Package 'RHRV'

July 2, 2014

Type Package

Title Heart rate variability analysis of ECG data

Version 4.0

Date 2013-01-18

Author A. Mendez, L. Rodriguez-Linares, A. Otero, C.A. Garcia, X. Vila, M. Lado

Maintainer L. Rodriguez-Linares <leandro@uvigo.es>

URL http://rhrv.r-forge.r-project.org/

Description Methods and tools for performing Heart Rate Variability analysis of ECG records. RHRV allows to import data files containing heartbeat positions in the most broadly used formats; eliminating outliers or spurious points present in the time series with unacceptable physiological values; plotting HRV data and performing time domain, frequency domain and nonlinear HRV analysis.

Depends R (>= 3.0.0), tcltk(>= 2.4.1), tkrplot(>= 0.0-18), waveslim(>= 1.6.4), nonlinearTseries (>= 0.2)

Suggests highlight

VignetteBuilder highlight

Repository CRAN

Repository/R-Forge/Project rhrv

Repository/R-Forge/Revision 122

Repository/R-Forge/DateTimeStamp 2014-01-17 17:45:38

Date/Publication 2014-01-19 18:32:43

NeedsCompilation yes

R topics documented:

`RHRV-package` *RHRV: An R-based software package for the heart rate variability analysis of ECG recordings*

Description

RHRV offers functions for performing power spectral analysis of heart rate data. We will use this package for the study of several diseases, such as obstructive sleep apnoea or chronic obstructive pulmonary disease.

Details

Package:	RHRV
Type:	Package
Version:	3.0.7
Date:	2013-01-18
License:	GPL-2
LazyLoad:	yes

This is a package for developing heart rate variability studies of ECG records. Data are read from an ascii file containing a column with beat positions in seconds. A function is included in order to build this file from an ECG record in WFDB format (visit the site http://www.physionet.org for more information).

Note

An example including all the necessary steps to obtain and to
analyze by episodes the power bands of a wfdb register is
giving below:

##Reading a wfdb register and storing into a data structure:
md = CreateHRVData(Verbose = TRUE)
md = LoadBeatWFDB(md, RecordName = "register_name",

```
RecordPath = "register_path")

##Loading information of episodes of apnea:
md = LoadApneaWFDB(md, RecordName = "register_name",
RecordPath = "register_path", Tag = "APN")

##Generating new episodes before and after previous episodes of
apnea:
md = GenerateEpisodes(md, NewBegFrom = "Beg", NewEndFrom = "Beg",
DispBeg = -600, DispEnd = -120, OldTag = "APN",
NewTag = "PREV_APN")
md = GenerateEpisodes(md, NewBegFrom = "End", NewEndFrom = "End",
DispBeg = 120, DispEnd = 600, OldTag = "APN",
NewTag = "POST_APN")

##Calculating heart rate signal:
md = BuildNIHR(md)

##Filtering heart rate signal:
md = FilterNIHR(md)

##Interpolating heart rate signal:
md = InterpolateNIHR(md)

##Calculating spectrogram and power per band:
md = CreateFreqAnalysis(md)
md = CalculatePowerBand(md, indexFreqAnalysis = 1, size = 120,
shift = 10, sizesp = 1024)

##Plotting power per band, including episodes information:
PlotPowerBand(md, indexFreqAnalysis = 1, hr = TRUE, ymax = 2400000,
ymaxratio = 3, Tag = "all")

##Splitting power per band using episodes before and after
episodes of apnea:
PrevAPN = SplitPowerBandByEpisodes(md, indexFreqAnalysis = 1,
Tag = "PREV_APN")
PostAPN = SplitPowerBandByEpisodes(md, indexFreqAnalysis = 1,
Tag = "POST_APN")

##Performing Student's t-test:
result = t.test(PrevAPN$InEpisodes$ULF, PostAPN$InEpisodes$ULF)
print(result)
```

Author(s)

A. Mendez, L. Rodriguez, A. Otero, C.A. Garcia, X. Vila, M. Lado

Maintainer: Leandro Rodriguez-Linares <leandro@uvigo.es>

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

`AddEpisodes` *Adds new episodes manually*

Description

Adds information of episodes manually, or annotated physiological events, and stores it into the data structure containing the beat positions

Usage

```
AddEpisodes(HRVData, InitTimes, Tags, Durations, Values, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`InitTimes`	Vector containing init times in seconds
`Tags`	Vector containing types of episodes
`Durations`	Vector containing durations in seconds
`Values`	Vector containing numerical values for episodes
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register and new episodes information

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

AnalyzeHRbyEpisodes *Analyzes Heart Rate using episodes information*

Description

Analyzes Heart Rate allowing to evaluate the application of a desired function inside and outside episodes

Usage

```
AnalyzeHRbyEpisodes(HRVData, Tag="", func, ..., verbose=NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
Tag	Type of episode
func	Function to be applied to Heart Rate Data inside and outside episodes
...	optional arguments to func
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns a list with two objects, that is, the values of the application of the selected function inside and outside episodes

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

SplitHRbyEpisodes for splitting in two parts Heart Rate Data using an specific episode type

AnalyzePowerBandsByEpisodes
Analyze power band by episodes

Description

Analyzes the ULF, VLF, LF and HF bands from a given indexFreqAnalysis allowing to evaluate the application of a desired function inside and outside each episode.

Usage

```
AnalyzePowerBandsByEpisodes(HRVData,
  indexFreqAnalysis = length(HRVData$FreqAnalysis),
  Tag = "", verbose = NULL, func, ...)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it.
`indexFreqAnalysis`	Integer value denoting which frequency analysis is going to be analyzed using func. Default: 1
`Tag`	Type of episode
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead
`func`	Function to be applied to each power band inside and outside episodes
`...`	Optional arguments for func.

Value

Returns a list with two objects, that is, the values of the application of the selected function inside ("resultIn") and outside ("resultOut") episodes in the given indexFreqAnalysis. Each of these list has another set of lists: the "ULF", "VLF", "LF" and "HF" lists.

Examples

```
## Not run:
hrv.data = CreateHRVData()
hrv.data = SetVerbose(hrv.data, TRUE)
hrv.data = LoadBeat(hrv.data, fileType = "WFDB", "a03", RecordPath ="beatsFolder/",
                    annotator = "qrs")
                    hrv.data = LoadApneaWFDB(hrv.data, RecordName="a03",Tag="Apnea",
                                             RecordPath="beatsFolder/")
hrv.data = BuildNIHR(hrv.data)
hrv.data = InterpolateNIHR (hrv.data, freqhr = 4)
hrv.data = CreateFreqAnalysis(hrv.data)
hrv.data = CalculatePowerBand( hrv.data , indexFreqAnalysis= 1,
                               type = "wavelet", wavelet = "la8",
                                bandtolerance = 0.01, relative = FALSE)
```

```
results = AnalyzePowerBandsByEpisodes(hrv.data,indexFreqAnalysis=1,
                                      Tag="Apnea",func=mean)
## End(Not run)
```

AvgIntegralCorrelation *Calculates the average of the Integral Correlations*

Description

WARNING: **deprecated** function. The Integral correlation is calculated for every vector of the m-dimensional space, and then the average of all these values is calculated

Usage

```
AvgIntegralCorrelation(HRVData, Data, m, tau, r)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
Data	Portion of HRVData to be analyzed
m	Value of the dimension of the expansion of data
tau	Delay of the expansion of data
r	Distance for calculating correlation

Value

Returns the value of the average of IntegralCorrelations

Note

This function is used in the CalculateApEn function, which is **deprecated**. We suggest the use of the CalculateSampleEntropy function instead of CalculateApEn.

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573 (2008)

See Also

IntegralCorrelation

BuildNIHR *Builds the instantaneous heart rate signal from a beat position array*

Description

The instantaneous heart rate can be defined as the inverse of the time separation between two consecutive heart beats. Once the beats have been identified, and since the only valid values contributing to the heart rate signal are the corresponding to normal beats preceded by other normal beats, a further operation should be performed for the calculation of the instantaneous heart rate.

