Package: tidyspec (via r-universe)

October 21, 2026

Type Package

Title Spectroscopy Analysis Using the Tidy Data Philosophy

Version 0.4.0

Maintainer Marcel Ferreira <marcel.ferreira@unesp.br>

Description Enables the analysis of spectroscopy data such as infrared ('IR'), Raman, and nuclear magnetic resonance ('NMR') using the tidy data framework from the 'tidyverse'. The 'tidyspec' package provides functions for data transformation, normalization, baseline correction, smoothing, derivatives, and both interactive and static visualization. It promotes structured, reproducible workflows for spectral data exploration and preprocessing. Implemented methods include Savitzky and Golay (1964) ``Smoothing and Differentiation of Data by Simplified Least Squares Procedures'' <doi:10.1021/ac60214a047>, Sternberg (1983) ``Biomedical Image Processing'' <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1654163>, Zimmermann and Kohler (1996) ``Baseline correction using the rolling ball algorithm'' <doi:10.1016/0168-583X(95)00908-6>, Beattie and Esmonde-White (2021) ``Exploration of Principal Component Analysis: Deriving Principal Component Analysis Visually Using Spectra'' <doi:10.1177/0003702820987847>, Wickham et al. (2019) ``Welcome to the tidyverse'' <doi:10.21105/joss.01686>, and Kuhn, Wickham and Hvitfeldt (2024) ``recipes: Preprocessing and Feature Engineering Steps for Modeling'' <https://CRAN.R-project.org/package=recipes>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://marceelrf.github.io/tidyspec

BugReports https://github.com/marceelrf/tidyspec/issues

RoxygenNote 7.2.3

Depends R (>= 4.1.0)

Imports akima, crayon, digest, dplyr, ggplot2, glue, plotly, pracma, purrr, RColorBrewer, recipes, readr, readxl, rlang, scales, signal, stats, tibble, tidyr, tidyselect, timetk, viridisLite

Suggests gridExtra, knitr, rmarkdown, tidyverse

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev libx11-dev

Repository https://marceelrf.r-universe.dev

Date/Publication 2026-04-13 19:45:09 UTC

RemoteUrl https://github.com/marceelrf/tidyspec

RemoteRef HEAD

RemoteSha 15b76027ce2602e3a28b52bb4e93a6001dd0d2fd

Contents

check_wn_col . . . 3
CoHAspec . . . 3
demo_rolling_ball . . . 4
plot_rolling_ball . . . 5
rolling_ball . . . 5
rolling_ball_morphology . . . 6
set_spec_wn . . . 7
smooth_baseline . . . 7
spec_abs2trans . . . 8
spec_bl_rollingBall . . . 8
spec_bl_rubberband . . . 9
spec_blc_rollingBall . . . 10
spec_blc_rubberband . . . 11
spec_diff . . . 12
spec_filter . . . 13
spec_interpolate_left . . . 13
spec_interpolate_regular . . . 14
spec_interpolate_right . . . 15
spec_join . . . 16
spec_norm_01 . . . 17
spec_norm_area . . . 17
spec_norm_by_wn . . . 18
spec_norm_minmax . . . 19
spec_norm_var . . . 19
spec_pca . . . 20
spec_pca_screeplot . . . 21
spec_pca_wn_contrib . . . 22
spec_read . . . 23
spec_select . . . 24
spec_smartplot . . . 24

spec_smartplotly 26
spec_smooth_avg 28
spec_smooth_sga 28
spec_trans2abs 29
spec_values_at 30

Index **32**

`check_wn_col` *Check the Currently Set Wavenumber Column*

Description

This function checks which column has been set as the default wavenumber column. If no column has been set, it prints a message prompting the user to define one.

Usage

```
check_wn_col(show_val = FALSE)
```

Arguments

`show_val`	Logical. If 'TRUE', the function returns the name of the current wavenumber column. Default is 'FALSE'.

Value

If 'show_val = TRUE', returns the name of the wavenumber column (character). Otherwise, it only prints a message.

Examples

```
set_spec_wn("Wavenumber")  # Set the wavenumber column
check_wn_col()             # Check which column is set
check_wn_col(show_val = TRUE)  # Check and return the column name
```

`CoHAspec` *CoHAspec Dataset*

Description

A dataset containing spectral absorbance measurements for different concentrations of CoHA.

Usage

```
CoHAspec
```

Format

A data frame with 6 rows and 5 columns:

Wavenumber Numeric. The spectral wavenumber (cm-1).

CoHA01 Numeric. Absorbance values for CoHA at 1 mM Cobalt concentration.

CoHA025 Numeric. Absorbance values for CoHA at 2.5 mM Cobalt concentration.

CoHA05 Numeric. Absorbance values for CoHA at 5 mM Cobalt concentration.

CoHA100 Numeric. Absorbance values for CoHA at 10 mM Cobalt concentration.

