Package 'eegAnalysis'

February 19, 2015

Version 0.0

Date 2014-04-01

Title Tools for analysis and classification of electroencephalography (EEG) data.

Author Murilo Coutinho Silva (University of Brasilia, Brazil), George Freitas von Borries (University of Brasilia, Brazil)

Maintainer Murilo Coutinho Silva <coutinho.stat@gmail.com>

Depends R (>= 3.0.0), e1071 (>= 1.6), wmtsa (>= 2.0-0) , fields (>= 6.9.1), splus2R (>= 1.2-0)

Description

Package with tools for classification of electroencephalography (EEG) data. Feature extraction techniques such as Fourier Transform and Continuous Wavelet Transform (CWT) are available. Support Vector Machines (SVM) can be used to classify the extracted features. An algorithm using Analysis of Variance (ANOVA), False Discovery Rate (FDR), and SVM is available to feature selection. Additionally, the package contains functions to plot data and features.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-22 15:48:01

R topics documented:

`eegAnalysis-package` *Feature selection and classification of electroencephalography data*

Description

This package consists of a set of tools to classify electroencephalography (EEG) and to successfully reduce the feature space dimension. More specifically, this package contains functions to simulate data (`randEEG`), to train classifiers (`svmEEG`), to classify new data (`classifyEEG`) and to plot data (`plotEEG` and `plotwindows`). Nevertheless, what differentiates this package from others available in the community are the functions to automatically select the best features to use in the classification model (`featureSelection` and `FeatureEEG`).

Details

Package:	eegAnalysis
Type:	Package
Version:	1.0
Date:	2014-04-08
License:	GLP (>=2)

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Bostanov, V. (2004) *BCI Competition 2003 - Data Sets Ib and IIb: Feature Extraction From Event-Related Brain Potentials With the Continuous Wavelet Transform and the t-Value Scalogram.* IEEE transactions on biomedical engineering, V. 51, no. 6.

Brockwell, P.J., Davis, R.A. (2002) *Introduction to Time Series and Forecasting*. 2nd ed. Colorado: Springer. Cap. 4.

Coutinho, M. (2013) *Selecting features for EEG classification and constructions of Brain-Machine Interfaces.* Universidade de Brasilia (UnB), Master dissertation.

Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Stanford: Springer.

Karatzoglou, A., Meyer, D., Hornik, K. (2006) Support Vector Machines in R. *Journal of Statistical Software*. Vol 15, issue 9.

`classifyEEG` *Classifies new data*

Description

Classifies a new sample of EEG data based on an object produced by `svmEEG`.

Usage

```
classifyEEG(y, data)
```

Arguments

`y`	an object produced by `svmEEG`.
`data`	a new data set. Each column represents each channel, and each row represents the signals collected by these channels at the same time. Note that only one recording shall be given, in other words, the number of lines must be equal to the length of each recording in the training phase.

Value

`pred`	a vector with the predicted class and an associated probability.

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Stanford: Springer.

See Also

`svmEEG`, `FeatureEEG`

Examples

```
library(eegAnalysis)

###Simulating the data set.
Sim <- randEEG(n.class=2,n.rec=10,n.signals=50,n.channels = 2,
vars = c(2,1))

### Uncomment the next line to choose your own features
# features<-easyFeatures()
```

```
### Selecting the features
### The selected features may differ because the algorithm
### uses some random functions
### Obs: features="example" is used to be fast. Use features="default"
### or choose your own set of features.
x<-FeatureEEG(Sim$data,Sim$classes.Id,Sim$rec.Id,features="example",
               Alpha=0.05, AlphaCorr=0.9,minacc=0.8,fast=FALSE)



### Calculating the classifier
y<-svmEEG(x)
y$model


### Generating new data to test the classifier
new <- randEEG(n.class=2,n.rec=30,n.signals=50,n.channels = 2,
                      vars = c(2,1))


### Classifying the new data and counting the number of successes
cont = 0
for(i in 1:30)
{
  data<-new$data[which((new$classes.Id==1)&(new$rec.Id==i)),]
  if(classifyEEG(y,data)[2]==1)  cont = cont + 1
}

for(i in 1:30)
{
  data<-new$data[which((new$classes.Id==2)&(new$rec.Id==i)),]
  if(classifyEEG(y,data)[2]==2)  cont = cont + 1
}

### The correct classification rate:
cont/60
```

`easyFeatures` *Choosing Features*

Description

This function helps the user to produce an object of class `Features`. The purpose of this object is to be used in the function `FeatureEEG` as the parameter `features`. It is possible to construct this object "by hand" but it can be a difficult task. It takes some time to choose all the features, so it is recommended to save the object afterwards.