Usage

```
BuildNIHR(HRVData, verbose=NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register and now associated heart rate instantaneous values also

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573 (2008)

BuildTakensVector *Calculates Takens expanded vectors*

Description

In order to calculate de Fractal Dimension and Approximate Entropy (or others properties of the data) a representation of the data in a space m-dimensional is needed

Usage

```
BuildTakensVector(HRVData, Data, m, tau)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`Data`	Portion of HRVData to be analyzed
`m`	Value of the dimension of the expansion of data
`tau`	Delay of the expansion of data

Value

Returns a matrix with the Expanded Data with N-(m-1)*tau rows (N is the length of the Data to be analyzed) and m columns

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573 (2008)

`CalculateApEn` *Calculates Approximate Entropy*

Description

WARNING: **deprecated** function. Calculates Approximate Entropy as indicated by Pincus

Usage

```
CalculateApEn(HRVData,
              indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
              m = 2, tau = 1,
              r = 0.2, N = 1000, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`indexNonLinearAnalysis`	Reference to the data structure that will contain the non linear analysis
`m`	Value of the dimension of the expansion of data
`tau`	Delay of the expansion of data
`r`	Distance for calculating correlation
`N`	Number of points of the portion of signal to be analyzed
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register and now associated heart rate instantaneous values also, including the value of the Approximate Entropy

Note

This function is **deprecated**. We suggest the use of the `CalculateSampleEntropy` function instead, which is faster.

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573 (2008) S. M. Pincus, "Approximate entropy as a measure of system complexity," Mathematics 88, 2297-2301 (1991)

See Also

`BuildTakensVector` for expand data
`IntegralCorrelation` for correlation calculations
`AvgIntegralCorrelation` for averaging correlation calculations

CalculateCorrDim — *Correlation sum, correlation dimension and generalized correlation dimension (order q >1)*

Description

Functions for estimating the correlation sum and the correlation dimension of the RR time series using phase-space reconstruction

Usage

```
CalculateCorrDim(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  minEmbeddingDim = NULL, maxEmbeddingDim = NULL,
  timeLag = NULL, minRadius, maxRadius,
  pointsRadius = 20, theilerWindow = 100, corrOrder = 2,
  doPlot = TRUE)

EstimateCorrDim(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
```

```
    regressionRange = NULL, useEmbeddings = NULL,
    doPlot = TRUE)

  PlotCorrDim(HRVData,
    indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
    ...)
```

Arguments

`HRVData` Data structure that stores the beats register and information related to it

`indexNonLinearAnalysis` Reference to the data structure that will contain the nonlinear analysis

`minEmbeddingDim` Integer denoting the minimum dimension in which we shall embed the time series

`maxEmbeddingDim` Integer denoting the maximum dimension in which we shall embed the time series. Thus, we shall estimate the correlation dimension between *minEmbeddingDim* and *maxEmbeddingDim*.

`timeLag` Integer denoting the number of time steps that will be use to construct the Takens' vectors.

`minRadius` Minimum distance used to compute the correlation sum C(r)

`maxRadius` Maximum distance used to compute the correlation sum C(r)

`pointsRadius` The number of different radius where we shall estimate C(r). Thus, we will estimate C(r) in pointsRadius between minRadius and maxRadius

`theilerWindow` Integer denoting the Theiler window: Two Takens' vectors must be separated by more than theilerWindow time steps in order to be considered neighbours. By using a Theiler window, we exclude temporally correlated vectors from our estimations.

`corrOrder` Order of the generalized correlation Dimension q. It must be greater than 1 (corrOrder>1). Default, corrOrder=2

`doPlot` Logical value. If TRUE (default), a plot of the correlation sum is shown

`regressionRange` Vector with 2 components denoting the range where the function will perform linear regression

`useEmbeddings` A numeric vector specifying which embedding dimensions should the algorithm use to compute the correlation dimension

`...` Additional plot parameters.

Details

The correlation dimension is the most common measure of the fractal dimensionality of a geometrical object embedded in a phase space. In order to estimate the correlation dimension, the correlation sum is defined over the points from the phase space:

$$C(r) = \{(number\ of\ points\ (x_i, x_j)\ verifying\ that\ distance\ (x_i, x_j) < r\})/N^2$$

However, this estimator is biased when the pairs in the sum are not statistically independent. For example, Taken's vectors that are close in time, are usually close in the phase space due to the non-zero autocorrelation of the original time series. This is solved by using the so-called Theiler window: two Takens' vectors must be separated by, at least, the time steps specified with this window in order to be considered neighbours. By using a Theiler window, we exclude temporally correlated vectors from our estimations.

The correlation dimension is estimated using the slope obtained by performing a linear regression of $\log 10(C(r))\ Vs.\ \log 10(r)$. Since this dimension is supposed to be an invariant of the system, it should not depend on the dimension of the Taken's vectors used to estimate it. Thus, the user should plot $\log 10(C(r))\ Vs.\ \log 10(r)$ for several embedding dimensions when looking for the correlation dimension and, if for some range $\log 10(C(r))$ shows a similar linear behaviour in different embedding dimensions (i.e. parallel slopes), these slopes are an estimate of the correlation dimension. The *estimate* routine allows the user to get always an estimate of the correlation dimension, but the user must check that there is a linear region in the correlation sum over different dimensions. If such a region does not exist, the estimation should be discarded.

Note that the correlation sum C(r) may be interpreted as: $C(r) =< p(r) >$, that is: the mean probability of finding a neighbour in a ball of radius r surrounding a point in the phase space. Thus, it is possible to define a generalization of the correlation dimension by writing:

$$C_q(r) =< p(r)^{(q-1)} >$$

Note that the correlation sum

$$C(r) = C_2(r)$$

It is possible to determine generalized dimensions Dq using the slope obtained by performing a linear regression of $log10(Cq(r))\ Vs.\ (q-1)log10(r)$. The case q=1 leads to the information dimension, that is treated separately in this package. The considerations discussed for the correlation dimension estimate are also valid for these generalized dimensions.

Value

The *CalculateCorrDim* returns the *HRVData* structure containing a *corrDim* object storing the results of the correlation sum (see `corrDim`) of the RR time series.

The *EstimateCorrDim* function estimates the correlation dimension of the RR time series by averaging the slopes of the embedding dimensions specified in the *useEmbeddings* parameter. The slopes are determined by performing a linear regression over the radius' range specified in *regressionRange*.If *doPlot* is TRUE, a graphic of the regression over the data is shown. The results are returned into the *HRVData* structure, under the *NonLinearAnalysis* list.

PlotCorrDim shows two graphics of the correlation integral: a log-log plot of the correlation sum Vs the radius and the local slopes of $log10(C(r))\ Vs\ log10(C(r))$.

Note

This function is based on the `timeLag` function from the nonlinearTseries package.

In order to run *EstimateCorrDim*, it is necessary to have performed the correlation sum before with *ComputeCorrDim*.

Author(s)

Constantino A. Garcia

References

H. Kantz and T. Schreiber: Nonlinear Time series Analysis (Cambridge university press)

See Also

corrDim.

Examples

```
## Not run:
 # ...
 hrv.data = CreateNonLinearAnalysis(hrv.data)
 hrv.data = CalculateCorrDim(hrv.data,indexNonLinearAnalysis=1,
             minEmbeddingDim=2, maxEmbeddingDim=8,timeLag=1,minRadius=1,
             maxRadius=15, pointsRadius=20,theilerWindow=10,
             corrOrder=2,doPlot=FALSE)
 PlotCorrDim(hrv.data,indexNonLinearAnalysis=1)
 hrv.data = EstimateCorrDim(hrv.data,indexNonLinearAnalysis=1,
             useEmbeddings=6:8,regressionRange=c(1,10))

## End(Not run)
```

CalculateDFA *Detrended Fluctuation Analysis*

Description

Performs Detrended Fluctuation Analysis (DFA) on the RR time series, a widely used technique for detecting long range correlations in time series. These functions are able to estimate several scaling exponents from the time series being analyzed. These scaling exponents characterize short or long-term fluctuations, depending of the range used for regression (see details).

Usage

```
CalculateDFA(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  windowSizeRange = c(10, 300), npoints = 25,
  doPlot = TRUE)

EstimateDFA(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  regressionRange = NULL, doPlot = TRUE)

PlotDFA(HRVData,
```

```
    indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
    ...)
```

Arguments

`HRVData` Data structure that stores the beats register and information related to it

`indexNonLinearAnalysis` Reference to the data structure that will contain the nonlinear analysis

`windowSizeRange` Range of values for the windows size that will be used to estimate the fluctuation function. Default: c(10,300).

`npoints` The number of different window sizes that will be used to estimate the Fluctuation function in each zone.

`doPlot` logical value. If TRUE (default value), a plot of the Fluctuation function is shown.

`regressionRange` Vector with 2 components denoting the range where the function will perform linear regression

`...` Additional plot parameters.

Details

The Detrended Fluctuation Analysis (DFA) has become a widely used technique for detecting long range correlations in time series. The DFA procedure may be summarized as follows:

1. Integrate the time series to be analyzed. The time series resulting from the integration will be referred to as the profile.
2. Divide the profile into N non-overlapping segments.
3. Calculate the local trend for each of the segments using least-square regression. Compute the total error for each of the segments.
4. Compute the average of the total error over all segments and take its root square. By repeating the previous steps for several segment sizes (let's denote it by t), we obtain the so-called Fluctuation function $F(t)$.
5. If the data presents long-range power law correlations: $F(t) \sim t^{\alpha}$ and we may estimate using regression.
6. Usually, when plotting $\log(F(t))\ Vs\ log(t)$ we may distinguish two linear regions. By regression them separately, we obtain two scaling exponents, α_1 (characterizing short-term fluctuations) and α_2 (characterizing long-term fluctuations).

Steps 1-4 are performed using the *CalculateDFA* function. In order to obtain a estimate of some scaling exponent, the user must use the *EstimateDFA* function specifying the regression range (window sizes used to detrend the series). α_1 is usually obtained by performing the regression in the $3 < t < 17$ range wheras that α_2 is obtained in the $15 < t < 65$ range (However the F(t) function must be linear in these ranges for obtaining reliable results).

Value

The *CalculateDFA* returns a HRVData structure containing the computations of the Fluctuation function of the RR time series under the *NonLinearAnalysis* list.

The *EstimateDFA* function estimates an scaling exponent of the RR time series by performing a linear regression over the time steps' range specified in *regressionRange*. If *doPlot* is TRUE, a graphic of the regression over the data is shown. In order to run *EstimateDFA*, it is necessary to have performed the Fluctuation function computations before with *ComputeDFA*. The results are returned into the *HRVData* structure, under the *NonLinearAnalysis* list. Since it is possible to estimate several scaling exponents, depending on the regression range used, the scaling exponents are also stored into a list.

PlotDFA shows a graphic of the Fluctuation functions vs window's sizes.

Note

This function is based on the `dfa` function from the nonlinearTseries package.

Author(s)

Constantino A. Garcia

See Also

`dfa`

`CalculateEmbeddingDim` *Estimate the proper embedding dimension for the RR time series*

Description

This function determines the minimum embedding dimension from a scalar time series using the algorithm proposed by L. Cao (see references).

Usage

