

Package: Kifidi (via r-universe)

October 10, 2026

Type Package

Title Summary Table and Means Plots

Version 0.1.0

Maintainer Oswald Omuron <oswaldomuron@gmail.com>

Description Optimized for handling complex datasets in environmental and ecological research, this package offers functionality that is not fully met by general-purpose packages. It provides two key functions, 'summarize_data()', which summarizes datasets, and 'plot_means()', which creates plots with error bars. The 'plot_means()' function incorporates error bars by default, allowing quick visualization of uncertainties, crucial in ecological studies. It also streamlines workflows for grouped datasets (e.g., by species or treatment), making it particularly user-friendly and reducing the complexity and time required for data summarization and visualization.

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports MuMIn, car, lme4, nlme

Config/pak/sysreqs cmake make libicu-dev

Repository <https://oswaldomuron.r-universe.dev>

Date/Publication 2025-11-27 07:19:16 UTC

RemoteUrl <https://github.com/oswaldomuron/kifidi>

RemoteRef HEAD

RemoteSha 58cc179e0407bbf7167b77ac540051c8a89e7e68

Contents

Kifidi-package	2
cor_vif_table	3
counts	4
generate_random_points	5
plot_group_regressions	6
plot_lme_regressions	7
plot_lmm_regressions	10
plot_means	12
summarize_data	14
Index	16

Kifidi-package	<i>Kifidi: Tools for summarizing and analyzing grouped environmental data</i>
----------------	---

Description

The Kifidi package provides tools for summarizing and visualizing grouped numerical data, especially for environmental and ecological datasets. It includes functions for generating statistical summaries, plotting means with error bars, performing grouped regression analysis, and generating frequency counts by group.

Main Functions

- `summarize_data` Provides statistical summaries (mean, SD, N, etc.) of numeric data grouped by one or two categorical variables.
- `plot_means` Creates bar plots of means with optional error bars.
- `counts` Generates frequency tables or counts of observations by grouping variables.
- `plot_group_regressions` Performs and plots linear regressions grouped by a factor variable.
- `plot_lmm_regressions` Plots group-level and fixed-effect regression lines from a linear mixed-effects model in lme4 package with lmer().
- `plot_lme_regressions` Plots group-level lines from a linear mixed-effects model in nlme with lme().
- `generate_random_points` Generates random (x, y) sampling coordinates within a rectangular plot area and optionally exports them as CSV.

Author(s)

Oswald Omuron

cor_vif_table	<i>Correlation Matrix and Variance Inflation Factors (VIFs) for a Set of Variables</i>
---------------	--

Description

Computes the Pearson correlation matrix for a set of numeric variables and calculates Variance Inflation Factors (VIFs) to assess multicollinearity. All variables are included in the VIF calculation using a dummy response variable in an additive linear model.

Usage

```
cor_vif_table(data, vars)
```

Arguments

data	A data frame containing the variables of interest.
vars	A character vector specifying the names of the numeric variables to include. All specified variables must exist in data.

Details

- The correlation matrix shows pairwise linear associations between variables.
- VIFs are computed using a linear model with all variables as predictors and a dummy response.
- The VIF calculation assumes an **additive linear model**: each variable is included as a main effect only, and no interaction terms or higher-order terms are included.
- The function automatically removes rows with missing values (NA) in the selected variables.
- VIFs reflect multicollinearity of each variable with respect to all other variables in the set.

Value

A list with two elements:

correlations A numeric matrix of pairwise Pearson correlations among the selected variables.

VIF A data frame with columns Variable and GVIF, giving the variance inflation factor for each variable.

Examples

```
# Create example data frame
set.seed(123)
Z <- data.frame(
  L.AREA = rnorm(20, mean = 50, sd = 10),
  L.DIST = rnorm(20, mean = 30, sd = 5),
  L.LDIST = rnorm(20, mean = 15, sd = 3),
  YR.ISOL = rnorm(20, mean = 10, sd = 2),
  ALT = rnorm(20, mean = 100, sd = 20),
```

```

    GRAZE = rnorm(20, mean = 5, sd = 1)
  )

  # Select variables to analyze
  vars <- c("L.AREA", "L.DIST", "L.LDIST", "YR.ISOL", "ALT", "GRAZE")

  # Run the correlation and VIF function
  result <- cor_vif_table(Z, vars)

  # View the correlation matrix
  result$correlations

  # View the variance inflation factors (VIFs)
  result$VIF

```

 counts

Count Unique Groups in a Column

Description

This function calculates the frequency of each unique value in a given column of data, excluding NA values.

Usage

```
counts(column_data)
```

Arguments

column_data A vector of data (numeric, character, or factor) from which the unique groups and their frequencies are calculated.

