

Package ‘soil.spec’

February 15, 2012

Type Package

Title Soil spectral data exploration and regression functions

Version 1.4

Date 2010-07-10

Author Thomas Terhoeven-Urselmans

Maintainer Thomas Terhoeven-Urselmans <t.urselmans@cgiar.org>

Description This package combines existing R functions with new code for soil spectral analysis. The result is an easy to use tool for (i) importing of spc-files, (ii) principal component analysis, (iii) sample selection using the Kennard-Stone algorithm, (iv) spectral transformation and (v) comparison of regression methods.

Suggests KernSmooth,pls,splines,survival,lattice,gbm,class,hexView,wavelets,car,e1071

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2010-07-24 20:43:40

R topics documented:

soil.spec-package	2
isric	2
ken.sto	4
make.comp	5
pc.space	6
prco	7
read.spc	9
regr	10
trans	13

Index	15
--------------	-----------

soil.spec-package *Soil spectral data exploration and regression functions*

Description

This package combines existing R functions with new code for soil spectral analysis. The result is an easy to use tool for (i) importing of spc-files, (ii) principal component analysis, (iii) sample selection using the Kennard-Stone algorithm, (iv) spectral transformation and (v) comparison of regression methods.

Details

Package: soil.spec
 Type: Package
 Version: 1.4
 Date: 2010-07-10
 License: GPL-2
 LazyLoad: yes

Author(s)

Thomas Terhoeven-Urselmans
 Maintainer: Thomas Terhoeven-Urselmans <t.urselmans@cgiar.org>

isric *Script for publication Terhoeven-Urselmans et al. 2010.*

Description

The script for the publication: Terhoeven-Urselmans,T., Vagen,T.G., Spaargaren,O. and Shepherd,K.D. 2010. Prediction of soil fertility properties from a globally distributed soil mid-infrared spectral library. Soil Science Society of America Journal. Accepted, doi:10.2136/sssaj2009.0218.

Usage

```
isric(spec, chem, plo)
```

Arguments

spec a numerical matrix containing the raw spectra.
 chem a numerical data frame containing the reference values.
 plo a data frame containing two columns. See details

Details

The first column of plo named "Plot_code" contains the plot code as factor. The second column named "Depth" contains the depth-layer as factor. An example is "0 to 14" and means that this soil sample was taken from 0 to 14 cm depth.

Value

isric returns a list with class "isric" containing the following components:

Table_1 a matrix containing the summary of principal component analysis of first derivative spectra. Given as Table 1 in the publication.

Table_2_summary a table containing the summary of chem. Given as Table 2 in the publication.

Table_2_sd a numeric vector containing the standard deviation of chem. Part of Table 2 in publication

Table_3_calibration a matrix containing the calibration set statistics. Given as Table 3 in the publication.

Table_3_validation a matrix containing the validation set statistics. Given as Table 3 in the publication.

Table_4_all a list containing the correlation between soil property values of the whole data set. Given as upper triangle in Table 4 in the publication.

Table_4_calibration a list containing the correlation between soil property values of the calibration data set only. Given as lower triangle in Table 4 in the publication.

Calibration_predicted a data frame containing for the soil property values of the calibration set the (i) untransformed values, (ii) box-cox transformed values, (iii) predicted box-cox transformed values and (iv) back-transformed predicted values.

Validation_predicted a data frame containing for the soil property values of the validation set the (i) untransformed values, (ii) box-cox transformed values, (iii) predicted box-cox transformed values and (iv) back-transformed predicted values.

Author(s)

Thomas Terhoeven-Urselmans

ken.sto

*Sample selection based on the Kennard-Stone algorithm***Description**

The function chooses based on Euclidean distance measure most representative samples. One can (i) select a number or a percentage of a sample set or (ii) divide a sample set into calibration and representative validation set.

