Package: depictr (via r-universe)

November 22, 2026

Type Package

Title A Unified Toolkit for Visualising Statistical Models and Data

Version 0.4.0

Description A toolkit of publication-ready plots that span the whole analysis workflow with one consistent appearance. It covers exploratory data analysis (distributions, categorical summaries, bivariate plots, scatter-plot matrices, correlation heatmaps, missing-data maps, outliers, estimation statistics and descriptive tables), multivariate analysis, clustering with diagnostics, Kaplan-Meier survival curves and time series (trends, autocorrelation, decomposition, seasonality and forecasting). It also covers model estimates and inference (forest plots, model comparison, frequentist and Bayesian estimates, predicted values, interactions, random effects and optimiser checks), diagnostics and classification (residual panels, binned residuals, influence, quantile-quantile, receiver operating characteristic (ROC) curves, calibration, threshold tuning and confusion matrices) and uncertainty and power. Reporting helpers provide a shared theme, colourblind-aware palettes, plot composition and saving. Every plotting function returns a 'ggplot2' object, or a 'patchwork' object for composite panels. Heavier modelling back-ends are optional. The package includes reproducibly simulated datasets on which every example and vignette runs without further setup.

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

LazyData true

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.5.2), stringr, scales, patchwork (>= 1.3.0), rlang, stats, grDevices, utils, Rdpack

Suggests lme4 (>= 1.1-19), lmerTest, MASS, broom, simr, knitr, rmarkdown, survival, ggdist, posterior, boot, cluster, colorspace, ragg, callr, spelling, testthat (>= 3.0.0), withr

VignetteBuilder knitr

RdMacros Rdpack

Config/testthat/edition 3

Config/Needs/website pkgdown

URL https://pablobernabeu.github.io/depictr/, https://github.com/pablobernabeu/depictr

BugReports https://github.com/pablobernabeu/depictr/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Pablo Bernabeu [aut, cre] (ORCID: <https://orcid.org/0000-0003-1083-2460>)

Maintainer Pablo Bernabeu <pcbernabeu@gmail.com>

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

Date/Publication 2026-11-22 08:30:08 UTC

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

RemoteRef HEAD

RemoteSha e517575e71f86b5abf375e8845f72714cf25a649

Contents

acf_plot 4
arrange_plots 5
binned_residual_plot 6
calibration_plot 7
check_figure 8
clinical_trial 13
cluster_plot 14
coefficient_plot 16
compare_models 19
confusion_matrix_plot 21
correlation_heatmap 22
crop_yield 23
decompose_plot 24
dendrogram_plot 26
depictr_options 27
depictr_palette 28
dumbbell_plot 30
ecdf_plot 32
effects_plot 33
estimation_plot 34
explore_bivariate 37
explore_categorical 38

explore_distribution . . . 39
explore_pairs . . . 40
format_terms . . . 41
frequentist_bayesian_plot . . . 42
gain_plot . . . 45
group_comparison_plot . . . 46
influence_plot . . . 48
interaction_plot . . . 48
k_diagnostic . . . 50
lexical_decision . . . 51
lift_plot . . . 52
missingness_map . . . 53
model_fit_table . . . 54
model_report . . . 56
monthly_sales . . . 57
optimizer_fixef_plot . . . 58
outlier_plot . . . 60
palette_preview . . . 61
palette_safety . . . 62
pca_plot . . . 63
posterior_plot . . . 65
power_curve_plot . . . 67
pr_curve_plot . . . 69
qq_plot . . . 71
raincloud_plot . . . 72
random_effects_plot . . . 73
residual_diagnostics_plot . . . 75
ridgeline_plot . . . 76
roc_curve_plot . . . 77
save_plot . . . 79
scale_colour_depictr . . . 81
scatter_trend . . . 83
scree_plot . . . 84
seasonal_plot . . . 85
silhouette_plot . . . 86
simulate_cvd . . . 88
simulate_figure . . . 89
summary_table . . . 90
survival_plot . . . 92
theme_depictr . . . 94
threshold_plot . . . 95
tidy_estimates . . . 97
timeseries_plot . . . 98
ts_forecast . . . 100
vif_plot . . . 102
wellbeing_survey . . . 103

Index **105**

`acf_plot` *Autocorrelation plot*

Description

Plots the autocorrelation or partial autocorrelation function of a series, with the approximate significance bounds, as a lollipop chart.

Usage

```
acf_plot(
  x,
  lag_max = NULL,
  type = c("correlation", "partial"),
  conf_level = 0.95,
  title = NULL,
  x_lab = "Lag",
  y_lab = NULL
)
```

Arguments

`x`	A numeric vector or `ts` object holding one series. A multivariate `ts` or a matrix with several columns is refused.
`lag_max`	Maximum lag, passed to `stats::acf()` or `stats::pacf()`.
`type`	`"correlation"` for the ACF or `"partial"` for the PACF.
`conf_level`	Confidence level for the significance bounds, a single number strictly between 0 and 1.
`title, x_lab, y_lab`	Title and axis labels.

Details

A missing value inside the series is kept in its place, and the correlations are computed from the pairs of observations that are both present (`na.action = na.pass`), with a warning. Dropping it would join the observations on either side of the gap as if they were neighbours.

Value

A ggplot2::ggplot object.

Examples

```
acf_plot(AirPassengers)
acf_plot(AirPassengers, type = "partial")
```

`arrange_plots` *Compose several plots into one figure*

Description

A wrapper around `patchwork::wrap_plots()` that adds the usual finishing touches: duplicate legends collected into one, automatic panel tags (A, B, C and so on) and an overall title and subtitle.

Usage

```
arrange_plots(
  ...,
  ncol = NULL,
  nrow = NULL,
  guides = c("collect", "keep"),
  title = NULL,
  subtitle = NULL,
  tag_levels = "A"
)
```

Arguments

`...`	Plots to combine, or a single list of plots.
`ncol`, `nrow`	Layout dimensions, passed to patchwork.
`guides`	How to treat legends: `"collect"` (merge duplicates) or `"keep"`.
`title`, `subtitle`	Overall title and subtitle.
`tag_levels`	Panel tag style, such as `"A"`, `"1"` or `"i"`, or `NULL` for no tags.

Value

A 'patchwork' object.

Examples

```
p1 <- explore_distribution(crop_yield, yield)
p2 <- scatter_trend(crop_yield, fertiliser, yield)
arrange_plots(p1, p2, ncol = 2, title = "Crop yield", tag_levels = "A")
```

`binned_residual_plot` *Binned-residual plot for a generalised linear model*

Description

For a fitted `glm` (especially binary or count models), a plot of raw residuals against fitted values is hard to read because the residuals take only a few values. Following Gelman and Hill (2007), this plot instead splits the data into equal-count bins of fitted values and plots, for each bin, the mean residual against the mean fitted value, inside a grey band of ± 2 standard errors of that mean. Under a well-specified model, the binned residuals scatter around zero, and about 95% of them should fall within the band. Systematic departures, such as a trend or many points outside the band, indicate a misspecified mean structure.

Usage

```
binned_residual_plot(
  model,
  bins = NULL,
  type = c("response", "pearson"),
  point_colour = depictr_brand(),
  band_colour = depictr_accent(),
  title = NULL,
  band = c("model", "empirical")
)
```

Arguments

`model`	A fitted `glm` object (an `lm` is also accepted and treated as a Gaussian GLM).
`bins`	Number of bins. The default follows Gelman and Hill's rule of thumb of about $\sqrt{n}$ bins, rounded down and kept between 2 and half the number of observations.
`type`	Residual type used for the bin means: `"response"` (observed minus fitted on the response scale, the Gelman-Hill default) or `"pearson"`.
`point_colour, band_colour`	Colours for the bin points and the ± 2 SE band. Default to the depictr brand and accent colours.
`title`	Plot title.
`band`	Where the standard errors come from. The default, `"model"`, takes them from the variance the fitted model implies for each residual. `"empirical"` takes them from the spread of the residuals in each bin, as `arm::binnedplot()` does.

Details

By default (`band = "model"`), the standard error is the one the fitted model implies. The model gives the response y_i the variance $\phi V(\hat{\mu}_i)/w_i$, where V is the family's variance function, w_i the prior weight and ϕ the dispersion. The dispersion is 1 for the binomial and Poisson families, `summary(model)$dispersion` for the others and $\hat{\sigma}^2$ for an `lm`. Taking this as the variance of

the response residual $y_i - \hat{\mu}_i$, the band gives the mean of the n_b residuals in bin b the standard error $\sqrt{\sum_{i \in b} \phi V(\hat{\mu}_i)/w_i}/n_b$. A Pearson residual, $(y_i - \hat{\mu}_i)\sqrt{w_i/V(\hat{\mu}_i)}$, then has variance ϕ, so its bin mean has standard error $\sqrt{\phi/n_b}$. The fit absorbs a little of each observation's variance, so a residual varies slightly less than this, and the band is slightly too wide. Observations with zero prior weight take no part in the fit and are left out of the bins.

`band = "empirical"` draws the band that `arm::binnedplot()` draws: twice the standard deviation of the residuals in the bin, divided by $\sqrt{n_b}$. It shrinks to almost nothing wherever the outcomes in a bin are all alike. The residuals in a logistic model's bin without events all lie just below zero, so its band nearly vanishes and the bin is flagged. Where events are rare, such bins are what the model expects. The example below, a correctly specified model, has five of its seventeen bins flagged by that band and none by the model's.

Value

A ggplot2::ggplot object. The per-bin summary (mean fitted value, mean residual, number of observations `n`, the band's half-width `se2` and an `outside` flag) is attached as `attr(plot, "bins")`.

References

Gelman A, Hill J (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, Cambridge, UK. ISBN 978-0-521-68689-1. doi:10.1017/CBO9780511790942.

Examples

```
gfit <- glm(adverse_event ~ biomarker + age + arm,
            data = clinical_trial, family = binomial)
binned_residual_plot(gfit)

# The band arm::binnedplot() draws, which flags the bins that hold no events:
binned_residual_plot(gfit, band = "empirical")
```

`calibration_plot` *Calibration plot*

Description

Assesses how well predicted probabilities match observed frequencies. The scores are split into bins, and for each bin, the mean predicted probability is plotted against the observed event rate, with the diagonal marking perfect calibration. A Wilson binomial confidence interval is drawn on each bin's observed proportion so that bins backed by few observations are not over-interpreted. Pass a named list of models or of (actual, score) pairs to overlay several colour-coded calibration curves with a legend.

Usage

```
calibration_plot(
  x,
  score = NULL,
  bins = 10,
  colour = depictr_brand(),
  conf_level = 0.95,
  title = NULL
)
```

Arguments

x	A binomial glm, the vector of observed outcomes or a named list of models or of (actual, score) pairs to overlay.
score	When x is an outcome vector, the matching predicted probabilities, or a list of them for several models.
bins	Number of equal-count bins.
colour	Colour of the points and lines for a single model. Defaults to the depictr brand blue. Ignored when several models are overlaid.
conf_level	Confidence level for the per-bin Wilson interval on the observed proportion, a single number strictly between 0 and 1. Use NA to omit the intervals. Intervals are drawn only for a single model, to keep the overlay legible.
title	Plot title.

Value

A ggplot2::ggplot object.

Examples

```
# Calibration is judged against the base rate, so the example uses the rare
# clinical-trial adverse event (about 10% positive).
gfit <- glm(adverse_event ~ biomarker + age + arm,
            data = clinical_trial, family = binomial)
calibration_plot(gfit, bins = 6)
```

---

check_figure *Audit a finished figure for accessibility and honesty*

Description

Checks a finished figure as it will be submitted. palette_safety() shows that the eight colours depictr ships stay apart under colour-vision deficiency, but not how many of them a figure uses, what replaced them, how small the text will be once the figure is reduced to a journal column or whether colour alone separates two groups. check_figure() reads the built plot and answers those questions with numbers.

Usage

```
check_figure(
  plot,
  width_cm = 17.78,
  render_width_cm = 17.78,
  min_delta_e = 5,
  min_text_pt = 6,
  metric = c("ciede2000", "cie76"),
  max_text_pt = NULL
)
```

Arguments

plot	A plot, as returned by any depictr plotting function, including one extended afterwards with +, or a multi-panel composite (see 'Composites').
width_cm	The width, in centimetres, that the figure will occupy in the finished document. Defaults to 17.78 cm, the seven inches save_plot() draws at, which means no scaling.
render_width_cm	The width, in centimetres, that the figure is drawn at. Defaults to the same 17.78 cm. The ratio of the two widths is the factor every point size is multiplied by.
min_delta_e	The smallest acceptable colour difference, on the scale of metric, for the colour separability checks, and the smallest acceptable difference in lightness for the greyscale check. Defaults to 5, matching palette_safety().
min_text_pt	The smallest acceptable printed text size, in points, for the text around the panel and the text inside it alike. Defaults to 6, below the floor most journals set (see 'Auditing at the printed size').
metric	The colour difference the colour separability checks measure: "ciede2000", the default, or "cie76", the Euclidean distance in CIELAB. "cie76" is deprecated and gives a warning once per session. It is kept so that results computed with it can be reproduced, and a later release will remove it.
max_text_pt	The largest acceptable printed text size, in points, or NULL, the default, for no ceiling. When given, a text_size_max row is appended (see 'What each check measures').

Details

Every row gives the value it measured next to the threshold it was measured against, so that each verdict can be weighed. A check passes when the measured value is at least the threshold. The one exception is text_size_max, a ceiling added when max_text_pt is given, which passes when the measured value is at most its threshold. A check that has nothing to measure, such as colour separability on a figure that encodes nothing by colour, reports NA and the verdict "not applicable", which does not count as a pass.

Value

A data frame with one row per check and columns check, measured, threshold, verdict ("pass", "fail" or "not applicable") and detail, a short note naming what produced the measurement.

There are ten rows, and an eleventh, `text_size_max`, when `max_text_pt` is given. For a composite, the rows of each panel in turn, after a leading integer column `panel`.

What each check measures

Colours and text are measured as drawn. A translucent colour is composited over the background it is drawn on with the alpha it is drawn with, so it is measured as the paler colour the reader sees. As ggplot2 draws them, bars, tiles, areas, densities, ribbons, polygons, violins, boxplots, crossbars and smooths apply a layer's alpha to their fill and leave the outline opaque. Lines, paths, segments, error bars and reference lines apply it to their colour, and points apply it to both. Any other geom is taken to apply it to both, the cautious assumption. A colour that carries its own alpha keeps it when the layer sets none, as ggplot2 draws it. A colour drawn at several alphas within one layer is measured at the smallest. Where marks overlap one another, the colour that shows depends on what lies beneath it, and that is not measured.

`colour_separability` is the smallest colour difference between two of the figure's encoding colours, by the CIEDE2000 formula (Sharma, Wu & Dalal, 2005) unless `metric` asks for CIE76. The three `colour_separability_*` rows repeat that measurement after simulating each dichromacy at full severity with `simulate_cvd()`. Encoding colours are the distinct colour and fill values that tell groups apart. A layer contributes its colours when it maps colour or fill to a discrete scale, even if it draws a single colour, or when it draws two or more. Groups drawn one per layer are therefore audited as they would be in one layer. A colour set as a constant, as in `geom_point(colour = "red")`, is left out, even when another layer sets a different one, and so is a lone colour passed through `ggplot2::scale_colour_identity()`. The audit cannot tell such a colour from decoration, so a colour meant to tell groups apart should be mapped, with `ggplot2::scale_colour_manual()` to fix its values. The outline colours, across every layer, are compared among themselves, and so are the fill colours, and the row reports the closest pair from either set. An outline is therefore never compared with its own fill, nor a group with itself as another layer draws it.

A continuous colour or fill scale, such as that of a heatmap, encodes values along a ramp, and none of the colour rows audits it. When nothing else in the figure encodes by colour, each of them abstains with a `detail` that says so. Whether a ramp reads in order, to a reader with a colour-vision deficiency and in greyscale as well, depends on how evenly its lightness changes, which the audit does not measure. A perceptually uniform map, such as viridis, is preferable to a rainbow (Crameri et al., 2020). depictr's own sequential and diverging ramps are perceptually ordered.

`greyscale_separability` is the smallest difference in CIE lightness between those same colours, which is what survives printing in black and white. It is the absolute difference in lightness, whichever `metric` the colour rows use.

`text_size` is the smallest point size of the text the theme draws around the panel: axis labels and titles, legends, facet strips, titles and captions. It is scaled by `width_cm / render_width_cm`, so a figure drawn seven inches wide and printed in an 8.9 cm column has every point size halved. Text drawn inside the panel is measured by `annotation_text_size`.

`text_contrast` is the WCAG 2.2 contrast ratio of the text that falls furthest short of its own threshold. Each text is measured against the background it is drawn on. Strip text is measured against the facet strip's fill, legend text against the legend box's fill and other text against the plot background. A translucent fill is measured as the colour it shows over what lies beneath it, and so is text drawn with an alpha. Text that is large at its printed size, 18 pt or 14 pt in bold, is held to 3:1, and other

text to 4.5:1, as success criterion 1.4.3 asks. The row's `threshold` is the one that applies to the text it reports, and its `detail` names that text's colour, background, printed size and weight.

`geometry_contrast` is the smallest WCAG 2.2 contrast ratio of any encoded group's drawn colour against the panel background, or against the plot background where the panel paints none. Where a layer maps its outline colour to the same groups as its fill, each group counts the better of the two, since WCAG lets a border carry a graphical object's contrast. For a translucent colour, the `detail` names the colour, its alpha and the colour it is drawn as.

`redundant_encoding` counts how many of shape and line type also vary across the layers that draw the encoding colours. They are gathered as the colours are. A layer contributes the values it maps to a discrete scale, or the two or more it draws, and a shape or line type set as a constant is left out. Zero means that colour alone tells the groups apart.

`annotation_text_size` is the smallest point size of the text drawn inside the panel, scaled as `text_size` is and held to the same `min_text_pt`. That text comes from layers such as `ggplot2::geom_text()` and `ggplot2::geom_label()`, from annotations and from depictr's own cell values and labels. Its contrast is not measured, since it sits on the marks, which give it no single background. A label that a layer builds only when the figure is drawn, as ggrepel builds its repelled labels, is not yet in the built plot, so it is not measured either.

`text_size_max` is reported only when `max_text_pt` is given, as the last row. It is the largest point size of all the text measured, around the panel and inside it, scaled as `text_size` is. It is the one row that passes at or below its threshold.

Limitations of the default palette

The eight-colour qualitative palette clears the colour-separability checks with a margin but fails `greyscale_separability`: its orange (`#e69f00`) and sky blue (`#56b4e9`) differ by only 0.79 in lightness, so they print as the same grey. depictr's colourblind-safety guarantee covers hue confusion only. A figure that may be printed in black and white should use fewer groups or a sequential palette, or add a redundant shape or line type. The first four colours of the palette are not safe in greyscale either, since the bluish green and the vermillion differ by 3.55. The threshold is not lowered to let the package's own defaults pass.

The palette falls short on contrast as well. Four of its eight colours are below the 3:1 against white that WCAG asks of a graphical object: the orange at 2.25:1, the sky blue at 2.31:1, the grey at 2.85:1 and the yellow at 1.32:1. The orange is the second colour, so a default figure on white whose colour tells two or more groups apart fails `geometry_contrast`, and drawing its marks translucent lowers the ratio further. Opaque marks, the vermillion (`#d55e00`, at 3.87:1) in the orange's place and a redundant shape clear that row and `redundant_encoding`, as the first example below shows. The contrast-first order of `depictr_palette()` puts the vermillion second and the reddish purple third, and `scale_colour_depictr()` takes it as `order = "contrast"`. Up to three groups then pass every colour row, and four keep 3:1 against white.

Auditing at the printed size

Text is drawn in points, so the size it prints at depends on the two widths. A figure drawn at the size it will be printed, as `save_plot()` draws one given `width = 8.9, units = "cm"`, needs no scaling. Pass the saved width as both `width_cm` and `render_width_cm`, or leave both at their defaults. Given alone, `width_cm` audits the figure as though it had been drawn seven inches wide and then shrunk to fit. `save_plot()` with `check = TRUE` audits a figure at the width it saves it at, given as both.

The default `min_text_pt` of 6 is lenient. As their guidelines stood on 30 September 2026, PLOS asked for text of 8 to 12 pt and APA Style for 8 to 14 pt. Elsevier asked for 7 pt as a rule of thumb, and Nature for 5 to 7 pt, with panel labels at 8 pt. Pass the floor of the journal the figure is for as `min_text_pt`, and its ceiling, where it sets one, as `max_text_pt`. Nature's 7 pt applies to all text but the panel labels. The tags of a composite, where patchwork puts them, are not measured, but a panel label drawn as a panel's title or annotation is measured with the rest. The four pages are listed under References.

Composites

A composite, such as `arrange_plots()`, `residual_diagnostics_plot()` and `model_report()` return, is audited panel by panel. Each panel has its own scales, theme and text, so each is audited as it would be on its own. The reports are stacked under a leading integer column, `panel`, which only the report of a composite has. The widths are those of the whole composite. A panel's text keeps its point size whatever share of the width the panel takes, so the text of every panel scales by the same ratio.

Panels are numbered depth-first through any composite nested inside, in the order the composite holds them. That is the order patchwork tags them in, and the reading order of a grid filled row by row. A spacer or a guide area draws no figure of its own and takes no number. A wrapped element, such as a grob or a table, cannot be read, so it keeps its number and is left out of the report with a warning.

A panel's legend is measured with the panel, as the panel would draw it on its own, even when `guides = "collect"` gathers it into one legend for the composite. What the composite draws around its panels is not measured: the title, subtitle and caption of `patchwork::plot_annotation()` and the panel tags.

References

APA Style. Figures. https://apastyle.apa.org/style-grammar-guidelines/tables-figures/figures

Crameri F, Shephard GE, Heron PJ (2020). "The misuse of colour in science communication." *Nature Communications*, **11**(1), 5444. doi:10.1038/s41467020191607.

Elsevier. Artwork sizing. https://www.elsevier.com/about/policies-and-standards/author/artwork-and-media-instructions/artwork-sizing

Machado GM, Oliveira MM, Fernandes LAF (2009). "A physiologically-based model for simulation of color vision deficiency." *IEEE Transactions on Visualization and Computer Graphics*, **15**(6), 1291–1298. doi:10.1109/TVCG.2009.113.

Nature. Guide to preparing final artwork. https://www.nature.com/documents/nature-final-artwork.pdf

PLOS ONE. Figures. https://journals.plos.org/plosone/s/figures

Sharma G, Wu W, Dalal EN (2005). "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations." *Color Research & Application*, **30**(1), 21–30. doi:10.1002/col.20070.

World Wide Web Consortium (2024). "Web Content Accessibility Guidelines (WCAG) 2.2." W3C Recommendation, https://www.w3.org/TR/WCAG22/. Accessed 2026-08-18.

See Also

palette_safety() for the palette in the abstract, and palette_preview() to look at it.

