

# Package ‘spectratrait’

March 15, 2022

**Title** A simple add-on package to aid in the fitting of leaf-level spectra-trait PLSR models

**Version** 1.2.1

**Maintainer** Shawn P. Serbin <sserbin@bnl.gov>

**Description** This package provides functions to conduct standardized spectra-trait PLSR model fitting, including uncertainty analysis that follows DOI: <https://doi.org/10.1111/nph.16123>

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.2

**Imports** httr (>= 1.4.2),  
readr (>= 1.3.1),  
pls (>= 2.7-2),  
dplyr (>= 1.0.1),  
magrittr (>= 2.0.1),  
reshape2 (>= 1.4.4),  
here (>= 0.1),  
plotrix (>= 3.7-8),  
ggplot2 (>= 3.3.2),  
gridExtra (>= 2.3),  
scales (>= 1.1.1),  
testthat (>= 3.1.2)

**Suggests** devtools (>= 2.3.1),  
remotes (>= 2.2.0),  
RCurl (>= 1.98-1.2),  
knitr (>= 1.37),  
markdown,  
rmarkdown

**Depends** R (>= 2.10)

**VignetteBuilder** knitr

**NeedsCompilation** no

## R topics documented:

create\_data\_split . . . . . 2

ely_plsr_data . . . . .	3
f.coef.valid . . . . .	3
f.plot.coef . . . . .	4
f.plot.spec . . . . .	4
find_optimal_components . . . . .	5
find_optimal_comp_by_groups . . . . .	6
get_ecosis_data . . . . .	7
percent_rmse . . . . .	7
pls_permutation . . . . .	8
pls_permutation_by_groups . . . . .	9
source_GitHubData . . . . .	10
testForPackage . . . . .	10
VIP . . . . .	11
VIPjh . . . . .	11
%notin% . . . . .	11

## Index 12

---

create_data_split	<i>Create a calibration (training) / validation data split for PLSR model fitting and testing</i>
-------------------	---

---

## Description

Create a calibration (training) / validation data split for PLSR model fitting and testing

## Usage

```
create_data_split(
  dataset = NULL,
  approach = NULL,
  split_seed = 123456789,
  prop = 0.8,
  group_variables = NULL
)
```

## Arguments

dataset	input full PLSR dataset to split into cal/val datasets
approach	approach to splitting the dataset. Options: base or dplyr
split_seed	random seed to use for splitting data
prop	the proportion of data to preserve for calibration (e.g. 0.8) and validation (0.2). This sets the calibration proportion
group_variables	Use factor variables to conduct a stratified sampling for cal/val

## Value

output\_list A list containing the calibration dataset (cal\_data) and validation dataset (val\_data)

## Author(s)

Julien Lamour, Jeremiah Anderson, Shawn P. Serbin

---

ely_plsr_data	<i>Ely et al (2019) example leaf-level PLSR dataset.</i>	<i>DOI:</i>
	<i><a href="https://doi.org/10.1093/jxb/erz061">https://doi.org/10.1093/jxb/erz061</a></i>	

---

**Description**

Ely et al (2019) example leaf-level PLSR dataset. DOI: <https://doi.org/10.1093/jxb/erz061>

**Usage**

```
ely_plsr_data
```

**Format**

An object of class `data.frame` with 178 rows and 1908 columns.

---

f.coef.valid	<i>f.coef.valid</i>
--------------	---------------------

---

**Description**

f.coef.valid

**Usage**

```
f.coef.valid(plsr.out, data_plsr, ncomp, inVar)
```

**Arguments**

plsr.out	plsr model obtained with <code>jackknife = TRUE</code>
data_plsr	data used for the plsr model with Spectra the matrix of spectra
ncomp	number of selection components
inVar	Name of the PLSR model response variable

**Value**

B returns the intercept and the coefficients of the jackknife or bootstrap validation

**Author(s)**

Julien Lamour

---

`f.plot.coef`*f.plot.coef*

---

**Description**`f.plot.coef`**Usage**

```
f.plot.coef(  
  Z,  
  wv,  
  xlim = NULL,  
  position = "topright",  
  type = "Coefficient",  
  plot_label = NULL  
)
```

**Arguments**

<code>Z</code>	Coefficient matrix with each row corresponding to the coefficients and wavelength in columns
<code>wv</code>	vector of wavelengths
<code>xlim</code>	vector to change the default xlim of the plots (ex <code>xlim = c(500, 2400)</code> )
<code>position</code>	Position of the legend (see base function legend for help)
<code>type</code>	Name of the y axis and of the legend
<code>plot_label</code>	optional plot label to include with the figure

