSE`))
p + coord_3d(light = "none")

# if provided, layer-level lighting overrides plot-level (coord_3d) lighting
p + coord_3d(light = light("direct"), # plot-level: affects original layer
      scales = "fixed") +
  geom_hull_3d(aes(x = x + 2.5), fill = "#9e2602", color = "#5e1600",
      light = light("direct", mode = "hsl", direction = c(0, -1, 0)))


# Light sources -------------------------------------------------------------

# directional light from the xmin-ymin-zmin direction
# (`direction` is relative to rotated axes with default `anchor = "scene"`)
p + coord_3d(light = light(direction = c(-1, -1, -1)))

# directional light from top-right corner of figure
# (`anchor = "camera"` makes `direction` fixed relative to the plot)
p + coord_3d(light = light(direction = c(1, 1, 0), anchor = "camera"))

# positional light source within plot
ggplot(mountain, aes(x, y, z)) +
  stat_surface_3d(fill = "red", color = "red") +
```

```
  coord_3d(
    light = light(position = c(.5, .7, 95), distance_falloff = TRUE,
                  mode = "hsl", contrast = .9),
    ratio = c(1.5, 2, 1))


# Backface lighting ----------------------------------------------------------

# backfaces get "opposite" lighting by default (`backface_scale = -1`)
p <- ggplot() +
  geom_function_3d(fun = function(x, y) x^2 + y^2,
    xlim = c(-3, 3), ylim = c(-3, 3),
    fill = "steelblue", color = "steelblue")
p + coord_3d(pitch = 0, roll = -70, yaw = 0,
             light = light(mode = "hsl"))

# use `backface_scale = 1` to light backfaces as if they're frontfaces
p + coord_3d(pitch = 0, roll = -70, yaw = 0,
             light = light(backface_scale = 1, mode = "hsl"))

# use `backface_offset` to uniformly darken (or lighten) backfaces
p + coord_3d(pitch = 0, roll = -70, yaw = 0,
             light = light(backface_scale = 1, mode = "hsl",
                           backface_offset = -.5))
```

`mountain` *Volcano terrain data*

Description

A terrain data set representing elevational variation over an x-y grid.

Usage

```
mountain
```

Format

`mountain`**:**

A data frame with 1290 rows and 3 columns (x, y, and z)

Source

derived from the `datasets::volcano` data set

`position_on_face` *Position for projecting 2D and 3D layers onto cube faces*

Description

Enables layers to be projected onto 2D faces of the 3D coordinate cube. It can be used to flatten 3D ggcube layers onto a single cube face as a way of visualizing them in 2D, or to add certain natively 2D ggplot2 layers like `geom_density_2d()` or `geom_smooth()` to a cube face.

Usage

```
position_on_face(faces = "zmin", axes = NULL)
```

Arguments

`faces`	Character string or vector specifying which cube face(s) to project onto. Valid options are: "xmin", "xmax", "ymin", "ymax", "zmin", "zmax", "3D". "3D" indicates the raw, non-flattened 3D position. Multiple faces and "3D" are only supported for 3D layers (when `axes = NULL`).
`axes`	For 2D layers only: Character vector of length 2 specifying which 3D dimensions the 2D layer's x and y aesthetics represent. For example, `c("x", "z")` means the 2D x-axis maps to the 3D x-axis and the 2D y-axis maps to the 3D z-axis. For 3D layers, use `NULL` (default) to preserve the existing x,y,z mapping.

Details

This position adjustment supports both 2D and 3D layers:

For 3D layers (`axes = NULL`):

- Data already has x,y,z coordinates in the correct order
- Simply adds projection metadata for coord_3d to place the layer on the specified face(s)
- The face coordinate will be overridden during coordinate transformation
- Multiple faces are supported - the layer will be duplicated on each specified face. All specified faces inherit aesthetics from the layer function; if you want different parameters for different faces and on the primary 3D layer, add each as a separate layer call.

For 2D layers (`axes = c("dim1", "dim2")`):

- Renames the layer's x,y columns to match the specified 3D axes
- Adds the missing third dimension
- Adds projection metadata for coord_3d
- Only single faces are supported for 2D layers

The actual projection happens during coordinate transformation in coord_3d.

Compatibility Note: This position adjustment is not compatible with all 2D stats. It works well with `stat_density_2d()` and other stats that don't depend heavily on scale ranges during computation, but may cause errors or rendering issues with `stat_density_2d_filled()` and similar stats that generate polygons based on scale domains, as well as with layers like `stat_bin_2d()` that return position variables other than x and y. For `geom_smooth()`, this position works only if you set `se = FALSE` as shown in the example.

Value

A ggplot position object that can be supplied to the `position` argument to layer functions.

See Also

`coord_3d()` for 3D coordinate systems, `ggplot2::stat_density_2d()` and other 2D statistical transformations that can be projected onto faces.

Examples

