Package 'rfriend'

May 30, 2026

Type Package

Title Provides Batch Functions and Visualisation for Basic Statistical Procedures

Version 3.1.0

Description Designed to streamline data analysis and statistical testing, reducing the length of R scripts while generating well-formatted outputs in 'pdf', 'Microsoft Word', and 'Microsoft Excel' formats. In essence, the package contains functions which are sophisticated wrappers around existing R functions that are called by using 'f_' (user f_riendly) prefix followed by the normal function name. This third version of the 'rfriend' package focuses primarily on data exploration, including tools for creating summary tables, f_summary(), summary figures, f_scan(), outlier detection and removal, f_outlier() and f_remove_outliers(), performing data transformations, f_boxcox() in part based on 'MASS/boxcox' and 'rcompanion', and f_bestNormalize() which wraps and extends functionality from the 'bestNormalize' package. Furthermore, 'rfriend' can automatically (or on request) generate visualizations such as boxplots, f_boxplot(), QQ-plots, f_qqnorm(), histograms f_hist(), and density plots f_density(). Additionally, the package includes several statistical test functions: f_aov(), f_chisq_test(), f_corplot(), f_kruskal_test(), f_lmer(), f_glm(), f_t_test(), f_wilcox_test(), for sequential testing and visualisation of the similar named 'stats' functions. These functions, except for f_chisq_test(), support testing multiple response variables and predictors, while also handling assumption checks, data transformations, and post hoc tests. Post hoc results are automatically summarized in a table using the compact letter display (cld) format for easy interpretation. The package also provides a function to do model comparison, f_model_comparison(), and several utility functions to simplify common R tasks. For example, f_clear() clears the workspace and restarts R with a single command; f_setwd() sets the working directory to match the directory of the current script; f_theme() quickly changes 'RStudio' themes; and f_factors() converts multiple columns of a data frame to factors, and much more.
If you encounter any issues or have feature requests, please feel free to contact me via email.

Note When loading, both MuMIn and rstatix are imported. Since rstatix internally depends on broom, this may trigger a warning about S3 method overwrites, specifically for nobs.fitdistr and nobs.multinom. These warnings are harmless and do not affect functionality.

License GPL-3

Encoding UTF-8

Depends R (>= 4.4.0)

Imports bestNormalize, crayon, DHARMa, dplyr, emmeans, ggplot2, grDevices, gridExtra, knitr, lme4, lmerTest, magick, multcomp, multcompView, MuMIn, nortest, pander, png, rlang, rmarkdown, rstatix, rstudioapi, this.path, tidyr, writexl, xfun

Suggests MASS, nnet, pbkrtest, testthat (>= 3.0.0), tibble

Config/testthat/edition 3

SystemRequirements Pandoc (>= 3.2)

NeedsCompilation no

Author Sander H. van Delden [aut, cre]

Maintainer Sander H. van Delden <plantmind@proton.me>

URL https://delde001.github.io/rfriend/

Config/roxygen2/version 8.0.0

Repository CRAN

Date/Publication 2026-05-30 19:10:02 UTC

Contents

`df_to_table` *Convert a data frame to a contingency table*

Description

Convert a data frame to a contingency table

Usage

```
df_to_table(df, label_col = NULL)
```

Arguments

`df`	A data frame. Either (a) one column contains row labels and the rest are numeric, (b) a fully numeric data frame with meaningful `rownames()` set, or (c) a fully numeric data frame with no row labels at all (in which case the data frame is coerced directly to a table without row labels).
`label_col`	Index or name of the column containing row labels. If NULL (default), the function auto-detects the first character/factor column. If no such column is found, the function falls back to using `rownames(df)` when these are meaningful, otherwise it coerces the data frame to a table as-is.

Value

A contingency table.

f_aov *Perform multiple* aov() *functions with optional data transformation, inspection and Post Hoc test.*

Description

Performs an Analysis of Variance (ANOVA) on a given dataset with options for (Box-Cox) transformations, normality tests, and post hoc analysis. Several response parameters can be analysed in sequence and the generated output can be in various formats ('Word', 'pdf', 'Excel').

Usage

```
f_aov(
  formula,
  data = NULL,
  norm_plots = TRUE,
  interaction_plots = TRUE,
  ANCOVA = FALSE,
  transformation = TRUE,
  force_transformation = NULL,
  force_aov = FALSE,
  alpha = 0.05,
  adjust = "sidak",
  intro_text = TRUE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  ...
)
```

Arguments

formula	A formula specifying the model to be fitted. More response variables can be added using - or + (e.g., response1 + response2 ~ predictor) to do a sequential aov() for each response parameter.
data	A data frame containing the variables in the model.
norm_plots	Logical. If TRUE, diagnostic residual plots are included in the output files. Default is TRUE.
interaction_plots	Logical. If TRUE, estimated means / interaction plots are included in the output files after the post hoc table. Default is TRUE.
ANCOVA	Logical. If TRUE, prevents automatic conversion of predictors to factors, allowing for Analysis of Covariance (ANCOVA). Default is FALSE.

`transformation` Logical or character string. If `TRUE`, or if `"boxcox"` applies a `f_boxcox()` transformation if residuals are not normal. If `"bestnormalize"`, applies `f_bestNormalize()` transformation. If `FALSE` no transformation will be applied. Default is `TRUE`.

`force_transformation`
Character string. A vector containing the names of response variables that should be transformed regardless of the normality test. Default is `NULL`

`force_aov` Logical. If `TRUE`, runs the ANOVA even when at least one cell has $n = 1$ (saturated model). By default (`FALSE`), such responses are skipped with a warning because F-statistics and p-values are undefined for saturated models. Set to `TRUE` only for diagnostic purposes – results should **not** be reported or interpreted as valid. Default is `FALSE`.

`alpha` Numeric. Significance level for ANOVA, post hoc tests, and Shapiro-Wilk test. Default is `0.05`.

`adjust` Character string specifying the method used to adjust p-values for multiple comparisons. Available methods include:

"tukey" Tukey's Honest Significant Difference method, appropriate for all pairwise comparisons. Controls family-wise error rate.

"sidak" Sidak correction that controls the family-wise error rate. Less conservative than Bonferroni.

"bonferroni" Conservative adjustment that multiplies p-values by the number of comparisons.

"none" No adjustment. Equivalent to Fisher's LSD method.

"fdr" False Discovery Rate adjustment, controls the expected proportion of false positives among significant results.

Default is `"sidak"`.

`intro_text` Logical. If `TRUE`, includes a short explanation about ANOVA assumptions in the output file. Default is `TRUE`.

`close_generated_files`
Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default `FALSE`. **WARNING:** Always save your work before using this option!!

`open_generated_files`
Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.

`output_type` Character string specifying the output format. Default is `"default"`.

- `"default"`: Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use `print(result)` or `plot(result)` to display the returned object.
- `"console"`: Forces immediate printing to the console regardless of object assignment.
- `"pdf"`, `"word"`, `"excel"`: Saves results to a file of the corresponding format. See `save_as`, `save_in_wdir`, and `open_generated_files` for file path and opening behavior.

- "rmd": Stores the raw markdown string inside the returned object for use in R Markdown documents.

save_as — Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in tempdir(). If only a directory is specified (providing an existing directory with trailing slash), the file is named "dataname_aov_output" in that directory. If an extension is provided the output format specified with option "output_type" will be overruled. Defaults to file.path(tempdir(), "dataname_summary.pdf").

save_in_wdir — Logical. If TRUE, saves the file in the working directory. Default is FALSE, this avoid unintended changes to the global environment. If save_as location is specified save_in_wdir is overwritten by save_as.

... — Additional arguments forwarded to aov. The arguments subset, na.action, and weights are handled specially: when supplied, they are applied via model.frame so that the n=1 cell check, Shapiro-Wilk test, Levene test, optional transformations, residual diagnostics, and emmeans post hoc tests all see the exact same row set as aov() itself. Any other aov() arguments (e.g. contrasts, projections, qr, contrasts.arg) are passed through unchanged.

Details

The function performs the following steps:

- Check if all specified variables are present in the data.
- Ensure that the response variable is numeric.
- Perform Analysis of Variance (ANOVA) using the specified formula and data.
- If shapiro = TRUE, check for normality of residuals using the Shapiro-Wilk test.
- If residuals are not normal and transformation = TRUE apply a data transformation.
- If significant differences are found in ANOVA, proceed with post hoc tests using estimated marginal means from emmeans() and Sidak adjustment (or another option of adjust =.

More response variables can be added using - or + (e.g., response1 + response2 ~ predictor) to do a sequential aov() for each response parameter captured in one output file.

Outputs can be generated in multiple formats ("pdf", "word", "excel" and "rmd") as specified by output_type. The function also closes any open 'Word' files to avoid conflicts when generating 'Word' documents. If output_type = "rmd" is used it is adviced to use it in a chunk with {r, echo=FALSE, results='asis'}

Non-significant ANOVA results: When the overall F-test is not significant, f_aov still reports the estimated marginal means table, but with all pairwise comparison letters replaced by *"ns"*. The numeric estimates (and their confidence intervals) are provided because they are often needed for manuscript tables, especially when the response was back-transformed from a Box-Cox or bestNormalize scale - the raw descriptive means and the emmeans values can differ, and it is the emmeans values that correspond to the actual model. The *"ns"* labels signal that pairwise differences should not be interpreted.

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder. (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

An object of class 'f_aov' containing results from `aov()`, normality tests, transformations, and post hoc tests. Using the option "output_type", it can also generate output in the form of: R Markdown code, 'Word', 'pdf', or 'Excel' files. Includes print and plot methods for 'f_aov' objects.

Multiple Testing Across Response Variables

When several response variables are analysed in a single call (e.g. `y1 + y2 + y3 ~ treatment`), each ANOVA is an independent null-hypothesis test at level `alpha`. The post hoc adjustments (`adjust = "sidak"`, `"tukey"`, etc.) only control the family-wise error rate **within** one ANOVA (across pairwise group comparisons for that response). They do **not** protect against the inflation of Type I error **across** the set of responses.

Practical implication: With k independent response variables all tested at $\alpha = 0.05$, the probability of obtaining at least one false positive is $1 - (1 - 0.05)^k$, which reaches ~40% for $k = 10$.

When this matters: The risk is highest in exploratory studies where many responses are screened simultaneously without a clear a priori hypothesis for each one. It is less of a concern when each response is a pre-specified primary outcome with its own biological rationale.

Possible remedies:

- **Bonferroni correction across responses:** use `alpha = 0.05 / k` where `k` is the number of response variables. Conservative but simple.
- **False Discovery Rate (FDR):** apply `p.adjust(p_values, method = "fdr")` to the vector of per-response ANOVA p-values after the fact.
- **MANOVA:** if the responses are correlated and you want a single omnibus test across all of them, use `manova()` before interpreting individual ANOVAs.
- **Pre-registration:** declare primary vs. exploratory responses before data collection to justify differential correction thresholds.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Make a factor of Species.
iris$Species <- factor(iris$Species)

# The left hand side contains two response variables,
# so two aov's will be conducted, i.e. "Sepal.Width"
# and "Sepal.Length" in response to the explanatory variable: "Species".
f_aov_out <- f_aov(Sepal.Width + Sepal.Length ~ Species,
                   data = iris,
                   # Save output in MS Word file (Default is console)
                   output_type = "word",
              # Do bestNormalize transformation for non-normal residual (Default is boxcox)
                   transformation = "bestnormalize"
                   )

# Print output to the console.
print(f_aov_out)

# Plot residual plots.
plot(f_aov_out)

#To print rmd output set chunck option to results = 'asis' and use cat().
f_aov_rmd_out <- f_aov(Sepal.Width ~ Species, data = iris, output_type = "rmd")
cat(f_aov_rmd_out$rmd)
```

`f_bestNormalize` *f_bestNormalize: Automated Data Normalization with bestNormalize*

Description

Applies optimal normalization transformations using 'bestNormalize', provides diagnostic checks, and generates comprehensive reports.

Usage

```
f_bestNormalize(
  data,
  alpha = 0.05,
  plots = FALSE,
  data_name = NULL,
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  ...
)
```

Arguments

`data`	Numeric vector or single-column data frame.
`alpha`	Numeric. Significance level for normality tests (default = `0.05`).
`plots`	Logical. If `TRUE`, plots Q-Q plots and Histograms of the original and transformed data. Default is `FALSE`.
`data_name`	A character string to manually set the name of the data for plot axis and reporting. Default extracts name from input object. `data`.
`output_type`	Character string specifying the output format. Default is `"default"`. • `"default"`: Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use `print(result)` or `plot(result)` to display the returned object. • `"console"`: Forces immediate printing to the console regardless of object assignment. • `"pdf"`, `"word"`, `"excel"`: Saves results to a file of the corresponding format. See `save_as`, `save_in_wdir`, and `open_generated_files` for file path and opening behavior. • `"rmd"`: Stores the raw markdown string inside the returned object for use in R Markdown documents.
`save_as`	Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in `tempdir()`. If only a directory is specified (providing an existing directory with trailing slash), the file is named "data_name_transformed" in that directory. If an extension is provided the output format specified with option "output_type" will be overruled. Defaults to `file.path(tempdir(), "data_name_transformed.pdf")`.
`save_in_wdir`	Logical. If `TRUE`, saves the file in the working directory. Default is `FALSE`, this avoid unintended changes to the global environment. If `save_as` location is specified `save_in_wdir` is overwritten by `save_as`.
`close_generated_files`	Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default `FALSE`. **WARNING:** Always save your work before using this option!!
`open_generated_files`	Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.
`...`	Additional arguments passed to bestNormalize.

Details

This is a wrapper around the 'bestNormalize' package. Providing a fancy output and the settings of 'bestNormalize' are tuned based on sample size n. If n < 100, `loo` = `TRUE`, `allow_orderNorm` = `FALSE` and `r` doesn't matter as `loo` = `TRUE`. If 100 <= n < 200, `loo` = `FALSE`, `allow_orderNorm` =

TRUE and r = 50. If n >= 200, loo = FALSE, allow_orderNorm = TRUE, r = 10. These setting can be overwritten by user options.

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

Returns an object of class 'f_bestNormalize' containing:

- transformed_data Normalized vector.
- bestNormalize Full bestNormalize object from original package.
- data_name Name of the analyzed dataset.
- transformation_name Name of selected transformation.
- shapiro_original Shapiro-Wilk test results for original data.
- shapiro_transformed Shapiro-Wilk test results for transformed data.
- norm_stats Data frame of normality statistics for all methods.
- rmd Rmd code if outputype = "rmd".

Also generates reports in 'Word', or 'pdf' files. When using output to console and plots = TRUE, the function prints QQ-plots, Histograms and a summary data transformation report. Includes print and plot methods for objects of class 'f_bestNormalize'.

Author(s)

Sander H. van Delden <plantmind@proton.me>

References

Peterson, C. (2025). **bestNormalize**: Flexibly calculate the best normalizing transformation for a vector. Available at: https://cran.r-project.org/package=bestNormalize

Examples

```
# Use set.seed to keep the outcome of bestNormalize stable.
set.seed(123)

# Create some skewed data (e.g., using a log-normal distribution).
```

```
skewed_data <- rlnorm(100, meanlog = 0, sdlog = 1)

# Basic usage: transform and store the full result object.
result <- f_bestNormalize(skewed_data, data_name = "Skewed log-normal data")

# Print a summary of the transformation.
print(result)

# Inspect normality statistics for all candidate transformations.
result$norm_stats

# Plot histograms and QQ-plots for original vs. transformed data.
plot(result)

# Use plots = TRUE to auto-plot when output_type = "default" (default).
result2 <- f_bestNormalize(skewed_data, plots = TRUE)

# Extract only the transformed (data) vector directly.
transformed_data <- f_bestNormalize(skewed_data)$transformed_data

# data.frame input: column name is used as data_name automatically.
df <- data.frame(measurement = skewed_data)
result_df <- f_bestNormalize(df)

# Data with NAs: NAs are preserved at their original positions.
skewed_na <- skewed_data
skewed_na[c(5, 20)] <- NA
result_na <- f_bestNormalize(skewed_na)

# Access a specific alternative transformation (first check what is available).
names(result$bestNormalize$other_transforms)
# Then extract the one you want, e.g.:
# result$bestNormalize$other_transforms$yeojohnson$x.t

# Force output to console (prints report + plots automatically).
f_bestNormalize(skewed_data, output_type = "console")

# Generate a PDF report saved to a custom path.
f_bestNormalize(skewed_data,
                output_type          = "pdf",
                save_as              = "my_report"
                )

# Generate R Markdown output for use inside a .Rmd chunk
# (set chunk option results = 'asis').
rmd_result <- f_bestNormalize(skewed_data, output_type = "rmd")
cat(rmd_result$rmd)
```

f_boxcox *f_boxcox: A User-Friendly Box-Cox Transformation*

Description

Performs a Box-Cox transformation on a dataset to stabilize variance and make the data more normally distributed. It also provides diagnostic plots and tests for normality. The transformation is based on code of MASS/R/boxcox.R. The function prints λ to the console and returns (output) the transformed data set.

Usage

