# Package 'waves'

June 2, 2026

**Title** Vis-NIR Spectral Analysis Wrapper

**Version** 0.2.7

**Maintainer** Jenna Hershberger <jmh579@cornell.edu>

**Description** Originally designed application in the context of resource-limited plant research and breeding programs, 'waves' provides an open-source solution to spectral data processing and model development by bringing useful packages together into a streamlined pipeline. This package is wrapper for functions related to the analysis of point visible and near-infrared reflectance measurements. It includes visualization, filtering, aggregation, preprocessing, cross-validation set formation, model training, and prediction functions to enable open-source association of spectral and reference data. This package is documented in a peer-reviewed manuscript in the Plant Phenome Journal <doi:10.1002/ppj2.20012>. Specialized cross-validation schemes are described in detail in Jarquín et al. (2017) <doi:10.3835/plantgenome2016.12.0130>. Example data is from Ikeogu et al. (2017) <doi:10.1371/journal.pone.0188918>.

**License** MIT + file LICENSE

**URL** https://GoreLab.github.io/waves/, https://github.com/GoreLab/waves

**BugReports** https://github.com/GoreLab/waves/issues

**Depends** R (>= 4.1.0)

**Imports** caret, dplyr, ggplot2, lifecycle, magrittr, pls, prospectr, randomForest, readr, rlang, scales, spectacles, stringr, tibble, tidyr (>= 1.0), tidyselect

**Suggests** testthat (>= 2.1.0), knitr, rmarkdown

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr, rmarkdown

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Jenna Hershberger [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3147-6867>),
Michael Gore [ths],
NSF BREAD IOS-1543958 [fnd]

**Repository** CRAN

**Date/Publication** 2026-06-02 18:10:02 UTC

# Contents

aggregate_spectra . . . . . 2
filter_spectra . . . . . 3
format_cv . . . . . 5
ikeogu.2017 . . . . . 7
plot_spectra . . . . . 8
predict_spectra . . . . . 10
pretreat_spectra . . . . . 11
save_model . . . . . 12
test_spectra . . . . . 16
train_spectra . . . . . 20

**Index** **24**

---

`aggregate_spectra` *Aggregate data based on grouping variables and a user-provided function*

---

## Description

Use grouping variables to collapse spectral `data.frame` by mean or median. Recommended for use after `filter_spectra`

## Usage

```
aggregate_spectra(df, grouping.colnames, reference.value.colname,
  agg.function)
```

## Arguments

| | |
|---|---|
| `df` | `data.frame` object containing one or multiple columns of grouping variables (must be consistent within each group), column of reference values (optional), and columns of spectra. Spectral column names must start with "X". |
| `grouping.colnames` | Names of columns to be used as grouping variables. Minimum 2 variables required. Default is c("trial", "plot"). |
| `reference.value.colname` | Name of reference column to be aggregated along with spectra. Default is "reference" |

agg.function — Name of function (string format) to be used for sample aggregation. Must be either "mean" or "median". Default is "mean".

## Value

data.frame object df aggregated based on grouping column by agg.function

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## Examples

```
library(magrittr)
aggregated.test <- ikeogu.2017 %>%
  dplyr::select(-TCC) %>%
  na.omit() %>%
  aggregate_spectra(
    grouping.colnames = c("study.name"),
    reference.value.colname = "DMC.oven",
    agg.function = "mean"
  )
aggregated.test[1:5, 1:5]
```

---

# filter_spectra *Filter spectral data frame based on Mahalanobis distance*

---

## Description

Determine Mahalanobis distances of observations (rows) within a given data.frame with spectral data. Option to filter out observations based on these distances.

## Usage

```
filter_spectra(df, filter, return.distances, num.col.before.spectra,
  window.size, verbose)
```

## Arguments

df — data.frame object containing columns of spectra and rows of observations. Spectral columns must be labeled with an "X" and then the wavelength (example: "X740" = 740nm). Left-most column must be unique ID. May also contain columns of metadata between the unique ID and spectral columns. Cannot contain any missing values. Metadata column names may not start with "X".

filter — boolean that determines whether or not the input data.frame will be filtered. If TRUE, df will be filtered according to squared Mahalanobis distance with a 95% cutoff from a chi-square distribution with degrees of freedom = number of spectral columns. If FALSE, a column of squared Mahalanobis distances h.distance will be added to the right side of df and all rows will be returned. Default is TRUE.

`return.distances`
: boolean that determines whether a column of squared Mahalanobis distances will be included in output `data.frame`. If `TRUE`, a column of Mahalanobis distances for each row will be added to the right side of `df`. Default is `FALSE`.

`num.col.before.spectra`
: number of columns to the left of the spectral matrix in `df`. Default is 4.