```
# 3D point layer in raw 3D form, and projected onto 2D face
ggplot(sphere_points, aes(x, y, z)) +
  geom_point_3d(position = position_on_face("zmin"), color = "red") +
  geom_point_3d(color = "black") + # add this layer last so it appears on top
  coord_3d()

# 3D layer projected to multiple faces
set.seed(1)
d <- data.frame(x = round(rnorm(10)), y = round(rnorm(10)), z = round(rnorm(10)))
ggplot(d, aes(x, y, z)) +
  stat_voxel_3d(color = "black", fill = "steelblue",
    light = light(direction = c(1, 1, 0), mode = "hsv"),
    position = position_on_face(c("3D", "zmin", "xmax", "ymax"))) +
  coord_3d()

# 3D layer projected differently on individual faces
ggplot(sphere_points, aes(x, y, z)) +
  stat_hull_3d(position = position_on_face("zmin"), fill = "black") +
  geom_point_3d(position = position_on_face("ymax")) +
  geom_path_3d(position = position_on_face("xmax")) +
  stat_hull_3d(color = "black") +
  coord_3d()

# 2D density contour on a specific face
ggplot(iris, aes(Sepal.Length, Sepal.Width, Petal.Length, color = Species)) +
  stat_density_2d(position = position_on_face(faces = "zmin", axes = c("x", "y"))) +
  geom_point_3d() +
  coord_3d()

# Distinct 2D layers projected to different faces
ggplot(mtcars) +
  geom_smooth(aes(mpg, qsec), color = "red", alpha = .5, se = FALSE,
    position = position_on_face(faces = "ymax", axes = c("x", "z"))) +
```

```
  geom_density_2d(aes(mpg, wt), alpha = .5,
    position = position_on_face(faces = "zmin", axes = c("x", "y"))) +
  geom_path(aes(wt, qsec), color = "forestgreen", alpha = .5,
    position = position_on_face(faces = "xmax", axes = c("y", "z"))) +
  geom_point_3d(aes(mpg, wt, qsec)) +
    coord_3d() +
  theme_light()
```

`renderers_3d` *Animation renderers for animate_3d*

Description

Renderer functions that combine individual frame images into a final animation file. These are factory functions that return the actual rendering function.

Usage

```
gifski_renderer_3d(file = NULL, loop = TRUE)

av_renderer_3d(file = NULL, vfilter = "null", codec = NULL)

file_renderer_3d(dir = tempdir(), prefix = "ggcube_frame", overwrite = FALSE)
```

Arguments

`file`	Output file path. If `NULL`, a temporary file is used.
`loop`	Logical. Should the GIF loop? Default is `TRUE`.
`vfilter`	A video filter string for ffmpeg (passed to **av**).
`codec`	Video codec name. Default lets **av** choose.
`dir`	Directory to copy frame files into. Defaults to a session temporary directory (`tempdir()`). For animations you want to keep, pass an explicit directory.
`prefix`	Filename prefix for copied frame files.
`overwrite`	Logical. Overwrite existing files? Default is `FALSE`.

Value

A function with signature `function(frames, fps, width, height)` that produces the animation output.

Examples

```
p <- ggplot() +
  geom_function_3d(
    fun = function(x, y) sin(x) * cos(y),
    xlim = c(-pi, pi), ylim = c(-pi, pi),
    fill = "steelblue", color = "steelblue") +
  coord_3d()

# GIF output (default)
animate_3d(p, yaw = c(0, 360), renderer = gifski_renderer_3d())

# Video output
animate_3d(p, yaw = c(0, 360), renderer = av_renderer_3d())

# Keep individual frame files in a chosen directory
animate_3d(p, yaw = c(0, 360),
    renderer = file_renderer_3d(dir = file.path(tempdir(), "my_frames")))
```

---

scale_z_continuous *Continuous z-axis scale for 3D plots*

Description

scale_z_continuous creates a continuous scale for the z aesthetic in 3D plots. It works similarly to scale_x_continuous and scale_y_continuous, providing control over axis breaks, labels, limits, and transformations for the z dimension.

Usage