```
f_boxcox(
  data = data,
  digits = 3,
  range = c(-2, 2),
  plots = NULL,
  transform.data = TRUE,
  eps = 1/50,
  xlab = expression(lambda),
  ylab = "log-Likelihood",
  alpha = 0.05,
  open_generated_files = interactive(),
  close_generated_files = FALSE,
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  ...
)
```

Arguments

data	A numeric vector or a data frame with a single numeric column. The data to be transformed.
digits	Numeric. Determines the accuracy of the estimate for lambda. Higher values increase computation time. Defaults to 3.
range	A numeric vector of length 2 defining the search interval for lambda. Defaults to c(-2, 2).
plots	Logical. If TRUE, plots log-likelihood of the Box-Cox transformation, Histograms and Q-Q plots of the original and transformed data. Default is FALSE.
transform.data	Logical. If TRUE, returns the transformed data. Default is TRUE.
eps	A small positive value used to determine when to switch from the power transformation to the log transformation for numerical stability. Default is 1/50.
xlab	Character string. Label for the x-axis in plots. Default is an expression object representing λ.
ylab	Character string. Label for the y-axis in plots. Default is "log-Likelihood".

`alpha` Numeric. Significance level for the Shapiro-Wilk test of normality. Default is `0.05`.

`open_generated_files`
Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.

`close_generated_files`
Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default `FALSE`. **WARNING:** Always save your work before using this option!!

`output_type` Character string specifying the output format. Default is `"default"`.

- `"default"`: Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use `print(result)` or `plot(result)` to display the returned object.
- `"console"`: Forces immediate printing to the console regardless of object assignment.
- `"pdf"`, `"word"`, `"excel"`: Saves results to a file of the corresponding format. See `save_as`, `save_in_wdir`, and `open_generated_files` for file path and opening behavior.
- `"rmd"`: Stores the raw markdown string inside the returned object for use in R Markdown documents.

`save_as` Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in `tempdir()`. If only a directory is specified (providing an existing directory with trailing slash), the file is named "data_name_aov_output" in that directory. If an extension is provided the output format specified with option "output_type" will be overruled. Defaults to `file.path(tempdir(), "data_name_summary.pdf")`.

`save_in_wdir` Logical. If `TRUE`, saves the file in the working directory. Default is `FALSE`, this avoid unintended changes to the global environment. If `save_as` location is specified `save_in_wdir` is overwritten by `save_as`.

`...` Additional arguments passed to plotting functions.

Details

The function uses the following formula for transformation:

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(y), & \lambda = 0 \end{cases}$$

where (y) is the data being transformed, and (λ) the transformation parameter, which is estimated from the data using maximum likelihood. The function computes the Box-Cox transformation for a range of λ values and identifies the λ that maximizes the log-likelihood function. The beauty of this transformation is that, it checks suitability of many of the common transformations in one run. Examples of most common transformations and their λ value is given below:

λ-Value	Transformation
-2	$\frac{1}{x^2}$
-1	$\frac{1}{x}$
-0.5	$\frac{1}{\sqrt{x}}$
0	$log(x)$
0.5	$\sqrt{x}$
1	x
2	x^2

If the estimated transformation parameter closely aligns with one of the values listed in the previous table, it is generally advisable to select the table value rather than the precise estimated value. This approach simplifies interpretation and practical application.

The function provides diagnostic plots: a plot of log-likelihood against λ values and a Q-Q plot of the transformed data.It also performs a Shapiro-Wilk test for normality on the transformed data if the sample size is less than or equal to 5000.

Note: For sample sizes greater than 5000, Shapiro-Wilk test results are not provided due to limitations in its applicability.

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

An object of class 'f_boxcox' containing, among others, results from the boxcox transformation, lambda, the input data, transformed data, Shapiro-Wilk test on original and transformed data. Using the option "output_type", it can also generate output in the form of: R Markdown code, 'Word', or 'pdf' files. Includes print and plot methods for 'f_boxcox' objects.

Author(s)

- Sander H. van Delden <plantmind@proton.me>
- Salvatore Mangiafico, <mangiafico@njaes.rutgers.edu>
- W. N. Venables and B. D. Ripley

References

The core of calculating λ and the plotting was taken from:
file MASS/R/boxcox.R copyright (C) 1994-2004 W. N. Venables and B. D. Ripley

- https://r-coder.com/box-cox-transformation-r/
- https://CRAN.R-project.org/package=MASS

Some code to present the result was taken and modified from file:
rcompanion/R/transformTukey.r. (Developed by Salvatore Mangiafico)

- https://rcompanion.org/handbook/I_12.html

The explanation on BoxCox transformation provided here was provided by r-coder:

- https://r-coder.com/box-cox-transformation-r/

See Also

boxcox

Examples

```
# Create non-normal data in a data.frame or vector.
df   <- data.frame(values = rlnorm(100, meanlog = 0, sdlog = 1))

# Store the transformation in object "bc".
bc <- f_boxcox(df$values)

# Print lambda and Shaprio.
print(bc)

# Plot the QQ plots, Histograms and Lambda Log-Likelihood estimation.
plot(bc)

# Or Directly use the transformed data from the f_boxcox object.
df$values_transformed <- f_boxcox(df$values)$transformed_data
print(df$values_transformed)
```

---

f_boxplot *Generate a Boxplot Report of a data.frame*

Description

Generates boxplots for all numeric variables in a given dataset, grouped by factor variables. The function automatically detects numeric and factor variables. It allows two output formats ('pdf', 'Word') and includes an option to add a general explanation about interpreting boxplots.

Usage

```
f_boxplot(x, ...)

## S3 method for class 'formula'
f_boxplot(formula, data = NULL, ...)

## S3 method for class 'data.frame'
f_boxplot(x, ...)

## S3 method for class 'numeric'
f_boxplot(x, ...)

## S3 method for class 'integer'
f_boxplot(x, ...)

f_boxplot_worker(
  formula = NULL,
  data,
  fancy_names = NULL,
  output_type = "pdf",
  outliers = TRUE,
  coef = 1.5,
  limit_columns = 7,
  save_as = NULL,
  save_in_wdir = FALSE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  boxplot_explanation = TRUE,
  detect_factors = TRUE,
  jitter = FALSE,
  width = 8,
  height = 7,
  units = "in",
  res = 300,
  las = 2,
  color = "rainbow",
  boxwidth = NULL,
  ...
)
```

Arguments

x	A data.frame, formula, or numeric/integer vector (dispatches to the correct method). When a single numeric or integer vector is supplied, it is treated as a single response variable, plotted on the y-axis with the variable name as label, and grouped by a single dummy factor (one box). When several unnamed numeric vectors are supplied (as in base R's `boxplot()`, e.g. `f_boxplot(x1, x2)`), each becomes its own box side by side, labelled with the original variable name on

	the x-axis.
...	Further arguments forwarded to f_boxplot_worker, such as fancy_names, title, fill, etc.
formula	A formula specifying the factor to be plotted. More response variables can be added using - or + (e.g., response1 + response2 ~ predictor) to generate multiple boxplots. If the formula is omitted and only data is provided all data will be used for creating boxplots.
data	A data.frame containing the data to be used for creating boxplots.
fancy_names	An optional named vector mapping column names in data to more readable names for display in plots (name map). Defaults to NULL.
output_type	Character string, specifying the output format: "pdf", "word", "rmd" or "png". The option "rmd" saves rmd code in the output object not in a file. Default is "pdf".
outliers	Logical. If TRUE, scans for outliers using Tukey's fences and if they exist, adds them to the report using f_outliers. Default TRUE.
coef	Numeric. The multiplier for the Interquartile Range (IQR) used for outlier detection. Default 1.5.
limit_columns	Integer or NULL. Defines the number of columns shown in the outlier table. Default = 7. NULL = all columns are shown.
save_as	Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in tempdir(). If only a directory is specified (providing an existing directory with trailing slash), the file is named "dataname_BoxPlot" in that directory. If an extension is provided the output format specified with option "output_type" will be overruled. Defaults to file.path(tempdir(), "dataname_BoxPlot.pdf").
save_in_wdir	Logical. If TRUE, saves the file in the working directory. Default is FALSE, this avoid unintended changes to the global environment. If save_as location is specified save_in_wdir is overwritten by save_as.
close_generated_files	Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default FALSE. **WARNING:** Always save your work before using this option!!
open_generated_files	Logical. Whether to open the generated output files after creation. Defaults to TRUE in an interactive R session and FALSE otherwise (e.g. in scripts or automated pipelines). Set to TRUE or FALSE to override this behaviour explicitly.
boxplot_explanation	A logical value indicating whether to include an explanation of how to interpret boxplots in the report. Defaults to TRUE.
detect_factors	A logical value indicating whether to automatically detect factor variables in the dataset. Defaults to TRUE.
jitter	A logical value, if TRUE all data per boxplot is shown, if FALSE (default) individual data points (except for outliers) are omitted.

width	Numeric, png figure width default 8 inch
height	Numeric, png figure height default 7 inch
units	Character string, png figure units default "in" = inch, other options are: "px" = Pixels, "cm" = centimeters, "mm" = millimeters.
res	Numeric, png figure resolution default 300 dpi
las	An integer (0 t/m 3), las = 0: Axis labels are parallel to the axis. las = 1: Axis labels are always horizontal. las = 2: Axis labels are perpendicular to the axis. (default setting). las = 3: Axis labels are always vertical.
color	Colour scheme for the boxes. One of: "rainbow" (default; one hue per group), "bw" (white fill with black lines, outliers and mean marker, suitable for publication), a single R colour name or hex string applied to all boxes (with a transparent fill and a darker outline derived from it), or a vector of colours which is recycled to the number of groups for a custom per-group palette.
boxwidth	Numeric or NULL. Relative width of each box, passed as boxwex to boxplot(). When NULL (default) the width is computed automatically as num.bars/18, keeping boxes roughly comparable across plots with different numbers of groups. Supply a numeric value (for example 0.5) to override. The numeric/integer vector method uses 0.4 by default to avoid an overly thin single box.

Details

The function performs the following steps:

- Detects numeric and factor variables in the dataset.
- Generates boxplots for each numeric variable grouped by each factor variable.
- Outputs the report in the specified format ('pdf', 'Word' or 'Rmd').

If output_type = "rmd" is used it is adviced to use it in a chunk with {r, echo=FALSE, results='asis'}

If no factor variables are detected, the function stops with an error message since factors are required for creating boxplots.

This function will plot all numeric and factor candidates, use the function subset() to prepare a selection of columns before submitting to f_boxplot().

Note that there is an optional jitter option to plot all individual data points over the boxplots.

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

Windows: Install Pandoc and ensure the installation folder (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.

macOS: If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.

Linux: Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.

If Pandoc is not found, this function may not work as intended.

Value

The return value depends on output_type:

- "pdf" and "word": Writes a report file to save_as (or tempdir() by default) and returns NULL invisibly. The file can optionally be opened with open_generated_files = TRUE.
- "png": Writes one PNG file per response x factor combination into the directory given by save_as and returns NULL invisibly.
- "rmd": Returns the generated R Markdown content as a single character string (invisibly). No file is written and nothing is printed to the console. The caller can cat() the string, assign it to a variable, or embed it in a larger report (see Examples).

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Example usage:
data(iris)

new_names = c(
  "Sepal.Length" = "Sepal length (cm)" ,
  "Sepal.Width" = "Sepal width (cm)",
  "Petal.Length" = "Petal length (cm)",
  "Petal.Width" = "Petal width (cm)",
  "Species" = "Cultivar"
)

# Use the whole data.frame to generate an MS Word report and don't open it.
f_boxplot(iris,
           fancy_names = new_names,
           output_type = "word"
           )

# Use a formula to plot several response parameters (response 1 + response 2 etc)
# and generate a rmd output without boxplot_explanation.
data(mtcars)
f_boxplot(hp + disp ~ gear*cyl,
           data=mtcars,
           boxplot_explanation = FALSE,
           output_type = "word"
           )

# Pass a bare numeric vector. Its name is used as the y-axis label
# and as the data_name in the output filename.
set.seed(1)
my_vec <- rnorm(50, mean = 10)
f_boxplot(my_vec, output_type = "png")

# Formula with bare vectors (no data.frame): group hp by cyl.
```

```
hp1  <- mtcars$hp
cyl1 <- mtcars$cyl
f_boxplot(hp1 ~ cyl1, output_type = "png")

# Multiple unnamed numeric vectors, base R's boxplot() convention:
# each vector becomes its own box, labelled on the x-axis with its
# original variable name. Use the formula syntax above when you
# instead want to group one response by a factor.
f_boxplot(hp1, cyl1, output_type = "png")

# Capture the R Markdown output as a string and render it inline.
# Use output_type = "rmd" to get the markdown back as a character value
# instead of writing a file. Useful for embedding in a larger knitr document.
rmd <- f_boxplot(iris,
                 output_type         = "rmd",
                 boxplot_explanation = FALSE,
                 outliers            = FALSE
                 )

# Display it in the console
cat(rmd)

# ...or splice it into a knitr child chunk with results = "asis":
#   ```{r, echo=FALSE, results='asis'}
#   cat(rmd)
#   ```
```

`f_chisq_test` *Chi-squared Test with post hoc Analysis*

Description

Performs a chi-squared test `chisq.test`, then automatically conducts post hoc analysis if the test is significant. The function provides adjusted p-values for each cell in the contingency table using a specified correction method.

Usage

```
f_chisq_test(
  x,
  y,
  p = NULL,
  method = "bonferroni",
  digits = 3,
  alpha = 0.05,
```

```
  force_posthoc = FALSE,
  ...
)
```

Arguments

x	A numeric vector (or factor), or a contingency table in matrix or table form. If a data frame is entered the function will try to convert it to a table using df_to_table().
y	A numeric vector; ignored if x is a matrix, table or data.frame. If x is a factor, y should be a factor of the same length.
p	A vector of probabilities of the same length as x. Default is NULL. An error is given if any entry of p is negative.
method	Character string specifying the adjustment method for p-values. Default is "bonferroni". Other options include "holm", "hochberg", "hommel", "BH", "BY", "fdr", and "none".
digits	Integer specifying the number of decimal places for rounding. Default is 3.
alpha	Numeric threshold for significance. Default is 0.05.
force_posthoc	Logical indicating whether to perform post hoc tests even if the chi-squared test is not significant. Default is FALSE.
...	Additional arguments passed to chisq.test.

Details

The function first performs a chi-squared test using chisq.test. If the test is significant ($p <$ alpha) or if force_posthoc = TRUE, it conducts post hoc analysis by examining the standardized residuals. The p-values for these residuals are adjusted using the specified method to control for multiple comparisons.

If the input is a data frame, the function attempts to convert it to a table and displays the resulting table for verification.

Value

An object of class f_chisq_test containing:

- chisq_test_output: The output from chisq.test.
- adjusted_p_values: Matrix of adjusted p-values (for table/matrix input).
- observed_vs_adj_p_value: Interleaved table of observed values and adjusted p-values.
- stdres_vs_adj_p_value: Interleaved table of standardized residuals and adjusted p-values.
- adj_p_values: Vector of adjusted p-values (for vector input).
- posthoc_output_table: Data frame with observed values, expected values, standardized residuals, and adjusted p-values (for vector input).
- observed_vs_adj_p_value: Interleaved table of observed values and adjusted p-values (for table/matrix input).
- stdres_vs_adj_p_value: Interleaved table of standardized residuals and adjusted p-values (for table/matrix input).

Author(s)

Sander H. van Delden <plantmind@proton.me>

References

This function implements a post hoc analysis for chi-squared tests inspired by the methodology in:

Beasley, T. M., & Schumacker, R. E. (1995). Multiple Regression Approach to Analyzing Contingency Tables: Post Hoc and Planned Comparison Procedures. The Journal of Experimental Education, 64(1), 79-93.

The implementation draws inspiration from the 'chisq.posthoc.test' package by Daniel Ebbert.

Examples

```
# Chi.square on independence: Association between two variables.
# Create a contingency table.
my_table <- as.table(rbind(c(100, 150, 50), c(120, 90, 40)))
dimnames(my_table) <- list(Gender = c("Male", "Female"),
                           Response = c("Agree", "Neutral", "Disagree"))

# Perform chi-squared test with post hoc analysis.
f_chisq_test(my_table)

# Use a different adjustment method.
f_chisq_test(my_table, method = "holm")

# Other forms still work like Goodness-of-Fit: Match to theoretical distribution.
# Observed frequencies of rolling with a die 1 - 6.
observed <- c(2, 2, 10, 20, 15, 11)

# Expected probabilities under a fair die.
expected_probs <- rep(1/6, 6)

# Chi-Square Goodness-of-Fit Test.
f_chisq_test(x = observed, p = expected_probs)
```

---

f_clear *f_clear: Clear Various Aspects of the R Environment*

Description

Provides a convenient way to clear different components of the R environment, including the console, memory, graphics, and more. It also offers the option to restart the R session. This can come in handy at the start of an R script.

Usage

```
f_clear(env = TRUE, gc = TRUE, console = TRUE, graph = TRUE, restart = FALSE)
```

Arguments

env	Logical. If TRUE, all objects in the global environment are removed. Default is TRUE.
gc	Logical. If TRUE, garbage collection is performed to free up memory. Default is TRUE.
console	Logical. If TRUE, the R console is cleared. Default is TRUE.
graph	Logical. If TRUE, all open graphics devices are closed. Default is TRUE.
restart	Logical. If TRUE, the R session is restarted using 'RStudio's' API. Default is FALSE.

Details

- Console Clearing: Clears the console output.
- Garbage Collection: Performs garbage collection to free memory from unreferenced objects.
- Graph Clearing: Closes all open graphics devices.
- Environment Clearing: Removes all objects from the global environment.
- Session Restart: Restarts the R session (only available in 'RStudio').

Value

No return value, called for side effects, see details.

Note

The restart parameter requires 'RStudio' and its API package ('rstudioapi') to be installed and available.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Clear console, memory, graphs, and for example NOT the environment.
f_clear(env = FALSE)
```

`f_conditional_round` *Conditional Rounding for Numeric Values*

Description

Conditionally formats numeric values based on their magnitude. Values that are very small or very large are formatted using scientific notation, while other values are rounded to a specified number of decimal places. Integers are preserved without decimal places. When applied to a data frame, only numeric columns are processed. All output is character string.

Usage

```
f_conditional_round(
  x,
  threshold_small = 0.01,
  threshold_large = 10000,
  digits = 3,
  replace_na = TRUE,
  na_string = "-",
  allow_integer_decimal_mix = FALSE
)
```

Arguments

`x`	A numeric vector or data frame containing numeric columns to be formatted.
`threshold_small`	Numeric value. Values with absolute magnitude smaller than this threshold will be formatted using scientific notation. Default is `0.01`.
`threshold_large`	Numeric value. Values with absolute magnitude larger than or equal to this threshold will be formatted using scientific notation. Default is `10000`.
`digits`	Integer. Number of decimal digits to use in formatting. Default is `3`.
`replace_na`	Logical. If `TRUE`, `NA` values in the output are replaced with the string specified by `na_string`. If `FALSE`, `NA` values are left as `NA`. Default is `TRUE`.
`na_string`	Character string used to replace `NA` values in the output when `replace_na = TRUE`. Use `"-"` (default) for a standard "not available" dash, `""` for empty cells, or any other string as needed. Note that special characters such as the em dash (`"\u2014"`) may require additional LaTeX declarations when rendering to PDF via `pdflatex`. Default is `"-"`.
`allow_integer_decimal_mix`	Logical. If `TRUE`, each individual cell is evaluated: integer values are displayed without decimal places, and non-integer values are displayed with the specified number of decimal places, i.e. `digits`. Default is `FALSE`, when a column contains a mix of integers and decimal values, all values are displayed with the specified number of decimal places. Note: columns containing only integers are **always** displayed without decimal places, regardless of `allow_integer_decimal_mix`.

Details

The function applies the following formatting rules:

- Values smaller than `threshold_small` or larger than `threshold_large` are formatted in scientific notation with decimal `digits`.
- Integer values are formatted without decimal places.
- Non-integer values that don't require scientific notation are rounded to `digits` decimal places.
- NA values are replaced with empty strings if `replace_na = TRUE`.
- Empty strings in the input are preserved.
- For data frames, only numeric columns are processed; other columns remain unchanged.

Value

- If input is a vector: A character vector of the same length as the input, with values formatted according to the specified rules.
- If input is a data frame: A data frame with the same structure as the input, but with character columns formatted according to the specified rules.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Vector examples.
f_conditional_round(c(0.0001, 0.5, 3, 10000))
# Returns: "1.000e-04" "0.500" "3" "1.000e+04".

f_conditional_round(c(0.0001, 0.5, 3, 10000, NA), replace_na = TRUE)
# Returns: "1.000e-04" "0.500" "3" "1.000e+04" ""

# Data frame example.
df <- data.frame(
  name = c("A", "B", "C"),
  small_val = c(0.0001, 0.002, 0.5),
  integer = c(1, 2, 3),
  integer_mix = c(10, 20, 30.1),
  large_val = c(10000, 5000, NA)
)

# Show only two digits.
f_conditional_round(df, digits = 2)

# To keep Integers as Integers (no digits)
# in columns with mixed data (Integers and digits)
# set allow_integer_decimal_mix = TRUE
f_conditional_round(df, allow_integer_decimal_mix = TRUE)
```

```
# Custom NA replacement string.
f_conditional_round(c(0.5, NA, 3), replace_na = TRUE, na_string = "-")
# Returns: "0.500" "-" "3"

f_conditional_round(c(0.5, NA, 3), replace_na = TRUE, na_string = "")
# Returns: "0.500" "" "3"
```

`f_corplot` *Correlation Plots with Factor Detection and Multiple Correlation Coefficients*

Description

Creates correlation plots for numeric variables in a data frame. The upper triangle displays Pearson r, Spearman ρ, and Kendall τ simultaneously for each pair. Factor variables are automatically detected and used for grouping, i.e. point colouring and shaping. Ordinal variables are supported via `ordinal_vars`: their diagonal labels are italicised and Pearson r is greyed and bracketed for any pair that involves them.A separate legend file documents both the grouping factors and the meaning of all three correlation symbols.

