Package 'windows.pls'

July 22, 2025

Title Segmentation Approaches in Chemometrics

Version 0.1.0

Maintainer Elia Gonzato <elia.gonzato@outlook.it>

Description Evaluation of prediction performance of smaller regions of spectra for Chemometrics. Segmentation of spectra, evolving dimensions regions and sliding windows as selection methods. Election of the best model among those computed based on error metrics. Chen et al.(2017) <doi:10.1007/s00216-017-0218-9>.

License MIT + file LICENSE

URL https://github.com/egonzato/windows.pls

BugReports https://github.com/egonzato/windows.pls/issues

Depends R (>= 2.10)

Imports dplyr, ggplot2, grDevices, mdatools, readr, stringr, tidyr, tidyverse

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.2.3

Author Elia Gonzato [aut, cre, cph]

Repository CRAN

Date/Publication 2023-08-09 17:30:02 UTC

Contents

`beer` *Beer Dataset from Near Infrared Spectroscopy*

Description

The beer dataset contains 60 samples published by Norgaard et al. Recorded with a 30mm quartz cell on the undiluted degassed beer and measured from 1100 to 2250 nm (576 data points) in steps of 2 nm. A good playing ground for regression methods starting from spectral intensities.

Usage

```
beer
```

Format

`beer`**:**

A data frame with 80 rows and 577 columns:

y Original extract concentration

xtrain Intesities measured on 576 different data points

Source

https://www.kaggle.com/datasets/robertoschimmenti/beer-nir?resource=download

References

Norgaard, L., Saudland, A., Wagner, J., Nielsen, J. P., Munck, L., & Engelsen, S. B. (2000). Interval partial least-squares regression (iPLS): a comparative chemometric study with an example from near-infrared spectroscopy. Applied Spectroscopy, 54(3), 413–419. Adapted from a R dataset available as part of the OHPL package (https://search.r-project.org/CRAN/refmans/OHPL/html/00Index.html).

`convert.names.wl` *Turns wavelengths into variable's names*

Description

Turns wavelengths into variable's names

Usage

```
convert.names.wl(start = NULL, stop = NULL, step = 2)
```

Arguments

`start`	First wavelength of the spectra.
`stop`	Last wavelength of the spectra.
`step`	Distance between each recorded wavelength.

Value

Returns vector with syntactically valid names for each wavelength

Examples

```
data(beer)
X=beer[,2:ncol(beer)]
head(names(X))
names(X)=convert.names.wl(1100,2250,2)
head(names(X))
```

`cv.wpls` *Cross-validation for segmented spectral regions of the original spectra.*

Description

Computes and stores cross-validation metrics for one of the three possible modes 'wpls', 'epls', 'swpls'.

Usage

```
cv.wpls(
  xblock = NULL,
  yblock = NULL,
  windows = 3,
  window.size = 30,
  increment = 10,
  cv = 10,
  scale = FALSE,
  ncp = 10,
  mode = "wpls"
)
```

Arguments

xblock	A matrix containing one spectra for each observation.
yblock	A vector containing the concentration associated to each spectra in the *xblock* matrix.
windows	Parameter used when either 'wpls' or 'ewpls' is chosen. Points out how many windows the user wants to divide the spectra in.
window.size	Parameter used when 'swpls' is chosen. Indicates the width of the window that slides along the spectra.
increment	Parameter used when 'swpls' is chosen. Indicates how many steps the window slides forward.
cv	Number of segments used for cross-validation.
scale	logical, asks to perform standardization.
ncp	Maximum number of principal components to be computed for each model.
mode	'wpls','ewpls' or 'swpls', see **Details** for more.

Details

NIR and Vis-NIR technologies are used to obtain spectra which might contain helpful information about the content of the samples the user is investigating. Since this method has been combined with multivariate statistical methods, researchers have been questioning the importance of using spectra in its entirety or if it might be a better solution to divide it in smaller regions which can guarantee higher performance in terms of predictions. Several methods have been proposed, from selecting only some regions to selecting combinations of those which are performing the best. This function provides three possibilities:

1. **'wpls'**, which stands for *Window PLS*, divides the original spectra into several windows, computes PLS and stores metrics of interest such as RMSE and R2 for calibration and cross-validation both.
2. **'ewpls'**, which stands for *Evolving Window PLS*, divides the original spectra into several windows, but each new window incorporates the previous ones, so that we are comparing smaller windows with the entire spectra.

3. **'swpls'**, which stands for *Sliding Window Window PLS*, ,asks the width of the window that will be used to compute the model and the step that the window will make forward in the spectra so that a new model is calculated. In this way the window slides along spectra and computes several models, which will be compared with metrics.

This function proposes a simpler version of **iPLS**, that can be found in the **mdatools** package, which divides the spectra in smaller segments and tries to find the combination with the lowest RMSE in cross-validation.

Value

Returns a list containing:

`xblock`	Matrix containing spectra used to train the model.
`yblock`	Vector containing values of the dependent variable.
`cal`	List containing RMSE and R2 of calibratrion.
`cv`	List containing RMSE and R2 of cross-validation.
`ncp`	Number of components used to compute the model.
`scale`	Contains logical condition used for standardization.
`cv.segment`	Number of segments used for cross-validation.