Examples

```
library(ggplot2)

# A figure that clears every check it can be held to: two well-separated
# colours, a redundant shape and text left at the size it was drawn. It
# draws no text inside the panel, so that row is not applicable.
good <- ggplot(crop_yield, aes(rainfall, yield, colour = treatment,
                               shape = treatment)) +
  geom_point() +
  scale_colour_manual(values = c("#005b96", "#d55e00")) +
  theme_depictr()
check_figure(good)

# The same figure destined for an 8.9 cm journal column, where the text is
# half the size it looks on screen.
check_figure(good, width_cm = 8.9)

# Saved at 8.9 cm in the first place, with save_plot(width = 8.9,
# units = "cm"), it is drawn at the size it prints, so both widths are 8.9.
check_figure(good, width_cm = 8.9, render_width_cm = 8.9)

# Nature allowed no text above 7 pt but the panel labels. The axis titles
# here are 11 pt, so the ceiling row fails.
subset(check_figure(good, max_text_pt = 7), check == "text_size_max")

# Each panel of a composite is audited, and a column says which panel each
# row describes. At base size 5, the second panel's text is too small.
small <- good + theme_depictr(base_size = 5)
both <- check_figure(arrange_plots(good, small))
both[both$verdict == "fail", c("panel", "check", "measured")]
```

`clinical_trial` *Simulated two-arm clinical trial*

Description

A reproducibly simulated randomised clinical trial, for time-to-event (survival) and imbalanced-classification examples. The two arms differ in survival: the `treatment` arm has a lower hazard, so the Kaplan-Meier curves separate clearly and the log-rank test is highly significant, and the treatment arm, whose event times are longer, is more often right-censored by the 36-month study end. `age` and `biomarker` also affect the hazard. The data suit survival_plot() (pass `time`, `event` and `arm`, or a `survival::survfit` object).

Usage

```
clinical_trial
```

Format

A data frame with 300 rows and 7 variables:

patient Patient identifier.

arm Treatment arm (factor): `placebo` or `treatment`.

age Age in years at enrolment.

biomarker Standardised baseline laboratory value.

time Follow-up time, in months.

event Event indicator: 1 event observed, 0 right-censored.

adverse_event Rare adverse-event indicator: 1 yes, 0 no (about 10% positive).

Details

`adverse_event` is a deliberately rare binary safety outcome (a base rate of about 10%) that `arm`, `age` and `biomarker` predict. It gives the precision-recall (`pr_curve_plot()`), cumulative-gains (`gain_plot()`), lift (`lift_plot()`) and calibration (`calibration_plot()`) examples an imbalanced target, where those charts are most informative.

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed, and they describe no real individuals.

Source

Simulated by `data-raw/generate_datasets.R` in the source repository.

`cluster_plot` *Cluster scatter plot*

Description

Runs k-means clustering on the numeric columns of a data frame, or uses a supplied cluster assignment, and plots the observations on the first two principal components, coloured by cluster, with optional convex hulls and labelled centroids. Because the plot uses principal components, it works for any number of variables.

Usage

```
cluster_plot(
  data,
  cols = NULL,
  k = 3,
  clusters = NULL,
  scale = TRUE,
  suggest_k = NULL,
  k_range = NULL,
  hulls = TRUE,
```

```
  label_centers = TRUE,
  point_alpha = 0.8,
  palette = NULL,
  title = NULL,
  seed = NULL,
  nstart = 10,
  iter.max = 10
)
```

Arguments

data	A data frame.
cols	Numeric columns to cluster on (by default, all numeric columns).
k	Number of clusters for k-means. Ignored if clusters is supplied, and overridden when suggest_k chooses a value.
clusters	Optional vector of cluster assignments, such as one from stats::kmeans() or stats::cutree(), to plot a clustering computed beforehand. Hierarchical or PAM assignments (such as stats::cutree() output matching dendrogram_plot()) are accepted. Must have exactly one entry per row of data. Rows with missing values in cols are dropped from both before plotting.
scale	Whether to scale variables to unit variance before clustering and the PCA. Columns with zero or near-zero variance cannot be scaled and are dropped with a message, as in correlation_heatmap().
suggest_k	Optionally choose k automatically with a cluster-quality diagnostic in place of the supplied k. Either TRUE (the average-silhouette criterion), a string naming the criterion ("silhouette", "wss" or "gap", see k_diagnostic()) or NULL (the default) to leave k unchanged. Ignored when clusters is supplied.
k_range	Candidate values of k searched when suggest_k is set. The default, NULL, searches 1:8 for "gap" and "wss" and 2:8 for "silhouette", as k_diagnostic() does, so the gap statistic can suggest a single cluster.
hulls	Whether to draw a shaded convex hull around each cluster.
label_centers	Whether to label the cluster centroids.
point_alpha	Point transparency.
palette	Colours for the clusters. Defaults to depictr_palette().
title	Plot title.
seed	Optional integer seed for reproducible k-means, ignored when clusters is supplied. It applies to this call alone, and the caller's random-number state is restored afterwards. The default, NULL, takes the random starts of stats::kmeans(), and any random draws of a suggest_k diagnostic, from the session's random-number stream, so call set.seed() first for a reproducible result.
nstart	Number of random starts for stats::kmeans().
iter.max	Maximum iterations for stats::kmeans().

Value

A ggplot2::ggplot object. When `suggest_k` is used, the chosen `k` and the full diagnostic are attached as the attribute `"k_diagnostic"`.

See Also

`k_diagnostic()` and `silhouette_plot()` for cluster-quality diagnostics.

Examples

```
cluster_plot(crop_yield, cols = c("rainfall", "fertiliser", "soil_ph",
                                  "yield"), k = 3, seed = 1)
# Let a silhouette diagnostic choose k:
cluster_plot(crop_yield, cols = c("rainfall", "fertiliser", "soil_ph",
                                  "yield"), suggest_k = TRUE, k_range = 2:6,
             seed = 1)
```

---

coefficient_plot *Forest (coefficient) plot*

Description

Draws a horizontal point-and-interval plot, or forest plot, of model estimates. The input can be a fitted model that `tidy_estimates()` reads or a data frame of pre-computed estimates.

Usage

```
coefficient_plot(
  x,
  conf_level = 0.95,
  intercept = FALSE,
  order = c("none", "ascending", "descending"),
  labels = NULL,
  interaction = c("times", "asterisk", "colon", "space"),
  point_colour = depictr_brand(),
  reference_colour = depictr_reference(),
  reference_line = if (exponentiate) 1 else 0,
  point_size = 2.2,
  line_size = 0.7,
  facet = FALSE,
  scales = c("fixed", "free"),
  standardise = FALSE,
  sd_multiple = 1,
  exponentiate = FALSE,
  title = NULL,
  subtitle = NULL,
  x_lab = NULL
)
```

Arguments

x	A fitted model or a tidy data frame of estimates.
conf_level	Confidence or credible level, a single number strictly between 0 and 1. It is passed to tidy_estimates(), which uses it for a model and for the limits it fills in from the standard errors of a data frame.
intercept	Whether to keep the intercept term. Defaults to FALSE, since the intercept is seldom of interest on a forest plot and its size often compresses the scale of the other terms.
order	Order of the terms. "ascending" puts the smallest estimate at the bottom and the largest at the top, so estimates increase up the axis. "descending" does the reverse, and "none" keeps model order from top to bottom. The faceted layout (facet = TRUE) stacks its panels the other way up.
labels	Optional display labels for the terms. An unnamed vector gives one label per term in model order (the order tidy_estimates() returns, after the intercept is dropped unless intercept = TRUE), whatever order is. A named vector mapping raw term names to labels, such as c(soil_ph = "Soil pH"), is safer. If NULL, the coefficient of a factor level is labelled with its variable and level, inside interaction terms too, so the contrast of enhanced with the reference level reads treatment: enhanced. Other names are tidied with format_terms().
interaction	Passed to format_terms() to control how interaction terms are rendered, in the automatic labels and for any term a named labels vector leaves out. It is ignored when labels is an unnamed vector.
point_colour, reference_colour	Colours for the estimates and the reference line.
reference_line	Position of a vertical reference line. Defaults to 0, or to 1 when exponentiate = TRUE, where a ratio of 1 means no effect. Use NA to omit it.
point_size, line_size	Size of the points and interval lines.
facet	Whether to give each term its own panel with a free x-axis, laid out one per row. This keeps terms on very different scales readable, such as a large intercept beside small slopes. A shortcut for scales = "free". Defaults to FALSE, the shared-axis layout.
scales	Either "fixed" (the default, a single shared x-axis) or "free" (one free-scaled panel per term). When facet = TRUE, it is set to "free".
standardise	Whether to scale each coefficient by the spread of its inputs, following Gelman (2008). A continuous input (numeric, with more than two distinct values) is scaled by sd_multiple times its standard deviation in the rows the model used, so its slope becomes the change in the outcome per SD. Factor, character, logical and two-valued numeric inputs, such as one coded 0 and 1, are left unscaled, since a contrast already compares two groups and its per-SD value would depend on how balanced the groups happen to be. An interaction is scaled through its inputs: rainfall:treatment by the SD of rainfall, and a product of two continuous inputs by both their SDs. Matrix-valued terms such as poly() and ns() are left unscaled, with a message. An interaction of continuous inputs draws a warning unless they are all centred. Otherwise, each input's main effect

is its slope where the others are zero, and Gelman's remedy is to centre the inputs and refit. A continuous slope and a contrast are comparable only roughly, most nearly with `sd_multiple = 2`. Requires a fitted model, and is ignored with a warning for a tidy data frame. Defaults to `FALSE`.

`sd_multiple` How many standard deviations of a continuous input one unit of the standardised scale spans when `standardise = TRUE`. Defaults to 1. Use 2 for Gelman's (2008) convention, under which the coefficient of a continuous input is comparable to that of a balanced binary one.

`exponentiate` Whether to draw the exponentiated estimates and intervals on a log10 axis, which shows the coefficients of a logistic regression as odds ratios and those of a Poisson or negative binomial model as rate ratios. It applies after `standardise`, so a standardised slope becomes a ratio per SD of its predictor. A data frame `x` must hold estimates on the link scale. Defaults to `FALSE`.

`title`, `subtitle`, `x_lab`
Plot title, subtitle and x-axis label. `x_lab` defaults to 'Estimate', or, when `standardise`, 'Estimate per SD of predictor' ('per 2 SD' with `sd_multiple = 2`), followed by '(contrasts unscaled)' when a factor or binary term is drawn. The label names the convention on the figure itself, because only the predictors are rescaled here: a fully standardised beta would rescale the outcome too. With `exponentiate = TRUE`, 'Estimate' gives way to the name of the ratio: 'Odds ratio' for a logit link, 'Rate ratio' for a log link in a Poisson, quasi-Poisson or negative binomial model, 'Risk ratio' for a binomial model with a log link and 'exp(Estimate)' otherwise. The last covers an `lm(log(y) ~ x)` slope, whose exponential is a ratio of geometric means.

Value

A ggplot2::ggplot object.

References

Gelman A (2008). "Scaling regression inputs by dividing by two standard deviations." *Statistics in Medicine*, **27**(15), 2865–2873. doi:10.1002/sim.3107.

Examples

```
fit <- lm(yield ~ rainfall + fertiliser + soil_ph + treatment,
          data = crop_yield)
coefficient_plot(fit)

# Order terms and add a title
coefficient_plot(fit, order = "descending", title = "Drivers of crop yield")

# When an intercept or a large term compresses the rest, give each term its
# own free-scaled panel:
coefficient_plot(fit, intercept = TRUE, facet = TRUE)

# Or scale the continuous slopes per SD of their predictors, leaving the
# treatment contrast in the outcome's units:
coefficient_plot(fit, standardise = TRUE)
```

```
# Odds ratios from a logistic regression, on a log axis with the line at 1:
gfit <- glm(accuracy ~ word_frequency + condition,
            data = lexical_decision, family = binomial)
coefficient_plot(gfit, exponentiate = TRUE)
```

`compare_models` *Compare estimates from several models or sources*

Description

Overlays the estimates from two or more models, or tidy estimate tables, on a single forest plot, with one colour per source. It underlies `frequentist_bayesian_plot()` (frequentist against Bayesian) and serves equally well for comparing nested models, several optimisers or estimates before and after a transformation.

Usage

```
compare_models(
  ...,
  names = NULL,
  conf_level = 0.95,
  intercept = FALSE,
  order = c("none", "ascending", "descending"),
  labels = NULL,
  interaction = c("times", "asterisk", "colon", "space"),
  dodge_width = 0.6,
  reference_line = if (exponentiate) 1 else 0,
  palette = NULL,
  point_size = 2.2,
  line_size = 0.7,
  facet = FALSE,
  scales = c("fixed", "free"),
  exponentiate = FALSE,
  legend_title = "Source",
  legend_ncol = 1,
  title = NULL,
  subtitle = NULL,
  x_lab = NULL
)
```

Arguments

`...`	Two or more fitted models or tidy data frames of estimates, in any mix. Name the arguments to label the sources, as in `compare_models(Frequentist = m1, Bayesian = m2)`.
`names`	Optional character vector of source labels, overriding the names of `...`.

`conf_level` Confidence or credible level for models, a single number strictly between 0 and 1.

`intercept` Whether to keep the intercept term. Defaults to `FALSE`.

`order` Order of the terms, by their average estimate across sources. `"ascending"` puts the smallest estimate at the bottom and the largest at the top, so estimates increase up the axis. `"descending"` does the reverse, and `"none"` keeps the terms in the order they first appear, from top to bottom. The faceted layout (`facet = TRUE`) stacks its panels the other way up.

`labels` Optional display labels. An unnamed vector gives one label per term, whatever `order` is. The terms are taken in the order they first appear across the sources, each source in the order `tidy_estimates()` returns, after the intercept is dropped unless `intercept = TRUE`. Terms drawn on one row, such as `stress` and brms's `b_stress`, take one label. A named vector mapping raw term names to labels, such as `c(soil_ph = "Soil pH")`, is safer. If `NULL`, the terms are labelled as in `coefficient_plot()`, each factor level with its variable and level (`treatment: enhanced`), from every fitted model among the sources.

`interaction` Passed to `format_terms()`.

`dodge_width` Vertical spacing between sources sharing a term.

`reference_line` Position of a vertical reference line. Defaults to 0, or to 1 when `exponentiate = TRUE`. Use `NA` to omit it.

`palette` Colours for the sources. Defaults to `depictr_palette()`.

`point_size, line_size`
Point and interval-line sizes.

`facet` Whether to give each term its own panel with a free x-axis, laid out one per row, so that terms on very different scales, such as a large intercept beside small slopes, stay legible. The faceted layout keeps the source overlay and its single legend. A shortcut for `scales = "free"`. Defaults to `FALSE`.

`scales` Either `"fixed"` (the default, a single shared x-axis) or `"free"` (one free-scaled panel per term). When `facet = TRUE`, it is set to `"free"`.

`exponentiate` Whether to draw every source's exponentiated estimates and intervals on a log10 axis, as odds or rate ratios. Each model is exponentiated by `tidy_estimates()`, and a data frame among the sources must hold estimates on the link scale, since it is exponentiated too. Defaults to `FALSE`.

`legend_title, legend_ncol`
Legend title and number of columns.

`title, subtitle, x_lab`
Title, subtitle and x-axis label. `x_lab` defaults to 'Estimate'. With `exponentiate = TRUE`, it defaults to the name of the ratio, as in `coefficient_plot()`, when the fitted models agree on it, and to 'exp(Estimate)' when they do not.

Value

A ggplot2::ggplot object.

Examples

```
m1 <- lm(yield ~ rainfall + fertiliser + soil_ph, data = crop_yield)
m2 <- lm(yield ~ rainfall + fertiliser + soil_ph,
         data = crop_yield[crop_yield$treatment == "standard", ])
compare_models(`All fields` = m1, `Standard only` = m2,
                       title = "Estimates by subset")

# Keep the intercept legible beside the slopes:
compare_models(`All fields` = m1, `Standard only` = m2,
               intercept = TRUE, facet = TRUE)

# Odds ratios from two logistic regressions:
g1 <- glm(accuracy ~ word_frequency + condition,
          data = lexical_decision, family = binomial)
g2 <- glm(accuracy ~ word_frequency + condition + modality,
          data = lexical_decision, family = binomial)
compare_models(Reduced = g1, Full = g2, exponentiate = TRUE)
```

---

confusion_matrix_plot *Confusion matrix heatmap*

Description

Cross-tabulates predicted against actual classes and displays the counts as a heatmap. The input can be a fitted binomial `glm` (with a probability threshold) or a pair of vectors of actual and predicted classes. Each count is drawn in black or white, whichever has the higher contrast against its cell.

Usage

```
confusion_matrix_plot(
  x,
  predicted = NULL,
  threshold = 0.5,
  normalise = c("none", "row", "col"),
  title = NULL
)
```

Arguments

`x`	A binomial `glm`, or the vector of actual classes.
`predicted`	When `x` is an actual-class vector, the matching predicted classes.
`threshold`	When `x` is a `glm`, the probability threshold for the positive class. As well as a number in `[0, 1]`, you may pass the string `"youden"` to reuse the Youden's J optimal threshold (the same operating point `roc_curve_plot()` marks), so the confusion matrix and the ROC curve agree on the cut-off.
`normalise`	One of `"none"`, `"row"` (by actual class) or `"col"` (by predicted class). Controls the fill shading and the cell annotation.
`title`	Plot title.

Value

A ggplot2::ggplot object. The threshold used is stored in `attr(plot, "threshold")`.

Examples

```
gfit <- glm(accuracy ~ word_frequency + RT + condition,
            data = lexical_decision, family = binomial)
confusion_matrix_plot(gfit, threshold = 0.5)

# Reuse the Youden-optimal operating point.
confusion_matrix_plot(gfit, threshold = "youden")
```

`correlation_heatmap` *Plot a correlation matrix as a heatmap*

Description

Computes the pairwise correlations between the numeric columns of a data frame and displays them as a colour-coded heatmap, optionally annotated with the correlation values.

Usage

```
correlation_heatmap(
  data,
  cols = NULL,
  method = "pearson",
  use = "pairwise.complete.obs",
  show_values = TRUE,
  digits = 2,
  palette = depictr_palette(5, "diverging")[c(1, 3, 5)],
  reorder = FALSE,
  title = NULL
)
```

Arguments

`data`	A data frame.
`cols`	Optional character vector of columns to include, each numeric or logical. Any other column, such as a factor or a date, is an error that names it. If `NULL`, all numeric columns are used.
`method`	Correlation method: `"pearson"`, `"spearman"` or `"kendall"`.
`use`	Missing-value handling passed to `stats::cor()`.
`show_values`	Whether to annotate each cell with its correlation, drawn in black or white, whichever has the higher contrast against the cell.
`digits`	Number of decimal places for the annotations.

palette	Length-3 vector of colours for the lowest, mid (zero) and highest correlations. Defaults to the endpoints and midpoint of the colourblind-aware depictr_palette() diverging ramp (negative correlations red, zero neutral, positive correlations brand blue).
reorder	Whether to reorder the variables by hierarchical clustering of the correlation matrix (using $1 - r$ as the distance), so that blocks of mutually correlated variables sit together and structure is easier to see. Defaults to FALSE (the order of cols). Skipped with a message if any correlation is undefined (NA).
title	Plot title.

Details

Columns with zero or near-zero variance cannot be correlated and are dropped automatically with an informative message, so the warning "the standard deviation is zero" from stats::cor() does not appear. If, after dropping them, any cells still come out NA, such as those of two variables that never co-occur under "pairwise.complete.obs", those cells are drawn in grey and labelled n/a, not left blank.

Value

A ggplot2::ggplot object.

Examples

```
correlation_heatmap(wellbeing_survey)
correlation_heatmap(crop_yield, method = "spearman", show_values = FALSE)
# Cluster correlated variables together:
correlation_heatmap(wellbeing_survey, reorder = TRUE)
```

---

crop_yield *Simulated crop-yield field trial*

Description

A reproducibly simulated agronomy field trial relating crop yield to rainfall, fertiliser, soil pH and a management treatment, for the regression, coefficient-plot, scatter-trend and interaction examples. The data-generating process contains a fertiliser-by-treatment interaction: fertiliser raises yield far more under the enhanced treatment than under standard, so interaction_plot(lm(yield ~ fertiliser * treatment + ..., crop_yield), "fertiliser", "treatment") shows diverging slopes that are not noise. The main effects (rainfall, fertiliser, soil pH and treatment) are retained.

Usage

```
crop_yield
```

Format

A data frame with 200 rows and 6 variables:

field Field identifier.

treatment Management treatment (factor): `standard` or `enhanced`.

rainfall Seasonal rainfall, in millimetres.

fertiliser Fertiliser applied, in kilograms per hectare.

soil_ph Soil pH.

yield Crop yield, in tonnes per hectare.

Details

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed.

Source

Simulated by `data-raw/generate_datasets.R` in the source repository.

`decompose_plot` *Time-series decomposition plot*

Description

Decomposes a seasonal time series into trend, seasonal and remainder components and shows them, with the original series, as stacked panels.

Usage

```
decompose_plot(
  x,
  frequency = NULL,
  method = c("stl", "classical"),
  robust = FALSE,
  s_window = "periodic",
  confidence = FALSE,
  level = 0.95,
  title = NULL
)
```

Arguments

`x`	A `ts` object, or a numeric vector (then `frequency` is required), holding one series. A multivariate `ts` or a matrix with several columns is refused. Missing values at either end are trimmed, and a missing value inside the series is an error.
`frequency`	Number of observations per period, such as 12 for monthly data. Taken from `x` when it is a `ts`.

method	"stl" (loess-based, Cleveland et al., 1990) or "classical" (stats::decompose()). STL needs more than two full cycles of data, and classical decomposition at least two.
robust	For method = "stl", whether to fit a robust STL (stats::stl() with robust = TRUE), which down-weights outliers in the loess fits so that an unusual point has less influence on the trend and seasonal components. Ignored for the classical method.
s_window	For method = "stl", the seasonal window, passed to stats::stl() as s.window. The default, "periodic", estimates one seasonal value per position in the cycle and repeats it in every cycle, as classical decomposition does. An odd whole number of at least 7 lets the seasonal component change from cycle to cycle. It is the number of consecutive cycles that go into each seasonal value, so a smaller window lets the pattern change faster. Ignored for the classical method.
confidence	Whether to draw a confidence ribbon around the trend component. The band is a normal-approximation interval based on the remainder's standard deviation (see Details).
level	Coverage of the trend confidence ribbon (a single number strictly between 0 and 1).
title	Plot title.

Details

The trend confidence ribbon is a pragmatic normal-approximation band: `trend +/- z * sd(remainder)`, where `z = qnorm((1 + level) / 2)` and the remainder standard deviation is computed on the non-missing remainder values. It shows the scale of the unexplained variation around the smoothed trend and is not a formal sampling-distribution interval.

Value

A 'patchwork' object (printable like a ggplot2::ggplot).

References

Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990). "STL: A seasonal-trend decomposition procedure based on loess." *Journal of Official Statistics*, **6**(1), 3–73.

