

Package ‘spectrakit’

January 26, 2026

Type Package
Title Spectral Data Handling and Visualization
Version 0.1.2
Description Provides functions to combine, normalize and visualize spectral data, perform principal component analysis (PCA), and assemble customizable image grids suitable for publication-quality scientific figures.
License MIT + file LICENSE
Encoding UTF-8
Imports dplyr, readr, ggplot2, ggrepel, tibble, purrr, rlang, magick, glue, data.table, stats, grid
Suggests RColorBrewer
RoxygenNote 7.3.3
NeedsCompilation no
Author Gianluca Pastorelli [aut, cre] (ORCID: <https://orcid.org/0000-0001-6926-1952>)
Maintainer Gianluca Pastorelli <gianluca.pastorelli@gmail.com>
Repository CRAN
Date/Publication 2026-01-26 20:10:03 UTC

Contents

combineSpectra	2
makeComposite	3
plotPCA	6
plotSpectra	8
Index	11

Description

Reads spectral data from multiple files in a folder, merges them by a common column (e.g., wavelength), optionally filters by a range, applies normalization and returns the data in either row-wise or column-wise format.

Usage

```
combineSpectra(
  folder = ".",
  file_type = "csv",
  sep = ",",
  header = TRUE,
  common_col_pos = 1,
  data_col_pos = 2,
  range = NULL,
  normalization = c("none", "simple", "min-max", "z-score"),
  orientation = c("columns", "rows")
)
```

Arguments

folder	Character. Path to the folder containing spectral files. Default is working directory ("").
file_type	Character. File extension (without dot) to search for. Default is "csv".
sep	Character. Delimiter for file columns. Use "," for comma-separated (default) or "\t" for tab-delimited files.
header	Logical. Whether the files contain a header row. Default is 'TRUE'.
common_col_pos	Integer. Column position for the common variable (e.g., wavelength). Defaults to '1'.
data_col_pos	Integer. Column position for the spectral intensity values. Defaults to '2'.
range	Numeric vector of length 2. Optional range filter for the common column (e.g., wavelength limits). Defaults to 'NULL' (no filtering).
normalization	Character. Normalization method to apply to spectra. One of "none", "simple" (divide by max), "min-max", or "z-score". Default is "none".
orientation	Character. Output orientation. Use "columns" (default) to keep each spectrum as a column, or "rows" to transpose so each spectrum is a row.

Value

A ‘tibble’ that can be exported as, for example, a CSV file. Each spectrum is either a column (default) or row, depending on ‘orientation’; a sum or mean spectrum can optionally be computed using ‘rowSums()’ or ‘colSums()’, or ‘rowMeans()’ or ‘colMeans()’, respectively. The common column (e.g., wavelength) is retained.

Examples

```
# Create a temporary directory for mock CSV files
tmp_dir <- tempdir()

# Define file paths
tmp1 <- file.path(tmp_dir, "file1.csv")
tmp2 <- file.path(tmp_dir, "file2.csv")

# Write two mock CSV files in the temporary folder
write.csv(data.frame(ID = c("A", "B", "C"), val = c(1, 2, 3)), tmp1, row.names = FALSE)
write.csv(data.frame(ID = c("A", "B", "C"), val = c(4, 5, 6)), tmp2, row.names = FALSE)

# Merge the CSV files in the temporary folder, normalize with z-score, and return transposed
result <- combineSpectra(
  folder = tmp_dir,
  file_type = "csv",
  sep = ",",
  common_col_pos = 1,
  data_col_pos = 2,
  normalization = "z-score",
  orientation = "rows"
)
```

makeComposite

Create a labeled image grid

Description

Generates a composite image grid with customizable layout, labels and resizing options. Suitable for spectra and other image types.

Usage

```
makeComposite(
  folder = ".",
  custom_order = NULL,
  rows = NULL,
  cols = NULL,
  spacing = 15,
  resize_mode = c("none", "fit", "fill", "width", "height", "both"),
```

```

labels = list(),
label_settings = list(),
background_color = "white",
desired_width = 15,
width_unit = "cm",
ppi = 300,
output_format = "tiff",
output_folder = NULL
)

```

Arguments

folder	Character. Path to the folder containing images. Default is working directory ('.').
custom_order	Character vector. Set of filenames (use NA for blank slots).
rows	Integer. Number of rows in the grid.
cols	Integer. Number of columns in the grid.
spacing	Integer. Spacing (in pixels) between tiles. Default is '15'.
resize_mode	Character. Method to resize panels in the composite. Options are: "none" Keep each panel at its original size. "fit" Scale each panel to fit within the smallest image dimensions, preserving aspect ratio. "fill" Scale and crop each panel to exactly fill the smallest dimensions. "width" Resize each panel to the minimum width, keeping the original height. "height" Resize each panel to the minimum height, keeping the original width. "both" Force panels to the exact width and height, which may distort aspect ratio.
labels	List of up to 4 character vectors. Each vector corresponds to one label layer and must be the same length as the number of non-NA images. Use empty strings "" or NULL entries to omit specific labels.
label_settings	List of named lists. Each named list specifies settings for a label layer. Options include: 'size' Font size (e.g., 100). 'color' Font color. 'font' Font family (e.g., "Arial"). 'boxcolor' Background color behind text, or 'NA' for none. 'location' Offset from the gravity anchor (e.g., "+10+10"). 'gravity' Placement anchor for the label (e.g., "northwest"). 'weight' Font weight (e.g., '400' = normal, '700' = bold).
background_color	Character. Background color used for blank tiles and borders. Use "none" for transparency. Default is "white".
desired_width	Numeric. Desired width of final image (in cm or px). Default is '15'.
width_unit	Character. Either "cm" or "px". Default is "cm".

