Package 'BasicStatsPlots'

September 23, 2026

Type Package

Title Publication-Ready Data Visualization and Simple Statistical Inference

Version 0.1.0

Description Creates publication-ready data visualizations using 'ggplot2', together with tools for common simple statistical analyses (confidence intervals, ANOVA, linear and generalized regression). The package provides common chart types and analyses with built-in validation, customization options, and publication-friendly themes.

License GPL-3

Encoding UTF-8

Imports broom, broom.helpers (>= 1.20.0), car, checkmate, dplyr, ggplot2 (>= 4.0.0), ggridges, gt, gtsummary, lmtest, MASS, nortest, patchwork, pROC, RColorBrewer, rlang, scales, tibble, tidyr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Quentin Ricros [aut], Théo Roussel [aut], Cédrik Carrière [aut, cre], Bouchra Nasri [aut]

Maintainer Cédrik Carrière <carced2009@live.ca>

Repository CRAN

Date/Publication 2026-09-23 03:30:13 UTC

Contents

`BasicStatsPlots-package`

BasicStatsPlots: Publication-Ready Data Visualization and Simple Statistical Inference

---

Description

Creates publication-ready data visualizations using 'ggplot2', together with tools for common simple statistical analyses (confidence intervals, ANOVA, linear and generalized regression). The package provides common chart types and analyses with built-in validation, customization options, and publication-friendly themes.

Author(s)

Maintainer: Cédrik Carrière <carced2009@live.ca>

Authors:

- Cédrik Carrière <carced2009@live.ca>
- Quentin Ricros
- Théo Roussel
- Bouchra Nasri

References

The following references are drawn from course notes for MSO6616 (Gestion et analyse de données de recherche) (Université de Montréal).

Davies, T. M. (2016). The Book of R: A first course in programming and statistics. No Starch Press, San Francisco. ISBN 978-1-59327-651-5.

Wickham, H. (n.d.). ggplot2: Elegant graphics for data analysis. Tidyverse. https://ggplot2.tidyverse.org

Holtz, Y. (n.d.). Privacy Raptor. The R Graph Gallery. https://r-graph-gallery.com/privacy_raptor.html

vanova *ANOVA*

Description

Run a one-way or two-way analysis of variance (ANOVA) on a numerical variable across the levels of one or two categorical variables, and produce a full report: a descriptive statistics table, the ANOVA table itself, group mean estimates with confidence intervals, and residual diagnostic plots checking two of the classic ANOVA assumptions — normality of residuals (Shapiro-Wilk) and homogeneity of variances across groups (Levene). When the overall test is significant, Tukey HSD post-hoc pairwise comparisons are added automatically, colored by whether each comparison remains significant after correction (red) or not (black) — independently of the palette used for the marginal mean plot, to avoid the two plots being read as encoding the same thing.

Usage

```
vanova(
  data,
  x,
  y,
  x2 = NULL,
  risk = 0.05,
  alpha = 0.6,
  palette = "Spectral",
  digits = 3,
  p_digits = 3,
  scientific = FALSE
)
```

Arguments

`data`	Dataframe
`x`	Categorical variable
`y`	Numerical variable
`x2`	Categorical variable
`risk`	Type I error
`alpha`	Opacity
`palette`	Brewer color palette name, or a custom vector of colors (hex codes or R color names)
`digits`	Number of digits to display for means, standard deviations, sums of squares, and other numeric statistics (default 3)
`p_digits`	Number of digits to display for p-values (default 3); values smaller than `10^-p_digits` are shown as e.g. `"<0.001"` unless `scientific = TRUE`
`scientific`	Whether to display numeric values (means, standard deviations, statistics, p-values) in scientific notation instead of fixed-decimal notation (default `FALSE`)

Details

Unlike the v* descriptive plotting functions, vanova does not return a single ggplot object. Because it produces several complementary results at once (tables and plots), it returns an object of class "vanova_result" with a dedicated print method: calling vanova() at the console (or explicitly printing the returned object) displays the descriptive table, the mean plot, the residual diagnostic plots, the ANOVA table, and (when applicable) the Tukey post-hoc table and plot; assigning the result to a variable does not display anything, so the individual components can be extracted silently (e.g. result$anova_table). The marginal mean plot and the Tukey post-hoc plot are shown as separate figures (not combined), since they show different things — group estimates versus pairwise comparisons. When the overall test is not significant, no post-hoc comparison is computed and a note explaining why is shown instead of an empty table/plot.

The normality and homogeneity-of-variance checks cover two of the three classic ANOVA assumptions. The third — independence of observations — cannot be verified from the data itself: it depends on how the data was collected (e.g. no repeated measurements on the same subject, no clustering), and needs to be established from that design rather than from a plot or test.

Value

An object of class "vanova_result" (a list) with the descriptive table, mean plot, residual diagnostic plots, ANOVA table, and (when the overall test is significant) the Tukey post-hoc table and plot. It has a print method that displays all tables and plots; this only happens when the result is auto-printed or explicitly passed to print(), not when it is assigned to a variable.

Examples

```
# Basic ANOVA
vanova(
  data = PlantGrowth,
  x = "group",
  y = "weight"
)

# One-way ANOVA
vanova(
  data = PlantGrowth,
  x = "group",
  y = "weight",
  risk = 0.01,
  alpha = 0.8,
  palette = "Set2"
)

# Two-way ANOVA
vanova(
  data = warpbreaks,
  x = "wool",
  y = "breaks",
  x2 = "tension",
  risk = 0.05,
  alpha = 0.8,
```

```
  palette = "RdBu"
)

# ANOVA with custom colors
vanova(
  data = PlantGrowth,
  x = "group",
  y = "weight",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)

# Custom digit precision and scientific notation
vanova(
  data = PlantGrowth,
  x = "group",
  y = "weight",
  digits = 4,
  p_digits = 4,
  scientific = TRUE
)
```

`varea_chart` *Area Chart*

Description

Create an area chart to visualize how a numerical variable evolves across an ordered continuous variable, typically time, with the area beneath the line filled in to emphasize magnitude and cumulative volume rather than just the trend itself. Area charts are especially effective for time series data, and the underlying curve can optionally be smoothed via cubic spline interpolation for a cleaner visual read.