`window.size`
: number defining the size of window to use when calculating the covariance of the spectra (required to calculate Mahalanobis distance). Default is 10.

`verbose`
: If `TRUE`, the number of rows removed through filtering will be printed to the console. Default is `TRUE`.

## Details

This function uses a chi-square distribution with 95% cutoff where degrees of freedom = number of wavelengths (columns) in the input `data.frame`.

## Value

If `filter` is `TRUE`, returns filtered data frame `df` and reports the number of rows removed. The Mahalanobis distance with a cutoff of 95% of chi-square distribution (degrees of freedom = number of wavelengths) is used as filtering criteria. If `filter` is `FALSE`, returns full input df with column `h.distances` containing the Mahalanobis distance for each row.

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## References

Johnson, R.A., and D.W. Wichern. 2007. Applied Multivariate Statistical Analysis (6th Edition). pg 189

## Examples

```
library(magrittr)
filtered.test <- ikeogu.2017 %>%
  dplyr::select(-TCC) %>%
  na.omit() %>%
  filter_spectra(
    df = .,
    filter = TRUE,
    return.distances = TRUE,
    num.col.before.spectra = 5,
    window.size = 15
  )
filtered.test[1:5, c(1:5, (ncol(filtered.test) - 5):ncol(filtered.test))]
```

format_cv *Format multiple trials with or without overlapping genotypes into training and test sets according to user-provided cross validation scheme*

## Description

Standalone function that is also used within `train_spectra` to divide trials or studies into training and test sets based on overlap in trial environments and genotype entries

## Usage

```
format_cv(
  trial1,
  trial2,
  trial3 = NULL,
  cv.scheme,
  stratified.sampling = TRUE,
  proportion.train = 0.7,
  seed = NULL,
  remove.genotype = FALSE
)
```

## Arguments

| | |
|---|---|
| `trial1` | `data.frame` object that is for use only when `cv.scheme` is provided. Contains the trial to be tested in subsequent model training functions. The first column contains unique identifiers, second contains genotypes, third contains reference values, followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X", reference column must be named "reference", and genotype column must be named "genotype". |
| `trial2` | `data.frame` object that is for use only when `cv.scheme` is provided. This data.frame contains a trial that has overlapping genotypes with `trial1` but that were grown in a different site/year (different environment). Formatting must be consistent with `trial1`. |
| `trial3` | `data.frame` object that is for use only when `cv.scheme` is provided. This data.frame contains a trial that may or may not contain genotypes that overlap with `trial1`. Formatting must be consistent with `trial1`. |
| `cv.scheme` | A cross validation (CV) scheme from Jarquín et al., 2017. Options for `cv.scheme` include:<br>• "CV1": untested lines in tested environments<br>• "CV2": tested lines in tested environments<br>• "CV0": tested lines in untested environments<br>• "CV00": untested lines in untested environments |

`stratified.sampling`
: If `TRUE`, training and test sets will be selected using stratified random sampling. Default is `TRUE`.

`proportion.train`
: Fraction of samples to include in the training set. Default is 0.7.

`seed`
: Number used in the function `set.seed()` for reproducible randomization. If `NULL`, no seed is set. Default is `NULL`.

`remove.genotype`
: boolean that, if `TRUE`, removes the "genotype" column is removed from the output `data.frame`. Default is `FALSE`.

## Details

Use of a cross-validation scheme requires a column in the input `data.frame` named "genotype" to ensure proper sorting of training and test sets. Variables `trial1` and `trial2` are required, while `trial 3` is optional.

## Value

List of data.frames ($train.set, $test.set) compiled according to user-provided cross validation scheme.

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## References

Jarquín, D., C. Lemes da Silva, R. C. Gaynor, J. Poland, A. Fritz, R. Howard, S. Battenfield, and J. Crossa. 2017. Increasing genomic-enabled prediction accuracy by modeling genotype × environment interactions in Kansas wheat. Plant Genome 10(2):1-15. <doi:10.3835/plantgenome2016.12.0130>

## Examples

```
# Must have a column called "genotype", so we'll create a fake one for now
# We will use CV00, which does not require any overlap in genotypes
# In real scenarios, CV schemes that rely on genotypes should not be applied
# when genotypes are unknown, as in this case.
library(magrittr)
trials <- ikeogu.2017 %>%
  dplyr::mutate(genotype = 1:nrow(ikeogu.2017)) %>% # fake for this example
  dplyr::rename(reference = DMC.oven) %>%
  dplyr::select(
    study.name, sample.id, genotype, reference,
    tidyselect::starts_with("X")
  )
trial1 <- trials %>%
  dplyr::filter(study.name == "C16Mcal") %>%
  dplyr::select(-study.name)
trial2 <- trials %>%
  dplyr::filter(study.name == "C16Mval") %>%
  dplyr::select(-study.name)
```

```
cv.list <- format_cv(
  trial1 = trial1, trial2 = trial2, cv.scheme = "CV00",
  stratified.sampling = FALSE, remove.genotype = TRUE
)
cv.list$train.set[1:5, 1:5]
cv.list$test.set[1:5, 1:5]
```