```
scale_z_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  oob = scales::censor,
  na.value = NA_real_,
  transform = "identity",
  guide = "none",
  ...
)
```

Arguments

name	The name of the scale, used as the axis title. Use waiver() for the default, or NULL to omit the title.
breaks	One of: • NULL for no breaks • waiver() for the default breaks • A numeric vector of positions • A function that takes the limits as input and returns breaks as output
minor_breaks	One of: • NULL for no minor breaks • waiver() for the default minor breaks • A numeric vector of positions • A function that takes the limits as input and returns minor breaks as output
n.breaks	An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels.
labels	One of: • NULL for no labels • waiver() for the default labels • A character vector giving labels (must be same length as breaks) • A function that takes the breaks as input and returns labels as output
limits	A numeric vector of length two providing limits of the scale. Use NA to refer to the existing minimum or maximum.
expand	For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes.
oob	One of: • Function that handles limits outside the scale limits (out of bounds). • scales::censor for replacing out of bounds values with NA • scales::squish for squishing out of bounds values into range
na.value	Missing values will be replaced with this value.
transform	The name of a transformation object or the object itself. Default is "identity", but works with standard transform options such as "log10", "sqrt", and "reverse", detailed in the documentation for ggplot2::scale_x_continuous().
guide	A function used to create a guide or its name. Since z-axis guides are not yet supported, this defaults to "none".
...	Other arguments passed on to continuous_scale().

Value

A ggplot2 scale object for the z aesthetic.

See Also

zlim for a shorthand way to set z-axis limits, coord_3d for the 3D coordinate system

Other 3D scale functions: scale_z_discrete(), zlim()

Examples

```
# Custom breaks, labels, and limits
ggplot(mtcars, aes(mpg, wt, z = qsec)) +
  geom_point() +
  scale_z_continuous(
    breaks = c(15, 17, 19, 21),
    labels = c("Fast", "Medium", "Slow", "Very Slow"),
    limits = c(10, NA)) +
  coord_3d()

# Works with standard scale transformations like "reverse", "log10", etc.
ggplot(mtcars, aes(mpg, wt, z = qsec)) +
  geom_point() +
  scale_z_continuous(transform = "reverse") +
  coord_3d()
```

---

scale_z_discrete *Discrete z-axis scale for 3D plots*

Description

scale_z_discrete creates a discrete scale for the z aesthetic in 3D plots. It works with categorical/factor data, positioning each level at integer coordinates (1, 2, 3, ...) in 3D space.

Usage

```
scale_z_discrete(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  guide = "none",
  na.translate = TRUE,
  na.value = NA_real_,
  drop = TRUE,
  ...
)
```

Arguments

name	The name of the scale, used as the axis title. Use `waiver()` for the default, or `NULL` to omit the title.
breaks	One of: • `NULL` for no breaks • `waiver()` for all breaks • A character vector of breaks • A function that takes the limits as input and returns breaks as output
labels	One of: • `NULL` for no labels • `waiver()` for the default labels • A character vector giving labels (must be same length as breaks) • A function that takes the breaks as input and returns labels as output
limits	A character vector that defines possible values of the scale and their order.
expand	A numeric vector of length two giving multiplicative and additive expansion constants. These constants ensure that data is placed some distance away from the axes. The defaults are to expand by 0.6 units on each side for discrete scales.
guide	A function used to create a guide or its name. Since z-axis guides are not yet supported, this defaults to `"none"`.
na.translate	Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.
na.value	If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.
drop	Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` uses all the levels in the factor.
...	Other arguments passed on to `discrete_scale()`.

Value

A ggplot2 scale object for the z aesthetic.

See Also

`scale_z_continuous` for continuous z-axis scaling, `coord_3d` for the 3D coordinate system

Other 3D scale functions: `scale_z_continuous()`, `zlim()`

Examples

```
library(ggplot2)

# Basic usage to control order, breaks, labels, expansion, etc.
ggplot(mpg, aes(displ, cty, drv, color = drv)) +
```

```
  geom_point() +
  scale_z_discrete(limits = c("f", "r", "4"), # change default order
                   breaks = c("f", "r", "4"),
                   labels = c("front", "rear", "4-wheel"),
                   expand = expansion(.5)) +
  coord_3d()
```

`sorting_methods` *Depth sorting method*

Description

Depth sorting method

Arguments

`sort_method`	Character indicating algorithm used to determine the order in which elements are rendered. This controls depth sorting for all geometry types within a layer, including polygons, points, segments, and text. Default varies by geometry type. • `"painter"`: Elements are sorted by the mean depth (distance from viewer after rotation) of their vertices. This is fast, but can give incorrect results when primitives overlap in screen space at different depths. • `"pairwise"`: A more intensive sorting algorithm that compares every pair of elements to determine occlusion order. Uses type-specific geometric tests: polygon overlap detection for polygon-polygon pairs, point-in-polygon tests for polygon-point pairs, line clipping for polygon-segment pairs, and line intersection for segment-segment pairs. When elements are coplanar, smaller primitives (points, segments) render on top of larger ones (polygons). Slower but more accurate. • `"auto"`: Uses pairwise if the data has fewer than 500 rows, and painter otherwise.

`sphere_points` *Sphere surface points*

Description

A toy data set representing points on a sphere surface, useful for testing hull functions

Usage