**Author(s)**

Julien Lamour

---

`f.plot.spec`*f.plot.spec*

---

**Description**`f.plot.spec`**Usage**

```
f.plot.spec(  
  Z,  
  wv,  
  xlim = NULL,  
  position = "topright",  
  type = "Reflectance",  
  plot_label = NULL  
)
```

**Arguments**

Z	Spectra matrix with each row corresponding to a spectra and wavelength in columns
wv	vector of wavelengths corresponding to the column of the spectra matrix Z
xlim	vector to change the default xlim of the plots (ex xlim = c(500, 2400))
position	Position of the legend (see base function legend for help)
type	Name of the y axis and of the legend. E.g. Reflectance, Transmittance
plot_label	optional plot label to include with the figure

**Author(s)**

Julien Lamour, Shawn P. Serbin

---

find\_optimal\_components

*Applies different methods for the determination of the optimal number of PLSR model components*

---

**Description**

Applies different methods for the determination of the optimal number of PLSR model components

**Usage**

```
find_optimal_components(
  dataset = NULL,
  targetVariable = NULL,
  method = "pls",
  maxComps = 20,
  iterations = 20,
  seg = 100,
  prop = 0.7,
  random_seed = 123456789
)
```

**Arguments**

dataset	input full PLSR dataset. Usually just the calibration dataset
targetVariable	What object or variable to use as the Y (predictand) in the PLSR model? Usually the "inVar" variable set at the beginning of a PLS script
method	Which approach to use to find optimal components. Options: pls, firstPlateau, firstMin
maxComps	maximum number of components to consider
iterations	how many different permutations to run
seg	For the built-in pls method, how many different data segments to select from the input dataset
prop	proportion of data to preserve for each permutation
random_seed	random seed to use for splitting data

**Value**

nComps the optimal number of PLSR components

**Author(s)**

Julien Lamour, Jeremiah Anderson, Shawn P. Serbin

---

find\_optimal\_comp\_by\_groups

*Uses the firstMin and firstPlateau methods for the determination of the optimal number of PLSR model components, by group (i.e. optimal selection by stratification)*

---

**Description**

Uses the firstMin and firstPlateau methods for the determination of the optimal number of PLSR model components, by group (i.e. optimal selection by stratification)

**Usage**

```
find_optimal_comp_by_groups(
  dataset = NULL,
  targetVariable = NULL,
  method = "firstPlateau",
  maxComps = 20,
  iterations = 20,
  prop = 0.7,
  random_seed = 123456789,
  group_variables = NULL
)
```

**Arguments**

dataset	input full PLSR dataset. Usually just the calibration dataset
targetVariable	What object or variable to use as the Y (predictand) in the PLSR model? Usually the "inVar" variable set at the beginning of a PLS script
method	Which approach to use to find optimal components. Options: firstPlateau, first-Min
maxComps	maximum number of components to consider
iterations	how many different permutations to run
prop	proportion of data to preserve for each permutation
random_seed	random seed to use for splitting data
group_variables	group_variables character vector of the form c("var1", "var2"... "varn") providing the factors used for stratified sampling.

**Value**

nComps the optimal number of PLSR components

**Author(s)**

asierrl, Shawn P. Serbin

---

get_ecosis_data	<i>Function to pull data from EcoSIS using the EcoSIS API</i>
-----------------	---

---

**Description**

Function to pull data from EcoSIS using the EcoSIS API

**Usage**

```
get_ecosis_data(ecosis_id = NULL)
```

**Arguments**

ecosis_id	the alphanumeric EcoSIS API dataset ID
-----------	--

**Value**

EcoSIS spectral dataset object

**Author(s)**

Shawn P. Serbin, Alexey Shiklomanov

**Examples**

```
## Not run:
ecosis_id <- "960dbb0c-144e-4563-8117-9e23d14f4aa9"
dat_raw <- get_ecosis_data(ecosis_id = ecosis_id)
head(dat_raw)
names(dat_raw)[1:40]

## End(Not run)
```

---

percent_rmse	<i>Calculate RMSE and percent RMSE with PLSR model results</i>
--------------	--

---

**Description**

Calculate RMSE and percent RMSE with PLSR model results

**Usage**

```
percent_rmse(
  plsr_dataset = NULL,
  inVar = NULL,
  residuals = NULL,
  range = "full"
)
```

**Arguments**

plsr_dataset	input plsr dataset
inVar	the trait variable used in the calculation of RMSE
residuals	predicted minus observed residual vector from either a cross-validation CV or independent validation
range	calculate over the full data range or the 95% of data range. options full or 95perc

**Value**

output a list containing the rmse and perc\_rmse. output <- list(rmse = rmse, perc\_rmse = perc\_rmse)

**Author(s)**

Shawn P. Serbin

---

pls_permutation	<i>Run a PLSR model permutation analysis. Can be used to determine the optimal number of components or conduct a bootstrap uncertainty analysis</i>
-----------------	---

