Package ‘QRSdetect’

February 11, 2019

Type Package

Title Detect QRS Complexes From The Electrocardiogran

Version 0.1.0.9005

Date 2019-02-11

Description A set of functions for detecting QRS complexes from the ECG. It includes functions to work with Physionet files, and functions to test the accuracy of the implemented algorithms. A Shiny app
is provided that allows correcting any errors in QRS detection.

License GPL-3

Encoding UTF-8

LazyData true

Imports signal (>= 0.7-6)

Depends R (>= 3.2.0)

URL http://github.com/gjmvanboxtel/QRSdetect

BugReports http://github.com/gjmvanboxtel/QRSdetect/issues

RoxygenNote 6.1.1

R topics documented:

afonso *Afonso et al. QRS detection*

Description

Detect R peaks of the QRS complex in a raw ECG record, based on the filter-bank algorithm proposed by Afonso et al. (1999)

Usage

```
afonso(ecg, fs)
```

Arguments

`ecg`	The input single-channel input vector (raw ECG)
`fs`	The frequency in Hz with which the ecg was sampled

Details

The Afonso et al. algorithm uses filter banks (polyphase implementation) and determines candidate R-peaks on downsampled signals in different frequency bands. Initially, a large number of false positives are generated. Then, logic is added to decrease the number of false positives while maintaining a low number of false negatives. This method is currently one of the most accurate available (> 99.5 (because the logic is applied on downsampled signals).

The present R implementation is based on the Matlab/Octave version named nqrsdetect.m, Copyright (C) 2006 by Rupert Ortner, retrieved from the internet pages maintained by Alois Schloegl (http://pub.ist.ac.at/~schloegl/). A few improvements and minor bug fixes were made, as well as comments added.

Value

Numeric array containing the indices (sample numbers) at which the fiducial R-peaks were found

Author(s)

Geert van Boxtel

References

Afonso, V.X., Tompkins, W.J., Nguyen, T.Q., & Luo, S. (1999). ECG beat detection using filter banks. IEEE Transactions on Biomedical Engineering, 46(2), 192-202. DOI: 10.1109/10.740882

Examples

```
data(rec100)
fs <- 360
pks <- afonso(rec100$MLII, fs)

## Not run:
# plot first 5 seconds of data
N <- 5 * fs
plot (rec100$time[1:N], rec100$MLII[1:N], type = "l", main = "MIT-BIH database, record 100",
      xlab = "Time (s)", ylab = "Amplitude (mV)")
points (pks[which(pks<=N)]/fs, ecg$MLII[pks[which(pks<=N)]], col="red")

## End(Not run)
```

`bahoura` *Bahoura et al. QRS detection*

Description

Detect R peaks of the QRS complex in a raw ECG record, based on the algorithm proposed by Bahoura, Hassani & Hubin (1997)

Usage

```
bahoura(ecg, fs)
```

Arguments

`ecg`	The input single-channel input vector (raw ECG)
`fs`	The frequency in Hz with which the ecg was sampled

Details

This function attempts to detect ECG fiducial points in a single-channel electrocardiogram signal using the algorithm proposed by Bahoura, Hassani & Hubin (1997), which is a simplified version of the method proposed by Li, Zheng & Tai (1995).

Value

Numeric array containing the indices (sample numbers) at which the fiducial R-peaks were found

Author(s)

Geert van Boxtel

References

Bahoura, M., Hassani, M., & Hubin, M. (1997). DSP implementation of wavelet transform for real time ECG wave forms detection and heart rate analysis. Computer Methods and Programs in Biomedicine, 52, 35-44. DOI: 10.1016/S0169-2607(97)01780-X.
Li, C., Zheng, C., & Tai, C. (1995). Detection of ECG Characteristic Points Using Wavelet Transforms. IEEE Transactions on Biomedical Engineering, 42(1), 21-28. DOI: 10.1109/10.362922.