Usage

```
f_corplot(
  data,
  detect_factors = TRUE,
  factor_table = FALSE,
  factor_exclude = NULL,
  factor_select = NULL,
  unique_num_treshold = 8,
  repeats_threshold = 2,
  color_factor = "auto",
  shape_factor = "auto",
  print_legend = TRUE,
  fancy_names = NULL,
  ordinal_vars = NULL,
  width = 15,
  height = 15,
  res = 600,
  pointsize = 10,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  output_type = "word",
  save_as = NULL,
  save_in_wdir = FALSE
)
```

Arguments

data	A data.frame containing the dataset. Must include at least two numeric variables.
detect_factors	Logical. If TRUE, factor variables are automatically detected for colouring and shaping points. Default TRUE.
factor_table	Logical. If TRUE, prints a detailed table of converted factors to the console. Default FALSE.
factor_exclude	A character vector specifying the names of the columns NOT to convert into factors. If NULL, no columns are excluded. Default is NULL.
factor_select	A character vector specifying the names of the columns to convert into factors. If NULL, the function automatically detects columns that should be factors based on their data type and unique value count. Default is NULL.
unique_num_treshold	Numeric. A threshold of the amount of unique numbers a numeric column should have to keep it numeric, i.e. omit factor conversion. Default 8.
repeats_threshold	Numeric. A threshold of the minimal number of repeats a numeric column should have to convert it to a factor. Default 2.
color_factor	Character. Name of the factor variable used for point colours; "auto" selects automatically. Default "auto".
shape_factor	Character. Name of the factor variable used for point shapes; "auto" selects automatically. Default "auto".
print_legend	Logical. If TRUE, a separate legend file is created. Default TRUE.
fancy_names	Named character vector or NULL. Maps column names to display names used in the plot and legend.
ordinal_vars	Character vector or NULL. Names of variables to treat as ordinal. Ordered factors are coerced to integer ranks; other non-numeric types are coerced similarly. Their diagonal labels are italicised and Pearson r is greyed and bracketed for any pair that involves them. Ordinal variables are included in the correlation panels but excluded from aesthetic factor detection. Default NULL.
width	Numeric. Plot width in centimetres. Default 15.
height	Numeric. Plot height in centimetres. Default 15.
res	Numeric. Resolution in DPI. Default 600.
pointsize	Numeric. Base font size. Default 8.
close_generated_files	Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default FALSE. **WARNING:** Always save your work before using this option!!
open_generated_files	Logical. Whether to open the generated output files after creation. Defaults to TRUE in an interactive R session and FALSE otherwise (e.g. in scripts or automated pipelines). Set to TRUE or FALSE to override this behaviour explicitly.

output_type	Character. One of "pdf", "word", "png", or "rmd". Default "word".
save_as	Character or NULL. Output file path without extension. A full path, a filename, or a directory (with trailing slash) are all accepted. Providing an extension overrides output_type. Default saves to tempdir().
save_in_wdir	Logical. If TRUE, saves to the working directory. Overridden by save_as. Default FALSE.

Details

- **Three correlations per panel:** Every upper-triangle panel shows r (Pearson), ρ (Spearman), and τ (Kendall) stacked vertically, so the reader can choose the most appropriate coefficient for each variable pair.
- **Ordinal variables:** Specify column names with ordinal_vars. Those variables appear in italic on the diagonal. For any pair where at least one variable is ordinal, Pearson r is shown greyed and in parentheses to signal it is technically inappropriate; Spearman and Kendall remain prominent.
- **Factor detection:** Only unordered factors are used for colour/shape aesthetics. Ordered factors (is.ordered()) are treated as ordinal data, not as grouping variables.
- **Legend:** The legend file documents the grouping factor levels (when present) and always includes an explanation of all three correlation symbols whenever a legend is generated.
- **Constant columns:** Zero-variance columns produce NA in all correlation panels.

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

No value is returned to the R environment. Output files are saved and opened automatically.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
data(mtcars)
mtcars_sub <- subset(mtcars, select = -c(am, qsec, vs))
f_corplot(mtcars_sub,
          color_factor = "gear",
```

```
          shape_factor = "cyl",
          output_type  = "png"
          )

# With ordinal variables
data(iris)
fancy_names <- c(Sepal.Length = "Sepal Length (cm)",
                 Sepal.Width  = "Sepal Width (cm)")
f_corplot(iris,
          fancy_names  = fancy_names,
          ordinal_vars = "Petal.Width",
          output_type  = "png",
          open_generated_files = FALSE)
```

`f_factors` *Convert multiple columns to Factors in a data frame*

Description

Converts multiple specified columns of a data frame into factors. If no columns are specified, it automatically detects and converts columns that are suitable to be factors. The function returns the entire data frame including non factor columns and can report the properties of this new data frame in the console (`properties = TRUE`).

Usage

```
f_factors(
  data,
  select = NULL,
  exclude = NULL,
  properties = FALSE,
  force_factors = FALSE,
  unique_num_treshold = 8,
  repeats_threshold = 2,
  ...
)
```

Arguments

`data`	A data frame containing the columns to be converted.
`select`	A character vector specifying the names of the columns to convert into factors. If `NULL`, the function automatically detects columns that should be factors based on their data type and unique value count. Default is `NULL`.
`exclude`	A character vector specifying the names of the columns NOT to convert into factors. If `NULL`, no columns are excluded. Default is `NULL`.

`properties` Logical. If `TRUE`, prints a detailed table about the properties of the new data frame to the console. If `FALSE` no property table will be printed to the console. Default is `FALSE`.

`force_factors` Logical. If `TRUE` all columns in the data.frame will be converted to factors except for the excluded columns using `exclude`.

`unique_num_treshold` Numeric. A threshold of the amount of unique numbers a numeric column should have to keep it numeric, i.e. omit factor conversion. Default `8`.

`repeats_threshold` Numeric. A threshold of the minimal number of repeats a numeric column should have to convert it to a factor. Default `2`.

`...` Additional arguments passed to the `factor()` function of baseR.

Details

- If `select` is `NULL`, the function identifies columns with character data or numeric data with fewer than 8 unique values as candidates for conversion to factors.
- The function checks if all specified columns exist in the data frame and stops execution if any are missing.
- Converts specified columns into factors, applying any additional arguments provided.
- Outputs a summary data frame with details about each column, including its type, class, number of observations, missing values, factor levels, and labels.

Value

Returns the modified data frame with the specified (or all suitable) columns converted to factors. Can also force a print of a summary of the data frame's structure to the console (properties = TRUE).

Author(s)

Sander H. van Delden <plantmind@proton.me>

See Also

`factor`

Examples

```
# Make a data.frame:
df <- data.frame(a = c("yes", "no", "yes", "yes", "no",
                       "yes", "yes", "no", "yes"),
                 b = c(1, 2, 3, 1, 2, 3, 1, 2, 3),
                 c = c("apple", "kiwi", "banana", "apple", "kiwi",
                       "banana", "apple", "kiwi", "banana"),
                 d = c(1.1, 1.1, 3.4, 4.5, 5.4, 6.7, 7.8, 8.1, 9.8)
                 )
str(df)
```

```
# Convert specified columns to factors:
df1 <- f_factors(df, select = c("a", "c"))
str(df1)


# Convert all potential factor columns to factor but exclude column "b":
df2 <- f_factors(df, exclude = c("b"))
str(df2)

# Convert all columns to factor but exclude column "b":
df3 <- f_factors(df, exclude = c("b"), force_factors = TRUE)
str(df3)

# Or automatically detect and convert suitable columns to factors.
# Thus obtaining the same results as above automatically:
df4 <- f_factors(df)
str(df4)

# In example above col b was converted to a factor as the number of repeats = 2
# and the amount of unique numbers < 8. In order to keep b numeric we can also
# adjust the unique_num_treshold and/or repeats_threshold:
df5 <- f_factors(df, unique_num_treshold = 2)
str(df5)

# Use `properties = TRUE` to view the data frame's structure.
# This forces a printed output which is more insight than standard str() output.
df6 <- f_factors(df, properties = TRUE)
```

`f_glm` *Perform multiple* `glm()` *functions with diagnostics, assumption checking, and post hoc analysis*

Description

Performs Generalized Linear Model (GLM) analysis on a given dataset with options for diagnostics, assumption checking, and post hoc analysis. Several response parameters can be analyzed in sequence and the generated output can be in various formats ('Word', 'pdf', 'Excel').

Usage

```
f_glm(
  formula,
  family = gaussian(),
  data = NULL,
  diagnostic_plots = TRUE,
  alpha = 0.05,
  adjust = "sidak",
```

```
  type = "response",
  intro_text = TRUE,
  dispersion_test = TRUE,
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  influence_threshold = 2,
  ...
)
```

Arguments

`formula`	A formula specifying the model to be fitted. More response variables can be added using - or + (e.g., `response1 + response2 ~ predictor`) to do a sequential GLM for each response parameter.
`family`	The error distribution and link function to be used in the model (default: gaussian()). This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions.)
`data`	A data frame containing the variables in the model.
`diagnostic_plots`	Logical. If `TRUE`, plots are included in the output files.
`alpha`	Numeric. Significance level for tests. Default is `0.05`.
`adjust`	Character string specifying the method used to adjust p-values for multiple comparisons. Available methods include: **"tukey"** Tukey's Honest Significant Difference method **"sidak"** Sidak correction **"bonferroni"** Bonferroni correction **"none"** No adjustment **"fdr"** False Discovery Rate adjustment Default is `"sidak"`.
`type`	Character string specifying the scale of emmeans post hoc results: `"response"` (back-transformed to original units, e.g. probabilities, counts) or `"link"` (on the linear predictor scale, e.g. log-odds). Default is `"response"`.
`intro_text`	Logical. If `TRUE`, includes a short explanation about GLM assumptions in the output file.
`dispersion_test`	Logical. If `TRUE`, includes a dispersion diagnostic section in the output: a DHARMa simulation-based test for overdispersion (Poisson/Binomial), the quasi-dispersion parameter (quasi-families), or a note explaining why the test is skipped (Bernoulli data). Default is `TRUE`.
`output_type`	Character string specifying the output format. Default is `"default"`.

- "default": Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use print(result) or plot(result) to display the returned object.
- "console": Forces immediate printing to the console regardless of object assignment.
- "pdf", "word", "excel": Saves results to a file of the corresponding format. See save_as, save_in_wdir, and open_generated_files for file path and opening behavior.
- "rmd": Stores the raw markdown string inside the returned object for use in R Markdown documents.

save_as Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in tempdir(). If only a directory is specified (providing an existing directory with trailing slash), the file is named "dataname_glm_output" in that directory. If an extension is provided the output format specified with option "output_type" will be overruled. Defaults to file.path(tempdir(), "dataname_summary.pdf").

save_in_wdir Logical. If TRUE, saves the file in the working directory. Default is FALSE, this avoid unintended changes to the global environment. If save_as location is specified save_in_wdir is overwritten by save_as.

close_generated_files
Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default FALSE. **WARNING:** Always save your work before using this option!!

open_generated_files
Logical. Whether to open the generated output files after creation. Defaults to TRUE in an interactive R session and FALSE otherwise (e.g. in scripts or automated pipelines). Set to TRUE or FALSE to override this behaviour explicitly.

influence_threshold
Numeric multiplier for the leverage threshold. Observations with hat values exceeding influence_threshold * mean(hat values) are flagged as high-leverage points. Default is 2, a common rule of thumb.

... Additional arguments passed to glm().

Details

The function first checks if all specified variables are present in the data and ensures that the response variable is numeric.

It fits a Generalized Linear Model (GLM) using the specified formula, family, and data. Model diagnostics are performed with DHARMa (simulation-based residual checks including a KS test, dispersion test, and outlier test). High-leverage observations are flagged using hat values.

Significance of each predictor is assessed via Type II Analysis of Deviance (stats::drop1()). If significant effects are found, post hoc pairwise comparisons are performed using estimated marginal means from emmeans() with the chosen p-value adjustment method (default: Sidak). When complete separation is detected, the function falls back to likelihood ratio test (LRT) based pairwise comparisons, which are robust to separation.

More response variables can be added using + (e.g., `response1 + response2 ~ predictor`) to fit a sequential GLM for each response variable, captured in one output file.

Outputs can be generated in multiple formats ("pdf", "word", "excel" and "rmd") as specified by `output_type`. The function also closes any open 'Word' files to avoid conflicts when generating 'Word' documents. If `output_type = "rmd"` is used it is advised to use it in a chunk with {r, echo=FALSE, results='asis'}

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

An object of class 'f_glm' (a named list, one entry per response variable) containing:

model The fitted `glm` object.

summary Output of `summary(glm_fit)`.

drop1 Type II Analysis of Deviance table from `stats::drop1()`.

diagnostics DHARMa residual checks and hat-value based leverage diagnostics.

posthoc Estimated marginal means, pairwise comparisons, CLD letters, and summary table.

sep_flag Logical indicating whether complete separation was detected.

lrt_pct_explained McFadden's Pseudo-R^2.

Using the option `output_type`, it can also generate output in the form of: R Markdown code, 'Word', 'pdf', or 'Excel' files. Includes print and plot methods for 'f_glm' objects.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# GLM Binomial example with output to console
mtcars_mod <- mtcars
mtcars_mod$cyl <- as.factor(mtcars_mod$cyl)

glm_bin <- f_glm(vs ~ cyl,
                 family = binomial,
                 data = mtcars_mod,
```

```
                  output_type = "default")
print(glm_bin)

# GLM Binomial example with output to MS Word file
glm_bin_word <- f_glm(vs ~ cyl,
                  family = binomial,
                  data = mtcars_mod,
                  output_type = "word"
                  )

# GLM Poisson example with output to rmd text
data(warpbreaks)

glm_pos <- f_glm(breaks ~ wool + tension,
                  data = warpbreaks,
                  family = poisson(link = "log"),
                  intro_text = FALSE,
                  output_type = "rmd")
cat(glm_pos$rmd)
```

---

f_hist *Plot a Histogram with an Overlaid Normal Curve*

Description

This function creates a histogram of the provided data and overlays it with a normal distribution curve.

Usage

```
f_hist(
  data,
  main = NULL,
  xlab = NULL,
  probability = TRUE,
  col = "white",
  border = "black",
  line_col = "red",
  save_png = FALSE,
  open_png = TRUE,
  save_as = NULL,
  save_in_wdir = FALSE,
  width = 8,
  height = 7,
  units = "in",
  res = 300,
  ...
)
```

Arguments

data	A numeric vector of data values to be plotted.
main	A character string specifying the title of the histogram. Default is "Histogram with Normal Curve".
xlab	A character string specifying the label for the x-axis. Default is the name of the data variable.
probability	A logical value indicating whether to plot a probability or frequency histogram. Default is TRUE.
col	A character string specifying the fill color of the histogram bars. Default is "white".
border	A character string specifying the color of the histogram bar borders. Default is "black".
line_col	A character string specifying the color of the normal curve line. Default is "red".
save_png	A logical value default FALSE, if TRUE a png file is saved under the name of the data of under the specified file name.
open_png	Logical. If TRUE, opens generated png files.
save_as	Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in tempdir(). If only a directory is specified (providing an existing directory with trailing slash), the file is named "data_name_histogram.png" in that directory. Defaults to file.path(tempdir(), "data_name_histogram.png").
save_in_wdir	Logical. If TRUE, saves the file in the working directory. Default is FALSE, this avoid unintended changes to the global environment. If save_as location is specified save_in_wdir is overwritten by save_as.
width	Numeric, png figure width default 8 inch.
height	Numeric, png figure height default 7 inch.
units	Character string, png figure units default "in" = inch, other options are: "px" = Pixels, "cm" centimeters, "mm" millimeters.
res	Numeric, png figure resolution default 300 dpi.
...	Additional arguments to be passed to the hist function.

Details

The function first captures the name of the input variable for labeling purposes. It then calculates a sequence of x-values and corresponding y-values for a normal distribution based on the mean and standard deviation of the data. The histogram is plotted with specified aesthetics, and a normal curve is overlaid. To increase resolution you can use png(...,res = 600) or the 'RStudio' chunk setting, e.g. dpi=600.

Value

A histogram plot is created and the function returns this as a recordedplot.

Author(s)

Sander H. van Delden <plantmind@proton.me>

See Also

hist, dnorm, lines

Examples

```
# Example usage:
set.seed(123)
sample_data <- rnorm(100)
f_hist(sample_data)
```

---

f_kruskal_test *Perform multiple Kruskal-Wallis tests with a user-friendly output file, do data inspection and Dunn's test (of 'rstatix') as post hoc.*

Description

Performs the Kruskal-Wallis rank sum test to assess whether there are statistically significant differences in the distributions (mean ranks) of three or more independent groups. It provides detailed outputs, including plots, assumption checks, and post hoc analyses using Dunn's test. Results can be saved in various formats ('pdf', 'Word', 'Excel', or console only) with customizable output options.

Usage

```
f_kruskal_test(
  formula,
  data = NULL,
  plot = TRUE,
  alpha = 0.05,
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  intro_text = TRUE,
  adjust = "bonferroni",
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  ...
)
```

Arguments

formula	A formula specifying the response and predictor variable (e.g., response ~ predictor). more response variables and predictors can be added using - or + (e.g., response1 + response2 ~ predictor1 + predictor2). The function iterates through these combinations or response and predictors, because the Kruskal-Wallis test itself only allows one response and one predictor combination to be tested simultaneously.
data	A data.frame containing the variables referenced in the formula.
plot	Logical. If TRUE, generates plots (e.g., density plots and boxplots) in the output files. Default is TRUE.
alpha	Numeric. The significance level for the Kruskal-Wallis test and Dunn's test. Default is 0.05.
output_type	Character string specifying the output format. Default is "default". • "default": Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use print(result) or plot(result) to display the returned object. • "console": Forces immediate printing to the console regardless of object assignment. • "pdf", "word", "excel": Saves results to a file of the corresponding format. See save_as, save_in_wdir, and open_generated_files for file path and opening behavior. • "rmd": Stores the raw markdown string inside the returned object for use in R Markdown documents.
save_as	Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in tempdir(). If only a directory is specified (providing an existing directory with trailing slash), the file is named "dataname_Kruskal_Wallis_output" in that directory. If an extension is provided the output format specified with option "output_type" will be overruled. Defaults to file.path(tempdir(), "dataname_summary.pdf").
save_in_wdir	Logical. If TRUE, saves the file in the working directory. Default is FALSE, this avoid unintended changes to the global environment. If save_as location is specified save_in_wdir is overwritten by save_as.
intro_text	Logical. If TRUE, includes a section about Kruskal-Wallis test assumptions in the output document. Default is TRUE.
adjust	Character string. Adjustment method for pairwise comparisons in Dunn's test. Options include "holm", "hommel", "bonferroni", "hochberg", "bh", "by", "fdr" or "none". Default is "bonferroni", if you don't want to adjust the p value (not recommended), use adjust = "none".
close_generated_files	Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default FALSE. **WARNING:** Always save your work before using this option!!