```
CalculateEmbeddingDim(HRVData, numberPoints = 5000,
  timeLag = 1, maxEmbeddingDim = 15, threshold = 0.95,
  maxRelativeChange = 0.01, doPlot = TRUE)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`numberPoints`	Number of points from the time series that will be used to estimate the embedding dimension. By default, 5000 points are used.
`timeLag`	Time lag used to build the Takens' vectors needed to estimate the embedding dimension (see buildTakens). Default: 1.

`maxEmbeddingDim`
: Maximum possible embedding dimension for the time series. Default: 15.

`threshold`
: Numerical value between 0 and 1. The embedding dimension is estimated using the E1(d) function. E1(d) stops changing when d is greater than or equal to embedding dimension, staying close to 1. This value establishes a threshold for considering that E1(d) is close to 1. Default: 0.95

`maxRelativeChange`
: Maximum relative change in E1(d) with respect to E1(d-1) in order to consider that the E1 function has been stabilized and it will stop changing. Default: 0.01.

`doPlot`
: Logical value. If TRUE (default value), a plot of E1(d) and E2(d) is shown.

Details

The Cao's algorithm uses 2 functions in order to estimate the embedding dimension from a time series: the E1(d) and the E2(d) functions, where d denotes the dimension.

E1(d) stops changing when d is greater than or equal to the embedding dimension, staying close to 1. On the other hand, E2(d) is used to distinguish deterministic signals from stochastic signals. For deterministic signals, there exists some d such that E2(d)!=1. For stochastic signals, E2(d) is approximately 1 for all the values.

Note

The current implementation of this function is fully written in R, based on the `estimateEmbeddingDim` function from the nonlinearTseries package. Thus it requires heavy computations and may be quite slow. The *numberPoints* parameter can be used for controlling the computational burden.

Future versions of the package will solve this issue.

Author(s)

Constantino A. Garcia

References

Cao, L. Practical method for determining the minimum embedding dimension of a scalar time series. Physica D: Nonlinear Phenomena, 110,1, pp. 43-50 (1997).

See Also

`estimateEmbeddingDim`.

CalculateFracDim *Calculates Fractal Dimension*

Description

WARNING: **deprecated** function. Calculates Fractal Dimension as indicated by Pincus

Usage

```
CalculateFracDim(HRVData, indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
    m = 10, tau = 3, Cra = 0.005, Crb = 0.75, N = 1000, verbose=NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
indexNonLinearAnalysis	Reference to the data structure that will contain the non linear analysis
m	Value of the dimension of the expansion of data
tau	Delay of the expansion of data
Cra	Minimum value of correlation for calculating Fractal Dimension
Crb	Maximum value of correlation for calculating Fractal Dimension
N	Number of points of the portion of signal to be analyzed
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register and now associated heart rate instantaneous values also, including the value of the Fractal Dimension

Note

This function is **deprecated**. We suggest the use of the `CalculateCorrDim` function instead, which is faster.

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573 (2008) S. M. Pincus, "Approximate entropy as a measure of system complexity," Mathematics 88, 2297-2301 (1991)

See Also

`CalculateRfromCorrelation` for finding r distance at which the correlation has a certain value

---

`CalculateInfDim` *Information dimension of the RR time series*

Description

Information dimension of the RR time series

Usage

```
CalculateInfDim(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  minEmbeddingDim = NULL, maxEmbeddingDim = NULL,
  timeLag = NULL, minFixedMass = 1e-04,
  maxFixedMass = 0.005, numberFixedMassPoints = 50,
  radius = 1, increasingRadiusFactor = 1.05,
  numberPoints = 500, theilerWindow = 100, doPlot = TRUE)

EstimateInfDim(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  regressionRange = NULL, useEmbeddings = NULL,
  doPlot = TRUE)

PlotInfDim(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  ...)
```

Arguments

`HRVData` Data structure that stores the beats register and information related to it

`indexNonLinearAnalysis`
Reference to the data structure that will contain the nonlinear analysis.

`minEmbeddingDim`
Integer denoting the minimum dimension in which we shall embed the time series.

`maxEmbeddingDim`
Integer denoting the maximum dimension in which we shall embed the time series. Thus, we shall estimate the correlation dimension between *minEmbeddingDim* and *maxEmbeddingDim*.

`timeLag` Integer denoting the number of time steps that will be use to construct the Takens' vectors.

`minFixedMass` Minimum percentage of the total points that the algorithm shall use for the estimation.

`maxFixedMass` Maximum percentage of the total points that the algorithm shall use for the estimation.

`numberFixedMassPoints` The number of different *fixed mass* fractions between *minFixedMass* and *maxFixedMass* that the algorithm will use for estimation.

`radius` Initial radius for searching neighbour points in the phase space. Ideally, it should be small enough so that the fixed mass contained in this radius is slightly greater than the *minFixedMass*. However, whereas the radius is not too large (so that the performance decreases) the choice is not critical.

`increasingRadiusFactor` Numeric value. If no enough neighbours are found within *radius*, the radius is increased by a factor *increasingRadiusFactor* until succesful. Default: sqrt(2) = 1.05.

`numberPoints` Number of reference points that the routine will try to use, saving computation time.

`theilerWindow` Integer denoting the Theiler window: Two Takens' vectors must be separated by more than theilerWindow time steps in order to be considered neighbours. By using a Theiler window, we exclude temporally correlated vectors from our estimations.

`doPlot` Logical value. If TRUE (default), a plot of the correlation sum with q=1 is shown

`regressionRange` Vector with 2 components denoting the range where the function will perform linear regression

`useEmbeddings` A numeric vector specifying which embedding dimensions should the algorithm use to compute the information dimension.

`...` Additional plot parameters.

Details

The information dimension is a particular case of the generalized correlation dimension when setting the order q = 1. It is possible to demonstrate that the information dimension D_1 may be defined as: $D_1 = lim_{r \to 0} < \log p(r) > / \log(r)$. Here, $p(r)$ is the probability of finding a neighbour in a neighbourhood of size r and <> is the mean value. Thus, the information dimension specifies how the average Shannon information scales with the radius r.

In order to estimate D_1, the algorithm looks for the scaling behaviour of the average radius that contains a given portion (a "fixed-mass") of the total points in the phase space. By performing a linear regression of $\log(p)\ Vs.\ \log(< r >)$ (being p the fixed-mass of the total points), an estimate of D_1 is obtained. The user should run the method for different embedding dimensions for checking if D_1 saturates.

The calculations for the information dimension are heavier than those needed for the correlation dimension.

Value

The *CalculateCorrDim* returns the *HRVData* structure containing a *infDim* object storing the results of the correlation sum (see `infDim`) of the RR time series.

The *EstimateInfDim* function estimates the information dimension of the RR time series by averaging the slopes of the correlation sums with q=1. The slopes are determined by performing a linear regression over the radius' range specified in *regressionRange*.If *doPlot* is TRUE, a graphic of the regression over the data is shown. The results are returned into the *HRVData* structure, under the *NonLinearAnalysis* list.

PlotInfDim shows a graphics of the correlation sum with q=1.

Note

In order to run *EstimateInfDim*, it is necessary to have performed the correlation sum before with *ComputeInfDim*.

Author(s)

Constantino A. Garcia

References

H. Kantz and T. Schreiber: Nonlinear Time series Analysis (Cambridge university press)

See Also

`CalculateCorrDim`.

`CalculateMaxLyapunov` *Maximum lyapunov exponent*

Description

Functions for estimating the maximal Lyapunov exponent of the RR time series.

Usage

```
CalculateMaxLyapunov(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  minEmbeddingDim = NULL, maxEmbeddingDim = NULL,
  timeLag = NULL, radius = 2, theilerWindow = 100,
  minNeighs = 5, minRefPoints = 500,
  numberTimeSteps = 20, doPlot = TRUE)

EstimateMaxLyapunov(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  regressionRange = NULL, useEmbeddings = NULL,
  doPlot = TRUE)

