Package: Applr (via r-universe)

October 22, 2026

Title Visualize and Explain Linear Models

Version 1.0.0

Description Tools for visualizing and explaining linear models fit with lm(). Draws 2D slices of higher-dimensional models in base graphics and as a 'ggplot2' layer, offers an interactive 'plotly' 3D scatter with a fitted surface for two-predictor models, and renders model equations as plain text or 'LaTeX'.

License MIT + file LICENSE

URL https://saundersg.github.io/Applr/, https://github.com/saundersg/Applr

BugReports https://github.com/saundersg/Applr/issues

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.5.0), plotly, rlang, graphics, grDevices, grid, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Cameron McClellan [aut], James Beeson [aut, cre], Garrett Saunders [ths]

Maintainer James Beeson <jamesbeeson01@gmail.com>

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

Date/Publication 2026-10-22 15:23:32 UTC

RemoteUrl https://github.com/cran/Applr

RemoteRef HEAD

RemoteSha 7dd97f2314de9bcc2de1042a68a7321618f404ac

Contents

Applr-package . . . 2
add_slice_2d . . . 3
autoplot.lm . . . 5
diagnose . . . 7
geom_slice . . . 7
geom_slice_caption . . . 10
geom_slice_subtitle . . . 12
geom_slice_text . . . 13
GeomSlice . . . 15
GeomSliceBand . . . 16
GeomSliceText . . . 16
lm_equation . . . 17
lm_latex . . . 18
scatter_3d . . . 19
slice_2d . . . 20
StatSlice . . . 21
StatSliceText . . . 22

Index **23**

`Applr-package` *Applr: Tools for Visualizing and Explaining Linear Models*

Description

Applr is a small R package that makes it easy to visualize slices of higher-dimensional linear models and to present model equations clearly. It includes base R and ggplot2 helpers, plus an interactive 3D view for models with two predictors.

Details

Core capabilities:

- 2D slices of multi-variable models in base graphics
- ggplot2 layer for plotting model slices across facets
- Interactive 3D scatter with fitted surface (two predictors)
- Quick text/LaTeX equations from lm() objects

Main functions

- autoplot: `autoplot.lm`
- Base graphics: `slice_2d`, `add_slice_2d`
- ggplot2 layer: `geom_slice` (built on `StatSlice` and `GeomSlice`)
- ggplot2 labeling: `geom_slice_text`, `geom_slice_subtitle`, `geom_slice_caption`
- 3D visualization: `scatter_3d`
- Equations: `lm_equation`, `lm_latex`

Quick start

Fit a model, then see it in one call:

```
library(Applr)
model <- lm(mpg ~ qsec + am, data = mtcars)
autoplot(model)
```

Or build the plot yourself with a geom_slice() layer:

```
library(ggplot2)
model <- lm(hwy ~ displ * drv + cyl, data = mpg)
ggplot(mpg, aes(displ, hwy, color = drv)) +
  geom_point() +
  geom_slice(model, band = TRUE) +
  geom_slice_text() +
  geom_slice_subtitle()
```

Author(s)

Maintainer: James Beeson <jamesbeeson01@gmail.com>

Authors:

- Cameron McClellan <cameron.m.mcclellan@gmail.com>

Other contributors:

- Garrett Saunders <saundersg@byui.edu> [thesis advisor]

See Also

- Function help pages: ?slice_2d, ?geom_slice, ?scatter_3d, ?lm_equation, ?lm_latex
- The package README for a brief tour

---

add_slice_2d *2D Slice to R Plot*

Description

Take a slice from a linear model and add the displayed graph to a preexisting R plot. This function is similar to slice_2d() but adds a line to an existing plot rather than creating a new one. Useful for overlaying multiple model slices or adding model predictions to scatter plots.

Usage

```
add_slice_2d(
  model,
  x_axis = NA,
  n = 100,
  caption = FALSE,
  back_transform = TRUE,
  interval = "none",
  ...,
  xaxis = NULL
)
```

Arguments

model	A linear model
x_axis	A non-required specification of the x-axis variable; default is the first x var in the lm. May also be an expression of the model's predictors, such as `x_axis = "x * x_pos"`, in which case predictions are made at the model's own data points and placed at the expression's value.
n	Number of displayed points; default is 100
caption	Boolean to include caption with values of predictor variables not shown in the chart, FALSE by default
back_transform	How to map predictions onto the y-axis when the model's response is transformed. Default `TRUE` auto-detects from the model formula; `FALSE` turns back-transformation off; a one-argument function (e.g. `exp`) or a name (`"log"`, `"log10"`, `"log2"`, `"sqrt"`, `"exp"`, `"inverse"`) applies that transformation.
interval	Draw a shaded band around the line: `"none"` (default), or `"confidence"` / `"prediction"` for the corresponding `predict.lm()` interval.
...	Remaining vars (color, linetype, etc.)
xaxis	Deprecated; use `x_axis` instead.