References

1. Chen, J., Yin, Z., Tang, Y. et al. Vis-NIR spectroscopy with moving-window PLS method applied to rapid analysis of whole blood viscosity. Anal Bioanal Chem 409, 2737–2745 (2017).
2. Y.P. Du, Y.Z. Liang, J.H. Jiang, R.J. Berry, Y. Ozaki, Spectral regions selection to improve prediction ability of PLS models by changeable size moving window partial least squares and searching combination moving window partial least squares, Analytica Chimica Acta, Volume 501, Issue 2, 2004, Pages 183-191,
3. **mdatools** package, https://github.com/svkucheryavski/mdatools

Examples

```
data(beer)
conc=beer[,1]
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
conc=unlist(conc)
mywpls=cv.wpls(sp, conc,mode='wpls', windows = 5)
```

`global.r2` *Plots in a single window the R2 of each model.*

Description

Plots in a single window the R2 of each model.

Usage

```
global.r2(
  wpls = NULL,
  col.cal = "blue",
  col.cv = "red",
  col.strip.background = "orange",
  xlab = NULL,
  ylab = NULL,
  title = NULL
)
```

Arguments

`wpls`	object obtained from **cv.wpls**.
`col.cal`	color for the calibration line.
`col.cv`	color for the cross-validation line.
`col.strip.background`	color of the banner for each window.
`xlab`	title of the x axis.
`ylab`	title of the y axis.
`title`	title of the plot.

Value

Plot of R2 of each spectra region used to compute PLS.

Examples

```
data(beer)
conc=beer[,1]
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
conc=unlist(conc)
mywpls=cv.wpls(sp, conc,mode='wpls', windows = 5)
global.r2(mywpls,col.cal='navy',
              col.cv='red',
              col.strip.background='orange',
              xlab='Component',
              ylab=expression(R^2))
```

`global.rmse` *Plots in a single window the RMSE of each model.*

Description

Plots in a single window the RMSE of each model.

Usage

```
global.rmse(
  wpls = NULL,
  col.cal = "blue",
  col.cv = "red",
  col.strip.background = "steelblue",
  xlab = NULL,
  ylab = NULL,
  title = NULL
)
```

Arguments

`wpls`	object obtained from **cv.wpls**.
`col.cal`	color for the calibration line.
`col.cv`	color for the cross-validation line.
`col.strip.background`	color of the banner for each window.
`xlab`	title of the x axis.
`ylab`	title of the y axis.
`title`	title of the plot.

Value

Plot of RMSE of each spectra region used to compute PLS.

Examples

```
data(beer)
conc=beer[,1]
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
conc=unlist(conc)
mywpls=cv.wpls(sp, conc,mode='wpls', windows = 5)
global.rmse(mywpls,col.cal='navy',
               col.cv='red',
               col.strip.background='orange',
               xlab='Component',
               ylab='RMSE')
```

`map.best.window` *Plots spectra highlighting windows with the best performance.*

Description

Plots spectra highlighting windows with the best performance.

Usage

```
map.best.window(
  wpls = NULL,
  fade = 0.7,
  col.window = "steelblue",
  xlab = "Wavelengths",
  ylab = "Absorbance",
  title = NULL,
  legend = NULL
)
```

Arguments

`wpls`	object obtained from **cv.wpls**.
`fade`	opacity of the window.
`col.window`	color of the window that highlights the region.
`xlab`	title of the x axis.
`ylab`	title of the y axis.
`title`	title of the plot.
`legend`	description description

Value

Plot of the spectra with a window that highlights the region with the lowest cross-validation error.

Examples

```
data(beer)
conc=beer[,1]
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
conc=unlist(conc)
mywpls=cv.wpls(sp, conc,mode='wpls', windows = 5)
map.best.window(mywpls)
```

`map.spectra.gradient` *Colors and plots each spectra based on the associated concentration of the outcome variable*

Description

Colors and plots each spectra based on the associated concentration of the outcome variable

Usage

```
map.spectra.gradient(
  xblock = NULL,
  yblock = NULL,
  legend.title = "Gradient",
  plot.title = "Spectra with gradient based on Y variable",
  xlab = "Wavelength",
  ylab = "Absorbance",
  grad = 10,
  l.width = 0.75,
  col.legend = NULL
)
```

Arguments

`xblock`	A matrix containing one spectra for each observation.
`yblock`	A vector containing the concentration associated to each spectra in the *xblock* matrix.
`legend.title`	Title of the legend which displays the gradient.
`plot.title`	Title of the plot.
`xlab`	Title of the x axis.
`ylab`	Title of the y axis.
`grad`	Number of colors for the gradient's palette.
`l.width`	Width of each spectra.
`col.legend`	Deletes presence of the legend.

Value

Plot with spectra of all observations, mapped with the intensity of the associated concentration.

