Package 'depictr'

September 1, 2026

Type Package

Title A Unified Toolkit for Visualising Statistical Models and Data

Version 0.3.0

Description A cohesive, publication-ready toolkit of plots that span the whole analysis workflow with one consistent look. 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 and Kaplan-Meier survival curves; time series (trends, autocorrelation, decomposition, seasonality and forecasting); 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); uncertainty and power; and reporting helpers (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, and the package ships with reproducibly simulated datasets so that 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 (>= 3.5.0), stringr, scales, patchwork (>= 1.3.0), rlang, stats, grDevices, utils, Rdpack

Suggests lme4, lmerTest, broom, simr, knitr, rmarkdown, survival, ggdist, posterior, boot, cluster, colorspace, 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 CRAN

Date/Publication 2026-09-01 08:30:09 UTC

Contents

acf_plot . . . 4
arrange_plots . . . 5
binned_residual_plot . . . 6
calibration_plot . . . 7
check_figure . . . 8
clinical_trial . . . 10
cluster_plot . . . 11
coefficient_plot . . . 13
compare_models . . . 14
confusion_matrix_plot . . . 16
correlation_heatmap . . . 17
crop_yield . . . 18
decompose_plot . . . 19
dendrogram_plot . . . 21
depictr_options . . . 22
depictr_palette . . . 23
dumbbell_plot . . . 25
ecdf_plot . . . 26
effects_plot . . . 27
estimation_plot . . . 28
explore_bivariate . . . 30
explore_categorical . . . 31
explore_distribution . . . 32
explore_pairs . . . 34
format_terms . . . 35
frequentist_bayesian_plot . . . 36
gain_plot . . . 38
group_comparison_plot . . . 39

influence_plot . . . 40
interaction_plot . . . 41
k_diagnostic . . . 42
lexical_decision . . . 43
lift_plot . . . 45
missingness_map . . . 46
model_fit_table . . . 47
model_report . . . 47
monthly_sales . . . 48
optimizer_fixef_plot . . . 49
outlier_plot . . . 51
palette_preview . . . 52
palette_safety . . . 53
pca_plot . . . 54
posterior_plot . . . 55
power_curve_plot . . . 57
pr_curve_plot . . . 58
qq_plot . . . 60
raincloud_plot . . . 61
random_effects_plot . . . 62
residual_diagnostics_plot . . . 63
ridgeline_plot . . . 64
roc_curve_plot . . . 66
save_plot . . . 67
scale_colour_depictr . . . 68
scatter_trend . . . 69
scree_plot . . . 70
seasonal_plot . . . 71
silhouette_plot . . . 73
simulate_cvd . . . 74
summary_table . . . 75
survival_plot . . . 76
theme_depictr . . . 78
threshold_plot . . . 79
tidy_estimates . . . 81
timeseries_plot . . . 82
ts_forecast . . . 83
vif_plot . . . 84
wellbeing_survey . . . 85

Index **87**

`acf_plot` *Autocorrelation plot*

Description

Plots the autocorrelation (or partial autocorrelation) function of a series, with the approximate significance bounds, as a clean 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.
`lag_max`	Maximum lag (passed to `stats::acf()` / `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.

Value

A ggplot2::ggplot object.

Examples

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

`arrange_plots` *Compose several plots into one figure*

Description

A thin, friendly wrapper around `patchwork::wrap_plots()` that adds the common finishing touches: collecting duplicate legends into one, an overall title and subtitle, and automatic panel tags (A, B, C, ...).

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, e.g. `"A"`, `"1"` or `"i"`; `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. Under a well-specified model the binned residuals scatter around zero, and about 95% of them should fall within the grey ± 2 standard-error band, computed per bin as $2\,\hat{\sigma}_{\text{bin}}/\sqrt{n_{\text{bin}}}$ from the residuals in that bin. Systematic departures (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
)
```

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 roughly $\sqrt{n}$ bins (bounded to a sensible range).
`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.

Value

A ggplot2::ggplot object. The per-bin summary (mean fitted, mean residual, SE bound 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)
```

`calibration_plot` *Calibration plot*

Description

Assesses how well predicted probabilities match observed frequencies. The scores are split into bins; 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 / (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 / (actual, score) pairs to overlay.
`score`	When `x` is an outcome vector, the matching predicted probabilities (or a list of them for the multi-model case).
`bins`	Number of (equal-count) bins.
`colour`	Point/line colour for the single-model case. 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. Use `NA` to omit the intervals. Intervals are only drawn in the single-model case 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 figure as it will be submitted, rather than the palette it was built from. `palette_safety()` can promise that the eight colours depictr ships stay apart under colour-vision deficiency, but it knows nothing about the figure in front of you: how many of those colours it uses, what you replaced them with, how small the text will be once the figure is squeezed into a journal column, or whether the only thing separating two groups is their colour. `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
)
```

Arguments

`plot`	A plot, as returned by any depictr plotting function, including one extended afterwards with +. A multi-panel composite is refused: its panels have their own scales, themes and text, and one table of numbers cannot describe them all. Check each panel on its own.
`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 CIE76 colour difference, used for the colour and greyscale separability checks. Defaults to 5, matching `palette_safety()`.
`min_text_pt`	The smallest acceptable printed text size, in points. Defaults to 6, a common publisher floor for figure text.

Details

Every row carries the value it measured next to the threshold it was measured against, so a verdict can be argued with rather than merely accepted. A check passes when the measured value is at least the threshold. A check that has nothing to measure, such as colour separability on a figure that encodes nothing by colour, reports `NA` and a verdict of `"not applicable"` instead of a free 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.

What each check measures

`colour_separability` is the smallest CIE76 colour difference (Delta-E) between any two of the figure's encoding colours, and 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 a layer uses to tell groups apart; a continuous colour or fill scale is a smooth ramp rather than a set of codes, so it is excluded.

`greyscale_separability` is the smallest difference in CIE lightness between those same colours, which is what survives printing in black and white.

`text_size` is the smallest point size of any text the figure draws, after scaling by `width_cm / render_width_cm`: a figure drawn seven inches wide and printed in an 8.9 cm column has every point size halved. Text drawn inside the panel, by a layer such as `ggplot2::geom_text()` or by an annotation, is deliberately left out. It sits on the marks rather than on the background, so there is no one background to measure its contrast against, and the two engines size a layer's text in different units, which would leave the check disagreeing with its Python twin on the same figure.

`text_contrast` and `geometry_contrast` are the smallest WCAG 2.x contrast ratios of, respectively, any drawn text against the plot background and any encoding colour against the panel background.

`redundant_encoding` counts how many of shape and line type also vary in a layer whose colour varies. Zero means the distinction between groups is carried by colour alone, which is the single most common way an otherwise careful figure becomes unreadable.

A limitation of the default palette

The eight-colour qualitative palette clears the colour-separability checks comfortably and fails `greyscale_separability`: its orange (`#e69f00`) and sky blue (`#56b4e9`) differ by only 0.79 in lightness, so they print as the same grey. The colourblind-safety guarantee depictr makes is about hue confusion, and it was never a claim about greyscale. 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 four leading colours of the palette are also not safe in greyscale, since the bluish green and the vermillion differ by 3.55. The threshold has been left where it is rather than moved to let the package's own defaults through.

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 (2023). "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: two well-separated colours, a redundant
# shape, and text left at the size it was drawn.
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)
```