Usage

```
easyFeatures()
```

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Stanford: Springer.

See Also

FeatureEEG

Examples

```
#### This example have to be uncommented to be used

#### Use the function
# features<-easyFeatures()

#### Save your features
# dump("features", file = "example.R")

#### In this case, you can find your features in your current directory
# getwd()

#### Read your features again if you need to:
# source("example.R")
```

FeatureEEG *Automatic Feature Selection*

Description

Select the best features to classify EEG data. This function receives as input a list of features defined by the user using the easyFeatures function. Then, the algorithm will use several statistical tests to search for the the best set of features in terms of classification. This kind of analysis is very useful to reduce the dimensionality of the data, producing much faster and accurate classifiers.

Usage

```
FeatureEEG(data, classes.Id, rec.Id, nselec = "default", features = "default",
Alpha = 0.05, AlphaCorr = 0.05, minacc = 0.7, Nfea = 10, fast = FALSE)
```

Arguments

data	is the data frame containing the EEG signals. The data frame must be organized as follows: each column represents a different channel. Thus, the signals collected by each channel are represented in each row. To identify the class and recording represented by each line, the vectors classes.Id and rec.Id should be given and must have length equal to the number of rows in the data frame.
classes.Id	is a vector with length equal to the number of rows in data. Thus each value in the array identifies the class ID of each signal in each row of the database. For example, let classes.Id = c(rep(1,5),rep(2,5)), this means that the first 5 rows of data represents the class with ID 1 and the lines 6 to 10 represent the class with ID 2. NOTE: Only two classes are allowed.
rec.Id	is a vector with length equal to the number of rows in data. Thus each value in the array identifies the recording ID of each signal in each row of the database. For example, let rec.Id <- c(rep(1,5),rep(2,5)), this means that the first 5 rows of data represents the recording with ID 1 of some class and the lines 6 to 10 represent the recording with ID 2 of some class. rec.Id must be numeric and numerated from 1 to the total number of recordings for each class.
nselec	is a vector of size 2, representing the number of recordings to be randomly chosen within each class to be used as training set of the SVM test in the feature selection algorithm.
features	is a list containing all the types of features to be used. It is easier to produce this list using the easyFeatures function. If features="default" uses a default set of features.
Alpha	a vector with the significance level for each FDR test to be performed. It has to be a number between 0 and 1. The closer to zero, the less features tend to be selected. See details.
AlphaCorr	significance level for the correlation test. It has to be a number between 0 and 1. The closer to zero, the less features tend to be selected. See details.
minacc	minimum accuracy to be achieved by the SVM test. It has to be a number between 0 and 1. The closer to one, the less features tend to be selected. See details.
Nfea	the maximum number of types of features selected by the feature selector algorithm.
fast	If FALSE it uses the SVM implementation of the e1071 package. If TRUE the algorithm uses another implementation optimized for the one dimensional case, usually much faster but with no guarantee of convergence.

Details

The feature selector is a statistical algorithm that performs several tests to select the best set of features to classify a new data set in the future. The feature selection is performed following the steps:

1- The training set is divided in two sets. The number of recordings of each class in each set is defined by the parameter nselec.

2- An analysis of variance is evaluated and a false discovery rate (FDR) test, with significance level given by the parameter `Alpha`, is performed to select the features with the greatest differences between classe. The mean is used to represent each class.

3- For each feature selected, a SVM classifier is trained individually for the first group of recordings defined by the parameter `nselec`.

4- The features are extracted from the second group of recordings and classified. The features with a correct classification rate of at least `minacc` (parameter) are selected.