PlotMaxLyapunov(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  ...)
```

Arguments

- `HRVData` Data structure that stores the beats register and information related to it
- `indexNonLinearAnalysis` Reference to the data structure that will contain the nonlinear analysis
- `minEmbeddingDim` Integer denoting the minimum dimension in which we shall embed the time series
- `maxEmbeddingDim` Integer denoting the maximum dimension in which we shall embed the time series. Thus, we shall estimate the correlation dimension between *minEmbeddingDim* and *maxEmbeddingDim*.
- `timeLag` Integer denoting the number of time steps that will be use to construct the Takens' vectors. Default: timeLag = 1
- `radius` Maximum distance in which will look for nearby trajectories. Default: radius = 2
- `theilerWindow` Integer denoting the Theiler window: Two Takens' vectors must be separated by more than *theilerWindow* time steps in order to be considered neighbours. By using a Theiler window, temporally correlated vectors are excluded from the estimations. Default: theilerWindow = 100
- `minNeighs` Minimum number of neighbours that a Takens' vector must have to be considered a reference point. Default: minNeighs = 5
- `minRefPoints` Number of reference points that the routine will try to use. The routine stops when it finds *minRefPoints* reference points, saving computation time. Default: minRefPoints = 500
- `numberTimeSteps` Integer denoting the number of time steps in which the algorithm will compute the divergence.
- `doPlot` Logical value. If TRUE (default value), a plot of $S(t)$ Vs t is shown.
- `regressionRange` Vector with 2 components denoting the range where the function will perform linear regression
- `useEmbeddings` A numeric vector specifying which embedding dimensions should the algorithm use to compute the maximal Lyapunov exponent.
- `...` Additional plot parameters.

Details

It is a well-known fact that close trajectories diverge exponentially fast in a chaotic system. The averaged exponent that determines the divergence rate is called the Lyapunov exponent (usually denoted with λ). If $\delta(0)$ is the distance between two Takens' vectors in the embedding.dim-dimensional space, we expect that the distance after a time t between the two trajectories arising from this two vectors fulfills:

$$\delta(n) \sim \delta(0) \cdot exp(\lambda \cdot t)$$

The lyapunov exponent is estimated using the slope obtained by performing a linear regression of $S(t) = \lambda \cdot t \sim log(\delta(t)/\delta(0))$ on t. $S(t)$ will be estimated by averaging the divergence of several reference points.

The user should plot $S(t) Vs t$ when looking for the maximal lyapunov exponent and, if for some temporal range $S(t)$ shows a linear behaviour, its slope is an estimate of the maximal Lyapunov exponent per unit of time. The estimate routine allows the user to get always an estimate of the maximal Lyapunov exponent, but the user must check that there is a linear region in the $S(t) Vs t$. If such a region does not exist, the estimation should be discarded. The user should also run the method for different embedding dimensions for checking if D_1 saturates.

Value

The *CalculateMaxLyapunov* returns a HRVData structure containing the divergence computations of the RR time series under the *NonLinearAnalysis* list.

The *EstimateMaxLyapunov* function estimates the maximum Lyapunov exponent of the RR time series by performing a linear regression over the time steps' range specified in *regressionRange*.If *doPlot* is TRUE, a graphic of the regression over the data is shown. The results are returned into the *HRVData* structure, under the *NonLinearAnalysis* list.

PlotMaxLyapunov shows a graphic of the divergence Vs time

Note

This function is based on the `maxLyapunov` function from the nonlinearTseries package.

In order to run *EstimateMaxLyapunov*, it is necessary to have performed the divergence computations before with *ComputeMaxLyapunov*.

Author(s)

Constantino A. Garcia

References

Eckmann, Jean-Pierre and Kamphorst, S Oliffson and Ruelle, David and Ciliberto, S and others. Liapunov exponents from time series. Physical Review A, 34-6, 4971–4979, (1986).

Rosenstein, Michael T and Collins, James J and De Luca, Carlo J.A practical method for calculating largest Lyapunov exponents from small data sets. Physica D: Nonlinear Phenomena, 65-1, 117–134, (1993).

See Also

`maxLyapunov`

Examples

```
## Not run:
# ...
hrv.data = CreateNonLinearAnalysis(hrv.data)
hrv.data = CalculateMaxLyapunov(hrv.data,indexNonLinearAnalysis=1,
                                 minEmbeddingDim=5,
                                 maxEmbeddingDim = 5,
                                 timeLag=1,radius=10,
                                 theilerWindow=100, doPlot=FALSE)
```

```
PlotMaxLyapunov(hrv.data,indexNonLinearAnalysis=1)
hrv.data = EstimateMaxLyapunov(hrv.data,indexNonLinearAnalysis=1,
                               regressionRange=c(1,10))

## End(Not run)
```

CalculatePowerBand *Calculates power per band*

Description

Calculates power of the heart rate signal at ULF, VLF, LF and HF bands

Usage

```
CalculatePowerBand(HRVData, indexFreqAnalysis=length(HRVData$FreqAnalysis),
size, shift, sizesp = NULL, scale="linear", ULFmin=0, ULFmax=0.03, VLFmin=0.03,
VLFmax=0.05, LFmin=0.05, LFmax=0.15, HFmin=0.15, HFmax=0.4, verbose=NULL,
type="fourier",wavelet="d4",bandtolerance=0.1,relative=FALSE)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
indexFreqAnalysis	Reference to the data structure that will contain the variability analysis
size	Size of window for calculations (seconds)
shift	Displacement of window for calculations (seconds)
sizesp	Points for calculation (zero padding). If the user does not specify it, the function estimates a propper value.
ULFmin	Lower limit ULF band
ULFmax	Upper limit ULF band
VLFmin	Lower limit VLF band
VLFmax	Upper limit VLF band
LFmin	Lower limit LF band
LFmax	Upper limit LF band
HFmin	Lower limit HF band
HFmax	Upper limit HF band
scale	Deprecated argument
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead
type	Type of analysis used to calculate the spectrogram. Possible options are "fourier" or "wavelet"

`wavelet` Mother wavelet used to calculate the spectrogram when a wavelet-based analysis is performed. The available wavelets are: "haar" wavelet; least asymmetric Daubechies wavelets of width 8 ("la8"), 16 ("la16") and 20 ("la20") samples; extremal phase Daubechies of width 4 ("d4"), 6 ("d6"), 8 ("d8") and 16 ("d16") samples; best localized wavelets of width 14 ("bl14") and 20 (" bl20") samples; Fejer-Korovkin wavelets of width 4 ("fk4"), 6 ("fk6"), 8 ("fk8"), 14("fk14") and 22 ("fk22") samples; minimum bandwidth wavelets of width 4 ("mb4"), 8 ("mb8"), 16 ("mb16") and 24 ("mb24"); and the biorthogonal wavelet "bs3.1"

`bandtolerance` Maximum error allowed when a wavelet-based analysis is performed. It can be specified as a absolute or a relative error depending on the "relative" parameter value

`relative` Logic value specifying which kind of bandtolerance shall be used (relative or absolute). The relative tolerance takes into account the width of each of the intervals of interest.

Value

Returns HRVData, the structure that contains beat positions register, associated heart rate instantaneous values, filtered heart rate signal equally spaced, and the analysis structure including spectral power at different bands of the heart rate signal

Note

An example including all the necessary steps to obtain the power
bands of a wfdb register is giving below:

##Reading a wfdb register and storing into a data structure:
md = CreateHRVData(Verbose = TRUE)
md = LoadBeatWFDB(md, RecordName = "register_name",
RecordPath = "register_path")

##Calculating heart rate signal:
md = BuildNIHR(md)

##Filtering heart rate signal:
md = FilterNIHR(md)

##Interpolating heart rate signal:
md = InterpolateNIHR(md)

##Calculating spectrogram and power per band using fourier
analysis:
md = CreateFreqAnalysis(md)
md = CalculatePowerBand(md, indexFreqAnalysis = 1, size = 120,
shift = 10, sizesp = 1024)

```
##Calculating spectrogram and power per band using wavelet analysis:
md = CreateFreqAnalysis(md)
md = CalculatePowerBand(md, indexFreqAnalysis = 2, type="wavelet",
wavelet="la8",bandtolerance=0.0025)
```

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

`CalculateRfromCorrelation` *Calculates ra and rb from Correlation*

Description

WARNING: **deprecated** function. Calculates ra and rb distances that verify that their correlation values are Cra and Crb

Usage

```
CalculateRfromCorrelation(HRVData, Data, m, tau, Cra, Crb)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`Data`	Portion of HRVData to be analyzed
`m`	Value of the dimension of the expansion of data
`tau`	Delay of the expansion of data
`Cra`	Minimum value of correlation for calculating Fractal Dimension
`Crb`	Maximum value of correlation for calculating Fractal Dimension

Value

Returns a 2 by 2 matrix containing ra and rb distance in the first row and their exact correlation values in the second row

Note

This function is used in the `CalculateFracDim` function, which is **deprecated**. We suggest the use of the `CalculateCorrDim` function instead of `CalculateFracDim`.

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011. S. M. Pincus, "Approximate entropy as a measure of system complexity," Mathematics 88, 2297-2301 (1991)

See Also

CalculateFracDim

CalculateSampleEntropy

Sample Entropy (also known as Kolgomorov-Sinai Entropy)

Description

These functions measure the complexity of the RR time series. Large values of the Sample Entropy indicate high complexity whereas that smaller values characterize more regular signals.

Usage

```
CalculateSampleEntropy(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  doPlot = TRUE)

EstimateSampleEntropy(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  regressionRange = NULL, useEmbeddings = NULL,
  doPlot = TRUE)

PlotSampleEntropy(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  ...)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`indexNonLinearAnalysis`	Reference to the data structure that will contain the nonlinear analysis
`doPlot`	Logical value. If TRUE (default), a plot of the correlation sum is shown
`regressionRange`	Vector with 2 components denoting the range where the function will perform linear regression

`useEmbeddings` A numeric vector specifying which embedding dimensions should the algorithm use to compute the sample entropy.