`open_generated_files`
: Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.

`...`
: Additional arguments forwarded to `kruskal.test`. The arguments `subset` and `na.action` are honored: when supplied, they are applied via `model.frame` so that the descriptive summary table, density plot, boxplot, Dunn's post hoc test and the Kruskal-Wallis test itself all see the exact same row set.

Details

This function offers a comprehensive workflow for non-parametric analysis using the Kruskal-Wallis test:

- Assumption Checks: Optionally includes a summary of assumptions in the output.
- Visualization: Generates density plots and boxplots to visualize group distributions.
- Post hoc Analysis: Conducts Dunn's test with specified correction methods if significant differences are found.

Output files are generated in the format specified by `output_type =` and saved to the working directory, options are `"pdf"`, `"word"` or `"excel"`. If `output_type = "rmd"` is used it is advised to use it in a chunk with {r, echo=FALSE, results='asis'}

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

An object of class 'f_kruskal_test' (a named list, one entry per response-predictor combination) containing:

kruskal.test The `htest` object from `kruskal.test()`.

dunn_test Data frame of pairwise Dunn's test results from `rstatix::dunn_test()`.

summary_table Descriptive statistics with compact letter display (Letters column).

alpha The significance level used.

DunnTest_adjust The p-value adjustment method used.

distributions ggplot density plot (if `plot = TRUE`).

Boxplot ggplot boxplot with CLD letters (if `plot = TRUE`).

Using the option `output_type`, it can also generate output in the form of: R Markdown code, 'Word', 'pdf', or 'Excel' files. Includes print and plot methods for 'f_kruskal_test' objects.

Multiple Testing Across Response Variables

When several response variables are analysed in a single call (e.g. `y1 + y2 + y3 ~ treatment`), each Kruskal-Wallis test is an independent null-hypothesis test at level `alpha`. The post hoc adjustment (e.g. `adjust = "bonferroni"`) only controls the family-wise error rate **within** one test (across pairwise Dunn comparisons for that response). It does **not** protect against the inflation of Type I error **across** the set of responses.

Practical implication: With k independent response variables all tested at $\alpha = 0.05$, the probability of obtaining at least one false positive is $1 - (1 - 0.05)^k$, which reaches ~40% for $k = 10$.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Example usage:
data(iris)

# Perform Kruskal-Wallis test on Sepal.Length and Sepal.Width by Species
# with "holm" correction for posthoc dunn_test, without showing the output.
output <- f_kruskal_test(
               Sepal.Width + Sepal.Length ~ Species,
               data = iris,
               plot = FALSE,
               output_type = "word",
               adjust = "holm"
               )

# Save Kruskal-Wallis test and posthoc to Excel sheets: Sepal.Width and Sepal.Length.
f_kruskal_out <- f_kruskal_test(
                     Sepal.Width + Sepal.Length ~ Species,
                     data = iris,
                     plot = FALSE,
                     output_type = "excel",
                     adjust = "holm"
                     )
```

f_lmer *Fit a linear mixed model with* `lme4::lmer()` *including assumption checks, diagnostics, R-squared and post hoc tests.*

Description

Fits a linear mixed-effects model using `lme4::lmer()` (with p-values from `lmerTest`) and produces a fully-formatted report containing the fixed-effects table, random-effects variance components, model-fit indices (AIC, BIC, logLik, marginal & conditional R^2), residual and BLUP diagnostics, convergence / singular-fit warnings, and post hoc comparisons (`emmeans`) on factor fixed effects. Results can be returned to the console or written to 'pdf', 'Word' or 'Excel'.

Usage

```
f_lmer(
  formula,
  data = NULL,
  REML = TRUE,
  ddf = "Satterthwaite",
  alpha = 0.05,
  adjust = "sidak",
  norm_plots = TRUE,
  post_hoc = TRUE,
  intro_text = TRUE,
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  ...
)
```

Arguments

`formula`	A two-sided formula passed to `lme4::lmer()`, e.g. `y ~ treatment + time + (1 | subject)` or `y ~ treatment * time + (1 + time | subject)`. The right-hand side must contain at least one random-effects term in the `(varying | grouping)` syntax. See *Details* for a guide to reading the random-effects syntax in study-design terms. More than one response variable can be supplied on the left-hand side using `+` (e.g. `y1 + y2 ~ treatment + (1 | subject)`). A separate model is then fit for each response, sharing the same right-hand side, and a multiple-testing warning is added to the report. See the *Multiple Testing Across Response Variables* section.
`data`	A data frame containing the variables in the model.

REML — Logical. If TRUE (default), the model is fit with restricted maximum likelihood, which gives less biased variance component estimates and is the appropriate choice for inference on fixed effects with Kenward-Roger or Satterthwaite degrees of freedom. Set to FALSE only when comparing nested models that differ in their fixed-effects structure.

ddf — Character. Method for computing denominator degrees of freedom for fixed-effects p-values. One of:

- "Satterthwaite" (default) Fast and accurate for most designs. Provided by lmerTest.
- "Kenward-Roger" Considered the gold standard, especially for small samples and unbalanced designs. Slower, and requires the pbkrtest package.
- "lme4" No p-values; only t-statistics are reported. Equivalent to a plain lme4::lmer() summary.

alpha — Numeric. Significance level for the fixed-effects table and the post hoc tests. Default is 0.05.

adjust — Character. Method used to adjust p-values for multiple pairwise comparisons in the post hoc step (passed to emmeans::emmeans()). One of "sidak" (default), "tukey", "bonferroni", "fdr", "none".

norm_plots — Logical. If TRUE (default), diagnostic plots (residuals vs fitted, Q-Q of level-1 residuals, Q-Q of random-effect BLUPs, scale-location) are included in the output.

post_hoc — Logical. If TRUE (default), runs emmeans::emmeans() pairwise comparisons **only when** the linear mixed model finds a significant fixed-effect term (ANOVA p-value below alpha). The post hoc is performed on each significant factor fixed-effect term separately. Numeric covariates are skipped because their slope is already reported in the fixed-effects coefficient table; pairwise contrasts are not meaningful for a continuous predictor. If no fixed-effect term is significant, no post hoc is run.

intro_text — Logical. If TRUE (default), prepends an explanation of LMM assumptions and the random-effects syntax linked to study design.

output_type — Character. Output format. One of:

- "default": returns the f_lmer object; auto-prints if unassigned.
- "console": forces immediate printing.
- "pdf", "word", "excel": writes a file.
- "rmd": stores the raw markdown string in the returned object for embedding in an R Markdown chunk with {r, echo=FALSE, results='asis'}.

save_as — Character. Output file path. See f_aov for the resolution rules; default is file.path(tempdir(), "<dataname>_lmer_output.<ext>").

save_in_wdir — Logical. If TRUE, save in the working directory instead of tempdir(). Default FALSE.

close_generated_files — Logical. Closes any open Word or Excel files before writing. Cross-platform (Windows taskkill, macOS / Linux pkill). Default FALSE. **WARNING:** save your work first.

`open_generated_files`
: Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.

`...`
: Additional arguments forwarded to `lmer`. The arguments `subset`, `na.action`, and `weights` are handled specially: when supplied, they are applied via `model.frame` **once** before the per-response loop, so every response in a multi-response call is fitted on the identical row set. Other `lmer()` arguments (e.g. `control = lmerControl(...)`, `contrasts`, `offset`) are forwarded unchanged on every fit.

Details

What is a linear mixed model?
A linear mixed model (LMM) extends ordinary regression / ANOVA by allowing two kinds of effects:

- **Fixed effects** - factors you actively manipulated or whose specific levels you care about (treatment, dose, time, genotype). Reported as estimates with confidence intervals.
- **Random effects** - grouping structure that creates non-independence in your data but whose levels are a random sample from a larger population (subjects measured repeatedly, plots within fields, observers, batches). Reported as variance components.

Use an LMM whenever observations share something that makes them more alike than two random observations from the dataset. Ignoring such grouping (running a plain `aov` or `lm`) is **pseudoreplication**, i.e. treating non-independent observations as if they were independent: standard errors shrink, p-values shrink, false positives explode.

Vocabulary.
Before going further, a few terms used throughout the report:

- **Subject** - the experimental unit that is measured repeatedly (a person, animal, pot and plot, cell line); in `lme4` syntax it is the grouping factor on the right of the `|`, e.g. `(1 | subject)`.
- **Within-subject factor** - a predictor whose levels vary within the same subject (time in a longitudinal study, treatment in a cross-over study).
- **Between-subject factor** - a predictor whose levels vary across subjects but are constant within a subject (sex, genotype, treatment arm in a parallel-groups trial). Both within- and between-subject factors are **fixed** effects.
- **BLUP** - *Best Linear Unbiased Predictor*. The model's estimate of the random-effect value for each subject (e.g. how much a particular subject deviates from the population intercept). BLUPs are checked for normality just like residuals.
- **ICC** - *intraclass correlation coefficient*. The share of total variance attributable to between-group differences. ICC = 0 means the grouping factor is irrelevant; ICC = 1 means observations within a group are identical.
- **REML** - *restricted maximum likelihood*. The default fitting method for variance components; gives less biased estimates than ordinary maximum likelihood.
- **Satterthwaite / Kenward-Roger** - methods to approximate the denominator degrees of freedom for fixed-effect p-values, since there is no exact df in an LMM.

Reading the `(1 | group)` syntax.
Every random-effects term has the form `( <varying> | <group> )`. The bar reads as "varies by". The grouping factor on the right is what creates the non-independence. The left side is what is allowed to differ between groups. Common patterns:

- `(1 | subject)` - random intercept per subject (each subject has its own baseline). Repeated measures, longitudinal data.
- `(1 | field)` - randomised block design or multi-site trial; one intercept per block.
- `(1 | field/plot)` - `plot` nested in `field`; equivalent to `(1|field) + (1|field:plot)`. Split-plot or hierarchical sampling.
- `(1 + time | subject)` - random intercept and random slope of `time` per subject. Subjects differ both in baseline and in how fast they change. Growth curves.
- `(1 | subject) + (1 | observer)` - crossed random effects: every observer can rate every subject. Inter-rater designs.

Rule of thumb: if you can answer "if I duplicated this experiment, would I draw new levels of this factor?" with *yes*, it belongs on the right of a `|`. If you would re-use the exact same levels (e.g. control vs treated) it is a fixed effect.

When to use a linear mixed model.
The most common reason is a **repeated-measures design**, in which the same experimental units are measured on more than one occasion or under more than one treatment. Compared with a between-groups design analysed by plain ANOVA this gives two real advantages: fewer experimental units are needed (each subject acts as its own control, removing between-subject variation from the comparison) and individual differences cannot bias the treatment groups (in a cross-over design every subject receives every treatment). Two canonical examples:

- **Longitudinal study** - same subjects measured at several time points: `y ~ time + (1 | subject)`. If subjects also differ in how fast they change, add a random slope: `y ~ time + (1 + time | subject)`.
- **Cross-over design** - every subject receives every treatment in sequence: `y ~ treatment + (1 | subject)`. If carry-over between periods is a concern, add `period` as a fixed effect.

LMMs also apply to non-repeated structures that still create non-independence: randomised block designs, split-plot trials, multi-site studies, inter-rater designs.

Assumptions of a linear mixed model:

1. Linearity in the parameters of the fixed-effects part.
2. Independence of observations *conditional on* the random effects. If structure remains (e.g. temporal autocorrelation), more random effects or a correlation structure are needed.
3. Normality of level-1 residuals (Q-Q plot of `residuals(m)`).
4. Normality of the random-effect BLUPs (Q-Q plot of `ranef(m)`). **This is the assumption most users forget.**
5. Homoscedasticity: residual variance roughly constant across fitted values and across grouping levels.
6. At least `~5` levels of each grouping factor; with `3-4` levels it is usually better to treat the factor as fixed.

If Levene's test or the Shapiro-Wilk tests on residuals or BLUPs indicate a violation, the report adds a *Recommendations for Heteroscedasticity and/or non-normal residuals* section after the diagnostics with concrete next steps (generalised mixed model, transformation).

Convergence and singular fits.
`f_lmer` surfaces `lme4` convergence warnings and the "boundary (singular) fit" message prominently in the output. A singular fit usually means the random-effects structure is too complex for the data (often a random slope with too few levels) - simplify the model before interpreting results.

This function requires Pandoc (>= 1.12.3) for `pdf`, `word` and `rmd` output. See `f_aov` for installation notes.

Value

An object of class `f_lmer`: a named list containing the fitted `lmerModLmerTest` model, the ANOVA-style fixed-effects table, the variance components and ICC, the R^2 values, the observed descriptives table (raw-data n, mean, sd, se, min, Q1, median, Q3, max grouped by the categorical fixed-effect predictors), post hoc results (if any), diagnostic plots, and convergence diagnostics. When more than one response variable is supplied on the left-hand side, these elements are nested one level deep under each response name, e.g. `out$y1$fixed_effects`, `out$y2$fixed_effects`. When `output_type = "rmd"` the markdown string is stored in `$rmd`.

Multiple Testing Across Response Variables

When several response variables are analysed in a single call (e.g. `y1 + y2 + y3 ~ treatment + (1 | subject)`), each linear mixed model is an independent null-hypothesis test at level `alpha`. The post hoc adjustments (`adjust = "sidak"`, `"tukey"`, etc.) only control the family-wise error rate **within** one model (across pairwise contrasts for that response). They do **not** protect against the inflation of Type I error **across** the set of responses.

Practical implication: With k independent response variables all tested at $\alpha = 0.05$, the probability of obtaining at least one false positive is $1 - (1 - 0.05)^k$, which reaches ~40% for $k = 10$.

When this matters: The risk is highest in exploratory studies where many responses are screened simultaneously without a clear a priori hypothesis for each one. It is less of a concern when each response is a pre-specified primary outcome with its own biological rationale.

Possible remedies:

- **Bonferroni correction across responses:** use `alpha = 0.05 / k` where `k` is the number of response variables. Conservative but simple.
- **False Discovery Rate (FDR):** apply `p.adjust(p_values, method = "fdr")` to the vector of per-response fixed-effect p-values after the fact.
- **Multivariate model:** if the responses are correlated and you want a single omnibus test, fit a joint multivariate mixed model (e.g. `MCMCglmm`, `brms`) before interpreting individual responses.
- **Pre-registration:** declare primary vs. exploratory responses before data collection to justify differential correction thresholds.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# sleepstudy: reaction time vs days of sleep deprivation,
# repeated measures within Subject (ships with lme4).
data(sleepstudy, package = "lme4")

# 1) Random intercept per subject - the simplest mixed model.
#    Each subject has its own baseline reaction time; the fixed
#    effect of Days is the average slope across subjects.
#    With output_type = "default" (the default), the result auto-
#    prints if not assigned, so no print() call is needed.
f_lmer_out <- f_lmer(Reaction ~ Days + (1 | Subject),
                     data = sleepstudy)

# Re-print the stored result and show the diagnostic plots.
print(f_lmer_out)
plot(f_lmer_out)

# 2) Random intercept AND random slope of Days per subject,
#    fitted with Kenward-Roger denominator df, saved to MS Word.
f_lmer(Reaction ~ Days + (1 + Days | Subject),
       data = sleepstudy,
       ddf  = "Kenward-Roger",
       output_type = "word"
       )

# 3) A factor fixed effect triggers a post hoc test.
#    Bin Days into three sleep-deprivation phases so that the
#    fixed effect is categorical and emmeans pairwise comparisons
#    with a compact letter display are produced automatically.
sleepstudy$Phase <- cut(sleepstudy$Days,
                        breaks = c(-Inf, 2, 6, Inf),
                        labels = c("early", "mid", "late"))
f_lmer(Reaction ~ Phase + (1 | Subject),
       data = sleepstudy,
       adjust = "tukey")

# 4) A minimal report: suppress the intro text and the diagnostic
#    plots, and save it directly to MS Word. Useful when embedding
#    many models in one document or when you only need the tables.
f_lmer(Reaction ~ Days + (1 | Subject),
       data = sleepstudy,
       intro_text = FALSE,
       norm_plots = FALSE,
       output_type = "word"
       )

# 5) Get the raw markdown back for embedding in an R Markdown
#    document. Use it inside a chunk with results = 'asis'.
f_lmer_rmd_out <- f_lmer(Reaction ~ Days + (1 | Subject),
                         data = sleepstudy,
                         output_type = "rmd")
cat(f_lmer_rmd_out$rmd)
```

```
# 6) Two response variables analysed in one call. A separate model
#    is fit for each, sharing the same right-hand side. The results
#    are nested under each response name.
sleepstudy$Reaction2 <- sleepstudy$Reaction + rnorm(nrow(sleepstudy), 0, 5)
multi_out <- f_lmer(Reaction + Reaction2 ~ Days + (1 | Subject),
                    data = sleepstudy,
                    intro_text = FALSE,
                    norm_plots = FALSE)
multi_out$Reaction$fixed_effects
multi_out$Reaction2$fixed_effects
```

`f_load_packages` *Install and Load Multiple R Packages*

Description

Checks if the specified packages are installed. If not, it installs them and then loads them into the global R session.

Usage

```
f_load_packages(...)
```

Arguments

`...`	Unquoted or quoted names of packages to be installed and loaded. These should be valid package names available on CRAN.

Details

The function takes a list or vector indicating package names, installs any that are missing, and loads all specified packages into the global environment of the R session. It uses `requireNamespace()` to check for installation and `library()` to load the packages.

Value

None. The function is called for its side effects of installing and loading packages.

Author(s)

Sander H. van Delden <plantmind@proton.me>

f_long *Transform 'Wide' (Excel) data to 'Long' (R) format*

Description

This function converts "wide" data (e.g. Excel tables) into a "long" list format. This is the essential first step to prepare your data for analysis and plotting in R.

Usage

```
f_long(
  data,
  measure_columns = NULL,
  keep_cols = NULL,
  category_name = "name",
  value_name = "value",
  category_labels = NULL,
  ...
)
```

Arguments

data	The input data frame (e.g., from read_excel).
measure_columns	(Optional) The columns containing your numeric measurements. These values are often the response variables, i.e. will end up on the Y-axis. • Use column names: c("OD_t0", "OD_t1") • Use column numbers: 2:5 • Use helpers: starts_with("Measure") If NULL (default), the function will pivot ALL columns except those in keep_cols.
keep_cols	(Optional) The columns that identify your samples (IDs). E.g., "SampleID", "PatientID", "Treatment", "Student number". These are repeated for every measurement. *If left empty, all non-measured columns are kept.* Important: If measure_columns is NULL, you MUST specify keep_cols.
category_name	Name for the new column containing the headers. Default is "name". Choose something logical like "Timepoints", "Genes", or "Condition".
value_name	Name for the new column containing the numbers. Default is "value". Choose something logical like "Absorbance", "Ct_Value", or "Weight".
category_labels	(Optional) A character vector of new, readable names for your categories, i.e. the measure_columns that you entered. **Note:** The order must match the order of measure_columns exactly. Useful for renaming "t0_raw" to "Start" instantly.
...	Additional arguments passed to tidyr::pivot_longer. E.g., values_drop_na = TRUE to remove empty cells immediately.