---

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 have a real survival difference: the `treatment` arm has a lower hazard, so the Kaplan-Meier curves separate clearly and the log-rank test is highly significant, and - because its event times are longer - the treatment arm is more often right-censored by the 36-month study end. `age` and `biomarker` also affect the hazard. The data feed `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 (about a 10% base rate) that is predictable from `arm`, `age` and `biomarker`. It gives the precision-recall (`pr_curve_plot()`), cumulative-gains (`gain_plot()`), lift (`lift_plot()`) and calibration (`calibration_plot()`) demos an imbalanced target, where those charts are most informative.

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed; they do not describe any real individuals.

Source

Simulated; see `data-raw/generate_datasets.R`.

`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. Projecting onto principal components means the plot works for any number of variables.

Usage

```
cluster_plot(
  data,
  cols = NULL,
  k = 3,
  clusters = NULL,
  scale = TRUE,
  suggest_k = NULL,
  k_range = 2:8,
  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 (default: all numeric).
k	Number of clusters for k-means (ignored if clusters is supplied, or overridden when suggest_k chooses a value).
clusters	Optional vector of cluster assignments (e.g. from stats::kmeans() or stats::cutree()); use this to plot a clustering you computed yourself. 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 (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 instead of using the supplied k. Either TRUE (uses 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.
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). Default NULL leaves the RNG state untouched.
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 ("forest") plot of model estimates. The input can be a fitted model (anything `tidy_estimates()` understands) 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 = 0,
  point_size = 2.2,
  line_size = 0.7,
  facet = FALSE,
  scales = c("fixed", "free"),
  standardise = FALSE,
  title = NULL,
  subtitle = NULL,
  x_lab = NULL
)
```

Arguments

`x`	A fitted model or a tidy data frame of estimates.
`conf_level`	Confidence/credible level, passed to `tidy_estimates()` when `x` is a model.
`intercept`	Whether to keep the intercept term. Defaults to `FALSE`, since the intercept is seldom of interest on a forest plot and its scale often overwhelms the other terms.
`order`	Order the terms by estimate: `"none"` (keep input order), `"ascending"` or `"descending"`.
`labels`	Optional display labels for the terms. Either a character vector the same length as the number of terms (in plotting order) or a named vector mapping raw term names to labels. If `NULL`, names are tidied with `format_terms()`.
`interaction`	Passed to `format_terms()` to control how interaction terms are rendered (ignored when `labels` is supplied).
`point_colour, reference_colour`	Colours for the estimates and the reference line.

`reference_line` Position of a vertical reference line (e.g. `0` for differences, `1` for odds/risk ratios). 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 removes the squish that occurs when terms live on very different scales (for example a large intercept alongside small slopes). Defaults to `FALSE`, preserving the shared-axis layout. A convenience alias for `scales = "free"`.

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

`standardise` Whether to standardise the coefficients by multiplying each by the standard deviation of its predictor column, putting them on a common scale so their magnitudes are comparable (and removing the empty band that otherwise appears when predictors are on very different scales). Requires a fitted model (ignored, with a warning, for a tidy data frame). Defaults to `FALSE`.

`title, subtitle, x_lab`
Plot title, subtitle and x-axis label. `x_lab` defaults to "Estimate", or "Estimate per SD of predictor" when `standardise`. 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.

Value

A ggplot2::ggplot object.

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 large term squishes the rest, give each term its own
# free-scaled panel:
coefficient_plot(fit, intercept = TRUE, facet = TRUE)