---

`ikeogu.2017` *Example vis-NIRS and reference dataset*

---

## Description

The 'ikeogu.2017' data set contains raw vis-NIRS scans, total carotenoid content, and cassava root dry matter content (using the oven method) from the 2017 PLOS One paper by Ikeogu et al. This dataset contains a subset of the original scans and reference values from the supplementary files of the paper. 'ikeogu.2017' is a 'data.frame' that contains the following columns:

- study.name = Name of the study as described in Ikeogu et al. (2017).
- sample.id = Unique identifier for each individual root sample
- DMC.oven = Cassava root dry matter content, the percentage of dry weight relative to fresh weight of a sample after oven drying.
- TCC = Total carotenoid content ($\mu g/g$, unknown whether on a fresh or dry weight basis) as measured by high performance liquid chromatography
- X350:X2500 = spectral reflectance measured with the QualitySpec Trek: S-10016 vis-NIR spectrometer. Each cell represents the mean of 150 scans on a single root at a single wavelength.

## Usage

```
ikeogu.2017
```

## Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 175 rows and 2155 columns.

## Author(s)

Original authors: Ikeogu, U.N., F. Davrieux, D. Dufour, H. Ceballos, C.N. Egesi, and J. Jannink. Reformatted by Jenna Hershberger.

## References

Ikeogu, U.N., F. Davrieux, D. Dufour, H. Ceballos, C.N. Egesi, et al. 2017. Rapid analyses of dry matter content and carotenoids in fresh cassava roots using a portable visible and near infrared spectrometer (Vis/NIRS). PLOS One 12(12): 1–17. doi: 10.1371/journal.pone.0188918.

## Examples

```
library(magrittr)
library(ggplot2)
data(ikeogu.2017)
ikeogu.2017[1:10, 1:10]
ikeogu.2017 %>%
  dplyr::select(-starts_with("X")) %>%
  dplyr::group_by(study.name) %>%
  tidyr::pivot_longer(cols = c(DMC.oven:TCC), names_to = "trait", values_to = "value") %>%
  tidyr::drop_na(value) %>%
  ggplot2::ggplot(aes(x = study.name, y = value, fill = study.name)) +
  facet_wrap(~trait, scales = "free_y", nrow = 2) +
  geom_boxplot()
```

---

`plot_spectra` *Plot spectral data, highlighting outliers as identified using Mahalanobis distance*

---

## Description

Generates a `ggplot` object of given spectra, with wavelength on the x axis and given spectral values on the y. Mahalanobis distance is used to calculate outliers, which are both identified on the plot. Rows from the original dataframe are printed to the console for each outlier that is identified.

## Usage

```
plot_spectra(
  df,
  num.col.before.spectra = 1,
  window.size = 10,
  detect.outliers = TRUE,
  color = NULL,
  alternate.title = "",
  verbose = TRUE,
  wavelengths = lifecycle::deprecated()
)
```

## Arguments

| | |
|---|---|
| `df` | `data.frame` object containing columns of spectra. Spectral columns must be labeled with an "X" and then the wavelength (example: "X740" = 740nm). Left-most column must be unique ID. May also contain columns of metadata between the unique ID and spectral columns. Cannot contain any missing values. Metadata column names may not start with "X". |
| `num.col.before.spectra` | Number of columns to the left of the spectral matrix (including unique ID). Default is 1. |

`window.size` number defining the size of window to use when calculating the covariance of the spectra (required to calculate Mahalanobis distance). Default is 10.

`detect.outliers` Boolean indicating whether spectra should be filtered before plotting. If `TRUE`, outliers are indicated by color in the resulting plot. If `verbose` is also set to `TRUE`, outlier metadata will be printed to the console. Default is `TRUE`.

`color` String or vector of strings indicating colors to be passed to `ggplot`. Default is default `ggplot` colors.

`alternate.title` String to be used as plot title. If `detect.outliers` is `TRUE`, a descriptive title will be supplied. If `detect.outliers` is `FALSE`, default is no title will be used.

`verbose` If `TRUE`, the number of rows removed through filtering will be printed to the console. Default is `TRUE`.

`wavelengths` DEPRECATED `wavelengths` is no longer supported; this information is now inferred from `df` column names

## Value

If verbose, prints unique ID and metadata for rows identified as outliers. Returns plot of spectral data with non-outliers in blue and outliers in red. X-axis is wavelengths and y-axis is spectral values.