Examples

```
data(beer)
X=beer[,2:ncol(beer)]
names(X)=convert.names.wl(1100,2250,2)
Y=unlist(beer[,1])
map.spectra.gradient(X,Y)
```

`r2.single.window` *Plots R2 of calibration and cross-validation of a single nindow.*

Description

Plots R2 of calibration and cross-validation of a single nindow.

Usage

```
r2.single.window(
  wpls = NULL,
  condition = "Complete",
  shape.cal = 19,
  shape.cv = 19,
  width = 1,
  size = 2,
  col.cal = "blue",
  col.cv = "red",
  xaxis.title = "Component",
  yaxis.title = expression(R^2),
  title = paste("Plot of R2 for the", condition, "model"),
  legend.name = NULL,
  x.legend = 0.9,
  y.legend = 0.2
)
```

Arguments

`wpls`	object obtained from **cv.wpls**.
`condition`	name of the Window the user wants to plot.
`shape.cal`	shape of the point of the calibration line.
`shape.cv`	shape of the point of the cross-validation line.
`width`	width of the line.
`size`	size of the points of calibration and cross-validation.
`col.cal`	color for the calibration line.
`col.cv`	color for the cross-validation line.
`xaxis.title`	title of the x axis.
`yaxis.title`	title of the y axis.
`title`	title of the plot.
`legend.name`	displays legend and its name.
`x.legend`	position of the legend on the x axis, ranges from 0 to 1.
`y.legend`	position of the legend on the y axis, ranges from 0 to 1.

Value

Plot of R2 of the region requested by the user.

Examples

```
data(beer)
conc=beer[,1]
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
conc=unlist(conc)
mywpls=cv.wpls(sp, conc,mode='wpls', windows = 5)
r2.single.window(mywpls,'Window2')
```

---

rmse.single.window *Plots RMSE of calibration and cross-validation of a single wnindow.*

Description

Plots RMSE of calibration and cross-validation of a single wnindow.

Usage

```
rmse.single.window(
  wpls = NULL,
  condition = "Complete",
  shape.cal = 19,
  shape.cv = 19,
  width = 1,
  size = 2,
  col.cal = "blue",
  col.cv = "red",
  xaxis.title = "Component",
  yaxis.title = "RMSE",
  title = paste("Plot of RMSE for the", condition, "model"),
  legend.name = NULL,
  x.legend = 0.1,
  y.legend = 0.2
)
```

Arguments

wpls	object obtained from **cv.wpls**.
condition	name of the Window the user wants to plot.
shape.cal	shape of the point of the calibration line.
shape.cv	shape of the point of the cross-validation line.
width	width of the line.

size	size of the points of calibration and cross-validation.
col.cal	color for the calibration line.
col.cv	color for the cross-validation line.
xaxis.title	title of the x axis.
yaxis.title	title of the y axis.
title	title of the plot.
legend.name	displays legend and its name.
x.legend	position of the legend on the x axis, ranges from 0 to 1.
y.legend	position of the legend on the y axis, ranges from 0 to 1.

Value

Plot of RMSE of the region requested by the user.

Examples

```
data(beer)
conc=unlist(beer[,1])
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
mywpls=cv.wpls(sp, conc,mode='wpls', windows = 5)
rmse.single.window(mywpls,'Window2')
```

---

segment.windows *Displays how spectra are divided in windows*

Description

Displays how spectra are divided in windows

Usage

```
segment.windows(
  xblock = NULL,
  yblock = NULL,
  windows = 3,
  fade = 0.3,
  xlab = "Wavelength",
  ylab = "Absorbance",
  title = paste("Spectra divided in", windows, "segments", sep = " "),
  legend = NULL,
  grad = 10
)
```

Arguments

xblock	A matrix containing one spectra for each observation.
yblock	A vector containing the concentration associated to each spectra in the *xblock* matrix.
windows	Number of windows the spectra has to be divided in.
fade	Opacity of the window.
xlab	Title of the x axis.
ylab	Title of the y axis.
title	Title of the plot.
legend	Name of the substance which drives the gradient of spectra's mapping.
grad	Number of colors that are used to build the gradient.

Value

Plot of spectra in which segments have a different background color.

Examples

```
data(beer)
conc=unlist(beer[,1])
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
segment.windows(sp,conc,windows=7,fade=0.25)
```

---

sel.best.window *Selection of the best window computed with cv.wpls*

Description

Takes as input the object containing metrics of the several models computed with cv.wpls and selects the best basing on the lowest RMSE available; then computes PLS and gives as output an object containing results.

Usage

```
sel.best.window(wpls = NULL)
```

Arguments

wpls	object obtained from cv.wpls.

Value

An object containing results of the best model. Has the same content of a model obtained from the function **pls** of **mdatools**.

Examples

```
data(beer)
conc=beer[,1]
sp=beer[,2:ncol(beer)]
names(sp)=convert.names.wl(1100,2250,2)
conc=unlist(conc)
mywpls=cv.wpls(sp, conc,mode='wpls', windows = 5)
best.pls=sel.best.window(mywpls)
```

Index