```
sphere_points
```

Format

`sphere_points`**:**
A data frame with 200 rows and 3 columns (x, y, and z)

stat_distributions_3d *3D ridgeline distributions*

Description

Computes 1D kernel density estimates for each group and arranges them as ridgeline polygons in 3D space. Similar to `ggridges::geom_density_ridges()`, but rendered as 3D surfaces using `geom_ridgeline_3d()`.

Usage

```
stat_distributions_3d(
  mapping = NULL,
  data = NULL,
  geom = "ridgeline_3d",
  position = "identity",
  ...,
  direction = NULL,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  n = 512,
  trim = FALSE,
  bounds = c(-Inf, Inf),
  rel_min_height = 0,
  joint_bandwidth = FALSE,
  base = 0,
  light = NULL,
  cull_backfaces = FALSE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_distributions_3d(
  mapping = NULL,
  data = NULL,
  stat = "distributions_3d",
  position = "identity",
  ...,
  direction = NULL,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
```

```
  n = 512,
  trim = FALSE,
  bounds = c(-Inf, Inf),
  rel_min_height = 0,
  joint_bandwidth = FALSE,
  base = 0,
  light = NULL,
  cull_backfaces = FALSE,
  sort_method = NULL,
  scale_depth = TRUE,
  force_convex = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). This stat requires x and y aesthetics. One of these serves as the grouping/position variable (determined by direction), and the other provides values for density estimation.
data	The data to be displayed in this layer.
geom	The geometric object to use to display the data. Defaults to geom_ridgeline_3d().
position	Position adjustment, defaults to "identity". To collapse the result onto one 2D surface, use position_on_face().
...	Other arguments passed to the layer.
direction	Direction of ridges: **"x"** One ridge per unique x value; ridge varies in y (default) **"y"** One ridge per unique y value; ridge varies in x
bw	The smoothing bandwidth to be used. If numeric, the standard deviation of the smoothing kernel. If character, a rule to choose the bandwidth, as listed in stats::bw.nrd(). Options include "nrd0" (default), "nrd", "ucv", "bcv", "SJ", "SJ-ste", and "SJ-dpi". Note that automatic calculation is performed per-group unless joint_bandwidth = TRUE.
adjust	A multiplicative bandwidth adjustment. This makes it possible to adjust the bandwidth while still using a bandwidth estimator. For example, adjust = 1/2 means use half of the default bandwidth. Default is 1.
kernel	Kernel function to use. One of "gaussian" (default), "rectangular", "triangular", "epanechnikov", "biweight", "cosine", or "optcosine". See stats::density() for details.
n	Number of equally spaced points at which the density is estimated. Should be a power of two for efficiency. Default is 512.
trim	If FALSE (the default), each density is computed on the full range of the data (extended by a factor based on bandwidth). If TRUE, each density is computed over the range of that group only.

bounds — A numeric vector of length 2 giving the lower and upper bounds for bounded density estimation. Density values outside bounds are set to zero. Data points outside bounds are removed with a warning. Default is `c(-Inf, Inf)` (unbounded).

rel_min_height — Lines with heights below this cutoff will be removed. The cutoff is measured relative to the maximum height within each group. For example, `rel_min_height = 0.01` removes points with density less than 1% of the peak. Default is 0 (no removal). This is similar to the parameter of the same name in `ggridges::geom_density_ridges()`.

joint_bandwidth — If `TRUE`, bandwidth is computed jointly across all groups using the specified `bw` method, ensuring consistent smoothing across all density curves. This matches the behavior of `ggridges::stat_density_ridges()`. If `FALSE` (the default), bandwidth is computed separately for each group, matching `ggplot2::stat_density()`. Only applies when `bw` is a character string (bandwidth rule), not when `bw` is provided as a numeric value.

base — Z-value for ridge polygon bottoms. If NULL, uses min(z).

light — A lighting specification object created by `light()`,`"none"` to disable lighting, or `NULL` to inherit plot-level lighting specs from the coord. Specify plot-level lighting in `coord_3d()` and layer-specific lighting in `geom_*3d()` functions.

cull_backfaces — Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).

sort_method — Depth sorting algorithm. See sorting_methods for details.

scale_depth — Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.

force_convex — Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.

na.rm — If `FALSE`, missing values are removed.

show.legend — Logical indicating whether this layer should be included in legends.

inherit.aes — If `FALSE`, overrides the default aesthetics.

stat — Statistical transformation to use on the data. Defaults to `stat_distributions_3d()`.