# Or put the coefficients on a common, comparable scale:
coefficient_plot(fit, standardise = 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 is the general engine behind `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 = 0,
  palette = NULL,
  point_size = 2.2,
  line_size = 0.7,
  facet = FALSE,
  scales = c("fixed", "free"),
  legend_title = "Source",
  legend_ncol = 1,
  title = NULL,
  subtitle = NULL,
  x_lab = "Estimate"
)
```

Arguments

`...`	Two or more fitted models and/or tidy data frames of estimates. Name the arguments to label the sources (e.g. `compare_models(Frequentist = m1, Bayesian = m2)`).
`names`	Optional character vector of source labels, overriding the names of `...`.
`conf_level`	Confidence/credible level for models.
`intercept`	Whether to keep the intercept term. Defaults to `FALSE`.
`order`	Order terms by their average estimate across sources: `"none"`, `"ascending"` or `"descending"`.
`labels`	Optional display labels (see `coefficient_plot()`).
`interaction`	Passed to `format_terms()`.
`dodge_width`	Vertical spacing between sources sharing a term.
`reference_line`	Position of a vertical reference line (`NA` to omit).
`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 (for example a large intercept alongside small slopes) stay legible. The source overlay and its single legend are preserved within the faceted layout. Defaults to `FALSE`. A convenience alias for `scales = "free"`.

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

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

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

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 alongside the slopes:
compare_models(`All fields` = m1, `Standard only` = m2,
               intercept = TRUE, facet = 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.

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 actually 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. 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.
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 (near-)zero variance cannot be correlated and are dropped automatically with an informative message, so the raw "the standard deviation is zero" warning from stats::cor() is not surfaced. If, after dropping them, any cells still come out NA (e.g. two variables that never co-occur under "pairwise.complete.obs"), those cells are rendered in grey and labelled n/a rather than 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. Used for the regression, coefficient-plot, scatter-trend and interaction examples. The data-generating process contains a genuine 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 real, diverging slopes rather than 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; see `data-raw/generate_datasets.R`.

`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,
  confidence = FALSE,
  level = 0.95,
  title = NULL
)
```

Arguments

x	A ts object, or a numeric vector (then frequency is required).
frequency	Number of observations per period (e.g. 12 for monthly data); taken from x when it is a ts.
method	"stl" (loess-based) or "classical" (stats::decompose()).
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 bleeds less into the trend and seasonal components. 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). FALSE reproduces the previous behaviour exactly.
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 conveys the scale of the unexplained variation around the smoothed trend rather than a formal sampling-distribution interval.

Value

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

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)
```

`dendrogram_plot` *Dendrogram*

Description

Hierarchical-clustering dendrogram, drawn with ggplot2 so it shares the package theme. Accepts a data frame (the distance matrix and clustering are computed for you), a stats::dist object, or an stats::hclust object. Optionally cut 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 (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 suppressed for larger ones, where they would otherwise collapse into an unreadable smear. `TRUE`/`FALSE` force them on or off. When labels are hidden and `k` is set, the cluster membership is instead conveyed by a coloured strip of leaf ticks along the bottom of the tree.
`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 the colour accessors `depictr_brand()`, `depictr_accent()` and `depictr_reference()`.

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 coincides with the default palette's first colour but does not alter a palette; use palette for that. Returned by depictr_brand().
accent	A secondary highlight colour. Returned by depictr_accent().
reference	The colour used for reference / annotation lines. Returned by depictr_reference().
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 built-in Okabe-Ito set. NULL restores the built-in 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)
```

---

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 widely recommended categorical palette that stays distinguishable under the common forms of colour-vision deficiency, with the depictr brand blue leading. The sequential and diverging palettes are perceptually ordered single-hue and red-blue ramps.

Usage

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

Arguments

n	Number of colours to return. If NULL (the default) the full qualitative palette is returned. 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).

Details

The guarantee belongs to the eight Okabe-Ito colours themselves, not to any number of colours: past eight the qualitative palette has to interpolate between them, and the interpolated colours sit close enough together to fail the package's own colour-distance check. That case warns rather than handing back colours under a guarantee it cannot keep.

The qualitative palette can be overridden globally with options(depictr.palette =) (see depictr_options()); when set, that custom palette replaces the built-in Okabe-Ito set 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.

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.

Examples

```
depictr_palette(3)
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 connecting segment, so the size and direction of the gap is read at a glance. It is a clearer alternative to paired or grouped bars for before/after, two-condition or two-period comparisons.

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); 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).
`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, which the default gap sort with the shortest dumbbell on top usually leaves clear) over a translucent background, instead of in a right-hand margin. 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 bin-width choice and makes quantiles, medians and group shifts easy to 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.
reference_quantiles	Optional numeric vector of probabilities in [0, 1] to mark with light horizontal guides (e.g. 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), draw the legend inside the panel (in the bottom-right corner the ECDF leaves empty once it saturates) over a translucent background, instead of 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, the most frequent level for factors). A confidence band (numeric predictor) or confidence intervals (factor predictor) convey 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 (lm, glm, merMod, ...).
predictor	Name of the focal predictor (string).
conf_level	Confidence level for the interval.
n	Number of points across the range of a numeric predictor.

`rug`	Whether to add a rug of the observed predictor values (numeric predictors).
`colour`	Colour for the line/points and band. Defaults to the depictr brand blue.
`title`, `x_lab`, `y_lab`	Title and axis labels.

Details

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. Mixed models fitted with `lme4::lmer()`/`lme4::glmer()` are supported too: predictions use only the fixed effects (`re.form = NA`) and standard errors come from the fixed-effect design matrix and `vcov()`. Works with `lm`, `glm` and `merMod`; other model classes are attempted on a best-effort basis.

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 *Gardner-Altman / Cumming estimation plot*

Description

An estimation plot puts the *effect size* and its uncertainty at the centre of the comparison, rather than a p-value. The upper panel shows each group's raw data (jittered) with its mean and confidence interval; the lower panel shows the pairwise mean difference(s) against a reference group, each with a bootstrap confidence interval. The two panels share an aligned outcome axis and are stacked with 'patchwork', so a difference of zero in the lower panel lines up with the reference group's mean above it.

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 sits at a difference of zero in the lower panel.
`conf_level`	Confidence level for both the group intervals (t-based) and the bootstrap difference intervals.
`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 both the two-group and multi-group cases.
`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

With exactly two groups this is the classic *Gardner-Altman* two-group plot (Gardner & Altman, 1986; Ho et al., 2019): a single mean difference is shown with its bootstrap interval, annotated with a standardised effect size (Cohen's *d* or, by default, the small-sample corrected Hedges' *g*; Hedges, 1981). With more than two groups it becomes a *Cumming* estimation plot (Cumming, 2012): every other group is compared with the reference group, each difference carrying its own bootstrap interval.

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; set a seed beforehand for reproducibility.

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 computed differences and their bootstrap intervals are attached as the `"differences"` attribute (a data frame).

References

Cumming, G. (2012). *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis*. Routledge.

Gardner, M. J., & Altman, D. G. (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, L. V. (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/s41592019-04703

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 vs 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 as box plots of the numeric variable by level; and two categorical variables 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 the numeric-numeric case (passed to `ggplot2::geom_smooth()`); `NULL` for no trend.
`palette`	Colours used when a fill/colour is needed; defaults to `depictr_palette()`.
`title, x_lab, y_lab`	Title and axis labels (default to variable names).

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.
`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 (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 instead.

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. A quick 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/fill.
type	One of "histogram", "density" or "both".
bins	Number of histogram bins.
alpha	Fill transparency (useful when groups overlap).
position	Histogram position adjustment, e.g. "identity", "stack" or "dodge". The default (NULL) chooses "dodge" when group is set (so overlapping bars stay readable) and "identity" otherwise.
palette	Colours for the groups; defaults to depictr_palette().
facet	When a group is given, draw one panel per group instead of overlaying them. This is much clearer than an overlay once there are more than two or three groups (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), draw the colour legend inside the panel (in the top-right corner a unimodal histogram/density leaves empty) over a translucent background, instead of in a right-hand margin. Defaults to FALSE.
title, x_lab	Plot title and x-axis label (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 colour everything 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. If `NULL`, all numeric columns are used (up to `max_cols`).
`group`	Optional grouping variable mapped to colour.
`max_cols`	Safety cap on the number of variables (a k-by-k matrix grows quickly).
`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

Cleans up the term names produced by modelling functions so that they read well on a plot: the intercept is renamed, an optional `b_`/`bs_` Bayesian prefix is stripped, interaction colons are converted to a chosen symbol, and underscores are shown as spaces (e.g. `word_frequency` becomes `"word frequency"`). The `b_`/`bs_` prefix is stripped from *each* component of an interaction (e.g. `b_x:b_y`), not just the leading term. Missing values (`NA`) are kept as `NA` rather than being rendered as the literal 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` (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, with the two sources distinguished by the first two colours of the colourblind-safe `depictr_palette()` (brand blue and orange).

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",
  ...
)
```