5- A correlation test is performed to eliminate redundant features with significance level given by `AlphaCorr`.

Value

`output`	This function outputs a list that can be used with the function `svmEEG` to produce a classifier.

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Stanford: Springer.

Coutinho, M. (2013) *Selecting features for EEG classification and constructions of Brain-Machine Interfaces.* Universidade de Brasilia (UnB), Master dissertation.

See Also

`svmEEG`, `easyFeatures`

Examples

```
library(eegAnalysis)

###Simulating the data set.
Sim <- randEEG(n.class=2,n.rec=10,n.signals=50,n.channels = 2,
vars = c(2,1))

### Uncomment the next line to choose your own features
# features<-easyFeatures()



### Selecting the features
### The selected features may differ because the algorithm
### uses some random functions
### Obs: features="example" is used just to be a fast example.
### Use features="default" or choose your own set of features.
x<-FeatureEEG(Sim$data,Sim$classes.Id,Sim$rec.Id,features="example",
               Alpha=0.05, AlphaCorr=0.9,minacc=0.8,fast=FALSE)
```

```
### Calculating the classifier
y<-svmEEG(x)
y$model


### Generating new data to test the classifier
new <- randEEG(n.class=2,n.rec=30,n.signals=50,n.channels = 2,
                      vars = c(2,1))


### Classifying the new data and counting the number of successes
cont = 0
for(i in 1:30)
{
  data<-new$data[which((new$classes.Id==1)&(new$rec.Id==i)),]
  if(classifyEEG(y,data)[2]==1)  cont = cont + 1
}

for(i in 1:30)
{
  data<-new$data[which((new$classes.Id==2)&(new$rec.Id==i)),]
  if(classifyEEG(y,data)[2]==2)  cont = cont + 1
}

### The correct classification rate:
cont/60
```

`featureSelection` *Feature Selection Algorithm*

Description

This algorithm allows the user to use his own features. Thus, the `featureSelection` function selects the best set of the features among the features given by the user. It can be used to select features of any number of classes.

Usage

```
featureSelection(features, classes.Id, alpha = 0.05, alphaCorr = 0.05,
minAcc = 0.7, fast = FALSE, testProp = 0.2)
```

Arguments

`features`	is the data set containing the features. The data frame must be organized as follows: each row represents a different feature, and each columns represents each replication of that feature for each class in the study.

`classes.Id` is a vector with length equal to the number of columns of `data`. Thus, each value in the vector identifies the class ID of each column. For example, let `classes.Id = c(rep(1,5),rep(2,5))`, this means that the first 5 columns of `data` represents the class with ID 1 and the columns 6 to 10 represent the class with ID 2.

`alpha` a vector with the significance level for each FDR test to be performed. It has to be a number between 0 and 1. The closer to zero, the less features tend to be selected. See details.

`alphaCorr` significance level for the correlation test. It has to be a number between 0 and 1. The closer to zero, the less features tend to be selected. See details.

`minAcc` minimum accuracy to be achieved by the SVM test. It has to be a number between 0 and 1. The closer to one, the less features tend to be selected. See details.

`fast` If `FALSE` it uses the SVM implementation of the `e1071` package. If `TRUE` the algorithm uses another implementation optimized for the one dimensional case that tends to be much faster, but it may not converge in some cases.

`testProp` the proportion of replicates of each class used in the test phase of the SVM test. See details.

Details

The feature selector is a statistical algorithm that performs several tests to select the best set of features to classify a new data set in the future. The feature selection is performed the following steps:

1- The training set is divided in two sets. The number of replicates (samples, recordings) of each class in each set is defined by the parameter `testProp`.

2- A analysis of variance is evaluated and a false discovery rate (FDR) test, with significance level given by the parameter `alpha`, is performed to select the features with the greatest differences between classes means.

3- For each feature selected, a SVM classifier is trained individually for the first group of replicates defined randomly by the parameter `testProp`.

4- The features are extracted from the second group of replicates and classified. The features with a correct classification rate of at least `minAcc` (parameter) are selected.

5- A correlation test is performed to eliminate redundant features with significance level given by `alphaCorr`.