Examples

```
data(rec100)
fs <- 360
pks <- bahoura(rec100$MLII, fs)

## Not run:
# plot first 5 seconds of data
N <- 5 * fs
plot (rec100$time[1:N], rec100$MLII[1:N], type = "l", main = "MIT-BIH database, record 100",
      xlab = "Time (s)", ylab = "Amplitude (mV)")
points (pks[which(pks<=N)]/fs, rec100$MLII[pks[which(pks<=N)]], col="red")

## End(Not run)
```

`findpeaks` *Find local maxima*

Description

Find local maxima (peaks) in an input signal vector.

Usage

```
findpeaks(data, minh = .Machine$double.eps, mind = 1, minw = 1,
  maxw = Inf, ds = FALSE)
```

Arguments

`data`	The input signal vector.
`minh`	Minimum peak height (non-negative scalar). Only peaks that exceed this value will be returned. For data taking positive and negative values use the option `ds`. Default: machine precision (.Machine$double.eps).
`mind`	Minimum separation between peaks (positive integer). Peaks separated by less than this distance are considered a single peak. This distance is also used to fit a second order polynomial to the peaks to estimate their width, (see `details`, therefore it acts as a smoothing parameter. Default value 1.
`minw`	Minimum width of peaks (positive integer). Default value 1.
`maxw`	Maximum width of peaks (positive integer). Default value `Inf`.

`ds` Double-sided. Tells the function that data takes positive and negative values. The base-line for the peaks is taken as the mean value of the function. This is equivalent as passing the absolute value of the data after removing the mean.

Details

This function searches for peaks in a signal vector. Peaks of a positive array of data are defined as local maxima. For double-sided data, they are maxima of the positive part and minima of the negative part. The function provides various options to search for peaks in noisy data, such specifying a minimum peak height (`minh`), a minimum distance between peaks (`mind`), and a minimum or maximum width of the peaks (`minw` and `maxw`).

The width of the peaks is estimated using a parabola fitted to the neighborhood of each peak. The width is caulculated with the formula $a * (width - x0)^2 = 1$, where a is the the concavity of the parabola and $x0$ its vertex. The neighborhood size is equal to the value of `mind`.

Value

When called with `minw = 0` and `maxw = Inf`, this function returns a `list` containing two values:

`$pks`: array containing the value of `data` at the peaks

`$idx`: array containing the peak indices

When called with either `minw > 0` or `maxw < Inf`, then the returned `list` contains these additional variables:

`$parabol`: a `list` containing additional information about the parabol fitted to the peak. The `list` `$pp` contains the coefficients of the 2nd degree polynomial (`a`, `b`, and `b2`), and `$x` the extrema of the interval where it was fitted (`$from`, `to`).

`$height`: The estimated height of the returned peaks (in units of `data`).

`$baseline`: The height at which the roots of the returned peaks were calculated (in units of `data`).

`$roots`: The abscissa values (in index units) at which the parabola fitted to each of the returned peaks realizes its width.

Author(s)

Geert van Boxtel

References

Octave Signal package

Examples

```
# Example 1: Finding the peaks of smooth data is not a big deal

t <- 2*pi*seq(0,1,length=1024)
y <- sin(3.14*t) + 0.5*cos(6.09*t) + 0.1*sin(10.11*t+1/6) + 0.1*sin(15.3*t+1/3)

data1 <- abs(y) # Positive values
```

```
peaks1 <- findpeaks(data1)

data2 <- y # Double-sided
peaks2 <- findpeaks(data2, ds=TRUE)
peaks3 <- findpeaks (data2, ds=TRUE, minh=0.5)

## Not run:
  op <- par(mfrow=c(1,2))
  plot(t,data1,type="l", xlab="", ylab="")
  points (t[peaks1$idx],peaks1$pks,col="red", pch=1)
  plot(t,data2,type="l", xlab="", ylab="")
  points (t[peaks2$idx],peaks2$pks,col="red", pch=1)
  points (t[peaks3$idx],peaks3$pks,col="red", pch=4)
  legend ("topleft", '0: >2*sd, x: >0.5', bty="n", text.col="red")
  par (op)
## End(Not run)

# Example 2: Noisy data may need tuning of the parameters. In this example,
# "mind" is used as a smoother of the peaks.

t <- 2*pi*seq(0,1,length=1024)
y <- sin(3.14*t) + 0.5*cos(6.09*t) + 0.1*sin(10.11*t+1/6) + 0.1*sin(15.3*t+1/3)
data <- abs(y + 0.1*rnorm(length(y),1)); # Positive values + noise
peaks1 <- findpeaks(data, minh=1)
dt <- t[2]-t[1]
peaks2 <- findpeaks(data, minh=1, mind=round(0.5/dt))

## Not run:
  op <- par(mfrow=c(1,2))
  plot(t, data, type="l", xlab="", ylab="")
  points (t[peaks1$idx],peaks1$pks,col="red", pch=1)
  plot(t, data, type="l", xlab="", ylab="")
  points (t[peaks2$idx],peaks2$pks,col="red", pch=1)
  par (op)
## End(Not run)
```