Arguments

`frequentist`	A frequentist model (e.g. from `lm`, `glm` or `lmerTest::lmer`) or a tidy data frame of estimates.
`bayesian`	A Bayesian model, a `posterior` draws object, a draws matrix/data frame, or a tidy data frame of posterior summaries.
`conf_level`	Confidence/credible level for models.
`labels`, `interaction`, `intercept`	See `compare_models()`. `intercept` defaults to `TRUE` here, matching the original behaviour.

`facet`, `scales` Layout controls. Because a Bayesian model almost always carries a large intercept alongside 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 the classic single shared-axis plot.

`note_frequentist_no_prior`
If `TRUE`, append "(no prior)" to the frequentist legend label, which is helpful 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.

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

Details

This is the modernised successor to the original `frequentist_bayesian_plot()` gist, which built on `brms::mcmc_plot()` to show the full Bayesian posterior with the frequentist estimate overlaid. That namesake behaviour is restored here: when `bayesian` carries posterior *draws* (a `brms`/`rstanarm` fit, a `posterior` draws object, a draws matrix, or a long/wide draws data frame), the full posterior *distribution* is drawn per term (a 'ggdist' half-eye) and the frequentist point and confidence interval is overlaid at the same position. When `bayesian` is only a tidy table of posterior *summaries* (columns such as `term`, `estimate`, `conf.low`/`conf.high`, or the `Estimate`, `l-95% CI`, `u-95% CI` of `brms::fixef()`), the function shows the familiar two-source forest plot via `compare_models()`.

Terms are aligned by their canonical display label, so the `brms`-style `b_` prefix is reconciled automatically against the frequentist term names.

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 toward zero and narrows its interval; a real posterior would
# come from the model fit rather than 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 + CI 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 / (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 (0/1, logical or a two-level factor with the positive class second); or a *named* list of models / (actual, score) pairs to overlay.
`score`	When `x` is an outcome vector, the matching scores or predicted probabilities (or a list of them for the multi-model case).
`colour`	Curve colour for the single-model case. Defaults to the depictr brand blue. Ignored when several models are overlaid.
`legend_inside`	When `TRUE` (and several models are overlaid), draw the legend inside the panel (in the bottom-right triangle the concave curve leaves empty) over a translucent background, instead of 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, over a backdrop of the raw (jittered) data. By showing both the estimate and its uncertainty, it conveys 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).
conf_level	Confidence level for the intervals (t-based).

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 difference(s) against a reference group, each with a bootstrap confidence interval, turning the plot into a full estimation plot via estimation_plot(). The return value is then a 'patchwork' object. Defaults to FALSE (the plain group-means plot, fully backward-compatible).
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.

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 (`lm`, `glm`, `merMod`, ...).
`predictor`	Name of the focal predictor on the x-axis (string).
`moderator`	Name of the moderating predictor, mapped to colour (string).
`moderator_values`	For a numeric moderator, the values to show. Defaults to the 10th, 50th and 90th percentiles.
`conf_level`	Confidence level for the bands/intervals.
`n`	Number of points across the range of a numeric focal predictor.
`band`	Whether to draw confidence bands (numeric focal predictor).
`palette`	Colours for the moderator; defaults to `depictr_palette()`.
`title`, `x_lab`, `y_lab`	Title and axis labels.

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 total within-cluster sum of squares "elbow" (the point of maximum curvature), and the gap statistic (the smallest `k` whose gap is within one standard error of the next, using the Tibshirani et al. heuristic).