Usage

```
varea_chart(
  data,
  x,
  y,
  alpha = 0.6,
  smooth = FALSE,
  smooth_n = 300,
  title = paste(y, "per", x),
  xlab = x,
  ylab = y,
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Ordered continuous variable
y	Numerical variable
alpha	Opacity
smooth	Logical. If TRUE, the area curve is smoothed via cubic spline interpolation instead of straight segments between points.
smooth_n	Integer. Number of interpolated points used when smooth = TRUE. Higher values produce a smoother curve. Ignored when smooth = FALSE.
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette

Value

ggplot object

Examples

```
# Basic area chart
varea_chart(
  data = ggplot2::economics,
  x = "date",
  y = "unemploy"
)

# Smoothed area chart
varea_chart(
  data = ggplot2::economics,
  x = "date",
  y = "unemploy",
  smooth = TRUE,
  smooth_n = 500
)

# Customized area chart
varea_chart(
  data = ggplot2::economics,
  x = "date",
  y = "unemploy",
  alpha = 0.8,
  title = "US unemployment over time",
  xlab = "Date",
  ylab = "Number of unemployed people",
  palette = "Pastel1"
)
```

vbarplot *Barplot*

Description

Create a barplot from a dataset.

Usage

```
vbarplot(
  data,
  x,
  y = NULL,
  group = NULL,
  position = "dodge",
  order = "none",
  error_bars = FALSE,
  value_labels = FALSE,
  alpha = 0.6,
  title = paste("Barplot of", x),
  xlab = x,
  ylab = "Count",
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Categorical variable
y	Numerical variable
group	Grouping variable
position	Bar position ("dodge" / "stack" / "fill" / "identity")
order	Category ordering ("none" / "asc" / "desc")
error_bars	Show error bars (mean +/- standard deviation) (TRUE / FALSE). Only valid when y is provided (not applicable to count barplots). Most meaningful when position = "dodge".
value_labels	Show value labels above/on the bars (TRUE / FALSE)
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic barplot
vbarplot(
  data = iris,
  x = "Species"
)

# Customized barplot
vbarplot(
  data = CO2,
  x = "Treatment",
  y = "uptake",
  group = "Type",
  position = "stack",
  order = "desc",
  alpha = 0.8,
  title = "Uptake by treatment and type",
  xlab = "Treatment",
  ylab = "Uptake",
  palette = "Spectral"
)

# Barplot with error bars and value labels
vbarplot(
  data = CO2,
  x = "Treatment",
  y = "uptake",
  group = "Type",
  position = "dodge",
  error_bars = TRUE,
  value_labels = TRUE,
  title = "Mean uptake (+/- SD) by treatment and type",
  ylab = "Uptake"
)

# Barplot with custom colors
vbarplot(
  data = iris,
  x = "Species",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

---

vboxplot *Boxplot*

Description

Create a boxplot to summarize and compare the distribution of a numerical variable across categories. Each box shows the median and interquartile range (IQR) of the data, the whiskers extend to the most extreme values within 1.5 times the IQR, and points beyond that range are shown individually as potential outliers. Optionally, the underlying data points, the group mean, and the sample size per group can be layered on top for a fuller picture of each distribution.

Usage

```
vboxplot(
  data,
  x,
  y,
  jitter = FALSE,
  jitter_width = 0.2,
  jitter_size = 0.9,
  average = FALSE,
  varwidth = FALSE,
  size = FALSE,
  alpha = 0.6,
  title = paste(y, "per", x),
  xlab = x,
  ylab = y,
  palette = "Spectral"
)
```

Arguments

`data`	Dataframe
`x`	Categorical variable
`y`	Numerical variable
`jitter`	Show data points (TRUE / FALSE)
`jitter_width`	Amount of horizontal jitter applied to the data points (only used when `jitter = TRUE`)
`jitter_size`	Size of the jittered data points (only used when `jitter = TRUE`)
`average`	Show average point (TRUE / FALSE)
`varwidth`	Vary width by size (TRUE / FALSE)
`size`	Show sample size (n=) per group (TRUE / FALSE)
`alpha`	Opacity
`title`	Plot title
`xlab`	X-axis label
`ylab`	Y-axis label
`palette`	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic boxplot
vboxplot(
  data = iris,
  x = "Species",
  y = "Sepal.Length"
)

# Customized boxplot
vboxplot(
  data = iris,
  x = "Species",
  y = "Petal.Length",
  jitter = TRUE,
  jitter_width = 0.15,
  jitter_size = 1.2,
  average = TRUE,
  varwidth = TRUE,
  size = TRUE,
  alpha = 0.8,
  title = "Petal length by species",
  xlab = "Species",
  ylab = "Petal Length",
  palette = "Set2"
)

# Boxplot with custom colors
vboxplot(
  data = iris,
  x = "Species",
  y = "Sepal.Length",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

---

vconf_interval *Confidence Interval*

Description

Create a confidence interval plot to show the precision of an estimate for each category of a grouping variable, with error bars marking the interval bounds at the chosen confidence level.