Value

`Selected` A vector with the number of the features selected. In other words, each number in this vector represents the number of the row selected from the original data set.

`FDRscore` A vector that represents the score of each selected feature in the FDR test. It is a value between 0 and 1. The closer to 1, the better is the selected feature in terms of this test.

`SVMscore` A vector that represents the score of each selected feature in the SVM test. It is a value between 0 and 1. The closer to 1, the better is the selected feature in terms of this test.

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Stanford: Springer.

Coutinho, M. (2013) *Selecting features for EEG classification and constructions of Brain-Machine Interfaces.* Universidade de Brasilia (UnB), Master dissertation.

Examples

```
library(eegAnalysis)

### Simulating some features

# number of features
n.features = 150
# number of replications of class A
n.classA = 200
# number of replications of class B
n.classB = 250

# initializing the features data frame
features<-mat.or.vec(n.features,n.classA+n.classB)
# initializing the classes.ID vector
classes.Id<-c(rep(0,n.classA),rep(1,n.classB))


# simulating the features with the normal distribution
# note that the higher the number of the row, the higher
# is the gap between the means of the simulated normal distribution
for(i in 1:n.features)
{
  features[i,]<-c(rnorm(n.classA,1,1),rnorm(n.classB,1+i/200,1))
}

### Selecting the features and comparing the speed changing the fast parameter
### NOTE: there may be differences between the selected features due to some
### randomness present in the algorithm.
system.time({
  feat<-featureSelection(features, classes.Id, alpha = 0.05, alphaCorr = 0.05,
                         minAcc = 0.65, fast = FALSE, testProp = 0.2)
})
system.time({
  fastFeat<-featureSelection(features, classes.Id, alpha = 0.05, alphaCorr = 0.05,
                             minAcc = 0.65, fast = TRUE, testProp = 0.2)
})


### Reducing the initial data frame to the selected features
```

```
names(fastFeat)
SelectedFeatures <- features[fastFeat$Selected,]
```

---

plotEEG *Plot EEG data*

Description

This function was designed to do different types of plots of EEG data. Graphs of the original data, of the spectrum, continuous wavelet transform and t-value scalogram of the signals can be plotted. The main idea is to help the user to find nice features to use in his final model.

Usage

```
plotEEG(data, classes.Id, rec.Id, which.classes = "ALL", which.rec = "ALL",
which.channels = "ALL", type = "original", m.a = 1, n.colors = 200,
wavelet = "gaussian2", abs = FALSE, Real = TRUE, variance = 1)
```

Arguments

`data`	is the data frame containing the EEG signals. The data frame must be organized as follows: each column represents a different channel. Thus, the signals collected by each channel are represented in each row. To identify the class and recording represented by each line, the vectors `classes.Id` and `rec.Id` should be given and must have length equal to the number of rows in the data frame.
`classes.Id`	is a vector with length equal to the number of rows in `data`. Thus each value in the array identifies the class ID of each signal in each row of the database. For example, let `classes.Id = c(rep(1,5),rep(2,5))`, this means that the first 5 rows of `data` represents the class with ID 1 and the lines 6 to 10 represent the class with ID 2.
`rec.Id`	is a vector with length equal to the number of rows in `data`. Thus each value in the array identifies the recording ID of each signal in each row of the database. For example, let `rec.Id <- c(rep(1,5),rep(2,5))`, this means that the first 5 rows of `data` represents the recording with ID 1 of some class and the lines 6 to 10 represent the recording with ID 2 of some class. `rec.Id` must be numeric and numerated from 1 to the total number of recordings for each class.
`which.classes`	a vector representing which classes will be plotted. For example, if `which.classes= c(1,3)` then the classes with ID 1 and 3 will be plotted (indicated by the vector `classes.Id`). If `which.classes= "ALL"` then all classes will be plotted. Obs: if `type = "T.pvalue"` then only two classes are allowed.
`which.rec`	a list representing which recordings shall be plotted for each class. For example, if `which.rec= list(c(1,3,4),c(1,2,4))` then the recordings with ID 1,3 and 4 will be plotted for the first class indicated in the vector `which.classes` and the recordings with ID 1,2 and 4 will be plotted for the second class indicated in the vector `which.classes`. If `which.rec="ALL"` then all recordings will be plotted.