Usage

```
ken.sto(inp, per = "TRUE", per.n = 0.3, num, va = "FALSE", sav = "TRUE", save.path = "NULL", output.name

## S3 method for class 'ken.sto'
plot(x, ...)
```

Arguments

inp	a numerical matrix or data.frame containing the input spectra
per	a logical value indicating whether the selected samples should be a percentage (given in per.n) or a set number (given in num) of inp. The default "TRUE" takes a percentage.
per.n	a numerical value between 0 and 1.
num	a numerical value between 1 and the sample number minus 1.
va	a logical value indicating whether to select samples out of inp or to divide them into a calibration and validation set.
sav	a logical value indicating whether the function output shall be saved.
save.path	a character giving the path name where the function output shall be saved.
output.name	a character giving the function output name, in case sav is "TRUE".
x	an object of class "ken.sto".
...	additional arguments.

Details

Sample selection is done following and adapted procedure from Kennard & Stone (1969). It is a stepwise procedure by maximizing the Euclidean distance based on the important number of principal components to the objects already chosen. The number of important principal components is selected so that the increase in cumulative explained variance within the next three components is lower than 4 percent. The starting samples are the two extreme samples (most negative and positive ones) of the important principal components.

per.n having a value of 0.4 while va equal to "FALSE" chooses 40 percent of the sample set. When va is equal to "TRUE" the validation set comprises 40 percent of the sample set.

A graph is given back showing the selected samples in the principal component space (only the important PC's). This is the same graphic generated by [plot.ken.sto](#).

Value

ken.sto returns a list with class "ken.sto" containing the following components:

Calibration and validation set

the logical object va.

Number important PC

integer giving the number of chosen important components - important for choosing the starting samples.

PC space important PC

score value matrix of important principal components.

Chosen samples names

chosen sample names when va equal to "FALSE".

Chosen row number

chosen row numbers when va equal to "FALSE".

Chosen calibration sample names

chosen calibration sample names when va equal to "TRUE".

Chosen calibration row number

chosen calibration row numbers when va equal to "TRUE".

Chosen validation sample names

chosen validation sample names when va equal to "TRUE".

Chosen validation row number

chosen validation row numbers when va equal to "TRUE".

Author(s)

Thomas Terhoeven-Urselmans

References

Kennard, R. W. and Stone, L. A. (1969) *Computer aided design of experiments*. *Technometrics* 11(1), 137-148.

make.comp

Making spectra compatible

Description

The function make.comp interpolates spectral absorbance/reflectance values to a new waveband vector by performing cubic spline interpolation using the spline function from the stats package.

Usage

make.comp(spec, new.waveb, repl.out.range = "FALSE", sav = "FALSE", save.path = "NULL", output.name = "O

Arguments

<code>spec</code>	a numerical matrix containing the spectra.
<code>new.waveb</code>	a numeric vector containing the new wavebands.
<code>repl.out.range</code>	a logical value indicating whether extrapolated values outside the waveband range of <code>spec</code> should be replaced by NA values.
<code>sav</code>	a logical value indicating whether the function output (original and compatible spectra) should be saved.
<code>save.path</code>	a character vector giving the path where to save the function output. If "NULL" (default), the current working directory is taken.
<code>output.name</code>	a character vector giving the name of the function output in case <code>sav</code> is "TRUE".
<code>save.as</code>	a character vector indicating the format of the saved output. "workspace" saves the function output named with <code>output.name</code> as workspace. "csv.file" saves the function output as csv-file.

Details

Some spectrometer adapt the wavenumber once in a while. Thus, waveband position might be different between spectra measured in different batches. This function makes the spectra compatible.

Cubic spline interpolation allows to extrapolate values outside the original waveband range. The user can choose if the extrapolated values are replaced by NA values.

The column names of `spec` need to be numeric (no letters e.g. in the first position allowed).

Value

`make.comp` returns a list with class "make.comp" containing the following components:

<code>original.spectra</code>	a matrix giving the original spectra.
<code>compatible.spectra</code>	a matrix giving the compatible spectra.

Author(s)

Thomas Terhoeven-Urselmans

pc.space

Projection of new spectra into a principal component space

Description

New spectra are projected into the principal component space of a user given spectral sample set in order to check if the new spectra belong to the same population. The `prcomp` function from the `stats` package is used.