Value

A line added to a preexisting R plot displaying a 2D graphable slice of an HD model

Examples

```
model <- lm(log(mpg) ~ disp + hp, data = mtcars)
plot(mpg ~ disp, data = mtcars)
add_slice_2d(model)

# Multiple slices on one plot
plot(mpg ~ disp, data = mtcars)
add_slice_2d(model, hp = min(mtcars$hp), col = "blue")
add_slice_2d(model, hp = max(mtcars$hp), col = "red", lty = 2)
```

autoplot.lm *Automatically plot a linear model*

Description

autoplot() turns a fitted lm() into a complete ggplot in one call: the data the model was fitted to as a scatter, with a geom_slice() prediction line — and, by default, its confidence ribbon — through it. The result is a regular ggplot, so it can be extended with + as usual (labels, scales, more layers).

Usage

```
## S3 method for class 'lm'
autoplot(
  object,
  mapping = NULL,
  type = c("2d", "3d"),
  summary = TRUE,
  interval = "confidence",
  ...
)
```

Arguments

object	A linear model fitted by lm() **with a** data **argument** — the data is recovered from the model, so lm(y ~ x, data = your_data) works but lm(your_data$y ~ your_data$x) does not.
mapping	Aesthetics created with ggplot2::aes(), passed to ggplot2::ggplot() — for example mapping = aes(color = factor(cyl)). They resolve against the model's full data frame, so a column the model never mentions (such as cyl above) still works. An entry named x chooses the x-axis (mapping = aes(x = hp)), overriding the default first numeric predictor; an entry named y overrides the response axis.
type	"2d" (default) for a geom_slice() ggplot, or "3d" for an interactive scatter_3d() surface (needs exactly two numeric predictors).
summary	Whether to print summary(object) to the console before returning the plot (default TRUE). Printing it up front means you can refit the model and inspect its coefficients from inside the same autoplot() call; pass summary = FALSE to suppress it.
interval	The ribbon drawn around the slice line, passed to geom_slice(): "confidence" (default) for the mean's confidence interval, "prediction" for a single new observation's interval, or "none" for a bare line. Ignored when type = "3d".
...	Passed on to geom_slice() — for example predict_vars = list(hp = 110) to choose the slice, or fixed aesthetics such as color = "red" — or to scatter_3d() when type = "3d".

Details

Pass `type = "3d"` for a model with exactly two numeric predictors to get an interactive `scatter_3d()` surface instead of the 2-D slice; `...` is then forwarded to `scatter_3d()` (e.g. `n`, `colors`).

The model's *first numeric* predictor goes on the x-axis (override with `mapping = aes(x = ...)`) and the raw response variable goes on the y-axis. A predictor that groups into a few discrete lines rather than varying continuously — non-numeric, or numeric with at most five distinct values (like `cyl`'s 4/6/8) — is worth showing, not holding at one value: the first such predictor not already on the plot becomes `aes(color = ...)` (one slice line per level) and a second one facets the plot. Everything past that is handled by `geom_slice()` and reported on the console: predictors not visible on the plot are imputed (mean for numeric, most common value for factor/character), and a transformed response such as `lm(log(y) ~ x)` is automatically back-transformed to match the raw y-axis.

Value

A ggplot of the model's data with a slice of the model drawn through it, or — when `type = "3d"` — a plotly surface from `scatter_3d()`.

See Also

`geom_slice()`, which draws the line and handles everything slice-shaped; `scatter_3d()` for the interactive surface; and `geom_slice_subtitle()` / `geom_slice_text()` to annotate the result.