Source

de Almeida GS, Ferreira MR, da Costa Fernandes CJ, et al. Development of cobalt (Co)-doped monetites for bone regeneration. J Biomed Mater Res. 2024; 112(1):e35319. <doi:10.1002/jbm.b.35319>

Examples

```
data(CoHAspec)
head(CoHAspec)
```

---

demo_rolling_ball *Usage Example and Test*

Description

Function to demonstrate the use of implemented functions

Usage

```
demo_rolling_ball(verbose = TRUE)
```

Arguments

verbose	Logical indicating whether to print progress messages

Value

A list containing two elements:

simple Results from the simple rolling ball method

morphology Results from the mathematical morphology method

Examples

```
# Run the demonstration
demo_results <- demo_rolling_ball()

# Access results
simple_method <- demo_results$simple
morphology_method <- demo_results$morphology
```

plot_rolling_ball *Plot Rolling Ball Results*

Description

Convenience function to visualize the results

Usage

```
plot_rolling_ball(
  result,
  title = "Rolling Ball Baseline Correction",
  x_values = NULL
)
```

Arguments

result	Result from rolling_ball or rolling_ball_morphology function
title	Plot title
x_values	X-axis values (optional)

Value

No return value, called for side effects (creates a plot)

rolling_ball *Rolling Ball Baseline Correction*

Description

Implements the rolling ball baseline correction method

Usage

```
rolling_ball(x, wm, ws = 0)
```

Arguments

x	Numeric vector containing the spectrum/signal values
wm	Window width (ball radius). Larger values = smoother baseline
ws	Smoothing window width (optional)

Value

A list with three components:

baseline Numeric vector containing the estimated baseline values

corrected Numeric vector containing the baseline-corrected signal (original - baseline)

original Numeric vector containing the original input signal

Examples

```
# Example with simulated data
x <- seq(1, 100, by = 1)
y <- sin(x/10) + 0.1*x + rnorm(100, 0, 0.1)
result <- rolling_ball(y, wm = 10)
plot(x, y, type = "l", col = "blue", main = "Rolling Ball Correction")
lines(x, result$baseline, col = "red", lwd = 2)
lines(x, result$corrected, col = "green")
legend("topright", c("Original", "Baseline", "Corrected"),
       col = c("blue", "red", "green"), lty = 1)
```

---

rolling_ball_morphology

Enhanced Rolling Ball with Mathematical Morphology

Description

More sophisticated version of rolling ball using mathematical morphology concepts

Usage

```
rolling_ball_morphology(x, radius, smooth = TRUE)
```

Arguments

x	Numeric vector containing the spectrum/signal values
radius	Radius of the structuring ball
smooth	Apply additional smoothing (logical)

Value

A list with three components:

baseline Numeric vector containing the estimated baseline values using morphological operations

corrected Numeric vector containing the baseline-corrected signal (original - baseline)

original Numeric vector containing the original input signal

set_spec_wn *Set the Default Wavenumber Column*

Description

Defines a default wavenumber column name to be used by other functions in the package when ‘wn_col‘ is not explicitly provided.

Usage

```
set_spec_wn(col_name)
```

Arguments

col_name	A string specifying the name of the wavenumber column.

Value

Invisibly returns the assigned column name.

See Also

[spec_select()]

Examples

```
set_spec_wn("Wavenumber")
```

smooth_baseline *Baseline Smoothing*

Description

Auxiliary function to smooth the baseline using moving average

Usage

```
smooth_baseline(baseline, ws)
```

Arguments

baseline	Vector with the baseline
ws	Smoothing window width

Value

Smoothed numeric vector of the same length as input

spec_abs2trans *Convert Absorbance Data to Transmittance*

Description

This function converts absorbance data to transmittance using the formula $T = 10^{(2-A)}$, where A is the absorbance and T is the transmittance.

Usage

```
spec_abs2trans(.data, wn_col = NULL)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data in absorbance.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.

Value

A ‘tibble‘ with the converted transmittance data, containing the wavelength column and the numeric transmittance columns. Any rows with infinite values are removed.

spec_bl_rollingBall *Extract Rolling Ball Baseline from Spectral Data*

Description

This function extracts the rolling ball baseline from spectral data within a specified wavelength range. It returns only the baseline, not the corrected data.

Usage

```
spec_bl_rollingBall(
  .data,
  wn_col = NULL,
  wn_min = NULL,
  wn_max = NULL,
  wm,
  ws = 0,
  is_abs = TRUE
)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
wn_min	A numeric value specifying the minimum wavelength to consider for the baseline correction.
wn_max	A numeric value specifying the maximum wavelength to consider for the baseline correction.
wm	A numeric value for the window size of the rolling ball algorithm.
ws	A numeric value for the smoothing factor of the rolling ball algorithm.
is_abs	A logical value indicating whether the data is already in absorbance. If ‘TRUE‘, absorbance is used directly; if ‘FALSE‘, the data is converted to absorbance before extracting the baseline.