`pantompkins` *Pan & Tompkins QRS detection*

Description

Detect R peaks of the QRS complex in a raw ECG record, based on the algorithm proposed by Pan & Tompkins (1985)

Usage

```
pantompkins(ecg, fs)
```

Arguments

`ecg`	The input single-channel input vector (raw ECG)
`fs`	The frequency in Hz with which the ecg was sampled

Details

This function attempts to detect ECG fiducial points in a single-channel electrocardiogram signal using the algorithm proposed by Pan & Tompkins (1985). The algorithm was originally developed to be implemented in hardware. Here a software implementation with digital filters, and some improvements:

1. Filter the signal 5-15 Hz (Butterworth); pre- and postpad the signal with 1 s of reversed data to ramp up and down the filter
2. Compute the derivative
3. Square the derivative
4. Apply a moving average (0.15 s filter length)
5. Search for the peak using findpeaks (mind = 200, minh = 2*sd of moving average), lookup local maximum in signal

Value

Numeric array containing the indices (sample numbers) at which the fiducial R-peaks were found

Author(s)

Geert van Boxtel

References

Pan, J., & Tompkins, W.J. (1985). A real-time QRS detection algorithm. IEEE Transactions on Biomedical Engineering, Vol. BME-32, Issue 3, 230-236, DOI: 10.1109/TBME.1985.325532

Examples

```
data(rec100)
fs <- 360
pks <- pantompkins(rec100$MLII, fs)

## Not run:
# plot first 5 seconds of data
N <- 5 * fs
plot (rec100$time[1:N], rec100$MLII[1:N], type = "l", main = "MIT-BIH database, record 100",
      xlab = "Time (s)", ylab = "Amplitude (mV)")
points (pks[which(pks<=N)]/fs, ecg$MLII[pks[which(pks<=N)]], col="red")

## End(Not run)
```

QRSdetect *QRSdetect*

Description

QRSdetect

Details

Here is the general help for the QRSdetect package (to be filled in)

QRSdetect-internal *QRSdetect-internal*

Description

Internal or barely commented functions not exported from the namespace.

Usage

```
is.scalar(x)

is.posscal(x)

is.wholenumber(x, tol = .Machine$double.eps^0.5)

RMS(x)
```

Arguments

`x`	object
`tol`	tolerance

Author(s)

Geert van Boxtel

`rec100` *Electrocardiogram data*

Description

A data set containing the raw electrocardiogram data from MIT-BIH database record 100, The data were collected with a sampling rate of 360 Hz and digitized with 11 bit resolution over a $\pm$ 5 mV range.

Usage

```
rec100
```

Format

A dataset with 3 variables and 108000 observations (the first 5 minutes out of the recording of about 30 minutes):

time the time base in seconds (0 - 299.997)

MLII ECG from a Modified Leg lead, in millivolts

V5 ECG from a V5 lead, in millivolts

Source

https://physionet.org/physiobank/database/mitdb/, (doi:10.13026/C2F305).

References

Moody GB, Mark RG. The impact of the MIT-BIH Arrhythmia Database. IEEE Eng in Med and Biol 20(3):45-50 (May-June 2001). (PMID: 11446209)

Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215-e220, 2000 (June 13).

Index