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## Examples

```
library(magrittr)
ikeogu.2017 %>%
  dplyr::rename(unique.id = sample.id) %>%
  dplyr::select(unique.id, dplyr::everything(), -TCC) %>%
  na.omit() %>%
  plot_spectra(
    df = .,
    num.col.before.spectra = 5,
    window.size = 15,
    detect.outliers = TRUE,
    color = NULL,
    alternate.title = NULL,
    verbose = TRUE
  )
```

predict_spectra *Use provided model object to predict trait values with input dataset*

## Description

Loads an existing model and cross-validation performance statistics (created with save_model) and makes predictions based on new spectra.

## Usage

```
predict_spectra(
  input.data,
  model.stats.location,
  model.location,
  model.method = "pls",
  wavelengths = lifecycle::deprecated()
)
```

## Arguments

| | |
|---|---|
| input.data | data.frame object of spectral data for input into a spectral prediction model. First column contains unique identifiers followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X". |
| model.stats.location | String containing file path (including file name) to save location of "(model.name)_stats.csv" as output from the save_model function. |
| model.location | String containing file path (including file name) to location where the trained model ("(model.name).Rds") was saved as output by the save_model function. |
| model.method | Model type to use for training. Valid options include:<br>• "pls": Partial least squares regression (Default)<br>• "rf": Random forest<br>• "svmLinear": Support vector machine with linear kernel<br>• "svmRadial": Support vector machine with radial kernel |
| wavelengths | DEPRECATED wavelengths is no longer supported; this information is now inferred from input.data column names |

## Value

data.frame object of predictions for each sample (row). First column is unique identifier supplied by input.data and second is predicted values

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## Examples

```
## Not run:
ikeogu.2017 %>%
  dplyr::select(sample.id, dplyr::starts_with("X")) %>%
  predict_spectra(
    input.data = .,
    model.stats.location = paste0(
      getwd(),
      "/my_model_stats.csv"
    ),
    model.location = paste0(getwd(), "/my_model.Rds")
  )

## End(Not run)
```

---

`pretreat_spectra` *Pretreat spectral data according to user-designated method*

---

## Description

Pretreatment, also known as preprocessing, is often used to increase the signal to noise ratio in vis-NIR datasets. The *waves* function `pretreat_spectra` applies common spectral pretreatment methods such as standard normal variate and the Savitzky-Golay filter.

## Usage

```
pretreat_spectra(
  df,
  test.data = NULL,
  pretreatment = 1,
  preprocessing.method = lifecycle::deprecated(),
  wavelengths = lifecycle::deprecated()
)
```

## Arguments

| | |
|---|---|
| `df` | `data.frame` object containing spectral data. First column(s) (optional) include metadata (with or without reference value column) followed by spectral columns. Spectral column names must be formatted as "X" followed by wavelength Include no other columns to right of spectra! No missing values permitted. |
| `test.data` | `data.frame` object with same format as train.data. Will be appended to `df` during pretreatment so that the same transformations are applied to each row. Default is `NULL`. |
| `pretreatment` | Number or list of numbers 1:13 corresponding to desired pretreatment method(s):<br>1. Raw data (default)<br>2. Standard normal variate (SNV) |

3. SNV and first derivative
4. SNV and second derivative
5. First derivative
6. Second derivative
7. Savitzky–Golay filter (SG)
8. SNV and SG
9. Gap-segment derivative (window size = 11)
10. SG and first derivative (window size = 5)
11. SG and first derivative (window size = 11)
12. SG and second derivative (window size = 5)
13. SG and second derivative (window size = 11)

`preprocessing.method`
: DEPRECATED `preprocessing.method` has been renamed "pretreatment"

`wavelengths`
: DEPRECATED `wavelengths` is no longer supported; this information is now inferred from `df` column names

## Value

Pretreated `df`‘ (or list of `data.frames`) with reference column intact

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## Examples

```
pretreat_spectra(df = ikeogu.2017, pretreatment = 3)[1:5, 1:5]
```

---

# save_model *Save spectral prediction model and model performance statistics*

---

## Description

Given a set of pretreatment methods, saves the best spectral prediction model and model statistics to `model.save.folder` as `model.name.Rds` and `model.name_stats.csv` respectively. If only one pretreatment method is supplied, results from that method are stored.