Value

A ‘tibble‘ with the baseline data, containing the wavelength column and the baseline for each numeric column.

References

Baseline estimation performed using a custom rolling ball implementation.

---

spec_bl_rubberband *Extract Rubberband Baseline from Spectral Data*

Description

This function extracts the rubberband baseline from spectral data within a specified wavelength range. The rubberband method fits a baseline by connecting local minima points in the spectrum. It returns only the baseline, not the corrected data.

Usage

```
spec_bl_rubberband(
  .data,
  wn_col = NULL,
  wn_min = NULL,
  wn_max = NULL,
  segment_length = 50,
  smooth_baseline = TRUE,
  is_abs = TRUE
)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
wn_min	A numeric value specifying the minimum wavelength to consider for the baseline correction.
wn_max	A numeric value specifying the maximum wavelength to consider for the baseline correction.
segment_length	A numeric value specifying the length of segments for finding local minima. Default is 50.
smooth_baseline	A logical value indicating whether to smooth the baseline using spline interpolation. Default is TRUE.
is_abs	A logical value indicating whether the data is already in absorbance. If ‘TRUE‘, absorbance is used directly; if ‘FALSE‘, the data is converted to absorbance before extracting the baseline.

Value

A ‘tibble‘ with the baseline data, containing the wavelength column and the baseline for each numeric column.

References

Rubberband baseline estimation connects local minima to estimate baseline.

---

spec_blc_rollingBall *Apply Rolling Ball Baseline Correction to Spectral Data*

Description

This function applies a rolling ball baseline correction to spectral data within a specified wavelength range. It allows for correction of either absorbance or transmittance data.

Usage

```
spec_blc_rollingBall(
  .data,
  wn_col = NULL,
  wn_min = NULL,
  wn_max = NULL,
  wm,
  ws = 0,
  is_abs = TRUE
)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
wn_min	A numeric value specifying the minimum wavelength to consider for the baseline correction.
wn_max	A numeric value specifying the maximum wavelength to consider for the baseline correction.
wm	A numeric value for the window size of the rolling ball algorithm.
ws	A numeric value for the smoothing factor of the rolling ball algorithm.
is_abs	A logical value indicating whether the data is already in absorbance. If ‘TRUE‘, absorbance is used directly; if ‘FALSE‘, the data is converted to absorbance before applying the baseline correction.

Value

A ‘tibble‘ with the baseline-corrected spectral data, containing the wavelength column and the corrected numeric columns.

References

Baseline estimation performed using a custom rolling ball implementation.

---

spec_blc_rubberband *Apply Rubberband Baseline Correction to Spectral Data*

Description

This function applies a rubberband baseline correction to spectral data within a specified wavelength range. The rubberband method fits a baseline by connecting local minima points in the spectrum. It allows for correction of either absorbance or transmittance data.

Usage

```
spec_blc_rubberband(
  .data,
  wn_col = NULL,
  wn_min = NULL,
  wn_max = NULL,
  segment_length = 50,
  smooth_baseline = TRUE,
  is_abs = TRUE
)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
wn_min	A numeric value specifying the minimum wavelength to consider for the baseline correction.
wn_max	A numeric value specifying the maximum wavelength to consider for the baseline correction.
segment_length	A numeric value specifying the length of segments for finding local minima. Default is 50.
smooth_baseline	A logical value indicating whether to smooth the baseline using spline interpolation. Default is TRUE.
is_abs	A logical value indicating whether the data is already in absorbance. If ‘TRUE‘, absorbance is used directly; if ‘FALSE‘, the data is converted to absorbance before applying the baseline correction.

Value

A ‘tibble‘ with the baseline-corrected spectral data, containing the wavelength column and the corrected numeric columns.

References

Rubberband baseline correction connects local minima to estimate baseline.

---

spec_diff *Apply Differentiation to Spectral Data*

Description

This function applies numerical differentiation to spectral data, allowing for the calculation of the first or higher-order differences.

Usage

```
spec_diff(.data, wn_col = NULL, degree = 1)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
degree	A numeric value specifying the degree of differentiation. If ‘degree‘ is 0, the original data is returned without any changes.

Value

A ‘tibble‘ with the differentiated spectral data, containing the wavelength column and the differentiated numeric columns. If ‘degree‘ is 0, the original data is returned.

`spec_filter` *Filter spectral data by wavenumber range*

Description

This function filters the spectral dataset based on a specified wavenumber (‘wn‘) range. It requires the wavenumber column to be previously set using [set_spec_wn()]. If ‘wn_min‘ and/or ‘wn_max‘ are provided, the data will be filtered accordingly. If neither is provided, the original dataset is returned unchanged.