Usage

```
k_diagnostic(
  data,
  k_range = 2:8,
  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. 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 (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.

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, P. J. (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*, 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/unrelated priming) and `modality` (visual/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 makes this a clean fit for crossed-random-effects models such as `lmer(RT ~ condition + modality + word_frequency + (1 | participant) + (1 | item))`, and hence for `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 a touch slower; 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 - it is not derived from `RT` - so classification demos built on it are not circular.

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed; they do not describe any real individuals.

Source

Simulated; see `data-raw/generate_datasets.R`.

`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 / (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 (0/1, logical or a two-level factor with the positive class second); or a *named* list of models / (actual, score) pairs to overlay.
`score`	When `x` is an outcome vector, the matching scores or predicted probabilities (or a list of them for the multi-model case).
`colour`	Curve colour for the single-model case. Defaults to the depictr brand blue. Ignored when several models are overlaid.
`legend_inside`	When `TRUE` (and several models are overlaid), draw the legend inside the panel (in the top-right corner, which the decaying lift curve leaves empty) over a translucent background, instead of 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, making it easy to spot variables and patterns that need attention before modelling.

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 (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 for present cells and the colourblind-safe `depictr_palette()` accent for missing cells.
`legend_inside`	When `TRUE` (and `sort = TRUE`), draw the legend inside the panel, in the top-right (where the most-complete columns put a solid "Present" block, so it hides no "Missing" mark) instead of 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 degrees of freedom, AIC, BIC, the log-likelihood, an R-squared (ordinary for `lm` and McFadden's pseudo-R-squared for `glm`) and the root-mean-square error.

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.

Value

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

References

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 residuals-against-fitted plot and a normal Q-Q plot, with a subtitle of key fit statistics. It is a convenience wrapper that composes several depictr plots with `arrange_plots()`, and serves well for a rapid model review or a report appendix.

Usage

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

Arguments

model	A fitted lm or glm model.
predictor	Focal predictor for the effect panel. If NULL, the first numeric predictor (or, failing that, the first predictor) is used.
standardise	Whether the coefficient panel shows standardised coefficients (each scaled by its predictor's standard deviation). Defaults to TRUE, which keeps the panel readable in this compact overview by putting predictors on a common scale; 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.

Value

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

Examples

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

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

---

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 carrying a linear trend, a clear twelve-month seasonal cycle (with different seasonal peaks per line) and noise. The two-series shape exercises the multi-group timeseries_plot() path on first-party data; extracting one series as a monthly ts (frequency 12) lets decompose_plot() and acf_plot() reveal 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; see `data-raw/generate_datasets.R`.

`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 offers a quick check that a model has settled on a stable solution: tight clusters of points indicate agreement between optimisers, whereas scatter signals 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	Either the object returned by lme4::allFit(), or a data frame with one row per optimiser-by-term combination (columns such as optimizer, term and value/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, e.g. c(conditionunrelated = "Unrelated priming"). When x is a raw lme4::allFit() object, names are prettified to the effect (variable) name automatically (e.g. conditionunrelated to condition) and these labels override that default.
number_optimizers	Whether to prefix each optimiser name with a number, so that the legend doubles 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 / violin plot highlighting outliers*

Description

Shows the distribution of a numeric variable (optionally by group) as a box and/or violin plot, 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.
`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. It is useful when choosing how many groups to show, selecting a palette type, or documenting a figure. Each swatch's hex label is drawn in near-black or white, whichever is more legible against that tile (chosen by the tile's relative luminance), so labels stay readable on both light and dark colours.

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.
`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, so you can see whether two groups would still be told apart by a colourblind reader.

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.

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 CIE L*ab** and the smallest pairwise colour difference (CIE76 Delta-E) is found. 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)
```

Arguments

`colours`	The palette to test. Defaults to the depictr qualitative palette. Needs at least two colours: a pairwise distance over fewer than two has no value, rather than an infinitely safe one.
`threshold`	The smallest acceptable Delta-E.

Details

The default `threshold` of 5 is calibrated against the reference colourblind-safe palette: the Okabe-Ito set's tightest pair (reddish purple against grey) sits at Delta-E 7.4 under full deuteranopia, so the cut must lie below that to pass the recommended 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; 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 Delta-E 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.

Examples

```
palette_safety()

# Two colours a hair apart are flagged.
palette_safety(c("#005b96", "#015c97"))
```

`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. Optionally colour the observations 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 (default: all numeric).
`group`	Optional grouping variable (a column name when `x` is a data frame, or a vector the length of the data) mapped to colour.
`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.

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 (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, rather than a point and two limits alone.

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/rstanarm), a posterior draws object, a matrix, or a long/wide data frame (see Details).
style	One of "halfeye" (density slab + interval, the default), "interval" (point and two nested intervals, no slab), "gradient" (a colour-graded interval) or "dots" (a quantile dotplot). Unknown values and a missing 'ggdist' fall back to "interval".
point	Central summary: "median" or "mean".
widths	Two interval widths (inner and outer), as probabilities. 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, e.g. c(conditionunrelated = "condition"). Unmatched parameters fall back to format_terms().
reference_line	Position of a vertical reference line, or NULL/NA to omit it. There is no universally meaningful reference for every parameter, so this defaults to 0 (the usual "no effect" line 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 the small parameters legible when a large one (typically the intercept) would otherwise stretch the shared axis and squish the rest. Defaults to FALSE. A convenience alias for scales = "free".
scales	Either "fixed" (the default, a single shared x-axis) or "free" (one free-scaled panel per parameter). When facet = TRUE this is forced to "free".
colour, fill	Colours for the point/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) auto-captions the interval mass; 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 (a parameter column and a value column). For fitted models only the fixed-effect (population) parameters are kept, with the `brms` `b_` prefix stripped.