`...` Additional plot parameters.

Details

The sample entropy is computed using:

$$h_q(m,r) = log(C_q(m,r)/C_q(m+1,r))$$

where *m* is the embedding dimension and *r* is the radius of the neighbourhood. When computing the correlation dimensions we use the linear regions from the correlation sums in order to do the estimates. Similarly, the sample entropy $h_q(m,r)$ should not change for both various *m* and *r*.

Value

The *CalculateSampleEntropy* returns a HRVData structure containing the sample entropy computations of the RR time series under the *NonLinearAnalysis* list.

The *EstimateSampleEntropy* function estimates the sample entropy of the RR time series by performing a linear regression over the radius' range specified in *regressionRange*. If *doPlot* is TRUE, a graphic of the regression over the data is shown. In order to run *EstimateSampleEntropy*, it is necessary to have performed the sample entropy computations before with *ComputeSampleEntropy*. The results are returned into the *HRVData* structure, under the *NonLinearAnalysis* list.

PlotSampleEntropy shows a graphic of the sample entropy computations.

Note

In order to run this functions, it is necessary to have used the *CalculateCorrDim* function.

This function is based on the `sampleEntropy` function from the nonlinearTseries package.

Author(s)

Constantino A. Garcia

References

H. Kantz and T. Schreiber: Nonlinear Time series Analysis (Cambridge university press)

See Also

`sampleEntropy`

Examples

```
## Not run:
# ...
hrv.data = CreateNonLinearAnalysis(hrv.data)
hrv.data = CalculateCorrDim(hrv.data,indexNonLinearAnalysis=1,minEmbeddingDim=2,
                            maxEmbeddingDim=8,timeLag=1,minRadius=1,maxRadius=15,
                            pointsRadius=20,theilerWindow=10,corrOrder=2,doPlot=FALSE)
```

```
hrv.data = CalculateSampleEntropy(hrv.data,indexNonLinearAnalysis=1,doPlot=FALSE)
PlotSampleEntropy(hrv.data,indexNonLinearAnalysis=1)
hrv.data = EstimateSampleEntropy(hrv.data,indexNonLinearAnalysis=1,regressionRange=c(6,10))

## End(Not run)
```

`CalculateSpectrogram` *Calculates the spectrogram of a signal*

Description

Calculates the spectrogram of the heart rate signal after filtering and interpolation in a window of a certain size

Usage

```
CalculateSpectrogram(HRVData, size, shift, sizesp = 1024, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`size`	Size of window for calculating spectrogram (seconds)
`shift`	Displacement of window for calculating spectrogram (seconds)
`sizesp`	Points for calculating spectrogram (zero padding)
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns the spectrogram of the heart rate signal

Note

An example including all the necessary steps to obtain the spectrogram
of a wfdb register is giving below:

##Reading a wfdb register and storing into a data structure:
md = CreateHRVData(Verbose = TRUE)
md = LoadBeatWFDB(md, RecordName = "register_name",
RecordPath = "register_path", verbose = TRUE)

##Calculating heart rate signal:
md = BuildNIHR(md)

##Filtering heart rate signal:
md = FilterNIHR(md)

```
##Interpolating heart rate signal:
md = InterpolateNIHR(md)

##Calculating spectrogram:
CalculateSpectrogram(md, size = 120, shift = 10, sizesp = 1024)
```

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

`CalculateTimeLag` *Estimate an appropiate time lag for the Takens' vectors*

Description

Given a time series (timeSeries), an embedding dimension (m) and a time lag (timeLag), the n^{th} Takens' vector is defined as

$$T[n] = timeSeries[n], timeSeries[n + timeLag], ...timeSeries[n + m * timeLag].$$

This function estimates an appropiate time lag by using the autocorrelation function.

Usage

```
CalculateTimeLag(HRVData, method = "first.e.decay",
  value = 1/exp(1), lagMax = NULL, doPlot = TRUE)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`method`	The method that we shall use to estimate the time lag (see the Details section). Available methods are *"first.zero"*, *"first.e.decay"*, *"first.minimum"* and *"first.value"*.
`value`	Numeric value indicating the value that the autocorrelation function must cross in order to select the time lag. It is used only with the "first.value" method.
`lagMax`	Maximum lag at which to calculate the acf. By default, the length of the timeSeries is used.
`doPlot`	Logical value. If TRUE (default value), a plot of the autocorrelation function is shown.

Details

A basic criteria for estimating a proper time lag is based on the following reasoning: if the time lag used to build the Takens' vectors is too small, the coordinates will be too highly temporally correlated and the embedding will tend to cluster around the diagonal in the phase space. If the time lag is chosen too large, the resulting coordinates may be almost uncorrelated and the resulting embedding will be very complicated. Thus, there is a wide variety of methods for estimating an appropiate time lag based on the study of the autocorrelation function of a given time series:

- Select the time lag where the autocorrelation function decays to 0 (first.zero method).
- Select the time lag where the autocorrelation function decays to 1/e (first.e.decay method).
- Select the time lag where the autocorrelation function reaches its first minimum (first.minimum method).
- Select the time lag where the autocorrelation function decays to the value specified by the user (first.value method and value parameter).

Value

The estimated time lag.

Note

If the autocorrelation function does not cross the specifged value, an error is thrown. This may be solved by increasing the lag.max or selecting a higher value to which the autocorrelation function must decay.

This function is based on the `timeLag` function from the nonlinearTseries package.

Author(s)

Constantino A. Garcia

References

H. Kantz and T. Schreiber: Nonlinear Time series Analysis (Cambridge university press)

See Also

`timeLag`.

CreateFreqAnalysis *Creates data analysis structure for frequency analysis calculations*

Description

Creates data analysis structure that stores the information extracted from a variability analysis of heart rate signal and joins it to HRVData as a member of a list

Usage

```
CreateFreqAnalysis(HRVData, verbose=NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register, associated heart rate instantaneous values, filtered heart rate signal equally spaced, and a new analysis structure as a member of a list

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

CreateHRVData

CreateHRVData *Creates data structure for all the calculations*

Description

Creates data structure that stores the beats register and all the information obtained from it

Usage

```
CreateHRVData(Verbose = FALSE)
```

Arguments

Verbose	Boolean argument that allows to specify if the function returns additional information

Value

Returns HRVData, the structure that will contain beat positions register, associated heart rate instantaneous values, filtered heart rate signal equally spaced, and one or more analysis structures

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

CreateFreqAnalysis, CreateTimeAnalysis, CreateNonLinearAnalysis

CreateNonLinearAnalysis *Creates data analysis structure for non linear analysis calculations*

Description

Creates data analysis structure that stores the information extracted from a non linear analysis of ECG signal and joins it to HRVData as a member of a list

Usage

```
CreateNonLinearAnalysis(HRVData, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register, associated heart rate instantaneous values, filtered heart rate signal equally spaced, and a new analysis structure as a member of a list

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

`CreateHRVData`

`CreateTimeAnalysis` *Creates data analysis structure for time analysis calculations*

Description

Creates data analysis structure that stores the information extracted from a time analysis of ECG signal and joins it to HRVData as a member of a list

Usage

```
CreateTimeAnalysis(HRVData, size=300, numofbins=NULL, interval=7.8125, verbose=NULL )
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`size`	Size of window (seconds)
`numofbins`	Number of bins in histogram. If it is not specified, the interval parameter is used (default)
`interval`	Width of bins in histogram (milliseconds)
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register, associated heart rate instantaneous values, filtered heart rate signal equally spaced, and a new analysis structure as a member of a list

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

`CreateHRVData`

`EditNIHR` *Manually edition of non-interpolated instantaneous heart rate*

Description

Plots non-interpolated instantaneous heart rate for manual removing of outliers

Usage

```
EditNIHR(HRVData, scale = 1, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`scale`	Allows scaling for small screens
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns Data, the structure that contains beat positions register, and manually edited associated heart rate instantaneous values

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

FilterNIHR *Artefact filter based in an adaptive threshold*

Description

An algorithm that uses adaptive thresholds for rejecting those beats different from the given threshold more than a certain value. The rule for beat acceptation or rejection is to compare with previous, following and with the updated mean. We apply also a comparison with acceptable physiological values (default values 25 and 200 bpm).