Usage

```
spec_filter(.data, wn_min = NULL, wn_max = NULL)
```

Arguments

`.data`	A data frame containing spectral data.
`wn_min`	Optional numeric value. Minimum wavenumber value to keep.
`wn_max`	Optional numeric value. Maximum wavenumber value to keep.

Value

A filtered data frame based on the wavenumber column.

Examples

```
set_spec_wn("Wavenumber")
spec_filter(CoHAspec, wn_min = 500, wn_max = 1800)
```

`spec_interpolate_left` *Interpolate Spectral Data to Match Reference Wavenumbers (Left Join Style)*

Description

This function interpolates spectral data to match the wavenumber grid of a reference dataset. Works like a left join - keeps all wavenumbers from the reference data and interpolates the spectral data to match those wavenumbers.

Usage

```
spec_interpolate_left(
  .data,
  reference,
  wn_col = NULL,
  method = "pchip",
  extrapolate = FALSE,
  ...
)
```

Arguments

.data	A 'data.frame' or 'tibble' containing spectral data to be interpolated.
reference	A 'data.frame' or 'tibble' containing the reference wavenumber grid.
wn_col	A character string specifying the column name for the wavenumber data. If NULL, uses the default set with set_spec_wn().
method	A character string specifying the interpolation method. Options: "linear", "pchip", "spline", "akima". Default is "pchip".
extrapolate	Logical. Should values be extrapolated outside the original range? Default is FALSE (returns NA for out-of-range values).
...	Additional arguments passed to the interpolation method.

Value

A 'tibble' with interpolated spectral data using the reference wavenumber grid.

---

spec_interpolate_regular

Interpolate to Regular Grid

Description

Creates a regular wavenumber grid and interpolates spectral data to match it.

Usage

```
spec_interpolate_regular(
  .data,
  resolution = 1,
  wn_min = NULL,
  wn_max = NULL,
  wn_col = NULL,
  method = "pchip",
  ...
)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
resolution	Numeric. The step size for the regular grid.
wn_min, wn_max	Numeric. Range for the regular grid. If NULL, uses data range.
wn_col	A character string specifying the column name for the wavenumber data.
method	A character string specifying the interpolation method.
...	Additional arguments passed to the interpolation method.

Value

A ‘tibble‘ with spectral data interpolated to a regular grid.

---

spec_interpolate_right

Interpolate Spectral Data to Include Both Grids (Right Join Style)

Description

This function interpolates reference data to match the wavenumber grid of the main dataset. Works like a right join - keeps all wavenumbers from .data and interpolates reference data to match those wavenumbers.

Usage

```
spec_interpolate_right(
  .data,
  reference,
  wn_col = NULL,
  method = "pchip",
  extrapolate = FALSE,
  suffix = c(".x", ".y"),
  ...
)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing the main spectral data.
reference	A ‘data.frame‘ or ‘tibble‘ containing spectral data to be interpolated.
wn_col	A character string specifying the column name for the wavenumber data.
method	A character string specifying the interpolation method.
extrapolate	Logical. Should values be extrapolated outside the original range?
suffix	Character vector of length 2 specifying suffixes for overlapping column names.
...	Additional arguments passed to the interpolation method.

Value

A ‘tibble‘ combining .data with interpolated reference data.

`spec_join` *Join multiple spectral tibbles*

Description

‘spec_join()‘ combines two or more spectral tibbles that share the same wavenumber column. The function performs a full join operation, keeping all wavenumber values from all input tibbles.

Usage

```
spec_join(..., wn_col = NULL, join_type = "full", suffix = c(".x", ".y"))
```

Arguments

`...`	Two or more tibbles containing spectral data
`wn_col`	Character string specifying the wavenumber column name. If NULL, uses the globally set wavenumber column from ‘set_spec_wn()‘.
`join_type`	Character string specifying the type of join operation. Options: "full" (default), "inner", "left". See Details.
`suffix`	Character vector of length 2 specifying suffixes to add to non-wavenumber columns when there are naming conflicts. Default: c(".x", ".y")

Details

Join types: - "full": Keep all wavenumber values from all tibbles (default) - "inner": Keep only wavenumber values present in all tibbles - "left": Keep wavenumber values from the first tibble

The function will show warnings about the wavenumber column being used (similar to other tidyspec functions) and will handle missing values appropriately.

Value

A tibble containing the joined spectral data

Examples

```
## Not run:
# Set wavenumber column
set_spec_wn("Wavenumber")

# Join two spectral datasets
joined_data <- spec_join(CoHAspec, other_spec_data)

# Join multiple datasets with inner join
joined_data <- spec_join(spec1, spec2, spec3, join_type = "inner")
```

```
# Specify wavenumber column explicitly
joined_data <- spec_join(spec1, spec2, wn_col = "Wavenumber")

## End(Not run)
```

---

spec_norm_01 *Normalize Spectral Data to the [0, 1] Range*

Description

This function normalizes the numeric spectral data in each column to the [0, 1] range, preserving the wavelength column.