See Also

ts_forecast(), seasonal_plot()

Examples

```
decompose_plot(AirPassengers)

decompose_plot(AirPassengers, method = "classical")
# Robust STL with a confidence ribbon on the trend
decompose_plot(AirPassengers, robust = TRUE, confidence = TRUE)
# A seasonal component free to grow with the level of the series
decompose_plot(AirPassengers, s_window = 13)
```

`dendrogram_plot` *Dendrogram*

Description

Hierarchical-clustering dendrogram, drawn with ggplot2 so that it shares the package theme. Accepts a data frame (from which the distance matrix and clustering are computed), a stats::dist object or an stats::hclust object. Optionally cuts the tree into `k` clusters, colouring the leaf labels and drawing the cut height.

Usage

```
dendrogram_plot(
  x,
  cols = NULL,
  distance = "euclidean",
  method = "complete",
  scale = TRUE,
  k = NULL,
  horizontal = FALSE,
  labels = NULL,
  palette = NULL,
  title = NULL
)
```

Arguments

`x`	A data frame, a `dist` object or an `hclust` object.
`cols`	When `x` is a data frame, the numeric columns to use.
`distance`	Distance measure passed to `stats::dist()`.
`method`	Linkage method passed to `stats::hclust()`.
`scale`	Whether to scale variables before computing distances, for data frame input.
`k`	Optional number of clusters to highlight (between 1 and the number of leaves). The cut-height line is drawn only when `2 <= k < n`.
`horizontal`	Whether to draw the tree horizontally.
`labels`	Whether to print the leaf labels. `NULL` (the default) chooses automatically: labels are shown for small trees (up to 40 leaves) and left out of larger ones, where they would overlap and be unreadable. `TRUE` or `FALSE` forces them on or off. When labels are hidden and `k` is set, a coloured strip of leaf ticks along the bottom of the tree shows the cluster membership.
`palette`	Colours for the `k` clusters. Defaults to `depictr_palette()`.
`title`	Plot title.

Value

A ggplot2::ggplot object.

Examples

```
# Cluster the survey regions by their wellbeing averages
d <- aggregate(cbind(stress, sleep_hours, life_satisfaction) ~ region,
               data = wellbeing_survey, FUN = mean)
rownames(d) <- d$region
dendrogram_plot(d[-1], k = 2)
```

---

depictr_options *Get or set the depictr look-and-feel options*

Description

depictr reads a small set of global options so that you can configure the shared look of every plot once, at the top of a script or in your `.Rprofile`, instead of passing the same arguments to each function. The options are honoured by `theme_depictr()` (base size, base family and the brand colour used for titles), by `depictr_palette()` and the `scale_colour_depictr()` family (an optional custom qualitative palette), and by every plotting function that draws a single series, a highlighted element or a reference line. Those functions list `depictr_brand()`, `depictr_accent()` or `depictr_reference()` as the default of a colour argument. The three are internal and are not called directly: each reads its option when the plotting function runs, so the colours are set here.

Usage

```
depictr_options(
  base_size,
  base_family,
  brand,
  accent,
  reference,
  palette,
  na_value
)
```

Arguments

`base_size`	Base font size in points for `theme_depictr()`.
`base_family`	Base font family for `theme_depictr()`.
`brand`	The depictr brand colour, used for plot titles and single-series geoms. It matches the default palette's first colour but does not change a palette, which `palette` sets.
`accent`	A secondary highlight colour, for the one element a figure draws attention to.
`reference`	The colour used for reference and annotation lines.

palette	An optional custom qualitative palette: a character vector of hex colours used by depictr_palette() (type "qualitative") and the discrete scales in place of the eight built-in colours. NULL restores the built-in palette. depictr_palette(order = "contrast") here draws every later plot in the contrast-first order (see depictr_palette()).
na_value	Colour used for NA levels by scale_colour_depictr() and scale_fill_depictr().

Details

Called with no arguments, depictr_options() returns the currently resolved values (option if set, otherwise package default). Called with named arguments, it sets the matching options(depictr.<name> =) entries and returns the previous resolved values invisibly, so the pattern old <- depictr_options(...); on.exit(do.call restores them. Pass NULL for an argument to clear that option and fall back to the package default.

Value

A named list of the resolved option values. When setting, the previous values are returned invisibly.

Examples

```
# Inspect the current settings
depictr_options()

# Set a larger base size and a different accent, then restore
old <- depictr_options(base_size = 14, accent = "#e69f00")
theme_depictr()              # now uses base_size 14
do.call(depictr_options, old)

# Use a custom qualitative palette everywhere
old <- depictr_options(palette = c("#1b9e77", "#d95f02", "#7570b3"))
depictr_palette(3)
do.call(depictr_options, old)

# Or the built-in palette in its contrast-first order
old <- depictr_options(palette = depictr_palette(order = "contrast"))
depictr_palette(3)
do.call(depictr_options, old)
```

---

depictr_palette *The depictr colour palettes*

Description

Colourblind-aware palettes shared by every depictr plot. The qualitative palette is based on the Okabe-Ito set (Okabe & Ito, 2008), a categorical palette that stays distinguishable under the common forms of colour-vision deficiency. It keeps six of that set's eight colours. The depictr brand blue (#005b96) takes the place of its blue (#0072b2) and leads the palette, and a grey (#999999) takes the place of its black. The sequential and diverging palettes are perceptually ordered single-hue and red-blue ramps.

Usage

```
depictr_palette(
  n = NULL,
  type = c("qualitative", "sequential", "diverging"),
  order = c("default", "contrast")
)
```

Arguments

n	Number of colours to return. If NULL (the default), the qualitative palette returns all of its base colours, the eight built-in colours unless options(depictr.palette =) supplies another set, and the sequential and diverging ramps return seven. For the qualitative palette, an n larger than the available base colours is interpolated. The sequential and diverging palettes are ramps and accept any n. Interpolating the built-in qualitative palette beyond its eight base colours loses the colour-vision-deficiency guarantee and warns, so facet the groups or use the sequential ramp when many groups are needed.
type	Palette type: "qualitative" (categorical groups), "sequential" (ordered low-to-high) or "diverging" (a midpoint with two directions).
order	The order of the built-in qualitative palette: "default", or "contrast" for the contrast-first order described below. The ramps have an order of their own and refuse "contrast".

Details

The guarantee covers the eight built-in colours only. Past eight, the qualitative palette interpolates between them, and the interpolated colours are close enough to fail the package's own colour-distance check, so the palette warns.

The qualitative palette can be overridden globally with options(depictr.palette =) (see depictr_options()). When set, that custom palette replaces the eight built-in colours and is interpolated when more colours are requested than it provides. The sequential and diverging ramps are unaffected.

Value

A character vector of hex colour codes.

The contrast-first order

In the default order the second colour is the orange (#e69f00), at 2.25:1 against white. Any figure on white whose colour tells two or more groups apart therefore fails the geometry_contrast row of check_figure(). No more than three of the eight colours pass every colour row of that audit together.

order = "contrast" puts those three first: the brand blue, the vermillion (#d55e00) and the reddish purple (#cc79a7), then the bluish green (#009e73) and the other four in their default order. Up to three groups then pass every colour row, contrast against white included. They stay at least 14.01 apart in CIEDE2000 under every vision type, 6.86 apart in lightness and 3.06:1 against white. Four groups keep 3:1 against white, but the bluish green and the reddish purple are 3.31 apart in lightness,

which fails `greyscale_separability`. Asking for more than four colours warns, since the fifth, the orange, falls below 3:1.

Under this order, the second group is drawn in the vermillion. That is also the accent (`depictr_options()`) that depictr uses for single highlights, such as the flagged points of `outlier_plot()` and the missing cells of `missingness_map()`.

The order applies to the built-in palette alone. A palette set with `depictr_options(palette = )` is the user's own, so it is returned in the order it was set, with a warning. That is also how to make the order the default for every later plot: `depictr_options(palette = depictr_palette(order = "contrast"))`. The palette is then the user's own, so asking for more than four colours no longer warns. `scale_colour_depictr()` and `scale_fill_depictr()` take `order` for a single plot.

References

Okabe M, Ito K (2008). "Color Universal Design (CUD): How to make figures and presentations that are friendly to colorblind people." https://jfly.uni-koeln.de/color/. Accessed 2026-06-14.

World Wide Web Consortium (2024). "Web Content Accessibility Guidelines (WCAG) 2.2." W3C Recommendation, https://www.w3.org/TR/WCAG22/. Accessed 2026-08-18.

Examples

```
depictr_palette(3)
depictr_palette(3, order = "contrast")
depictr_palette(7, type = "sequential")
scales::show_col(depictr_palette())
```

---

dumbbell_plot *Dumbbell plot*

Description

Compares a numeric value between exactly two groups across a set of categories: for each category, the two group values are drawn as points joined by a segment, so the size and direction of the gap can be read at a glance. It is a clearer alternative to paired or grouped bars for comparisons of before and after, two conditions or two periods.

Usage

```
dumbbell_plot(
  data,
  category,
  value,
  group,
  sort = c("gap", "value", "none"),
  point_size = 3,
  palette = NULL,
  legend_inside = FALSE,
```

```
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`data`	A data frame.
`category`	The categorical axis (string or unquoted name), with one row of the plot per level.
`value`	The numeric value to compare (string or unquoted name).
`group`	The two-level grouping variable whose levels are the two ends of each dumbbell (string or unquoted name).
`sort`	How to order the categories up the axis: `"gap"` (by the signed difference between the two groups, the default), `"value"` (by the second group's value) or `"none"` (the data's own order: the order of the levels when `category` is a factor, and otherwise the order in which the categories first appear).
`point_size`	Size of the end points.
`palette`	Length-2 colours for the two groups. Defaults to `depictr_palette()`.
`legend_inside`	When `TRUE`, draw the two-group legend inside the panel, in the top-right corner, over a translucent background, instead of in a right-hand margin. Check that it covers no dumbbell: the default gap sort puts the category with the largest signed difference (the second level of `group` minus the first) at the top. Defaults to `FALSE`.
`title`, `x_lab`, `y_lab`	Plot title and axis labels.

Details

When a category has several rows per group, the values are averaged. Rows with a missing category, value or group are dropped.

Value

A ggplot2::ggplot object.

Examples

```
wb <- wellbeing_survey
wb$age_group <- ifelse(wb$age < median(wb$age), "younger", "older")
dumbbell_plot(wb, region, life_satisfaction, age_group)
```

`ecdf_plot` *Empirical cumulative distribution function (ECDF) plot*

Description

Draws the empirical CDF of a numeric variable (the proportion of observations at or below each value), optionally split by a grouping variable. Unlike a histogram, it needs no choice of bin width, and quantiles, medians and group shifts can be read directly off the curve.

Usage

```
ecdf_plot(
  data,
  x,
  group = NULL,
  reference_quantiles = NULL,
  palette = NULL,
  legend_inside = FALSE,
  title = NULL,
  x_lab = NULL,
  y_lab = "Cumulative proportion"
)
```

Arguments

`data`	A data frame.
`x`	The numeric variable (string or unquoted name).
`group`	Optional grouping variable (string or unquoted name) mapped to colour, giving one ECDF per group. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
`reference_quantiles`	Optional numeric vector of probabilities in [0, 1] to mark with light horizontal guides, such as `c(0.25, 0.5, 0.75)`. `NULL` (the default) draws none.
`palette`	Colours for the groups. Defaults to `depictr_palette()`.
`legend_inside`	When `TRUE` and a `group` is given, the legend is drawn inside the panel, over a translucent background, in the bottom-right corner, which the ECDF leaves empty once it nears 1. Otherwise, it is drawn in a right-hand margin. Defaults to `FALSE`.
`title, x_lab, y_lab`	Plot title and axis labels.

Value

A ggplot2::ggplot object.

Examples

```
ecdf_plot(lexical_decision, RT)
ecdf_plot(lexical_decision, RT, group = condition,
          reference_quantiles = c(0.25, 0.5, 0.75))
```

`effects_plot` *Plot predicted values for one predictor*

Description

Shows the values a model predicts as one focal predictor varies, holding the other predictors at typical values (the mean for numeric predictors and the most frequent level for factors). A confidence band for a numeric predictor, or confidence intervals for a factor, show the uncertainty.

Usage

```
effects_plot(
  model,
  predictor,
  conf_level = 0.95,
  n = 100,
  rug = TRUE,
  colour = depictr_brand(),
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`model`	A fitted model, such as an `lm`, `glm` or `merMod`.
`predictor`	Name of the focal predictor (string): a variable in the model's formula, or a model-frame name such as `"log(rainfall)"`, which stands for the one variable it transforms. A term built from several variables, such as `"I(rainfall * fertiliser)"`, is refused.
`conf_level`	Confidence level for the interval, a single number strictly between 0 and 1.
`n`	Number of points across the range of a numeric predictor.
`rug`	Whether to add a rug of the observed values of a numeric predictor.
`colour`	Colour for the line, points and band. Defaults to the depictr brand blue.
`title, x_lab, y_lab`	Title and axis labels.

Details

The prediction grid holds the variables the formula names, on their own scale, and `stats::predict()` re-applies each transformation to it. A model with terms such as `log(rainfall)`, `poly(fertiliser, 2)`, `splines::ns()`, `scale()`, `factor()` or `I()` is therefore predicted as fitted, and the x-axis shows the raw variable. Typical values and the range of the focal predictor are taken over the rows the model was fitted to. A numeric variable that enters only through `factor()` is held at its most frequent value. A variable that enters only through a transformation is recovered from the data named in the model's call, so such a model fitted to workspace vectors without `data =` is refused. A term computed from statistics of its rows, such as `I(x - mean(x))`, is refused too, since `predict()` would compute it afresh over the grid. Centre such a variable in the data, or use `scale()`, which `predict()` re-applies as fitted.

An offset, whether an `offset()` term or the `offset` argument, is held at its mean over the rows the model was fitted to, on the link scale. For a Poisson rate model with `offset(log(exposure))`, the line is the expected count at the geometric-mean exposure. The offset stays at its mean when the focal predictor enters it too, as in `log(exposure) + offset(log(exposure))`, so the line then shows the predictor's effect beyond the offset. The y-axis label ends in 'at the mean offset'.

Predictions and standard errors come from `stats::predict()`. `glm` predictions are formed on the link scale and back-transformed, so a binomial model shows predicted probabilities. The band uses the multiplier of the model's own tests, as `tidy_estimates()` does. An `lm`, and a `glm` whose dispersion is estimated, get t on the residual degrees of freedom, so a Gaussian `glm` draws the band of the same `lm`. Binomial and Poisson fits and mixed models get z. Mixed models fitted with `lme4::lmer()` or `lme4::glmer()` are supported too. Their predictions use only the fixed effects, as `re.form = NA` does, and their standard errors come from the fixed-effect design matrix and `vcov()`. Works with `lm`, `glm` and `merMod`. Other model classes are attempted without a guarantee.

Value

A ggplot2::ggplot object.

Examples

```
fit <- lm(yield ~ rainfall + fertiliser + treatment, data = crop_yield)
effects_plot(fit, "fertiliser")
effects_plot(fit, "treatment")

gfit <- glm(accuracy ~ word_frequency + condition,
            data = lexical_decision, family = binomial)
effects_plot(gfit, "word_frequency")        # predicted probability
```

`estimation_plot` *Cumming estimation plot*

Description

An estimation plot puts the effect size and its uncertainty, not a p-value, at the centre of the comparison. The upper panel shows each group's raw data (jittered) with its mean and confidence interval.

The lower panel shows the difference between each other group's mean and a reference group's mean, with a bootstrap confidence interval. Each difference is drawn in its group's column, beneath that group's data. The two panels are stacked with 'patchwork', and each has its own outcome axis.

Usage

```
estimation_plot(
  data,
  y,
  group,
  reference = NULL,
  conf_level = 0.95,
  n_boot = 5000,
  effsize = c("hedges_g", "cohens_d", "none"),
  show_points = TRUE,
  point_alpha = 0.25,
  palette = NULL,
  title = NULL,
  y_lab = NULL,
  heights = c(2, 1.4)
)
```

Arguments

`data`	A data frame.
`y`	The numeric outcome (string or unquoted name).
`group`	The grouping variable (string or unquoted name).
`reference`	The reference (control) group that the others are compared with. Defaults to the first level of `group`. The reference is drawn first in the upper panel and has an empty column in the lower panel, where the dashed line marks a difference of zero.
`conf_level`	Confidence level for both the group intervals (t-based) and the bootstrap difference intervals, a single number strictly between 0 and 1.
`n_boot`	Number of bootstrap resamples for the difference intervals.
`effsize`	Standardised effect size annotated beside each difference: `"hedges_g"` (the default, small-sample corrected), `"cohens_d"` or `"none"` to omit it. Every contrast against the reference is labelled, so the standardised effect is shown for two groups and for more.
`show_points`	Whether to draw the raw data behind the group means.
`point_alpha`	Transparency of the raw points.
`palette`	Colours for the groups. Defaults to `depictr_palette()`.
`title`, `y_lab`	Title and outcome-axis label.
`heights`	Relative heights of the upper (raw data) and lower (difference) panels, passed to `patchwork::plot_layout()`.

Details

This stacked layout is the Cumming estimation plot (Cumming, 2012, and Ho et al., 2019), with two groups as with more. Each difference is annotated with a standardised effect size: Cohen's d or, by default, the small-sample corrected Hedges' g (Hedges, 1981). The Gardner-Altman plot (Gardner & Altman, 1986) is a different layout, which sets a single difference beside the two groups on a floating axis whose zero is level with the reference group's mean.

The lower-panel interval is a non-parametric bootstrap of the mean difference: the two groups are resampled with replacement `n_boot` times, and the requested percentile interval is read off the resampled differences. This makes no normality assumption about the sampling distribution of the difference. The bootstrap uses base R only. Call `set.seed()` first for reproducible intervals.

A group with fewer than two observations has no estimable spread, so its confidence interval is omitted (the mean is still drawn), and a difference involving it is shown as a point without a bootstrap interval. A warning is issued in both cases.

Value

A 'patchwork' object (printable like a ggplot2::ggplot). The `"differences"` attribute holds a data frame with one row per group other than the reference and the columns `group`, `reference`, `diff` (the group's mean minus the reference's), `lower` and `upper` (the bootstrap interval at `conf_level`), `cohens_d` and `hedges_g`. The interval limits and the effect sizes are `NA` where either group has fewer than two observations, and the effect sizes are also `NA` when the pooled standard deviation is zero.

References

Cumming G (2012). *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis*. Routledge, New York, NY. ISBN 978-0-415-87968-2.

Gardner MJ, Altman DG (1986). "Confidence intervals rather than P values: Estimation rather than hypothesis testing." *BMJ*, **292**(6522), 746–750. doi:10.1136/bmj.292.6522.746.

Hedges LV (1981). "Distribution theory for Glass's estimator of effect size and related estimators." *Journal of Educational Statistics*, **6**(2), 107–128. doi:10.3102/10769986006002107.

Ho J, Tumkaya T, Aryal S, Choi H, Claridge-Chang A (2019). "Moving beyond P values: Data analysis with estimation graphics." *Nature Methods*, **16**(7), 565–566. doi:10.1038/s415920190470-3.

Examples

```
set.seed(1)
# n_boot is kept small here for speed. Use the default for real work.
estimation_plot(lexical_decision, RT, condition, n_boot = 1000)

estimation_plot(crop_yield, yield, treatment)
# More than two groups: differences from a chosen reference
estimation_plot(wellbeing_survey, life_satisfaction, region,
                reference = "North")
```

`explore_bivariate` *Plot any pair of variables*

Description

Chooses an appropriate plot for the relationship between two variables according to their types. Two numeric variables are shown as a scatter plot with a fitted trend. A numeric and a categorical variable are shown as box plots of the numeric variable by level. Two categorical variables are shown as a filled bar chart of proportions.

Usage

```
explore_bivariate(
  data,
  x,
  y,
  method = "lm",
  palette = NULL,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`data`	A data frame.
`x`, `y`	The two variables (string or unquoted name).
`method`	Smoothing method for two numeric variables, passed to `ggplot2::geom_smooth()`, or `NULL` for no trend.
`palette`	Colours used when a fill or colour is needed. Defaults to `depictr_palette()`.
`title`, `x_lab`, `y_lab`	Title and axis labels, which default to the variable names.

Details

Dates, date-times and durations (`Date`, `POSIXct` and `difftime` columns) count as numeric. Against a numeric variable, they give a scatter plot with a trend, drawn by `scatter_trend()`, and against a categorical one box plots of their values by level. A duration is drawn as a number in its own units, named in the default axis title, as in `stay (days)`.

Value

A ggplot2::ggplot object.

Examples

```
explore_bivariate(crop_yield, fertiliser, yield)        # numeric ~ numeric
explore_bivariate(lexical_decision, condition, RT)      # categorical ~ numeric
explore_bivariate(wellbeing_survey, region, education)  # categorical ~ categorical
```

`explore_categorical` *Bar chart of a categorical variable*

Description

Counts or proportions of the levels of a categorical variable, optionally split by a grouping variable.

Usage

```
explore_categorical(
  data,
  x,
  group = NULL,
  proportion = FALSE,
  position = c("dodge", "stack", "fill"),
  sort = TRUE,
  horizontal = FALSE,
  palette = NULL,
  title = NULL,
  x_lab = NULL
)
```

Arguments

`data`	A data frame.
`x`	The categorical variable (string or unquoted name). Numeric columns are accepted only when they have at most 20 distinct values.
`group`	Optional grouping variable mapped to fill, with the groups in the order of its levels. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
`proportion`	Whether to show proportions instead of counts. When `group` is set, proportions are computed within each group.
`position`	Bar position when `group` is set: `"dodge"`, `"stack"` or `"fill"`.
`sort`	Whether to order the bars from most to least frequent.
`horizontal`	Whether to draw horizontal bars, which helps when there are many levels or long labels.
`palette`	Colours for the groups. Defaults to `depictr_palette()`.
`title`, `x_lab`	Plot title and category-axis label, which defaults to the variable name.

Details

A continuous numeric column (more than 20 distinct values) is rejected with an error, since coercing it to a factor would draw one bar per value. Use `explore_distribution()` for such variables.

Value

A ggplot2::ggplot object.

Examples

```
explore_categorical(wellbeing_survey, region)
explore_categorical(wellbeing_survey, education, group = region,
                    proportion = TRUE, position = "dodge")
```

---

explore_distribution *Plot the distribution of a variable*

Description

Draws a histogram, a density curve or both, optionally split by a grouping variable, for a first look at a continuous variable.

Usage

```
explore_distribution(
  data,
  x,
  group = NULL,
  type = c("histogram", "density", "both"),
  bins = 30,
  alpha = 0.6,
  position = NULL,
  palette = NULL,
  facet = FALSE,
  legend_inside = FALSE,
  title = NULL,
  x_lab = NULL
)
```

Arguments

data	A data frame.
x	The continuous variable to display. Either a string or an unquoted column name.
group	Optional grouping variable (string or unquoted name) mapped to colour and fill. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.

type	One of "histogram", "density" or "both".
bins	Number of histogram bins.
alpha	Fill transparency, which helps when groups overlap.
position	Histogram position adjustment, such as "identity", "stack" or "dodge". The default (NULL) chooses "dodge" when group is set, so that overlapping bars stay readable, and "identity" otherwise.
palette	Colours for the groups. Defaults to depictr_palette().
facet	When a group is given, whether to draw one panel per group in place of an overlay. Panels are much clearer once there are more than two or three groups, since overlaid histograms in particular become hard to read. Defaults to FALSE. Ignored when there is no group.
legend_inside	When TRUE and a group is given without facet, the colour legend is drawn inside the panel, over a translucent background, in the top-right corner, which a unimodal histogram or density leaves empty. Otherwise, it is drawn in a right-hand margin. Defaults to FALSE.
title, x_lab	Plot title and x-axis label, which defaults to the variable name.

Value

A ggplot2::ggplot object.

Examples