Usage

```
pc.space(base, new)

## S3 method for class 'pc.space'
plot(x, ...)
```

Arguments

base	a numerical spectral matrix containing the population samples.
new	a numerical matrix containing the new samples, which are checked being part of the population of base.
x	an object of class "pc.space".
...	additional arguments.

Details

A principal component analysis is performed on base and the spectra from new are projected/predicted into this score space. The output (the same then using `plot.pc.space`) is plotted that the user can see if the spectra in new (in red) belong to the population of base.

`plot.pc.space` does the plotting of an object of class "pc.space".

Value

`pc.space` returns a list with class "pc.space" containing the following components:

<code>prco.base</code>	a matrix containing the score values of base.
<code>prco.new</code>	a matrix containing the score values of new.

Author(s)

Thomas Terhoeven-Urselmans

prco

Principal component analysis

Description

`prco` uses the `prcomp` function in the `stats` package. In addition the important number of principal components is determined.

Usage

```
prco(x, sav = "FALSE", save.path = "NULL", output.name = "prco")

## S3 method for class 'prco'
plot(x, ...)

## S3 method for class 'prco'
summary(object, ...)
```

Arguments

<code>x</code>	a numerical, spectral matrix in <code>prco</code> . An object of class <code>"prco"</code> in <code>plot.prco</code> .
<code>sav</code>	a logical value indicating whether the function output (original and compatible spectra) should be saved.
<code>save.path</code>	a character vector giving the path where to save the function output. If <code>"NULL"</code> (default), the current working directory is taken.
<code>output.name</code>	a character vector giving the name of the function output in case <code>sav</code> is <code>"TRUE"</code> .
<code>...</code>	additional arguments.
<code>object</code>	an object of class <code>"prco"</code> .

Details

The number of important principal components is selected so that the increase in cumulative explained variance within the next three components is lower than 4 percent.

The `plot.prco` function plots two plots: (i) the score values of the important principal components using the `pairs` function and (ii) the loading values against the wavebands for the first two principal components.

The `summary.prco` function prints the cumulative explained variance of the first ten principal components and the number of important principal components.

Value

`prco` returns a list with class `"prco"` containing the following components:

<code>prcomp</code>	a list of class <code>"prcomp"</code> .
<code>prco</code>	a numerical matrix containing the score values of the important principal components.
<code>i.pc</code>	numeric indicating the number of important principal components.

Author(s)

Thomas Terhoeven-Urselmans

read.spc	<i>Reads spectral spc-files into R</i>
----------	--

Description

read.spc reads binary spectral spc-files from a folder into R. The spectra can be made compatible (see details in [make.comp](#)) either to the first sample wavebands or to the standard wavebands of the ICRAF spectral lab. Information from the scanning method is gathered to check on spectral comparability.

Usage

```
read.spc(loa = "NULL", save.path = "NULL", sav = "FALSE", output.name = "Spectral matrix", save.as = "wo
```

Arguments

loa	a character giving the path name where the spc-files are stored. If "NULL" (default), the spc-files are stored in the current working directory.
save.path	a character giving the path name where the function output shall be saved.
sav	a logical value indicating whether the function output shall be saved.
output.name	a character giving the function output name, in case sav is "TRUE".
save.as	a character vector indicating the format of the saved output. "workspace" saves the function output named with output.name as workspace. "csv.file" saves the function output as csv-file.
wavenumber	a character giving the way how the samples should be made compatible. If "first.sample", all spectra are made compatible to the first read sample. If "ICRAF", all spectra are made compatible to the standard wavebands of the ICRAF spectral lab.

Details

Spectra from the near- and mid-infrared range can be read. In case the spc-files saved in loa comprise both ranges the function output is given separate for each range.

The function allows to read only spectra in one go, when they have the same material (e.g. soil or plant), were scanned with the same resolution and have the same zero filling. If there are still small differences in the number of wavebands, the spectra are made compatible depending on the argument wavenumber. In case spectra with different materials shall be read, the user has to decide the material via graphical interface.