Examples

```
library(ggplot2)

# A complete plot from a model in one call; summary() prints to the console
autoplot(lm(Petal.Length ~ Sepal.Length + I(Sepal.Length^2):Species, iris))

# Predictors that group into a few values become color, then facets
model <- lm(hwy ~ displ + drv + cyl, data = mpg)
autoplot(model, summary = FALSE)

# Change the defaults like a normal ggplot; the rest is still inferred
autoplot(model, aes(color = factor(cyl)), summary = FALSE) + facet_wrap(~drv)

# Choose the x-axis and pass geom_slice() options through
autoplot(model,
         aes(x = cyl, color = factor(drv)),
         interval = "none", summary = FALSE) +
  labs(title = "The Same Model From a Different Angle")

# Transformed response, back-transformed onto the raw mpg axis
autoplot(lm(log(mpg) ~ disp + hp, data = mtcars), summary = FALSE)

# An interactive 3-D surface for a model with two numeric predictors
if (interactive()) {
  autoplot(lm(mpg ~ wt + I(wt^2) + hp, data = mtcars), type = "3d")
}
```

diagnose — *Diagnostic plots for linear models*

Description

Generates a set of three diagnostic plots for a linear model object, side by side: Residuals vs Fitted (checks linearity and constant variance), Normal Q-Q (checks normality of residuals), and residuals in data order (checks independence). Use it right after fitting a model to check the regression assumptions in one call instead of building each plot yourself. Graphical parameters (`mfrow`) are restored on exit.

Usage

```
diagnose(model)
```

Arguments

`model`	An object of class `lm`.

Value

None. The function is called for its side effect of producing plots.

Examples

```
# Simple regression
diagnose(lm(mpg ~ wt, data = mtcars))

# Works the same for multiple regression
diagnose(lm(mpg ~ wt + hp + disp, data = mtcars))

# A poorly-specified model shows curvature in Residuals vs Fitted
diagnose(lm(dist ~ speed, data = cars))
```

geom_slice — *Display a 2D slice of a linear model*

Description

`geom_slice()` draws the prediction line of a fitted `lm()` on a ggplot — a 2D *slice* of a possibly high-dimensional model. It looks like `ggplot2::geom_smooth()`, but where `geom_smooth()` fits its own model to the plotted data, `geom_slice()` draws *your* model. The predictor mapped to the plot's x-axis varies along the line; every other predictor is fixed, and `geom_slice()` reports how, so it is always clear which slice of the model you are looking at:

Usage

```
geom_slice(
  model,
  n = 100,
  inherit.aes = TRUE,
  predict_vars = list(),
  back_transform = NULL,
  interval = "none",
  band = FALSE,
  full_range = FALSE,
  ...,
  fullrange = NULL
)
```

Arguments

`model` A linear model fitted by `lm()`.

`n` Number of prediction points along the line (default 100).

`inherit.aes` If `TRUE` (default), inherit aesthetics from the `ggplot()` call.

`predict_vars` A named list of values at which to hold predictors not shown on the plot, such as `predict_vars = list(hp = 110)`. Unlisted predictors are imputed (with a message). Giving a variable several values draws one line per value, and several multi-value variables are crossed: `predict_vars = list(x2 = c(1, 2, 3), x3 = c(1, 4))` draws 6 lines.

`back_transform` How to map predictions onto the y-axis when the model's response is transformed. Default `NULL` (and `TRUE`) auto-detects from the model formula; `FALSE` turns back-transformation off; a one-argument function (e.g. `exp`) or a name (`"log"`, `"log10"`, `"log2"`, `"sqrt"`, `"exp"`, `"inverse"`) applies that transformation.

`interval` Draw a ribbon around the line: `"none"` (default), or `"confidence"` / `"prediction"` for the corresponding `predict.lm()` interval. The ribbon follows its line's color unless you set `fill`. Cannot be combined with `band`.

`band` Draw a *projection band* — two edge slices with a translucent ribbon between them — instead of a single line. `band = "variable"` spans that predictor: between the values you gave in `predict_vars` (e.g. `predict_vars = list(x2 = c(1, 4))`), or its observed data range when `predict_vars` leaves it out. `band = TRUE` infers the variable: the one multi-value `predict_vars` entry, or the single predictor the plot does not otherwise show. Default `FALSE`.

`full_range, fullrange` If `TRUE`, each line spans the full x range of the panel instead of stopping at its group's own data range (like `fullrange` in `ggplot2::geom_smooth()`; `fullrange` is accepted as an alias). Default `FALSE`. Only applies when the x-axis maps a single predictor; with a composite x-axis expression predictions are made at the data points, so the lines keep their data extent.

`...` Other arguments passed to the layer, such as fixed aesthetics (`color = "red"`, `linewidth = 1.2`).

Details

- Variables named in `predict_vars` are held at your chosen values.
- Variables mapped to a grouping aesthetic (e.g. `aes(color = g)`) are pinned to each group's own value — one line per group. Any `interval` or band ribbon is drawn in its own line's color.
- Facet variables are pinned to each panel's value.
- Anything left over is *imputed* (mean for numeric, most common value for factor/character), with a console message naming the value used.