## Usage

```
save_model(
  df,
  write.model = TRUE,
  pretreatment = 1,
  model.save.folder = NULL,
  model.name = "PredictionModel",
  best.model.metric = "RMSE",
```

```
  k.folds = 5,
  proportion.train = 0.7,
  tune.length = 50,
  model.method = "pls",
  num.iterations = 10,
  stratified.sampling = TRUE,
  cv.scheme = NULL,
  trial1 = NULL,
  trial2 = NULL,
  trial3 = NULL,
  seed = 1,
  verbose = TRUE,
  save.model = lifecycle::deprecated(),
  wavelengths = lifecycle::deprecated(),
  autoselect.preprocessing = lifecycle::deprecated(),
  preprocessing.method = lifecycle::deprecated()
)
```

## Arguments

| | |
|---|---|
| `df` | `data.frame` object. First column contains unique identifiers, second contains reference values, followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X" and reference column must be named "reference" |
| `write.model` | If `TRUE`, the trained model will be saved in .Rds format to the location specified by `model.save.folder`. If `FALSE`, the best model will be output by the function but will not save to a file. Default is `TRUE`. |
| `pretreatment` | Number or list of numbers 1:13 corresponding to desired pretreatment method(s):<br>1. Raw data (default)<br>2. Standard normal variate (SNV)<br>3. SNV and first derivative<br>4. SNV and second derivative<br>5. First derivative<br>6. Second derivative<br>7. Savitzky–Golay filter (SG)<br>8. SNV and SG<br>9. Gap-segment derivative (window size = 11)<br>10. SG and first derivative (window size = 5)<br>11. SG and first derivative (window size = 11)<br>12. SG and second derivative (window size = 5)<br>13. SG and second derivative (window size = 11) |
| `model.save.folder` | Path to folder where model will be saved. If not provided, will save to working directory. |
| `model.name` | Name that model will be saved as in `model.save.folder`. Default is "PredictionModel". |

`best.model.metric`
: Metric used to decide which model is best. Must be either "RMSE" or "Rsquared"

`k.folds`
: Number indicating the number of folds for k-fold cross-validation during model training. Default is 5.

`proportion.train`
: Fraction of samples to include in the training set. Default is 0.7.

`tune.length`
: Number delineating search space for tuning of the PLSR hyperparameter `ncomp`. Must be set to 5 when using the random forest algorithm (`model.method == rf`). Default is 50.

`model.method`
: Model type to use for training. Valid options include:
  - "pls": Partial least squares regression (Default)
  - "rf": Random forest
  - "svmLinear": Support vector machine with linear kernel
  - "svmRadial": Support vector machine with radial kernel

`num.iterations`
: Number of training iterations to perform

`stratified.sampling`
: If `TRUE`, training and test sets will be selected using stratified random sampling. This term is only used if `test.data == NULL`. Default is `TRUE`.

`cv.scheme`
: A cross validation (CV) scheme from Jarquín et al., 2017. Options for `cv.scheme` include:
  - "CV1": untested lines in tested environments
  - "CV2": tested lines in tested environments
  - "CV0": tested lines in untested environments
  - "CV00": untested lines in untested environments

`trial1`
: `data.frame` object that is for use only when `cv.scheme` is provided. Contains the trial to be tested in subsequent model training functions. The first column contains unique identifiers, second contains genotypes, third contains reference values, followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X", reference column must be named "reference", and genotype column must be named "genotype".

`trial2`
: `data.frame` object that is for use only when `cv.scheme` is provided. This data.frame contains a trial that has overlapping genotypes with `trial1` but that were grown in a different site/year (different environment). Formatting must be consistent with `trial1`.

`trial3`
: `data.frame` object that is for use only when `cv.scheme` is provided. This data.frame contains a trial that may or may not contain genotypes that overlap with `trial1`. Formatting must be consistent with `trial1`.

`seed`
: Integer to be used internally as input for `set.seed()`. Only used if `stratified.sampling = TRUE`. In all other cases, seed is set to the current iteration number. Default is 1.

`verbose`
: If `TRUE`, the number of rows removed through filtering will be printed to the console. Default is `TRUE`.

`save.model`
: DEPRECATED `save.model = FALSE` is no longer supported; this function will always return a saved model.

`wavelengths` DEPRECATED `wavelengths` is no longer supported; this information is now inferred from `df` column names

`autoselect.preprocessing` DEPRECATED `autoselect.preprocessing = FALSE` is no longer supported. If multiple pretreatment methods are supplied, the best will be automatically selected as the model to be saved.

`preprocessing.method` DEPRECATED `preprocessing.method` has been renamed "pretreatment"

## Details

Wrapper that uses `pretreat_spectra`, `format_cv`, and `train_spectra` functions.

## Value

List of model stats (in `data.frame`) and trained model object. If the parameter `write.model` is TRUE, both objects are saved to `model.save.folder`. To use the optimally trained model for predictions, use tuned parameters from `$bestTune`.