---

**Description**

See Serbin et al. (2019). DOI: <https://doi.org/10.1111/nph.16123>

**Usage**

```
pls_permutation(
  dataset = NULL,
  targetVariable = NULL,
  maxComps = 20,
  iterations = 20,
  prop = 0.7,
  verbose = FALSE
)
```

**Arguments**

dataset	input full PLSR dataset. Usually just the calibration dataset
targetVariable	What object or variable to use as the Y (predictand) in the PLSR model? Usually the "inVar" variable set at the beginning of a PLS script
maxComps	maximum number of components to use for each PLSR fit
iterations	how many different permutations to run
prop	proportion of data to preserve for each permutation
verbose	Should the function report the current iteration status/progress to the terminal or run silently? TRUE/FALSE. Default FALSE

**Value**

output a list containing the PRESS and coef\_array. output <- list(PRESS=press.out, coef\_array=coefs)



**Author(s)**

Julien Lamour, Shawn P. Serbin

---

```
pls_permutation_by_groups
```

*Run a PLSR model permutation analysis stratified by selected "groups". Can be used to determine the optimal number of components or conduct a bootstrap uncertainty analysis*

---

**Description**

Run a PLSR model permutation analysis stratified by selected "groups". Can be used to determine the optimal number of components or conduct a bootstrap uncertainty analysis

**Usage**

```
pls_permutation_by_groups(  
  dataset = NULL,  
  targetVariable = NULL,  
  maxComps = 20,  
  iterations = 20,  
  prop = 0.7,  
  group_variables = NULL,  
  verbose = FALSE  
)
```

**Arguments**

dataset	input full PLSR dataset. Usually just the calibration dataset
targetVariable	What object or variable to use as the Y (predictand) in the PLSR model? Usually the "inVar" variable set at the beginning of a PLS script
maxComps	maximum number of components to use for each PLSR fit
iterations	how many different permutations to run
prop	proportion of data to preserve for each permutation
group_variables	Character vector of the form c("var1", "var2" ... "varn") providing the factors used for stratified sampling in the PLSR permutation analysis
verbose	Should the function report the current iteration status/progress to the terminal or run silently? TRUE/FALSE. Default FALSE

**Value**

output a list containing the PRESS and coef\_array. `output <- list(PRESS=press.out, coef_array=coefs)`

**Author(s)**

asierrl, Shawn P. Serbin, Julien Lamour

---

source_GitHubData	<i>Function to source text data from GitHub</i>
-------------------	---

---

**Description**

Function to source text data from GitHub

**Usage**

```
source_GitHubData(url, sep = ",", header = TRUE)
```

**Arguments**

url	http/https URL to the github dataset
sep	dataset file delimiter
header	TRUE/FALSE does the file have a column header?

**Author(s)**

[gist.github.com/christophergandrud/4466237](https://gist.github.com/christophergandrud/4466237)

---

testForPackage	<i>Function to check for installed package</i>
----------------	--

---

**Description**

Function to check for installed package

**Usage**

```
testForPackage(pkg)
```

**Arguments**

pkg	name of package to check if installed not presently used
-----	--

---

VIP	<i>VIP returns all VIP values for all variables and all number of components, as a ncomp x nvars matrix.</i>
-----	--

---

**Description**

VIP returns all VIP values for all variables and all number of components, as a ncomp x nvars matrix.

**Usage**

VIP(object)

**Arguments**

object                  fitted pls::plsr object

---

VIPjh	<i>VIPjh returns the VIP of variable j with h components</i>
-------	--

---

**Description**

VIPjh returns the VIP of variable j with h components

**Usage**

VIPjh(object, j, h)

**Arguments**

object                  fitted pls::plsr object  
 j                        which variable in the fitted pls::plsr object  
 h                        the number of components in the fitted pls::plsr object to calculate the VIP

---

%notin%	<i>Not %in% function</i>
---------	--------------------------

---

**Description**

Not %in% function

**Usage**

x %notin% table

**Arguments**

x                        initial list  
 table                  list to check against

# Index

## \* datasets

ely\_plsr\_data, [3](#)

%notin%, [11](#)

create\_data\_split, [2](#)

ely\_plsr\_data, [3](#)

f.coef.valid, [3](#)

f.plot.coef, [4](#)

f.plot.spec, [4](#)

find\_optimal\_comp\_by\_groups, [6](#)

find\_optimal\_components, [5](#)

get\_ecosis\_data, [7](#)

percent\_rmse, [7](#)

pls\_permutation, [8](#)

pls\_permutation\_by\_groups, [9](#)

source\_GitHubData, [10](#)

testForPackage, [10](#)

VIP, [11](#)

VIPjh, [11](#)