If the model's response is transformed (e.g. `lm(log(y) ~ x)`) but the plot shows raw `y`, predictions are automatically back-transformed to match the y-axis (a message says so). Axis expressions (`aes(log(y))`, `aes(x * x2)`) and transformed scales (`scale_x_log10()`) are also handled.

Value

A ggplot2 layer that draws the slice.

See Also

- `geom_slice_text()` to label each line with the values that produced it.
- `geom_slice_subtitle()` / `geom_slice_caption()` to describe the slice (model equation, held values) in the plot's subtitle or caption.
- `autoplot.lm()` for a complete data-plus-slice plot in one call.
- `ggplot2::geom_smooth()`, which this layer resembles, except that it draws a model you fitted rather than fitting its own.

Examples

```
library(ggplot2)

# Basic use, like geom_smooth() but drawing *your* model. Sepal.Width is
# not on the plot, so it is held at its mean (a console message says so).
model <- lm(Sepal.Length ~ Sepal.Width + Petal.Length, data = iris)
ggplot(iris, aes(Petal.Length, Sepal.Length)) +
  geom_point() +
  geom_slice(model)

# Several values draw one line per value (label them with
# geom_slice_text()); several multi-value variables are crossed
ggplot(iris, aes(Petal.Length, Sepal.Length)) +
  geom_point() +
  geom_slice(model, predict_vars = list(Sepal.Width = c(2, 3, 4))) +
  geom_slice_text()

# Confidence or prediction ribbon around the line
model2 <- lm(hwy ~ displ + I(displ^2), data = mpg)
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_slice(model2, interval = "prediction")
```

```
# A grouping aesthetic pins its predictor to each group's own value:
# one line per drive type, each stopping at the end of its group's data
model3 <- lm(hwy ~ displ + I(displ^2) + displ:drv, data = mpg)
ggplot(mpg, aes(displ, hwy, color = drv)) +
  geom_point() +
  geom_slice(model3)

# full_range = TRUE extends each group's line across the whole panel
ggplot(mpg, aes(displ, hwy, color = drv)) +
  geom_point() +
  geom_slice(model3, full_range = TRUE)

# Facet variables pin per panel the same way
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_slice(model3) +
  facet_wrap(~drv)

# A projection band shows the reach of a predictor, here cyl from 5 to 7;
# band = TRUE infers the variable and its data range when it can
model4 <- lm(hwy ~ displ * drv + cyl, data = mpg)
ggplot(mpg, aes(displ, hwy, color = drv)) +
  geom_point() +
  geom_slice(model4, band = "cyl", predict_vars = list(cyl = c(5, 7)))

# A transformed response is back-transformed to match the y-axis: the
# model predicts log(mpg), the line appears in raw mpg units
model5 <- lm(log(mpg) ~ disp + hp, data = mtcars)
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  geom_slice(model5)

# ... adapts when the axis itself is transformed
ggplot(mtcars, aes(disp, log(mpg))) +
  geom_point() +
  geom_slice(model5)

# ... or control the mapping yourself with back_transform
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  geom_slice(model5, back_transform = exp)

# n controls how many prediction points make up the line
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  geom_slice(model5, n = 10)
```

`geom_slice_caption` *Describe the lines drawn by geom_slice() in the plot caption*

Description

geom_slice_caption() is geom_slice_subtitle() for the plot caption: it fills the caption (right-aligned in ggplot2's default themes) with the model equation and the held values the plot does not otherwise show. All behavior — what is reported, what is skipped because the legend, facets, or geom_slice_text() already label it, and every parameter — matches geom_slice_subtitle(); see its documentation for the details.

Usage

```
geom_slice_caption(model = TRUE, prepend = "", append = "", ..., wrap = TRUE)
```

Arguments

model	If TRUE (default), the subtitle's first line is the model equation; FALSE drops it, leaving only the held-values line.
prepend, append	Plain strings pasted before the first line and after the last line of the default subtitle.
...	Passed to lm_equation() when the equation is written, under its own argument names — chiefly style, which controls how factor terms are labelled. Left unset, style is chosen to fit: with wrap on, the equation is tried spelled out (4.09*(Species="setosa")) and briefly (4.09*[setosa]), each with and without the hanging indent, preferring the spelled-out form and giving up the indent last. The first that fits wins, and a message says so if the brief form did. Because the indent goes last, a narrower plot can bring the full names *back*. The brief form is never offered when two terms would shorten to the same label — a bare [High] from two different factors no longer says which it meant. Naming style yourself pins it and switches all of that off. Pin it (or give wrap a width) when the annotation must come out the same whatever size device it is drawn on.
wrap	Controls wrapping of lines too long to fit. TRUE (default) measures how much room the subtitle has in this plot on the current graphics device and breaks long lines to fit, between terms only, with continuation lines hanging under the right-hand side of the equal sign. FALSE leaves the text as one line per item, however far it runs off the plot. A number is a width in inches to wrap to instead of measuring — useful when the plot will be drawn at a size other than the current device, as with ggsave(width =).