```
explore_distribution(lexical_decision, RT)
explore_distribution(lexical_decision, RT, group = condition, type = "density")
# One panel per group keeps many groups legible:
explore_distribution(wellbeing_survey, life_satisfaction, group = region,
                     type = "both", facet = TRUE)
```

---

explore_pairs *Scatter-plot matrix*

Description

A matrix of pairwise plots for a set of numeric variables: scatter plots below the diagonal, densities on the diagonal and correlation coefficients above the diagonal, optionally coloured by a grouping variable. Built with 'patchwork', so it shares the package theme and palette.

Usage

```
explore_pairs(
  data,
  cols = NULL,
  group = NULL,
  max_cols = 8,
  point_alpha = 0.5,
```

```
  method = c("pearson", "spearman", "kendall"),
  palette = NULL,
  title = NULL
)
```

Arguments

data	A data frame.
cols	Numeric columns to include, which cannot include group. If NULL, every numeric column other than group is used, so a frame with more than max_cols of them needs cols to choose among them.
group	Optional grouping variable mapped to colour, with each level in the same colour in every panel. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
max_cols	The largest number of variables accepted, since a k-by-k matrix grows quickly. More columns than this is an error, so the matrix never leaves a variable out without saying so.
point_alpha	Point transparency in the scatter panels.
method	Correlation method for the upper-triangle coefficients, passed to stats::cor(): "pearson", "spearman" or "kendall". A pair whose correlation is undefined (a constant column, or too few complete cases) is labelled n/a, as in correlation_heatmap().
palette	Colours for the groups. Defaults to depictr_palette().
title	Overall title for the matrix.

Value

A 'patchwork' object (printable like a ggplot2::ggplot).

Examples

```
explore_pairs(crop_yield, cols = c("rainfall", "fertiliser", "yield"))

explore_pairs(crop_yield, cols = c("rainfall", "fertiliser", "yield"),
              group = treatment, method = "spearman")
```

---

format_terms *Tidy raw coefficient names for display*

Description

Tidies the term names produced by modelling functions so that they read well on a plot: the intercept is renamed, an optional Bayesian prefix `b_` or `bs_` is stripped, interaction colons are converted to a chosen symbol and underscores are shown as spaces, so `word_frequency` becomes `"word frequency"`. The prefix is stripped from every component of an interaction, such as `b_x:b_y`, and not only from the leading term. Missing values (`NA`) stay `NA` and are not rendered as the text `"NA"`.

Usage

```
format_terms(
  x,
  interaction = c("times", "asterisk", "colon", "space"),
  strip_prefix = TRUE,
  tidy_intercept = TRUE,
  wrap = NULL
)
```

Arguments

`x`	Character vector of term names.
`interaction`	How to render interaction colons: `"times"` (the default, a Unicode multiplication sign), `"asterisk"`, `"colon"` (unchanged) or `"space"`.
`strip_prefix`	Whether to remove a leading `b_` or `bs_` (as added by 'brms') from each interaction component.
`tidy_intercept`	Whether to replace `(Intercept)` with `"Intercept"`.
`wrap`	Optional integer width at which to wrap long labels onto new lines (see `base::strwrap()`). `NULL`, the default, leaves labels unwrapped.

Value

A character vector the same length as `x`, with `NA` preserved.

Examples

```
format_terms(c("(Intercept)", "b_conditionB", "freq:condition"))
format_terms("region:education:age", interaction = "asterisk")
format_terms(c("word_frequency", "b_sleep_hours"))
format_terms(c("b_x:b_y", NA))
```

`frequentist_bayesian_plot` *Plot frequentist and Bayesian estimates together*

Description

Presents the estimates from a frequentist model and a Bayesian model on one plot, in the first two colours of the colourblind-safe depictr_palette(): brand blue for the frequentist estimates and orange for the Bayesian ones, whether the Bayesian side is drawn from draws or from summaries.

Usage

```
frequentist_bayesian_plot(
  frequentist,
  bayesian,
  conf_level = 0.95,
  labels = NULL,
  interaction = c("times", "asterisk", "colon", "space"),
  intercept = TRUE,
  facet = TRUE,
  scales = c("free", "fixed"),
  note_frequentist_no_prior = FALSE,
  vertical_line_at_x = 0,
  title = NULL,
  subtitle = NULL,
  x_lab = "Estimate",
  caption = NULL,
  ...
)
```

Arguments

frequentist	A frequentist model, such as one from lm, glm or lmerTest::lmer, or a tidy data frame of estimates.
bayesian	A Bayesian model, a posterior draws object, a draws matrix or data frame, a tidy data frame of posterior summaries or a summary matrix such as brms::fixef() returns.
conf_level	Level of the intervals the function computes: the confidence intervals of a frequentist model and, when bayesian holds draws, the outer credible interval of each posterior (see Details). A single number strictly between 0 and 1.
labels, interaction, intercept	See compare_models(). On both paths, an unnamed labels vector takes the frequentist terms first, then any that only the Bayesian input holds. intercept defaults to TRUE here.
facet, scales	Layout controls. Because a Bayesian model almost always has a large intercept beside small slopes, the comparison defaults to a faceted, free-scaled layout (facet = TRUE): each term gets its own panel and free x-axis, so every posterior and its frequentist overlay stay legible. Pass facet = FALSE (or scales = "fixed") for a single shared axis.
note_frequentist_no_prior	If TRUE, append "(no prior)" to the frequentist legend label, which helps when the title names the Bayesian prior.

vertical_line_at_x
: Position of the vertical reference line (NA to omit).

title, subtitle, x_lab
: Title, subtitle and x-axis label.

caption
: Plot caption. The default (NULL) names the level of each source's intervals. Pass NA to omit the caption.

...
: Further arguments passed to compare_models() on the summary path (ignored on the distribution path).

Details

When bayesian holds posterior draws (a brms or rstanarm fit, a posterior draws object, a draws matrix or a long or wide draws data frame), the full posterior distribution of each term is drawn as a 'ggdist' half-eye, with the frequentist point and confidence interval overlaid at the same position. When bayesian is only a table of posterior summaries, the function draws the familiar two-source forest plot through compare_models(). It reads tidy columns (term, estimate, conf.low, conf.high). It also reads the Estimate, Q2.5 and Q97.5 of the matrix brms::fixef() returns and the l-95% CI and u-95% CI of summary(fit)$fixed.

A data frame counts as draws when it holds a sampler index column (.chain, .iteration or .draw), when it is long, with a parameter column (parameter, term, .variable or variable) and a value column (value, .value or draw) but no summary column such as estimate or conf.low, or when it is wide, with at least 50 rows and no summary column. Any other data frame is read as summaries. The half-eyes need the 'ggdist' package. Without it, each posterior is drawn as its median and conf_level interval, with a message.

Terms are aligned by their display label, so the b_ prefix of brms is matched automatically against the frequentist term names. Draws are read as in posterior_plot(), so a draws object holding b_ parameters keeps only those, and two terms that would share a label are an error. With draws, only the terms both sides share are drawn, and a message names any dropped, such as a sigma in the posterior. The forest plot of summaries draws every term of either input.

conf_level is the level of every interval the function computes. For a frequentist model, that is its confidence interval. For posterior draws, it is the outer of the equal-tailed credible intervals drawn beneath each half-eye, so the two sources are compared at one level. The inner interval is 66%, drawn only when conf_level exceeds 0.66 so that it stays the narrower of the two. A table of estimates keeps the interval limits it holds, at whatever level they were computed. By default, a caption names the level of each source's intervals or says that they are as supplied.

Value

A ggplot2::ggplot object.

Examples

```
# Summary path: a tidy 'Bayesian' summary as a data frame. The summary here
# stands in for a regularised posterior, so a weakly informative prior pulls
# each estimate towards zero and narrows its interval. A real posterior would
# come from the model fit, not from arithmetic on the frequentist one.
freq <- lm(life_satisfaction ~ stress + sleep_hours + exercise_days,
           data = wellbeing_survey)
```

```
bayes <- tidy_estimates(freq)
bayes$estimate <- bayes$estimate * 0.9
half <- (bayes$conf.high - bayes$conf.low) / 2 * 0.8
bayes$conf.low <- bayes$estimate - half
bayes$conf.high <- bayes$estimate + half
frequentist_bayesian_plot(freq, bayes,
                          title = "Frequentist vs. Bayesian estimates")

# Distribution path: simulated posterior draws (one column per term) drawn as
# full posteriors with the frequentist point and interval overlaid.
set.seed(1)
co <- coef(freq)
draws <- as.data.frame(lapply(co, function(m) rnorm(400, m, abs(m) * 0.1 + 0.05)))
names(draws) <- names(co)
frequentist_bayesian_plot(freq, draws,
                          title = "Posterior with frequentist overlay")
```

`gain_plot` *Cumulative gains chart*

Description

Shows how many of the positive cases are captured as a growing share of the population is targeted in order of predicted score. It is the customary chart for judging a classifier's value for ranking and targeting, as in marketing, triage and fraud detection. The diagonal marks the no-model baseline, and a second reference line marks a perfect model. Pass a named list of models or of (actual, score) pairs to overlay several colour-coded gains curves with a legend. The perfect-model line is then omitted, since it depends on the prevalence of the outcome and the overlaid models need not share one.

Usage

```
gain_plot(
  x,
  score = NULL,
  colour = depictr_brand(),
  legend_inside = FALSE,
  title = NULL
)
```

Arguments

`x`	A binomial `glm`, the vector of observed outcomes (0 and 1, logical or a two-level factor with the positive class second) or a named list of models or of (actual, score) pairs to overlay.
`score`	When `x` is an outcome vector, the matching scores or predicted probabilities, or a list of them for several models.

colour	Curve colour for a single model. Defaults to the depictr brand blue. Ignored when several models are overlaid.
legend_inside	When TRUE and several models are overlaid, the legend is drawn inside the panel, over a translucent background, in the bottom-right triangle, which the concave curve leaves empty. Otherwise, it is drawn in a right-hand margin. Defaults to FALSE.
title	Plot title.

Value

A ggplot2::ggplot object.

Examples

```
# Targeting only pays off when the positive class is scarce, so the example
# uses the rare clinical-trial adverse event (about 10% positive).
gfit <- glm(adverse_event ~ biomarker + age + arm,
            data = clinical_trial, family = binomial)
gain_plot(gfit)

# Compare two models.
reduced <- glm(adverse_event ~ biomarker, data = clinical_trial,
               family = binomial)
gain_plot(list(Full = gfit, Reduced = reduced))
```

---

group_comparison_plot *Compare group means with confidence intervals*

Description

An estimation-style plot of a numeric outcome across the levels of a grouping variable: each group's mean with a confidence interval, drawn over the raw (jittered) data. Because it shows both the estimate and its uncertainty, it shows whether the groups differ more faithfully than a bar chart does.

Usage

```
group_comparison_plot(
  data,
  y,
  group,
  conf_level = 0.95,
  show_points = TRUE,
  point_alpha = 0.25,
  differences = FALSE,
  reference = NULL,
  n_boot = 5000,
  palette = NULL,
```

```
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

data	A data frame.
y	The numeric outcome (string or unquoted name).
group	The grouping variable (string or unquoted name). A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
conf_level	Confidence level for the intervals (t-based), a single number strictly between 0 and 1.
show_points	Whether to draw the raw data behind the means.
point_alpha	Transparency of the raw points.
differences	If TRUE, append a lower panel showing the pairwise mean differences against a reference group, each with a bootstrap confidence interval, which turns the plot into a full estimation plot through estimation_plot(). The return value is then a 'patchwork' object. Defaults to FALSE, the plain group-means plot.
reference	Reference group for the difference panel when differences = TRUE. Defaults to the first level of group. Ignored otherwise.
n_boot	Number of bootstrap resamples for the difference intervals when differences = TRUE. Ignored otherwise.
palette	Colours for the groups. Defaults to depictr_palette().
title, x_lab, y_lab	Title and axis labels. x_lab is not drawn when differences = TRUE, since the estimation plot names its groups on the axis alone.

Details

A group with a single observation has no degrees of freedom for a t-based interval, so only its mean is drawn (no interval) and a warning is issued.

Value

A ggplot2::ggplot object, or a 'patchwork' object when differences = TRUE.

Examples

```
group_comparison_plot(lexical_decision, RT, condition)
group_comparison_plot(crop_yield, yield, treatment)
# Append the pairwise mean-difference panel (an estimation plot):
set.seed(1)
group_comparison_plot(crop_yield, yield, treatment, differences = TRUE)
```

influence_plot *Influence plot*

Description

A bubble plot of leverage against the studentised residuals, with bubble area proportional to Cook's distance. It summarises in a single picture which observations most influence a fitted model. Reference lines mark large residuals and high-leverage points, and the most influential observations are labelled.

Usage

```
influence_plot(model, n_label = 3, colour = depictr_brand(), title = NULL)
```

Arguments

`model`	A fitted `lm` or `glm` model.
`n_label`	Number of most-influential points (by Cook's distance) to label.
`colour`	Bubble colour. Defaults to the depictr brand blue.
`title`	Plot title.

Value

A ggplot2::ggplot object.

References

Cook RD (1977). "Detection of influential observation in linear regression." *Technometrics*, **19**(1), 15–18. doi:10.1080/00401706.1977.10489493.

Examples

```
fit <- lm(yield ~ rainfall + fertiliser + soil_ph, data = crop_yield)
influence_plot(fit)
```

interaction_plot *Plot a two-way interaction of predicted values*

Description

Shows how the predicted relationship between a focal predictor and the response changes across the levels, or representative values, of a second, moderating predictor. Other predictors are held at typical values.

Usage

```
interaction_plot(
  model,
  predictor,
  moderator,
  moderator_values = NULL,
  conf_level = 0.95,
  n = 80,
  band = TRUE,
  palette = NULL,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`model`	A fitted model, such as an `lm`, `glm` or `merMod`.
`predictor`	Name of the focal predictor on the x-axis, as a string, given as in `effects_plot()`: `"log(rainfall)"` means `rainfall`.
`moderator`	Name of the moderating predictor, mapped to colour, as a string given in the same way as `predictor`. It must be a different variable.
`moderator_values`	For a numeric moderator, the values to show, on the variable's own scale. Defaults to the 10th, 50th and 90th percentiles in the estimation sample.
`conf_level`	Confidence level for the bands and intervals, a single number strictly between 0 and 1.
`n`	Number of points across the range of a numeric focal predictor.
`band`	Whether to draw confidence bands for a numeric focal predictor.
`palette`	Colours for the moderator. Defaults to `depictr_palette()`.
`title`, `x_lab`, `y_lab`	Title and axis labels.

Details

The grid is built as in `effects_plot()`. It holds the raw variables, so transformed terms are predicted as fitted. Typical values come from the estimation sample, and an offset is held at its mean on the link scale.

Value

A ggplot2::ggplot object.

Examples

```
fit <- lm(yield ~ fertiliser * treatment + rainfall, data = crop_yield)
interaction_plot(fit, "fertiliser", "treatment")
```

k_diagnostic *Suggest a number of clusters*

Description

Computes a cluster-quality diagnostic across a range of k and draws the diagnostic curve, with the suggested k highlighted. Three criteria are available: the average silhouette width (maximised), the elbow of the total within-cluster sum of squares (the candidate farthest from the straight line joining the first and last points of the curve) and the gap statistic (the smallest k whose gap is within one standard error of the next, by the heuristic of Tibshirani et al., 2001).

Usage

```
k_diagnostic(
  data,
  k_range = NULL,
  method = c("silhouette", "wss", "gap"),
  cols = NULL,
  scale = TRUE,
  nstart = 10,
  B = 50,
  title = NULL
)
```

Arguments

data	A data frame or numeric matrix.
k_range	Integer vector of candidate cluster counts to evaluate. The default, NULL, searches 1:8 for "gap" and "wss" and 2:8 for "silhouette". Candidates the chosen criterion cannot evaluate are skipped with a message naming them: k = 1 under "silhouette" (a lone cluster has no neighbouring cluster to compare against, so the width is undefined) and any k at or above the number of observations under every criterion.
method	Diagnostic: "silhouette", "wss" (within sum of squares elbow) or "gap".
cols	When data is a data frame, the numeric columns to use.
scale	Whether to scale variables to unit variance first. Columns with zero or near-zero variance cannot be scaled and are dropped with a message, as in correlation_heatmap().
nstart	Number of random starts for stats::kmeans().
B	Number of reference bootstrap samples for the gap statistic.
title	Plot title.

Details

Only the gap statistic can conclude that the data hold no clusters, by suggesting `k = 1`, and only when the search starts at 1, as it does by default. Both ends of the curve lie on the line the elbow is measured from, so among three or more candidates the elbow is never the last, and it is the first only when the curve is a straight line. A range therefore has to extend one step beyond any `k` the elbow should be able to choose.

The k-means fits take random starts, and the gap statistic draws `B` uniform reference data sets, all from the session's random-number stream, so call `set.seed()` first for a reproducible suggestion.

Value

A ggplot2::ggplot object showing the diagnostic value against `k`, with the suggested `k` marked and named in the subtitle. The underlying computation is attached as attributes: `attr(p, "k_table")` (a data frame with one row per `k`), `attr(p, "suggested")` (the chosen `k`) and `attr(p, "method")`.

References

Rousseeuw PJ (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." *Journal of Computational and Applied Mathematics*, **20**, 53–65. doi:10.1016/0377-0427(87)901257.

Tibshirani R, Walther G, Hastie T (2001). "Estimating the number of clusters in a data set via the gap statistic." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**(2), 411–423. doi:10.1111/14679868.00293.

Examples

```
p <- k_diagnostic(crop_yield, k_range = 2:6,
                  cols = c("rainfall", "fertiliser", "soil_ph", "yield"))
attr(p, "suggested")
```

`lexical_decision` *Simulated lexical-decision experiment*

Description

A reproducibly simulated psycholinguistic lexical-decision experiment, in which participants judge whether letter strings are real words. The design is fully crossed and counterbalanced: 24 participants each respond to all 40 items once (960 trials). `condition` (related or unrelated priming) and `modality` (visual or auditory) are both within-participant and within-item, assigned by a Latin-square list rotation so that, across participants, every item appears equally often in each condition and in each modality. Neither factor is therefore collinear with the item random intercept, which suits crossed-random-effects models such as `lmer(RT ~ condition + modality + word_frequency + (1 | participant) + (1 | item))`, and hence `optimizer_fixef_plot()`, `coefficient_plot()` and the distribution plots.

Usage

```
lexical_decision
```

Format

A data frame with 960 rows and 7 variables:

participant Participant identifier (factor, 24 levels).

item Item identifier (factor, 40 levels).

condition Priming condition (factor): `related` or `unrelated`.

modality Presentation modality (factor): `visual` or `auditory`.

word_frequency Item-level word frequency on the Zipf scale.

RT Reaction time, in milliseconds.

accuracy Response accuracy: 1 correct, 0 incorrect.

Details

`RT` is generated log-normally from participant and item random intercepts plus the fixed effects (unrelated priming about 35 ms slower, auditory presentation a little slower and more frequent words faster), with a 250 ms floor. RT analyses should use correct trials (`accuracy == 1`). `accuracy` is generated from the exogenous predictors (`word_frequency`, `condition`, `modality`) and random effects only, and not from `RT`, so classification examples built on it are not circular.

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed, and they describe no real individuals.

Source

Simulated by `data-raw/generate_datasets.R` in the source repository.

`lift_plot` *Cumulative lift chart*

Description

Shows how many times more positive cases a classifier captures, at each depth of the score-ordered population, than random targeting would. A lift of 3 at the top 10% means that decile contains three times the baseline rate of positives. The horizontal line at 1 is the no-model baseline. Pass a named list of models or of (actual, score) pairs to overlay several colour-coded lift curves with a legend.

Usage

```
lift_plot(
  x,
  score = NULL,
  colour = depictr_brand(),
  legend_inside = FALSE,
  title = NULL
)
```

Arguments

x	A binomial glm, the vector of observed outcomes (0 and 1, logical or a two-level factor with the positive class second) or a named list of models or of (actual, score) pairs to overlay.
score	When x is an outcome vector, the matching scores or predicted probabilities, or a list of them for several models.
colour	Curve colour for a single model. Defaults to the depictr brand blue. Ignored when several models are overlaid.
legend_inside	When TRUE and several models are overlaid, the legend is drawn inside the panel, over a translucent background, in the top-right corner, which the decreasing lift curve leaves empty. Otherwise, it is drawn in a right-hand margin. Defaults to FALSE.
title	Plot title.

Value

A ggplot2::ggplot object.

Examples

```
# Lift is measured against the base rate, so the example uses the rare
# clinical-trial adverse event (about 10% positive).
gfit <- glm(adverse_event ~ biomarker + age + arm,
            data = clinical_trial, family = binomial)
lift_plot(gfit)

# Compare two models.
reduced <- glm(adverse_event ~ biomarker, data = clinical_trial,
               family = binomial)
lift_plot(list(Full = gfit, Reduced = reduced))
```

---

missingness_map *Map the missing values in a data frame*

Description

Draws a tile map of the data frame with one column per variable and one row per observation, shading the cells that are missing. The variables are ordered by their proportion of missing values, and that proportion is shown in the axis labels, so variables and patterns that need attention before modelling are easy to spot.

Usage

```
missingness_map(
  data,
  cols = NULL,
  sort = TRUE,
  show_pct = TRUE,
  colours = c("grey85", depictr_accent()),
  legend_inside = FALSE,
  title = NULL
)
```

Arguments

data	A data frame.
cols	Optional character vector of columns to include (by default, all).
sort	Whether to order variables by their proportion of missing values.
show_pct	Whether to append the percentage missing to each variable label.
colours	Length-2 vector: colours for present and missing cells. Defaults to a muted grey ("grey85") for present cells and, for missing cells, the accent colour set with accent in depictr_options(), the colourblind-safe vermillion #d55e00 unless changed.
legend_inside	When TRUE and sort = TRUE, the legend is drawn inside the panel, in the top-right corner, where the most complete columns put a solid block of present cells, so it hides no missing cell. Otherwise, it is drawn in a right-hand margin. Defaults to FALSE.
title	Plot title.

Value

A ggplot2::ggplot object.

Examples

```
missingness_map(wellbeing_survey)
```

---

model_fit_table *Goodness-of-fit statistics across models*

Description

Collects the usual model-comparison statistics into one tidy data frame, one row per model. These comprise the number of observations, the number of estimated parameters (df, as stats::logLik() counts them for AIC and BIC), AIC, BIC, the log-likelihood, an R-squared and the root-mean-square of the residuals (response residuals for a glm, deviance residuals otherwise, which for an unweighted linear or linear mixed model are the ordinary residuals).

Usage