Usage

```
FilterNIHR(HRVData, long=50, last=13, minbpm=25, maxbpm=200, mini=NULL,
maxi=NULL, fixed=NULL, verbose=NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
long	Number of beats to calculate the updated mean
last	Initial threshold
minbpm	Minimum physiologically acceptable value for HR
maxbpm	Maximum physiologically acceptable value for HR
mini	Deprecated argument maintained for compatibility
maxi	Deprecated argument maintained for compatibility
fixed	Deprecated argument maintained for compatibility
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register, associated heart rate instantaneous values also, and now filtered heart rate signal

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

X. Vila, F. Palacios, J. Presedo, M. Fernandez-Delgado, P. Felix, S. Barro, "Time-Frequency analysis of heart-rate variability," IEEE Eng. Med. Biol. Magazine 16, 119-125 (1997) L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573 (2008)

`GenerateEpisodes` *Creates new episodes from old ones*

Description

Creates new episodes, or annotated physiological events, from existing ones and stores them into the data structure containing the beat positions

Usage

```
GenerateEpisodes(HRVData, NewBegFrom, NewEndFrom, DispBeg, DispEnd,
OldTag = "", NewTag = "", verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`NewBegFrom`	Source of new beginning of episodes ("Beg" for indicating the beginning as the beginning of the old episode, "End" for end)
`NewEndFrom`	Source of new end of episodes ("Beg" for indicating the end as the beginning of the old episode, "End" for end)
`DispBeg`	Absolute displacement from the beginning for new episodes in seconds
`DispEnd`	Absolute displacement from the end for new episodes in seconds
`OldTag`	Tag of old episodes
`NewTag`	Tag for new episodes (if empty, copies OldTag)
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register and new episodes information

Note

##Example of arguments for creating episodes displaced one
minute before old ones:
##NewBegFrom = "Beg", NewEndFrom = "End", DispBeg = -60,
DispEnd = -60
##Example of arguments for creating episodes just after previous
ones of 1 minute length:
##NewBegFrom = "End", NewEndFrom = "End", DispBeg = 0,
DispEnd = 60

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

HRVData *HRVData*

Description

HRVData structure containing the occurrence times of the hearbeats of patient suffering from paraplegia and hypertension. The subject from whom the HR was obtained is a patient suffering from paraplegia and hypertension (systolic blood pressure above 200 mmHg). During the recording, he is supplied with prostaglandin E1 (a vasodilator that is rarely employed) and systolic blood pressure fell to 100 mmHg for over an hour. Then, the blood pressure was slowly recovering until 150 mmHg, more or less

Usage

```
data(HRVData)
```

Format

A HRVData structure containing the occurrence times of the heartbeats

See Also

HRVProcessedData

HRVProcessedData *HRVProcessedData*

Description

HRV data containing the heart rhythm of patient suffering from paraplegia and hypertension. The subject from whom the HR was obtained is a patient suffering from paraplegia and hypertension (systolic blood pressure above 200 mmHg). During the recording, he is supplied with prostaglandin E1 (a vasodilator that is rarely employed) and systolic blood pressure fell to 100 mmHg for over an hour. Then, the blood pressure was slowly recovering until 150 mmHg, more or less

Usage

```
data(HRVProcessedData)
```

Format

A HRVData structure containing the interpolated and filtered HR series

See Also

HRVData

IntegralCorrelation *Calculates the Integral Correlation*

Description

WARNING: **deprecated** function. The Integral correlation is calculated for every vector of the m-dimensional space

Usage

```
IntegralCorrelation(HRVData, Data, m, tau, r)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`Data`	Portion of HRVData to be analyzed
`m`	Value of the dimension of the expansion of data
`tau`	Delay of the expansion of data
`r`	Distance for calculating correlation

Value

Returns the value of the average of IntegralCorrelations

Note

This function is used in the CalculateApEn function, which is **deprecated**. We suggest the use of the CalculateSampleEntropy function instead of CalculateApEn.

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573 (2008)

See Also

BuildTakensVector

InterpolateNIHR — *Linear or Spline interpolator for build the sample heart rate signal*

Description

An algorithm to obtain a heart rate signal with equally spaced values at a certain sampling frequency

Usage

```
InterpolateNIHR(HRVData, freqhr = 4, method="linear", verbose=NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
freqhr	Sampling frequency
method	"linear" interpolation or "spline" monotone interpolation
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register, associated heart rate instantaneous values also, and filtered heart rate signal equally spaced

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

`LoadApneaWFDB` *Loads apnea episodes for WFDB record*

Description

Loads the information of apnea episodes and stores it into the data structure containing the beat positions and other related information

Usage

```
LoadApneaWFDB(HRVData, RecordName, RecordPath = ".", Tag = "APNEA",
verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`RecordName`	The WFDB file to be used
`RecordPath`	The path of the WFDB file
`Tag`	to include APNEA episodes
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register and other related information and apnea episodes information

Note

On Windows and Macosx operating systems is necessary to define
a .Renviron file in the user workspace indicating the directory
of the wfdbtools commands. Examples for both OS are given below:

.Renviron on Windows
PATH = "c:\\cygwin\\bin"
DYLD_LIBRARY_PATH = "c:\\cygwin\\lib"

.Renviron on Macosx
PATH = "/opt/local/bin"
DYLD_LIBRARY_PATH = "/opt/local/bin"

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

`LoadBeat` *Builds an array of beats positions from different type of files*

Description

Reads the specific file with data of beat positions and stores the values in a data structure

Usage

```
LoadBeat(fileType, HRVData, Recordname, RecordPath = ".",
annotator = "qrs", scale = 1, datetime = "1/1/1900 0:0:0",
annotationType = "QRS", verbose = NULL)
```

Arguments

`fileType`	The format of the file to be used
`HRVData`	Data structure that stores the beats register and information related to it
`Recordname`	The file to be used
`RecordPath`	The path of the file
`annotator`	The extension of the file, only if we are working with a WFDB file
`scale`	1 if beat positions in seconds or 0.001 if beat positions in milliseconds, only if we are working with a RR or an Ascii file
`datetime`	Date and time (DD/MM/YYYY HH:MM:SS), only if we are working with a RR or an Ascii file
`annotationType`	The type of annotation wished, only if we are working with an EDF+ file
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register

Author(s)

I. Garcia

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

LoadBeatAscii — *Builds an array of beats positions from an ascii file*

Description

Reads an ascii file with data of beat positions and stores the values in a data structure

Usage

```
LoadBeatAscii(HRVData, RecordName, RecordPath=".", scale = 1,
datetime = "1/1/1900 0:0:0", verbose = NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`RecordName`	The Ascii file to be used
`RecordPath`	The path of the file
`scale`	1 if beat positions in seconds or 0.001 if beat positions in milliseconds
`datetime`	Date and time (DD/MM/YYYY HH:MM:SS)
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Loads beats positions into the structure that contains RHRV information. The file containing the heartbeats positions must be a single column file with no headers. Each line should denote the occurrence time of each heartbeat. An example of a valid file could be the following:
0
0.3280001
0.7159996
1.124
1.5
1.88
(...)

Author(s)

I. Garcia

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

`LoadBeatEDFPlus` *Imports data from a record in EDF+ format*

Description

Basically, this algorithm reads the annotation file for the ECG register, and stores the information obtained in a data structure.

Usage

```
LoadBeatEDFPlus(HRVData, RecordName, RecordPath = ".",
annotationType ="QRS", verbose = NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`RecordName`	The EDF+ file to be used
`RecordPath`	The path of the file
`annotationType`	The type of annotation wished
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register

Author(s)

I. Garcia

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

LoadBeatPolar *Imports data from a record in Polar format*

Description

Reads a Polar file with data of beat positions and stores the values in a data structure

Usage

```
LoadBeatPolar(HRVData, RecordName, RecordPath=".", verbose = NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
RecordName	The Polar file to be used
RecordPath	The path of the file
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register

Author(s)

I. Garcia

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

LoadBeatRR *Builds an array of beats positions from an ascii file*

Description

Reads an ascii file containing RR values, i.e. distances between two successive beats.

Usage

```
LoadBeatRR(HRVData, RecordName, RecordPath=".", scale = 1,
datetime = "1/1/1900 0:0:0", verbose = NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`RecordName`	The Ascii file to be used
`RecordPath`	The path of the file
`scale`	1 if beat positions in seconds or 0.001 if beat positions in milliseconds
`datetime`	Date and time (DD/MM/YYYY HH:MM:SS)
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register

Author(s)

I. Garcia

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

`LoadBeatSuunto` *Imports data from a record in Suunto format*

Description

Reads a Suunto file with data of beat positions and stores the values in a data structure

Usage

```
LoadBeatSuunto(HRVData, RecordName, RecordPath = ".", verbose = NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`RecordName`	The Suunto file to be read
`RecordPath`	The path of the file
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register

Author(s)

I. Garcia

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

LoadBeatWFDB — *Imports data from a record in WFDB format*

Description

Basically, this algorithm reads the annotation file for the ECG register, and stores the information obtained in a data structure.