The slab styles use the 'ggdist' package. When 'ggdist' is not installed the function falls back to the point-and-nested-interval display (equivalent to `style = "interval"`) and emits 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 dwarfs 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, allowing the plot to 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")
)
```

Arguments

x	A powerCurve object from 'simr', or a data frame with a sample size column (nlevels, n or x), a power column (mean or power), and optional lower/upper confidence limits.
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.

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")
```

---

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 / (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 (0/1, logical or a two-level factor with the positive class second); or a *named* list of models / (actual, score) pairs to overlay.
score	When x is an outcome vector, the matching scores or predicted probabilities (or a list of them for the multi-model case).
colour	Curve colour for the single-model case. Defaults to the depictr brand blue. Ignored when several models are overlaid.
f1	Logical; if TRUE, mark the maximum-F1 operating point on each curve (the threshold maximising the harmonic mean of precision and recall).
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).

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 that stray outside the band are unusual under normality; under a normal sample roughly `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`/`glm` model (its 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/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.

Value

A ggplot2::ggplot object.

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" combines three views of a distribution: a half-violin density (the cloud), a narrow boxplot, and the raw jittered points (the rain). It conveys the shape, the summary and the individual observations together, giving a fuller and more transparent picture than a boxplot alone. 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.
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 ("caterpillar") plot. It is the usual way to inspect by-group departures from the average, and to identify unusual groups.

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/group, estimate, and either conf.low/conf.high or std.error. An optional term column facets the plot.
conf_level	Confidence level when intervals are derived from standard errors.
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 are made GLM-aware (`glm_panels = TRUE`): the "residuals vs. fitted" panel 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 regardless of the response family. The scale-location and leverage panels are unchanged. For an `lm` the behaviour is identical to before. Set `glm_panels = FALSE` to force 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, making shifts in location and shape easy to follow, and is a clearer choice 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); one ridge per level.
`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; 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 implementation is base R + 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 (single-model) curve 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 (0/1, logical or a two-level factor with the positive class second); or a *named* list of models / (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 named/positional list of score vectors.
`colour`	Curve colour for the single-model case. Defaults to the depictr brand blue. Ignored when several models are overlaid (the colourblind-aware `scale_colour_depictr()` palette is used instead).
`ci`	Bootstrap confidence band for a single ROC curve and its AUC. `FALSE` (default) draws none; `TRUE` uses 2000 resamples; a positive integer sets the number of resamples. Ignored when several models are overlaid.
`conf_level`	Confidence level for the bootstrap band.
`youden`	Logical; if `TRUE`, mark the Youden's J operating point (the threshold maximising sensitivity + specificity - 1) on each curve.
`legend_inside`	When `TRUE` (and several models are overlaid), draw the legend inside the panel (in the bottom-right corner the curve leaves empty) over a translucent background, instead of in a right-hand margin. Defaults to `FALSE`.
`title`	Plot title.

Details

A named list overlays one curve per element, e.g. `roc_curve_plot(list("Full" = fit_full, "Reduced" = fit_reduced))`. Each element may be a `glm`, a length-2 list/data frame of `(actual, score)`, or an outcome vector paired with the matching element of a `score` list. Single-model calls are unchanged.

Value

A ggplot2::ggplot object. The AUC(s) are stored in `attr(plot, "auc")` (a named vector 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 convenience wrapper around `ggplot2::ggsave()` with sensible defaults for figures in papers and reports: a moderate size, 300 dpi, and the output directory created if needed. The device is inferred from the file extension.

Usage

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

Arguments

`filename`	Output file path. The extension sets the device (e.g. `.png`, `.pdf`, `.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.
`...`	Passed to `ggplot2::ggsave()`.

Value

The `filename`, invisibly.

Examples

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

`scale_colour_depictr` *depictr colour and fill scales*

Description

Discrete ggplot2 scales using `depictr_palette()`. These are the canonical colour and fill scales used throughout the package. They honour the global `options(depictr.palette = )` (via `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"),
  ...
)

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

scale_fill_depictr(
  n = NULL,
```

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

Arguments

n	Optional number of colours to draw from depictr_palette(). By default ggplot2 requests exactly as many colours as there are groups; pass n only to force a fixed slice of the palette. More than eight groups means the built-in qualitative palette is interpolated, which loses the colour-vision-deficiency guarantee and warns at draw time; 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).
...	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()
```

---

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.
method	Smoothing method passed to ggplot2::geom_smooth(), e.g. "lm", "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 (default to 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 (e.g. one panel per calendar month for monthly data), and within each panel the value is drawn across successive cycles (e.g. across years). A horizontal reference line in every panel marks that season's mean. This makes the seasonal pattern (differences *between* panels) and the within-season trend over time (the slope *inside* each panel) visible 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).
frequency	Number of observations per period (e.g. 12 for monthly data); taken from x when it is a ts.
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 (panel/axis) labels, e.g. month.abb. When NULL (default) sensible labels are inferred (month abbreviations for frequency 12, quarter labels for frequency 4, otherwise the season index).
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 sequential ramp from depictr_palette().
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 is handy for spotting an unusual year.

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; negative suggests 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 (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"`.

References

Rousseeuw, P. J. (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. Intermediate values interpolate the transform towards the identity, an approximation to Machado et al.'s per-severity matrices.

Details

This is the simulator behind `palette_preview()`'s `cvd` argument, behind `palette_safety()`, and behind the colour-separability rows of `check_figure()`.

Value

A character vector of lower-case hex colours, the same length as `colours`.