Value

An object that sets the plot caption when added to a ggplot.

See Also

geom_slice_subtitle() for the full documentation of what is reported; geom_slice() for the layers being described; geom_slice_text() to label the lines themselves.

Examples

```
library(ggplot2)

# Caption reports the model equation and the held Sepal.Width; Species
# is already shown by the legend, so it is left out
model <- lm(Petal.Length ~ Sepal.Length + I(Sepal.Length^2):Species +
              Sepal.Width, data = iris)
ggplot(iris, aes(Sepal.Length, Petal.Length, color = Species)) +
  geom_point() +
  geom_slice(model, interval = "confidence") +
  geom_slice_caption()
```

`geom_slice_subtitle` *Describe the lines drawn by geom_slice() in the plot subtitle*

Description

`geom_slice_subtitle()` fills the plot subtitle with a description of the plot's `geom_slice()` layers: the model equation (`lm_equation()` style) on the first line, and the held values the plot does not otherwise show on the second (`"held at: x2 = 2.507"`, formatted like `geom_slice()`'s console messages).

Usage

```
geom_slice_subtitle(model = TRUE, prepend = "", append = "", ..., wrap = TRUE)
```

Arguments

`model`	If `TRUE` (default), the subtitle's first line is the model equation; `FALSE` drops it, leaving only the held-values line.
`prepend, append`	Plain strings pasted before the first line and after the last line of the default subtitle.
`...`	Passed to `lm_equation()` when the equation is written, under its own argument names — chiefly `style`, which controls how factor terms are labelled. Left unset, `style` is chosen to fit: with `wrap` on, the equation is tried spelled out (`4.09*(Species="setosa")`) and briefly (`4.09*[setosa]`), each with and without the hanging indent, preferring the spelled-out form and giving up the indent last. The first that fits wins, and a message says so if the brief form did. Because the indent goes last, a narrower plot can bring the full names *back*. The brief form is never offered when two terms would shorten to the same label — a bare `[High]` from two different factors no longer says which it meant. Naming `style` yourself pins it and switches all of that off. Pin it (or give `wrap` a width) when the annotation must come out the same whatever size device it is drawn on.

`wrap` Controls wrapping of lines too long to fit. `TRUE` (default) measures how much room the subtitle has in this plot on the current graphics device and breaks long lines to fit, between terms only, with continuation lines hanging under the right-hand side of the equal sign. `FALSE` leaves the text as one line per item, however far it runs off the plot. A number is a width in inches to wrap to instead of measuring — useful when the plot will be drawn at a size other than the current device, as with `ggsave(width = )`.

Details

It takes no `model` or `predict_vars` — everything is borrowed from the plot's existing `geom_slice()` layers, so add it *after* them (and after any `geom_slice_text()`). The held-values line is conscious of what the plot already labels: variables labeled by `geom_slice_text()`, pinned by a grouping aesthetic (the legend covers those), or pinned by faceting are left out. User-chosen `predict_vars` values and imputed defaults are treated the same — both are unlabeled held values. When nothing is held, the line is dropped entirely.

Value

An object that sets the plot subtitle when added to a ggplot.

See Also

`geom_slice()` for the layers being described; `geom_slice_caption()` for the same text in the caption; `geom_slice_text()` to label the lines themselves.

Examples

```
library(ggplot2)

# Subtitle reports the model equation and the held Sepal.Width; Species
# is already shown by the legend, so it is left out
model <- lm(Petal.Length ~ Sepal.Length + I(Sepal.Length^2):Species +
              Sepal.Width, data = iris)
ggplot(iris, aes(Sepal.Length, Petal.Length, color = Species)) +
  geom_point() +
  geom_slice(model, interval = "confidence") +
  geom_slice_subtitle()
```

`geom_slice_text` *Label the lines drawn by geom_slice()*

Description

`geom_slice_text()` writes a text label at the end of each line drawn by the plot's `geom_slice()` layers. Multi-value `predict_vars` draw visually identical lines with nothing distinguishing them; labels fix that, and double as a legend replacement for grouping aesthetics.