Usage

```
vconf_interval(
  data,
  x,
  y = NULL,
  value = NULL,
  expected = NULL,
  risk = 0.05,
  alpha = 0.6,
  title = "",
  xlab = x,
  ylab = "",
  palette = "Spectral"
)
```

Arguments

`data`	Dataframe
`x`	Categorical variable
`y`	Variable
`value`	Success value
`expected`	Expected proportion
`risk`	Type I error
`alpha`	Opacity
`title`	Plot title
`xlab`	X-axis label
`ylab`	Y-axis label
`palette`	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Details

The interval type is selected automatically from the arguments supplied. Set `y` to a numerical column for a confidence interval on the mean (t-interval). Set `y` to a categorical column together with `value` for a confidence interval on the proportion of `value` within `y` (Wilson-corrected proportion test). Leave `y` `NULL` and supply `expected` for a goodness-of-fit comparison of observed versus expected category proportions.

Value

ggplot object

Examples

```
# Basic confidence interval
vconf_interval(
  data = iris,
  x = "Species",
  y = "Petal.Length"
)

# Mean confidence interval
vconf_interval(
  data = iris,
  x = "Species",
  y = "Petal.Length",
  risk = 0.01,
  alpha = 0.8,
  title = "Mean of petal length by species",
  xlab = "Species",
  ylab = "Petal Length",
  palette = "Pastel1"
)

# Proportion confidence interval
vconf_interval(
  data = warpbreaks,
  x = "wool",
  y = "tension",
  value = "L",
  risk = 0.01,
  alpha = 0.7,
  title = "Proportion of low tension by wool type",
  xlab = "Wool type",
  ylab = "Low tension proportion",
  palette = "Spectral"
)

# Fit confidence interval
vconf_interval(
  data = iris,
  x = "Species",
  expected = c(1/3, 1/3, 1/3),
  risk = 0.01,
  alpha = 0.9,
  title = "Observed vs expected species proportions",
  xlab = "Species",
  ylab = "Proportion",
  palette = "RdBu"
)

# Confidence interval with custom colors
vconf_interval(
  data = iris,
  x = "Species",
```

```
  y = "Petal.Length",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

---

vdensity_plot *Density Plot*

Description

Create a density plot to visualize the underlying probability density of a continuous numerical variable, estimated via kernel density estimation. Density plots offer a smoothed alternative to histograms for comparing the shape, center, and spread of one or more distributions. A second variable can be mirrored below the axis to directly compare two distributions on the same scale.

Usage

```
vdensity_plot(
  data,
  x,
  x2 = NULL,
  group = NULL,
  facet = FALSE,
  center_line = "none",
  rug = FALSE,
  alpha = 0.6,
  title = paste("Density of", x),
  xlab = x,
  ylab = "Density",
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Numerical variable
x2	Mirror numerical variable
group	Grouping variable
facet	Add facet (TRUE / FALSE)
center_line	Show a vertical reference line at the mean or median of x ("none" / "mean" / "median")
rug	Show a rug plot (tick marks for each observation of x) (TRUE / FALSE)
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic density plot
vdensity_plot(
  data = iris,
  x = "Sepal.Length"
)

# Mirror density plot
vdensity_plot(
  data = iris,
  x = "Sepal.Length",
  x2 = "Petal.Length",
  title = "Mirror density plot",
  xlab = "Value",
  ylab = "Density"
  )

# Customized density plot
vdensity_plot(
  data = iris,
  x = "Petal.Length",
  group = "Species",
  facet = TRUE,
  center_line = "median",
  rug = TRUE,
  alpha = 0.8,
  title = "Petal length distribution by species",
  xlab = "Petal Length",
  ylab = "Density",
  palette = "Set2"
  )

# Density plot with custom colors
vdensity_plot(
  data = iris,
  x = "Sepal.Length",
  group = "Species",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

---

vgen_reg *Generalized Regression*

Description

Fit a generalized linear model — Poisson, negative binomial, or logistic regression — from a model formula, and produce a diagnostic report matched to the chosen family: a histogram of the response and the fitted regression curve (for Poisson and negative binomial counts), or a confusion matrix and ROC curve (for logistic regression), together with a coefficient table and a model diagnostic panel (Cook's distance, residuals, and variance inflation factors).

Usage

```
vgen_reg(
  data,
  formula,
  family,
  risk = 0.05,
  alpha = 0.6,
  palette = "Spectral",
  bins = NULL,
  p_value = TRUE,
  digits = 3,
  p_digits = 3,
  scientific = FALSE,
  strat_var = NULL
)
```

Arguments

`data`	Dataframe
`formula`	A model formula, e.g. `y ~ x1 + x2` or `y ~ x1 * x2`, optionally with an exposure offset (e.g. `y ~ x1 + offset(log(exposure))`). The response (left-hand side) must be a single numerical column of `data`.
`family`	Model family ("poisson" / "negbin" / "logit")
`risk`	Type I error
`alpha`	Opacity
`palette`	Brewer color palette name, or a custom vector of colors (hex codes or R color names)
`bins`	Number of bins for the response histogram (Poisson/negative binomial only). Defaults to `NULL`, which uses a bin width of 1 - appropriate since the response in a count regression is integer-valued - instead of ggplot2's own default of 30 bins (which triggers a "Pick better value" message and can look arbitrary on count data). Set to a positive integer to choose the number of bins yourself.
`p_value`	Whether to include the p-value column in the coefficient table (default `TRUE`)
`digits`	Number of digits to display for coefficients, confidence intervals, and the dispersion parameter (default 3)
`p_digits`	Number of digits to display for p-values (default 3); values smaller than `10^-p_digits` are shown as e.g. `"<0.001"` unless `scientific = TRUE`

`scientific` Whether to display numeric values (coefficients, confidence intervals, dispersion, p-values) in scientific notation instead of fixed-decimal notation (default `FALSE`)

`strat_var` Optional name of a categorical column of `data` used to stratify the descriptive table into side-by-side columns (e.g. one column per group). It does not need to be one of the variables in `formula`, and no comparison test is run between strata — it is a description, not a hypothesis test.

