Package: PhenoSpectra (via r-universe)

March 6, 2025

Title Multispectral Data Analysis and Visualization

Version 0.1.0

Description Provides tools for processing, analyzing, and visualizing spectral data collected from 3D laser-based scanning systems. Supports applications in agriculture, forestry, environmental monitoring, industrial quality control, and biomedical research. Enables evaluation of plant growth, productivity, resource efficiency, disease management, and pest monitoring. Includes statistical methods for extracting insights from multispectral and hyperspectral data and generating publication-ready visualizations. See Zieschank & Junker (2023) <doi:10.3389/fpls.2023.1141554> and Saric et al. (2022) <doi:10.1016/J.TPLANTS.2021.12.003> for related work.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports readxl, writexl, dplyr, tidyr, data.table, lubridate, openxlsx, broom, magrittr, rlang, utils, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Medhat Mahmoud [aut, cre], Arianna Del Papa [ctb], Melina Rampelmann [ctb], Patrick Dohle [ctb]

Maintainer Medhat Mahmoud <medhat.mahmoud@bayer.com>

Date/Publication 2025-03-05 13:00:09 UTC

Config/pak/sysreqs libicu-dev zlib1g-dev

Repository https://medhat-mahmoud.r-universe.dev

RemoteUrl https://github.com/cran/PhenoSpectra

RemoteRef HEAD

RemoteSha ed47850803dfb5d12af1d39bb087c99839f8bcd5

Contents

`feature_selection` *Feature Selection for Spectral Data*

Description

This function filters healthy vs diseased samples, selects the most discriminative spectral variables, applies FDR correction, and exports the results.

Usage

```
feature_selection(
  file_path,
  output_path = "selected_features.xlsx",
  fdr_threshold = 0.01
)
```

Arguments

`file_path`	Path to the cleaned dataset (output of qaqcs function).
`output_path`	Path to save the selected features table.
`fdr_threshold`	Threshold for filtering significant features (default: 0.01).

Value

A data.table containing selected spectral variables.

Examples

```
# Create mock spectral data
library(openxlsx)
mock_data <- data.frame(
  treatment = sample(0:1, 100, replace = TRUE),
  var1 = rnorm(100),
  var2 = rnorm(100),
  var3 = rnorm(100)
)
temp_file <- tempfile(fileext = ".xlsx")
write.xlsx(mock_data, temp_file)
```

```
# Perform feature selection
output_path <- tempfile(fileext = ".xlsx")
selected_features <- feature_selection(temp_file, output_path, fdr_threshold = 0.01)
head(selected_features)
```

`predict_SDS` *Predict Spectral Disease Severity (SDS)*

Description

This function predicts Spectral Disease Severity (SDS) using a standard linear regression model (`lm()`). It automatically handles column names with special characters by using backticks and constrains predictions to the range `[0, 100]`.

Usage

```
predict_SDS(cleaned_data, sf_test, fixed_effects = NULL)
```

Arguments

`cleaned_data`	A dataframe containing spectral measurements and treatment labels.
`sf_test`	A dataframe containing selected important features (from statistical tests).
`fixed_effects`	A character vector of fixed effects to include (default: NULL). Example: c("Scan.date").

Value

A dataframe with predicted SDS values for all treatments, constrained between `0` and `100`.

Examples

```
# Create mock spectral data
library(openxlsx)
cleaned_data <- data.frame(
  treatment = sample(0:1, 100, replace = TRUE),
  var1 = rnorm(100),
  var2 = rnorm(100),
  var3 = rnorm(100),
  Scan.date = sample(
    seq.Date(
      from = as.Date('2023-01-01'),
      to = as.Date('2023-12-31'),
      by = 'day'
    ),
    100
  ),
  Scan.time = format(Sys.time(), "%H:%M:%S")
)
```

qaqcs *Perform QA/QC on spectral data while preserving original column names*

Description

Perform QA/QC on spectral data while preserving original column names

Usage

```
qaqcs(
  file_path,
  output_path,
  handle_missing = "NA",
  handle_outliers = "NA",
  group_by_col = "treatment"
)
```

Arguments

file_path	Path to the input file
output_path	Path to save the cleaned data
handle_missing	Method to handle missing values ('impute', 'remove', or 'NA')
handle_outliers	Method to handle outliers ('impute', 'remove', or 'NA')
group_by_col	Column name for grouping

Value

A list with cleaned data and a summary table

Examples

```
library(openxlsx)
# Create mock raw data
raw_data <- data.frame(
  treatment = sample(0:1, 100, replace = TRUE),
  var1 = rnorm(100),
  var2 = rnorm(100),
  var3 = rnorm(100)
)

# Save mock data to a temporary file
raw_data_file <- tempfile(fileext = ".xlsx")
output_file <- tempfile(fileext = ".xlsx")

write.xlsx(raw_data, raw_data_file)
```

```
# Run QA/QC with missing values imputed and outliers removed
cleaned_result <- qaqcs(
  file_path = raw_data_file,
  output_path = output_file,
  handle_missing = "impute",
  handle_outliers = "remove",
  group_by_col = "treatment"
)
head(cleaned_result$cleaned_data)
```

`reads` *Read and merge spectral data using data.table exclusively*

Description

Read and merge spectral data using data.table exclusively

Usage

```
reads(directory, pattern, output_path)
```

Arguments

`directory`	Path to the directory containing files
`pattern`	File pattern to search for (e.g., 'input')
`output_path`	Path to save the processed output

Value

A merged data.table

Examples

```
library(data.table)
# Create mock data files with all required columns
mock_data1 <- data.frame(
  treatment = sample(0:1, 50, replace = TRUE),
  var1 = rnorm(50),
  var2 = rnorm(50),
  Scan.date = sample(
    seq.Date(
      from = as.Date('2023-01-01'),
      to = as.Date('2023-12-31'),
      by = 'day'
    ),
    50,
    replace = TRUE
  ),
  Scan.time = format(Sys.time(), "%H:%M:%S"),
```

```
  timestamp = Sys.time()  # Add timestamp column
)

mock_data2 <- data.frame(
  treatment = sample(0:1, 50, replace = TRUE),
  var1 = rnorm(50),
  var2 = rnorm(50),
  Scan.date = sample(
    seq.Date(
      from = as.Date('2023-01-01'),
      to = as.Date('2023-12-31'),
      by = 'day'
    ),
    50,
    replace = TRUE
  ),
  Scan.time = format(Sys.time(), "%H:%M:%S"),
  timestamp = Sys.time()  # Add timestamp column
)

# Save mock data to temporary CSV files
temp_dir <- tempdir()
file1 <- file.path(temp_dir, "input_file1.csv")
file2 <- file.path(temp_dir, "input_file2.csv")
fwrite(mock_data1, file1)
fwrite(mock_data2, file2)

# Run the reads() function on mock CSV data
merged_data <- reads(
  directory = temp_dir,
  pattern = "input",
  output_path = tempfile(fileext = ".csv")
)
head(merged_data)
```

`setup_environment` *Setup Environment for PhenoSpectra*

Description

Automatically installs and loads all required packages.

This function installs missing packages required by the PhenoSpectra package.

Usage

```
setup_environment()
```

```
setup_environment()
```

Value

NULL. This function installs required packages if missing and loads them.

Examples

```
# Example usage:
setup_environment()  # Automatically installs and loads required packages
# Run to manually install missing packages
setup_environment()
```

Index