Value

read.spc returns a list with class "read.spc" containing the following components:

spectra	a numerical matrix containing the read spectra.
additional.information	a data frame containing some scanning method details.

Author(s)

Thomas Terhoeven-Urselmans

regr

*Regression functions for soil spectral analysis***Description**

Different regression functions (partial least-squares, boosted regression trees, support vector machines) can be chosen for calibrations of one or more constituents. Function settings are optimized for soil spectral analysis, but can be varied. Possible spectral transformations are described in the [trans](#) function.

Usage

```
regr(x, y, sav = "NULL", spec.type = "wavelet transformed", reg = "svm", per = "TRUE", per.n = 0.3, num, n
```

```
## S3 method for class 'regr'
plot(x, ...)
```

```
## S3 method for class 'regr'
summary(object, ...)
```

```
## S3 method for class 'regr'
pred(new, model, output.name="test", sav="NULL")
```

Arguments

x	a numerical data.frame or matrix containing the raw spectra in regr. An object of class "regr" in plot.regr.
y	a numerical data.frame or matrix containing the constituents.
sav	a character vector giving the path where to save the function output. If "NULL" (default), the current working directory is taken. As well used in pred.regr.
spec.type	a character giving the desired spectral transformation. Available are "raw" (raw spectra), "derivative" (derivative spectra), "continuum removed" (continuum removed spectra) and "wavelet transformed" (wavelet coefficients).
reg	a character giving the regression method. Available are "pls" (partial least-squares), "brt" (boosted regression trees) and "svm" (support vector machines).
per	a logical indicating whether validation samples should be chosen as a percentage from x (given in per.n). If "FALSE" object num is taken.
per.n	a numeric between 0 and 1 giving the percentage of validation samples to choose.
num	an integer giving the number of validation samples when per is "FALSE".
model.name	a character naming the model output.

<code>drv</code>	an integer between 0 and 3 giving the order of derivative. The value 0 performs smoothing based on bandwidth.
<code>bandwidth</code>	an integer between 1 and 30 defining the smoothing interval in wavebands.
<code>validation</code>	a character defining the type of cross-validation procedure when <code>reg</code> is equal to "pls". Available are "none" (no cross-validation procedure), "CV" (cross-validation in 10 segments) and "LOO" (leave-one-out cross-validation).
<code>filte</code>	a character defining the wavelet filter in <code>dwt</code> function from <code>wavelets</code> package.
<code>level</code>	a character defining the level of wavelet coefficients extraction (1 to 10 possible; 1 yields 512 coefficients, 2 yields 256 coefficients...).
<code>distribution</code>	a character giving in the distribution in the <code>gbm.fit</code> function.
<code>n.trees</code>	an integer giving the total number of trees to fit in the <code>gbm.fit</code> function.
<code>shrinkage</code>	a character giving in the shrinkage parameter in the <code>gbm.fit</code> function.
<code>kerne</code>	a character giving in the kernel used in the <code>svm</code> function.
<code>...</code>	additional arguments.
<code>object</code>	an object of class "regr".
<code>new</code>	a numerical data frame or matrix containing the new spectra.
<code>model</code>	an object of class "regr".
<code>output.name</code>	a character naming the prediction output csv-file in <code>pred.regr</code> .

Details

Missing values in `y` are allowed.

`regr` uses the `mvr` function in the `pls` package for partial least-squares regression, the `gbm.fit` function in the `gbm` package for boosted regression trees and the `svm` function in the `e1071` package for support vector machines regression. The number of important PLS latent variables and the `svm` parameter optimization is done automatically based on experience with soil spectra.

`spec.type` uses for spectral transformation (i) the `locpoly` function in `KernSmooth` package for derivative calculation, (ii) the `chull` and `approx` functions in "KernSmooth" package for continuum removal and (iii) the `dwt` function in `wavelets` package for extraction of wavelet coefficients. Experiences showed for wavelet decomposition that the best ratio of prediction performance and sparse spectral representation is reached when all 128 wavelet coefficients from decomposition level three are taken (which is the default).