Details

This stat is modeled after `ggplot2::stat_density()`, with similar parametrization for bandwidth selection, kernel choice, and boundary handling.

Value

A `Layer` object that can be added to a ggplot.

Aesthetics

`stat_distributions_3d()` understands the following aesthetics (required aesthetics are in bold):

x X coordinate - either density variable or position variable depending on `direction`

y Y coordinate - either position variable or density variable depending on `direction`

group Grouping variable (typically derived from the position aesthetic)

fill, colour, alpha, linewidth, linetype Passed to `geom_ridgeline_3d()`

Direction

The `direction` parameter determines how the data is interpreted:

`direction = NULL` **(default)** Automatically detects direction based on whether `x` or `y` is discrete (factor/character). If `x` is discrete and `y` is continuous, uses `"x"`; if `y` is discrete and `x` is continuous, uses `"y"`. Falls back to `"x"` if ambiguous.

`direction = "x"` Ridges march along the x-axis. Each unique x value defines a group, and density is computed from the y values within that group. The resulting density curves lie in the y-z plane.

`direction = "y"` Ridges march along the y-axis. Each unique y value defines a group, and density is computed from the x values. The resulting density curves lie in the x-z plane.

Computed variables

The following variables are computed and available via `after_stat()`:

density The kernel density estimate at each point

ndensity Density normalized to a maximum of 1 within each group

count Density multiplied by number of observations (expected count)

n Number of observations in the group

bw Bandwidth actually used for this group

See Also

`geom_ridgeline_3d()` for rendering pre-computed ridgeline data, `stat_density_3d()` for 2D kernel density surfaces, `ggplot2::stat_density()` for the parametrization this stat follows, `ggridges::geom_density_ri` for the 2D ridgeline equivalent.

Examples

```
library(ggplot2)

# Basic usage with iris data
p <- ggplot(iris, aes(y = Sepal.Length, x = Species, fill = Species)) +
  coord_3d() +
  scale_z_continuous(expand = expansion(mult = c(0, NA))) + # remove gap beneath ridges
  theme(legend.position = "none")
p + stat_distributions_3d()

# Normalize max ridge heights
p + stat_distributions_3d(aes(z = after_stat(ndensity)))

# Adjust smoothing bandwidth
p + stat_distributions_3d(adjust = 0.5)

# Use joint bandwidth for consistent smoothing across groups
p + stat_distributions_3d(joint_bandwidth = TRUE)

# Different bandwidth selection rules
p + stat_distributions_3d(bw = "SJ")

# Remove tails with rel_min_height
p + stat_distributions_3d(rel_min_height = 0.05)

# Trim to data range
p + stat_distributions_3d(trim = TRUE)

# Rotated to reduce perspective distortion
p + stat_distributions_3d(alpha = .7) +
   coord_3d(pitch = 0, roll = -90, yaw = 90, dist = 5,
      panels = c("zmin", "xmin"))
```

---

stat_function_3d *3D surface from a function*

Description

Evaluates a function f(x,y) = z over a regular grid and renders the result as a 3D surface or ridgeline plot.