Details

The function first removes any NA values from the input data and identifies the unique groups. It then counts the occurrences of each unique value using a loop and returns the results as a data frame with two columns: **group** (the unique values) and **counts** (their respective frequencies).

Value

A data frame with:

group The unique values from the input data.

counts The frequency of each unique value.

Note

This implementation uses a loop and may be slower for very large datasets. For faster performance, consider using `table()` or `dplyr::count()`.

Author(s)

Oswald Omuron

See Also[unique](#), [table](#), [count](#)**Examples**

```
data <- c("A", "B", "A", "C", "B", "B", NA, "A", "C")
result <- counts(data)
print(result)
```

generate_random_points*Generate Random Sampling Points in a Plot*

Description

This function generates random (x, y) coordinates within a rectangular plot area for biomass or other sampling. It can plot the points and optionally export them as CSV.

Usage

```
generate_random_points(
  plot_length = 3,
  plot_width = 3,
  n_points = 5,
  seed = NULL,
  export_csv = FALSE,
  filename = "random_coordinates.csv"
)
```

Arguments

plot_length	Numeric. Length of the plot in meters. Default is 3.
plot_width	Numeric. Width of the plot in meters. Default is 3.
n_points	Integer. Number of random points to generate. Default is 5.
seed	Integer or NULL. Seed for random number generator to reproduce results. Default is NULL (random every run).
export_csv	Logical. Whether to export the coordinates as a CSV file. Default is FALSE.
filename	Character. Name of the CSV file to export if export_csv = TRUE. Default is "random_coordinates.csv".

Value

A data.frame with columns: Point, X_meters, Y_meters.

Examples

```
# Generate 5 random points in a 3x3 m plot, plot and export csv
generate_random_points(plot_length = 3, plot_width = 3, n_points = 5, seed = 42,
                       export_csv = TRUE, filename = "points.csv")

# Generate random points without fixed seed (different each run)
generate_random_points(n_points = 10)
```

plot_group_regressions

Plot Group Regressions with Optional Grouping

Description

This function plots x vs y and fits linear models, either by group or for all data.

Usage

```
plot_group_regressions(
  x,
  y,
  group = NULL,
  colors = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  legend = TRUE,
  legend_position = "topright",
  return_models = FALSE,
  conf.int = FALSE,
  label_equations = FALSE,
  draw_lm = TRUE,
  add = FALSE,
  theme = "default",
  lty = 1,
  lwd = 2,
  pch = 16,
  ...
)
```

Arguments

x	A numeric vector for the x-axis.
y	A numeric vector for the y-axis.
group	Optional factor for grouping. If NULL, a single regression is drawn.
colors	Named vector of colors for groups or a vector matching number of groups.
main	Main title of the plot.
xlab	Label for x-axis.
ylab	Label for y-axis.
legend	Logical; whether to show the legend.
legend_position	Position of the legend (e.g., "topright").
return_models	Logical; return list of lm models.
conf.int	Logical; whether to draw confidence intervals.
label_equations	Logical; whether to label each group with its regression equation.
draw_lm	Logical; whether to draw the regression line(s).
add	Logical; whether to add to an existing plot.
theme	Plot theme (currently unused).
lty	Line type(s) for regression line.
lwd	Line width(s) for regression line.
pch	Plotting character(s) for points.
...	Additional plotting parameters passed to points().

Value

Optionally returns a list of lm models if return_models = TRUE.

plot_lme_regressions *Plot Linear Mixed-Effects (LME) Regressions*

Description

Fits a linear mixed-effects model using **nlme** and plots the observed data and regression lines for each group, including fixed and random effects. Optionally plots the overall fixed effect regression line and displays model statistics (R^2 values and AIC).

Usage

```

plot_lme_regressions(
  model_or_formula,
  random = NULL,
  data = NULL,
  legend = TRUE,
  legend_position = "right",
  inset = 0,
  return_model = FALSE,
  lty = NULL,
  pch = 16,
  lwd = 2,
  axes = TRUE,
  ann = TRUE,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  oma = c(0, 0, 0, 0),
  mar = c(5, 4, 4, 2),
  draw_fixed_effects = FALSE,
  fixed_col = "black",
  fixed_lty = 2,
  fixed_lwd = 3,
  fixed_confi = FALSE,
  ...
)