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## Examples

```
library(magrittr)
test.model <- ikeogu.2017 %>%
  dplyr::filter(study.name == "C16Mcal") %>%
  dplyr::rename(reference = DMC.oven,
                unique.id = sample.id) %>%
  dplyr::select(unique.id, reference, dplyr::starts_with("X")) %>%
  na.omit() %>%
  save_model(
    df = .,
    write.model = FALSE,
    pretreatment = 1:13,
    model.name = "my_prediction_model",
    tune.length = 3,
    num.iterations = 3
  )
summary(test.model$best.model)
test.model$best.model.stats
```

`test_spectra` *Test the performance of spectral models*

## Description

Wrapper that trains models based spectral data to predict reference values and reports model performance statistics

## Usage

```
test_spectra(
  train.data,
  num.iterations,
  test.data = NULL,
  pretreatment = 1,
  k.folds = 5,
  proportion.train = 0.7,
  tune.length = 50,
  model.method = "pls",
  best.model.metric = "RMSE",
  stratified.sampling = TRUE,
  cv.scheme = NULL,
  trial1 = NULL,
  trial2 = NULL,
  trial3 = NULL,
  split.test = FALSE,
  seed = 1,
  verbose = TRUE,
  wavelengths = lifecycle::deprecated(),
  preprocessing = lifecycle::deprecated(),
  output.summary = lifecycle::deprecated(),
  rf.variable.importance = lifecycle::deprecated()
)
```

## Arguments

| | |
|---|---|
| `train.data` | `data.frame` object of spectral data for input into a spectral prediction model. First column contains unique identifiers, second contains reference values, followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X" and reference column must be named "reference". |
| `num.iterations` | Number of training iterations to perform |
| `test.data` | `data.frame` with same specifications as `df`. Use if specific test set is desired for hyperparameter tuning. If `NULL`, function will automatically train with a stratified sample of 70%. Default is `NULL`. |
| `pretreatment` | Number or list of numbers 1:13 corresponding to desired pretreatment method(s): |

1. Raw data (default)
2. Standard normal variate (SNV)
3. SNV and first derivative
4. SNV and second derivative
5. First derivative
6. Second derivative
7. Savitzky–Golay filter (SG)
8. SNV and SG
9. Gap-segment derivative (window size = 11)
10. SG and first derivative (window size = 5)
11. SG and first derivative (window size = 11)
12. SG and second derivative (window size = 5)
13. SG and second derivative (window size = 11)

`k.folds` Number indicating the number of folds for k-fold cross-validation during model training. Default is 5.

`proportion.train`
Fraction of samples to include in the training set. Default is 0.7.

`tune.length` Number delineating search space for tuning of the PLSR hyperparameter `ncomp`. Must be set to 5 when using the random forest algorithm (`model.method == rf`). Default is 50.

`model.method` Model type to use for training. Valid options include:

- "pls": Partial least squares regression (Default)
- "rf": Random forest
- "svmLinear": Support vector machine with linear kernel
- "svmRadial": Support vector machine with radial kernel

`best.model.metric`
Metric used to decide which model is best. Must be either "RMSE" or "Rsquared"

`stratified.sampling`
If `TRUE`, training and test sets will be selected using stratified random sampling. This term is only used if `test.data == NULL`. Default is `TRUE`.

`cv.scheme` A cross validation (CV) scheme from Jarquín et al., 2017. Options for `cv.scheme` include:

- "CV1": untested lines in tested environments
- "CV2": tested lines in tested environments
- "CV0": tested lines in untested environments
- "CV00": untested lines in untested environments

`trial1` `data.frame` object that is for use only when `cv.scheme` is provided. Contains the trial to be tested in subsequent model training functions. The first column contains unique identifiers, second contains genotypes, third contains reference values, followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X", reference column must be named "reference", and genotype column must be named "genotype".

trial2 — data.frame object that is for use only when cv.scheme is provided. This data.frame contains a trial that has overlapping genotypes with trial1 but that were grown in a different site/year (different environment). Formatting must be consistent with trial1.

trial3 — data.frame object that is for use only when cv.scheme is provided. This data.frame contains a trial that may or may not contain genotypes that overlap with trial1. Formatting must be consistent with trial1.

split.test — boolean that allows for a fixed training set and a split test set. Example// train model on data from two breeding programs and a stratified subset (70%) of a third and test on the remaining samples (30%) of the third. If FALSE, the entire provided test set test.data will remain as a testing set or if none is provided, 30% of the provided train.data will be used for testing. Default is FALSE.

seed — Integer to be used internally as input for set.seed(). Only used if stratified.sampling = TRUE. In all other cases, seed is set to the current iteration number. Default is 1.

verbose — If TRUE, the number of rows removed through filtering will be printed to the console. Default is TRUE.

wavelengths — DEPRECATED wavelengths is no longer supported; this information is now inferred from df column names

preprocessing — DEPRECATED please use pretreatment to specify the specific pretreatment(s) to test. For behavior identical to that of preprocessing = TRUE, set pretreatment = 1:13‘.

output.summary — DEPRECATED output.summary = FALSE is no longer supported; a summary of output is always returned alongside the full performance statistics.

rf.variable.importance — DEPRECATED rf.variable.importance = FALSE is no longer supported; variable importance results are always returned if the model.method is set to ‘pls‘ or ‘rf‘.