```
model_fit_table(..., digits = 3)
```

Arguments

...	One or more fitted models. Name the arguments to label the rows.
digits	Number of decimal places to round to.

Details

The R-squared depends on the kind of model. An lm reports the ordinary coefficient of determination. Every glm, including quasi-likelihood families and negative binomial fits from MASS::glm.nb(), reports the deviance R-squared, $1 - D/D_0$. Here, D is the residual deviance of the model and D_0 the deviance of the null model, which stats::glm() fits to the same rows with the same prior weights and offset. The null model has an intercept only if the model has one. Both deviances are stored with the fit, so nothing is refitted. The value is the share of the null deviance that the predictors explain. For exponential-family models, it is the Kullback-Leibler R-squared of Cameron and Windmeijer (1997). It equals the ordinary R-squared for a Gaussian model with the identity link and McFadden's (1974) pseudo-R-squared for binary logistic regression. For Poisson, grouped binomial, Gamma and other models, the two measures differ. Mixed models (merMod) report an R-squared of NA, since no single definition is standard for them.

A linear mixed model fitted with lme4::lmer() by restricted maximum likelihood (REML, its default) is refitted by maximum likelihood with lme4::refitML(). Its AIC, BIC and log-likelihood come from the refit, and a message says so. A REML likelihood is the likelihood of the residuals left once the fixed effects are removed. Two models with different fixed effects therefore have REML likelihoods of different data, and their criteria cannot be compared. The anova() method for lme4 fits refits them by maximum likelihood for the same reason. If a refit fails, a warning says so, and that model's df, AIC, BIC and logLik are NA. The other columns describe the model as supplied.

Value

A data frame with one row per model and columns model, n, df, AIC, BIC, logLik, R2 and RMSE.

References

Cameron AC, Windmeijer FAG (1997). "An R-squared measure of goodness of fit for some common nonlinear regression models." *Journal of Econometrics*, **77**(2), 329–342. doi:10.1016/S0304-4076(96)018180.

McFadden D (1974). "Conditional logit analysis of qualitative choice behavior." In Zarembka P (ed.), *Frontiers in econometrics*, 105–142. Academic Press, New York, NY.

Examples

```
m1 <- lm(yield ~ rainfall, data = crop_yield)
m2 <- lm(yield ~ rainfall + fertiliser, data = crop_yield)
m3 <- lm(yield ~ rainfall + fertiliser + soil_ph + treatment,
         data = crop_yield)
model_fit_table(simple = m1, medium = m2, full = m3)
```

`model_report` *A one-figure model report*

Description

Assembles a compact, consistent overview of a fitted model in a single figure: the coefficient estimates, the predicted effect of a focal predictor, a residual panel and a normal Q-Q plot, with a subtitle of key fit statistics. It composes several depictr plots with `arrange_plots()` and suits a quick model review or a report appendix.

Usage

```
model_report(
  model,
  predictor = NULL,
  standardise = TRUE,
  title = NULL,
  subtitle = NULL,
  glm_panels = TRUE,
  seed = NULL
)
```

Arguments

`model`	A fitted `lm` or `glm` model.
`predictor`	Focal predictor for the effect panel, given as in `effects_plot()`. If `NULL`, the first numeric variable in the formula, or failing that the first variable, is used, so a model with `log(rainfall)` shows the effect of `rainfall`.
`standardise`	Whether the coefficient panel scales each continuous predictor's coefficient by that predictor's standard deviation, leaving factor and binary contrasts in the outcome's units, as described for `standardise` in `coefficient_plot()`. Defaults to `TRUE`, which keeps the panel readable in this compact overview when the slopes are on very different scales. Set `FALSE` for raw estimates.
`title`	Overall title.
`subtitle`	Overall subtitle. If `NULL`, a line of fit statistics (number of observations, R-squared and AIC) is used. For a `glm`, the R-squared is the deviance R-squared described in `model_fit_table()`.
`glm_panels`	For a `glm`, whether to draw the binned-residual and quantile-residual panels (the default) instead of the classic `lm` panels. Ignored for an `lm`.
`seed`	Optional integer seed for the randomisation of discrete quantile residuals, for a reproducible Q-Q panel of a binomial or Poisson `glm`. The caller's random number stream is left as it was.

Details

For an `lm`, the residual panel plots the residuals against the fitted values, and the Q-Q panel is `qq_plot()` of the standardised residuals. A `glm` gets the panels `residual_diagnostics_plot()` draws for it (`glm_panels = TRUE`). The residual panel is a `binned_residual_plot()` (Gelman & Hill, 2007). The Q-Q panel plots randomised quantile residuals (Dunn & Smyth, 1996), which are standard normal under a correctly specified model whatever the response family, so the band of `qq_plot()` applies to them. The standardised deviance residuals of a binary or count model are not expected to be normal, so even a correctly specified model can put many of them outside that band (about a third, in a simulated logistic model with 400 observations). The quantile residuals of a binomial or Poisson model are randomised, and even under a correct model a run of them occasionally leaves the pointwise band. A departure worth acting on persists when the panel is drawn again with another `seed`. For a family with no quantile residual here, such as a quasi-likelihood family, the Q-Q panel plots the deviance residuals and its axis says so.

Value

A 'patchwork' object (printable like a ggplot2::ggplot).

References

Gelman A, Hill J (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, Cambridge, UK. ISBN 978-0-521-68689-1. doi:10.1017/CBO9780511790942.

Dunn PK, Smyth GK (1996). "Randomized quantile residuals." *Journal of Computational and Graphical Statistics*, **5**(3), 236–244. doi:10.1080/10618600.1996.10474708.

Examples

```
fit <- lm(yield ~ rainfall + fertiliser + soil_ph + treatment,
          data = crop_yield)
model_report(fit, title = "Crop-yield model")

# A logistic model gets binned and quantile-residual panels, and a seed
# fixes the randomisation of its quantile residuals:
gfit <- glm(accuracy ~ word_frequency + RT + condition,
            data = lexical_decision, family = binomial)
model_report(gfit, seed = 1)
```

---

monthly_sales *Simulated monthly sales time series*

Description

A reproducibly simulated seasonal monthly time series in tidy long form, for the time-series examples. It holds two product lines (`indoor` and `outdoor`), each with six years of monthly observations with a linear trend, a clear twelve-month seasonal cycle (with different seasonal peaks per line) and noise. The two series show the grouped `timeseries_plot()` on the package's own data, and one series extracted as a monthly `ts` (frequency 12) lets `decompose_plot()` and `acf_plot()` show the trend and seasonality.

Usage

```
monthly_sales
```

Format

A data frame with 144 rows and 3 variables:

date First day of the month (`Date`), from January 2018.

series Product line (factor): `indoor` or `outdoor`.

sales Monthly sales (units).

Details

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed.

Source

Simulated by `data-raw/generate_datasets.R` in the source repository.

`optimizer_fixef_plot` *Plot fixed effects across optimisers*

Description

Visualises how the fixed-effect estimates of a mixed model vary across the optimisers tried by `lme4::allFit()`. It checks that a model has reached a stable solution: tight clusters of points indicate agreement between optimisers, whereas scattered points signal a fragile fit.

Usage

```
optimizer_fixef_plot(
  x,
  intercept = TRUE,
  select_terms = NULL,
  interaction = c("times", "asterisk", "colon", "space"),
  labels = NULL,
  number_optimizers = TRUE,
  free_y = TRUE,
  ncol = NULL,
  point_size = 2,
  palette = NULL,
  y_lab = "Fixed effect",
  title = NULL
)
```

Arguments

x	The object returned by lme4::allFit(), its summary() or a data frame with one row per optimiser-by-term combination: an optimiser column (optimizer, Optimizer or optimiser), a term column (term, fixed_effect, parameter or Var2) and a value column (value, estimate or Estimate).
intercept	Whether to keep the intercept panel. Defaults to TRUE.
select_terms	Optional character vector of terms to display. The intercept is always kept when intercept = TRUE.
interaction	Passed to format_terms() for the panel titles.
labels	Optional named character vector renaming the term panels, such as c(conditionunrelated = "Unrelated priming"). When x is a raw lme4::allFit() object, a factor's coefficients are labelled with the variable and the level automatically (conditionunrelated becomes condition: unrelated), and these labels override that default.
number_optimizers	Whether to prefix each optimiser name with a number, so that the legend also serves as an index.
free_y	Whether to give each panel its own y-axis range. This is advisable, because the intercept and the slopes usually occupy very different scales.
ncol	Number of facet columns. If NULL, chosen automatically.
point_size	Point size.
palette	Colours for the optimisers. Defaults to depictr_palette().
y_lab, title	Axis label and plot title.

Details

Each fixed effect occupies its own panel, with its own y-axis. The function also accepts a plain data frame, so it can be used without 'lme4'.

Value

A ggplot2::ggplot object.

Examples

```
# Without lme4, build the input data frame directly:
set.seed(1)
df <- expand.grid(
  optimizer = c("bobyqa", "Nelder_Mead", "nlminbwrap"),
  term = c("(Intercept)", "rainfall", "fertiliser")
)
df$value <- c(5, 5.01, 4.99, 0.3, 0.31, 0.29, -0.2, -0.18, -0.21)
optimizer_fixef_plot(df)


if (requireNamespace("lme4", quietly = TRUE)) {
  m <- lme4::lmer(life_satisfaction ~ stress + (1 | region),
                  data = wellbeing_survey)
```

```
  af <- lme4::allFit(m)
  optimizer_fixef_plot(af)
}
```

`outlier_plot` *Box and violin plots highlighting outliers*

Description

Shows the distribution of a numeric variable, optionally by group, as a box plot, a violin plot or both, with points beyond the 1.5 * IQR fences highlighted so that outliers are easy to spot before modelling.

Usage

```
outlier_plot(
  data,
  y,
  group = NULL,
  type = c("box", "violin", "both"),
  flag = TRUE,
  outlier_colour = depictr_accent(),
  palette = NULL,
  title = NULL,
  y_lab = NULL
)
```

Arguments

`data`	A data frame.
`y`	The numeric variable (string or unquoted name).
`group`	Optional grouping variable on the x-axis. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
`type`	One of `"box"`, `"violin"` or `"both"`.
`flag`	Whether to highlight outliers (points beyond 1.5 * IQR).
`outlier_colour`	Colour for highlighted outliers.
`palette`	Colours for the groups. Defaults to `depictr_palette()`.
`title`, `y_lab`	Plot title and value-axis label.

Value

A ggplot2::ggplot object.

Examples

```
outlier_plot(crop_yield, yield)
outlier_plot(lexical_decision, RT, group = condition, type = "both")
```

`palette_preview` *Preview the depictr palettes*

Description

Displays the colours returned by `depictr_palette()` as labelled swatches, which helps when choosing how many groups to show, selecting a palette type or documenting a figure. Each swatch's hex label is drawn in black or white, whichever has the higher WCAG contrast against that tile, so every label reaches the 4.5:1 that WCAG 2.2 asks of text.

Usage

```
palette_preview(
  n = 8,
  type = c("qualitative", "sequential", "diverging", "all"),
  cvd = c("none", "deutan", "protan", "tritan")
)
```

Arguments

`n`	Number of colours to preview.
`type`	Palette type to preview: `"qualitative"`, `"sequential"`, `"diverging"` or `"all"` to show all three, one panel each, without the hex labels.
`cvd`	Colour-vision-deficiency simulation for the tiles: `"none"` (the default, true colours), `"deutan"` (red-green, deuteranopia), `"protan"` (red-green, protanopia) or `"tritan"` (blue-yellow, tritanopia).

Details

Set `cvd` to render the swatches as they would appear under a colour-vision deficiency, using the Machado et al. (2009) simulation. The tiles are recoloured to the simulated appearance, while the labels keep the original hex code, which shows whether a colourblind reader could still tell two groups apart.

Value

A ggplot2::ggplot object.

References

Machado GM, Oliveira MM, Fernandes LAF (2009). "A physiologically-based model for simulation of color vision deficiency." *IEEE Transactions on Visualization and Computer Graphics*, **15**(6), 1291–1298. doi:10.1109/TVCG.2009.113.

World Wide Web Consortium (2024). "Web Content Accessibility Guidelines (WCAG) 2.2." W3C Recommendation, https://www.w3.org/TR/WCAG22/. Accessed 2026-08-18.

Examples

```
palette_preview()
palette_preview(7, type = "sequential")
palette_preview(type = "all")
palette_preview(cvd = "deutan")
```

---

palette_safety *Check that a palette stays distinguishable under each deficiency*

Description

For normal vision and each deficiency at full severity, the palette's colours are converted to CIELAB and the smallest colour difference between any two of them is found. The difference is CIEDE2000 (Sharma, Wu & Dalal, 2005) unless `metric` asks for CIE76. The lower this minimum, the more likely two categories are to be confused. A palette counts as safe when the minimum across all four conditions is at least `threshold`.

Usage

```
palette_safety(colours = NULL, threshold = 5, metric = c("ciede2000", "cie76"))
```

Arguments

`colours`	The palette to test. Defaults to the depictr qualitative palette. Needs at least two colours, since fewer than two have no pairwise distance.
`threshold`	The smallest acceptable colour difference, on the scale of `metric`.
`metric`	The colour difference to measure: `"ciede2000"`, the default, or `"cie76"`, the Euclidean distance in CIELAB. `"cie76"` is deprecated and gives a warning once per session. It is kept so that results computed with it can be reproduced, and a later release will remove it.

Details

The Euclidean distance in CIELAB, known as CIE76, overstates the difference between saturated colours: the blues `#005b96` and `#0060a8` are 8.20 apart by CIE76 and 2.69 by CIEDE2000. ggplot2's default palette of five colours passes the check under CIE76, although under deuteranopia its sky blue and its pink come within 2.74 of each other in CIEDE2000.

The default `threshold` of 5 is calibrated against the built-in palette of `depictr_palette()`, which is based on the Okabe-Ito set with the brand blue and a grey in place of its blue and black. That palette's closest pair (reddish purple against grey) is 6.43 apart under full deuteranopia, so the cut must lie below that to pass the built-in palette, while still flagging colours that become near-identical under a deficiency. The difference includes lightness, which survives colour-vision deficiency, so two colours that share a hue but differ in lightness are correctly treated as distinguishable. Full severity is the worst case, and most colour-vision deficiency is milder.

This looks at a palette in the abstract. To audit a finished figure, which uses only as many colours as it has groups and has text and a background besides, see `check_figure()`.

Value

A list with `min_delta_e` (the worst case across conditions), `by_condition` (a named numeric vector, one minimum colour difference for normal vision and for each deficiency), `worst_condition` and `worst_pair` (the closest colours and where they were closest), `safe` and `threshold`.

References

Okabe M, Ito K (2008). "Color Universal Design (CUD): How to make figures and presentations that are friendly to colorblind people." https://jfly.uni-koeln.de/color/. Accessed 2026-06-14.

Machado GM, Oliveira MM, Fernandes LAF (2009). "A physiologically-based model for simulation of color vision deficiency." *IEEE Transactions on Visualization and Computer Graphics*, **15**(6), 1291–1298. doi:10.1109/TVCG.2009.113.

Sharma G, Wu W, Dalal EN (2005). "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations." *Color Research & Application*, **30**(1), 21–30. doi:10.1002/col.20070.

Examples

```
palette_safety()

# Two nearly identical colours are flagged.
palette_safety(c("#005b96", "#015c97"))

# ggplot2's default palette of five colours fails under CIEDE2000.
hue5 <- c("#F8766D", "#A3A500", "#00BF7D", "#00B0F6", "#E76BF3")
palette_safety(hue5)$safe
```

---

pca_plot *PCA biplot*

Description

Runs a principal component analysis on the numeric columns of a data frame, or takes an existing `stats::prcomp()` object, and draws a biplot: the observations projected onto two components, with the variable loadings shown as arrows. The observations can be coloured by a grouping variable.

Usage

```
pca_plot(
  x,
  cols = NULL,
  group = NULL,
  components = c(1, 2),
  scale = TRUE,
  loadings = TRUE,
  point_alpha = 0.7,
  palette = NULL,
  title = NULL
)
```

Arguments

x	A data frame or a stats::prcomp() object.
cols	When x is a data frame, the numeric columns to analyse (by default, all numeric columns).
group	Optional grouping variable mapped to colour. When x is a data frame, give a column of x, unquoted or as a string, or a vector with one value per row of x. Rows with missing values count, although the analysis drops them. When x is a stats::prcomp() object, give a vector with one value per observation. A name that matches no column, or a vector of another length, is an error.
components	Length-2 integer vector: which components to plot.
scale	Whether to scale the variables to unit variance before the PCA. This is advisable when the variables are on different scales.
loadings	Whether to draw variable-loading arrows.
point_alpha	Point transparency.
palette	Colours for the groups. Defaults to depictr_palette().
title	Plot title.

Details

Each arrow is a variable's loadings on the two components, taken from the rotation matrix and multiplied by a factor that all the arrows share. The factor makes the largest coordinate of any arrow three quarters of the largest absolute score. Both axes use that one factor, as stats::biplot() uses one ratio for both, and the aspect ratio is fixed at 1. An arrow's angle on the page is therefore its angle in loading space, whatever the size of the figure. The panel's height and width follow the ranges of its two axes, so a figure of another shape leaves blank space around the panel. Because the scores are drawn as they are and the arrows as loadings, the angle between two arrows does not estimate the correlation between their variables. Arrows and labels are drawn in a neutral dark grey (grey20), which is not in the default palette, so no arrow reads as part of a group.

Value

A ggplot2::ggplot object.

Examples

```
pca_plot(crop_yield, cols = c("rainfall", "fertiliser", "soil_ph", "yield"),
         group = "treatment")
```

`posterior_plot` *Plot posterior distributions*

Description

Displays posterior draws, or more generally bootstrap or simulation draws, as a distribution per parameter, in the style of a half-eye (a density slab with a point-and-interval beneath it), an interval-only forest plot, a gradient interval or a dotplot. The full shape of the posterior is shown, so skew, multimodality and the relative mass on either side of a reference value are all visible, where a point and two limits would hide them. The intervals are central (equal-tailed) quantile intervals, which leave the same share of the draws in either tail.

Usage

```
posterior_plot(
  draws,
  style = c("halfeye", "interval", "gradient", "dots"),
  point = c("median", "mean"),
  widths = c(0.66, 0.95),
  interaction = c("times", "asterisk", "colon", "space"),
  labels = NULL,
  reference_line = 0,
  rope = NULL,
  pd = FALSE,
  facet = FALSE,
  scales = c("fixed", "free"),
  colour = depictr_brand(),
  fill = depictr_brand(),
  rope_fill = depictr_reference(),
  rope_alpha = 0.15,
  title = NULL,
  x_lab = "Value",
  caption = NULL
)
```

Arguments

`draws`	Posterior draws: a fitted model (`brms` or `rstanarm`), a `posterior` draws object, a matrix or a long or wide data frame (see Details).
`style`	One of `"halfeye"` (density slab and interval, the default), `"interval"` (point and two nested intervals, no slab), `"gradient"` (a colour-graded interval) or `"dots"` (a quantile dotplot). Any other value is an error. When 'ggdist' is missing, every style falls back to `"interval"`.

point	Central summary: "median" or "mean".
widths	Two interval widths (inner and outer), as probabilities, each drawn as a central (equal-tailed) quantile interval. The outer width is used for the caption and the displayed interval mass.
interaction	Passed to format_terms() for the parameter labels.
labels	Optional named character vector renaming parameters, such as c(conditionunrelated = "condition"). Unmatched parameters fall back to format_terms(). For a brms fit, or a draws object holding b_ parameters, the names are given without the b_ prefix. An unnamed vector gives one label per parameter, in the order the parameters appear in draws.
reference_line	Position of a vertical reference line, or NULL or NA to omit it. No reference is meaningful for every parameter, so this defaults to 0, the usual line of no effect for differences and slopes, but should be set or cleared deliberately.
rope	Optional length-2 numeric c(lo, hi) giving a region of practical equivalence to shade behind the distributions.
pd	If TRUE, annotate each parameter with its probability of direction relative to reference_line, the posterior probability that the parameter lies on its majority side of the reference (a value in [0.5, 1]). Requires a finite reference_line.
facet	Whether to give each parameter its own panel with a free x-axis, laid out one per row. This keeps small parameters legible when a large one, typically the intercept, would otherwise stretch the shared axis. A shortcut for scales = "free". Defaults to FALSE.
scales	Either "fixed" (the default, a single shared x-axis) or "free" (one free-scaled panel per parameter). When facet = TRUE, it is set to "free".
colour, fill	Colours for the point and interval and for the density slab. Default to the depictr brand blue.
rope_fill, rope_alpha	Fill colour and opacity of the ROPE band.
title, x_lab	Plot title and value-axis label.
caption	Plot caption. The default (NULL) states the interval mass in the caption. Pass NA to omit a caption.

Details

Draws may be supplied in many shapes: a fitted Bayesian model (brms or rstanarm), a posterior draws object, a wide data frame or matrix with one column of draws per parameter, or a long data frame with a parameter column named parameter, term, .variable or variable and a value column named value, .value, draw or estimate. In a wide data frame, the numeric columns .chain, .iteration, .draw, draw, chain, iteration and .row are read as sampler indices and left out. For fitted models, only the fixed-effect (population) parameters are kept, with the brms b_ prefix stripped. A draws object that holds b_ parameters, as posterior::as_draws_df() returns for a brms fit, is read the same way, with a message naming the variables it dropped. Pass it through as.data.frame() to keep the other variables, as a model of your own whose names happen to start with b_ may need. The log densities lp__ and lprior are never drawn as parameters. Two parameters that would share a display label, such as the b_Intercept and centred Intercept of

a plain table of brms draws, are an error until one is renamed with labels. A matrix of posterior summaries, such as brms::fixef() returns, holds no draws and is refused.

The slab styles use the 'ggdist' package. When 'ggdist' is not installed, the function falls back to the point-and-nested-interval display (as style = "interval") with an informative message.

Value

A ggplot2::ggplot object.

Examples

```
# Wide draws: one column per parameter
set.seed(1)
draws <- data.frame(
  intercept = rnorm(400, 5, 0.3),
  slope = rnorm(400, 0.8, 0.15),
  `slope:group` = rnorm(400, -0.2, 0.2),
  check.names = FALSE
)
posterior_plot(draws)

# A region of practical equivalence and a probability-of-direction label
posterior_plot(draws, rope = c(-0.1, 0.1), pd = TRUE)

# When one parameter is much larger than the others, give each its own
# free-scaled panel:
posterior_plot(draws, facet = TRUE)
```

---

power_curve_plot *Plot a power analysis curve*

Description

Draws statistical power against sample size, with a dashed line at a target power (80% by default). The input is usually a power curve produced by simr::powerCurve(), though a tidy data frame works equally well, so the plot can be redrawn without repeating a power simulation that is often slow to run.

Usage

```
power_curve_plot(
  x,
  target = 0.8,
  x_lab = "Sample size",
  x_breaks = NULL,
  x_expand = NULL,
  ribbon = TRUE,
```