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)
```

`summary_table` *A "Table 1" style descriptive summary*

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`	Columns to summarise. If `NULL`, all columns except `group` are used, with high-cardinality identifier-like columns skipped (see Details).
`group`	Optional grouping variable; 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/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 suppress these.

When vars is NULL, high-cardinality character/factor columns (those whose number of distinct levels approaches the number of rows, e.g. identifier columns) are skipped automatically, with a message, rather than being expanded into hundreds of one-per-row entries. Pass such a column explicitly via 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"
)
```

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 one is an error rather than a silent exclusion, since dropping it would move the number at risk unannounced.
status	Event indicator when time is a vector. Either the 0/1 convention (0 = censored, 1 = event) or the survival::Surv() 1/2 convention (1 = censored, 2 = event) is accepted; logical values are also allowed. Other codings raise an error. 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). Observations whose group is missing are dropped with a message, since they belong to no arm.
conf_level	Confidence level for the limits (NA to omit them).
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), draw the group legend inside the panel (in the bottom-left corner a monotone-decreasing survival curve always leaves empty) over a translucent background, instead of in a right-hand margin. Defaults to FALSE.
title, x_lab, y_lab	Title and axis labels.

Details

For publication-ready figures the plot offers the three annotations that define a "survminer-style" Kaplan-Meier display, each behind its own argument and off by default so existing behaviour is unchanged:

- **Number-at-risk table** (risk_table = TRUE): a small panel composed beneath the curves with 'patchwork', giving the number of subjects still at risk in each group at the x-axis breaks.
- **Median-survival guides** (median_line = TRUE): dashed reference lines from the 0.5 survival level down to each group's median survival time, with the median value labelled. Groups whose curve never reaches 0.5 (median not estimable) are skipped.

- **Log-rank test** (`logrank = TRUE`): for two or more groups, the chi-squared log-rank statistic and its p-value are added as a subtitle. `survival::survdiff()` is used when the 'survival' package is installed, otherwise an equivalent base-R log-rank test is computed from the risk/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). *The natural duration of cancer*, volume 33 of *Reports on Public Health and Medical Subjects*. His Majesty's Stationery Office, London.

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; passing the arguments explicitly overrides them. The title colour is the resolved depictr_brand(), which in turn honours options(depictr.brand =).

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. This is the natural companion to the ROC and precision-recall curves for *choosing* an operating point: it shows directly how each metric trades off as the cut-off moves, and (by default) marks the Youden's J and maximum-F1 optimal 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 (0/1, 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 dashed vertical line: any subset of "youden" (max sensitivity + specificity - 1) and "f1" (max F1). 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 rather than a coarse grid.

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)

# Sensitivity / specificity trade-off only, no 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`. It is the common currency used by `coefficient_plot()` and `compare_models()`, but is useful on its own.

Usage

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

Arguments

`x`	A fitted model or a data frame of results.
`conf_level`	Confidence (or credible) level for the interval.
`...`	Passed to methods (and to `broom::tidy()` in the fallback). The `merMod` method additionally accepts `effects`, which currently only supports `"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 computed with the normal approximation (estimate +/- z * standard error) for `glm` and `merMod` objects, which is fast and dependency-free; `lm` objects use the exact t-based interval. Supply a data frame with your own intervals (for example profiled or posterior intervals) to override this.

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)

# 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 a data frame with a time column, a value column and an optional grouping column.

Usage

```
timeseries_plot(
  x,
  time = NULL,
  value = NULL,
  group = NULL,
  rolling = NULL,
  forecast = NULL,
  frequency = NULL,
  level = 0.95,
  palette = NULL,
  point = FALSE,
  title = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`x`	A `ts` object, a numeric vector, or a data frame.
`time`	When `x` is a data frame, the time column (string or unquoted name); when `x` is a numeric vector, an optional vector of times.
`value`	When `x` is a data frame, the value column.
`group`	When `x` is a data frame, an optional grouping column mapped to colour.
`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.
`forecast`	Optional forecast overlay for a single (non-grouped) series. Either an integer horizon (number of future steps) to forecast with the built-in STL + seasonal-naive-with-drift method (see `ts_forecast()`), or a pre-computed forecast supplied as a data frame with columns `time`, `fit` and (optionally) `lwr`/`upr`, for

	instance from a fitted `forecast::forecast()` object. The point forecast continues the line and a shaded prediction-interval ribbon, which widens with the horizon, is drawn behind it.
`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, e.g. `0.95`).
`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.

See Also

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

Examples

```
timeseries_plot(AirPassengers, rolling = 12,
                title = "Air passengers", y_lab = "Passengers")
# 24-month forecast with a 90% prediction interval
timeseries_plot(AirPassengers, forecast = 24, level = 0.9)
```

---

ts_forecast *Forecast a seasonal series with STL plus seasonal-naive drift*

Description

A lightweight, dependency-free forecaster for a single seasonal series. The series is decomposed with `stats::stl()` into trend, seasonal and remainder. The trend is extrapolated linearly from its last `frequency` fitted values (the recent local slope), the seasonal pattern is carried forward by repeating the last full cycle of the seasonal component (a seasonal-naive forecast), and the point forecast is their sum.

Usage

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

Arguments

`x`	A `ts` object, or a numeric vector (then `frequency` is required).
`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).

Details

Prediction intervals are a random-walk-style normal band: the one-step standard deviation is estimated from the STL remainder and the interval at horizon `h` is `fit +/- z * sigma * sqrt(h)`, so the band necessarily widens with the horizon. This mirrors how seasonal-naive forecast variance accumulates over time and gives an honest, monotonically growing uncertainty band without pulling in a heavy modelling dependency. For a fully specified statistical model use, for example, `forecast::forecast()` and pass the resulting fit/interval columns to `timeseries_plot()` directly.

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`.

See Also

`timeseries_plot()`

Examples

```
fc <- ts_forecast(AirPassengers, h = 12)
head(fc)
# Interval width grows with the horizon
diff(fc$upr - fc$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 from base R (no 'car' dependency).

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 / 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/polynomial bases) the function reports the generalised VIF of Fox and Monette (1992): with R the correlation matrix of the (centred) predictor columns, R_{11} the block for the term and R_{22} the block for the remaining columns,

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

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 $\sqrt{\mathrm{threshold}}$ accordingly. The x-axis is kept tight to the data: when every bar is comfortably below the threshold the line is reported in the caption rather than drawn into a wide empty band.