```

Arguments

<code>model_or_formula</code>	Either a fitted <code>nlme::lme</code> model or a formula specifying the fixed effects, e.g. <code>y ~ x</code> .
<code>random</code>	A random effects formula, e.g. <code>~ x group</code> . Required only if a formula (not a model) is supplied.
<code>data</code>	A data frame containing the variables in the model. Required only if a formula (not a model) is supplied.
<code>legend</code>	Logical, whether to display a legend (default = TRUE).
<code>legend_position</code>	Position of the legend ("right", "topright", etc.).
<code>inset</code>	Inset for the legend.
<code>return_model</code>	Logical, if TRUE returns the fitted model (default = FALSE).
<code>lty</code>	Line type for group-specific regression lines.
<code>pch</code>	Plotting character for data points.

lwd	Line width for group-specific regression lines (default = 2).
axes	Logical, whether to draw axes (default = TRUE).
ann	Logical, whether to include plot annotations (default = TRUE).
xlim, ylim	Axis limits for the plot.
main	Plot title.
xlab, ylab	Axis labels.
col	Colors for groups. Defaults to distinct colors for each group.
oma	Outer margin areas.
mar	Margins of the plot.
draw_fixed_effects	Logical, whether to plot the fixed-effect regression line (default = FALSE).
fixed_col	Color of the fixed-effect regression line (default = "black").
fixed_lty	Line type of the fixed-effect regression line (default = 2).
fixed_lwd	Line width of the fixed-effect regression line (default = 3).
...	Additional arguments passed to nlme::lme() or plot().

Details

The function automatically computes and plots regression lines for each grouping level based on both fixed and random effects. If `draw_fixed_effects = TRUE`, the overall fixed-effect regression line is drawn across the full x-range. The plot legend includes regression equations for each group and, optionally, the fixed effect line. Model performance metrics, including marginal R^2 (R^2_m), conditional R^2 (R^2_c), and AIC, are displayed in the legend panel.

Value

Invisibly returns NULL unless `return_model = TRUE`, in which case it returns the fitted `nlme::lme` model object.

Examples

```
## Not run:
library(nlme)
data(Orthodont)
plot_lme_regressions(distance ~ age, random = ~ age | Subject, data = Orthodont,
                     draw_fixed_effects = TRUE, fixed_col = "red")

## End(Not run)
```

plot_lmm_regressions *Plot Linear Mixed Model Regressions by Group*

Description

This function fits a linear mixed-effects model using `lmer()` and plots group-level regression lines and optionally the fixed effect regression line. It includes group-specific points and regression lines, and can display model statistics such as Nakagawa R^2 values and AIC.

Usage

```
plot_lmm_regressions(
  formula,
  data,
  colors = NULL,
  lty = 1,
  lwd = 2,
  pch = 16,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  draw_fixed_line = FALSE,
  draw_group_lines = TRUE,
  label_equations = FALSE,
  legend_position = "topright",
  inset = 0,
  xpd = TRUE,
  ann = TRUE,
  axes = TRUE,
  legend = TRUE,
  return_model = FALSE,
  mar = c(5, 4, 4, 15),
  oma = c(0, 0, 0, 4),
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

<code>formula</code>	A formula specifying the model (e.g., $y \sim x + (x \mid \text{group})$).
<code>data</code>	A data frame containing the variables in the model.
<code>colors</code>	A vector of colors for each group. Defaults to rainbow colors if not specified.
<code>lty</code>	Line type(s) for regression lines. Can be a single value or vector.
<code>lwd</code>	Line width(s) for regression lines. Can be a single value or vector.

plot_means

*Plot Mean Values with Error Bars by Group***Description**

Creates a bar plot of mean values from a summary data frame with optional error bars showing standard errors.

Usage