Usage

```
stat_function_3d(
  mapping = NULL,
  data = ensure_nonempty_data,
  geom = "surface_3d",
  position = "identity",
```

```
  ...,
  fun = NULL,
  xlim = NULL,
  ylim = NULL,
  grid = "rectangle",
  n = 40,
  direction = "x",
  trim = TRUE,
  cull_backfaces = FALSE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_function_3d(
  mapping = NULL,
  data = ensure_nonempty_data,
  stat = "function_3d",
  position = "identity",
  ...,
  fun = NULL,
  xlim = NULL,
  ylim = NULL,
  grid = "rectangle",
  n = 40,
  direction = "x",
  trim = TRUE,
  cull_backfaces = FALSE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). By default, fill is mapped to after_stat(z).
data	Ignored; this stat generates its own data.
geom	Geom to use for rendering. Defaults to "surface_3d" for mesh surfaces. Use "ridgeline_3d" for ridgeline rendering.
position	Position adjustment, defaults to "identity".
...	Other arguments passed to the layer.
fun	Function to evaluate. Must accept (x, y) and return numeric z values.
xlim, ylim	Length-2 numeric vectors giving the x and y ranges over which to plot the function.

`grid`, `n`, `direction`, `trim`
: Parameters determining the geometry, resolution, and orientation of the surface grid. See grid_generation for details.

`cull_backfaces`
: Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).

`light`
: A lighting specification object created by `light()`,`"none"` to disable lighting, or `NULL` to inherit plot-level lighting specs from the coord. Specify plot-level lighting in `coord_3d()` and layer-specific lighting in `geom_*3d()` functions.

`na.rm`
: If `FALSE`, missing values are removed.

`show.legend`
: Logical indicating whether this layer should be included in legends.

`inherit.aes`
: If `FALSE`, overrides the default aesthetics.

`stat`
: Statistical transformation to use on the data. Defaults to `"surface_3d"`.

Value

A `Layer` object that can be added to a ggplot.

Computed variables

x, y, z Grid coordinates and function values

dzdx, dzdy Partial derivatives at each point

slope Gradient magnitude: sqrt(dzdx^2 + dzdy^2)

aspect Direction of steepest slope: atan2(dzdy, dzdx)

See Also

`geom_surface_3d()`, `geom_ridgeline_3d()`, `coord_3d()`

Examples

```
# Basic function surface
ggplot() +
  geom_function_3d(fun = function(x, y) sin(x) * cos(y),
                   xlim = c(-pi, pi), ylim = c(-pi, pi)) +
  coord_3d()



# Fill by slope
ggplot() +
  geom_function_3d(fun = function(x, y) x^2 + y^2,
                   xlim = c(-2, 2), ylim = c(-2, 2),
                   aes(fill = after_stat(slope)),
```

```
                  grid = "equilateral") +
  scale_fill_viridis_c() +
  coord_3d()

# As ridgelines
ggplot() +
  stat_function_3d(fun = function(x, y) dnorm(x) * dnorm(y) * 10,
                  xlim = c(-1.5, 1.5), ylim = c(-1.5, 1.5), n = c(15, 30),
                  geom = "ridgeline_3d", base = 0, light = "none",
                  fill = "black", color = "white") +
  coord_3d()
```

`stat_identity_3d` *3D-aware identity transformation*

Description

This stat performs identity transformation (passes data through unchanged) while properly handling discrete scales and lighting specifications for 3D coordinate systems. It also converts group values to hierarchical format to prevent reordering withing groups during depth sorting.

Usage

```
stat_identity_3d(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  light = NULL,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`.
`data`	The data to be displayed in this layer.
`geom`	The geometric object to use display the data.
`position`	Position adjustment, defaults to "identity".
`na.rm`	If `FALSE`, missing values are removed with a warning.
`show.legend`	Logical indicating whether this layer should be included in legends.

inherit.aes	If FALSE, overrides the default aesthetics.
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
...	Other arguments passed on to geom layer.

Details

This stat is primarily intended for use with 3D geoms that need discrete scale or lighting support, following the same pattern as other ggcube stats.

Value

A Layer object that can be added to a ggplot.

Computed variables

- x_raw, y_raw, z_raw: Original values before discrete-to-numeric conversion
- group: Converted to hierarchical format (e.g., "1__group", "2__group") for proper depth sorting.

See Also

geom_point_3d(), geom_polygon_3d() which use this stat for discrete scale support.

---

stat_surface_3d *3D surface from point data*

Description

Takes user-provided (x, y, z) point data and prepares it for surface rendering. If data form a regular grid, can render either a GeomSurface3D of rectangular or right-triangular tiles, or a GeomRidgeline3D or GeomContour3D set of surface slices; otherwise, renders irregular triangular tiles via Delaunay trianuglation.