which.channels	a vector representing which channels shall be plotted. For example, if which.channels =c(1,3) then the channels 1 and 3 (columns 1 and 3 of data) will be plotted. If which.channels="ALL" then all channels will be plotted.
type	if type="original" then the original recordings will be plotted, if type= "spectrum" then the spectrum of each recording will be plotted, if type="wavelet" then the continuous wavelet transform matrix of the recordings will be plotted, if type="T.pvalue" then the t-value scalogram of the recordings will be plotted (see Bostanov 2004).
m.a	is the moving average parameter, if m.a=1 it is not used. If type= "original" then the moving average is applied for each channel of the original recordings. If type= "spectrum" and m.a is a number, the moving average is applied on the spectrum of the recordings, otherwise, if m.a is a vector of size 2, the first value defines the moving average for the original recordings and the second value defines the moving average for the spectrum. If type= "wavelet" or "T.pvalue", m.a should be a vector of size 2, indicating the dimensions in which the moving average will be applied on the CWT matrix.
n.colors	is the number of colors for the contour plot. Used if type="wavelet" or type= "T.pvalue".
wavelet	if type="wavelet" defines which wavelet is used. See wavCWT for more details.
abs	if TRUE then the absolute value of the continuous wavelet transform matrix is used. Used if type="wavelet" or type="T.pvalue".
Real	if TRUE takes the real part of the continuous wavelet transform matrix, if FALSE takes the imaginary part. Used if wavelet="morlet".
variance	the variance parameter for the continuous wavelet transform. See wavCWT for more details.

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Bostanov, V. (2004) *BCI Competition 2003 - Data Sets Ib and IIb: Feature Extraction From Event-Related Brain Potentials With the Continuous Wavelet Transform and the t-Value Scalogram.* IEEE transactions on biomedical engineering, V. 51, no. 6.

Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Stanford: Springer.

Brockwell, P.J., Davis, R.A. (2002) *Introduction to Time Series and Forecasting*. 2nd ed. Colorado: Springer. Cap. 4.

See Also

wavCWT, spectrum

Examples

```
library(eegAnalysis)

#### Simulating some data
sim<-randEEG()

#### Plot some of the recordings:
plotEEG(sim$data, sim$classes.Id, sim$rec.Id, which.classes = "ALL",
        which.rec=list(c(1,2,3),c(2,3,4)), which.channels=c(1,2,3),
        type = 'original', m.a = 10)


#### Plot the spectrum
plotEEG(sim$data, sim$classes.Id, sim$rec.Id, which.classes = "ALL",
        which.rec="ALL", which.channels=1, type = 'spectrum', m.a = 10)


#### Plot the spectrum
plotEEG(sim$data, sim$classes.Id, sim$rec.Id, which.classes = "ALL",
        which.rec="ALL", which.channels=5, type = 'spectrum', m.a = c(5,20))


#### Plot the continuous wavelet transform
plotEEG(sim$data, sim$classes.Id, sim$rec.Id, which.classes = 1,
        which.rec=list(c(1)), which.channels=c(2), type = 'wavelet',
        wavelet="gaussian2", abs=TRUE,variance=1)


### Plot the T-value scalogram
plotEEG(sim$data, sim$classes.Id, sim$rec.Id, which.classes = "ALL",
        which.rec="ALL", which.channels=c(2), type = 'T.pvalue',
        wavelet="gaussian2", abs=TRUE,variance=10)
```

`plotwindows` *Plot statistics calculated by windows*

Description

This function was designed to do charts of EEG data. Basically, the data (for each channel, class and recording) is divided by windows and a statistical function (as the mean or variance) is applied for all signals in each window. Then, the sequence of values obtained for each window is plotted.

