Chapter 2
Loading, Plotting, and Filtering RR Intervals

Abstract The initial steps to work with RHRV functions are presented in this chapter. The process starts with the loading of records containing beat positions that should be preprocessed prior to frequency, time, or nonlinear analysis. Data can be stored in various types of files, and RHRV routines can deal with different data formats. Next, heart rate must be obtained from beat positions. It may occur that spurious points appear in the heart rate signal. RHRV allows users to delete these outliers, when necessary. Besides, the signal can be filtered to reject automatically points that do not correspond to acceptable physiological values.

2.1 Getting Started

RHRV is a free software package to analyze HRV in frequency, time, and nonlinear domains. The starting point is a beat positions series that must be loaded into a specific structure, described later in the text. RHRV is not able to process ECG directly. In this case, before starting working with RHRV, it is necessary to use a tool capable of identifying the beats and exporting them to any of the multiple formats supported by RHRV (interested readers can check Appendix B).

Once loaded into RHRV, data containing beat positions must be processed to obtain a heart rate signal that can be used to perform different analyses. In order to obtain satisfactory results, outliers should be first removed (manually or automatically), and data must be interpolated.

Figure 2.1 summarizes this process, including the RHRV functions that should be used. The rest of this chapter explains all these functionalities in depth.

C.A. García Martínez et al., *Heart Rate Variability Analysis with the R package RHRV*, Use R!, DOI 10.1007/978-3-319-65355-6_2

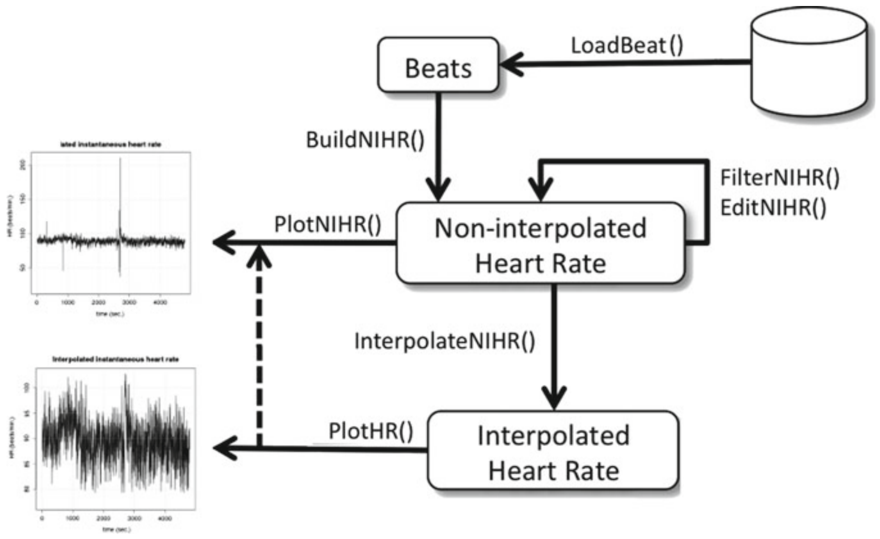

Fig. 2.1 Steps of the preprocessing procedure to obtain the heart rate signal

2.2 Data File Format

To perform HRV analysis from the sequence of beat positions, it will be necessary to generate the corresponding heart rate signal. This signal can then be analyzed in terms of frequency, time, or nonlinear parameters.

In RHRV, data acquisition can be performed from a simple ASCII (American Standard Code for Information Interchange) file, that is, a single-column file containing beat positions, which may appear in different units (seconds, milliseconds,etc.). However, data is not always available in this format, thus implying a previous conversion or using alternative functions to load it into RHRV. In the following paragraphs, we will describe the most commonly used formats to store beat position information.

The WFDB (WaveForm DataBase) software is a collection of tools for viewing, analyzing, and creating recordings of physiological signals. These tools are freely distributed from the PhysioNet Web site [5]. We will use the term "WFDB format" to refer to those records that follow the standard accepted by the WFDB software tools, which is one of the most used formats by researchers in ECG signal processing.