```
  title = NULL,
  interaction = c("times", "asterisk", "colon", "space"),
  n = NULL,
  power = NULL,
  group = NULL,
  lower = NULL,
  upper = NULL,
  y_lab = NULL,
  solved = NULL
)
```

Arguments

x	A powerCurve object from 'simr', or a data frame with a sample size column (n, nlevels, x, sample_size or n_subject), a power column (power, mean or p) and optional confidence limits (<power>_lo and <power>_hi, lower and upper, ci_low and ci_high or conf.low and conf.high).
target	Target power, drawn as a horizontal reference line. Use NA to omit it.
x_lab	X-axis label.
x_breaks	Approximate number of x-axis breaks.
x_expand	Optional value(s) to extend the x-axis to.
ribbon	Whether to draw the confidence band as a shaded ribbon (TRUE) or as error bars (FALSE).
title	Plot title. If NULL and x is a 'simr' power curve, the predictor name stored in the object is used.
interaction	Passed to format_terms() when deriving the title from a 'simr' object.
n, power	Columns holding the sample size and the power, as strings or unquoted names. By default, they are found among the names listed under x. power may name any proportion on the 0-1 scale, such as pilotr's p_meaningful.
group	Column that separates several curves, as a string or an unquoted name. Each level is drawn as its own line, in its own colour and line type. By default, an effect, param or term column with more than one level is used, with a message. Rows that share a sample size with no such column are an error, because joining them would draw a zigzag.
lower, upper	Columns holding the lower and upper confidence limits. Name both or neither. By default, they are found among the names listed under x.
y_lab	Y-axis label. By default, 'Power' when the power column is one of the names listed under x, and the column's name otherwise.
solved	A sample size solved from the curve, with its confidence interval: a list with numeric value, lo and hi, as pilotr's target_n() and solve_curve() return, or a one-row data frame with the same columns. Other fields are ignored. The solved value is drawn as a dashed vertical line and its interval as a grey band behind the curve. An interval that reaches past the swept sizes widens the x-axis to show it.

Details

A table may hold several curves, one per effect or design, stacked by row. Each is drawn as its own line, told apart by colour and line type. The tables pilotr returns are read as they are: `power_curve_mixed()` gives `n_subject`, `effect`, `power`, `power_lo` and `power_hi`, and `precision_curve()` gives `n_subject`, `param` and the decision probabilities `p_meaningful` and `p_equivalent` with their own `_lo` and `_hi` bounds, which `power` can name. The sample size that pilotr's `target_n()` solves from such a curve, and its interval, can be marked on the plot with `solved`.

Value

A ggplot2::ggplot object.

Examples

```
pc <- data.frame(
  nlevels = c(10, 20, 30, 40, 50, 60),
  mean = c(0.18, 0.34, 0.52, 0.66, 0.79, 0.88),
  lower = c(0.10, 0.25, 0.42, 0.56, 0.70, 0.81),
  upper = c(0.28, 0.44, 0.62, 0.75, 0.86, 0.93)
)
power_curve_plot(pc, title = "Power for the condition effect")

# Two curves in one table, laid out as pilotr's power_curve_mixed() returns
# them: one row per sample size and effect.
curves <- data.frame(
  n_subject = rep(c(20, 40, 60, 80), each = 2),
  effect = rep(c("condition", "frequency"), times = 4),
  power = c(0.31, 0.12, 0.58, 0.21, 0.78, 0.33, 0.90, 0.45),
  power_lo = c(0.23, 0.07, 0.48, 0.14, 0.69, 0.25, 0.83, 0.36),
  power_hi = c(0.41, 0.19, 0.67, 0.30, 0.85, 0.43, 0.94, 0.55)
)
power_curve_plot(curves, group = "effect", x_lab = "Participants")

# The sample size at which pc reaches 80% power, with its interval, in the
# form pilotr's target_n() returns. The values here are illustrative.
power_curve_plot(pc, solved = list(value = 51, lo = 44, hi = 60))
```

---

pr_curve_plot *Precision-recall curve*

Description

Plots precision against recall for a binary classifier and reports the average precision (the area under the curve). The precision-recall curve is more informative than the ROC curve when the positive class is rare, because it ignores the many true negatives. A horizontal line marks the no-skill baseline (the positive-class prevalence). Pass a named list of models or of (actual, score) pairs to overlay several colour-coded curves with a legend and per-curve average precision.

Usage

```
pr_curve_plot(
  x,
  score = NULL,
  colour = depictr_brand(),
  f1 = FALSE,
  title = NULL
)
```

Arguments

x	A binomial glm, the vector of observed outcomes (0 and 1, logical or a two-level factor with the positive class second) or a named list of models or of (actual, score) pairs to overlay.
score	When x is an outcome vector, the matching scores or predicted probabilities, or a list of them for several models.
colour	Curve colour for a single model. Defaults to the depictr brand blue. Ignored when several models are overlaid.
f1	Whether to mark the maximum-F1 operating point, the threshold that maximises the harmonic mean of precision and recall, on each curve.
title	Plot title.

Value

A ggplot2::ggplot object. The average precision is stored in attr(plot, "average_precision"), a named vector when several models are supplied. With f1 = TRUE, the maximum-F1 point is stored in attr(plot, "max_f1"): a list of threshold, recall, precision and f1, or a list of such lists named by model when several models are supplied.

Examples

```
# The clinical-trial adverse event is rare (about 10% positive), which is
# where a precision-recall curve is worth drawing.
gfit <- glm(adverse_event ~ biomarker + age + arm,
            data = clinical_trial, family = binomial)
pr_curve_plot(gfit)

# Mark the maximum-F1 operating point.
pr_curve_plot(gfit, f1 = TRUE)

# Compare two models with per-curve average precision in the legend.
reduced <- glm(adverse_event ~ biomarker, data = clinical_trial,
               family = binomial)
pr_curve_plot(list(Full = gfit, Reduced = reduced))
```

qq_plot *Normal quantile-quantile plot*

Description

A normal Q-Q plot for a numeric vector or for the standardised residuals of a fitted model, with a reference line and an optional confidence band that makes departures from normality easier to judge. Points outside the band are unusual under normality. In a normal sample, about `level` of the points fall inside a pointwise band.

Usage

```
qq_plot(
  x,
  colour = depictr_brand(),
  title = NULL,
  x_lab = "Theoretical quantiles",
  y_lab = NULL,
  band = TRUE,
  band_type = c("pointwise", "simulate"),
  level = 0.95,
  n_sim = 1000,
  band_fill = depictr_reference(),
  seed = NULL
)
```

Arguments

`x`	A numeric vector, or a fitted `lm` or `glm` model, whose standardised residuals are used.
`colour`	Point colour. Defaults to the depictr brand blue.
`title`, `x_lab`, `y_lab`	Title and axis labels.
`band`	Whether to draw a confidence band or envelope.
`band_type`	Band construction: `"pointwise"` (analytic order-statistic standard errors, the default) or `"simulate"` (a Monte-Carlo envelope).
`level`	Confidence level for the band.
`n_sim`	Number of simulations for `band_type = "simulate"`.
`band_fill`	Fill colour of the band.
`seed`	Optional integer seed for the simulated envelope, for reproducibility.

Details

Two band constructions are offered. `"pointwise"` is analytic: it uses the large-sample standard error of the i-th order statistic, $\mathrm{se} = \frac{\hat{\sigma}}{\phi(z_i)}\sqrt{p_i(1-p_i)/n}$, around the fitted reference line. `"simulate"` builds a Monte-Carlo envelope by repeatedly drawing normal samples of the same size and taking the empirical quantiles of the simulated order statistics, which needs no large-sample approximation.

The band measures departures from a normal distribution, which the residuals of a linear model should follow. The standardised deviance residuals of a non-Gaussian `glm` are not expected to be normal, even when the model is correctly specified: those of a binary model take only two values at each fitted value. Points outside the band then say little about the fit. For a `glm`, `residual_diagnostics_plot()` and `model_report()` plot randomised quantile residuals (Dunn & Smyth, 1996), which are standard normal under a correct model.

Value

A ggplot2::ggplot object.

References

Dunn PK, Smyth GK (1996). "Randomized quantile residuals." *Journal of Computational and Graphical Statistics*, **5**(3), 236–244. doi:10.1080/10618600.1996.10474708.

Examples

```
qq_plot(rnorm(100))
qq_plot(rt(100, df = 3), band_type = "simulate")
fit <- lm(yield ~ rainfall + fertiliser, data = crop_yield)
qq_plot(fit)
```

---

raincloud_plot *Raincloud plot*

Description

A raincloud plot combines three views of a distribution: a half-violin density (the cloud), a narrow boxplot and the raw jittered points (the rain). It shows the shape, the summary and the individual observations together, which a boxplot alone does not. The plot is built with base R and ggplot2 alone, with no extra package dependency.

Usage

```
raincloud_plot(
  data,
  y,
  group = NULL,
  width = 0.4,
  point_alpha = 0.4,
```

```
  palette = NULL,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`data`	A data frame.
`y`	The numeric variable (string or unquoted name).
`group`	Optional grouping variable on the x-axis. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
`width`	Maximum width of the half-violin.
`point_alpha`	Transparency of the rain points.
`palette`	Colours for the groups. Defaults to `depictr_palette()`.
`title`, `x_lab`, `y_lab`	Title and axis labels.

Details

Groups with fewer than two observations cannot have a density estimated, so their half-violin is omitted (the points and box are still drawn) and a warning is issued.

Value

A ggplot2::ggplot object.

Examples

```
raincloud_plot(lexical_decision, RT, group = condition)
raincloud_plot(crop_yield, yield, group = treatment)
```

---

random_effects_plot *Caterpillar plot of random effects*

Description

Displays the conditional modes (BLUPs) of a mixed model's random effects as a sorted point-and-interval plot, known as a caterpillar plot. It is the usual way to inspect by-group departures from the average and to identify unusual groups. When the plot holds more than 30 distinct level names across all its panels, the names and their tick marks are left off the y-axis, since they would overlap. Add `ggplot2::theme(axis.text.y = ggplot2::element_text())` to bring the names back.

Usage

```
random_effects_plot(
  x,
  conf_level = 0.95,
  sort = TRUE,
  point_colour = depictr_brand(),
  title = NULL,
  x_lab = "Random effect"
)
```

Arguments

x	Either a mixed model fitted with 'lme4' (merMod), or a data frame with one row per group level and columns such as level or group for the level, estimate for the estimate and either conf.low and conf.high or std.error for the interval. An optional term column facets the plot. A model may have several grouping factors and several terms per factor, correlated or not. The forms (1 + x | g), (1 + x || g) and (1 | g) + (0 + x | g) are all drawn. The intervals are the conditional modes plus or minus the normal quantile times the conditional standard deviations that lme4::ranef(condVar = TRUE) reports. A model with one term in all is drawn in a single panel labelled 'Random effect', and otherwise each term has a panel labelled 'factor: term'.
conf_level	Confidence level when intervals are derived from standard errors, a single number strictly between 0 and 1.
sort	Whether to order the levels by their estimate.
point_colour	Colour for the points and intervals.
title, x_lab	Plot title and value-axis label.

Value

A ggplot2::ggplot object.

Examples

```
# From a data frame of by-group estimates:
re <- data.frame(
  level = paste0("G", 1:10),
  estimate = sort(rnorm(10)),
  std.error = runif(10, 0.2, 0.5)
)
random_effects_plot(re)


if (requireNamespace("lme4", quietly = TRUE)) {
  m <- lme4::lmer(RT ~ condition + (1 | participant), data = lexical_decision)
  random_effects_plot(m)
}
```

`residual_diagnostics_plot` *Residual-diagnostics panel for a fitted model*

Description

Combines the classic regression diagnostic plots (residuals against fitted values, a normal Q-Q plot of the standardised residuals, a scale-location plot and residuals against leverage) into a single panel using 'patchwork'. The function works with `lm` and `glm` objects.

Usage

```
residual_diagnostics_plot(
  model,
  which = c("resid_fitted", "qq", "scale_location", "resid_leverage"),
  ncol = 2,
  point_alpha = 0.6,
  smooth = TRUE,
  title = NULL,
  glm_panels = TRUE,
  seed = NULL
)
```

Arguments

`model`	A fitted `lm` or `glm` model.
`which`	Character vector choosing which panels to show, any of `"resid_fitted"`, `"qq"`, `"scale_location"` and `"resid_leverage"`.
`ncol`	Number of columns in the panel layout.
`point_alpha`	Point transparency.
`smooth`	Whether to add a loess guide line to the residual panels.
`title`	Overall title for the panel.
`glm_panels`	For a `glm`, whether to use the GLM-appropriate binned and quantile-residual panels (the default) instead of the classic `lm` panels. Ignored for an `lm`.
`seed`	Optional integer seed for the randomisation of discrete quantile residuals, for reproducible `glm` Q-Q panels.

Details

For a `glm`, the default panels suit a GLM (`glm_panels = TRUE`): the panel of residuals against fitted values is replaced by a binned-residual plot (Gelman & Hill, 2007, see `binned_residual_plot()`), which is readable even for binary or count models, and the Q-Q panel uses randomised quantile residuals (Dunn & Smyth, 1996), which are standard normal under a correctly specified model whatever the response family. In a Gaussian or Gamma model, prior weights enter these residuals as they enter the fit, through each observation's variance. The scale-location and leverage panels are unchanged. Set `glm_panels = FALSE` to draw the classic `lm`-style panels for a `glm` as well.

Value

A 'patchwork' object (printable like a ggplot2::ggplot).

References

Gelman A, Hill J (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, Cambridge, UK. ISBN 978-0-521-68689-1. doi:10.1017/CBO9780511790942.

Dunn PK, Smyth GK (1996). "Randomized quantile residuals." *Journal of Computational and Graphical Statistics*, **5**(3), 236–244. doi:10.1080/10618600.1996.10474708.

Examples

```
fit <- lm(yield ~ rainfall + fertiliser + soil_ph, data = crop_yield)
residual_diagnostics_plot(fit)

residual_diagnostics_plot(fit, which = c("resid_fitted", "qq"))

# A logistic GLM gets binned and quantile-residual panels automatically:
gfit <- glm(adverse_event ~ biomarker + age + arm,
            data = clinical_trial, family = binomial)
residual_diagnostics_plot(gfit)
```

---

ridgeline_plot *Ridgeline plot*

Description

Shows the distribution of a numeric variable across the levels of a grouping variable as a column of partially overlapping density curves, one ridge per group, stacked up the y-axis. It compares many distributions in a compact space, so shifts in location and shape are easy to follow, and is clearer than many overlaid densities once there are several groups.

Usage

```
ridgeline_plot(
  data,
  x,
  group,
  overlap = 1.4,
  alpha = 0.85,
  scale_height = TRUE,
  palette = NULL,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`data`	A data frame.
`x`	The numeric variable whose distribution is shown (string or unquoted name).
`group`	The grouping variable (string or unquoted name), with one ridge per level. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
`overlap`	How far each ridge extends into the next, as a multiple of the row spacing. `1` makes the tallest ridge just touch the next baseline, and larger values overlap more. Defaults to `1.4`.
`alpha`	Fill transparency of the ridges.
`scale_height`	Whether to scale every ridge to the same peak height (`TRUE`, the default) or keep the true relative densities (`FALSE`).
`palette`	Colours for the groups. Defaults to `depictr_palette()`.
`title, x_lab, y_lab`	Plot title and axis labels.

Details

Densities are computed with `stats::density()` and scaled to a common height. The `overlap` argument controls how far each ridge reaches into the one above. Groups with fewer than two non-missing values are dropped with a warning. The plot is built with base R and ggplot2, with no extra package dependency.

Value

A ggplot2::ggplot object.

Examples

```
ridgeline_plot(wellbeing_survey, life_satisfaction, region)
ridgeline_plot(lexical_decision, RT, condition)
```

`roc_curve_plot` *ROC curve*

Description

Plots the receiver operating characteristic (ROC) curve for a binary classifier and reports the area under the curve (AUC). The input can be a fitted binomial `glm`, a pair of vectors (observed binary outcome and a continuous score), or, to compare several models, a named list of models or of (actual, score) pairs, which are overlaid as colour-coded curves with a legend and a per-curve AUC. An optional bootstrap confidence band can be drawn for the curve of a single model and its AUC, and the Youden's J operating point can be marked.

Usage

```
roc_curve_plot(
  x,
  score = NULL,
  colour = depictr_brand(),
  ci = FALSE,
  conf_level = 0.95,
  youden = FALSE,
  legend_inside = FALSE,
  title = NULL
)
```

Arguments

x	A binomial glm, the vector of observed outcomes (0 and 1, logical or a two-level factor with the positive class second) or a named list of models or of (actual, score) pairs to overlay (see Details).
score	When x is an outcome vector, the matching vector of scores or predicted probabilities. When x is a named list of outcome vectors, a matching list of score vectors, named or in the same order.
colour	Curve colour for a single model. Defaults to the depictr brand blue. Ignored when several models are overlaid, which take the colourblind-aware scale_colour_depictr() palette.
ci	Bootstrap confidence band for a single ROC curve and its AUC. FALSE, the default, draws none, TRUE uses 2000 resamples and a positive integer sets the number of resamples. The resamples come from the session's random-number stream, so call set.seed() first for a reproducible band. Ignored when several models are overlaid.
conf_level	Confidence level for the bootstrap band, a single number strictly between 0 and 1.
youden	Whether to mark the Youden's J operating point, the threshold that maximises sensitivity + specificity - 1, on each curve.
legend_inside	When TRUE and several models are overlaid, the legend is drawn inside the panel, over a translucent background, in the bottom-right corner, which the curve leaves empty. Otherwise, it is drawn in a right-hand margin. Defaults to FALSE.
title	Plot title.

Details

A named list overlays one curve per element, such as roc_curve_plot(list("Full" = fit_full, "Reduced" = fit_reduced)). Each element may be a glm, a list or data frame of length 2 holding (actual, score), or an outcome vector paired with the matching element of a score list.

Value

A ggplot2::ggplot object. The AUC is stored in attr(plot, "auc"), a named vector when several models are supplied. When ci draws a band, which it does for a single model, the bootstrap interval

of the AUC is stored in `attr(plot, "auc_ci")`, a vector named `lower` and `upper`. With `youden = TRUE`, the Youden point is stored in `attr(plot, "youden")`: a list of `threshold`, `fpr`, `tpr`, `j`, `sensitivity` and `specificity`, or a list of such lists named by model when several models are supplied.

Examples

```
gfit <- glm(accuracy ~ word_frequency + condition + RT,
            data = lexical_decision, family = binomial)
roc_curve_plot(gfit)

# Mark the Youden operating point and add a bootstrap band.
roc_curve_plot(gfit, youden = TRUE, ci = 200)

# Compare two models with a colour-coded legend and per-curve AUC.
reduced <- glm(accuracy ~ word_frequency, data = lexical_decision,
               family = binomial)
roc_curve_plot(list(Full = gfit, Reduced = reduced))
```

---

save_plot *Save a plot with publication-ready defaults*

Description

A wrapper around `ggplot2::ggsave()` with defaults for figures in papers and reports: a moderate size, 300 dpi and an output directory created if needed. The device is inferred from the file extension, and PDF, EPS and TIFF files are written to pass the technical checks journals run on submitted figures. With `check = TRUE`, the figure is also audited with `check_figure()` at the width it is saved at.

Usage

```
save_plot(
  filename,
  plot = ggplot2::last_plot(),
  width = 7,
  height = 4.5,
  units = "in",
  dpi = 300,
  check = FALSE,
  ...
)
```

Arguments

`filename`	Output file path. The extension sets the device, such as `.png`, `.pdf`, `.eps` or `.tiff`.
`plot`	The plot to save. Defaults to the last plot drawn.

width, height	Dimensions.
units	Units for width and height.
dpi	Resolution for raster devices. An EPS rasterises its translucent layers, and whatever they overlap, at this resolution or at 600 dpi, whichever is higher.
check	Whether to audit the figure with check_figure() at the width it is saved at, given as both width_cm and render_width_cm, so that the text sizes it reports are the printed ones. A warning names each check the figure fails. Defaults to FALSE. The audit runs before the file is written, so a plot it cannot read is refused without saving. For other thresholds, such as a journal's own text floor, call check_figure() with the saved width as both widths.
...	Passed to ggplot2::ggsave(), which passes what it does not use on to the device. A device or device argument given here overrides the defaults described under 'Formats journals accept'.

Value

The filename, invisibly. With check = TRUE, the audit report is attached to it as the attribute "check", which attr(, "check") reads.

Formats journals accept

When no device is given, three formats are written differently from ggplot2's defaults.

A PDF is written with grDevices::cairo_pdf(), which embeds its fonts. ggplot2's default, grDevices::pdf(), names the fonts without embedding them.

An EPS or PostScript file is written with grDevices::cairo_ps(), with its fonts embedded too. PostScript has no transparency, and grDevices::postscript() leaves every translucent layer out of the file, so the points and the confidence band of the plot in the examples below would be missing. cairo_ps() rasterises those layers, together with whatever they overlap, at fallback_resolution, by default max(dpi, 600) here. A message says so when a layer of the plot, or of any panel of a patchwork, is translucent. A PDF keeps such layers as vector graphics.

A TIFF is compressed with LZW, which is lossless. Uncompressed, a figure of 7 by 4.5 inches at 300 dpi takes 8.1 MiB, and one at the largest size PLOS accepts takes more than the 10 MB PLOS allows. The TIFF is drawn on white unless bg is given, so a transparent background, as in ggplot2::theme_void(), leaves no alpha channel in the file.

Without cairo, a PDF is written with pdf() and a message says how to embed its fonts with grDevices::embedFonts(), and an EPS is written with postscript() and a warning. R for macOS from CRAN loads cairo only where XQuartz (https://www.xquartz.org) is installed, and a Mac with neither XQuartz nor ragg writes the TIFF uncompressed, with a warning from grDevices::tiff(). A device, or a device argument such as fallback_resolution, compression or bg, passed in ... is used as given, so device = "pdf", device = "eps" or device = "tiff" restores ggplot2's own behaviour.

Which formats three publishers accepted on 2 October 2026:

Format	PLOS	Nature (final artwork)	Elsevier
PDF	No	Yes, as vector art	Yes
EPS	Yes, with fonts embedded	Yes, as vector art	Yes, with fon

TIFF	Yes, LZW-compressed RGB or greyscale, no alpha channel, at most 10 MB	No	Yes

Nature also asks for fonts embedded as TrueType 2 or 42, never Type 3, and for line art and text that are not rasterised. A figure with translucent layers therefore goes to Nature as a PDF. Elsevier asks for a graph sent as a TIFF at 1000 dpi, so pass `dpi = 1000` for one.

References

PLOS ONE. Figures. https://journals.plos.org/plosone/s/figures

Nature. Guide to preparing final artwork. https://www.nature.com/documents/nature-final-artwork.pdf

Nature research figure guide. Preparing figures: Our specifications. https://research-figure-guide.nature.com/figures/preparing-figures-our-specifications/

Elsevier. Artwork overview. https://www.elsevier.com/about/policies-and-standards/author/artwork-and-media-instructions/artwork-overview

Examples