## Details

Calls pretreat_spectra, format_cv, and train_spectra functions.

## Value

list of 5 objects:

1. ‘model.list‘ is a list of trained model objects, one for each pretreatment method specified by the pretreatment argument. Each model is trained with all rows of df.
2. ‘summary.model.performance‘ is a data.frame containing summary statistics across all model training iterations and pretreatments. See below for a description of the summary statistics provided.
3. ‘model.performance‘ is a data.frame containing performance statistics for each iteration of model training separately (see below).
4. ‘predictions‘ is a data.frame containing both reference and predicted values for each test set entry in each iteration of model training.

5. ‘importance‘ is a `data.frame` containing variable importance results for each wavelength at each iteration of model training. If `model.method` is not "pls" or "rf", this list item is `NULL`.

‘summary.model.performance‘ and ‘model.performance‘ `data.frames` summary statistics include:

- Tuned parameters depending on the model algorithm:
  - **Best.n.comp**, the best number of components
  - **Best.ntree**, the best number of trees in an RF model
  - **Best.mtry**, the best number of variables to include at every decision point in an RF model
- **RMSECV**, the root mean squared error of cross-validation
- **R2cv**, the coefficient of multiple determination of cross-validation for PLSR models
- **RMSEP**, the root mean squared error of prediction
- **R2p**, the squared Pearson's correlation between predicted and observed test set values
- **RPD**, the ratio of standard deviation of observed test set values to RMSEP
- **RPIQ**, the ratio of performance to interquartile difference
- **CCC**, the concordance correlation coefficient
- **Bias**, the average difference between the predicted and observed values
- **SEP**, the standard error of prediction
- **R2sp**, the squared Spearman's rank correlation between predicted and observed test set values

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## Examples

```
library(magrittr)
ikeogu.2017 %>%
  dplyr::rename(reference = DMC.oven,
                unique.id = sample.id) %>%
  dplyr::select(unique.id, reference, dplyr::starts_with("X")) %>%
  na.omit() %>%
  test_spectra(
    train.data = .,
    tune.length = 3,
    num.iterations = 3,
    pretreatment = 1
  )
```

`train_spectra` *Train a model based predict reference values with spectral data*

## Description

Trains spectral prediction models using one of several algorithms and sampling procedures.

## Usage

```
train_spectra(
  df,
  num.iterations,
  test.data = NULL,
  k.folds = 5,
  proportion.train = 0.7,
  tune.length = 50,
  model.method = "pls",
  best.model.metric = "RMSE",
  stratified.sampling = TRUE,
  cv.scheme = NULL,
  trial1 = NULL,
  trial2 = NULL,
  trial3 = NULL,
  split.test = FALSE,
  seed = 1,
  verbose = TRUE,
  save.model = lifecycle::deprecated(),
  rf.variable.importance = lifecycle::deprecated(),
  output.summary = lifecycle::deprecated(),
  return.model = lifecycle::deprecated()
)
```

## Arguments

| | |
|---|---|
| `df` | `data.frame` object. First column contains unique identifiers, second contains reference values, followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X" and reference column must be named "reference" |
| `num.iterations` | Number of training iterations to perform |
| `test.data` | `data.frame` with same specifications as `df`. Use if specific test set is desired for hyperparameter tuning. If `NULL`, function will automatically train with a stratified sample of 70%. Default is `NULL`. |
| `k.folds` | Number indicating the number of folds for k-fold cross-validation during model training. Default is 5. |
| `proportion.train` | Fraction of samples to include in the training set. Default is 0.7. |

`tune.length` Number delineating search space for tuning of the PLSR hyperparameter `ncomp`. Must be set to 5 when using the random forest algorithm (`model.method == rf`). Default is 50.

`model.method` Model type to use for training. Valid options include:

- "pls": Partial least squares regression (Default)
- "rf": Random forest
- "svmLinear": Support vector machine with linear kernel
- "svmRadial": Support vector machine with radial kernel

`best.model.metric` Metric used to decide which model is best. Must be either "RMSE" or "Rsquared"

`stratified.sampling` If `TRUE`, training and test sets will be selected using stratified random sampling. This term is only used if `test.data == NULL`. Default is `TRUE`.