Details

Unlike the `v*` descriptive plotting functions, `vgen_reg` does not return a single ggplot object. Because it produces several complementary results at once (plots and a coefficient table), it returns an object of class `"vgen_reg_result"` with a dedicated `print` method: calling `vgen_reg()` at the console (or explicitly printing the returned object) displays all of them; assigning the result to a variable does not display anything, so the individual components can be extracted silently (e.g. `result$regression_table`).

A descriptive table is always produced first, summarizing every variable used in `formula` (predictors and response) : counts and percentages for categorical variables, mean and standard deviation for numeric ones. Passing `strat_var` splits this table into one column per level of that variable, which does not need to appear in `formula` itself ; no comparison test is run between the resulting columns, it is a description only.

`formula` is passed on to `glm()` (or `MASS::glm.nb()` for `family = "negbin"`) as-is, so any structure they accept is supported — additive (`y ~ x1 + x2`), interaction (`y ~ x1 * x2` or `y ~ x1 + x2 + x1:x2`), and so on. The left-hand side must be a single, untransformed variable (a bare column name, not e.g. `log(y) ~ x`), and `.` is not supported on the right-hand side — explanatory variables must be listed explicitly.

The coefficient table is always exponentiated : for `family = "logit"` it reports Odds Ratios (OR). For `family = "poisson"` or `"negbin"` it reports exp(β) rather than a specific named ratio, because what exp(β) represents depends on the study design : it is an Incidence Rate Ratio (IRR) when modeling event counts together with an offset for exposure/time, a Risk Ratio / Relative Risk (RR) when Poisson regression is used as a working model for a binary or bounded-count outcome, or simply the multiplicative change in the expected count otherwise. `vgen_reg` has no way to infer which of these applies, so it reports the unambiguous exp(β) and leaves the interpretation and naming to you. For `family = "negbin"`, the fitted dispersion parameter (θ) and its confidence interval are also reported in the table's source note.

If observations were not all observed over the same unit of exposure, add an exposure offset directly in `formula` with `offset()`, e.g. `y ~ x1 + x2 + offset(log(exposure))` ; the offset variable is excluded from the explanatory variables used for the plot/coloring logic described below.

The fitted regression curve (Poisson/negative binomial) or the colored point cloud (all families) depends on how many explanatory variables are in `formula` and what type they are: with a single numerical variable, a fitted curve is drawn; with two variables where the second is categorical, one colored point cloud and fitted curve per category is drawn; with two numerical variables, a scatter plot colored by the second variable on a continuous gradient is drawn (no fitted curve, since a two-predictor fit is a surface and cannot be represented as a single 2D curve); with more than two explanatory variables, no scatter/curve plot is shown. For `family = "poisson"` or `"negbin"`, a histogram of the response variable is shown alongside the regression plot. For `family = "logit"`,

a confusion matrix (accuracy, sensitivity, specificity, at a 0.5 classification threshold) and a ROC curve are shown instead.

A model diagnostic panel is always produced, regardless of `family`: Cook's distance (influential observations), a raw residuals-versus-fitted plot (response residuals, i.e. $y - \hat{y}$; standardized residuals are not used here, since their usual interpretation assumes constant variance, which does not hold for Poisson, negative binomial, or logistic models), and, when `formula` has more than one explanatory variable, the variance inflation factors (VIF) for multicollinearity (VIF is not meaningful with a single predictor, so it is omitted in that case). When the regression plot described above is available, the panel is a 2x2 grid combining it with these three diagnostics; when it is not (more than two explanatory variables), the diagnostics alone are arranged two per row. As with `vlin_reg`, the VIF calculation always uses the additive main-effects model regardless of interactions requested in `formula`, since VIF is only meaningful for main effects.

Value

An object of class `"vgen_reg_result"` (a list) with the descriptive table (predictors and response, stratified by `strat_var` when supplied), the regression table, the Cook's distance plot, the residuals plot, and (when applicable) the VIF table/plot, together with, depending on `family`, either the response histogram and regression plot, or the confusion matrix and ROC curve. It has a `print` method that displays all plots and tables; this only happens when the result is auto-printed or explicitly passed to `print()`, not when it is assigned to a variable.

Examples

```
# Basic Poisson regression
vgen_reg(
  data = warpbreaks,
  formula = breaks ~ wool,
  family = "poisson"
)


# Customized Poisson regression
vgen_reg(
  data = warpbreaks,
  formula = breaks ~ tension + wool,
  family = "poisson",
  risk = 0.01,
  alpha = 0.8,
  palette = "Pastel1"
)

# Customized negative binomial regression
vgen_reg(
  data = warpbreaks,
  formula = breaks ~ tension + wool,
  family = "negbin",
  risk = 0.01,
  alpha = 0.8,
  palette = "Pastel1"
)
```

```
# Logistic regression : probability of low birth weight (MASS::birthwt),
# from mother's age and smoking status
bw <- transform(MASS::birthwt, smoke = factor(smoke, labels = c("Non-smoker", "Smoker")))
vgen_reg(
  data = bw,
  formula = low ~ age + smoke,
  family = "logit",
  risk = 0.01,
  alpha = 0.8,
  palette = "Pastel1"
)

# Poisson regression with an exposure offset (MASS::Insurance) :
# number of claims per age/district group, accounting for the number
# of policyholders in each group
vgen_reg(
  data = MASS::Insurance,
  formula = Claims ~ Age + District + offset(log(Holders)),
  family = "poisson",
  alpha = 0.8,
  palette = "Pastel1"
)

# Generalized regression with custom colors
vgen_reg(
  data = warpbreaks,
  formula = breaks ~ wool,
  family = "poisson",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)

# Poisson regression with a continuous predictor : seizure counts as
# a function of age (MASS::epil), with a custom histogram bin count
# and no p-value column
vgen_reg(
  data = MASS::epil,
  formula = y ~ age,
  family = "poisson",
  bins = 15,
  p_value = FALSE
)

# Custom digit precision and scientific notation
vgen_reg(
  data = warpbreaks,
  formula = breaks ~ wool,
  family = "poisson",
  digits = 4,
  p_digits = 4,
  scientific = TRUE
)
```

```
# Descriptive table stratified by a variable not in the formula
vgen_reg(
  data = warpbreaks,
  formula = breaks ~ wool,
  family = "poisson",
  strat_var = "tension"
)
```