```
p <- scatter_trend(crop_yield, fertiliser, yield)
tmp <- file.path(tempdir(), "yield.png")
save_plot(tmp, p)

# A TIFF well within the 10 MB that PLOS allows.
save_plot(file.path(tempdir(), "yield.tiff"), p)

# Audited at the 8.9 cm column it is saved for, where its text prints at
# the size it is drawn.
saved <- save_plot(file.path(tempdir(), "yield-column.png"), p,
                   width = 8.9, height = 6, units = "cm", check = TRUE)
attr(saved, "check")
```

---

scale_colour_depictr *depictr colour and fill scales*

Description

Discrete ggplot2 scales using `depictr_palette()`. These are the colour and fill scales used throughout the package. They honour the global `options(depictr.palette = )` (through `depictr_palette()`) and `options(depictr.na_value = )` settings (see `depictr_options()`).

Usage

```
scale_colour_depictr(
  n = NULL,
  palette = NULL,
  na.value = depictr_opt("na_value"),
```

```
  order = c("default", "contrast"),
  ...
)

scale_color_depictr(
  n = NULL,
  palette = NULL,
  na.value = depictr_opt("na_value"),
  order = c("default", "contrast"),
  ...
)

scale_fill_depictr(
  n = NULL,
  palette = NULL,
  na.value = depictr_opt("na_value"),
  order = c("default", "contrast"),
  ...
)
```

Arguments

n	Optional number of colours to draw from depictr_palette(). By default, ggplot2 requests exactly as many colours as there are groups, so pass n only to force a fixed slice of the palette. With more than eight groups, the built-in qualitative palette is interpolated, which loses the colour-vision-deficiency guarantee and warns when the plot is drawn. Facet the groups or use a sequential scale when there are that many.
palette	Optional palette override: a function of one argument (the number of colours) returning a character vector of colours. Defaults to depictr_palette().
na.value	Colour for NA levels. Defaults to the resolved depictr.na_value option (the muted grey "grey80" unless changed).
order	"default", or "contrast" to draw the groups in the contrast-first order of depictr_palette(), with which two or three groups pass every colour row of check_figure(). More than four groups warn at draw time. It cannot be combined with palette.
...	Passed to ggplot2::discrete_scale().

Value

A ggplot2 scale that can be added to a plot.

Examples

```
library(ggplot2)
ggplot(crop_yield, aes(rainfall, yield, colour = treatment)) +
  geom_point() +
  scale_colour_depictr() +
```

```
  theme_depictr()

# The vermillion in the orange's place, at 3.87:1 against white
ggplot(crop_yield, aes(rainfall, yield, colour = treatment,
                       shape = treatment)) +
  geom_point() +
  scale_colour_depictr(order = "contrast") +
  theme_depictr()
```

`scatter_trend` *Scatter plot with a fitted trend*

Description

Plots `y` against `x` with an optional fitted trend line and confidence band, optionally split by a grouping variable.

Usage

```
scatter_trend(
  data,
  x,
  y,
  group = NULL,
  method = "lm",
  se = TRUE,
  point_alpha = 0.6,
  palette = NULL,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`data`	A data frame.
`x`, `y`	The variables for the horizontal and vertical axes (string or unquoted column name).
`group`	Optional grouping variable mapped to colour. A numeric code counts as a group, but a numeric column with more than 20 distinct values is refused as continuous. Rows with a missing group are dropped with a message, and levels without rows are left out of the plot and its palette.
`method`	Smoothing method passed to `ggplot2::geom_smooth()`, such as `"lm"` or `"loess"`, or `NULL` for no trend line.
`se`	Whether to draw the confidence band around the trend.
`point_alpha`	Point transparency.

palette	Colours for the groups. Defaults to depictr_palette().
title, x_lab, y_lab	Title and axis labels, which default to the variable names.

Value

A ggplot2::ggplot object.

Examples

```
scatter_trend(crop_yield, fertiliser, yield)
scatter_trend(crop_yield, fertiliser, yield, group = treatment,
                  method = "lm")
```

---

scree_plot *Scree plot*

Description

Shows the proportion of variance explained by each principal component, as bars with a cumulative line. It is the customary aid for deciding how many components to retain.

Usage

```
scree_plot(x, cols = NULL, scale = TRUE, n = NULL, title = NULL)
```

Arguments

x	A data frame or a stats::prcomp() object.
cols	When x is a data frame, the numeric columns to analyse.
scale	Whether to scale variables to unit variance before the PCA.
n	Maximum number of components to display.
title	Plot title.

Value

A ggplot2::ggplot object.

Examples

```
scree_plot(wellbeing_survey,
           cols = c("age", "income", "stress", "sleep_hours",
                    "exercise_days", "life_satisfaction"))
```

`seasonal_plot` *Seasonal-subseries (cycle) plot*

Description

Draws a seasonal-subseries plot, also called a cycle plot: the series is split into one small panel per season, such as one panel per calendar month for monthly data, and within each panel, the value is drawn across successive cycles, such as years. A horizontal reference line in every panel marks that season's mean. The plot shows the seasonal pattern (the differences between panels) and the within-season trend over time (the slope inside each panel) at the same time, which a single overlaid line cannot show.

Usage

```
seasonal_plot(
  x,
  frequency = NULL,
  style = c("subseries", "season"),
  season_labels = NULL,
  means = TRUE,
  point = TRUE,
  palette = NULL,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`x`	A `ts` object, or a numeric vector (then `frequency` is required), holding one series. A multivariate `ts` or a matrix with several columns is refused.
`frequency`	Number of observations per period, such as 12 for monthly data. Taken from `x` when it is a `ts`. A frequency that is not a whole number, such as the 365.25 / 7 of weekly data, is rounded down to whole seasons, as `stats::stl()` rounds it, and a message says so. The seasons are then counted from the first observation.
`style`	`"subseries"` for the faceted cycle plot (one panel per season) or `"season"` for one line per cycle across the seasons.
`season_labels`	Optional character vector of length `frequency` giving the season labels of the panels or the axis, such as `month.abb`. When `NULL` (the default), labels are inferred: month abbreviations for frequency 12, quarter labels for frequency 4 and the season index otherwise.
`means`	Whether to draw the per-season mean reference line (`style = "subseries"` only).
`point`	Whether to add points as well as the connecting line.

`palette` Colours used for the cycle lines when `style = "season"`. Defaults to a blue ramp from `#4a8fc0` through the brand blue to `#08315a`, whose palest colour has a contrast of 3.51:1 against white. The sequential ramp of `depictr_palette()` starts paler, at `#e6eff5`, which suits a fill but leaves a line close to invisible.

`title`, `x_lab`, `y_lab` Title and axis labels.

Details

Two layouts are offered. With `style = "subseries"` (the default), the panels are faceted side by side, each tracing one season across cycles: the classic Cleveland cycle plot, matching `feasts::gg_subseries()`. With `style = "season"`, every cycle is drawn as its own line over the seasons on a shared axis (the seasonal-plot layout of `forecast::ggseasonplot()`), which helps to spot an unusual year. The lines run from a mid blue for the first cycle to a dark blue for the last, so they read in time order. Past about six cycles, neighbouring cycles take colours too close to tell apart, and the colours then show the order of the cycles more than which is which. When individual cycles must be told apart, use `style = "subseries"`, where each cycle keeps its own place on the axis of every panel, or plot fewer cycles.

Value

A ggplot2::ggplot object.

See Also

`decompose_plot()`, `timeseries_plot()`

Examples

```
# Monthly air passengers: rising within-month trend, summer peak
seasonal_plot(AirPassengers)
seasonal_plot(AirPassengers, style = "season")
```

---

silhouette_plot *Silhouette plot*

Description

Draws a silhouette plot for a clustering: one horizontal bar per observation, grouped and sorted within cluster, with the average silhouette width shown per cluster and overall. The silhouette width $s(i)$ compares how close an observation is to its own cluster against the nearest other cluster, and lies in $[-1, 1]$. Higher is better, and a negative width suggests that the point may belong to another cluster. It is a standard internal measure of cluster quality.

Usage

```
silhouette_plot(
  data,
  clusters,
  cols = NULL,
  scale = TRUE,
  distance = "euclidean",
  palette = NULL,
  title = NULL
)
```

Arguments

`data`	A data frame, a numeric matrix or a stats::dist object.
`clusters`	A vector of cluster assignments, one per observation (per row of `data`, or matching the objects in a `dist`). Required.
`cols`	When `data` is a data frame, the numeric columns to use (by default, all numeric columns).
`scale`	Whether to scale variables to unit variance before computing the distances (ignored when `data` is a `dist`).
`distance`	Distance measure passed to `stats::dist()` (ignored when `data` is already a `dist`).
`palette`	Colours for the clusters. Defaults to `depictr_palette()`.
`title`	Plot title.

Details

The widths are computed with `cluster::silhouette()` when the 'cluster' package is installed, and otherwise with an equivalent base-R implementation, so the function works with no extra dependency.

Value

A ggplot2::ggplot object. The per-observation silhouette table is attached as the attribute `"silhouette"` and the average width as `"avg_width"`. The table's `cluster` and `neighbor` columns carry the assignments passed in `clusters`, as does each band label.

References

Rousseeuw PJ (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." *Journal of Computational and Applied Mathematics*, **20**, 53–65. doi:10.1016/0377-0427(87)901257.

Examples

```
cl <- kmeans(scale(crop_yield[c("rainfall", "fertiliser", "soil_ph",
                                "yield")]), 3)$cluster
silhouette_plot(crop_yield, cl,
                cols = c("rainfall", "fertiliser", "soil_ph", "yield"))
```

`simulate_cvd` *Simulate how colours appear under a colour-vision deficiency*

Description

Maps each colour to the colour a reader with a given form and severity of colour-vision deficiency would perceive, using the physiologically based model of Machado, Oliveira and Fernandes (2009). The transformation is defined on linear-light RGB, so a colour is decoded from sRGB to linear RGB (the IEC 61966-2-1 transfer functions), transformed, then re-encoded.

Usage

```
simulate_cvd(colours, deficiency, severity = 1)
```

Arguments

`colours`	Character vector of colours, in any form `grDevices::col2rgb()` understands.
`deficiency`	The deficiency to simulate: `"protan"` or `"deutan"` (the two red-green forms) or `"tritan"` (blue-yellow).
`severity`	Severity in `[0, 1]`. Zero leaves the colours unchanged and one is the full deficiency. Machado et al. publish a matrix for each tenth of severity, which is used as it stands at 0.1, 0.2 and so on. A severity between two tenths takes the linear interpolation of their matrices, as the authors suggest and as the colorspace package does.

Details

This is the simulator behind `palette_preview()`'s `cvd` argument, behind `palette_safety()` and behind the colour-separability rows of `check_figure()`. `simulate_figure()` applies it to every colour of a finished figure.

Value

A character vector of opaque, lower-case six-digit hex colours, the same length as `colours`. Any alpha is dropped, whereas `simulate_figure()` keeps it.

References

Machado GM, Oliveira MM, Fernandes LAF (2009). "A physiologically-based model for simulation of color vision deficiency." *IEEE Transactions on Visualization and Computer Graphics*, **15**(6), 1291–1298. doi:10.1109/TVCG.2009.113.

Examples

```
simulate_cvd(c("#005b96", "#e69f00"), "deutan")

# Half severity moves the colours only part of the way.
simulate_cvd(c("#005b96", "#e69f00"), "deutan", severity = 0.5)
```

`simulate_figure` *Show a finished figure as a reader with a colour-vision deficiency sees it*

Description

`simulate_cvd()` simulates a set of colours, and this function the figure they end up in, legend, colourbar, text and background included.

Usage

```
simulate_figure(plot, deficiency, severity = 1)
```

Arguments

`plot`	A ggplot or a patchwork, as it would be saved.
`deficiency`	The deficiency to simulate: `"protan"` or `"deutan"` (the two red-green forms) or `"tritan"` (blue-yellow).
`severity`	Severity in `[0, 1]`. Zero leaves the colours unchanged and one is the full deficiency. Machado et al. publish a matrix for each tenth of severity, which is used as it stands at 0.1, 0.2 and so on. A severity between two tenths takes the linear interpolation of their matrices, as the authors suggest and as the colorspace package does.

Details

The plot is turned into the table of grobs that would be drawn, with `ggplot2::ggplotGrob()` or, for a patchwork, `patchwork::patchworkGrob()`. Every colour in that table then goes through `simulate_cvd()`. That covers the line and fill colours of each grob, at any depth, and each cell of a raster, such as a colourbar, a `ggplot2::geom_raster()` layer or an image added with `ggplot2::annotation_raster()`. Each colour keeps its alpha channel. Missing and transparent colours are left as they are, and so are gradient and pattern fills such as `grid::linearGradient()`, which keep their original colours.

The colours are changed after the plot is built. Changing the colours of the data before it is built would leave the legend keys and any colourbar in their original colours, so the legend would no longer match the figure.

The figure stays vector graphics, so it can be saved at any size.

Value

A gtable, which `grid::grid.draw()` draws and `save_plot()` saves.

References

Machado GM, Oliveira MM, Fernandes LAF (2009). "A physiologically-based model for simulation of color vision deficiency." *IEEE Transactions on Visualization and Computer Graphics*, **15**(6), 1291–1298. doi:10.1109/TVCG.2009.113.

See Also

`check_figure()`, which measures what this shows.

Examples

```
library(ggplot2)
p <- ggplot(crop_yield, aes(rainfall, yield, colour = treatment)) +
  geom_point() +
  scale_colour_depictr() +
  theme_depictr()

# The figure as a reader with deuteranopia sees it.
seen <- simulate_figure(p, "deutan")
grid::grid.newpage()
grid::grid.draw(seen)

# Saved for a side-by-side comparison with the original.
save_plot(file.path(tempdir(), "yield-deutan.png"), seen)
```

---

summary_table *A descriptive summary in the style of 'Table 1'*

Description

Builds the kind of descriptive table that opens many empirical papers: numeric variables are summarised as mean (SD), categorical variables as counts and percentages, optionally split into one column per level of a grouping variable. The result is a plain data frame, ready to pass to `knitr::kable()` or a table package.

Usage

```
summary_table(
  data,
  vars = NULL,
  group = NULL,
  digits = 1,
  missing = TRUE,
  max_levels = 20
)
```

Arguments

data	A data frame.
vars	Names of the columns to summarise, as a character vector. If NULL, all columns except group are used, with high-cardinality identifier-like columns skipped (see Details).
group	Optional name of the grouping column, given as a string rather than unquoted. One summary column is produced per level, alongside an overall column. Records whose group value is missing get a trailing Missing column of their own, so the group sizes still sum to the overall N.
digits	Number of decimal places for numeric summaries.
missing	Whether to add a Missing, n (%) row for variables that contain missing values. Defaults to TRUE.
max_levels	When vars is NULL, character or factor columns whose distinct-level count is at least max_levels and exceeds half the number of rows are treated as identifiers and skipped. Defaults to 20.

Details

The first row of the table always reports the sample size (N) overall and per group. By default, each variable is also followed by a Missing, n (%) row whenever it contains missing values. Set missing = FALSE to leave these out.

A date or date-time column (Date, POSIXct) gets a single Range row, written as <earliest> to <latest> in each column. The cells give the date alone when every time in the column falls at midnight, and the time to the minute, or to the second when any time has seconds. A duration (difftime) is summarised as mean (SD) in its own units, which the statistic names, as in Mean (SD), days.

When vars is NULL, high-cardinality character or factor columns, whose number of distinct levels approaches the number of rows, such as identifier columns, are skipped automatically, with a message, so that they are not expanded into hundreds of one-per-row entries. Pass such a column explicitly in vars to override this.

Value

A data frame with columns variable, statistic, Overall, one column per group level and, when the grouping variable has missing values, a trailing Missing column. The first row reports N.

Examples

```
summary_table(crop_yield, vars = c("yield", "rainfall", "treatment"))
summary_table(wellbeing_survey,
              vars = c("life_satisfaction", "education"),
              group = "region")
```

`survival_plot` *Kaplan-Meier survival plot*

Description

Draws Kaplan-Meier survival curves, optionally by group, with stepwise confidence limits and censoring marks. The Kaplan-Meier estimate and its Greenwood standard error are computed with base R, so no modelling package is required. A `survfit` object from the 'survival' package is also accepted.

Usage

```
survival_plot(
  time,
  status = NULL,
  group = NULL,
  conf_level = 0.95,
  censor_marks = TRUE,
  risk_table = FALSE,
  median_line = FALSE,
  logrank = FALSE,
  risk_breaks = NULL,
  palette = NULL,
  legend_inside = FALSE,
  title = NULL,
  x_lab = "Time",
  y_lab = "Survival probability",
  conf_type = c("log-log", "log", "plain")
)
```

Arguments

`time`	A numeric vector of follow-up times, a data frame with `time`, `status` and optional `group` columns, or a `survfit` object. Every time must be finite. A missing or infinite time is an error, since dropping it would change the number at risk without notice.
`status`	Event indicator: a vector, or a column name when `time` is a data frame (by default its `status` or `event` column). Either the 0/1 convention (`0` = censored, `1` = event) or the `survival::Surv()` 1/2 convention (`1` = censored, `2` = event) is accepted, read over all groups together. Logical values are also allowed. Other codings raise an error, among them a code that is not a whole number and a factor or character indicator, whose values do not say which one is the event. Observations whose status is missing are dropped with a message, since they record neither an event nor a censoring.
`group`	Optional grouping variable: a vector, or a column name when `time` is a data frame (by default its `group`, `strata` or `arm` column, if any). Observations whose group is missing are dropped with a message, since they belong to no arm.

conf_level	Confidence level for the limits, a single number strictly between 0 and 1 (NA to omit them). For a survfit object, the limits are recomputed at this level, even for one made with conf.type = "none", since it keeps its standard errors. An object that stores no standard errors, or whose lower limits follow conf.lower = "peto" or "modified", keeps the limits it holds, with a warning when they are not the ones asked for.
censor_marks	Whether to mark censoring times with a +.
risk_table	Whether to add a number-at-risk table beneath the curves (composed with 'patchwork'). Defaults to FALSE.
median_line	Whether to draw dashed guides to each group's median survival time and label it. Defaults to FALSE.
logrank	Whether to add a log-rank test of the group difference as a subtitle (two or more groups only). Defaults to FALSE.
risk_breaks	Optional numeric vector of times at which to report the number at risk. Defaults to the curve's x-axis breaks.
palette	Colours for the groups. Defaults to depictr_palette().
legend_inside	When TRUE and there are several groups, the group legend is drawn inside the panel, over a translucent background, in the bottom-left corner, which a decreasing survival curve always leaves empty. Otherwise, it is drawn in a right-hand margin. Defaults to FALSE.
title, x_lab, y_lab	Title and axis labels.
conf_type	The scale on which the confidence limits are built: "log-log" (the default), "log" or "plain", as set out under Details.

Details

The confidence limits are pointwise. By default, they are built on the log-log scale, as $S^{\exp(\pm z\sigma/\log S)}$, where σ is the Greenwood standard error of $\log S$. These limits stay between 0 and 1, and in small samples they cover better than the untransformed limits $S \pm zS\sigma$ (Borgan & Liestøl, 1990). conf_type = "plain" draws the untransformed limits, clipped to that range, and conf_type = "log" draws $S\exp(\pm z\sigma)$, capped at 1, as survival::survfit() does by default. The limits are 1 while the estimate is 1, and where a curve falls to 0, they keep the values of the step before. The limits of a survfit object are recomputed from its standard errors at conf_level and conf_type, so they equal those that survfit() gives with the same conf.int and conf.type (conf_level names the objects that keep their own).

For publication-ready figures, the plot offers the three annotations of a Kaplan-Meier display in the style of the 'survminer' package, each behind its own argument and off by default:

- risk_table = TRUE adds a number-at-risk table, a small panel composed beneath the curves with 'patchwork' that gives the number of subjects still at risk in each group at the x-axis breaks.
- median_line = TRUE draws median-survival guides, dashed reference lines from the 0.5 survival level down to each group's median survival time, with the median value labelled. The median is the time at which the curve falls below 0.5. Where the curve stays at 0.5 until a later drop, or until its last time, the median is the midpoint of that stretch, as in survival::quantile.survfit(). A group whose curve never reaches 0.5 has no estimable median and is skipped.

- `logrank = TRUE` adds a log-rank test for two or more groups, with the chi-squared statistic and its p-value as a subtitle. For vector or data-frame input, `survival::survdiff()` computes it when the 'survival' package is installed. For a `survfit` object, which keeps only its risk and event tables, or without 'survival', an equivalent base-R log-rank test is computed from the per-group risk and event tables.

Value

A ggplot2::ggplot object or, when `risk_table = TRUE`, a 'patchwork' object stacking the curves above the risk table.

References

Kaplan EL, Meier P (1958). "Nonparametric estimation from incomplete observations." *Journal of the American Statistical Association*, **53**(282), 457–481. doi:10.1080/01621459.1958.10501452.

Greenwood M (1926). *A report on the natural duration of cancer*, volume 33 of *Reports on Public Health and Medical Subjects*. His Majesty's Stationery Office, London.

Borgan Ø, Liestøl K (1990). "A note on confidence intervals and bands for the survival function based on transformations." *Scandinavian Journal of Statistics*, **17**(1), 35–41.

Mantel N (1966). "Evaluation of survival data and two new rank order statistics arising in its consideration." *Cancer Chemotherapy Reports*, **50**(3), 163–170.

Examples

```
set.seed(1)
n <- 200
grp <- sample(c("control", "treated"), n, replace = TRUE)
time <- rexp(n, rate = ifelse(grp == "treated", 0.05, 0.1))
cens <- runif(n, 0, 30)
obs  <- pmin(time, cens)
event <- as.integer(time <= cens)
survival_plot(obs, event, group = grp, title = "Survival by treatment")

# survminer-style figure: risk table, median guides and a log-rank test
data(clinical_trial)
survival_plot(clinical_trial$time, clinical_trial$event,
              group = clinical_trial$arm, risk_table = TRUE,
              median_line = TRUE, logrank = TRUE,
              x_lab = "Months", title = "Overall survival")
```

`theme_depictr` *The depictr ggplot2 theme*

Description

The minimal theme used by every plotting function in the package. It is a light modification of `ggplot2::theme_minimal()` with subtle gridlines, centred titles and comfortable margins.

Usage

```
theme_depictr(
  base_size = depictr_opt("base_size"),
  base_family = depictr_opt("base_family"),
  grid = "xy"
)
```

Arguments

base_size	Base font size, in points. Defaults to the depictr.base_size option.
base_family	Base font family. Defaults to the depictr.base_family option.
grid	Which major gridlines to keep: "xy", "x", "y" or "none".

Details

The default base_size and base_family come from the global options depictr.base_size and depictr.base_family (see depictr_options()), so the package-wide font size can be set once. Arguments passed explicitly override them. The title is drawn in the brand colour, which options(depictr.brand =) sets and which the depictr_brand() defaults of other functions stand for.

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(crop_yield, aes(fertiliser, yield)) +
  geom_point() +
  theme_depictr()
```

---

threshold_plot *Classification metrics versus decision threshold*

Description

Sweeps the decision threshold across the full range of scores and plots the chosen classification metrics, any of sensitivity (recall), specificity, precision and F1, as colour-coded curves. It complements the ROC and precision-recall curves when choosing an operating point: it shows directly how each metric changes as the cut-off moves and, by default, marks the Youden's J and maximum-F1 thresholds.