`cv.scheme` A cross validation (CV) scheme from Jarquín et al., 2017. Options for `cv.scheme` include:

- "CV1": untested lines in tested environments
- "CV2": tested lines in tested environments
- "CV0": tested lines in untested environments
- "CV00": untested lines in untested environments

`trial1` `data.frame` object that is for use only when `cv.scheme` is provided. Contains the trial to be tested in subsequent model training functions. The first column contains unique identifiers, second contains genotypes, third contains reference values, followed by spectral columns. Include no other columns to right of spectra! Column names of spectra must start with "X", reference column must be named "reference", and genotype column must be named "genotype".

`trial2` `data.frame` object that is for use only when `cv.scheme` is provided. This data.frame contains a trial that has overlapping genotypes with `trial1` but that were grown in a different site/year (different environment). Formatting must be consistent with `trial1`.

`trial3` `data.frame` object that is for use only when `cv.scheme` is provided. This data.frame contains a trial that may or may not contain genotypes that overlap with `trial1`. Formatting must be consistent with `trial1`.

`split.test` boolean that allows for a fixed training set and a split test set. Example// train model on data from two breeding programs and a stratified subset (70%) of a third and test on the remaining samples (30%) of the third. If `FALSE`, the entire provided test set `test.data` will remain as a testing set or if none is provided, 30% of the provided `train.data` will be used for testing. Default is `FALSE`.

`seed` Integer to be used internally as input for `set.seed()`. Only used if `stratified.sampling = TRUE`. In all other cases, seed is set to the current iteration number. Default is 1.

`verbose` If `TRUE`, the number of rows removed through filtering will be printed to the console. Default is `TRUE`.

`save.model` DEPRECATED `save.model = FALSE` is no longer supported; this function will always return a saved model.

`rf.variable.importance`
: DEPRECATED `rf.variable.importance = FALSE` is no longer supported; variable importance results are always returned if the `model.method` is set to 'pls' or 'rf'.

`output.summary`
: DEPRECATED `output.summary = FALSE` is no longer supported; a summary of output is always returned alongside the full performance statistics.

`return.model`
: DEPRECATED `return.model = FALSE` is no longer supported; a trained model object is always returned alongside the full performance statistics and summary.

## Value

list of the following:

1. `model` is a model object trained with all rows of `df`.
2. `summary.model.performance` is a `data.frame` with model performance statistics in summary format (2 rows, one with mean and one with standard deviation of all training iterations).
3. `full.model.performance` is a `data.frame` with model performance statistics in long format (number of rows = `num.iterations`)
4. `predictions` is a `data.frame` containing predicted values for each test set entry at each iteration of model training.
5. `importance` is a `data.frame` that contains variable importance for each wavelength. Only available for `model.method` options "rf" and "pls".

Included summary statistics:

- Tuned parameters depending on the model algorithm:
  - **Best.n.comp**, the best number of components
  - **Best.ntree**, the best number of trees in an RF model
  - **Best.mtry**, the best number of variables to include at every decision point in an RF model
- **RMSECV**, the root mean squared error of cross-validation
- **R2cv**, the coefficient of multiple determination of cross-validation (k-fold CV for pls/svm; OOB for rf)
- **RMSEP**, the root mean squared error of prediction
- **R2p**, the squared Pearson's correlation between predicted and observed test set values
- **RPD**, the ratio of standard deviation of observed test set values to RMSEP
- **RPIQ**, the ratio of performance to interquartile difference
- **CCC**, the concordance correlation coefficient
- **Bias**, the average difference between the predicted and observed values
- **SEP**, the standard error of prediction
- **R2sp**, the squared Spearman's rank correlation between predicted and observed test set values

## Author(s)

Jenna Hershberger <jmh579@cornell.edu>

## Examples

```
library(magrittr)
ikeogu.2017 %>%
  dplyr::filter(study.name == "C16Mcal") %>%
  dplyr::rename(reference = DMC.oven,
                unique.id = sample.id) %>%
  dplyr::select(unique.id, reference, dplyr::starts_with("X")) %>%
  na.omit() %>%
  train_spectra(
    df = .,
    tune.length = 3,
    num.iterations = 3,
    best.model.metric = "RMSE",
    stratified.sampling = TRUE
  ) %>%
  summary()
```

# Index

∗ **datasets**
ikeogu.2017, 7

aggregate_spectra, 2

filter_spectra, *2*, 3
format_cv, 5, *15*, *18*

ggplot, *8, 9*

ikeogu.2017, 7

plot_spectra, 8
predict_spectra, 10
pretreat_spectra, 11, *15*, *18*

save_model, *10*, 12

test_spectra, 16
train_spectra, *5*, *15*, *18*, 20