Usage

```
stat_surface_3d(
  mapping = NULL,
  data = NULL,
  geom = "surface_3d",
  position = "identity",
  ...,
  cull_backfaces = FALSE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
```

```
  inherit.aes = TRUE
)

geom_surface_3d(
  mapping = NULL,
  data = NULL,
  stat = "surface_3d",
  position = "identity",
  ...,
  method = "auto",
  grid = NULL,
  cull_backfaces = FALSE,
  sort_method = "auto",
  scale_depth = TRUE,
  force_convex = TRUE,
  light = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). Must include x, y, and z. By default, fill is mapped to after_stat(z).
data	Data frame containing point coordinates.
geom	Geom to use for rendering. Defaults to "surface_3d" for mesh surfaces. Use "ridgeline_3d" for ridgeline rendering.
position	Position adjustment, defaults to "identity".
...	Other arguments passed to the layer.
cull_backfaces	Logical indicating whether to remove back-facing polygons from rendering. This is primarily for performance optimization but may be useful for aesthetic reasons in some situations. Backfaces are determined using screen-space winding order after 3D transformation. Defaults vary by geometry type: FALSE for open surface-type geometries, TRUE for solid objects (hulls, voxels, etc. where backfaces are generally hidden unless frontfaces are transparent or explicitly disabled).
light	A lighting specification object created by light(),"none" to disable lighting, or NULL to inherit plot-level lighting specs from the coord. Specify plot-level lighting in coord_3d() and layer-specific lighting in geom_*3d() functions.
na.rm	If FALSE, missing values are removed.
show.legend	Logical indicating whether this layer should be included in legends.
inherit.aes	If FALSE, overrides the default aesthetics.
stat	Statistical transformation to use on the data. Defaults to "surface_3d".
method	Tessellation method passed to geom_surface_3d(): "auto" (default), "grid", or "delaunay".

grid	Tile geometry for regular grids: "rectangle" (default), "right1", or "right2".
sort_method	Depth sorting algorithm. See sorting_methods for details.
scale_depth	Logical indicating whether polygon linewidths should be scaled to make closer lines wider and farther lines narrower. Default is TRUE. Scaling is based on the mean depth of a polygon.
force_convex	Logical indicating whether to remove polygon vertices that are not part of the convex hull. Default value varies by geom. Specifying TRUE can help reduce artifacts in surfaces that have polygon tiles that wrap over a visible horizon. For prism-type geoms like columns and voxels, FALSE is safe because polygons fill always be convex.

Details

For regular grids, computes point-level gradients. Works with both `geom_surface_3d()` for mesh rendering and `geom_ridgeline_3d()` for ridgeline rendering.

Value

A `Layer` object that can be added to a ggplot.

Computed variables

For regular grid data:

dzdx, dzdy Partial derivatives at each point

slope Gradient magnitude: sqrt(dzdx^2 + dzdy^2)

aspect Direction of steepest slope: atan2(dzdy, dzdx)

For irregular data, gradient variables are NA.

Grid detection

The stat automatically detects whether data forms a regular grid by checking if `nrow(data) == length(unique(x)) * length(unique(y))`. Regular grids get point-level gradient computation; irregular point clouds are passed through for Delaunay tessellation by the geom.

See Also

`geom_surface_3d()`, `geom_ridgeline_3d()`, `stat_function_3d()`, `coord_3d()`

Examples

```
# Regular grid data ------------------------------------------

# simulated data and base plot for basic surface
d <- dplyr::mutate(tidyr::expand_grid(x = -10:10, y = -10:10),
                   z = sqrt(x^2 + y^2) / 1.5,
                   z = cos(z) - z)
p <- ggplot(d, aes(x, y, z)) +
      coord_3d(light = light(mode = "hsl", direction = c(1, 0, 0)))
```

```
# surface with 3d lighting
p + geom_surface_3d(fill = "steelblue", color = "steelblue", linewidth = .2)



# mesh wireframe (`fill = NA`) with aes line color
p + geom_surface_3d(aes(color = z), fill = NA,
                    linewidth = .5, light = "none") +
      scale_color_gradientn(colors = c("black", "blue", "red"))

# triangulated surface (can prevent lighting flaws)
p + geom_surface_3d(fill = "#9e2602", color = "black", grid = "right2")

# use after_stat to access computed surface-orientation variables
p + geom_surface_3d(aes(fill = after_stat(slope)), grid = "right2") +
      scale_fill_viridis_c() +
      guides(fill = guide_colorbar_3d())

# use `group` to plot data for multiple surfaces
d <- expand.grid(x = -5:5, y = -5:5)
d$z <- d$x^2 - d$y^2
d$g <- "a"
d2 <- d
d2$z <- d$z + 15
d2$g <- "b"
ggplot(rbind(d, d2), aes(x, y, z, group = g, fill = g)) +
      coord_3d(light = "none") +
      geom_surface_3d(color = "black", alpha = .5, light = NULL)