Usage

```
plotwindows(data, classes.Id, rec.Id, which.classes = "ALL",
which.rec = "ALL", which.channels = "ALL", win = 10, stat = "sum",
power = 2, abs = FALSE, log = FALSE, complete = FALSE, mintomax = FALSE)
```

Arguments

data	is the data frame containing the EEG signals. The data frame must be organized as follows: each column represents a different channel. Thus, the signals collected by each channel are represented in each row. To identify the class and recording represented by each line, the vectors classes.Id and rec.Id should be given and must have length equal to the number of rows in the data frame.
classes.Id	is a vector with length equal to the number of rows in data. Thus each value in the array identifies the class ID of each signal in each row of the database. For example, let classes.Id = c(rep(1,5),rep(2,5)), this means that the first 5 rows of data represents the class with ID 1 and the lines 6 to 10 represent the class with ID 2.
rec.Id	is a vector with length equal to the number of rows in data. Thus each value in the array identifies the recording ID of each signal in each row of the database. For example, let rec.Id <- c(rep(1,5),rep(2,5)), this means that the first 5 rows of data represents the recording with ID 1 of some class and the lines 6 to 10 represent the recording with ID 2 of some class. rec.Id must be numeric and numerated from 1 to the total number of recordings for each class.
which.classes	a vector representing which classes will be plotted. For example, if which.classes= c(1,3) then the classes with ID 1 and 3 will be plotted (indicated by the vector classes.Id). If which.classes= "ALL" then all classes will be plotted. Obs: if type = "T.pvalue" then only two classes are allowed.
which.rec	a list representing which recordings shall be plotted for each class. For example, if which.rec= list(c(1,3,4),c(1,2,4)) then the recordings with ID 1,3 and 4 will be plotted for the first class indicated in the vector which.classes and the recordings with ID 1,2 and 4 will be plotted for the second class indicated in the vector which.classes. If which.rec="ALL" then all recordings will be plotted.
which.channels	a vector representing which channels shall be plotted. For example, if which.channels =c(1,3) then the channels 1 and 3 (columns 1 and 3 of data) will be plotted. If which.channels ="ALL" then all channels will be plotted.
win	the window size.
stat	is the statistic applied on the signals of each window. If stat= "sum" then the sum is used. If stat="mean" then the mean is used. If stat="var" then the variance is used. If stat="sd" then the standard deviation is used. If stat="max" then the maximum value is used. If stat="min" then the minimum value is used. If stat="prod" then the product is used. If stat= "median" then the median is used. If stat="geometric" then the geometric mean is used. If stat="harmonic" then the harmonic mean is used.
power	a transformation of the data can be performed before using the chosen statistic. In this case the transformation will be data^power. OBS: watch out for negative values if power is not an integer!
abs	a transformation of the data can be used before using the chosen statistic, in this case, if abs=TRUE, then the transformation will be abs(data)^power.
log	a transformation of the data can be used before using the chosen statistic, in this case, if log=TRUE and abs=TRUE, then the transformation will be log(abs(data) ^power+1). OBS: watch out for negative values if abs= FALSE

`complete` if `complete=FALSE`, then the windows will be completely separated. For example, if `win=10` then the first window will consist of signals 1 to 10, the second window will consist of signals 11 to 20, and so on. If `complete=TRUE`, then some signals will overlap in consecutive windows. For example, if `win=10` then the first window will consist of observations 1 to 10, the second window will consist of observations 2 to 11, the third will consist of observations 3 to 12, and so on.

`mintomax` if `mintomax=FALSE` then the function will plot each obtained value for each window in order. If `mintomax=TRUE` then the function will sort the obtained values for each window and then plot these values from minimum to maximum, and thus, providing a quantile analysis.

Value

`data` Is a data frame with all the statistics computed by the function for each window. Arranged in the same way as the input parameter `data`.

`classes.Id` Is a vector with the same meaning as the input parameter `classes.Id` but related to the output parameter `data`.

`rec.Id` Is a vector with the same meaning as the input parameter `rec.Id` but related to the output parameter `data`.

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Brockwell, P.J., Davis, R.A. (2002) *Introduction to Time Series and Forecasting*. 2nd ed. Colorado: Springer. Cap. 4.