ppi	Numeric. Resolution (pixels per inch) for output file. Default is '300'
output_format	Character. File format for saving plots. Examples: "tiff", "png", "pdf". Default is "tiff".
output_folder	Character. Path to folder where image is saved. If NULL (default), image is not saved; if ".", image is saved in the working directory.

Value

Saves image composite to a specified output folder. Returns 'NULL' (used for side-effects).

Examples

```
library(magick)

tmp_dir <- file.path(tempdir(), "spectrakit_imgs")
dir.create(tmp_dir, showWarnings = FALSE)

# Create and save img1
img1 <- image_blank(100, 100, "white")
img1 <- image_draw(img1)
symbols(50, 50, circles = 30, inches = FALSE, add = TRUE, bg = "red")
dev.off()
img1_path <- file.path(tmp_dir, "img1.png")
image_write(img1, img1_path)

# Create and save img2
img2 <- image_blank(100, 100, "white")
img2 <- image_draw(img2)
rect(20, 20, 80, 80, col = "blue", border = NA)
dev.off()
img2_path <- file.path(tmp_dir, "img2.png")
image_write(img2, img2_path)

# Create composite
makeComposite(
  folder = tmp_dir,
  custom_order = c("img1.png", "img2.png"),
  rows = 1,
  cols = 2,
  labels = list(c("Red Circle", "Blue Rectangle")),
  label_settings = list(
    list(size = 5, font = "Arial", color = "black", boxcolor = "white",
         gravity = "northwest", location = "+10+10", weight = 400)
  ),
  resize_mode = "none",
  desired_width = 10,
  width_unit = "cm",
  ppi = 300,
  output_format = "png",
  output_folder = tmp_dir
)
```

plotPCA

*Perform PCA and Create a Plot of Scores, Loadings, or Biplot***Description**

Computes principal component analysis (PCA) on numeric variables in a dataset and generates a PC1 vs PC2 visualization (scores, loadings, or biplot).

Usage

```
plotPCA(
  data,
  color_var = NULL,
  shape_var = NULL,
  plot_type = c("score", "loadings", "biplot"),
  palette = "Dark2",
  show_labels = TRUE,
  ellipses = FALSE,
  ellipse_var = NULL,
  display_names = TRUE,
  legend_title = NULL,
  return_pca = FALSE,
  output_format = "tiff",
  output_folder = NULL
)
```

Arguments

<code>data</code>	A data frame containing numeric variables. Non-numeric columns are ignored.
<code>color_var</code>	Optional character. Column name for coloring points by group. Converted to factor internally.
<code>shape_var</code>	Optional character. Column name for shaping points by group. Converted to factor internally.
<code>plot_type</code>	Character. Type of PCA plot to generate: "score" Plot PCA scores (observations). "loadings" Plot PCA loadings (variables). "biplot" Combine scores and loadings in a biplot.
<code>palette</code>	Character or vector. Color palette for groups: "Dark2" Use Dark2 palette from RColorBrewer (requires the package). single color A single color repeated for all groups. vector of colors Custom vector of colors, recycled to match number of groups.
<code>show_labels</code>	Logical. Display labels for points (scores) or variables (loadings). Default is TRUE.

ellipses	Logical. Draw confidence ellipses around groups in score/biplot. Grouping logic for ellipses follows this priority: • If ellipse_var is provided, ellipses are drawn by that variable. • Else, if color_var is provided, ellipses are drawn by color groups. • Else, if shape_var is provided, ellipses are drawn by shape groups. • If none are provided, no ellipses are drawn. Default is FALSE.
ellipse_var	Optional character. Name of the variable used to group ellipses. Takes precedence over all other grouping variables. Converted to factor internally. Default is NULL.
display_names	Logical. Show legend if TRUE. Default is TRUE.
legend_title	Optional character. Legend title corresponding to 'color_var' or 'shape_var'. Default is NULL.
return_pca	Logical. If TRUE, return a list with plot and PCA object. Default is FALSE.
output_format	Character. File format for saving plots. Examples: "tiff", "png", "pdf". Default is "tiff".
output_folder	Character. Path to folder where plots are saved. If NULL (default), returns a ggplot object (or list with plot and PCA if 'return_pca = TRUE'). If specified, plot is saved automatically (function returns PCA object only if 'return_pca = TRUE'); if ".", plot is saved in the working directory.

Value

A ggplot2 object representing the PCA plot, or a list with 'plot' and 'pca' if 'return_pca = TRUE'.