Usage

```
threshold_plot(
  x,
  score = NULL,
  metrics = c("sensitivity", "specificity", "precision", "f1"),
  mark = c("youden", "f1"),
  title = NULL
)
```

Arguments

x	A binomial glm, or the vector of observed outcomes (0 and 1, logical or a two-level factor with the positive class second).
score	When x is an outcome vector, the matching scores or predicted probabilities.
metrics	Which metrics to draw: any subset of "sensitivity", "specificity", "precision" and "f1". "recall" is accepted as an alias for "sensitivity".
mark	Which optimal operating points to mark with a vertical line: any subset of "youden" (max sensitivity + specificity - 1), drawn dashed, and "f1" (max F1), drawn dotted. Use character(0) or NULL to mark none.
title	Plot title.

Details

At threshold t, a case is predicted positive when its score is $\geq t$. The metrics are evaluated at every distinct score, the points at which the confusion matrix can change, so the curves are exact step functions and not a coarse grid. Between two adjacent scores, each metric keeps its value at the upper one, so each step changes at a score and runs flat below it.

Value

A ggplot2::ggplot object. The Youden and max-F1 thresholds are stored in attr(plot, "thresholds").

Examples

```
gfit <- glm(accuracy ~ word_frequency + RT + condition,
            data = lexical_decision, family = binomial)
threshold_plot(gfit)

# The trade-off between sensitivity and specificity only, without markers.
threshold_plot(gfit, metrics = c("sensitivity", "specificity"),
               mark = NULL)
```

tidy_estimates *Extract a tidy table of estimates*

Description

tidy_estimates() turns the output of a model (or an existing data frame of results) into a single standardised table with the columns term, estimate, std.error, conf.low and conf.high. coefficient_plot() and compare_models() build on it, and it is useful on its own.

Usage

```
tidy_estimates(x, conf_level = 0.95, exponentiate = FALSE, ...)
```

Arguments

x	A fitted model or a data frame of results (see Details for the columns read).
conf_level	Confidence (or credible) level for the interval, a single number strictly between 0 and 1.
exponentiate	Whether to exponentiate the estimates and interval limits, which turns log-odds into odds ratios and log rates into rate ratios. The std.error column stays on the link scale, as in broom::tidy(). A data frame is read as estimates on the link scale. Defaults to FALSE.
...	Passed to methods, and to broom::tidy() in the fallback. The merMod method also accepts effects, which supports only "fixed".

Details

Methods are provided for lm, glm and merMod (mixed models fitted with 'lme4') objects, and for data frames. For any other model class, the function falls back to broom::tidy() when the 'broom' package is installed.

Confidence intervals are Wald intervals, estimate +/- multiplier * standard error, with the multiplier the model's own tests use. An lm object gets the t interval of stats::confint(). A glm whose dispersion is estimated (Gaussian, Gamma, inverse Gaussian and quasi families) gets t on the residual degrees of freedom, since stats::summary.glm() then tests with t. One whose dispersion is fixed (binomial, Poisson and MASS::glm.nb()) gets z. The choice is read from the column the summary reports, t value or z value. At 12 residual degrees of freedom, the t interval is about 11% wider than the z interval, and at 100 about 1% wider. A merMod fit gets z, since 'lme4' gives no degrees of freedom. Supply a data frame with your own intervals (for example profiled or posterior intervals) to override this.

A data frame is read by column name, taking for each quantity the first of these names that it holds. The term comes from term, parameter, Parameter, predictor, variable, fixed_effect or rowname, otherwise from the row names. The estimate comes from estimate, Estimate, frequentist_estimate, coef, mean or value, and a data frame without one is refused. The standard error comes from std.error, Std. Error, std_error, SE or se. The limits come from conf.low, CI_2.5, 2.5 %, lower, ci_low, Q2.5, lwr or l-95% CI and from conf.high, CI_97.5, 97.5 %, upper, ci_high, Q97.5, upr or u-95% CI. A row that holds both limits keeps them as

they stand. A row that lacks either limit but has a standard error gets both from it, as a z interval at `conf_level`. Other columns are dropped.

Value

A data frame with one row per term and the columns `term`, `estimate`, `std.error`, `conf.low` and `conf.high`.

Examples

```
fit <- lm(yield ~ rainfall + fertiliser + treatment, data = crop_yield)
tidy_estimates(fit)

# Odds ratios from a logistic regression:
gfit <- glm(accuracy ~ word_frequency + condition,
            data = lexical_decision, family = binomial)
tidy_estimates(gfit, exponentiate = TRUE)

# A data frame of pre-computed estimates is standardised, not re-fitted:
df <- data.frame(
  parameter = c("a", "b"),
  Estimate = c(0.2, -0.4),
  "2.5 %" = c(0.1, -0.6),
  "97.5 %" = c(0.3, -0.2),
  check.names = FALSE
)
tidy_estimates(df)
```

`timeseries_plot` *Time-series plot*

Description

Plots one or more series over time, with an optional moving-average overlay. The input can be a `ts` object, a numeric vector or matrix, or a data frame with a time column, a value column and an optional grouping column. A multivariate `ts`, such as `cbind(mdeaths, fdeaths)`, or a matrix with several columns gives one series per column, named after its column.

Usage

```
timeseries_plot(
  x,
  time = NULL,
  value = NULL,
  group = NULL,
  rolling = NULL,
  forecast = NULL,
  frequency = NULL,
  level = 0.95,
```

```
  lambda = NULL,
  palette = NULL,
  point = FALSE,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

x	A ts object, a numeric vector or matrix, or a data frame.
time	When x is a data frame, the time column (string or unquoted name). When x is a numeric vector or matrix, an optional vector of times, one per value or per row.
value	When x is a data frame, the value column.
group	When x is a data frame, an optional grouping column mapped to colour. A factor keeps the order of its levels in the legend and the palette, and levels no row uses are left out. Other columns are ordered by first appearance.
rolling	Optional integer window for a centred moving-average overlay. The data are ordered by time (within each group) before the moving average is computed, so unsorted input is handled correctly. An odd window averages each observation with the same number of neighbours on either side. An even window, such as 12 for monthly data, gives the 2×12 moving average, a 12-point average followed by a 2-point one (Hyndman & Athanasopoulos, 2021, Section 3.3). The two observations at the ends of the 13 it spans count half, which centres the average on its observation, as equal weights over an even window cannot. It needs a series one observation longer than the window.
forecast	Optional forecast overlay for a single, ungrouped series. Either an integer horizon (number of future steps) to forecast with the built-in STL plus random walk with drift (see ts_forecast()), or a pre-computed forecast supplied as a data frame with columns time, fit and, optionally, lwr and upr, for instance from a fitted forecast::forecast() object. The point forecast continues the line and a shaded prediction-interval ribbon is drawn behind it. The built-in forecast is captioned with its coverage and method, and its band widens with the horizon on the scale it is made on (see lambda). For a numeric x, it is made from the values in time order and continues the history's own time axis, one typical step at a time. Missing values at either end of the series are trimmed before forecasting, so the forecast starts from the last observation, and a missing value inside the series is an error.
frequency	Number of observations per period, used only to coerce a numeric x to a ts when an integer forecast horizon is requested.
level	Prediction-interval coverage for the built-in forecast, a single number strictly between 0 and 1, such as 0.95.
lambda	Optional Box-Cox transformation for the built-in forecast, passed to ts_forecast(). 0 forecasts on the log scale, which suits a series whose seasonal swing grows with its level.
palette	Colours for the groups. Defaults to depictr_palette().

`point` Whether to add points as well as the line.

`title`, `x_lab`, `y_lab`
Title and axis labels.

Value

A ggplot2::ggplot object.

References

Hyndman RJ, Athanasopoulos G (2021). *Forecasting: Principles and practice*, 3rd edition. OTexts, Melbourne, Australia. https://otexts.com/fpp3/.

See Also

`ts_forecast()`, `decompose_plot()`, `seasonal_plot()`

Examples

```
timeseries_plot(AirPassengers, rolling = 12,
                title = "Air passengers", y_lab = "Passengers")
# One line per column of a multivariate ts
timeseries_plot(cbind(mdeaths, fdeaths), y_lab = "Deaths")
# 24-month forecast with a 90% prediction interval, made on the log scale
# because the seasonal swing grows with the number of passengers
timeseries_plot(AirPassengers, forecast = 24, level = 0.9, lambda = 0)
```

`ts_forecast` *Forecast a seasonal series with STL and a random walk with drift*

Description

A dependency-free forecaster for a single seasonal series, which forecasts by decomposition (Hyndman & Athanasopoulos, 2021, Section 5.7). `stats::stl()` with `s.window = "periodic"` estimates a seasonal component that repeats unchanged in every cycle (Cleveland et al., 1990). The seasonally adjusted series, the data minus that component, is forecast with a random walk with drift, and the last cycle of the seasonal component is added back. The result reproduces `forecast::stlf(x, method = "rwdrift", s.window = "periodic")`.

Usage

```
ts_forecast(x, h = 12, frequency = NULL, level = 0.95, lambda = NULL)
```

Arguments

x	A ts object, or a numeric vector (then frequency is required), holding one series. It needs more than two full cycles of observations. Missing values at either end are trimmed, so the forecast starts after the last observation, and a missing value inside the series is an error. A multivariate ts or a matrix with several columns is refused.
h	Forecast horizon: the number of future steps (a positive integer).
frequency	Number of observations per period. Taken from x when it is a ts.
level	Prediction-interval coverage (a single number strictly between 0 and 1).
lambda	Optional Box-Cox transformation parameter. NULL, the default, forecasts the series as it is. 0 forecasts its logarithm, and any other number forecasts (x^lambda - 1) / lambda. Every observation must then be positive. The forecasts and limits are transformed back to the original scale, where the point forecast is the median of the forecast distribution, without the bias adjustment that would make it the mean (Hyndman & Athanasopoulos, 2021, Section 5.6). A forecast or limit beyond the range the transformation can reach comes back as 0 (for a positive lambda) or Inf (for a negative one).

Details

The prediction interval at horizon h is fit +/- z * sigma_h, with z = qnorm((1 + level) / 2). The forecast standard deviation is that of a random walk with drift, sigma_h = sigma * sqrt(h * (1 + h / (T - 1))) (Hyndman & Athanasopoulos, 2021, Section 5.5), where sigma is the standard deviation of the steps of the seasonally adjusted series and T the length of the series. The band widens with the horizon, and the second term carries the uncertainty in the estimated drift.

The model assumes a seasonal pattern that adds to the level of the series and stays the same from cycle to cycle. When the seasonal swing grows with the level, as in AirPassengers, forecast on the log scale with lambda = 0. The band treats the seasonal component as known, so it leaves out the uncertainty in estimating it. In a rolling-origin backtest on five classic series whose seasonal swing does not grow with their level, the nominal 95% band covered 92% of one-step-ahead outcomes (dev/ts_forecast_backtest.R in the source repository). From three to 12 steps ahead, it covered 99%. The band therefore runs somewhat narrow at the shortest horizons and wide further out, where the uncertainty of a random walk keeps growing while these series stay close to their trend. For a different model, fit it with, for example, forecast::forecast() and pass its columns to timeseries_plot() as a data frame.

Value

A data frame with one row per forecast step and columns time (the future times, on the same scale as stats::time()), fit, lwr and upr.

References

Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990). "STL: A seasonal-trend decomposition procedure based on loess." *Journal of Official Statistics*, **6**(1), 3–73.

Hyndman RJ, Athanasopoulos G (2021). *Forecasting: Principles and practice*, 3rd edition. OTexts, Melbourne, Australia. https://otexts.com/fpp3/.

See Also

timeseries_plot(), decompose_plot()

Examples

```
# The seasonal swing of AirPassengers grows with the number of passengers,
# so it is forecast on the log scale.
fc <- ts_forecast(AirPassengers, h = 12, lambda = 0)
head(fc)
# On that scale, the band widens with the horizon
diff(log(fc$upr) - log(fc$lwr)) > 0
```

---

vif_plot *Variance inflation factor plot*

Description

Computes a variance inflation factor for each term in a model and shows them as a bar chart, with a reference line at the usual rule of thumb (`threshold`). High bars flag predictors whose coefficients are unstable because they are collinear with the others. Values are computed with base R, without the 'car' package.

Usage

```
vif_plot(model, threshold = 5, palette = depictr_palette(2), title = NULL)
```

Arguments

`model`	A fitted `lm` or `glm` model with at least two predictors.
`threshold`	Reference value for the ordinary VIF, drawn as a line. For models with multi-column terms, it is shown on the $\mathrm{GVIF}^{1/(2\,\mathrm{df})}$ scale as $\sqrt{\mathrm{threshold}}$.
`palette`	Length-2 colours encoding terms below and above the threshold. Defaults to the colourblind-safe depictr_palette() pair (blue and orange).
`title`	Plot title.

Details

For single-degree-of-freedom terms, this is the ordinary VIF, $1/(1-R^2)$. For terms that span several design-matrix columns (multi-level factors, or spline or polynomial bases), the function reports the generalised VIF of Fox and Monette (1992): with R the correlation matrix of the coefficient estimates other than the intercept, R_{11} the block for the term and R_{22} the block for the remaining coefficients,

$$\mathrm{GVIF} = \frac{\det(R_{11})\,\det(R_{22})}{\det(R)}.$$

When every term has a single degree of freedom, the bars are the ordinary VIFs, on a plain 'Variance inflation factor' axis with the reference line at `threshold`. If any term spans several columns, the

bars switch to the comparable $\mathrm{GVIF}^{1/(2\,\mathrm{df})}$ (which for a single-df term equals $\sqrt{\mathrm{VIF}}$), and the reference line moves to $\sqrt{\text{threshold}}$ accordingly. The x-axis is kept tight to the data: when every bar is well below the threshold, the caption reports the line, which is then not drawn.

The correlations come from the fit's unscaled coefficient covariance, so a `glm` or a weighted `lm` is measured with the weights of its fit, and the values agree with those of `car::vif()`. For an unweighted `lm` with an intercept, they are the values the formula gives with R the correlation matrix of the predictor columns, as Fox and Monette state it. A model with aliased coefficients has no VIF and is refused, with the terms they belong to named. A model without an intercept warns, as `car::vif()` does, because its VIFs then come from columns that are not centred.

Value

A ggplot2::ggplot object.

References

Fox J, Monette G (1992). "Generalized collinearity diagnostics." *Journal of the American Statistical Association*, **87**(417), 178–183. doi:10.1080/01621459.1992.10475190.

Examples

```
# Two deliberately collinear predictors: soil moisture is largely driven by
# rainfall, so both carry an inflated VIF (around 6, above the line at 5)
# while fertiliser stays near 1.
set.seed(1)
d <- crop_yield
d$soil_moisture <- 0.05 * d$rainfall + rnorm(nrow(d), sd = 2)
vif_plot(lm(yield ~ rainfall + soil_moisture + fertiliser, data = d))

# A factor with more than two levels gets a single generalised VIF per term,
# and the axis switches to GVIF^(1/(2 df))
fit2 <- lm(life_satisfaction ~ stress + sleep_hours + region,
           data = wellbeing_survey)
vif_plot(fit2)
```

---

wellbeing_survey *Simulated wellbeing survey*

Description

A reproducibly simulated cross-sectional wellbeing survey for descriptive, correlation, regression and missing-data examples. `life_satisfaction` depends on its predictors: higher stress lowers it, more sleep and exercise raise it, log-income has a positive effect of about +0.45 per natural-log unit, and age has a mild inverted-U effect that peaks in mid-life. `education` is an ordered factor. Income is missing more often at higher stress (missing at random), and because income predicts the outcome, this missingness is informative for the missing-data examples.

Usage

```
wellbeing_survey
```

Format

A data frame with 300 rows and 9 variables:

id Respondent identifier.

region Region of residence: `North`, `South`, `East` or `West`.

age Age in years.

education Highest education level (ordered factor: `secondary` < `undergraduate` < `postgraduate`).

income Annual income, with missing values, more of them at high stress.

stress Self-reported stress (1-7).

sleep_hours Typical hours of sleep per night, with missing values.

exercise_days Days of exercise per week (0-7), with missing values.

life_satisfaction Life satisfaction (1-7), with missing values.

Details

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed, and they describe no real individuals.

Source

Simulated by `data-raw/generate_datasets.R` in the source repository.

Index

∗ **datasets**
clinical_trial, 13
crop_yield, 23
lexical_decision, 51
monthly_sales, 57
wellbeing_survey, 103

acf_plot, 4
acf_plot(), *57*
arrange_plots, 5
arrange_plots(), *12*, *56*

base::strwrap(), *42*
binned_residual_plot, 6
binned_residual_plot(), *57*, *75*
broom::tidy(), *97*

calibration_plot, 7
calibration_plot(), *14*
check_figure, 8
check_figure(), *29*, *63*, *79, 80*, *82*, *88*, *90*
clinical_trial, 13
cluster::silhouette(), *87*
cluster_plot, 14
coefficient_plot, 16
coefficient_plot(), *20*, *51*, *56*, *97*
compare_models, 19
compare_models(), *43, 44*, *97*
confusion_matrix_plot, 21
correlation_heatmap, 22
correlation_heatmap(), *15*, *41*, *50*
crop_yield, 23

decompose_plot, 24
decompose_plot(), *57*, *86*, *100*, *102*
dendrogram_plot, 26
dendrogram_plot(), *15*
depictr_options, 27
depictr_options(), *29, 30*, *54*, *81*, *95*
depictr_palette, 28
depictr_palette(), *11*, *15*, *20*, *23*, *26–28*, *31, 32*, *35*, *37, 38*, *40, 41*, *43*, *47*, *49*, *59–61*, *63, 64*, *73*, *77*, *81, 82*, *84*, *86, 87*, *93*, *99*, *102*
dumbbell_plot, 30

ecdf_plot, 32
effects_plot, 33
effects_plot(), *49*, *56*
estimation_plot, 34
estimation_plot(), *47*
explore_bivariate, 37
explore_categorical, 38
explore_distribution, 39
explore_distribution(), *39*
explore_pairs, 40

format_terms, 41
format_terms(), *17*, *20*, *59*, *66*, *68*
frequentist_bayesian_plot, 42
frequentist_bayesian_plot(), *19*

gain_plot, 45
gain_plot(), *14*
ggplot2::annotation_raster(), *89*
ggplot2::discrete_scale(), *82*
ggplot2::geom_label(), *11*
ggplot2::geom_raster(), *89*
ggplot2::geom_smooth(), *37*, *83*
ggplot2::geom_text(), *11*
ggplot2::ggplot, *4*, *7, 8*, *16*, *18*, *20*, *22, 23*, *25*, *27*, *31, 32*, *34*, *36, 37*, *39–41*, *44*, *46–49*, *51*, *53, 54*, *57*, *59–61*, *64*, *67*, *69, 70*, *72–74*, *76–78*, *84*, *86, 87*, *94*, *96*, *100*, *103*
ggplot2::ggplotGrob(), *89*
ggplot2::ggsave(), *79, 80*
ggplot2::scale_colour_identity(), *10*
ggplot2::scale_colour_manual(), *10*
ggplot2::theme_minimal(), *94*

ggplot2::theme_void(), *80*
grDevices::cairo_pdf(), *80*
grDevices::cairo_ps(), *80*
grDevices::col2rgb(), *88*
grDevices::embedFonts(), *80*
grDevices::pdf(), *80*
grDevices::postscript(), *80*
grDevices::tiff(), *80*
grid::grid.draw(), *89*
grid::linearGradient(), *89*
group_comparison_plot, 46

influence_plot, 48
interaction_plot, 48

k_diagnostic, 50
k_diagnostic(), *15, 16*
knitr::kable(), *90*

lexical_decision, 51
lift_plot, 52
lift_plot(), *14*
lme4::allFit(), *58, 59*
lme4::glmer(), *34*
lme4::lmer(), *34*, *55*
lme4::refitML(), *55*

missingness_map, 53
missingness_map(), *30*
model_fit_table, 54
model_fit_table(), *56*
model_report, 56
model_report(), *12*, *72*
monthly_sales, 57

optimizer_fixef_plot, 58
optimizer_fixef_plot(), *51*
outlier_plot, 60
outlier_plot(), *30*

palette_preview, 61
palette_preview(), *13*, *88*
palette_safety, 62
palette_safety(), *8, 9*, *13*, *88*
patchwork::patchworkGrob(), *89*
patchwork::plot_annotation(), *12*
patchwork::plot_layout(), *35*
patchwork::wrap_plots(), *5*
pca_plot, 63
posterior_plot, 65
posterior_plot(), *44*
power_curve_plot, 67
pr_curve_plot, 69
pr_curve_plot(), *14*

qq_plot, 71
qq_plot(), *57*

raincloud_plot, 72
random_effects_plot, 73
residual_diagnostics_plot, 75
residual_diagnostics_plot(), *12*, *57*, *72*
ridgeline_plot, 76
roc_curve_plot, 77
roc_curve_plot(), *21*

save_plot, 79
save_plot(), *9*, *11*, *89*
scale_color_depictr
*(*scale_colour_depictr*)*, 81
scale_colour_depictr, 81
scale_colour_depictr(), *11*, *27, 28*, *30*, *78*
scale_fill_depictr
*(*scale_colour_depictr*)*, 81
scale_fill_depictr(), *28*, *30*
scatter_trend, 83
scatter_trend(), *37*
scree_plot, 84
seasonal_plot, 85
seasonal_plot(), *25*, *100*
silhouette_plot, 86
silhouette_plot(), *16*
simulate_cvd, 88
simulate_cvd(), *10*, *89*
simulate_figure, 89
simulate_figure(), *88*
stats::acf(), *4*
stats::biplot(), *64*
stats::confint(), *97*
stats::cor(), *22, 23*, *41*
stats::cutree(), *15*
stats::decompose(), *25*
stats::density(), *77*
stats::dist, *26*, *87*
stats::dist(), *26*, *87*
stats::glm(), *55*
stats::hclust, *26*
stats::hclust(), *26*
stats::kmeans(), *15*, *50*

stats::logLik(), *54*
stats::pacf(), *4*
stats::prcomp(), *63, 64*, *84*
stats::predict(), *34*
stats::stl(), *25*, *85*, *100*
stats::summary.glm(), *97*
stats::time(), *101*
summary_table, 90
survival::quantile.survfit(), *93*
survival::Surv(), *92*
survival::survdiff(), *94*
survival::survfit(), *93*
survival_plot, 92
survival_plot(), *13*

theme_depictr, 94
theme_depictr(), *27*
threshold_plot, 95
tidy_estimates, 97
tidy_estimates(), *16, 17*, *20*, *34*
timeseries_plot, 98
timeseries_plot(), *57*, *86*, *101, 102*
ts_forecast, 100
ts_forecast(), *25*, *99, 100*

vif_plot, 102

wellbeing_survey, 103