Settings in the used functions for regression and transformation are chosen based on experience with soil spectra calibrations. It is recommended to take the given default values. Nevertheless, the settings can be adapted to a certain degree. In case you want to use complete functionality use the named functions directly. If `reg` is "brt", the number of samples has to be more than 70.

Column names of `x` and `new` must contain the wavebands. Wavebands are made automatically compatible if needed (see details in [read.spc](#)).

Constituent values are not always normally distributed. This can violate prerequisites for regression methods. Thus, transformation prior regression can solve this problem. The `regr` function uses log, square root and box-cox transformation aside untransformed values and let the user decide graphically which transformation to take for each constituent.

Predictions from `pred.regr` are given back with the prediction uncertainty for each individual sample (based on the validation set prediction error). The prediction uncertainty is calculated as the root median square error of prediction (RMedianSEP) using a moving window of in maximum 50 samples with similar predicted values. From the RMedianSEP the confidence interval is calculated. Predictions are only made if (i) the new spectrum lies within the mahalanobis space of the calibration set, (ii) there is a local neighbor within of 5 and (iii) the predicted value lies within the calibration set range. Otherwise they are set to NA values. Mahalanobis distance can only be calculated when the number of calibration samples is higher than the number of wavebands/variables.

Calibration statistics contains for each constituents (i) `n` the number of samples used in calibration, (ii) `r2` the coefficient of determination for the linear regression of measured against predicted values, (iii) `a` the slope of the regression line, (iv) `bias` the bias, (v) `RMSEC` the root means square error of calibration, (vi) `RPD` the ratio of constituent standard deviation to `RMSEC`, (vii) `n LV` the number of latent variables used when `reg` is equal to "pls", (viii) `n bc out` the number of backtransformed values being NA values after box-cox transformation and (ix) `n trees` the number of trees when `reg` is equal to "brt". Validation statistics contains for each constituents points (i) to (vi). The `RMSEC` is logically the `RMSEP`.

The calibration and validation regressions of all constituents are plotted and the statistics printed in the Console.

Nearly each run of `regr` yields following warning message: "1: in optimize(f=function(lambda)...". Its related with the box-cox transformation, but does not have any impact or negative side effects.

Value

`regr` returns a list with class "regr" containing the following components excluding the last four ones. `pred.regr` returns a list with class "pred.regr" containing the last four components `output.name`, `predicted.values`, `method`, and `spectral.transformation` (see below):

<code>model.name</code>	a character naming how the model output was named.
<code>model</code>	a list containing the regression output of class "mvr", "gbm" or "svm".
<code>x.tr</code>	a matrix containing the transformed spectra.
<code>spectral.transformation</code>	a character naming the spectral transformation.
<code>constituents</code>	a character naming the constituents.
<code>constituents.transformation</code>	a character naming the constituent transformations. Needed for <code>pred.regr</code> .
<code>lambda</code>	a numeric giving the lambda values in case the box-cox transformation was chosen as constituents transformation. Needed for <code>pred.regr</code> .
<code>method</code>	a character naming the used regression method.
<code>cal.samples</code>	a list containing the row names of the calibration samples for each soil constituent.
<code>val.samples</code>	a list containing the row names of the validation samples for each soil constituent.
<code>cal.statistics</code>	a matrix containing the calibration statistics for all constituents. See details.
<code>cal.meas.pre</code>	a data frame containing the calibration set measured and predicted values for all constituents.