Examples

```
plotPCA(
  data = iris,
  color_var = "Species",
  shape_var = "Species",
  plot_type = "biplot",
  palette = "Dark2",
  show_labels = TRUE,
  ellipses = FALSE,
  display_names = TRUE,
  legend_title = "Iris Species"
)
```

plotSpectra

*Plot Spectral Data from Multiple Files***Description**

Reads, normalizes and plots spectral data from files in a folder. Supports multiple plot modes, color palettes, axis customization, annotations and automatic saving of plots to files.

Usage

```
plotSpectra(
  folder = ".",
  file_type = "csv",
  sep = ",",
  header = TRUE,
  normalization = c("none", "simple", "min-max", "z-score", "area", "vector"),
  x_config = NULL,
  x_reverse = FALSE,
  y_trans = c("linear", "log10", "sqrt"),
  x_label = NULL,
  y_label = NULL,
  line_size = 0.5,
  palette = "black",
  plot_mode = c("individual", "overlapped", "stacked"),
  display_names = FALSE,
  vertical_lines = NULL,
  shaded_ROIs = NULL,
  annotations = NULL,
  output_format = "tiff",
  output_folder = NULL
)
```

Arguments

folder	Character. Path to the folder containing spectral files. Default is working directory ("").
file_type	Character. File extension (without dot) to search for. Default is "csv".
sep	Character. Delimiter for file columns. Use "," for comma-separated (default) or "\t" for tab-delimited files.
header	Logical. Whether the files contain a header row. Default is 'TRUE'.
normalization	Character. Normalization method to apply to y-axis data. Options are: "none" No normalization is applied (default). "simple" Divide by the maximum intensity. "min-max" Scale intensities to the [0,1] range.

	"z-score" Subtract the mean and divide by the standard deviation of intensities.
	"area" Divide by the total sum of intensities so the spectrum area = 1.
	"vector" Normalize the spectrum as a unit vector by dividing by the square root of the sum of squared intensities (L2 normalization).
x_config	Numeric vector of length 3. Specifies x-axis range and breaks: 'c(min, max, step)'.
x_reverse	Logical. If 'TRUE', reverses the x-axis. Default is 'FALSE'.
y_trans	Character. Transformation for the y-axis. One of "linear", "log10", or "sqrt". Default is "linear".
x_label	Character or expression. Label for the x-axis. Supports mathematical notation via 'expression()'.
y_label	Character or expression. Label for the y-axis. Supports mathematical notation via 'expression()'.
line_size	Numeric. Width of the spectral lines. Default is '0.5'.
palette	Character or vector. Color setting: a single color (e.g., "black"), a ColorBrewer palette name (e.g., "Dark2"), or a custom color vector.
plot_mode	Character. Plotting style. One of "individual" (one plot per spectrum), "overlapped" (all in one), or "stacked" (faceted). Default is "individual".
display_names	Logical. If 'TRUE', adds file names as titles to individual spectra or a legend to combined spectra. Default is 'FALSE'.
vertical_lines	Numeric vector. Adds vertical dashed lines at given x positions.
shaded_ROIs	List of numeric vectors. Each vector must have two elements ('xmin', 'xmax') to define shaded x regions.
annotations	Data frame with columns 'file' (file name without extension), 'x', 'y', and 'label'. Adds annotation labels to specific points in spectra.
output_format	Character. File format for saving plots. Examples: "tiff", "png", "pdf". Default is "tiff".
output_folder	Character. Path to folder where plots are saved. If NULL (default), plots are not saved; if ".", plots are saved in the working directory.

Details

Color settings can support color-blind-friendly palettes from 'RColorBrewer'. Use 'display.brewer.all(colorblindFriendly = TRUE)' to preview.

Value

Saves plots to a specified output folder. Returns 'NULL' (used for side-effects).

Examples

```
# Create a temporary directory and write mock spectra files
tmp_dir <- tempdir()
write.csv(data.frame(Energy = 0:30, Counts = rpois(31, lambda = 100)),
          file.path(tmp_dir, "spec1.csv"), row.names = FALSE)
write.csv(data.frame(Energy = 0:30, Counts = rpois(31, lambda = 120)),
          file.path(tmp_dir, "spec2.csv"), row.names = FALSE)

# Plot the mock spectra using various configuration options
plotSpectra(
  folder = tmp_dir,
  file_type = "csv",
  sep = ",",
  normalization = "min-max",
  x_config = c(0, 30, 5),
  x_reverse = FALSE,
  y_trans = "linear",
  x_label = expression(Energy~(keV)),
  y_label = expression(Counts/1000~s),
  line_size = 0.7,
  palette = c("black", "red"),
  plot_mode = "overlapped",
  display_names = TRUE,
  vertical_lines = c(10, 20),
  shaded_ROIs = list(c(12, 14), c(18, 22)),
  output_format = "png",
  output_folder = tmp_dir
)
```

Index

`combineSpectra`, [2](#)

`makeComposite`, [3](#)

`plotPCA`, [6](#)

`plotSpectra`, [8](#)