The WFDB format uses header files to specify the format and attributes of signal and annotation files. Although the WFDB format can be used to store records of different physiological signals, it is mainly used to store ECG records. In this case, the header contains information about the number of leads, calibration format, etc. ECG samples are stored as binary files, and, if there is information about beat positions or other annotations, it will be stored in one or more annotation files.

Nowadays, WFDB is being widely used by a considerable part of the research community, as well as by different organizations. However, there are many other format files worth presenting here. EDF+ (European Data Format) is another extended format, characterized by its flexibility and simplicity [1, 16]. It is commonly used for exchange and storage of multichannel biological and physical signals, which are recorded in .edf files.

EDF+ is based on EDF [17], but it can store many more types of data. It is a binary file with two main parts: header, which identifies the record and specifies the technical characteristics of the stored signals; and data, corresponding to the samples of the signals.

Improvements in technology have enabled the development of an increasing number of mobile devices (smartphones, tablets, watches, etc.), which can be used for obtaining real-time heart rate data by means of a chest strap. Products can be found for sport applications, such as Sports Tracker [7] or Endomondo [2]. Most of these applications obtain and show only the mean heart rate, but some recent chest straps are also able to detect instantaneous beat positions and to store and export these values. One of these is the Polar WearLink band, which can record the heart rate in a very simple and easy way using a specific data format [6].

Some Polar devices generate a .hrm file, storing data in ASCII format. Each file is composed of several data sections that are separated by empty lines. In each line, the data section name is separated from the data by using brackets. Multiple data in each row are separated by tab characters, including optional information about the exercise, such as speed, cadence, and altitude.

Other widely used format is the one used by Suunto (.sdf files), one of the most important watch manufacturers in the field of trekking, scuba diving, or climbing [8]. These .sdf files also store information in ASCII format and contain several sections delimited by their names between brackets. A header section specifies all the parameters that must be considered, each of them in one line, if they are present in the file (not all of them are mandatory). Data is stored in other section, separating the different attributes by colons.

Apart from these most commonly used formats, data can be also stored in RR files that are usually text files which only include the time differences between each consecutive beats.

In the next section, we will explain how to load data into RHRV from different file formats, namely ASCII, WFDB, EDF+, Polar, Suunto, and RR formats.

2.3 Loading Beat Series into RHRV

In order to load the heart beat positions from the data files, a custom data structure, called *HRVData*, must be initially created by using the RHRV software package. Figure 2.2 shows the fields contained in *HRVData*. This structure is implemented in R language as a list data object, and it will store in different fields all the information related to the digital records, including date and time (*datetime* field), verbose (returning of information to the user, *Verbose* field), beat positions (*Beat* field), interpolated heart rate series (*HR* field), frequency of interpolation (*Freq_HR* field), episodes