Usage

```
spec_norm_01(.data, wn_col = NULL)
```

Arguments

.data	A 'data.frame' or 'tibble' containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is '"Wn"'.

Value

A 'tibble' with the normalized spectral data, containing the wavelength column and the normalized numeric columns.

---

spec_norm_area *Normalize Spectral Data by Area Under the Curve*

Description

This function normalizes the numeric spectral data in each column so that the area under the curve equals a specified value (default is 1), while preserving the wavelength column.

Usage

```
spec_norm_area(.data, wn_col = NULL, target_area = 1)
```

Arguments

.data	A 'data.frame' or 'tibble' containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. If NULL, uses the default set by 'set_spec_wn()'.
target_area	A numeric value specifying the target area under the curve. Default is 1.

Value

A ‘tibble‘ with the area-normalized spectral data, containing the wavelength column and the normalized numeric columns.

`spec_norm_by_wn` *Normalize Spectral Data by Specific Wavenumber Value*

Description

This function normalizes the numeric spectral data in each column so that the intensity at a specified wavenumber equals a target value (default is 1), while preserving the wavelength column.

Usage

```
spec_norm_by_wn(
  .data,
  wn_col = NULL,
  target_wn,
  target_value = 1,
  method = "linear"
)
```

Arguments

`.data`	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
`wn_col`	A character string specifying the column name for the wavelength data. If NULL, uses the default set by ‘set_spec_wn()‘.
`target_wn`	A numeric value specifying the wavenumber to use as reference for normalization.
`target_value`	A numeric value specifying the target intensity value at the reference wavenumber. Default is 1.
`method`	A character string specifying the interpolation method when the exact wavenumber is not found. Options are "linear" (default) or "nearest".

Value

A ‘tibble‘ with the wavenumber-normalized spectral data, containing the wavelength column and the normalized numeric columns.

spec_norm_minmax *Normalize Spectral Data to a Specified Range*

Description

This function normalizes the numeric spectral data in each column to a specified range [min, max], preserving the wavelength column.

Usage

```
spec_norm_minmax(.data, wn_col = NULL, min = 0, max = 1)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
min	A numeric value specifying the minimum value of the desired range. Default is 0.
max	A numeric value specifying the maximum value of the desired range. Default is 1.

Value

A ‘tibble‘ with the normalized spectral data, containing the wavelength column and the normalized numeric columns.

spec_norm_var *Standardize Spectral Data to Unit Variance*

Description

This function standardizes the numeric spectral data in each column to have a mean of 0 and a standard deviation of 1 (unit variance), while preserving the wavelength column.

Usage

```
spec_norm_var(.data, wn_col = NULL)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.

Value

A ‘tibble‘ with the standardized spectral data, containing the wavelength column and the standardized numeric columns.

`spec_pca` *Perform Principal Component Analysis (PCA) on Spectral Data*

Description

This function computes a Principal Component Analysis (PCA) on spectral data, excluding the wavenumber column from the analysis.

Usage

```
spec_pca(.data, wn_col = NULL, scale = TRUE, center = TRUE)
```

Arguments

`.data`	A data frame containing spectral data, with one column representing wavenumbers and the remaining columns containing spectral intensity values.
`wn_col`	A string specifying the name of the column that contains the wavenumber values. If NULL, the function attempts to retrieve the default wavenumber column set by ‘set_spec_wn()‘.
`scale`	A logical value indicating whether the spectral data should be scaled (default is TRUE).
`center`	A logical value indicating whether the spectral data should be centered (default is TRUE).

Value

A ‘prcomp‘ object containing the PCA results, including principal components, standard deviations, and loadings.

Examples

```
set_spec_wn("Wavenumber")
pca_result <- spec_pca(CoHAspec)
summary(pca_result)
```

spec_pca_screeplot *Scree plot for PCA results*

Description

Creates a customizable scree plot based on a ‘prcomp‘ object showing variance explained by each component.

Usage

```
spec_pca_screeplot(
  pca,
  n = 10,
  show_labels = TRUE,
  show_cumulative = TRUE,
  bar_color = "steelblue",
  line_color = "darkred",
  show_kaiser = FALSE,
  title = "Scree Plot",
  subtitle = NULL,
  accuracy = 1
)
```

Arguments

pca	A PCA object returned by [prcomp()].
n	Number of components to display. Defaults to 10.
show_labels	Logical. Show percentage labels on bars? Default is TRUE.
show_cumulative	Logical. Show cumulative variance line? Default is TRUE.
bar_color	Fill color for bars. Default is "steelblue".
line_color	Color of the cumulative line and points. Default is "darkred".
show_kaiser	Logical. Show Kaiser criterion line? Default is FALSE.
title	Plot title. Default is "Scree Plot".
subtitle	Optional plot subtitle.
accuracy	Number of decimal places for variance percentages. Default is 1.