Details

Research data in Excel or output from lab instruments often contains measurements side-by-side (in columns). Many R functions require measurements in a single column (rows). ‘f_long‘ performs this translation for you.

It performs three actions in one go: 1. Selects your measurement columns (‘measure_columns‘). 2. Keeps your important ID columns (‘keep_cols‘) and removes the rest. 3. (Optional) Renames cryptic column headers into readable labels (‘category_labels‘).

Value

A "Tidy" data frame (tibble) of class `f_long`.

Note

The custom class and attributes (`f_long_value`, `f_long_category`) are used by the `plot` and `summary` methods. Be aware that most **dplyr** or **tidyr** operations (e.g., `filter`, `mutate`) will silently strip these attributes. If that happens, use `f_scan` or `f_summary` directly with explicit column names instead.

Examples

```
# --- Example 1: Using the 'iris' dataset ---
# Scenario: The iris dataset looks clean, but it is actually "Wide".
# It has 4 columns of measurements side-by-side.
# To compare Sepal Length vs Width in a plot, we must stack them.

head(iris)

# Reshape: Combine Length and Width into one column and plot the data.
iris_long <- f_long(
  data = iris,
  measure_columns = c("Sepal.Length", "Sepal.Width"),
  keep_cols = "Species",
  category_name = "Sepal_Dimension",    # Describes the grouping (What did we measure?)
  value_name = "Size_cm",               # Describes the value (What is the number?)
  category_labels=c("Length", "Width")  # New category labels
  )

 head(iris_long)

 # Plot the data using f_scan
 plot(iris_long)

 # Make a f_summary table of iris_long
 summary(iris_long)

# --- Example 2: Using the 'airquality' dataset ---
# Scenario: Pivot daily measurements of Wind and Temperature over time.

head(airquality)
```

```
weather_long <- f_long(
  data = airquality,
  measure_columns = c("Wind", "Temp"),
  keep_cols = c("Month", "Day"),
  category_name = "Climate_Parameter", # Descriptive name
  value_name = "Reading_Value",        # Generic name (since units differ: mph vs F)
  values_drop_na = TRUE
)

head(weather_long)
```

`f_model_compare` *Compare Two Statistical Models*

Description

Compares two statistical models by calculating key metrics such as AIC, BIC, log-likelihood, R^2, and others. Supports comparison of nested models using ANOVA tests.

Usage

```
f_model_compare(
  model1,
  model2,
  nested = NULL,
  model1_name = NULL,
  model2_name = NULL,
  digits = 3
)
```

Arguments

`model1`	The first model object. Supported classes include: `"lm"`, `"glm"`, `"aov"`, `"lmerMod"`, `"glmerMod"`, and `"nls"`.
`model2`	The second model object. Supported classes include: `"lm"`, `"glm"`, `"aov"`, `"lmerMod"`, `"glmerMod"`, and `"nls"`.
`nested`	Logical. If `TRUE`, assumes the models are nested and performs an ANOVA comparison. If `NULL` (default), the function attempts to automatically determine if the models are nested.
`model1_name`	Optional character string. A custom name for model1 in the output. If `NULL` (default), the function uses `deparse(substitute(model1))`.
`model2_name`	Optional character string. A custom name for model2 in the output. If `NULL` (default), the function uses `deparse(substitute(model2))`.
`digits`	Integer. The number of decimal places to round the output metrics. Defaults to `3`.

Details

Calculate various metrics to assess model fit:

- **AIC/BIC:** Lower values indicate better fit.
- **Log-Likelihood:** Higher values (less negative) indicate better fit.
- R^2**:** Proportion of variance explained by the model.
- **Adjusted** R^2**:** R^2 penalized for the number of parameters (for linear models).
- **Nagelkerke** R^2**:** A pseudo-R^2 for generalized linear models (GLMs).
- **Marginal/Conditional** R^2**:** For mixed models, marginal R^2 reflects fixed effects, while conditional R^2 includes random effects.
- **Sigma:** Residual standard error.
- **Deviance:** Model deviance.
- **SSE:** Sum of squared errors.
- **Parameters (df):** Number of model parameters.
- **Residual df:** Residual degrees of freedom.

When nested models are detected or specified, model1 is always treated as the simpler model (fewer parameters). If the user passes the complex model first, the function automatically swaps them and issues a message.

If the models are nested, an ANOVA test is performed to compare them, and a p-value is provided to assess whether the more complex model significantly improves fit.

Value

A list of class "f_model_comparison" containing:

`model1_name` The name of the first model (always the simpler model when nested).

`model2_name` The name of the second model (always the more complex model when nested).

`model1_class` The class of the first model.

`model2_class` The class of the second model.

`metrics_table` A data frame summarizing metrics for both models, their differences, and (if applicable) the ANOVA p-value.

`formatted_metrics_table` A formatted version of the metrics table for printing.

`anova_comparison` The ANOVA comparison results if the models are nested and an ANOVA test was performed.

`nested` Logical indicating whether the models were treated as nested.

`swapped` Logical indicating whether the model order was swapped to ensure model1 is the simpler model.

Supported Model Classes

The function supports the following model classes:

- Linear models ("lm")
- Generalized linear models ("glm")
- Analysis of variance models ("aov")
- Linear mixed models ("lmerMod")
- Generalized linear mixed models ("glmerMod")
- Nonlinear least squares models ("nls")

Note: Multi-stratum AOV models (fitted with `Error()`) are not supported and will produce a warning.

Note

- The function supports a variety of model types but may issue warnings if unsupported or partially supported classes are used.
- For GLMs, Nagelkerke's R^2 is used as a pseudo-R^2 approximation, computed from the model's null deviance to avoid refitting a null model.
- For mixed models, the function relies on the 'r.squaredGLMM' function from the 'MuMIn' package for R^2 calculation.
- For NLS models, R^2 is provided for convenience but should be interpreted with caution as it does not have the same statistical properties as in linear models.
- The idea of this function (not the code), I got from Dustin Fife's function 'model.comparison' in the super cool 'flexplot package'.

Author(s)

Sander H. van Delden <plantmind@proton.me>

See Also

`AIC`, `BIC`, `anova`, `logLik`, `r.squaredGLMM`

Examples

```
# Example with linear models.
model1 <- lm(mpg ~ wt, data = mtcars)
model2 <- lm(mpg ~ wt + hp, data = mtcars)
comparison <- f_model_compare(model1, model2)
print(comparison)

# Example with GLMs.

model1 <- glm(am ~ wt, data = mtcars, family = binomial)
model2 <- glm(am ~ wt + hp, data = mtcars, family = binomial)
comparison <- f_model_compare(model1, model2)
```

```
print(comparison)


# Models can be passed in any order - the function auto-swaps if needed.
complex <- lm(mpg ~ wt + hp + qsec, data = mtcars)
simple  <- lm(mpg ~ wt, data = mtcars)
comparison <- f_model_compare(complex, simple)
# model1 will be "simple", model2 will be "complex" in the output

# Example with custom model names (useful when calling from wrapper functions).
comparison <- f_model_compare(model1, model2,
                              model1_name = "Weight only",
                              model2_name = "Weight + Horsepower")
print(comparison)
```

`f_open_file` *Open a File with the Default Application*

Description

Opens a specified file using the default application associated with its file type. It automatically detects the operating system (Windows, Linux, or macOS) and uses the appropriate command to open the file.

Usage

```
f_open_file(filepath)
```

Arguments

`filepath`	A character string specifying the path to the file to be opened. The path can be absolute or relative.

Details

- On **Windows**, the `f_open_file()` function uses `shell.exec()` to open the file. - On **Linux**, it uses `xdg-open` via the `system()` function. - On **macOS**, it uses `open` via the `system()` function.

If an unsupported operating system is detected, the function will throw a message.

Value

Does not return a value; it is called for its side effect of opening a file.

Author(s)

Sander H. van Delden <plantmind@proton.me>

See Also

[shell.exec()], [system()]

Examples

```
# NOTE: The use of "if(interactive())" prevents this example from running
# during automated CRAN checks. This is necessary because the example
# opens a file, a behavior restricted by CRAN policies for automated
# testing.You don't need to use "if(interactive())" in your own scripts.
if(interactive()) {
# Open a PDF file.
f_open_file("example.pdf")

# Open an image file.
f_open_file("image.png")

# Open a text file.
f_open_file("document.txt")
}
```

`f_outliers` *Identify Outliers within Groups using Tukey's Fences*

Description

'f_outliers()' scans numerical column(s) for outliers based on the Interquartile Range (IQR) method. It can detect outliers across the entire dataset or within specified subgroups.

It returns a dataframe containing only the outlier rows, preserving the original data structure and adding a `row_id` column for traceability.

Usage

```
f_outliers(x, ...)

## S3 method for class 'numeric'
f_outliers(x, ...)

## S3 method for class 'integer'
f_outliers(x, ...)

## S3 method for class 'formula'
f_outliers(formula, data, ...)

## S3 method for class 'data.frame'
f_outliers(
  x,
```

```
  columns = NULL,
  group_vars = NULL,
  id_var = NULL,
  coef = 1.5,
  digits = NULL,
  export_to_excel = FALSE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  save_as = NULL,
  save_in_wdir = FALSE,
  check_input = TRUE,
  digits_excel = NULL,
  allow_integer_decimal_mix = FALSE,
  ...
)
```

Arguments

x	A data.frame or formula (dispatches to the right method).
...	Further arguments forwarded to f_outliers.data.frame.
formula	A formula specifying the columns (right hand side) to be checked per subgroup(s) (left hand side). More columns or groups can be added using - or + (e.g., col1 + col2 ~ group1 + group2) to do a sequential analysis for each column parameter.
data	A vector, data.frame, data.table, or tibble.
columns	The numerical columns to analyze if no formula is used. Can be entered as a single character string (e.g., "weight") or as a character vector c("weight", "length"). When omitted, defaults to all numeric columns in data (excluding any columns named in group_vars or id_var).
group_vars	A character vector specifying the grouping variables in data If NULL (default), the entire dataset is treated as one group (e.g., c("species", "fertilizer")).
id_var	(Optional) A character string naming a user-specific ID columns (e.g., "EmployeeID"). If provided, this columns is placed at the start of the output for easy identification.
coef	A number indicating the IQR multiplier. Default is 1.5 (standard Tukey fence). Common values: • 1.5: Standard outliers (approximately +/- 2.7 SD in normal distribution). • 3.0: Extreme outliers.
digits	Integer. Number of decimal places for the R console output. Default is 2. If NULL, no rounding is applied. (Note: This does not affect the raw numbers exported to Excel).
export_to_excel	Logical. If TRUE, exports results to an 'Excel' file. Default FALSE.
close_generated_files	Logical. If TRUE, forces Excel to close before saving (Windows only). Default FALSE.

open_generated_files
: Logical. Whether to open the generated output files after creation. Defaults to TRUE in an interactive R session and FALSE otherwise (e.g. in scripts or automated pipelines). Set to TRUE or FALSE to override this behaviour explicitly.

save_as
: Character string. Custom path or filename for the Excel export.
  - If full path: Saves to that location.
  - If filename only: Saves to tempdir() (unless save_in_wdir = TRUE).
  - If directory: Saves as "dataname_summary.xlsx" in that directory.

save_in_wdir
: Logical. If TRUE, saves to the current working directory. Default FALSE.

check_input
: Logical. If TRUE, performs validation checks on inputs. Default TRUE.

digits_excel
: Integer. Number of decimal places for the Excel file cells. Default NULL (no rounding). Defining digits_excel, sets export_to_excel = TRUE when excel output is not intended use: digits instead.

allow_integer_decimal_mix
: Logical. If TRUE, intergers in columns with a mix of integers and non-integers are displayed without decimals. Default FALSE, meaning if there are one or more numbers with decimals the whole column contains the number of decimals set by digits.

Details

The Outlier Logic (Tukey's Method): An observation is flagged as an outlier if it falls outside the calculated fences:

- **Lower Fence:** $Q1 - (coef \times IQR)$
- **Upper Fence:** $Q3 + (coef \times IQR)$

Where $Q1$ is the 25th percentile, $Q3$ is the 75th percentile, and $IQR = Q3 - Q1$.

Output Structure: The function returns a subset of the original data. It automatically adds a row_id columns, which corresponds to the row number in the original dataframe. This ensures you can strictly map the outliers back to the source data.

Value

A data.frame containing the identified outlier rows. Returns NULL (with a message) if no outliers are found.

See Also

f_remove_outliers to remove the rows identified by this function.

Examples

```
# --- Setup: Create Dummy Data ---
set.seed(42)
df <- data.frame(
  Team       = rep(c("A", "B"), each = 20),
  Department = rep(c("Sales", "IT"), each = 10, times = 2),
```

```
  Salary     = rnorm(40, mean = 50000, sd = 2000),
  Age        = rnorm(40, mean = 35, sd = 3),
  EmployeeID = paste0("E", sprintf("%03d", 1:40))
)

# Inject outliers
df[2,  "Salary"] <- 57000   # Mild outlier (between 1.5 and 3.0 fence)
df[1,  "Salary"] <- 100000  # Extreme high
df[35, "Salary"] <- 1000    # Extreme low

# --- Example 1: Basic detection (data.frame notation) ---
# Scan the entire dataset for Salary outliers (no grouping)
out <- f_outliers(df, columns = "Salary")
print(out)

# --- Example 2: Basic detection (formula notation) ---
# Equivalent to Example 1 using the formula interface
# LHS = column(s) to scan, RHS = grouping variable(s)
out <- f_outliers(Salary ~ 1, data = df)
print(out)

# --- Example 3: Grouped detection (both notations) ---
# Outliers are now evaluated *within* each Team separately,
# making detection sensitive to group-level distributions

# data.frame notation:
out <- f_outliers(df, columns = "Salary", group_vars = "Team")

# Formula notation (identical result):
out <- f_outliers(Salary ~ Team, data = df)
print(out)

# --- Example 4: Multi-column + multi-group (formula notation) ---
# Scan both Salary and Age for outliers, grouped by Team and Department.
# Returns a named list: one data.frame per column scanned.
out <- f_outliers(Salary + Age ~ Team + Department, data = df)
print(out)            # prints both result tables
out$Salary            # access Salary outliers directly
out$Age               # access Age outliers directly

# --- Example 5: Strict detection with a custom ID column ---
# coef = 3.0 flags only extreme outliers (the "far out" Tukey fence).
# id_var places EmployeeID first in the output for easy identification.

# data.frame notation:
out <- f_outliers(df,
                  columns   = "Salary",
                  group_vars = "Team",
                  id_var    = "EmployeeID",
                  coef      = 3.0)

# Formula notation (identical result):
out <- f_outliers(Salary ~ Team, data = df,
```

```
                  id_var = "EmployeeID",
                  coef   = 3.0)
print(out)

# --- Example 6: Sensitivity comparison ---
# Compare how coef = 1.5 (standard) vs coef = 3.0 (extreme-only)
# affects the number of flagged rows.

out_standard <- f_outliers(Salary ~ Team, data = df, coef = 1.5)
out_extreme  <- f_outliers(Salary ~ Team, data = df, coef = 3.0)

nrow(out_standard$output_df)  # 3 -- catches mild + extreme outliers
nrow(out_extreme$output_df)   # 2 -- catches extreme outliers only

# --- Example 7: Vector input ---
# Pass a column directly as a vector -- no data.frame needed.
# The column name is captured automatically from the call.

out <- f_outliers(df$Salary)
print(out)

# Works with coef and other parameters too
out <- f_outliers(df$Salary, coef = 3.0)
print(out)

# Inline vectors fall back to the column name "value"
out <- f_outliers(c(1, 2, 3, 4, 5, 100))
print(out)
```

`f_pander` *Fancy Pander Table Output*

Description

Is a wrapper around the pander function from the 'pander' package, designed to produce a fancy table output with specific formatting options.

Usage

```
f_pander(
  table,
  col_width = 10,
  table_width = NULL,
  limit_columns = NULL,
  style = "multiline",
  console = TRUE,
  ...
)
```

Arguments

table	A data frame, matrix, or other table-like structure to be rendered.
col_width	Integer. Specifies the maximum number of characters allowed in table header columns before a line break is inserted. Defaults to 10. Note that latex will not break the table header if the col with is longer due to long names below the header.
table_width	Integer or NULL. Defines the number of characters after which the table is split into separate sections. Defaults to NULL, meaning no break is applied.
limit_columns	Integer or NULL. Defines the number of columns shown in a table.
style	Character. Pander table style. Defaults to "multiline".
console	Logical. Whether to process headers for console output. Defaults to TRUE.
...	Additional arguments passed to the pander function.

Details

This function sets several pander options to ensure that the table output is formatted in a visually appealing manner. The options set include:

- table.alignment.default: Aligns all columns to the left.
- table.alignment.rownames: Aligns row names to the left.
- keep.trailing.zeros: Keeps trailing zeros in numeric values.
- knitr.auto.asis: Ensures output is not automatically treated as 'asis'.
- table.caption.prefix: Removes the default "Table" prefix in captions.
- keep.line.breaks: Preserves line breaks in cell content.
- table.split.table: Controls table splitting (set to Inf if table_width is NULL or FALSE).
- table.split.cells: Inserts line breaks in headers every col_width characters.