Usage

```
geom_slice_text(
  style = "variable",
  location = "right",
  offset = 5,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  expand = TRUE
)
```

Arguments

`style`	How to write the labels: `"variable"` (default) writes `"x2: 0"`; `"value"` writes the bare `"0"`; `"legend"` writes bare values plus a key in the panel's top-right corner naming the variables (`"labels: x2; x3"`).
`location`	Which end of the line to label: `"right"` (default) or `"left"`.
`offset`	Gap between the line end and the label edge, in points (default 5). The gap is constant regardless of label text, axis range, or plot size.
`hjust, vjust`	Manual text justification. Default `NULL` anchors the label at the line end, displaced by `offset`; setting `hjust` yourself disables the automatic offset entirely.
`color`	Label color. Default `NULL` inherits each line's color.
`expand`	If `TRUE` (default), widen the x-range on the label side so the labels fit (plus y-headroom for the `"legend"` key). `FALSE` leaves the scales alone. Only the expansion is touched: a `scale_x_*()` you set yourself keeps its limits, breaks, name and transform, and an expansion you set on the side away from the labels is kept as well. The room reserved is the widest label's drawn width measured against the panel's own width, taken from the graphics device open when the layer is added; a plot re-sized much narrower afterwards may still clip. Labels needing more than half the panel are clipped with a warning rather than crowding out the data.

Details

It takes no `model` or `predict_vars` — everything is borrowed from the plot's existing `geom_slice()` layers, so add it *after* them. Labels describe the `predict_vars` values (`"x2: 0"`, joined with `"; "` when several variables are crossed); when the slice layer has no `predict_vars`, labels describe the grouping aesthetic instead (`"g: A"`). Each label inherits its line's color.

Value

An object that adds the label layers when added to a ggplot.

See Also

`geom_slice()` for the layers being labeled; `geom_slice_subtitle()` / `geom_slice_caption()` to describe the slice in the plot's text instead.

Examples

```
library(ggplot2)

# One label per predict_vars value
model <- lm(Sepal.Length ~ Sepal.Width + Petal.Length, data = iris)
ggplot(iris, aes(Petal.Length, Sepal.Length)) +
  geom_point() +
  geom_slice(model, predict_vars = list(Sepal.Width = c(2, 3, 4))) +
  geom_slice_text()

# A projection band, labeled on each edge
ggplot(iris, aes(Petal.Length, Sepal.Length)) +
  geom_point() +
  geom_slice(model, band = TRUE) +
  geom_slice_text()

# Labels take the color of their group
model2 <- lm(hwy ~ displ * drv + cyl, data = mpg)
ggplot(mpg, aes(displ, hwy, color = drv)) +
  geom_point() +
  geom_slice(model2, predict_vars = list(cyl = c(4, 8))) +
  geom_slice_text()

# Lines defined by several variables read best with a corner key
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_slice(model2, predict_vars = list(cyl = c(4, 8), drv = c("f", "r"))) +
  geom_slice_text(style = "legend")
```

GeomSlice *GeomSlice*

Description

The geom behind `geom_slice()`: ggplot2::GeomSmooth with slice-flavored default aesthetics. Like `geom_smooth()`, it draws a line plus — when the stat supplies `ymin`/`ymax` (i.e. `interval = "confidence"` or `"prediction"`) — a ribbon; `alpha` styles the ribbon, not the line. The ribbon follows its line's colour unless `fill` is set explicitly.

Usage

```
GeomSlice
```

Format

An object of class `ggproto`, inheriting from `GeomSmooth`.

GeomSliceBand *GeomSliceBand*

Description

The geom behind geom_slice() when band is requested: a translucent ribbon between the two edge slices, with each edge drawn as an ordinary slice line. Like GeomSlice, alpha styles the ribbon, not the lines.

Usage

```
GeomSliceBand
```

Format

An object of class ggproto, inheriting from GeomSlice.

GeomSliceText *GeomSliceText*

Description

The geom behind geom_slice_text(): ggplot2::GeomText whose labels are displaced by a fixed number of points at draw time (like axis tick label margins), so the gap between line end and label is constant regardless of label text, axis range, or plot size.

Usage

```
GeomSliceText
```

Format

An object of class ggproto, inheriting from GeomText.

`lm_equation` *Write out the equation of a linear model*

Description

Takes a fitted linear model and returns a human-readable equation string showing the relationship between the response variable and predictors. Coefficients are rounded to 3 significant figures for readability. Terms are named as in the fitted coefficients, so factor predictors show one term per dummy level (e.g. `2.1*gB`) and models without an intercept print no intercept.