Value

A ggplot2 scree plot object.

Examples

```
pca <- prcomp(USArrests, scale. = TRUE)
spec_pca_screeplot(pca, n = 4)
spec_pca_screeplot(pca, show_kaiser = TRUE, bar_color = "darkblue")
```

spec_pca_wn_contrib *Compute Wavenumber Contributions to Principal Components*

Description

This function calculates the contribution of each wavenumber to the principal components (PCs) in a PCA result. Contributions are computed as the squared loadings multiplied by 100.

This function calculates the contribution of each wavenumber to the principal components (PCs) in a PCA result. Contributions are computed as the squared loadings multiplied by 100.

Usage

```
spec_pca_wn_contrib(PCA)

spec_pca_wn_contrib(PCA)
```

Arguments

PCA	An object of class 'prcomp', containing the results of a principal component analysis.

Details

The function extracts the PCA loadings (rotation matrix) and computes the squared values of each loading, scaled to percentage values. This helps interpret the importance of each wavenumber in defining the principal components.

The function extracts the PCA loadings (rotation matrix) and computes the squared values of each loading, scaled to percentage values. This helps interpret the importance of each wavenumber in defining the principal components.

Value

A tibble containing the wavenumber column and the percentage contribution of each wavenumber to each principal component.

A tibble containing the wavenumber column and the percentage contribution of each wavenumber to each principal component.

Examples

```
pca_result <- spec_pca(CoHAspec)
wn_contrib <- spec_pca_wn_contrib(pca_result)
print(wn_contrib)



pca_result <- spec_pca(CoHAspec)
wn_contrib <- spec_pca_wn_contrib(pca_result)
```

```
print(wn_contrib)
```

---

spec_read *Read Spectral Data from Various File Formats*

Description

This function reads spectral data from a file, automatically detecting the format and using either ‘readr‘ or ‘readxl‘ functions. It also sets the specified wavenumber column as the default using ‘set_spec_wn()‘.

Usage

```
spec_read(file, wn_col, ...)
```

Arguments

file	Path to the file to be read.
wn_col	Character. Name of the column containing wavenumber values. This column will be set as the default wavenumber column.
...	Additional arguments passed to ‘readr::read_delim()‘, ‘readr::read_csv()‘, or ‘readxl::read_excel()‘, depending on the file format.

Details

The function automatically determines the file format based on the extension: - ‘.txt‘, ‘.csv‘, ‘.tsv‘, and ‘.csv2‘ are read using ‘readr‘ functions. - ‘.xls‘ and ‘.xlsx‘ are read using ‘readxl::read_excel()‘.

If the specified ‘wn_col‘ does not exist in the data, an error is returned.

Value

A tibble containing the spectral data.

Examples

```
file_path <- system.file("extdata", "CoHAspec.csv", package = "tidyspec")
df <- spec_read(file_path, wn_col = "Wavenumber")
check_wn_col() # Verifica se a coluna de wavenumber está definida
```

spec_select *Select Specific Columns in a Spectral Data Frame*

Description

This function selects user-specified columns from a spectral dataset, always ensuring that the wavenumber column (‘wn_col‘) is included, unless explicitly excluded.

Usage

```
spec_select(.data, ...)
```

Arguments

.data	A data frame containing spectral data.
...	Column selection helpers (e.g., column names, -column_to_exclude).

Value

A data frame containing the selected columns.

See Also

[dplyr::select()], [set_spec_wn()]

spec_smartplot *Create a Custom Plot for Spectral Data*

Description

This function generates a customizable plot for spectral data, allowing for the selection of plot type (absorbance or transmittance), x-axis direction, plot geometry (points or lines), and color palette.

Usage

```
spec_smartplot(
  .data,
  wn_col = NULL,
  xdir = c("reverse", "standard"),
  geom = c("point", "line"),
  xmin = NULL,
  xmax = NULL,
  alpha = 0.8,
  type = c("absorbance", "transmittance"),
 palette = c("viridis", "plasma", "magma", "cividis", "turbo", "Set1", "Set2", "Dark2",
    "Paired", "custom"),
  custom_colors = NULL
)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength or wavenumber data. This parameter is required.
xdir	A character string specifying the direction of the x-axis. Choices are ‘"reverse"‘ (typically used for wavenumber) or ‘"standard"‘.
geom	A character string specifying the geometry of the plot. Choices are ‘"point"‘ for a scatter plot or ‘"line"‘ for a line plot.
xmin	A numeric value specifying the minimum x-axis value. If not provided, the minimum value from ‘wn_col‘ will be used.
xmax	A numeric value specifying the maximum x-axis value. If not provided, the maximum value from ‘wn_col‘ will be used.
alpha	A numeric value (0–1) specifying the transparency of plotted points or lines. Default is ‘0.8‘.
type	A character string specifying the y-axis label. Either ‘"absorbance"‘ (default) or ‘"transmittance"‘.
palette	A character string specifying the color palette to use. Built-in options: **‘"viridis"‘** Perceptually uniform, colorblind-friendly (default). **‘"plasma"‘** High-contrast warm-to-cool gradient. **‘"magma"‘** Dark-to-light, suitable for dark backgrounds. **‘"cividis"‘** Optimized for color vision deficiency. **‘"turbo"‘** Full-spectrum rainbow with high contrast. **‘"Set1"‘** Qualitative palette from RColorBrewer (up to 9 colors). **‘"Set2"‘** Softer qualitative palette (up to 8 colors). **‘"Dark2"‘** Darker qualitative palette (up to 8 colors). **‘"Paired"‘** Paired qualitative palette (up to 12 colors). **‘"custom"‘** Use a custom vector of colors via ‘custom_colors‘.
custom_colors	A character vector of valid color strings (hex codes or R color names) used when ‘palette = "custom"‘. The number of colors should match or exceed the number of spectra in the data. If fewer colors are provided than spectra, colors will be recycled with a warning.

Value

A ‘ggplot‘ object representing the spectral plot.

Examples