---

vhistogram *Histogram*

Description

Create a histogram to visualize the distribution of a continuous numerical variable by dividing its range into bins and counting the number of observations that fall within each one. Histograms reveal the shape, center, spread, and skewness of a distribution, and make it easy to spot gaps, clusters, or unusual values. When a grouping variable is supplied, histograms are overlaid with transparency by default so that overlapping distributions remain visible through one another; an empirical density curve and reference lines for the mean or median can also be layered on top for additional context.

Usage

```
vhistogram(
  data,
  x,
  binwidth,
  group = NULL,
  position = "identity",
  facet = FALSE,
  density_overlay = FALSE,
  center_line = "none",
  alpha = 0.6,
  title = paste("Histogram of", x),
  xlab = x,
  ylab = "Frequency",
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Numerical variable
binwidth	Width of bins
group	Grouping variable

position	Bar position ("identity" / "stack" / "dodge" / "fill"). "identity" (the default) overlays the histograms directly on top of one another, which combined with alpha lets you see overlapping distributions through each other
facet	Add facet (TRUE / FALSE)
density_overlay	Show the histogram on a density scale and overlay the empirical density curve (TRUE / FALSE)
center_line	Show a vertical reference line at the mean or median of x ("none" / "mean" / "median")
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic histogram
vhistogram(
  data = iris,
  x = "Sepal.Length",
  binwidth = 0.5
)

# Overlaid histograms by group (default position = "identity"):
# each group's color remains visible through the others
vhistogram(
  data = iris,
  x = "Petal.Length",
  binwidth = 0.5,
  group = "Species",
  alpha = 0.5,
  title = "Petal length distribution by species",
  xlab = "Petal Length",
  palette = "Set2"
)

# Dodged (side-by-side) histogram instead of overlaid
vhistogram(
  data = iris,
  x = "Petal.Length",
  binwidth = 0.5,
  group = "Species",
  position = "dodge",
```

```
  facet = TRUE,
  alpha = 0.8,
  title = "Petal length distribution by species",
  xlab = "Petal Length",
  ylab = "Count",
  palette = "Set2"
)

# Histogram with density overlay and a median reference line
vhistogram(
  data = iris,
  x = "Petal.Length",
  binwidth = 0.5,
  density_overlay = TRUE,
  center_line = "median"
)

# Histogram with custom colors
vhistogram(
  data = iris,
  x = "Sepal.Length",
  binwidth = 0.5,
  group = "Species",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

`vline_chart` *Line Chart*

Description

Create a line chart to visualize how a numerical variable evolves across an ordered continuous variable, most commonly time. Line charts connect successive observations to reveal trends, trajectories, and patterns such as growth, decline, seasonality, or fluctuation, and make it easy to compare several trends at once when a grouping variable is supplied.

Usage

```
vline_chart(
  data,
  x,
  y,
  group = NULL,
  points = FALSE,
  point_size = 2,
  hline = NULL,
  alpha = 0.6,
  title = paste(y, "per", x),
  xlab = x,
```

```
  ylab = y,
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Ordered continuous variable
y	Numerical variable
group	Grouping variable
points	Show a point marker at each observation (TRUE / FALSE)
point_size	Size of the point markers (only used when points = TRUE)
hline	Draw a horizontal reference line at this y value (e.g. a target or threshold). Set to NULL to omit it
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic line chart
vline_chart(
  data = pressure,
  x = "temperature",
  y = "pressure"
)

# Customized line chart
vline_chart(
  data = Orange,
  x = "age",
  y = "circumference",
  group = "Tree",
  points = TRUE,
  point_size = 1.5,
  alpha = 0.8,
  title = "Tree circumference growth",
  xlab = "Age",
  ylab = "Circumference",
  palette = "Set2"
```

```
)

# Line chart with a reference line and custom colors
vline_chart(
  data = pressure,
  x = "temperature",
  y = "pressure",
  hline = 400,
  palette = c("#1b9e77", "#d95f02")
)
```

---

vlin_reg *Linear Regression*

Description

Fit a simple or multiple linear regression from a model formula, and produce a full diagnostic report: the fitted regression line or a color-coded scatter plot (depending on the number and type of explanatory variables), a coefficient table, model fit statistics, Cook's distance for influential observations, residual normality and homoscedasticity assessments, and — for multiple predictors — variance inflation factors (VIF) for multicollinearity together with an AIC-based stepwise selection table when the global model is significant.

Usage