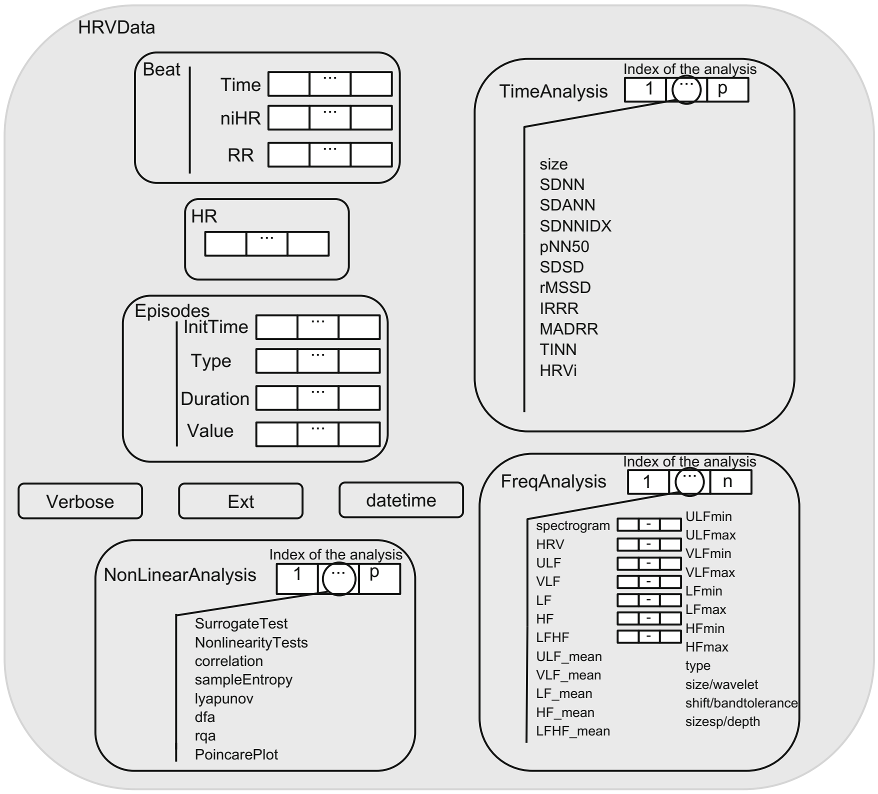

Fig. 2.2 *HRVData* structure

(specific intervals within the heart beat time series that can exist or not, *Episodes* field), and parameters resulting from the different time (*TimeAnalysis* field), frequency (*FreqAnalysis* field), and nonlinear analysis (*NonLinearAnalysis* field).

All the fields in this structure will be carefully described in this book. Meanwhile, in this paragraph, we will explain the fields of the data structure needed to load and preprocess the original data:

- *datetime*: information related to the date and time associated with the original record, provided in the original data file or specified by the user.
- *Verbose*: boolean argument to enable or disable verbose mode.
- *Beat*: dataframe to store information on the instantaneous heart rate data. It contains the original beat positions (*Time*), the instantaneous, noninterpolated frequency (*niHR*), and the RR intervals (distances between two consecutive beats, *RR*).
- *HR*: vector containing the interpolated heart rate data.
- *Freq_HR*: the heart rate interpolation frequency.
- *Ext*: string used as file extension by loading/writing functions.

It is important to notice that some fields of the *HRVData* structure may be empty. These fields will be filled by processing routines when applied to the original heart beat position signal.

Listing 2.3.1 shows the code to create the *HRVData* structure.

R listing 2.3.1

```
# HRVData structure containing the heart beats
library(RHRV)
hrv.data <- CreateHRVData()
hrv.data <- SetVerbose(hrv.data, TRUE)
```

Once the data structure has been created, the beats should be loaded. Depending on the original file format, the user must employ a different R routine. We will show how to use various loading routines in the following paragraphs.

Different functions can be used to load beat data into the *HRVData* structure:

- *LoadBeatAscii*: for data stored in an ASCII file.
- *LoadBeatWFDB*: for WFDB format records.
- *LoadBeatEDFPlus*: for EDF+ data files.

However, a more general function (*LoadBeat*) can be used to load records in the RHRV software package. In this case, the format should be specified as an argument of this function.

As a practical example, we will consider now a record stored in the WFDB format that can be freely donwloaded from PhysioNet [3]. It belongs to the European ST-T Database, intended to be used in the evaluation of algorithms for the analysis of ST and T-wave changes [14, 23]. In the example, we are going to use the “e0115” file (header, data, and annotation files).

As previously said, to load the beat position information, the user can employ either the generic *LoadBeat* or the *LoadWFDB* functions. In both cases, the *Load-HeaderWFDB* function, which reads the header file in order to obtain information, is implicitly executed. In Listing 2.3.2, the code to load the original data in two different *HRVData* structures, employing both functions, is presented. Note that an argument giving the extension of the annotation file must also be provided (*annotator*, by default “qrs”).

R listing 2.3.2

```
# Loading the beats in WFDB format
hrv.data1 <- CreateHRVData()
hrv.data <- SetVerbose(hrv.data1, TRUE)
hrv.data2 <- CreateHRVData()
hrv.data <- SetVerbose(hrv.data2, TRUE)
hrv.data1 <- LoadBeat("WFDB", hrv.data1,"sampleData/e0115", annotator = "atr")
hrv.data2 <- LoadBeatWFDB(hrv.data2, "sampleData/e0115", annotator = "atr")
```

As it can be observed, both routines yield the same original data, stored in *hrv.data1* and *hrv.data2*.