```
## Not run:
# Default viridis palette
spec_smartplot(spec_data, wn_col = "wavenumber")

# Built-in qualitative palette
spec_smartplot(spec_data, wn_col = "wavenumber", palette = "Set1")
```

```
# Custom colors
spec_smartplot(spec_data, wn_col = "wavenumber",
               palette = "custom",
               custom_colors = c("#E63946", "#457B9D", "#2A9D8F"))

## End(Not run)
```

`spec_smartplotly` *Create an Interactive Plot for Spectral Data (plotly)*

Description

This function generates a customizable interactive plot for spectral data using plotly, allowing for the selection of plot type (absorbance or transmittance), x-axis direction, plot geometry (points or lines), and color palette.

Usage

```
spec_smartplotly(
  .data,
  wn_col = NULL,
  xdir = c("reverse", "standard"),
  geom = c("point", "line"),
  xmin = NULL,
  xmax = NULL,
  alpha = 0.8,
  type = c("absorbance", "transmittance"),
 palette = c("viridis", "plasma", "magma", "cividis", "turbo", "Set1", "Set2", "Dark2",
    "Paired", "custom"),
  custom_colors = NULL
)
```

Arguments

`.data`	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
`wn_col`	A character string specifying the column name for the wavelength or wavenumber data. This parameter is required.
`xdir`	A character string specifying the direction of the x-axis. Choices are ‘"reverse"‘ (typically used for wavenumber) or ‘"standard"‘.
`geom`	A character string specifying the geometry of the plot. Choices are ‘"point"‘ for a scatter plot or ‘"line"‘ for a line plot.
`xmin`	A numeric value specifying the minimum x-axis value. If not provided, the minimum value from ‘wn_col‘ will be used.
`xmax`	A numeric value specifying the maximum x-axis value. If not provided, the maximum value from ‘wn_col‘ will be used.

`alpha` A numeric value (0–1) specifying the transparency of plotted points or lines. Default is ‘0.8‘.

`type` A character string specifying the y-axis label. Either ‘"absorbance"‘ (default) or ‘"transmittance"‘.

`palette` A character string specifying the color palette to use. Built-in options:

‘"viridis"‘ Perceptually uniform, colorblind-friendly (default).

‘"plasma"‘ High-contrast warm-to-cool gradient.

‘"magma"‘ Dark-to-light, suitable for dark backgrounds.

‘"cividis"‘ Optimized for color vision deficiency.

‘"turbo"‘ Full-spectrum rainbow with high contrast.

‘"Set1"‘ Qualitative palette from RColorBrewer (up to 9 colors).

‘"Set2"‘ Softer qualitative palette (up to 8 colors).

‘"Dark2"‘ Darker qualitative palette (up to 8 colors).

‘"Paired"‘ Paired qualitative palette (up to 12 colors).

‘"custom"‘ Use a custom vector of colors via ‘custom_colors‘.

`custom_colors` A character vector of valid color strings (hex codes or R color names) used when ‘palette = "custom"‘. The number of colors should match or exceed the number of spectra in the data. If fewer colors are provided than spectra, colors will be recycled with a warning.

Value

A ‘plotly‘ object representing the interactive spectral plot.

Examples

```
## Not run:
# Default viridis palette, line geometry
spec_smartplotly(spec_data, wn_col = "wavenumber", geom = "line")

# Built-in qualitative palette
spec_smartplotly(spec_data, wn_col = "wavenumber", palette = "Set1")

# Custom colors
spec_smartplotly(spec_data, wn_col = "wavenumber",
                     palette = "custom",
                     custom_colors = c("#E63946", "#457B9D", "#2A9D8F"))

## End(Not run)
```

spec_smooth_avg *Apply Smoothing to Spectral Data Using a Moving Average*

Description

This function applies a moving average smoothing to numeric spectral data using a specified window size and polynomial degree, while preserving the wavelength column.

Usage

```
spec_smooth_avg(.data, wn_col = NULL, window = 15, degree = 2)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
window	A numeric value specifying the window size for the moving average smoothing. Default is 15.
degree	A numeric value specifying the degree of the polynomial for smoothing. Default is 2.