```
vlin_reg(
  data,
  formula,
  risk = 0.05,
  alpha = 0.6,
  palette = "Spectral",
  p_value = TRUE,
  digits = 3,
  p_digits = 3,
  scientific = FALSE,
  strat_var = NULL
)
```

Arguments

data	Dataframe
formula	A model formula, e.g. y ~ x1 + x2 or y ~ x1 * x2. The response (left-hand side) must be a single numerical column of data.
risk	Type I error
alpha	Opacity
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

p_value	Whether to include the p-value column in the coefficient and stepwise tables (default TRUE)
digits	Number of digits to display for coefficients, confidence intervals, and other numeric statistics (default 3)
p_digits	Number of digits to display for p-values (default 3); values smaller than 10^-p_digits are shown as e.g. "<0.001" unless scientific = TRUE
scientific	Whether to display numeric values (coefficients, confidence intervals, statistics, p-values) in scientific notation instead of fixed-decimal notation (default FALSE)
strat_var	Optional name of a categorical column of data used to stratify the descriptive table into side-by-side columns (e.g. one column per group). It does not need to be one of the variables in formula, and no comparison test is run between strata — it is a description, not a hypothesis test.

Details

Unlike the v* descriptive plotting functions, vlin_reg does not return a single ggplot object. Because it produces several complementary diagnostics at once (tables and plots), it returns an object of class "vlin_reg_result" with a dedicated print method: calling vlin_reg() at the console (or explicitly printing the returned object) displays all of them; assigning the result to a variable does not display anything, so the individual components can be extracted silently (e.g. result$regression_table).

A descriptive table is always produced first, summarizing every variable used in formula (predictors and response) : counts and percentages for categorical variables, mean and standard deviation for numeric ones. Passing strat_var splits this table into one column per level of that variable, which does not need to appear in formula itself ; no comparison test is run between the resulting columns, it is a description only.

formula is passed on to lm() as-is, so any structure lm() accepts is supported — additive (y ~ x1 + x2), interaction (y ~ x1 * x2 or y ~ x1 + x2 + x1:x2), and so on. The left-hand side must be a single, untransformed variable (a bare column name, not e.g. log(y) ~ x), and . is not supported on the right-hand side — explanatory variables must be listed explicitly. Variance inflation factors and the stepwise selection table always use the additive main-effects model (y ~ x1 + x2 + ...) regardless of interactions requested in formula, since VIF is only meaningful for main effects.

The scatter/line plot depends on how many explanatory variables are in formula and what type they are: with a single numerical variable, a fitted regression line is drawn; with two variables where the second is categorical, one colored point cloud and fitted line per category is drawn; with two numerical variables, a scatter plot colored by the second variable on a continuous gradient is drawn (no fitted line, since a two-predictor fit is a plane and cannot be represented as a single 2D line); with more than two explanatory variables, no scatter/line plot is shown and only the tabular/model diagnostics are produced. When the scatter/line plot is available, it is combined with Cook's distance, the normality assessment, and the homoscedasticity assessment into a single 2x2 panel (as in vgen_reg); when it is not, these three diagnostics alone are arranged two per row. The VIF plot, when applicable, is always shown separately, since a fifth panel would not fit the 2x2 layout.

Value

An object of class "vlin_reg_result" (a list) with the descriptive table (predictors and response, stratified by strat_var when supplied), the Cook's distance plot, residual diagnostic plots, the

coefficient and model fit tables, and (when applicable) the VIF table/plot and the stepwise selection table. It has a `print` method that displays all tables and plots; this only happens when the result is auto-printed or explicitly passed to `print()`, not when it is assigned to a variable.

Examples

```
# Basic simple linear regression
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length
)


# Customized simple linear regression
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length,
  risk = 0.01,
  alpha = 0.8,
  palette = "Pastel1"
)

# Multiple linear regression with an interaction
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length * Species,
  risk = 0.01,
  alpha = 0.8,
  palette = "Pastel1"
)

# Multiple linear regression, additive, three predictors
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length + Petal.Width + Sepal.Width,
  risk = 0.01,
  alpha = 0.8,
  palette = "Pastel1"
)

# Linear regression with custom colors
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length,
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)

# Linear regression without the p-value column
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length,
  p_value = FALSE
)
```

```
# Custom digit precision and scientific notation
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length,
  digits = 4,
  p_digits = 4,
  scientific = TRUE
)

# Descriptive table stratified by a variable not in the formula
vlin_reg(
  data = iris,
  formula = Sepal.Length ~ Petal.Length,
  strat_var = "Species"
)
```

---

vlollipop_plot *Lollipop Plot*

Description

Create a lollipop plot to compare a numerical variable across categories, combining a thin segment with a point marker at its tip. It conveys the same information as a bar chart while drawing less visual weight to each value, which keeps the plot readable with many categories or when comparing two numerical variables per category side by side. A very common use case is plotting one lollipop per observation (e.g. an id, a row name, a subject) to spot the largest or most extreme values at a glance, such as Cook's distance per observation in a regression diagnostic plot.

Usage

```
vlollipop_plot(
  data,
  x,
  y,
  y2 = NULL,
  order = "none",
  flip = FALSE,
  center_line = "none",
  value_labels = FALSE,
  alpha = 0.6,
  title = paste(y, "per", x),
  xlab = x,
  ylab = y,
  palette = "Spectral"
)
```

Arguments

data	Data frame
x	Categorical variable
y	Numerical variable
y2	Second numerical variable
order	Category ordering ("none" / "asc" / "desc")
flip	Flip plot (TRUE / FALSE)
center_line	Show a reference line at the mean or median of y ("none" / "mean" / "median")
value_labels	Show a value label at the tip of each point (TRUE / FALSE)
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Details

Each value of x (and of y2, if used) is expected to appear only once in data — one row per category or per observation. If x contains repeated values, the points and segments for those rows will be drawn on top of one another at the same location. If your data has multiple rows per category, summarize it first (e.g. with aggregate() or dplyr::summarise()).

Value

ggplot object

Examples

```
# Basic lollipop plot on pre-summarized data (one value per category)
avg_length = aggregate(Sepal.Length ~ Species, data = iris, FUN = mean)
vlollipop_plot(
  data = avg_length,
  x = "Species",
  y = "Sepal.Length"
)