<code>val.statistics</code>	a matrix containing the validation statistics for all constituents. See details.
<code>val.meas.pre</code>	a data frame containing the validation set measured and predicted values for all constituents.
<code>cal.pca</code>	a list containing objects of the class "prcomp" for each constituent calibration set. Needed for <code>pred.regr</code> .
<code>mahalanobis</code>	a list containing numeric vectors having the spectral mahalanobis distance of the constituents calibration sets. Needed for <code>pred.regr</code> .
<code>cal.range</code>	a list containing numeric vectors having the ranges of the constituents calibration sets. Needed for <code>pred.regr</code> .
<code>rmsep</code>	a list containing numeric vectors having for each constituent the root median square error of prediction for each validation set sample. See details for further explanation. Needed for <code>pred.regr</code> .
<code>lm</code>	a list containing numeric vectors having for each constituent validation set the fitted values calculated by linear regression of measured against predicted values. Needed for <code>pred.regr</code> .
<code>wavebands</code>	a numeric vector containing the wavebands of x. Needed for <code>pred.regr</code> .
<code>drv</code>	an integer giving the order of derivative. Needed for <code>pred.regr</code> .
<code>bandwidth</code>	an integer defining the smoothing interval in wavebands. Needed for <code>pred.regr</code> .
<code>filter</code>	a character defining the wavelet filter. Needed for <code>pred.regr</code> .
<code>level</code>	a character defining the level of wavelet coefficients extraction. Needed for <code>pred.regr</code> .
<code>output.name</code>	a character string giving the name of the saved csv-file from <code>pred.regr</code> .
<code>predicted.values</code>	a matrix containing the predicted values and its respective confidence interval limits.
<code>method</code>	a character naming the used regression method.
<code>spectral.transformation</code>	a character naming the used spectral transformation method.

Author(s)

Thomas Terhoeven-Urselmans

trans

Spectral transformation

Description

trans transforms spectra either by calculation of (i) derivatives, (ii) continuum removal or (iii) wavelet transform. The used functions are (i) `locpoly` function in "KernSmooth" package, (ii) `chull` and `approx` functions in "KernSmooth" package and (iii) `dwt` function in "wavelets" package, respectively.

Usage

```
trans(raw, tr = "derivative", order = 1, gap = 21)
```

```
## S3 method for class 'trans'
plot(x,...)
```

Arguments

raw	a matrix containing the raw spectra.
tr	a character naming the transformation method. See details.
order	an integer between 0 and 3 giving the order of derivative. The value 0 performs smoothing based on the gap.
gap	an integer between 1 and 30 defining the smoothing interval in wavebands.
x	an object of class "trans".
...	additional arguments.

Details

spec.type uses for spectral transformation (i) the locpoly function in KernSmooth package for derivative calculation, (ii) the chull and approx functions in "KernSmooth" package for continuum removal and (iii) the dwt function in wavelets package for extraction of wavelet coefficients. Experiences showed for wavelet decomposition that the best ratio of prediction performance and sparse spectral representation is reached when all 128 wavelet coefficients from decomposition level three are taken.

Possible options for tr are "derivative", "continuum removed" or "wavelet transformed".

A graph is returned showing the raw and transformed spectra for comparison. Its the same output than from [plot.trans](#).

Value

trans returns a list with class "trans" containing the following components:

raw	a matrix containing the raw spectra.
trans	a matrix containing the transformed spectra.
transformation	a character naming the transformation method.

Author(s)

Thomas Terhoeven-Urselmans

Index

[isric](#), [2](#)

[ken.sto](#), [4](#)

[make.comp](#), [5](#), [9](#)

[pc.space](#), [6](#)

[plot.ken.sto](#), [4](#)

[plot.ken.sto\(ken.sto\)](#), [4](#)

[plot.pc.space](#), [7](#)

[plot.pc.space\(pc.space\)](#), [6](#)

[plot.prco\(prco\)](#), [7](#)

[plot.regr\(regr\)](#), [10](#)

[plot.trans](#), [14](#)

[plot.trans\(trans\)](#), [13](#)

[prco](#), [7](#)

[pred.regr\(regr\)](#), [10](#)

[read.spc](#), [9](#), [11](#)

[regr](#), [10](#)

[soil.spec\(soil.spec-package\)](#), [2](#)

[soil.spec-package](#), [2](#)

[summary.prco\(prco\)](#), [7](#)

[summary.regr\(regr\)](#), [10](#)

[trans](#), [10](#), [13](#)