Value

A ‘tibble‘ with the smoothed spectral data, containing the wavelength column and the smoothed numeric columns.

spec_smooth_sga *Apply Savitzky-Golay Smoothing to Spectral Data*

Description

This function applies Savitzky-Golay smoothing to numeric spectral data using a specified window size, polynomial order, and differentiation degree, while preserving the wavelength column.

Usage

```
spec_smooth_sga(.data, wn_col = NULL, window = 15, forder = 4, degree = 0)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.
window	A numeric value specifying the window size for the Savitzky-Golay smoothing. Default is 15.
forder	A numeric value specifying the polynomial order for smoothing. Default is 4.
degree	A numeric value specifying the degree of differentiation. Default is 0 (no differentiation).

Value

A ‘tibble‘ with the smoothed spectral data, containing the wavelength column and the smoothed numeric columns.

---

spec_trans2abs *Convert Spectral Data from Transmittance to Absorbance*

Description

This function converts transmittance data to absorbance using the formula ‘A = 2 - log10(T)‘, where ‘T‘ is the transmittance. It also filters out any infinite values resulting from the transformation, while preserving the wavelength column.

Usage

```
spec_trans2abs(.data, wn_col = NULL)
```

Arguments

.data	A ‘data.frame‘ or ‘tibble‘ containing spectral data.
wn_col	A character string specifying the column name for the wavelength data. Default is ‘"Wn"‘.

Value

A ‘tibble‘ with the converted absorbance data, containing the wavelength column and the absorbance numeric columns.

spec_values_at *Extract spectral values at specific wavenumbers*

Description

'spec_values_at()' extracts spectral values at specified wavenumber positions. The function finds the closest wavenumber matches and returns the corresponding spectral intensities for each sample column.

Usage

```
spec_values_at(
  spec_data,
  wn_values,
  wn_col = NULL,
  method = "closest",
  tolerance = Inf,
  format = "wide"
)
```

Arguments

spec_data	A tibble containing spectral data
wn_values	Numeric vector of wavenumber values to extract
wn_col	Character string specifying the wavenumber column name. If NULL, uses the globally set wavenumber column from 'set_spec_wn()'.
method	Character string specifying the extraction method. Options: "closest" (default), "exact", "interpolate". See Details.
tolerance	Numeric value specifying the maximum allowed difference for "closest" method. Default: Inf (no limit).
format	Character string specifying output format. Options: "wide" (default), "long". See Details.

Details

Extraction methods: - "closest": Find the closest wavenumber match (default) - "exact": Only return exact matches (NA for non-matches) - "interpolate": Use linear interpolation between adjacent points

Output formats: - "wide": Each wavenumber becomes a row, samples as columns - "long": Tidy format with wavenumber, sample, and value columns

Value

A tibble with extracted values in the specified format

Examples

```
## Not run:
# Set wavenumber column
set_spec_wn("Wavenumber")

# Extract values at specific wavenumbers
peaks <- c(1650, 1450, 1250, 1050)
extracted <- spec_values_at(CoHAspec, peaks)

# Use exact matching only
extracted <- spec_values_at(CoHAspec, peaks, method = "exact")

# Get results in long format
extracted <- spec_values_at(CoHAspec, peaks, format = "long")

# Use interpolation
extracted <- spec_values_at(CoHAspec, peaks, method = "interpolate")

## End(Not run)
```

Index

∗ **datasets**
    CoHAspec, 3

check_wn_col, 3
CoHAspec, 3

demo_rolling_ball, 4

plot_rolling_ball, 5

rolling_ball, 5
rolling_ball_morphology, 6

set_spec_wn, 7
smooth_baseline, 7
spec_abs2trans, 8
spec_bl_rollingBall, 8
spec_bl_rubberband, 9
spec_blc_rollingBall, 10
spec_blc_rubberband, 11
spec_diff, 12
spec_filter, 13
spec_interpolate_left, 13
spec_interpolate_regular, 14
spec_interpolate_right, 15
spec_join, 16
spec_norm_01, 17
spec_norm_area, 17
spec_norm_by_wn, 18
spec_norm_minmax, 19
spec_norm_var, 19
spec_pca, 20
spec_pca_screeplot, 21
spec_pca_wn_contrib, 22
spec_read, 23
spec_select, 24
spec_smartplot, 24
spec_smartplotly, 26
spec_smooth_avg, 28
spec_smooth_sga, 28
spec_trans2abs, 29
spec_values_at, 30