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))

# Multi-level factors get a single generalised VIF per term
fit2 <- lm(yield ~ rainfall + fertiliser + treatment, data = crop_yield)
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` responds meaningfully to its predictors: higher stress lowers it, more sleep and exercise raise it, log-income has a positive effect of about +0.4 per natural-log unit, and age has a mild inverted-U (peaking 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; contains missing values (more at high stress).

stress Self-reported stress (1-7).

sleep_hours Typical hours of sleep per night; contains missing values.

exercise_days Days of exercise per week (0-7); contains missing values.

life_satisfaction Life satisfaction (1-7); contains missing values.

Details

The data are synthetic, generated by `data-raw/generate_datasets.R` with a fixed seed; they do not describe any real individuals.

Source

Simulated; see `data-raw/generate_datasets.R`.

Index

∗ **datasets**
clinical_trial, 10
crop_yield, 18
lexical_decision, 43
monthly_sales, 48
wellbeing_survey, 85

acf_plot, 4
acf_plot(), *48*
arrange_plots, 5
arrange_plots(), *47*

base::strwrap(), *35*
binned_residual_plot, 6
binned_residual_plot(), *64*
broom::tidy(), *81*

calibration_plot, 7
calibration_plot(), *11*
check_figure, 8
check_figure(), *53*, *74*
clinical_trial, 10
cluster::silhouette(), *73*
cluster_plot, 11
coefficient_plot, 13
coefficient_plot(), *15*, *44*, *81*
compare_models, 14
compare_models(), *36, 37*, *81*
confusion_matrix_plot, 16
correlation_heatmap, 17
correlation_heatmap(), *12*, *34*, *43*
crop_yield, 18

decompose_plot, 19
decompose_plot(), *48*, *72*, *83*
dendrogram_plot, 21
dendrogram_plot(), *12*
depictr_options, 22
depictr_options(), *24*, *68*, *79*
depictr_palette, 23
depictr_palette(), *12*, *15*, *18*, *22, 23*, *25*, *27*, *29*, *31–34*, *36*, *40*, *42*, *46*, *50–52*, *55*, *61*, *65*, *68–70*, *72, 73*, *77*, *83, 84*
dumbbell_plot, 25

ecdf_plot, 26
effects_plot, 27
estimation_plot, 28
estimation_plot(), *40*
explore_bivariate, 30
explore_categorical, 31
explore_distribution, 32
explore_distribution(), *32*
explore_pairs, 34

format_terms, 35
format_terms(), *13*, *15*, *50*, *56*, *58*
frequentist_bayesian_plot, 36
frequentist_bayesian_plot(), *15*

gain_plot, 38
gain_plot(), *11*
ggplot2::discrete_scale(), *69*
ggplot2::geom_smooth(), *31*, *70*
ggplot2::geom_text(), *9*
ggplot2::ggplot, *4*, *6, 7*, *12*, *14*, *16–18*, *20*, *22*, *26–28*, *30–33*, *35*, *37, 38*, *40–43*, *45, 46*, *48*, *50–52*, *55*, *57–59*, *61, 62*, *64, 65*, *67*, *70–73*, *78*, *80*, *83*, *85*
ggplot2::ggsave(), *67, 68*
ggplot2::theme_minimal(), *78*
grDevices::col2rgb(), *74*
group_comparison_plot, 39

influence_plot, 40
interaction_plot, 41

k_diagnostic, 42
k_diagnostic(), *12*
knitr::kable(), *75*

lexical_decision, 43
lift_plot, 45
lift_plot(), *11*
lme4::allFit(), *49, 50*
lme4::glmer(), *28*
lme4::lmer(), *28*

missingness_map, 46
model_fit_table, 47
model_report, 47
monthly_sales, 48

optimizer_fixef_plot, 49
optimizer_fixef_plot(), *44*
outlier_plot, 51

palette_preview, 52
palette_preview(), *10*, *74*
palette_safety, 53
palette_safety(), *8*, *10*, *74*
patchwork::plot_layout(), *29*
patchwork::wrap_plots(), *5*
pca_plot, 54
posterior_plot, 55
power_curve_plot, 57
pr_curve_plot, 58
pr_curve_plot(), *11*

qq_plot, 60

raincloud_plot, 61
random_effects_plot, 62
residual_diagnostics_plot, 63
ridgeline_plot, 64
roc_curve_plot, 66
roc_curve_plot(), *17*

save_plot, 67
save_plot(), *8*
scale_color_depictr
*(*scale_colour_depictr*)*, 68
scale_colour_depictr, 68
scale_colour_depictr(), *22, 23*, *66*
scale_fill_depictr
*(*scale_colour_depictr*)*, 68
scale_fill_depictr(), *23*
scatter_trend, 69
scree_plot, 70
seasonal_plot, 71
seasonal_plot(), *20*, *83*

silhouette_plot, 73
silhouette_plot(), *12*
simulate_cvd, 74
simulate_cvd(), *9*
stats::acf(), *4*
stats::cor(), *18*, *34*
stats::cutree(), *12*
stats::decompose(), *20*
stats::density(), *65*
stats::dist, *21*, *73*
stats::dist(), *21*, *73*
stats::hclust, *21*
stats::hclust(), *21*
stats::kmeans(), *12*, *43*
stats::pacf(), *4*
stats::prcomp(), *54*, *71*
stats::predict(), *28*
stats::stl(), *20*, *83*
stats::time(), *84*
summary_table, 75
survival::Surv(), *77*
survival::survdiff(), *78*
survival_plot, 76
survival_plot(), *10*

theme_depictr, 78
theme_depictr(), *22, 23*
threshold_plot, 79
tidy_estimates, 81
tidy_estimates(), *13*
timeseries_plot, 82
timeseries_plot(), *48*, *72*, *84*
ts_forecast, 83
ts_forecast(), *20*, *82, 83*

vif_plot, 84

wellbeing_survey, 85