We will explain now how to load data in ASCII format. An example of an ASCII file that can be freely downloaded is "beat_ascii.txt", available from the MILE Group Web page [4]. This file contains heart beat positions, and it was generated employing the VARVI (Variability of the heArt Rate in response to Visual stImuli) software package, a free software tool developed to perform heart rate variability analysis in response to different visual stimuli [22]. In this case, the code to load the original data and its result are given in Listing 2.3.3.

R listing 2.3.3

```
# Loading the beats in ASCII format
hrv.data3 <- CreateHRVData()
hrv.data <- SetVerbose(hrv.data3, TRUE)
hrv.data3 <- LoadBeatAscii(hrv.data3, "sampleData/beat_ascii.txt")
```

The same result could be obtained using *LoadBeat("Ascii", hrv.data3, "beat_ascii.txt").*

The rest of the file formats supported by RHRV can be loaded in a similar way by using the functions *LoadBeatEDFPlus*, *LoadBeatSuunto*, *LoadBeatPolar*, and *LoadBeatRR*.

Both the generic *LoadBeat* function and the rest of the functions for loading the various data file formats can include several input parameters, such as the record path (*RecordPath*, by default '.') if the file to load is not in the working directory; the scale, factor applied in RR or ASCII format if data are not in seconds (*scale*, by default 1); the datetime, also for ASCII and RR files (*datetime*, by default "1/1/1900 0:0:0"); and the annotation type for EDF+ format (*annotationType*, by default "QRS"). For ASCII files, the possibility of loading only a specific portion of the record is also controlled by two other parameters: *starttime* and *endtime*. If they are not specified, the full record is loaded. When specific values (in seconds) are provided, only the corresponding portion of the record is loaded into the *HRVData* structure.

2.4 Preprocessing

At this point, a heartbeat sequence has been loaded into the *HRVData* structure, and it is time to preprocess the signal in order to obtain the heart rate signal. This is the starting point to perform time, frequency, and nonlinear analysis. A detailed explanation is given in the following sections.

2.4.1 Instantaneous Heart Rate Signal Extraction

The distance between two consecutive beats is the well-known RR interval [15], which corresponds to the distance of the R waves associated with each beat. The instantaneous heart rate can be defined as the inverse of the time separation between two consecutive heart beats. In RHRV, this corresponds to the calculation of the series:

$$RR[i] = (Time[i] - Time[i-1]) * 1000$$

$$HR(i) = \frac{1000}{RR[i] * 60}$$

where *Time[i]* is the time when the beat i occurs, measured in seconds, *RR[i]* is the beat-to-beat distance, measured in milliseconds, and *HR(i)* is the instantaneous heart rate, measured in beats per minute.

2.4.2 Removing Artifacts

Algorithms to detect and classify heartbeats from ECG signals often fail or may yield incorrect outputs. This means that the instantaneous heart rate signal that is directly obtained from the RR intervals can be corrupted with artifacts, that is, undesired information originated by either extern sources or physiological events. Some QRS complexes can be missed, some anomalous, ectopic beats can be incorrectly classified as normal, or the original signal can be affected by noise. Ectopic beats are not triggered as the result of the heart rate control mechanisms, but they arise from the automatism that heart muscle fibers present. Given that the goal of HRV analysis is the study of the heart rate control mechanisms, these beats must be removed before performing a HRV analysis. Besides, as the beats are not regularly originated, the heart rate is not constant, which means that the signal will be not sampled regularly in time.

Several scientific works have tried to establish if the existing detection algorithms are adequate to perform heart rate variability analysis, or if a manual revision of detection is required [11, 18]. Conclusions of those works indicate that errors may significantly affect some variability indices, specifically those corresponding to spectral analysis [20, 24].

Researchers have proposed removing artifacts employing a broad variety of algorithms. Some of these works include wavelet-based methods [19, 27], different filters and thresholding values [9, 21], or independent component analysis techniques [13].

The RHRV software package provides an algorithm to automatically remove artifacts. This algorithm uses adaptive thresholding for rejecting beats whose RR value differs from previous and following beats, and from a mobile mean more than a threshold value. The filter also removes points that are not within acceptable physiological values [25]. This algorithm is presented as pseudo-code in Listing 2.4.1.