# Classic use case: one lollipop per observation, ordered and flipped,
# to spot the most extreme values - here, Cook's distance per car
# model from a regression model, used to identify influential points
model = lm(mpg ~ wt + hp, data = mtcars)
cooks_data = data.frame(
  car = rownames(mtcars),
  cooks_distance = cooks.distance(model)
)
vlollipop_plot(
```

```
  data = cooks_data,
  x = "car",
  y = "cooks_distance",
  order = "desc",
  flip = TRUE,
  center_line = "mean",
  value_labels = TRUE,
  title = "Cook's Distance by Car Model",
  xlab = "Car Model",
  ylab = "Cook's Distance"
)

# Double lollipop plot comparing two numerical variables per category
mtcars_data = data.frame(car = rownames(mtcars), mtcars)
vlollipop_plot(
  data = mtcars_data,
  x = "car",
  y = "mpg",
  y2 = "hp",
  order = "desc",
  flip = TRUE,
  title = "MPG and horsepower by car model",
  xlab = "Car",
  ylab = "Value",
  palette = "RdBu"
)

# Lollipop plot with custom colors
vlollipop_plot(
  data = avg_length,
  x = "Species",
  y = "Sepal.Length",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

---

vpie_chart *Pie Chart*

Description

Create a pie chart (or, optionally, a donut chart) to show how a numerical variable is distributed across the levels of a categorical variable, as proportions of a whole. Small categories can automatically be grouped into a single "Other" slice to keep the chart readable when there are many categories.

Usage

```
vpie_chart(
  data,
  x,
```

```
  y,
  label = FALSE,
  donut = FALSE,
  other_threshold = 0,
  alpha = 0.6,
  title = paste(y, "per", x),
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Categorical variable
y	Numerical variable
label	Show labels (TRUE / FALSE)
donut	Show a donut chart (with a hole in the middle) instead of a full pie chart (TRUE / FALSE)
other_threshold	Group categories whose share is below this fraction (between 0 and 1) into a single "Other" slice. Set to 0 (the default) to disable grouping
alpha	Opacity
title	Plot title
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic pie chart
vpie_chart(
  data = ToothGrowth,
  x = "supp",
  y = "len"
)

# Donut chart with labels
vpie_chart(
  data = ToothGrowth,
  x = "supp",
  y = "len",
  label = TRUE,
  donut = TRUE,
  alpha = 0.8,
  title = "Tooth growth by supplement",
  palette = "Spectral"
```

```
)

# Grouping small categories into "Other"
vpie_chart(
  data = data.frame(
    browser = c("Chrome", "Safari", "Edge", "Firefox", "Opera", "Other"),
    share = c(65, 18, 5, 3, 2, 1)
  ),
  x = "browser",
  y = "share",
  other_threshold = 0.05,
  label = TRUE
)

# Pie chart with custom colors
vpie_chart(
  data = ToothGrowth,
  x = "supp",
  y = "len",
  palette = c("#1b9e77", "#d95f02")
)
```

`vridgeline_chart` *Ridgeline Chart*

Description

Create a ridgeline chart (also known as a joyplot) to compare the distribution of a numerical variable across several categories, each shown as a stacked, partially overlapping density curve or histogram. Ridgeline charts make it easy to compare the shape, center, and spread of many distributions at once in a compact layout.

Usage

```
vridgeline_chart(
  data,
  x,
  y,
  type = "density",
  bins = 30,
  bandwidth = NULL,
  scale = 1,
  order = "none",
  quantile_lines = FALSE,
  quantiles = c(0.25, 0.5, 0.75),
  alpha = 0.6,
  title = paste(y, "per", x),
  xlab = x,
  ylab = y,
```

```
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Numerical variable
y	Categorical variable
type	Ridgeline type ("density" / "histogram")
bins	Number of bins (only used when type = "histogram")
bandwidth	Kernel bandwidth for the density estimate (only used when type = "density"). Set to NULL (the default) to let the bandwidth be computed automatically from the data
scale	Amount of overlap between adjacent ridgelines. A value of 1 means adjacent ridgelines just touch; values greater than 1 create increasing overlap
order	Category ordering by the median of x within each category ("none" / "asc" / "desc")
quantile_lines	Show quantile reference lines within each ridgeline (only used when type = "density") (TRUE / FALSE)
quantiles	Quantiles to show as reference lines when quantile_lines = TRUE (defaults to the quartiles: 0.25, 0.5, 0.75)
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic ridgeline chart
vridgeline_chart(
  data = iris,
  x = "Sepal.Length",
  y = "Species"
)

# Ridgeline chart ordered by median, with quartile lines
vridgeline_chart(
  data = iris,
  x = "Petal.Length",
  y = "Species",
```

```
  order = "asc",
  quantile_lines = TRUE,
  scale = 1.5,
  title = "Petal length distribution by species",
  xlab = "Petal Length",
  ylab = "Species"
)

# Histogram ridgeline
vridgeline_chart(
  data = iris,
  x = "Petal.Length",
  y = "Species",
  type = "histogram",
  bins = 15,
  alpha = 0.8,
  palette = "Set2"
)

# Ridgeline chart with custom colors
vridgeline_chart(
  data = iris,
  x = "Sepal.Length",
  y = "Species",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

`vscatterplot` *Scatterplot*

Description

Create a scatterplot to visualize the relationship between two numerical variables, with each point representing one observation. Scatterplots reveal patterns such as association, clustering, and outliers between the two variables, and can be further enriched by mapping additional variables to point color, shape, or size (bubble chart).

Usage