Usage

```
lm_equation(model, style = c("prettier", "brackets", "raw"))
```

Arguments

`model`	A linear model
`style`	How factor terms are named. `"prettier"` (the default) spells out the factor name and level (e.g. `4.09*(Species="setosa")`), `"brackets"` shows the level alone (e.g. `4.09*[setosa]`), and `"raw"` keeps the design-matrix names (e.g. `4.09*Speciessetosa`).

Details

Use it when you want to report or sanity-check a fitted model as an equation rather than a coefficient table — for example when writing up homework or checking which dummy terms a factor produced.

Value

A character string of length 1 containing the fitted equation, e.g. `"mpg = 30.7 - 0.0248*disp - 0.0245*hp"`.

Examples

```
# Simple regression
lm_equation(lm(mpg ~ wt, data = mtcars))

# Multiple predictors
lm_equation(lm(mpg ~ disp + hp, data = mtcars))

# Factor predictor: one term per non-reference level, spelled out by default
model <- lm(Sepal.Length ~ Sepal.Width + Species, data = iris)
lm_equation(model)

# `style = "brackets"` shows just the level; `"raw"` keeps design-matrix names
lm_equation(model, style = "brackets")
lm_equation(model, style = "raw")
```

```
# Transformed terms and interactions
lm_equation(lm(mpg ~ wt + I(wt^2), data = mtcars))
lm_equation(lm(Sepal.Length ~ Sepal.Width * Species, data = iris))

# No-intercept models print no intercept
lm_equation(lm(mpg ~ 0 + wt, data = mtcars))
```

`lm_latex` *Write out the equation of a linear model in LaTeX format*

Description

Takes a fitted linear model and generates a properly formatted LaTeX equation suitable for inclusion in R Markdown documents, academic papers, or presentations. The output includes underbrace notation to clearly label predicted values and predictor variables, making it ideal for educational or presentation purposes. Coefficients are rounded to 3 significant figures.

Usage

```
lm_latex(model, style = c("prettier", "brackets", "raw"))
```

Arguments

`model`	A linear model
`style`	How factor terms are labelled. `"prettier"` (the default) spells out the factor name and level (e.g. `(Species="setosa")`), `"brackets"` shows the level alone (e.g. `[setosa]`), and `"raw"` keeps the design-matrix names (e.g. `Speciessetosa`).

Details

Use it when a model needs to appear as typeset math — in an R Markdown chunk with `results = "asis"`, the LaTeX string prints ready to render.

Value

A character string of length 1 containing the display-math LaTeX equation (wrapped in `$$...$$`), printed to the console with `cat()` and returned invisibly.

Examples

```
# Simple regression
lm_latex(lm(mpg ~ wt, data = mtcars))

# Multiple predictors and a transformed term
lm_latex(lm(mpg ~ disp + hp + I(hp^2), data = mtcars))

# Factor predictor with interactions, spelled out by default
```

```
model <- lm(Sepal.Length ~ Sepal.Width * Species, data = iris)
lm_latex(model)

# `style = "brackets"` shows just the level; `"raw"` keeps design-matrix names
lm_latex(model, style = "brackets")
lm_latex(model, style = "raw")

# Capture the string instead of just printing it
eq <- lm_latex(lm(mpg ~ wt, data = mtcars))
nchar(eq)
```

`scatter_3d` *Plot a 3D-graphable linear model*

Description

Creates an interactive 3D visualization of a linear model with exactly two predictor variables. The model's own data is drawn as scattered points and a semi-transparent regression surface is overlaid across the whole predictor space, so you can *see* the fitted surface rather than read its coefficients.

Usage

```
scatter_3d(model, n = 100, colors = c("blue", "yellow"))
```

Arguments

`model`	A fitted `lm()` with exactly two numeric predictor variables.
`n`	Number of grid points along each predictor axis used to evaluate the prediction surface (default 100); larger values give a smoother surface at the cost of speed.
`colors`	Character vector defining the marker color gradient, mapped to the response value so points read from low to high.

Details

Reach for it when a model has exactly two predictors and you want to understand how it behaves across their combinations. Transformed terms are fine — the surface is evaluated with `predict()` on the raw predictors, so `lm(z ~ x + I(x^2) + y)` graphs over `x` and `y` as expected.

Value

A plotly object (an interactive 3D plot). When printed — e.g. at the console or in RStudio — it renders in the Viewer pane.