See Also

plotEEG

Examples

```
library(eegAnalysis)


### Simulating a data set
sim<-randEEG()


### Plotting the sum of the signals in windows of size 10
plotwindows(sim$data, sim$classes.Id , sim$rec.Id , which.classes = "ALL", which.rec="ALL",
            which.channels=c(1), win=10, stat="sum", power = 2, abs=FALSE,
            log=FALSE,complete = FALSE, mintomax=FALSE)

### Plotting the sorted sum of the signals in windows of size 10
plotwindows(sim$data, sim$classes.Id , sim$rec.Id , which.classes = "ALL", which.rec="ALL",
```

```
which.channels=c(1), win=10, stat="sum", power = 2, abs=FALSE,
log=FALSE,complete = FALSE, mintomax=TRUE)
```

---

randEEG *EEG simulation*

Description

Creates an object simulating ARIMA random variables. The created object contains data in the format required to use other methods of this package. Makes a simulation similar to EEG data to test the capabilities those methods. NOTE: The only purpose of the simulated data is to test the package features, it is not to be used to study properties of real EEG data!

Usage

```
randEEG(n.classes = 2, n.rec = 10, n.channels = 20, n.signals = 250,
ar="default", ma="default", order="default",vars = c(1, 2))
```

Arguments

`n.classes`	number of different classes.
`n.rec`	number of recordings of each class.
`n.channels`	number of channels in the simulation.
`n.signals`	number of signals, observations or samples for each recording of each channel.
`ar`	is a matrix with AR coefficients of the ARIMA model. Each row contains the AR parameters for each class and each channel. One can make the number of rows in `ar` equal to the number of classes and, in this case, the function uses the same coefficients for all channels of each class. Another option is to choose the number of rows equal to `n.class*n.channels` and, in this case, the simulation uses different ARIMA models for each class and each channel. For example if we have 2 classes and `ar=matrix(c(0.1,0.5,-0.2,0.3,0,0),2,3,byrow=T)` then the first class will be simulated from an AR(3) and the second class will be simulated from an AR(1) with parameter 0.3.
`ma`	is a matrix with MA coefficients, defined the same way as `ar`. See the `ar` parameter for more information.
`order`	is a matrix with the same number of rows of `ar` and `ma` but with 3 columns. Thus, each line forms a vector which is used in the function `arima.sim`. For more information see the parameter `order` in the function `arima.sim`.
`vars`	a vector with length equal to the number of rows in `ar`, `ma` and `order`. Denote the variance for each ARIMA model.

Value

`data`	the simulated data frame. The data frame is organized as follows: each column represents a different channel. Thus, each signal collected by each channel is represented in each row. To identify the class and recording represented by each line, the vectors `classes.Id` and `rec.Id` are given and have length equal to the number of rows in the data frame.
`classes.Id`	the vector indicating the class id of each row of `data`.
`rec.Id`	the vector indicating the recording id of each row of `data`.
`n.classes`	number of different classes.
`n.rec`	number of recordings of each class.
`n.channels`	number of channels in the simulation.
`n.signals`	number of signals, observations or samples for each recording of each channel.
`vars`	a vector with length equal to the number of rows in `ar`, `ma` and `order`. Denote the variance for each ARIMA model.

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Brockwell, P.J., Davis, R.A. (2002) *Introduction to Time Series and Forecasting*. 2nd ed. Colorado: Springer. Cap. 4.

See Also

`plotEEG`, `arima.sim`

Examples

```
library(eegAnalysis)

#### Simulating EEG data
X<-randEEG(n.classes = 2, n.rec = 10, n.channels = 20, n.signals = 250,
        vars = c(1, 2))


#### What do we have here?
names(X)


#### length of classes.Id and rec.Id are equal to the number of rows of data:
length(X$classes.Id)==nrow(X$data)
length(X$rec.Id)==nrow(X$data)


#### We have two classes:
unique(X$classes.Id)
```

```
#### We have 10 recordings for each class:
unique(X$rec.Id[which(X$classes.Id==1)])
unique(X$rec.Id[which(X$classes.Id==2)])


#### We have 20 channels:
ncol(X$data)


#### We have 250 signals for each class, recording and channel:
length(X$data[which(X$classes.Id==1 & X$rec.Id==1),1])
```

---

svmEEG *Support Vector Machines*

Description

svmEEG is used to train a support vector machine classifier of the features selected by the function FeatureEEG. Internally, this function uses the svm function available in the e1071 package. Thus, it is recommended to understand the svm function before using svmEEG.