# terrain surface with topographic hillshade and elevational fill
ggplot(mountain, aes(x, y, z, fill = z, color = z)) +
      geom_surface_3d(light = light(direction = c(1, 0, .5),
                                    mode = "hsv", contrast = 1.5),
                      linewidth = .2) +
      coord_3d(ratio = c(1, 1.5, .75)) +
      theme_light() +
      scale_fill_gradientn(colors = c("darkgreen", "rosybrown4", "gray60")) +
      scale_color_gradientn(colors = c("darkgreen", "rosybrown4", "gray60")) +
      guides(fill = guide_colorbar_3d())

# stack of flat surfaces (e.g. a time series of raster maps)
d <- expand.grid(lon = 1:20, lat = 1:20, time = 1:6)
d$value <- sin(-d$lon * d$lat / 10) + d$time / 2
ggplot(d, aes(lon, lat, time, group = time,
              fill = value, color = value)) +
   geom_surface_3d(light = "none") +
   coord_3d() +
   scale_color_viridis_c() + scale_fill_viridis_c()

# stat_surface_3d with alternative geoms -----------------------------

# horizontal slices with geom_ridgeline_3d
```

```
ggplot(mountain, aes(x, y, z)) +
  stat_surface_3d(geom = "ridgeline_3d",
                  fill = "black", color = "white",
                  light = "none", linewidth = .1) +
      coord_3d(ratio = c(1, 1.5, .75), yaw = 45)

# elevation contours with geom_contour_3d
ggplot(mountain, aes(x, y, z, fill = z)) +
  stat_surface_3d(geom = "contour_3d", light = "none",
                  bins = 25, sort_method = "pairwise",
                  color = "black") +
      coord_3d(ratio = c(1, 1.5, .75), yaw = 45) +
      scale_fill_viridis_c(option = "B")


# Irregular point data --------------------------------------

set.seed(42)
pts <- data.frame(x = runif(200, -2, 2), y = runif(200, -2, 2))
pts$z <- with(pts, sin(x) * cos(y))

ggplot(pts, aes(x, y, z = z, fill = z)) +
  stat_surface_3d(sort_method = "pairwise") +
  scale_fill_viridis_c() +
  coord_3d(light = "none")
```

`text_outlines` *Convert text to polygon outlines*

Description

Converts a text string into polygon vertices suitable for plotting with ggplot2's geom_polygon. Uses systemfonts for font matching and glyph extraction.

Usage

```
text_outlines(
  text,
  family = "sans",
  weight = "normal",
  italic = FALSE,
  size = 12,
  tolerance = 0.01,
  spacing = 0,
  hjust = 0.5,
  vjust = 0.5,
```

```
  width = NA,
  align = "left",
  lineheight = 1,
  internal_size = 144
)
```

Arguments

text	Character string to convert
family	Font family name (e.g., "Arial", "Helvetica", "sans")
weight	Font weight ("normal", "bold", "thin", "light", etc.)
italic	Logical; use italic variant
size	Font size in points
tolerance	Bezier curve tolerance; lower values give more detailed outlines
spacing	Additional letter spacing in em units
hjust, vjust	The justification of the textbox surrounding the text
width	Maximum line width in inches for word wrapping; NA disables wrapping
align	Text alignment: "left", "center", or "right"
lineheight	Line height multiplier
internal_size	Internal rendering size for kerning precision; generally shouldn't need adjustment

Value

A data frame with columns: x, y, contour, glyph, letter, poly_id. X and y are vertex coordinates, in point units.

Examples

```
data <- text_outlines("Howdy, partner", family = "Arial")

ggplot2::ggplot(data, ggplot2::aes(x, y, group = letter, subgroup = poly_id)) +
  ggplot2::geom_polygon() +
  ggplot2::coord_fixed()
```

---

zlim *Set z-axis limits*

Description

This is a shorthand for `scale_z_continuous(limits = c(min, max))`. It's a convenient way to set the z-axis limits without specifying other scale parameters.

Usage

```
zlim(min, max, ...)
```

Arguments

min, max	The minimum and maximum values for the z-axis.
...	Additional arguments passed to scale_z_continuous().

Value

A ggplot2 scale object for the z aesthetic with specified limits.

See Also

scale_z_continuous for more control over z-axis scaling, xlim, ylim for x and y axis limits

Other 3D scale functions: scale_z_continuous(), scale_z_discrete()

Examples

```
library(ggplot2)

# Set z-axis limits
ggplot(mtcars, aes(mpg, wt, z = qsec)) +
  geom_point() +
  zlim(15, 20) +
  coord_3d()

# Equivalent to:
ggplot(mtcars, aes(mpg, wt, z = qsec)) +
  geom_point() +
  scale_z_continuous(limits = c(15, 20)) +
  coord_3d()
```

Index