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

None. The function is called for its side effects of setting 'pander' options and creates a pander formatted table in R Markdown.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Example usage of f_pander
df <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 35),
  Score = c(88.5, 92.3, 85.0)
)

# Render the data frame as a fancy table
f_pander(df)
```

---

f_qqnorm *Normal Q-Q Plot with Confidence Bands*

Description

This function creates a normal Q-Q plot for a given numeric vector and adds confidence bands to visualize the variability of the quantiles.

Usage

```
f_qqnorm(
  x,
  main = NULL,
  ylab = NULL,
  conf_level = 0.95,
  col = NULL,
  pch = NULL,
  cex = NULL,
  save_png = FALSE,
  open_png = TRUE,
  save_as = NULL,
  save_in_wdir = FALSE,
  width = 8,
  height = 7,
  units = "in",
  res = 300,
  ...
)
```

Arguments

x	A numeric vector of data values.
main	A character string specifying the title of the histogram. Default is "Histogram with Normal Curve".
ylab	A character string specifying the y-axsis label. Default name is "Quantiles of: data_name".
conf_level	Numeric, between 0 and 1. Confidence level for the confidence bands. Default is 0.95 (95% confidence).
col	Numeric, optional parameter for color of point with default 'black'.
pch	Numeric, optional parameter shape of points default pch = 19.
cex	Numeric, optional parameter for graph cex with default cex = 0.6.
save_png	A logical value default FALSE, if TRUE a png file is saved under the name of the data of under the specified file name.
open_png	Logical. If TRUE, opens generated png files.
save_as	Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in tempdir(). If only a directory is specified (providing an existing directory with trailing slash), the file is named "data_name_QQplot.png" in that directory. Defaults to file.path(tempdir(), "data_name_histogram.png").
save_in_wdir	Logical. If TRUE, saves the file in the working directory. Default is FALSE, this avoid unintended changes to the global environment. If save_as location is specified save_in_wdir is overwritten by save_as.
width	Numeric, png figure width default 8 inch.
height	Numeric, png figure height default 7 inch.
units	Numeric, png figure units default inch.
res	Numeric, png figure resolution default 300 dpi.
...	Additional graphical parameters to be passed to the qqnorm function.

Details

The function calculates theoretical quantiles for a normal distribution and compares them with the sample quantiles of the input data.

It also computes confidence intervals for the order statistics using the Blom approximation and displays these intervals as shaded bands on the plot.

The reference line is fitted based on the first and third quartiles of both the sample data and theoretical quantiles.

To increase resolution you can use png(...,res = 600) or the 'RStudio' chunck setting, e.g. dpi = 600.

Value

A Q-Q plot is created and the function returns this as a recordedplot.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Generate random normal data
set.seed(123)
data <- rnorm(100)

# Create a Q-Q plot with confidence bands
f_qqnorm(data)

# Customize the plot with additional graphical parameters
f_qqnorm(data, conf_level = 0.99, pch = 16, col = "blue")
```

---

f_remove_outliers *Remove Outliers from Data*

Description

'f_remove_outliers()' removes specific rows from a dataframe based on a list of identifiers. It is designed to work seamlessly with the output of `f_outliers`, but can also accept a custom vector of IDs.

Usage

```
f_remove_outliers(data, outliers, by = "row_id", verbose = TRUE)
```

Arguments

`data`	A data.frame, tibble, or data.table containing the original data.
`outliers`	Either: • A dataframe returned by `f_outliers`. • A vector of IDs/row numbers to remove.
`by`	A character string specifying the column to match on. Default is `"row_id"`. If the source `data` does not have a `row_id` column, the function effectively uses the row numbers (1, 2, 3...) to ensure safe deletion.
`verbose`	Logical. If `TRUE` (default), prints a summary of how many rows were removed.

Details

Safe Deletion Logic: This function performs a "anti-join" style filtering. It keeps rows where the identifier in by is **not** found in the `outliers` list.

Handling Row IDs: If you use the default by = `"row_id"` and your original `data` does not have a column named `"row_id"`, the function assumes you are referring to the intrinsic row numbers of the data.frame, tibble, or data.table. It will temporarily generate IDs to perform the deletion and then return the clean data with the original structure (without adding a permanent `row_id` column to the result).

Value

An object of the same class as the input `data` (data.frame, tibble, or data.table) with the specified outlier rows removed.

See Also

`f_outliers` to identify the rows to be removed.

Examples

```
# --- Setup: Create Dummy Data ---
set.seed(42)
df <- data.frame(
  Team       = rep(c("A", "B"), each = 20),
  Department = rep(c("Sales", "IT"), each = 10, times = 2),
  Salary     = c(rnorm(19, 50000, 500), 100000,
                 rnorm(18, 50000, 500), 57000, 1000),
  Age        = c(rnorm(38, 35, 2), 90, 35),
  EmployeeID = paste0("E", sprintf("%03d", 1:40)),
  stringsAsFactors = FALSE
)
# row 20:  extreme high Salary (Team A)
# row 39:  mild Salary outlier at coef = 1.5 only
# row 40:  extreme low  Salary (Team B)
# row 39:  extreme high Age

# --- Example 1: Basic two-step workflow (data.frame notation) ---
# The most common use case: find then remove in two lines.
bad_rows <- f_outliers(df, columns = "Salary")
clean_df <- f_remove_outliers(df, bad_rows)
nrow(df)       # 40
nrow(clean_df) # 40 minus flagged rows

# --- Example 2: Basic two-step workflow (formula notation) ---
# Identical result to Example 1 using the formula interface.
bad_rows <- f_outliers(Salary ~ 1, data = df)
clean_df <- f_remove_outliers(df, bad_rows)
nrow(clean_df)

# --- Example 3: Grouped detection then removal (both notations) ---
# Outliers are identified *within* each Team separately before removal.
```

```
# data.frame notation:
bad_rows <- f_outliers(df, columns = "Salary", group_vars = "Team")
clean_df <- f_remove_outliers(df, bad_rows)

# Formula notation (identical result):
bad_rows <- f_outliers(Salary ~ Team, data = df)
clean_df <- f_remove_outliers(df, bad_rows)
nrow(clean_df)

# --- Example 4: Selective removal -- only act on a subset of outliers ---
# Find all flagged rows, but only remove the extreme high salaries.
# Step 1: Identify all Salary outliers grouped by Team
bad_rows    <- f_outliers(Salary ~ Team, data = df)
all_flagged <- bad_rows$output_df

# Step 2: Filter to keep only the rows where Salary > 90000
really_bad  <- all_flagged[all_flagged$Salary > 90000, ]

# Step 3: Remove only those rows -- low outlier (row 40) is preserved
clean_df <- f_remove_outliers(df, really_bad)
range(clean_df$Salary)  # low outlier still present, high one is gone

# --- Example 5: Multi-column outlier removal ---
# f_outliers scans both Salary and Age; f_remove_outliers removes
# every row flagged by either column in one call.

# Formula notation:
bad_rows <- f_outliers(Salary + Age ~ Team, data = df)
clean_df <- f_remove_outliers(df, bad_rows)

# data.frame notation (identical result):
bad_rows <- f_outliers(df, columns = c("Salary", "Age"), group_vars = "Team")
clean_df <- f_remove_outliers(df, bad_rows)
nrow(clean_df)  # rows flagged by Salary OR Age are removed

# --- Example 6: Strict detection + custom ID column ---
# coef = 3.0 flags only extreme outliers. EmployeeID is used
# as the matching key instead of the default row_id.

# Formula notation:
bad_rows <- f_outliers(Salary ~ Team, data = df,
                       id_var = "EmployeeID", coef = 3.0)

# data.frame notation (identical result):
bad_rows <- f_outliers(df, columns = "Salary", group_vars = "Team",
                       id_var = "EmployeeID", coef = 3.0)

# Remove by EmployeeID rather than row position
clean_df <- f_remove_outliers(df, bad_rows$output_df, by = "EmployeeID")

# Confirm the flagged employees are no longer in the clean data
bad_ids  <- bad_rows$output_df$EmployeeID
```

```
any(clean_df$EmployeeID %in% bad_ids)  # FALSE
```

`f_rename_columns` *Rename Specific Columns in a Data Frame*

Description

Renames specific columns in a data frame based on a named vector (name_map). It ensures that only the specified columns are renamed, while others remain unchanged.

Usage

```
f_rename_columns(df, name_map)
```

Arguments

`df`	A data frame whose columns are to be renamed.
`name_map`	A named vector where the names correspond to the current column names in `df`, and the values are the new names to assign. All names in `name_map` must exist in the column names of `df`. Yet, not all names in the data.frame have to be in `name_map`. This allows for selective renaming of just one or two columns.

Details

This function is particularly useful when you want to rename only a subset of columns in a data frame. It performs input validation to ensure that:

- `name_map` is a named vector.
- All names in `name_map` exist as column names in `df`.

If these conditions are not met, the function will throw an error with an appropriate message.

Value

A data frame with updated column names. Columns not specified in `name_map` remain unchanged.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Create a sample data frame.
df <- data.frame(a = 1:3, b = 4:6, c = 7:9)

# Define a named vector for renaming specific columns.
name_map <- c(a = "alpha", c = "gamma")

# Rename columns.
df <- f_rename_columns(df, name_map)

# View updated data frame.
print(df)
```

---

f_rename_vector *Rename Elements of a Vector Based on a Mapping*

Description

Renames elements of a vector based on a named mapping vector. Elements that match the names in the mapping vector are replaced with their corresponding values, while elements not found in the mapping remain unchanged.

Usage

```
f_rename_vector(vector, name_map)
```

Arguments

vector	A character vector containing the elements to be renamed.
name_map	A named vector where the names correspond to the elements in vector that should be renamed, and the values are the new names to assign.

Details

This function iterates through each element of vector and checks if it exists in the names of name_map. If a match is found, the element is replaced with the corresponding value from name_map. If no match is found, the original element is retained. The result is returned as an unnamed character vector.

Value

A character vector with updated element names. Elements not found in name_map remain unchanged.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# Define a vector and a name map.
vector   <- c("Species", "Weight", "L")
name_map <- c(Species = "New_species_name", L = "Length_cm")

# Rename elements of the vector.
updated_vector <- f_rename_vector(vector, name_map)

# View updated vector
print(updated_vector)
```

`f_scan` *Perform a visual check on your data*

Description

Creates a 3-panel diagnostic dashboard to check data distribution and assumptions. It can also output a data summary table and identify outliers.

Usage

```
f_scan(x, ...)

## S3 method for class 'formula'
f_scan(formula, data = NULL, ...)

## S3 method for class 'numeric'
f_scan(x, ...)

## S3 method for class 'integer'
f_scan(x, ...)

## S3 method for class 'data.frame'
f_scan(
  x,
  columns = NULL,
  group_vars = NULL,
  summary = TRUE,
  outliers = TRUE,
  coef = 1.5,
  limit_columns = 7,
  fancy_names = NULL,
  advice = FALSE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  output_type = "default",
```

```
  save_as = NULL,
  save_in_wdir = FALSE,
  digits = NULL,
  ...
)
```

Arguments

x	A data.frame or formula (dispatches to the right method).
...	Further arguments forwarded to f_scan.data.frame.
formula	A formula specifying the columns (right hand side) to be summarized by maximal 3 groups (left hand side). More columns or groups can be added using - or + (e.g., col1 + col2 ~ group1 + group2) to do a sequential summary for each column parameter.
data	A 'data.frame', 'data.table', or 'tibble'.
columns	The numerical column(s) to summarize if no formula is used. Can be entered as a single character string (e.g., "weight") or as a character vector c("weight", "length"). When omitted, defaults to all numeric columns in data (excluding any columns named in group_vars).
group_vars	Character vector of up to 3 grouping variables (e.g., c("species", "fertilizer")).
summary	Logical. Show a summary table of the data. Default is TRUE.
outliers	Logical. If TRUE, scans for outliers using Tukey's fences and if they exist, adds them to the result object. Default TRUE.
coef	Numeric. The multiplier for the Interquartile Range (IQR) used for outlier detection. Default 1.5.
limit_columns	Integer or NULL. Defines the number of columns shown in the outlier table. Default = 7. NULL = all columns are shown.
fancy_names	Named character vector or NULL. Optional mapping of column names to more readable names for display in plots and legends.
advice	Logical. If TRUE, runs f_stat_wizard() on each response column and appends the recommendation to the result. The advice is accessible via result[["column_name"]]$advice and is printed automatically. Default FALSE.
close_generated_files	Logical. Closes open Excel or Word (NOT pdf) files before writing, depending on the output format. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default FALSE. **WARNING:** Always save your work before using this option!!
open_generated_files	Logical. Whether to open the generated output files after creation. Defaults to TRUE in an interactive R session and FALSE otherwise (e.g. in scripts or automated pipelines). Set to TRUE or FALSE to override this behaviour explicitly.
output_type	Character string specifying the output format. Default is "default". • "default": Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use print(result) or plot(result) to display the returned object.

- "console": Forces immediate printing to the console regardless of object assignment.
- "pdf", "word", "excel": Saves results to a file of the corresponding format. See save_as, save_in_wdir, and open_generated_files for file path and opening behavior.
- "rmd": Stores the raw markdown string inside the returned object for use in R Markdown documents.

save_as — Character string specifying the output file path (without extension). If a full path is provided, output is saved to that location. If only a filename is given, the file is saved in tempdir(). If only a directory is specified (providing an existing directory with trailing slash), the file is named "dataname_fscan" in that directory. If an extension is provided the output format specified with option "output_type" will be overruled. Defaults to file.path(tempdir(), "dataname_fscan.pdf").

save_in_wdir — Logical. If TRUE, saves the file in the working directory. Default is FALSE, this avoid unintended changes to the global environment. If save_as location is specified save_in_wdir is overwritten by save_as.

digits — Integer. Decimal places for printed tables in 'pdf' and 'Word' output files. Default 3.

Details

f_scan automatically adapts the visualization based on the number of grouping variables provided:

- **0 groups:** Univariate analysis (Single density/boxplot).
- **1 group :** Main grouping variable (X-axis and Color).
- **2 groups:** Adds Facet Wrapping.
- **3 groups:** Adds Facet Grid (Row vs Column).

This function requires [Pandoc](https://github.com/jgm/pandoc/releases/tag) (version 1.12.3 or higher), a universal document converter.

- **Windows:** Install Pandoc and ensure the installation folder. (e.g., "C:/Users/your_username/AppData/Local/Pandoc") is added to your system PATH.
- **macOS:** If using Homebrew, Pandoc is typically installed in "/usr/local/bin". Alternatively, download the .pkg installer and verify that the binary's location is in your PATH.
- **Linux:** Install Pandoc through your distribution's package manager (commonly installed in "/usr/bin" or "/usr/local/bin") or manually, and ensure the directory containing Pandoc is in your PATH.
- If Pandoc is not found, this function may not work as intended.

Value

A list of class f_scan containing plots, the summary table, and the outlier table. Using the option "output_type", it can also generate output in the form of: R Markdown code, 'Word', 'pdf', or 'Excel' files. Includes print, summary and plot methods for 'f_scan' objects.

Examples

```
# 1. Non-formula | No groups | Default output (default)
result <- f_scan(iris, columns = "Sepal.Length")
print(result)

# 2. Non-formula | 1 group | Console output
result <- f_scan(
  mtcars,
  columns   = "mpg",
  group_vars = "cyl",
  output_type = "console"
)


# 3. Non-formula | 2 groups | Multiple columns | Excel output
 result <- f_scan(
   mtcars,
   columns    = c("mpg", "hp"),
   group_vars = c("cyl", "am"),
   outliers   = TRUE,
   coef       = 1.5,
   output_type = "excel",
   save_as    = "mtcars_scan"
 )

# 4. Formula | 1 group | Strict outlier detection | Word output
result <- f_scan(
  Sepal.Width ~ Species,
  data        = iris,
  outliers    = TRUE,
  coef        = 3.0,
  output_type = "word",
  save_as     = "iris_scan"
 )

# 5. Formula | 2 groups | Multiple columns | Fancy names
result <- f_scan(
 mpg + hp + wt ~ vs + am,
 data        = mtcars,
 fancy_names = c(mpg = "Fuel Efficiency", hp = "Horsepower",
                 wt  = "Weight",          vs = "Engine Type",
                 am  = "Transmission"),
 summary     = TRUE
)
print(result)


#Create a small reproducible dataset with 3 grouping variables
set.seed(42)
plant_data <- data.frame(
  weight    = c(rnorm(60, 10, 2), rnorm(60, 14, 2)),
  species   = rep(c("A", "B"), each = 60),
```

```
  treatment = rep(rep(c("control", "treated"), each = 30), 2),
  batch     = factor(rep(c("1", "2", "3"), 40))
 )

# 6. Formula | 3 groups | Facet Grid
result <- f_scan(
  weight ~ species + treatment + batch,
  data         = plant_data,
  coef         = 2.0,
  digits       = 2,
  output_type  = "word"
)
print(result)

# 7. With statistical advice
result <- f_scan(
  Sepal.Length ~ Species,
  data    = iris,
  advice  = TRUE
)
#' print(result)
result[["Sepal.Length"]]$advice$y_type


# 8. Vector input | Single numeric vector (no formula, no data.frame)
# When you only have loose vectors in your workspace, pass one
# directly to f_scan(). The vector's name is used as the column label
# in the dashboard and outlier table.
disp1 <- mtcars$disp
result <- f_scan(disp1)
print(result)

# 9. Formula on vectors | Multiple responses | One grouping vector
# f_scan() also accepts a formula built from bare vectors, i.e.
# no `data =` argument is needed. Multiple
# response variables are combined with `+` on the
# left hand side of the formula, exactly as
# in the data.frame form.
disp1 <- mtcars$disp
hp1   <- mtcars$hp
cyl1  <- factor(mtcars$cyl)
result <- f_scan(disp1 + hp1 ~ cyl1)
print(result)

# 10. Positional vector form: equivalent to f_scan(disp1 ~ cyl1).
# The first vector is the response, the rest are grouping variables.
disp1 <- mtcars$disp
cyl1  <- factor(mtcars$cyl)
f_scan(disp1, cyl1)
```

f_setwd *Set Working Directory Based on Current File or Specified Path*

Description

A wrapper around `setwd()` that sets the working directory to the location of the currently open file in 'RStudio' if no path is provided. If a path is specified, it sets the working directory to that path instead.

Usage

```
f_setwd(path = NULL)
```

Arguments

`path`	A character string specifying the desired working directory. If `NULL` (default), the function sets the working directory to the location of the currently open and saved file in 'RStudio'.

Details

If `path` is not provided (`NULL`), this function uses the `this.path` package to determine the location of the currently open file and sets that as the working directory. The file must be saved for this to work properly.

If a valid `path` is provided, it directly sets the working directory to that path.

Value

None. The function is called for its side effects of changing the working directory.

Note

- The function checks whether the currently open file is saved before setting its location as the working directory.
- If the function is called from an unsaved script or directly from the console, an error will be thrown.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# NOTE: The use of "if(interactive())" prevents this example from running
# during automated CRAN checks. This is necessary because the example
# requires to be run from an R script. You don't need to use
# "if(interactive())" in your own scripts.
if(interactive()) {
# Store the current working directory, so we can reset it after the example.
current_wd <-  getwd()
print(current_wd)

# Run this commando from a saved R script file, or R Notebook to set the working
# directory to scripts' file location
f_setwd()

# Restore your current working directory
f_setwd(current_wd)
}
```

---

f_stat_wizard *Statistical Test Wizard*

Description

Analyzes your data structure based on a formula and recommends the appropriate statistical test. Checks variable types, normality of residuals, homogeneity of variance, and checks if f_boxcox transformation can fix non-normality. Recommends rfriend functions as primary code, with base R alternatives shown as fallback.

Supports standard formulas including `y ~ .`, `y ~ as.factor(x)`, and interaction terms. Formulas with random effects (e.g. `(1|ID)`) are detected and handled separately. Multivariate responses (e.g. `cbind(y1, y2) ~ x`) and transformed responses (e.g. `log(y) ~ x`) are not supported.

Usage

```
f_stat_wizard(x, ...)