Examples

```
if (interactive()) {
  # Transformed terms graph over their raw predictors
  model <- lm(mpg ~ wt + I(wt^2) + hp, data = mtcars)
  scatter_3d(model)

  # Custom marker gradient and a finer surface grid
  scatter_3d(model, n = 200, colors = c("purple", "orange"))

  # Interaction models produce a twisted (non-planar) surface
  twisted <- lm(Sepal.Length ~ Sepal.Width * Petal.Length, data = iris)
  scatter_3d(twisted)
}
```

---

slice_2d *2D Slicer*

Description

Take a linear model and display a 2D slice in base R. This function creates a new plot comparing the response y variable with one predictor x variable and displaying a linear model line with all other predictor variables held at constant values. For unspecified variables, the x_axis is assumed to be the first predictor in the model and all other variable are imputed (mean for numeric, first level for factors).

Usage

```
slice_2d(
  model,
  x_axis = NULL,
  n = 100,
  caption = TRUE,
  back_transform = TRUE,
  interval = "none",
  ...,
  xaxis = NULL
)
```

Arguments

`model`	A linear model
`x_axis`	A non-required specification of the x-axis variable; default is the first x var in the lm. May also be an expression of the model's predictors, such as `x_axis = "x * x_pos"`, in which case predictions are made at the model's own data points and placed at the expression's value.
`n`	Number of displayed points; default is 100

caption	Boolean to include caption with values of predictor variables not shown in the chart, TRUE by default
back_transform	How to map predictions onto the y-axis when the model's response is transformed. Default TRUE auto-detects from the model formula; FALSE turns back-transformation off; a one-argument function (e.g. exp) or a name ("log", "log10", "log2", "sqrt", "exp", "inverse") applies that transformation.
interval	Draw a shaded band around the line: "none" (default), or "confidence" / "prediction" for the corresponding predict.lm() interval.
...	Remaining vars (color, linetype, etc.)
xaxis	Deprecated; use x_axis instead.

Value

A 2D graph of a sliced model

Examples

```
model <- lm(log(mpg) ~ disp + hp, data = mtcars)
slice_2d(model)

# Choose the x-axis, hold disp at a value, and style the line
slice_2d(model, x_axis = "hp", disp = 250, n = 150, col = "blue", lwd = 2)

# With a 95% confidence band around the line
slice_2d(model, interval = "confidence")
```

StatSlice *StatSlice*

Description

The stat behind geom_slice(). Once per layer it resolves a "slice plan" from the model and the plot's aesthetic mapping (which predictor varies along x, which are pinned by groups/facets, which are held constant, and how predictions map onto the y-axis); then, for each group, it generates an n-point prediction line from the model.

Usage

```
StatSlice
```

Format

An object of class ggproto, inheriting from Stat.

StatSliceText *StatSliceText*

Description

The stat behind `geom_slice_text()`. It resolves the same slice spec as StatSlice (borrowed from the plot's `geom_slice()` layer), reduces each prediction line to its endpoint, and attaches the label text.

Usage

```
StatSliceText
```

Format

An object of class `ggproto`, inheriting from `Stat`.

Index

∗ **datasets**
GeomSlice, 15
GeomSliceBand, 16
GeomSliceText, 16
StatSlice, 21
StatSliceText, 22

add_slice_2d, *2*, 3
Applr *(*Applr-package*)*, 2
Applr-package, 2
autoplot.lm, *2*, 5
autoplot.lm(), *9*

diagnose, 7

geom_slice, *2*, 7
geom_slice(), *5, 6*, *11–16*, *21*
geom_slice_caption, *2*, 10
geom_slice_caption(), *9*, *13, 14*
geom_slice_subtitle, *2*, 12
geom_slice_subtitle(), *6*, *9*, *11*, *14*
geom_slice_text, *2*, 13
geom_slice_text(), *6*, *9*, *11*, *13*, *16*, *22*
GeomSlice, *2*, 15, *16*
GeomSliceBand, 16
GeomSliceText, 16
ggplot2::aes(), *5*
ggplot2::geom_smooth(), *7–9*
ggplot2::GeomSmooth, *15*
ggplot2::GeomText, *16*
ggplot2::ggplot(), *5*

lm(), *5*, *7, 8*, *19*
lm_equation, *2*, 17
lm_equation(), *11, 12*
lm_latex, *2*, 18

predict(), *19*
predict.lm(), *4*, *8*, *21*

scatter_3d, *2*, 19
scatter_3d(), *5, 6*
slice_2d, *2*, 20
StatSlice, *2*, 21, *22*
StatSliceText, 22