Usage

```
svmEEG(x, method = "C-classification", scale = TRUE, kernel = "radial",
degree = 3, gamma = if (is.vector(x)) 1 else 1/ncol(x), coef0 = 0,
cost = 1, nu = 0.5, class.weights = NULL, cachesize = 40, tolerance = 0.001,
epsilon = 0.1, shrinking = TRUE, cross = 0, probability = TRUE,
fitted = TRUE, seed = 1L, subset, na.action = na.omit)
```

Arguments

x	the features to be classified. Must be a list of class featureEEG produced by the function FeatureEEG.
method	the method to be used in svm. It has to be a classification machine.
scale	a logical vector indicating the variables to be scaled. If scale is of length 1, the value is recycled as many times as needed. Per default, data are scaled internally to zero mean and unit variance. The center and scale values are returned and used for later predictions. See svm for more details.
kernel	the kernel used in training and predicting. One of: linear, polynomial, radial basis or sigmoid. See svm for more details.
degree	parameter needed for kernel of type polynomial. See svm for more details.
gamma	parameter needed for all kernels except linear. See svm for more details.
coef0	parameter needed for kernels of type polynomial and sigmoid. See svm for more details.

cost	cost of constraints violation (default: 1) - it is the C-constant of the regularization term in the Lagrange formulation. See svm for more details.
nu	parameter needed for nu-classification. See svm for more details.
class.weights	a named vector of weights for the different classes, used for asymmetric class sizes. Not all factor levels have to be supplied (default weight: 1). All components have to be named. See svm for more details.
cachesize	cache memory in MB (default 40). See svm for more details.
tolerance	tolerance of termination criterion (default 0.001). See svm for more details.
epsilon	epsilon in the insensitive-loss function (default: 0.1). See svm for more details.
shrinking	option whether to use the shrinking-heuristics (default: TRUE). See svm for more details.
cross	if a integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification. See svm for more details.
probability	logical indicating whether the model should allow for probability predictions. See svm for more details.
fitted	logical indicating whether the fitted values should be computed and included in the model or not. See svm for more details.
seed	integer seed for libsvm. See svm for more details.
subset	an index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named). See svm for more details.
na.action	A function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named). See svm for more details.

Details

Internally, this function uses the svm function available in the e1071 package.

Value

list	An object to be used in classifyEEG.

Author(s)

Murilo Coutinho Silva (coutinho.stat@gmail.com), George Freitas von Borries

References

Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Stanford: Springer.

Karatzoglou, A., Meyer, D., Hornik, K. (2006) Support Vector Machines in R. *Journal of Statistical Software*. Vol 15, issue 9.

See Also

classifyEEG, FeatureEEG, svm

Examples

```
library(eegAnalysis)

###Simulating the data set.
Sim <- randEEG(n.class=2,n.rec=10,n.signals=50,n.channels = 2,
vars = c(2,1))

### Uncomment the next line to choose your own features
# features<-easyFeatures()


### Selecting the features
### The selected features may differ because the algorithm
### uses some random functions
### Obs: features="example" is used to be fast. Use features="default"
### or choose your own set of features.
x<-FeatureEEG(Sim$data,Sim$classes.Id,Sim$rec.Id,features="example",
               Alpha=0.05, AlphaCorr=0.9,minacc=0.8,fast=FALSE)


### Calculating the classifier
y<-svmEEG(x)
y$model


### Generating new data to test the classifier
new <- randEEG(n.class=2,n.rec=30,n.signals=50,n.channels = 2,
                      vars = c(2,1))


### Classifying the new data and counting the number of successes
cont = 0
for(i in 1:30)
{
  data<-new$data[which((new$classes.Id==1)&(new$rec.Id==i)),]
  if(classifyEEG(y,data)[2]==1)  cont = cont + 1
}

for(i in 1:30)
{
  data<-new$data[which((new$classes.Id==2)&(new$rec.Id==i)),]
  if(classifyEEG(y,data)[2]==2)  cont = cont + 1
}

### The correct classification rate:
cont/60
```

Index