## S3 method for class 'formula'
f_stat_wizard(
  formula,
  data,
  id_col = NULL,
  run = FALSE,
  plots = FALSE,
  output_type = "word",
  interactive = FALSE,
  data_name = NULL,
  ...
```

```
)

## S3 method for class 'data.frame'
f_stat_wizard(
  x,
  formula,
  id_col = NULL,
  run = FALSE,
  plots = FALSE,
  output_type = "word",
  interactive = FALSE,
  data_name = NULL,
  ...
)
```

Arguments

x	A formula (e.g., y ~ x) or a data frame. When a formula is provided, data must also be supplied. When a data frame is provided, formula must be supplied as the second argument.
...	Additional arguments (currently unused).
formula	A formula specifying the relationship (used with the data.frame method).
data	A data frame containing the variables referenced in the formula.
id_col	Character string. Name of the column identifying subjects/blocks for paired or repeated-measures designs. When supplied, the wizard (a) verifies the pairing structure (each subject should appear in every group exactly once), (b) treats the design as paired/repeated measures, and (c) embeds the real column name into the generated code. Omit for independent-samples designs. Default NULL.
run	Logical. If TRUE, the wizard attempts to execute the recommended rfriend function and stores the result in $run_result. Only works for unambiguous single-function recommendations (not multi-step or external packages). Default FALSE.
plots	Logical. If TRUE, generates diagnostic plots using f_hist() (histogram of the response) and f_qqnorm() (QQ-plot of model residuals). Plots are stored in the result as $histogram and $qqplot (recordedplot objects) and displayed by the print method. Default FALSE.
output_type	Character string specifying the output format of the recommended rfriend function (when run=TRUE) and the displayed code strings. Passed through to f_aov(), f_t_test(), f_glm(), etc. Valid values match those of the underlying function ("word", "pdf", "excel", "console", "default", "rmd"). Default "word".
interactive	Logical. If TRUE, asks the user questions about study design. Default FALSE.
data_name	Character string to name the data base used. Default NULL, which automatically derives the data name from the data frame used as input.

Value

An object of class "f_stat_wizard": a list containing:

formula The formula used.

formula_text Character string of the formula.

data_name Name of the data object as passed by the user.

n Effective sample size (after NA removal).

n_dropped Number of rows removed due to missing values.

paired Logical. Whether a paired/repeated-measures design was detected (via `id_col`).

id_col Character. Name of the subject/block column supplied, or `NULL`.

y_var Name of the response variable.

y_type Detected type of the response: `"binary"`, `"count"`, `"multinomial"`, `"ratio_normal"`, `"ratio_non_normal"`, `"ratio_unknown"`, or `"unsupported"`.

x_vars Character vector of explanatory variable names.

x_types Character vector of detected types (`"nominal"`, `"ordinal"`, `"ratio"`).

n_groups Number of groups (for single categorical X), or `NULL`.

group_sizes Table of per-group sample sizes, or `NULL`.

is_ancova Logical. `TRUE` if the model mixes nominal and ratio predictors.

has_interaction Logical. `TRUE` if interaction terms were detected.

normality A list with `p_value` (Shapiro-Wilk) and `is_normal` (logical or `NA`).

variance A list with `test_used` (`"Levene"` or `"Bartlett"`), `p_value`, and `is_equal` (logical).

boxcox A list with `attempted` (logical), `can_fix` (logical), and `p_value_after` (numeric or `NA`).

overdispersion A list with `is_overdispersed` (logical, from DHARMa dispersion test) and `p_value`. Only meaningful for count data.

recommended_call A language object representing the rfriend function call, or `NULL` if no single function could be determined.

run_result The result of executing the recommended test (when `run=TRUE`), or `NULL`.

histogram A `recordedplot` from `f_hist()` (when `plots=TRUE`), or `NULL`.

qqplot A `recordedplot` from `f_qqnorm()` of model residuals (when `plots=TRUE` and Y is continuous), or `NULL`.

report Character vector of the human-readable report lines (used by `print.f_stat_wizard`).

Examples

```
# Formula interface (recommended)
f_stat_wizard(Sepal.Length ~ Species, data = iris)

# Data-first interface (backward compatible)
f_stat_wizard(iris, Sepal.Length ~ Species)

# Paired design -- supply the id_col that identifies matched subjects
f_stat_wizard(extra ~ group, data = sleep, id_col = "ID")

# With diagnostic plots
f_stat_wizard(Sepal.Length ~ Species, data = iris, plots = TRUE)
```

```
# Run the recommended test directly
result <- f_stat_wizard(Sepal.Length ~ Species, data = iris, run = TRUE)
result$run_result

# Inspect metadata
result <- f_stat_wizard(Sepal.Length ~ Species, data = iris)
result$y_type
result$normality
result$group_sizes
```

`f_summary` *Summarize a Data Frame with Grouping Variables*

Description

Computes summary statistics (n, mean, sd, etc.) for a specified numerical columns in a data frame. The dataset can be analyzed as a whole or split by one or more grouping variables.

The function returns a formatted data frame and includes options to export the results directly to an 'Excel' file.

Usage

```
f_summary(x, ...)

## S3 method for class 'formula'
f_summary(x, data, ...)

## S3 method for class 'data.frame'
f_summary(
  x,
  columns = NULL,
  group_vars = NULL,
  show_name = TRUE,
  show_n = TRUE,
  show_mean = TRUE,
  show_sd = TRUE,
  show_se = TRUE,
  show_ci = FALSE,
  conf_level = 0.95,
  show_min = TRUE,
  show_max = TRUE,
  show_median = TRUE,
  show_Q1 = TRUE,
  show_Q3 = TRUE,
  show_skew = FALSE,
  show_kurtosis = FALSE,
  digits = NULL,
```

```
  export_to_excel = FALSE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  save_as = NULL,
  save_in_wdir = FALSE,
  check_input = TRUE,
  digits_excel = NULL,
  allow_integer_decimal_mix = FALSE,
  ...
)
```

Arguments

x	A data.frame or formula (dispatches to the right method).
...	Further arguments forwarded to f_summary.data.frame.
data	A 'data.frame', 'data.table', or 'tibble'.
columns	The numerical column(s) to summarize if no formula is used. Can be entered as a single character string (e.g., "weight") or as a character vector c("weight", "length"). When omitted, defaults to all numeric columns in data (excluding any columns named in group_vars).
group_vars	A character vector specifying the grouping variables in data (e.g., c("species", "fertilizer")) if no formula is used. If NULL, the entire dataset is summarized.
show_name	Logical. Include variable name. Default TRUE.
show_n	Logical. Include count (n). Default TRUE.
show_mean	Logical. Include mean. Default TRUE.
show_sd	Logical. Include standard deviation. Default TRUE.
show_se	Logical. Include standard error. Default TRUE.
show_ci	Logical. Include the lower and upper bounds of a confidence interval for the mean (columns CI_lower and CI_upper). Default FALSE. This interval is most meaningful when the data are approximately normal or n is large; see Details.
conf_level	Numeric. Confidence level for the interval requested by show_ci, given as a proportion between 0 and 1. Default 0.95 (a 95% confidence interval).
show_min	Logical. Include minimum value. Default TRUE.
show_max	Logical. Include maximum value. Default TRUE.
show_median	Logical. Include median. Default TRUE.
show_Q1	Logical. Include first quartile (25th percentile). Default TRUE.
show_Q3	Logical. Include third quartile (75th percentile). Default TRUE.
show_skew	Logical. Include Skewness (measure of asymmetry). Default FALSE.
show_kurtosis	Logical. Include Excess Kurtosis (measure of "tailedness"). Default FALSE.
digits	Integer. Number of decimal places for the R console output. Default is 2. If NULL, no rounding is applied. (Note: This does not affect the raw numbers exported to Excel).

`export_to_excel`
: Logical. If `TRUE`, exports results to an 'Excel' file. Default `FALSE`.

`close_generated_files`
: Logical. If `TRUE`, forces Excel to close before saving (Windows only). Default `FALSE`.

`open_generated_files`
: Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.

`save_as`
: Character string. Custom path or filename for the Excel export.
  - If full path: Saves to that location.
  - If filename only: Saves to `tempdir()` (unless `save_in_wdir = TRUE`).
  - If directory: Saves as "dataname_summary.xlsx" in that directory.

`save_in_wdir`
: Logical. If `TRUE`, saves to the current working directory. Default `FALSE`.

`check_input`
: Logical. If `TRUE`, performs validation checks on inputs. Default `TRUE`.

`digits_excel`
: Integer. Number of decimal places for the Excel file cells. Default `NULL` (no rounding). Defining `digits_excel`, sets `export_to_excel = TRUE` when excel output is not intended use: `digits` instead.

`allow_integer_decimal_mix`
: Logical. If `TRUE`, intergers in columns with a mix of integers and non-integers are displayed without decimals. Default `FALSE`, meaning if there are one or more numbers with decimals the whole column contains the number of decimals set by `digits`.

`formula`
: A formula specifying the columns (right hand side) to be summarized by groups (left hand side). More columns or groups can be added using `-` or `+` (e.g., `col1 + col2 ~ group1 + group2`) to do a sequential summary for each column parameter.

Details

The function computes the following statistics:

- `n`: number of observations
- `mean`: arithmetic mean
- `sd`: standard deviation
- `se`: standard error ($sd/\sqrt{n}$)
- `CI_lower`, `CI_upper`: lower and upper bounds of the confidence interval for the mean (if requested)
- `min`: minimum value
- `max`: maximum value
- `median`: median value
- `Q1`: 25th percentile
- `Q3`: 75th percentile
- `skew`: Sample skewness (if requested).

- kurt: Sample excess kurtosis (if requested).

skew stands for Skewness which is a measure of asymmetry of a distribution around its mean. Where skew values near **0** indicate approximate symmetry, while large positive or negative values indicate noticeable asymmetry.

- > 0: Right-skewed (long or heavier tail to the right).
- < 0: Left-skewed (long or heavier tail to the left).

kurt stands for Excess Kurtosis: Tells you about the "tails" and the peak.

- 0: Same tail heaviness as the normal distribution (mesokurtic).
- > 0: Heavier tails than normal (Leptokurtic) – indicates frequent outliers.
- < 0: Lighter tails than normal (Platykurtic) – indicates fewer (or less extreme) outliers than a normal distribution.

The confidence interval reported when show_ci = TRUE is a parametric interval for the mean based on the t-distribution, computed as $mean \pm t_{(1-(1-conf\_level)/2,\, n-1)} \times se$, where n is the number of non-missing observations. This matches the interval reported by t.test. It assumes the data are approximately normally distributed (or that n is large enough for the central limit theorem to apply); for strongly skewed data, indicated for example by a large skew or kurt, the interval may be unreliable. Groups with fewer than two non-missing observations yield NA bounds.

If group_vars are provided, the statistics are calculated for each group combination. When export_to_excel = TRUE, the file is automatically generated.

Value

A list of class f_summary containing the results data frame.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# --- Example 1: Basic Usage (data.frame notation) ---
# Summarize "hp" grouped by "cyl"; columns and group_vars can be positional
summary_mtcars <- f_summary(mtcars, columns = "hp", group_vars = "cyl")
summary_mtcars <- f_summary(mtcars, "hp", "cyl")  # shorthand equivalent
print(summary_mtcars)

# --- Example 2: Multiple Columns & Groups with Custom Toggles ---
# Summarize "hp" and "disp", grouped by "cyl" and "gear", hide Q1/Q3
summary_custom <- f_summary(mtcars,
                            columns    = c("hp", "disp"),
                            group_vars = c("cyl", "gear"),
                            show_Q1    = FALSE,
                            show_Q3    = FALSE)
print(summary_custom)

# --- Example 3: Formula Notation ---
```

```
# Identical result to Example 2 using formula interface
# and export output to excel
summary_formula <- f_summary(hp + disp ~ cyl + gear,
                             data    = mtcars,
                             show_Q1 = FALSE,
                             show_Q3 = FALSE,
                             export_to_excel = TRUE)
print(summary_formula)

# --- Example 4: Distributional Stats & Digits ---
# Add skewness and kurtosis, control rounding
summary_dist <- f_summary(Sepal.Length + Petal.Length ~ Species,
                          data          = iris,
                          show_skew     = TRUE,
                          show_kurtosis = TRUE,
                          digits        = 3)
print(summary_dist)

# --- Example 5: Custom Print Formatting ---
summary_iris <- f_summary(iris, "Sepal.Length", group_vars = "Species")
print(summary_iris, col_width = 10, table_width = 70)


# --- Example 6: Confidence Interval for the Mean ---
# Add a 95% CI for the mean of Sepal.Length within each Species.
summary_ci <- f_summary(Sepal.Length ~ Species,
                        data    = iris,
                        show_ci = TRUE)
print(summary_ci)

# Use a 90% interval instead
summary_ci90 <- f_summary(Sepal.Length ~ Species,
                          data       = iris,
                          show_ci    = TRUE,
                          conf_level = 0.90)
print(summary_ci90)
```

`f_theme` *Apply a black or white 'RStudio' Theme and Zoom Level*

Description

This comes in hand when teaching, the function allows users to apply a "black" or "white" 'RStudio' theme and adjust the zoom level in the 'RStudio' IDE. It includes error handling for invalid inputs.

Usage

```
f_theme(color = "black", zlevel = 0)
```

Arguments

color	A character string. The theme color to apply. Must be either "black" (dark theme) or "white" (light theme). Default is "black".
zlevel	A numeric value. The zoom level to apply, ranging from 0 (default size) to 4 (maximum zoom). Default is 0.

Details

The function performs the following actions:

1. Applies the specified 'RStudio' theme:
   - "black": Applies the "Tomorrow Night 80s" dark theme.
   - "white": Applies the "Textmate (default)" light theme.
2. Adjusts the zoom level in 'RStudio':
   - zlevel = 0: Resets to default zoom level.
   - zlevel = 1: Zooms in once.
   - zlevel = 2: Zooms in twice.
   - zlevel = 3: Zooms in three times.
   - zlevel = 4: Zooms in four times.

The function includes error handling to ensure valid inputs:

- color must be a character string and one of "black" or "white".
- zlevel must be a numeric value, an integer, and within the range of 0 to 4. If a non-integer is provided, it will be rounded to the nearest integer with a warning.

Value

None. The function is called for its side effects of changing the 'RStudio' theme or Zoomlevel.

This function does not return a value. It applies changes directly to the 'RStudio' IDE.

Author(s)

Sander H. van Delden <plantmind@proton.me>

Examples

```
# NOTE: This example will change your RStudio theme hence the dont run warning.
## Not run:
# Apply a dark theme with with zoom level 2:
f_theme(color = "black", zlevel = 2)

# Apply a black theme with maximum zoom level:
f_theme(color = "black", zlevel = 4)

# Apply the default light theme default zoom level:
f_theme(color = "white", zlevel = 0)
```

```
## End(Not run)
```

`f_t_test` *Perform multiple t-tests with optional data transformation, inspection and visualization.*

Description

Performs One-sample, Two-sample (Independent), or Paired t-tests on a given dataset with options for (Box-Cox/BestNormalize) transformations, normality tests, and visualization. Several response parameters can be analysed in sequence (formula interface). Additionally, a vector interface similar to `stats::t.test()` is supported.

Usage

```
f_t_test(x, ...)

## S3 method for class 'formula'
f_t_test(
  formula,
  data = NULL,
  paired = FALSE,
  var.equal = NULL,
  conf.level = NULL,
  mu = 0,
  alternative = "two.sided",
  norm_plots = TRUE,
  transformation = TRUE,
  force_transformation = NULL,
  alpha = 0.05,
  intro_text = TRUE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  ...
)

## Default S3 method:
f_t_test(
  x,
  y = NULL,
  paired = FALSE,
  var.equal = NULL,
```

```
  conf.level = NULL,
  mu = 0,
  alternative = "two.sided",
  norm_plots = TRUE,
  transformation = TRUE,
  force_transformation = NULL,
  alpha = 0.05,
  intro_text = TRUE,
  close_generated_files = FALSE,
  open_generated_files = interactive(),
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  ...
)
```

Arguments

x	Numeric vector of data values (one-sample or first group for two-sample), or a formula of the form `response ~ group` or `response ~ 1`.
...	For the formula method: additional arguments forwarded to the row-filtering step. The arguments `subset` and `na.action` are honored: when supplied, they are spliced (still unevaluated) into a `model.frame` call built once before the per-response loop, so the `subset` expression is evaluated in the data's column scope (e.g. `subset = cyl == 6` works). All responses in a multi-response call (`y1 + y2 ~ group`) are then tested on the identical row set. For the default (vector) method, `...` is currently unused.
formula	A formula specifying the model (alternative to using x/y). • **Two-sample (Independent/Paired):** `response ~ group` (where group has exactly 2 levels). • **One-sample:** `response ~ 1` or `response ~ NULL`. More response variables can be added using `+` (e.g., `y1 + y2 ~ group`).
data	A data frame containing the variables when using the formula interface.
paired	Logical. If `TRUE`, performs a paired t-test. **Note:** For the formula interface, data must be sorted so that all observations of group 1 appear before group 2 (AABB order). For the vector interface, `x` and `y` must have the same length.
var.equal	Logical or `NULL`. If `TRUE`, forces Student's t-test (equal variances). If `FALSE` or `NULL` (default), Welch's t-test is used. Bartlett's and Levene's tests are always reported as diagnostics but do **not** affect this choice. See Delacre, Lakens & Leys (2017).
conf.level	Numeric. Confidence level. Default is `1 - alpha`. If `conf.level` is specified, `alpha` is set to `1 - conf.level`.
mu	Numeric. The true value to test against: the mean (one-sample), the mean of differences (paired), or the difference in means (two-sample). Default is 0. For transformed analyses, `mu` is forward-transformed for one-sample and paired tests. For two-sample tests with `mu != 0` a warning is issued.

`alternative` Character string. `"two.sided"` (default), `"greater"`, or `"less"`.

`norm_plots` Logical. If `TRUE`, diagnostic plots are included in the output. Default is `TRUE`.

`transformation` Logical or character string. If `TRUE` or `"boxcox"`, applies `f_boxcox()` when Shapiro-Wilk indicates non-normality. If `"bestnormalize"`, applies `f_bestNormalize()`. If `FALSE` or `"none"`, no transformation is applied. **Note:** For paired tests, `bestNormalize` (Yeo-Johnson) is always used on the differences, since Box-Cox requires strictly positive values. Default is `TRUE`.

`force_transformation` Character vector. Names of variables to transform regardless of normality results.

`alpha` Numeric. Significance level. Default is `0.05`.

`intro_text` Logical. If `TRUE`, includes explanation of t-test assumptions. Default is `TRUE`.

`close_generated_files` Logical. Closes open Excel/Word files before writing. Default `FALSE`. **Windows only.**

`open_generated_files` Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.

`output_type` Character string specifying the output format. Default is `"default"`.

- `"default"`: Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use `print(result)` or `plot(result)` to display the returned object.
- `"console"`: Forces immediate printing to the console regardless of object assignment.
- `"pdf"`, `"word"`, `"excel"`: Saves results to a file of the corresponding format. See `save_as`, `save_in_wdir`, and `open_generated_files` for file path and opening behavior.
- `"rmd"`: Stores the raw markdown string inside the returned object for use in R Markdown documents.

`save_as` Character. Specific path/filename for output.

`save_in_wdir` Logical. Save in working directory. Default `FALSE`.

`y` Optional numeric vector (second group) for two-sample tests if using the vector interface. Ignored when a formula is supplied.