```
vscatterplot(
  data,
  x,
  y,
  color = NULL,
  shape = NULL,
  size_var = NULL,
  size_range = c(1, 6),
  point_size = 3,
  line = FALSE,
```

```
  label = FALSE,
  alpha = 0.6,
  title = paste(y, "per", x),
  xlab = x,
  ylab = y,
  palette = "Spectral"
)
```

Arguments

data	Dataframe
x	Numerical variable
y	Numerical variable
color	Color grouping variable
shape	Shape grouping variable
size_var	Numerical variable mapped to point size (bubble chart). Set to NULL (the default) to use a fixed point size
size_range	Output range (min, max) for point sizes when size_var is used
point_size	Fixed point size, used when size_var is NULL
line	Connect points (TRUE / FALSE)
label	Show row names (TRUE / FALSE)
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Value

ggplot object

Examples

```
# Basic scatterplot
vscatterplot(
  data = iris,
  x = "Sepal.Length",
  y = "Petal.Length"
)

# Bubble chart: point size mapped to a third numerical variable
vscatterplot(
  data = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
```

```
  size_var = "Petal.Width",
  color = "Species",
  alpha = 0.7,
  title = "Sepal length vs petal length, sized by petal width"
)

# Customized scatterplot
vscatterplot(
  data = iris,
  x = "Petal.Length",
  y = "Petal.Width",
  color = "Sepal.Length",
  shape = "Species",
  alpha = 0.8,
  title = "Pepal length vs petal width by sepal length and species",
  xlab = "Petal Length",
  ylab = "Petal Width",
  palette = "RdBu"
)

# Scatterplot with connecting lines, grouped by category
# (color = "Tree" is essential here: without grouping, the line would
# zigzag across all 5 trees' interleaved measurements)
vscatterplot(
  data = Orange,
  x = "age",
  y = "circumference",
  color = "Tree",
  line = TRUE,
  alpha = 0.8,
  title = "Tree growth over time",
  xlab = "Age",
  ylab = "Circumference",
  palette = "Dark2"
)

# Scatterplot with row-name labels (subset for readability)
vscatterplot(
  data = head(mtcars, 10),
  x = "wt",
  y = "mpg",
  label = TRUE,
  alpha = 0.8,
  title = "Fuel efficiency vs weight, by car model",
  xlab = "Weight (1000 lbs)",
  ylab = "Miles per gallon"
)

# Scatterplot with custom colors
vscatterplot(
  data = iris,
  x = "Sepal.Length",
  y = "Petal.Length",
```

```
  color = "Species",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

`vviolin_chart` *Violin Chart*

Description

Create a violin chart to compare the full distribution of a numerical variable across categories. Each violin mirrors a kernel density estimate on both sides of its axis, showing the shape of the distribution (e.g. multiple modes) in more detail than a boxplot alone. A boxplot, quantile lines, and the sample size per group can optionally be layered on top for additional context.

Usage

```
vviolin_chart(
  data,
  x = "key",
  y = "value",
  format = "long",
  order = "none",
  size = FALSE,
  flip = FALSE,
  boxplot = FALSE,
  quantile_lines = FALSE,
  quantiles = c(0.25, 0.5, 0.75),
  scale = "area",
  alpha = 0.6,
  title = paste(y, "per", x),
  xlab = x,
  ylab = y,
  palette = "Spectral"
)
```

Arguments

`data`	Dataframe
`x`	Categorical variable
`y`	Numerical variable
`format`	Data format ("long" / "wide")
`order`	Category ordering ("none" / "asc" / "desc")
`size`	Show sample size (n=) per group (TRUE / FALSE)
`flip`	Flip plot (TRUE / FALSE)
`boxplot`	Show boxplot (TRUE / FALSE)

quantile_lines	Show quantile reference lines within each violin (TRUE / FALSE)
quantiles	Quantiles to show as reference lines when quantile_lines = TRUE (defaults to the quartiles: 0.25, 0.5, 0.75)
scale	How violin widths are scaled relative to each other ("area": all violins have the same area: "count": widths are proportional to the number of observations; "width": all violins have the same maximum width)
alpha	Opacity
title	Plot title
xlab	X-axis label
ylab	Y-axis label
palette	Brewer color palette name, or a custom vector of colors (hex codes or R color names)

Details

Data can be supplied in two shapes. In "long" format (the default), x is a categorical column and y a numerical column already present in data. In "wide" format, each column of data is treated as its own category (e.g. several numerical variables to compare side by side); the data is pivoted internally and x/y are ignored.

Value

ggplot object

Examples

```
# Basic violin chart
vviolin_chart(
  data = iris,
  x = "Species",
  y = "Sepal.Length"
)

# Customized violin chart
vviolin_chart(
  data = iris,
  x = "Species",
  y = "Petal.Length",
  format = "long",
  order = "desc",
  size = TRUE,
  flip = TRUE,
  boxplot = TRUE,
  alpha = 0.8,
  title = "Petal length by species",
  xlab = "Species",
  ylab = "Petal Length",
  palette = "Set2"
)
```

```
# Violin chart with quantile lines and widths scaled by sample size
vviolin_chart(
  data = iris,
  x = "Species",
  y = "Sepal.Length",
  quantile_lines = TRUE,
  scale = "count"
)

# Wide format data
vviolin_chart(
  data = USJudgeRatings,
  format = "wide",
  order = "asc",
  size = FALSE,
  flip = FALSE,
  boxplot = TRUE,
  alpha = 0.8,
  title = "Distribution of judge ratings",
  xlab = "Criterion",
  ylab = "Score",
  palette = "Spectral"
)

# Violin chart with custom colors
vviolin_chart(
  data = iris,
  x = "Species",
  y = "Sepal.Length",
  palette = c("#1b9e77", "#d95f02", "#7570b3")
)
```

Index