R listing 2.4.1

```
ULAST = 13
LONG = 50
UMEAN = 1.5 * ULAST
MINIMUM = 12
MAXIMUM = 20
MINBMP = 25
MAXBPM = 200
from (i = 2 to NBEATS) {
  if (i < LONG)
    MEAN = mean of previous beats
  else
    MEAN = mean of the last long beats
  if ((100 * abs(hr(i)  hr(i - 1)) / hr(i - 1)) < ULAST) ||
      (100 * abs(hr(i)  hr(i + 1)) / hr(i - 1)) < ULAST) ||
      (100 * abs(hr(i)  MEAN) / MEAN) < UMEAN) &&
      (MINBPM <= hr(i) <= MAXBPM)) {
      valid beat
      SIGNALDEV = 10 + SIGNALDEV(last LONG beats)
      if (SIGNALDEV < MINIMUM)
        SIGNALDEV = MINIMUM
      if (SIGNALDEV > MAXIMUM)
        SIGNALDEV = MAXIMUM
      update ULAST = SIGNALDEV
      update UMEAN = 1.5 * SIGNALDEV
  }
  else
    i = i + 1
}
```

This algorithm uses as starting point some prefixed values (experimentally calculated) that include the initial threshold values for the filter (*ULAST = 13* and *UMEAN = 1.5 * ulast*), the number of beats (*LONG*) employed to calculate the mean value and corresponding standard deviation, and minimum (*MINIMUM = 12*) and maximum (*MAXIMUM = 20*) values allowed for the *ULAST* threshold. A fixed quantity is also added to the signal deviation (*SIGNALDEV*) to obtain the adaptive threshold employed by the filter. Accepted physiological minimum (*MINBPM*) and maximum (*MAXBPM*) values are also fixed.

Using a loop, all the consecutive beats are analyzed, excluding the first and last beat, since it is not possible to compare them with the previous or next beats, respectively. The mean value of the beats is calculated considering the *LONG* previous beats, if possible. A beat will be considered as a valid one if the corresponding heart rate differs from the previous or from the next beat less than *ULAST* (in %), or from the mean less than *UMEAN* (in %).

Threshold values are updated after a number of beats equal to *LONG*. If an artifact is detected, the next beat will be discarded, since its associated RR distance is not valid.

Most artifacts are removed by this algorithm. However, RHRV also supplies a function that provides an intuitive method to manually remove the remaining artifacts, by selecting them from a graphical window.

After the automatic and/or manual edition of the artifacts, the noninterpolated, nonequally spaced heart rate signal is obtained as result.

2.4.3 Interpolation of the Heart Rate Signal

The noninterpolated heart rate signal obtained in the previous step can be used to obtain time and nonlinear measurements. However, in the frequency analysis, several problems may arise, since one of the main drawbacks is that sometimes the continuity of the signal is broken, and in the frequency domain, this can lead to erroneous results. Because of this, an important aspect that should be carefully studied is handling artifacts and noninterpolated signals. Furthermore, the RR signal does not have equidistant sampling, and most of the frequency algorithms require an evenly interpolated signal.

Many different interpolation algorithms can be found in the literature. The most usual are linear or cubic splines interpolation, but many others can be used. Some researchers performed interpolation using the K-nearest neighbor (k-NN) algorithm [10], or used Fourier-based methods to interpolate data [26].

When using RHRV, users can select either linear or cubic splines interpolation [12]. The sampling frequency to obtain the equally spaced heart rate signal has to be set in both cases (default is 4 Hz). After this step, a new heart rate signal that is adequate for performing spectral analysis, as well as time and nonlinear studies is obtained.

2.5 Preprocessing Beat Data with RHRV

In this section, a detailed explanation about the calculation of RR intervals with RHRV will be provided. Filtering and edition of the heart rate signal will also be explained in a practical way.

As it was explained in former paragraphs, the heart rate signal needs to be obtained from the heart beats data before performing different types of analysis. First, the noninterpolated heart rate signal is obtained and must be filtered to remove outliers and detection errors.