```
plot_means(
  summary_df,
  main_title = "Mean Values by Group",
  ylab = NULL,
  xlab = NULL,
  bar_color = "skyblue",
  error_bar_color = "red",
  bar_width = 0.7,
  error_bar_length = 0.1,
  axes = TRUE,
  space = NULL,
  density = NULL,
  angle = 45,
  col = NULL,
  names_arg = NULL,
  xlab_custom = NULL,
  ylab_custom = NULL,
  ann = TRUE,
  xlim = NULL,
  ylim = NULL,
  xaxt = "s",
  las = NULL
)
```

Arguments

summary_df	A data frame containing summary statistics including means, standard errors, and group identifiers.
main_title	Main title for the plot. Default is "Mean Values by Group".
ylab	Deprecated. Label for the y-axis.
xlab	Deprecated. Label for the x-axis.
bar_color	Color for the bars. Default is "skyblue".
error_bar_color	Color for the error bars. Default is "red".
bar_width	Width of the bars. Default is 0.7.

error_bar_length	Length of the error bar end caps. Default is 0.1.
axes	Logical indicating whether axes are drawn. Default is TRUE.
space	Numeric or vector indicating spacing between bars.
density	Numeric vector for shading density lines on bars.
angle	Angle of shading lines on bars.
col	Optional colors for shading lines (overrides bar_color).
names_arg	Character vector specifying names for x-axis labels. Defaults to group labels in summary_df.
xlab_custom	Custom label for the x-axis. Defaults to "Groups".
ylab_custom	Custom label for the y-axis. Defaults to "Mean".
ann	Logical indicating whether to draw axis labels and titles. Default is TRUE.
xlim	Numeric vector of length 2 defining x-axis limits.
ylim	Numeric vector of length 2 defining y-axis limits.
xaxt	Character specifying x-axis type; "s" for standard, "n" for none. Default is "s".
las	Numeric controlling orientation of axis labels.

Details

If the input data frame contains two grouping variables (e.g., Group1 and Group2), these are combined with a hyphen to create the x-axis labels. The function draws a bar plot of the means with error bars representing $\text{Mean} \pm \text{SE}$.

Value

Invisibly returns the midpoints of the bars (as from barplot).

Note

This function uses base R graphics and does not depend on external packages.

Author(s)

Oswald Omuron

References

See [barplot](#) and [arrows](#) in base R for details.

See Also

[summary](#) for creating summary data frames.

Examples

```
example_data <- c(
  445, 372, 284, 247, 328, 98.8, 108.7, 100.8, 123.6, 129.9, 133.3,
  130.1, 123.1, 186.6, 215, 19.4, 19.3, 27.8, 26, 22, 30.9, 19.8,
  16.5, 20.2, 31, 21.1, 16.5, 19.7, 18.9, 27, 161.8, 117, 94.6, 97.5,
  142.7, 109.9, 118.3, 111.4, 96.5, 109, 114.1, 114.9, 101.2, 112.7,
  111.1, 194.8, 169.9, 159.1, 100.8, 130.8, 93.6, 105.7, 178.4, 203,
  172.2, 127.3, 128.3, 110.9, 124.1, 179.1, 293, 197.5, 139.1, 98.1,
  84.6, 81.4, 87.2, 71.1, 70.3, 120.4, 194.5, 167.5, 121, 86.5, 81.7
)

example_group1 <- c(
  rep("Palm", 15), rep("Papyrus", 10), rep("Typha", 15),
  rep("Eucalyptus", 15), rep("Rice farm", 20)
)

example_group2 <- rep(c(50, 40, 30, 20, 10), 15)

example_df <- data.frame(
  Vegetation_types = example_group1,
  Depth_revised = example_group2,
  EC_uS_cm = example_data
)

summary_one_group <- summarize_data(
  example_df$EC_uS_cm,
  example_df$Vegetation_types
)

summary_two_groups <- summarize_data(
  example_df$EC_uS_cm,
  example_df$Vegetation_types,
  example_df$Depth_revised
)

plot_means(
  summary_two_groups,
  ylim = c(0, 350),
  las = 2,
  space = c(0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0)
)
```

summarize_data

Summarize Data by Groups

Description

This function summarizes a numeric vector by one or two grouping variables. It calculates mean, standard deviation, sample size, min, max, median, and standard error.

Usage

```
summarize_data(column_data, group_var1, group_var2 = NULL)
```

Arguments

<code>column_data</code>	A numeric vector containing the values to summarize.
<code>group_var1</code>	A factor or vector to group by (required).
<code>group_var2</code>	An optional second grouping factor or vector.

Value

A data frame containing summary statistics by group(s):

Group1 The first grouping variable.

Group2 The second grouping variable (if provided).

Mean Group mean.

SD Standard deviation.

N Sample size.

Min Minimum value.

Max Maximum value.

Median Median value.

SE Standard error of the mean.

Author(s)

Oswald Omuron

Examples

```
data <- c(10, 20, 30, 40, 50, 60)
group1 <- c("A", "A", "B", "B", "C", "C")
group2 <- c(1, 1, 2, 2, 3, 3)
summarize_data(data, group1)
summarize_data(data, group1, group2)
```

Index

- * **package**
 - Kifidi-package, [2](#)
- arrows, [13](#)
- barplot, [13](#)
- cor_vif_table, [3](#)
- count, [5](#)
- counts, [2, 4](#)
- generate_random_points, [2, 5](#)
- Kifidi (Kifidi-package), [2](#)
- Kifidi-package, [2](#)
- plot_group_regressions, [2, 6](#)
- plot_lme_regressions, [2, 7](#)
- plot_lmm_regressions, [2, 10](#)
- plot_means, [2, 12](#)
- summarize_data, [2, 14](#)
- summary, [13](#)
- table, [5](#)
- unique, [5](#)