Value

An object of class `'f_t_test'`, a named list with one element per response variable. Each element contains the t-test result, normality test results, variance diagnostic results, transformation object (if applied), and back-transformed confidence interval (if applicable).

Author(s)

Sander H. van Delden <plantmind@proton.me>

References

Delacre, M., Lakens, D., & Leys, C. (2017). Why psychologists should by default use Welch's t-test instead of Student's t-test. *International Review of Social Psychology*, 30(1), 92-101. doi:10.5334/irsp.82

Examples

```
# 1. Two-sample independent Welch's t-test (default)
f_t_test(mpg ~ am, data = mtcars, output_type = "console", norm_plots = FALSE)

# 2. Multiple response variables in one call
f_t_test(mpg + hp ~ am, data = mtcars, output_type = "console", norm_plots = FALSE)

# 3. One-sample t-test: test if mean mpg equals 20
f_t_test(mpg ~ 1, data = mtcars, mu = 20,
         output_type = "console", norm_plots = FALSE)

# 4. Paired t-test (sleep dataset is already in AABB order)
f_t_test(extra ~ group, data = sleep, paired = TRUE,
         output_type = "console", norm_plots = FALSE)

# 5. Vector interface: two-sample independent
group_auto   <- mtcars$mpg[mtcars$am == 0]
group_manual <- mtcars$mpg[mtcars$am == 1]
f_t_test(group_auto, group_manual, output_type = "console", norm_plots = FALSE)

# 6. Vector interface: one-sample
f_t_test(mtcars$mpg, mu = 20, output_type = "console", norm_plots = FALSE)

# 7. Force Student's t-test (equal variances assumed)
f_t_test(mpg ~ am, data = mtcars, var.equal = TRUE,
         output_type = "console", norm_plots = FALSE)

# 8. One-sided test
f_t_test(mpg ~ am, data = mtcars, alternative = "greater",
         output_type = "console", norm_plots = FALSE)

# 9. Custom significance level (alpha = 0.01 is equivalent to conf.level = 0.99)
f_t_test(mpg ~ am, data = mtcars, alpha = 0.01,
         output_type = "console", norm_plots = FALSE)

# 10. Box-Cox transformation with back-transformed CI
# The back-transformed CI estimates the MEDIAN, not the arithmetic mean.
result <- f_t_test(hp ~ am, data = mtcars, transformation = TRUE,
                   output_type = "console", norm_plots = FALSE)
result[["hp"]]$ci_backtransformed

# 11. One-sample with non-zero mu and back-transformation
f_t_test(hp ~ 1, data = mtcars, mu = 100, transformation = TRUE,
         output_type = "console", norm_plots = FALSE)

# 12. BestNormalize transformation (set seed for reproducibility)
```

```
set.seed(123)
f_t_test(hp ~ am, data = mtcars, transformation = "bestnormalize",
         output_type = "console", norm_plots = FALSE)

# 13. Force transformation regardless of normality
f_t_test(mpg + hp ~ am, data = mtcars, force_transformation = "mpg",
         output_type = "console", norm_plots = FALSE)

# 14. Suppress transformation (diagnostic mode)
f_t_test(hp ~ am, data = mtcars, transformation = FALSE,
         output_type = "console", norm_plots = FALSE)

# 15. Access return object fields directly
result <- f_t_test(mpg + hp ~ am, data = mtcars,
                   output_type = "default", norm_plots = FALSE, intro_text = FALSE)
result[["mpg"]]$t_test          # standard htest object
result[["hp"]]$shapiro_res      # Shapiro-Wilk result
result[["hp"]]$homog_p_bartlett # Bartlett p-value (diagnostic only)
result[["hp"]]$homog_p_levene   # Levene p-value (diagnostic only)
result[["hp"]]$ci_backtransformed # back-transformed CI if transformed
```

`f_wilcox_test` *Perform multiple Wilcoxon rank sum and signed rank tests with inspection and visualization.*

Description

Performs One-sample (Wilcoxon signed rank), Two-sample independent (Wilcoxon rank sum / Mann-Whitney U), or Paired (Wilcoxon signed rank) tests on a given dataset. Several response parameters can be analysed in sequence (formula interface). Additionally, a vector interface similar to stats::wilcox.test() is supported.

Usage

```
f_wilcox_test(x, ...)

## S3 method for class 'formula'
f_wilcox_test(
  formula,
  data = NULL,
  paired = FALSE,
  conf.level = NULL,
  mu = 0,
  alternative = "two.sided",
  norm_plots = TRUE,
  alpha = 0.05,
  intro_text = TRUE,
  close_generated_files = FALSE,
```

```
  open_generated_files = TRUE,
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  ...
)

## Default S3 method:
f_wilcox_test(
  x,
  y = NULL,
  paired = FALSE,
  conf.level = NULL,
  mu = 0,
  alternative = "two.sided",
  norm_plots = TRUE,
  alpha = 0.05,
  intro_text = TRUE,
  close_generated_files = FALSE,
  open_generated_files = TRUE,
  output_type = "default",
  save_as = NULL,
  save_in_wdir = FALSE,
  ...
)
```

Arguments

x	Numeric vector of data values (one-sample or first group for two-sample), or a formula of the form `response ~ group` or `response ~ 1`.
...	For the formula method: additional arguments forwarded to the row-filtering step. The arguments `subset` and `na.action` are honored: when supplied, they are spliced (still unevaluated) into a `model.frame` call built once before the per-response loop, so the `subset` expression is evaluated in the data's column scope (e.g. `subset = cyl == 6` works). All responses in a multi-response call (`y1 + y2 ~ group`) are then tested on the identical row set. For the default (vector) method, ... is currently unused.
formula	A formula specifying the model (alternative to using x/y). • **Two-sample (Independent/Paired):** `response ~ group` (where group has exactly 2 levels). • **One-sample:** `response ~ 1` or `response ~ NULL`. More response variables can be added using + (e.g., `y1 + y2 ~ group`).
data	A data frame containing the variables when using the formula interface.
paired	Logical. If `TRUE`, performs a paired Wilcoxon test. **Note on row order:** For the formula interface, data must be sorted so that the observations in the two groups match row-for-row (i.e. row 1 of group 1 is paired with row 1 of group 2). For the vector interface, `x` and `y` must have the same length.

Note on factor level order: The formula interface computes differences as `level1 - level2` based on factor level order, which defaults to alphabetical. To control the direction, set the reference level explicitly:

```
# Set levels at creation
group <- factor(group, levels = c("pre", "post"))

# Or relevel an existing factor
group <- relevel(group, ref = "pre")
```

A reversed level order flips the sign of the estimate and CI but does not affect the W statistic or p-value.

`conf.level` Numeric. Confidence level of the interval. Default is `1 - alpha`. If `conf.level` is specified, then `alpha <- 1 - conf.level`.

`mu` Numeric. The hypothesised value of the pseudo-median (one-sample) or location shift (paired/two-sample) under H0. Default is 0.

`alternative` Character string. `"two.sided"` (default), `"greater"`, or `"less"`.

`norm_plots` Logical. If `TRUE`, descriptive diagnostic plots are included in the output. Default is `TRUE`.

`alpha` Numeric. Significance level. Default is `0.05`.

`intro_text` Logical. If `TRUE`, includes explanation about Wilcoxon test assumptions.

`close_generated_files`
Logical. Closes open Excel/Word files before writing. Works on Windows (taskkill), macOS (pkill) and Linux (pkill/soffice). Default `FALSE`. **WARNING:** Always save your work before using this option!

`open_generated_files`
Logical. Whether to open the generated output files after creation. Defaults to `TRUE` in an interactive R session and `FALSE` otherwise (e.g. in scripts or automated pipelines). Set to `TRUE` or `FALSE` to override this behaviour explicitly.

`output_type` Character string specifying the output format. Default is `"default"`.

- `"default"`: Returns the object and lets R decide whether to print; auto-prints if unassigned, silent if assigned to a variable. Use `print(result)` or `plot(result)` to display the returned object.
- `"console"`: Forces immediate printing to the console regardless of object assignment.
- `"pdf"`, `"word"`, `"excel"`: Saves results to a file of the corresponding format. See `save_as`, `save_in_wdir`, and `open_generated_files` for file path and opening behavior.
- `"rmd"`: Stores the raw markdown string inside the returned object for use in R Markdown documents.

`save_as` Character. Specific path/filename for output.

`save_in_wdir` Logical. Save in working directory.

`y` Optional numeric vector (second group) for two-sample tests if using the vector interface. Ignored when a formula is supplied.

Value

An object of class `'f_wilcox_test'`.

Median vs Pseudo-median

By default this function calls `stats::wilcox.test(conf.int = TRUE)`, which bases its confidence interval and hypothesis test on the **Hodges-Lehmann estimator**, not the raw sample median. This is standard behaviour of the Wilcoxon test, not something specific to this function. This function explicitly labels the estimator for what it is, because it is commonly mislabelled as "CI for the median" in textbooks and software output.

The estimator works differently depending on the test type:

One-sample: The *pseudo-median* is the middle value of all possible pairwise averages of your data points (including each value paired with itself). For a perfectly symmetric distribution it equals the sample median; for skewed data the two can differ.

Paired: The paired differences (observation 1 minus observation 2 within each pair) are computed first, and the pseudo-median of those differences is estimated. This is conceptually a one-sample problem applied to the differences, not a comparison of two independent groups. The CI is for the pseudo-median of the differences, **not** for the difference between the two separate sample medians.

Two-sample independent: The *location shift* is the median of all $n_1 \times n_2$ pairwise differences (one value from Group 1 minus one from Group 2). It answers: *by how much does a randomly chosen value from Group 1 tend to exceed a randomly chosen value from Group 2?* When both groups have the same distributional shape it equals the raw difference in sample medians; when shapes differ, the two values can diverge.

In all three cases the sample median(s) are reported separately for descriptive purposes only.

`plot.f_bestNormalize` *Plot an f_bestNormalize object*

Description

Plots diagnostics for an object of class `f_bestNormalize`.

Usage

```
## S3 method for class 'f_bestNormalize'
plot(x, which = 1:2, ask = FALSE, ...)
```

Arguments

`x`	An object of class `f_bestNormalize`.
`which`	Integer determining which graph to plot. Default is `1:2`.
`ask`	Logical. `TRUE` waits with plotting each graph until <Return> is pressed. Default is `FALSE`.
`...`	Further arguments passed to or from other methods.

Details

Plot method for f_bestNormalize objects

Value

This function is called for its side effect of generating plots and does not return a useful value. It invisibly returns 'NULL'.

`plot.f_boxcox` *Plot an f_boxcox object*

Description

Create diagnostic plots of an object of class `f_boxcox`.

Usage

```
## S3 method for class 'f_boxcox'
plot(x, which = 1:3, ask = FALSE, ...)
```

Arguments

`x`	An object of class `f_boxcox`.
`which`	Integer determining which graph to plot. Default is `1:2`.
`ask`	Logical. `TRUE` waits with plotting each graph until <Return> is pressed. Default is `FALSE`.
`...`	Further arguments passed to or from other methods.

Details

Plot method for f_boxcox objects

Value

This function is called for its side effect of generating plots and does not return a useful value. It invisibly returns `1`.

plot.f_kruskal_test *Plot method for f_kruskal_test objects*

Description

Displays the density plot and/or boxplot stored in an `f_kruskal_test` object. Plots are only available when the original call used `plot = TRUE`.

Usage

```
## S3 method for class 'f_kruskal_test'
plot(x, which = c("distributions", "Boxplot"), ...)
```

Arguments

`x`	An object of class `f_kruskal_test`.
`which`	Character vector indicating which plots to show. Options are `"distributions"` (density plot), `"Boxplot"`, or both (default).
`...`	Additional arguments (currently ignored).

Value

Returns `x` invisibly.

Examples

```
result <- f_kruskal_test(Sepal.Width ~ Species, data = iris,
                         output_type = "default")
plot(result)                          # both plots
plot(result, which = "Boxplot")       # boxplot only
```

plot.f_lmer *Plot method for f_lmer objects*

Description

Replays the four-panel diagnostic figure (residuals vs fitted, Q-Q of residuals, Q-Q of random-effect BLUPs, scale-location) produced by `f_lmer()`.

Usage

```
## S3 method for class 'f_lmer'
plot(x, ...)
```

Arguments

x	An object of class f_lmer.
...	Additional arguments (currently ignored).

Value

Returns x invisibly.

---

plot.f_long *Plot method for f_long objects*

Description

Automatically runs a f_scan diagnostic plot on data created by f_long.

Usage

```
## S3 method for class 'f_long'
plot(x, summary = TRUE, ...)
```

Arguments

x	An object of class f_long (output from f_long).
summary	Logical. If TRUE, generates the summary table within the scan. Default is TRUE.
...	Additional arguments passed to f_scan.

Value

Returns the output of f_scan (an object of class f_scan) invisibly.

---

predict.f_boxcox *Predict method for f_boxcox objects*

Description

Applies the fitted Box-Cox transformation to new data (forward transform), or reverses it back to the original scale (inverse transform). This is useful for transforming hypothesis test parameters (e.g., mu) to the transformed scale, or for back-transforming confidence intervals to the original scale.

Usage

```
## S3 method for class 'f_boxcox'
predict(object, newdata, inverse = FALSE, ...)
```

Arguments

`object`	An object of class `f_boxcox`, as returned by `f_boxcox`.
`newdata`	A numeric vector of values to transform. For the forward transform (`inverse = FALSE`), all values must be strictly positive (Box-Cox requires $y > 0$). For the inverse transform (`inverse = TRUE`), values are assumed to be on the Box-Cox transformed scale.
`inverse`	Logical. If `FALSE` (default), applies the forward Box-Cox transformation to `newdata` using the estimated λ from the original fit. If `TRUE`, reverses the transformation, mapping values from the Box-Cox scale back to the original scale. Default is `FALSE`.
`...`	Further arguments passed to or from other methods (currently unused).

Details

The forward transformation applies the standard Box-Cox formula:

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(y), & \lambda = 0 \end{cases}$$

The inverse transformation reverses this process to recover the original scale:

$$y = \begin{cases} (y(\lambda) \cdot \lambda + 1)^{1/\lambda}, & \lambda \neq 0 \\ \exp(y(\lambda)), & \lambda = 0 \end{cases}$$

Note on inverse validity: When $\lambda > 0$, not all transformed-scale values have a valid inverse. If $y(\lambda) \cdot \lambda + 1 \leq 0$, the result is undefined and `NaN` is returned with a warning.

Value

A numeric vector of the same length as `newdata`, containing either the forward-transformed or back-transformed values.

See Also

`f_boxcox`

Examples

```
# Assuming mtcars is available and f_boxcox is loaded
bc <- f_boxcox(mtcars$hp)

# Forward: transform a hypothesis value (mu) to the Box-Cox scale
mu <- 100
mu_transformed <- predict(bc, newdata = mu)

# Inverse: back-transform a confidence interval to the original scale
ci_transformed <- c(5.5, 6.8)
predict(bc, newdata = ci_transformed, inverse = TRUE)
```

```
# Round-trip sanity check should return exactly mu (e.g., 100)
predict(bc, newdata = mu_transformed, inverse = TRUE)
```

`print.f_outliers` *Print method for f_outliers objects*

Description

Prints a formatted summary table to the console.

Usage

```
## S3 method for class 'f_outliers'
print(
  x,
  col_width = 6,
  table_width = 90,
  digits = 2,
  allow_integer_decimal_mix = FALSE,
  ...
)
```

Arguments

`x`	Object of class f_outliers.
`col_width`	Integer. Max characters in header before line break. Default `6`.
`table_width`	Integer or `NULL`. Characters after which table splits. Default `90`.
`digits`	Integer. Number of decimal digits to use in formatting. Default is `3`.
`allow_integer_decimal_mix`	Logical. If `TRUE`, each individual cell is evaluated: integer values are displayed without decimal places, and non-integer values are displayed with the specified number of decimal places, i.e. `digits`. Default is `FALSE`, when a column contains a mix of integers and decimal values, all values are displayed with the specified number of decimal places. Note: columns containing only integers are **always** displayed without decimal places, regardless of `allow_integer_decimal_mix`.
`...`	Additional arguments passed to `pander`.

Value

Invisibly returns `1`.

print.f_scan *Print method for f_scan objects*

Description

Print method for f_scan objects

Summary method for f_scan objects

Plot method for f_scan objects

Usage

```
## S3 method for class 'f_scan'
print(
  x,
  summary = TRUE,
  outliers = TRUE,
  boxplot = TRUE,
  histogram = TRUE,
  qqplot = TRUE,
  main_plot = TRUE,
  advice = TRUE,
  digits = 3,
  ...
)

## S3 method for class 'f_scan'
summary(object, digits = 3, ...)

## S3 method for class 'f_scan'
plot(x, boxplot = TRUE, histogram = TRUE, qqplot = TRUE, main_plot = TRUE, ...)
```

Arguments

x	An f_scan object.
summary	Logical. Print summary statistics table? Default TRUE.
outliers	Logical. Print outlier table? Default TRUE.
boxplot, histogram, qqplot, main_plot	Logical. Which plots to render?
advice	Logical. Print statistical test recommendations? Default TRUE (shown only if advice=TRUE was used during f_scan).
digits	Integer. Decimal places for printed tables. Default 3.
...	Further arguments passed to or from other methods. Currently unused by the f_scan methods themselves, but accepted so the methods remain consistent with the base generics print, summary, and plot.
object	f_scan object to make a summary table from.

print.f_stat_wizard *Print method for f_stat_wizard*

Description

Print method for f_stat_wizard

Usage

```
## S3 method for class 'f_stat_wizard'
print(x, plots = TRUE, ...)
```

Arguments

x	An object of class f_stat_wizard.
plots	Logical. If TRUE, display diagnostic plots (histogram and QQ-plot) if they were generated with plots=TRUE. Default TRUE.
...	Additional arguments (ignored).

print.f_summary *Print method for f_summary objects*

Description

Prints a formatted summary table to the console.

Usage

```
## S3 method for class 'f_summary'
print(
  x,
  col_width = 6,
  table_width = 90,
  digits = 2,
  allow_integer_decimal_mix = FALSE,
  ...
)
```

Arguments

x	Object of class f_summary.
col_width	Integer. Max characters in header before line break. Default 6.
table_width	Integer or NULL. Characters after which table splits. Default 90.
digits	Integer. Number of decimal digits to use in formatting. Default is 3.

allow_integer_decimal_mix
: Logical. If TRUE, each individual cell is evaluated: integer values are displayed without decimal places, and non-integer values are displayed with the specified number of decimal places, i.e. digits. Default is FALSE, when a column contains a mix of integers and decimal values, all values are displayed with the specified number of decimal places. Note: columns containing only integers are **always** displayed without decimal places, regardless of allow_integer_decimal_mix.

...
: Additional arguments passed to pander.

Value

Invisibly returns 1.

---

summary.f_long *Summary method for f_long objects*

Description

Automatically runs the f_summary function on data created by f_long using the attributes stored in the object.

Usage

```
## S3 method for class 'f_long'
summary(object, ...)
```

Arguments

object
: An object of class f_long (output from f_long).

...
: Additional arguments passed to f_summary.

Value

Returns the summary table (usually a data frame or tibble) produced by f_summary.

Index