Let us consider the *hrv.data3* structure, containing the heart beats data, created in Listing 2.3.3. From this signal, the instantaneous heart rate will be now obtained

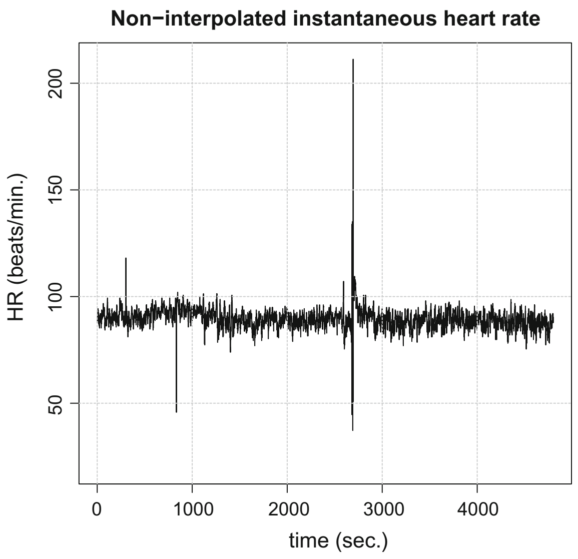

Fig. 2.3 Instantaneous heart rate signal for the "beat_ascii.txt" data

with the *BuildNIHR* function. The corresponding R code developed in the RHRV software package, and results are given in Listing 2.5.1.

R listing 2.5.1

```
# Building the non-interpolated heart rate signal
hrv.data3 <- BuildNIHR(hrv.data3)
```

Now, the instantaneous, noninterpolated heart rate signal has been calculated. The previously empty fields in *hrv.data3* corresponding to *Beat*, that is, *niHR* and *RR* are now filled with the appropriate values. A graphical representation can also be obtained using the function *PlotNIHR(hrv.data3)*, a plot function that allows the representation of data values for the instantaneous heart rate (Fig. 2.3).

The automatic filtering algorithm can now be applied over the heart rate signal, employing the *FilterNIHR* function. Listing 2.5.2 provides the R code and results of this function.

R listing 2.5.2

```
# Filtering the non-interpolated heart rate signal
hrv.data3 <- FilterNIHR(hrv.data3)
```

When the function *FilterNIHR* is used, the algorithm previously described will reject those beats that differ a certain value from the established threshold. If not specified, default parameters are used (*long* $= 50$ and *last* $= 13$).

This filtering also eliminates points that are not within acceptable physiological values. Users can modify these physiological values as well, by varying the values

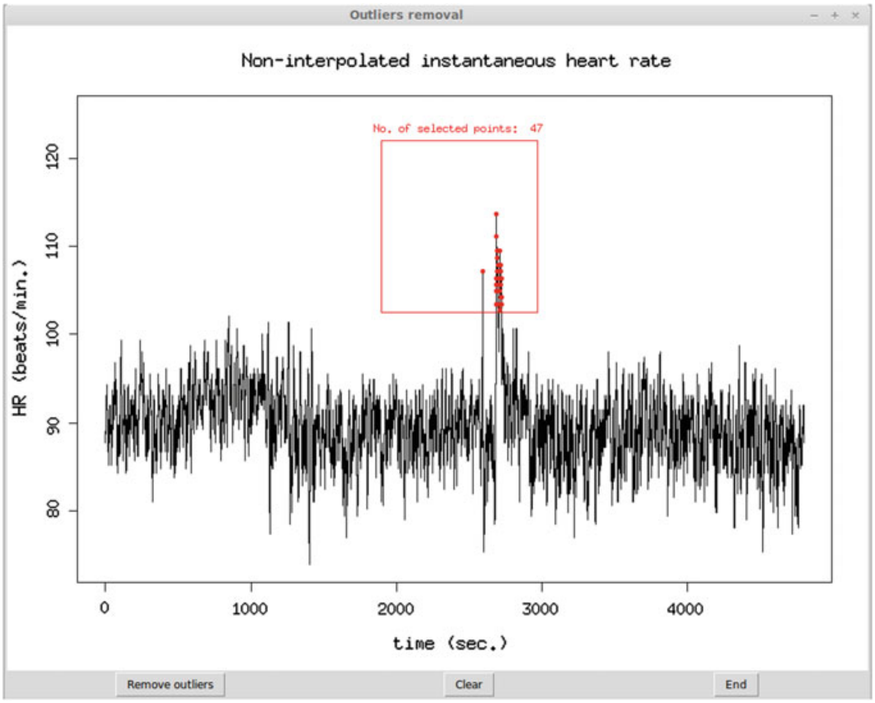

Fig. 2.4 Example of use of the *EditNIHR* function. Outliers can be manually removed