Usage

```
LoadBeatWFDB(HRVData, RecordName, RecordPath = ".", annotator = "qrs",
verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`RecordName`	The WFDB file to be used
`RecordPath`	The path of the file
`annotator`	The extension of the file
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns HRVData, the structure that contains beat positions register

Author(s)

I. Garcia

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

LoadEpisodesAscii *Loads episodes file*

Description

Loads the information of episodes, or annotated physiological events, and stores it into the data structure containing the beat positions

Usage

```
LoadEpisodesAscii(HRVData, FileName, RecordPath=".", Tag="", InitTime="0:0:0",
verbose=NULL,header = TRUE)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
FileName	The episodes file to be used
RecordPath	The path of the file
Tag	Type of episode
InitTime	Time (HH:MM:SS)
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead
header	Logical value. If TRUE, then the first line of the file is skipped. Default: TRUE.

Value

Returns HRVData, the structure that contains beat positions register and episodes information

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

LoadHeaderWFDB *Imports header information from a record in wfdb format*

Description

Reads the header file for the ECG register, and stores the information obtained in a data structure

Usage

```
LoadHeaderWFDB(HRVData, RecordName, RecordPath = ".", verbose=NULL)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
RecordName	The ECG file to be used
RecordPath	The path of the ECG file
verbose	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns Data, the structure that contains beat positions register and data extracted from header file

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

NonlinearityTests *Nonlinearity tests*

Description

Nonlinearity tests

Usage

```
NonlinearityTests(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis))
```

Arguments

`HRVData`	Structure containing the RR time series.
`indexNonLinearAnalysis`	Reference to the data structure that will contain the nonlinear analysis

Details

This function runs a set of nonlinearity tests on the RR time series implemented in other R packages including:

- Teraesvirta's neural metwork test for nonlinearity (`terasvirta.test`).
- White neural metwork test for nonlinearity (`white.test`).
- Keenan's one-degree test for nonlinearity (`Keenan.test`).
- Perform the McLeod-Li test for conditional heteroscedascity (ARCH). (`McLeod.Li.test`).
- Perform the Tsay's test for quadratic nonlinearity in a time series. (`Tsay.test`).
- Perform the Likelihood ratio test for threshold nonlinearity. (`tlrt`).

Value

A *HRVData* structure containing a *NonlinearityTests* field storing the results of each of the tests. The *NonlinearityTests* list is stored under the *NonLinearAnalysis* structure.

`NonLinearNoiseReduction` *Nonlinear noise reduction*

Description

Function for denoising the RR time series using nonlinear analysis techniques.

Usage

```
NonLinearNoiseReduction(HRVData, embeddingDim = NULL,
  radius = NULL, ECGsamplingFreq = NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`embeddingDim`	Integer denoting the dimension in which we shall embed the RR time series.
`radius`	The radius used to looking for neighbours in the phase space (see details). If the radius is not specified, a radius depending on the resolution of the RR time series is used. The resolution depends on the *ECGsamplingFreq* parameter. If the *ECGsamplingFreq* is not supplied, a resolution of 1 ms is assumed.
`ECGsamplingFreq`	The sampling frequency of the ECG from which the RR time series was derived. Although it is not necessary, if it is provided it may improve the noise reduction.

Details

This function takes the RR time series and denoises it. The denoising is achieved by averaging each Takens' vector in an m-dimensional space with his neighbours (time lag=1). Each neighbourhood is specified with balls of a given radius (max norm is used).

Value

A HRVData structure containing the denoised RR time series.

Note

This function is based on the `nonLinearNoiseReduction` function from the nonlinearTseries package.

Author(s)

Constantino A. Garcia

References

H. Kantz and T. Schreiber: Nonlinear Time series Analysis (Cambridge university press)

See Also

`nonLinearNoiseReduction`

PlotHR *Simple plot of interpolated heart rate*

Description

Plots in a simple way the interpolated instantaneous heart rate signal

Usage

```
PlotHR(HRVData, Tag = NULL, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`Tag`	List of tags to specify which episodes, as apnoea or oxygen desaturation, are included in the plot. Tag="all" plots all episodes present in the data.
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

PlotNIHR *Simple plot of non-interpolated heart rate*

Description

Plots in a simple way the non-interpolated instantaneous heart rate signal

Usage

```
PlotNIHR(HRVData, Tag = NULL, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`Tag`	List of tags to specify which episodes, as apnoea or oxygen desaturation, are included in the plot. Tag="all" plots all episodes present in the data.
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

PlotPowerBand *Plots power determined by CalculatePowerBand function*

Description

Plots the power of the heart rate signal at different bands of interest.

Usage

```
PlotPowerBand(HRVData, indexFreqAnalysis = length(HRVData$FreqAnalysis),
normalized = FALSE,
hr = FALSE, ymax = NULL, ymaxratio = NULL, ymaxnorm = 1,
Tag = NULL, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`indexFreqAnalysis`	Reference to the data structure that will contain the variability analysis
`normalized`	Plots normalized powers if TRUE
`hr`	Plots heart rate signal if TRUE
`ymax`	Maximum value for y axis (unnormalized plots)
`ymaxratio`	Maximum value for y axis in LF/HF band (normalized and unnormalized plots)
`ymaxnorm`	Maximum value for y axis (normalized plots)
`Tag`	Argument that allows to specify if episodes contained in Data are represented by means of coloured boxes, for example apnoea or oxygen desaturation, "ALL" for all episodes
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Note

An example including all the necessary steps to obtain the power
bands of a wfdb register is given below:

##Reading a wfdb register and storing into a data structure:
md = CreateHRVData(Verbose = TRUE)
md = LoadBeatWFDB(md, RecordName = "register_name",
RecordPath = "register_path")

##Calculating heart rate signal:
md = BuildNIHR(md)

##Filtering heart rate signal:
md = FilterNIHR(md)

##Interpolating heart rate signal:
md = InterpolateNIHR(md)

##Calculating spectrogram and power per band:
md = CreateFreqAnalysis(md)
md = CalculatePowerBand(md, indexFreqAnalysis = 1, size = 120,
shift = 10, sizesp = 1024)

Plotting Power per Band
PlotPowerBand(md, hr = TRUE, ymax = 700000, ymaxratio = 4)

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, L., A.J. Mendez, M.J. Lado, D.N. Olivieri, X.A. Vila, and I. Gomez-Conde, "An open source tool for heart rate variability spectral analysis", Computer Methods and Programs in Biomedicine 103(1):39-50, july 2011.

See Also

`CalculatePowerBand` for power calculation

`PlotSpectrogram` *Calculates and Plots spectrogram*

Description

Plots spectrogram of the heart rate signal as calculated by CalculateSpectrogram() function

Usage

```
PlotSpectrogram(HRVData, size, shift, sizesp=NULL, freqRange = NULL,
                scale="linear", verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`size`	Size of window for calculating spectrogram (seconds)
`shift`	Displacement of window for calculating spectrogram (seconds)
`sizesp`	Points for calculation (zero padding). If the user does not specify it, the function estimates a propper value.
`freqRange`	Vector with two components specifying the frequency range that the program should plot. If the user does not specify it, the function uses the whole frequency range.
`scale`	Scale used to plot spectrogram, linear or logarithmic
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Note

An example including all the necessary steps to obtain the power
bands of a wfdb register is giving below:

##Reading a wfdb register and storing into a data structure:
md = CreateHRVData(Verbose = TRUE)
md = LoadBeatWFDB(md, RecordName = "register_name",
RecordPath = "register_path")

```
##Calculating heart rate signal:
md = BuildNIHR(md)

##Filtering heart rate signal:
md = FilterNIHR(md)

##Interpolating heart rate signal:
md = InterpolateNIHR(md)

##Calculating and Plotting Spectrogram
PlotSpectrogram(md, size = 120, shift = 10, sizesp = 1024)
```

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

`CalculateSpectrogram` for spectrogram calculation

`PoincarePlot` *Poincare Plot*

Description

The Poincare plot is a graphical representation of the dependance between successive RR intervals obtained by plotting the $RR_{j+\tau}$ as a function of RR_j. This dependance #' is often quantified by fitting an ellipse to the plot. In this way, two parameters are obtained: SD_1 and SD_2. SD_1 characterizes short-term variability whereas that SD_2 characterizes long-term variability.