of the arguments *minbpm* (minimum value) and *maxbpm* (maximum value), set by default to 25 and 200, respectively. The rest of the arguments (*mini*, *maxi*, *fixed*, and *verbose*) are deprecated ones and fixed to NULL by default.

For manually removing outliers, the *EditNIHR* function can be used. Figure 2.4 shows an example of use of the *EditNIHR* function. The user can interact with the graphical window and remove outliers. The user has to select the points to be removed by clicking on the upper-left corner, and on the bottom-right corner of the area enclosing the points. This way, a red rectangular area will be marked, and the user can decide to remove the points within (remove outliers button), to clear the selected area (Clear button), or to finish the manual edition. After removal, RHRV gives information about the removed points.

In case the user does not want to use the *EditNIHR* function, but wants to remove more spurious points, the function *FilterNIHR* can be applied more than once. This can result in the elimination of more beats because, after running once the function and eliminating some of the beats, the dynamic thresholds of the filter will change. Results can be similar to the application of the *EditNIHR* function, as shown in Listing 2.5.3.

R listing 2.5.3

```
# Filtering the non-interpolated heart rate signal
hrv.data3 <- FilterNIHR(hrv.data3)
```

It is time now to estimate the interpolated heart rate signal. The function that performs this task is *InterpolateNIHR*, with the following arguments:

- sampling frequency (*freqhr*): in Hz. (default value is 4).
- interpolation method (*method*): default value is linear, although spline can be selected.
- *verbose*: deprecated.

Listing 2.5.4 provides the code and results of the calculation and graphical representation of the results.

R listing 2.5.4

```
#Interpolation of the instantaneous heart rate
hrv.data3 <- InterpolateNIHR(hrv.data3)
```

After applying these functions, the *HRVData* structure will contain in its fields: the original heart beat positions, the RR intervals, the noninterpolated heart rate signal, and the equally spaced heart rate signal. This last field can be graphically represented employing the *PlotHR* function (Fig. 2.5). The interpolated heart rate signal will be the starting point for the time, frequency, and nonlinear analysis that will be explained in next chapters of this book.

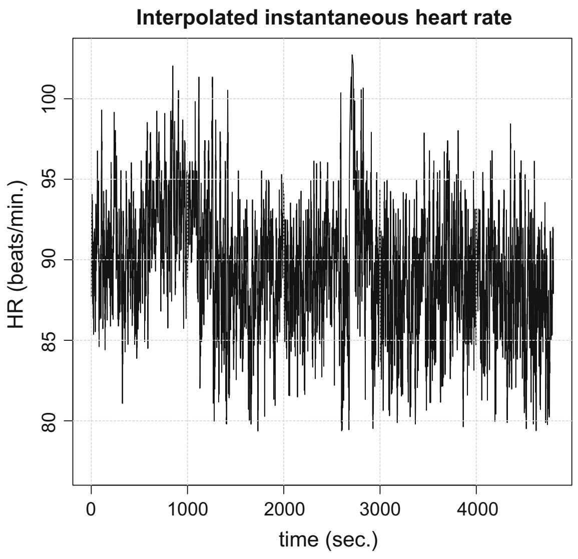

Fig. 2.5 Interpolated heart rate signal

References

1. EDF+: http://www.edfplus.info. Last accessed: 27-05-2016
2. Endomondo: http://www.endomondo.com. Last accessed: 27-05-2016
3. European ST-T Database: http://www.physionet.org/physiobank/database/edb. Last accessed: 27-05-2016
4. gHRV: http://milegroup.github.io/ghrv/packages.html. Last accessed: 27-05-2016
5. PhysioNet: http://www.physionet.org. Last accessed: 27-05-2016
6. Polar: http://www.polar.com. Last accessed: 27-05-2016
7. SportsTracker: http://www.sports-tracker.com. Last accessed: 27-05-2016
8. Suunto: http://www.suunto.com. Last accessed: 27-05-2016
9. Al Osman H, Eid M, El Saddik A (2015) A pattern-based windowed impulse rejection filter for nonpathological HRV artifacts correction. IEEE Trans Instrum Meas 64(7):1944–1956
10. Begum S, Islam MS, Ahmed MU, Funk P (2011) K-NN based interpolation to handle artifacts for heart rate variability analysis. In: IEEE international symposium on signal processing and information technology (ISSPIT), 2011. IEEE, pp 387–392
11. Bigger JT, Fleiss JL, Steinman RC, Rolnitzky LM, Kleiger RE, Rottman JN (1992) Frequency domain measures of heart period variability and mortality after myocardial infarction. Circulation 85(1):164–171
12. Friedman J, Hastie T, Tibshirani R (2009) The elements of statistical learning, vol 2. Springer
13. Gimeno-Blanes F, Rojo-Álvarez JL, Requena-Carrión J, Everss E, Hernández-Ortega J, Alonso-Atienza F, García-Alberola A (2007) Denoising of heart rate variability signals during tilt test using independent component analysis and multidimensional recordings. In: Computers in cardiology, 2007. IEEE, pp 399–402
14. Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE (2000) PhysioBank, PhysioToolkit, and PhysioNet. Components of a new research resource for complex physiologic signals. Circulation 101(23):e215–e220
15. Goldman MJ (1986) Principles of clinical electrocardiography. Lange Medical Publications
16. Kemp B, Olivan J (2003) European Data Format plus(EDF+), an EDF alike standard format for the exchange of physiological data. Clin Neurophysiol 114(9):1755–1761
17. Kemp B, Värri A, Rosa AC, Nielsen KD, Gade J (1992) A simple format for exchange of digitized polygraphic recordings. Electroencephalog Clin Neurophysiol 82(5):391–393
18. Malik M, Farrell T, Cripps T, Camm A (1989) Heart rate variability in relation to prognosis after myocardial infarction: selection of optimal processing techniques. Eur Heart J 10(12):1060–1074
19. Mcnames J, Thong T, Aboy M (2004) Impulse rejection filter for artifact removal in spectral analysis of biomedical signals. In: IEMBS'04. 26th annual international conference of the IEEE engineering in medicine and biology society, 2004, vol. 1. IEEE, pp 145–148
20. Molina-Picó A, Cuesta-Frau D, Miró-Martínez P, Oltra-Crespo S, Aboy M (2013) Influence of QRS complex detection errors on entropy algorithms. Application to heart rate variability discrimination. Comput Methods Programs Biomed 110(1):2–11
21. Niskanen J-P, Tarvainen MP, Ranta-Aho PO, Karjalainen PA (2004) Software for advanced HRV analysis. Comput Methods Programs Biomed 76(1):73–81
22. Rodríguez-Liñares L, Cuesta P, Alonso R, Méndez A, Lado M, Vila X (2013) VARVI: a software tool for analyzing the variability of the heart rate in response to visual stimuli. In: Computing in cardiology conference (CinC), 2013. IEEE, pp 401–404.
23. Taddei A, Distante G, Emdin M, Pisani P, Moody G, Zeelenberg C, Marchesi C (1992) The European ST-T database: standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography. Eur Heart J 13(9):1164–1172

24. Thuraisingham R (2006) Preprocessing RR interval time series for heart rate variability analysis and estimates of standard deviation of RR intervals. Comput Methods Programs Biomed 83(1):78–82
25. Vila J, Palacios F, Presedo J, Fernández-Delgado M, Félix P, Barro S (1997) Time-frequency analysis of heart-rate variability. Eng Med Biol Mag IEEE 16(5):119–126
26. Wei Z, Dechang C, Xueyun W, Hongxing L (2014) Heart rate estimation by iterative Fourier interpolation algorithm. Electron Lett 50(24):1799–1801
27. Zidelmal Z, Amirou A, Adnane M, Belouchrani A (2012) QRS detection based on wavelet coefficients. Comput Methods Programs Biomed 107(3):490–496