Usage

```
PoincarePlot(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  timeLag = 1, confidenceEstimation = FALSE,
  confidence = 0.95, doPlot = FALSE,
  main = "Poincare plot", xlab = "RR[n]",
  ylab = paste(sep = "", "RR[n+", timeLag, "]"), ...)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
indexNonLinearAnalysis	Reference to the data structure that will contain the nonlinear analysis
timeLag	Integer denoting the number of time steps that will be use to construct the dependance relation: $RR_{j+timeLag}$ as a function of RR_j.
confidenceEstimation	Logical value. If TRUE, the ellipse-like confidence region for the two dimensional plot is used for fitting the ellipse and computing the SD_1 and SD_2 parameters (see details). Default: FALSE.
confidence	The confidence for computing the confidence region if *confidenceEstimation = TRUE*.
doPlot	Logical value. If TRUE (default), the PoincarePlot is shown.
xlab	A title for the x axis.
ylab	A title for the y axis.
main	An overall title for the Poincare plot.
...	Additional parameters for the Poincare plot figure.

Details

In the HRV literature, when *timeLag = 1*, the SD_1 and SD_2 parameters are computed using time domain measures. This is the default approach in this function if *timeLag=1*. However, sometimes the ellipse that is fitted using this approach is too small. This function also allows the user to fit a ellipse by estimating a confidence region (by setting *confidenceEstimation = TRUE*). If *timeLag > 1*, the confidence region approach is always used.

Value

A *HRVData* structure containing a *PoincarePlot* field storing the SD_1 and SD_2 parameters. The *PoincarePlot* field is stored under the *NonLinearAnalysis* list.

ReadFromFile *Reads data structure from file*

Description

Reads the data structure containing beat positions and all derived calculations from file

Usage

```
ReadFromFile(name, verbose=FALSE)
```

Arguments

`name`	The name of the file to be used (without the .hrv extension)
`verbose`	Logical value that sets the verbose mode on or off

Value

Returns the HRVData structure previously stored in the given file.

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

`RecurrencePlot` *Recurrence Plot*

Description

Plot the recurrence matrix of the RR time series.

Usage

```
RecurrencePlot(HRVData, numberPoints = 1000,
  embeddingDim = NULL, timeLag = NULL, radius = 1)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`numberPoints`	Number of points from the RR time series to be used in the RQA computation. Default: 1000 heartbeats.
`embeddingDim`	Integer denoting the dimension in which we shall embed the RR time series.
`timeLag`	Integer denoting the number of time steps that will be use to construct the Takens' vectors.
`radius`	Maximum distance between two phase-space points to be considered a recurrence.

Details

WARNING: This function is computationally very expensive. Use with caution.

Note

This function is based on the recurrencePlot function from the nonlinearTseries package.

Author(s)

Constantino A. Garcia

References

Zbilut, J. P. and C. L. Webber. Recurrence quantification analysis. Wiley Encyclopedia of Biomedical Engineering (2006).

See Also

recurrencePlot, RQA

RQA *Recurrence Quantification Analysis (RQA)*

Description

The Recurrence Quantification Analysis (RQA) is an advanced technique for the nonlinear analysis that allows to quantify the number and duration of the recurrences in the phase space. This function computes the RQA of the RR time series.

Usage

```
RQA(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  numberPoints = NULL, embeddingDim = NULL,
  timeLag = NULL, radius = 1, lmin = 2, vmin = 2,
  distanceToBorder = 2, doPlot = FALSE)
```

Arguments

HRVData	Data structure that stores the beats register and information related to it
indexNonLinearAnalysis	Reference to the data structure that will contain the nonlinear analysis
numberPoints	Number of points from the RR time series to be used in the RQA computation. If the number of points is not specified, the whole RR time series is used.
embeddingDim	Integer denoting the dimension in which we shall embed the RR time series.
timeLag	Integer denoting the number of time steps that will be use to construct the Takens' vectors.
radius	Maximum distance between two phase-space points to be considered a recurrence.

`lmin` Minimal length of a diagonal line to be considered in the RQA. Default *lmin* = 2.

`vmin` Minimal length of a vertical line to be considered in the RQA. Default *vmin* = 2.

`distanceToBorder` In order to avoid border effects, the *distanceToBorder* points near the border of the recurrence matrix are ignored when computing the RQA parameters. Default, *distanceToBorder* = 2.

`doPlot` Logical. If TRUE, the recurrence plot is shown. However, plotting the recurrence matrix is computationally expensive. Use with caution.

Value

A HRVData structure that stores an *rqa* field under the NonLinearAnalysis list. The *rqa* field consist of a list with the most important RQA parameters:

- *REC*: Recurrence. Percentage of recurrence points in a Recurrence Plot.
- *DET*: Determinism. Percentage of recurrence points that form diagonal lines.
- *LAM*: Percentage of recurrent points that form vertical lines.
- *RATIO*: Ratio between *DET* and *RR*.
- *Lmax*: Length of the longest diagonal line.
- *Lmean*: Mean length of the diagonal lines. The main diagonal is not taken into account.
- *DIV*: Inverse of *Lmax*.
- *Vmax*: Longest vertical line.
- *Vmean*: Average length of the vertical lines. This parameter is also referred to as the Trapping time.
- *ENTR*: Shannon entropy of the diagonal line lengths distribution
- *TREND*: Trend of the number of recurrent points depending on the distance to the main diagonal
- *diagonalHistogram*: Histogram of the length of the diagonals.
- *recurrenceRate*: Number of recurrent points depending on the distance to the main diagonal.

Note

This function is based on the `rqa` function from the nonlinearTseries package.

References

Zbilut, J. P. and C. L. Webber. Recurrence quantification analysis. Wiley Encyclopedia of Biomedical Engineering (2006).

See Also

`rqa`, `RecurrencePlot`

`SetVerbose` *Sets verbose mode on or off*

Description

Sets verbose mode on or off, verbose is a boolean component of the data structure HRVData that allows to specify if all the functions return additional information

Usage

```
SetVerbose(HRVData, Verbose)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`Verbose`	Boolean argument that allows to specify if the function returns additional information

Value

Returns HRVData, the structure that will contain beat positions register, associated heart rate instantaneous values, filtered heart rate signal equally spaced, and one or more analysis structures

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

`SplitHRbyEpisodes` *Splits Heart Rate Data using Episodes information*

Description

Splits Heart Rate Data in two parts using an specific episode type: data inside episodes and data outside episodes

Usage

```
SplitHRbyEpisodes(HRVData, Tag = "", verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`Tag`	Type of episode
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns a list with two vectors that is, the values of Heart Rate Data inside and outside episodes

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

`AnalyzeHRbyEpisodes` for processing Heart Rate Data using an specific episode type

---

`SplitPowerBandByEpisodes` *Splits Power Per Band using Episodes information*

Description

Splits Power per Band in two lists using an specific episode type: data inside episodes and data outside episodes

Usage

```
SplitPowerBandByEpisodes(HRVData, indexFreqAnalysis  =
length(HRVData$FreqAnalysis), Tag = "",
verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`indexFreqAnalysis`	Reference to the data structure that will contain the variability analysis
`Tag`	Type of episode
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Value

Returns a list with two lists: InEpisodes and OutEpisodes, both lists include ULF, VLF, LF and HF bands

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

See Also

`CalculatePowerBand` for power calculation

`SurrogateTest` *Surrogate data testing*

Description

Surrogate data testing

Usage

```
SurrogateTest(HRVData,
  indexNonLinearAnalysis = length(HRVData$NonLinearAnalysis),
  significance = 0.05, doPlot = TRUE, useFunction, ...)
```

Arguments

`HRVData` Structure containing the RR time series.

`indexNonLinearAnalysis` Reference to the data structure that will contain the nonlinear analysis

`significance` Significance of the test

`doPlot` Logical value. If TRUE, a graphical representation of the statistic value for both surrogates and original data is shown.

`useFunction` The function that computes the discriminating statistic that shall be used for testing.

`...` Additional arguments for the *useFunction* function.

Details

This function tests the null hypothesis (H0) stating that the series describes a linear process. The test is performed by generating several surrogate data according to H0 and comparing the values of a discriminating statistic between both original data and the surrogate data. If the value of the statistic is significantly different for the original series than for the surrogate set, the null hypothesis is rejected and nonlinearity assumed. The surrogate data is generated by using a phase randomization procedure.

Value

A *HRVData* structure containing a *SurrogateTest* field storing the statistics computed for the set (*surrogates.statistics* field) and the RR time series (*data.statistic field*). The *SurrogateTest* list is stored under the *NonLinearAnalysis* structure.

Author(s)

Constantino A. Garcia

References

SCHREIBER, Thomas; SCHMITZ, Andreas. Surrogate time series. Physica D: Nonlinear Phenomena, 2000, vol. 142, no 3, p. 346-382.

`WriteToFile` *Writes data structure to a file*

Description

Writes the data structure containing beat positions and all derived calculations to a file

Usage

```
WriteToFile(HRVData, name, overwrite = TRUE, verbose=NULL)
```

Arguments

`HRVData`	Data structure that stores the beats register and information related to it
`name`	The name of the file to be used
`overwrite`	Boolean argument for indicating what to do if the file already exists
`verbose`	Deprecated argument maintained for compatibility, use SetVerbose() instead

Author(s)

M. Lado, A. Mendez, D. Olivieri, L. Rodriguez, X. Vila

References

L. Rodriguez-Linares, X. Vila, A. Mendez, M. Lado, D. Olivieri, "RHRV: An R-based software package for heart rate variability analysis of ECG recordings," 3rd Iberian Conference in Systems and Information Technologies (CISTI 2008), Proceedings I, 565-573, ISBN: 978-84-612-4476-8 (2008)

Index