

1 **The R package CardiacDP**

2 The R package CardiacDP consists of two functions that can be used to collate
3 separate data files into a single data table for subsequent analyses (function
4 collatedata), and to compute heart rate from the cardiac data (function computeHR).
5 Depending on the input file format (whether it is a .zip file that requires data
6 collation, or a .csv file that is ready for heart rate computation), function
7 collatedata will be automatically called within the function computeHR, and the
8 overall workflow is summarized in **Figure 1**.

9

10 ***1 Function collatedata: data collation***

11 Some oscilloscope software (e.g. Picoscope) produces small and numerous data files
12 as outputs instead of large, collated data files with all measurements, presumably
13 because large data files are difficult to operate. These numerous files, however, have
14 to be collated into a single data table to facilitate subsequent data analysis
15 procedures. The function collatedata, therefore, serves to collate separate data
16 files to generate a single data table of cardiac data (each column represents
17 measurements of one individual; referred to as “channels”) measured across fixed
18 time intervals.

19 This function automatically reads data files and collates them in chronological order
20 as inferred by the file names. Taking the data files produced from Picoscope (v6, Pico
21 Technology, UK) as an example (**Figure 2**), each ‘page’ as visualized on the software
22 is saved as separate .csv files in one folder. After a defined number of pages, the
23 software will reset, producing another folder with, again, separate .csv files
24 representing different pages. All files are then nested inside one folder which can
25 then be compressed (as a .zip file) for this function. As long as the folders and files
26 are named in chronological order, this function is designed to automatically read
27 them accordingly, and one by one in hierarchy (i.e. collate all files within one folder
28 before moving on to the next folder). After collating all files, time will be rewritten as
29 a sequence increasing from zero at a fixed time interval. As such, only measurements

30 taken across fixed time intervals should be collated as one data table for analyses, or
31 else the data should be analyzed separately.

32

33 **2 Function computeHR: heart rate computation**

34 From the collated data table, the function `computeHR` analyzes channel (i.e.
35 individual) by channel and computes heart rates of each sequence. Users can
36 customize the parameters used in this function (**Table 1**).

37 To minimize computation time, the cardiac data are first analyzed at a reduced time
38 interval (default to be 0.01s; **Table 1**). As a result of inferring heart rates from time
39 lags in ACF, the time interval used in the analysis determines the resolution of heart
40 rates computed (which is implied by the difference in heart rate across a time lag
41 difference of 1). Taking the time interval of 0.01s as an example (**Table 2** and **Figure**
42 **3**), the difference in heart rate decreases with time lag (i.e. the resolution increases
43 with time lag) and such a relationship is not linear: for the same time lag difference
44 of 1, the difference in heart rate computed is large at small time lags (coarse
45 resolution), whereas the difference is small at large time lags (fine resolution). As
46 such, the data are automatically re-analysed at the finest time interval when the
47 corresponding percentage difference is > 2% (**Table 2** and **Figure 3**).

48 By employing ACF with a genetic algorithm framework, the positions and durations
49 of the final periodic sub-sequences (“finalsubseq”) and their candidate heart rates
50 (“candidateHR”) are determined from the cardiac data per sequence (**Table 3**; see
51 also **Section 3** for the examples). From these sub-sequences, the final heart rates are
52 computed for each sequence, either with or without implementing the tracking
53 index (i.e. “results_ACF” and “results_TI” refer to the “ACF + GA” and “ACF + GA +
54 TI” approaches as described in the main manuscript respectively; **Table 3**; see also
55 **Section 3** for the examples).

56

57 ***3 User guidelines***

58 The functions of this package are illustrated with the examples below.
59 When a .zip file (with the corresponding file path) comprising folders and files named
60 in chronological order is provided:

```
output <- computeHR (file_path = "C:/Users/Documents/20210518A.zip")
```

61 The function `collatedata` will be automatically called to read and collate the
62 separate files. The data structure will be analyzed and displayed for the user's
63 reference, and the progress of collation will be indicated.

```
[[1]]
[1] "Zipped file name: 20210518A"

[[2]]
[1] "Number of files: 6"

[[3]]
      folders pages
1:    20210518A-0001     90
2: 20210518A-0001 (2)     90
3: 20210518A-0001 (3)     90
4: 20210518A-0001 (4)     90
5: 20210518A-0001 (5)     90
6: 20210518A-0001 (6)     61

[[4]]
[1] "Total duration: 170 mins"

[[5]]
[1] "Number of channels: 8"

[[6]]
[1] "Names of channels: Channel A, Channel B, Channel C, Channel D, Chann
el E, Channel F, Channel G, Channel H"
```

```
[1] "Reading data: 17%"
[1] "Reading data: 33%"
[1] "Reading data: 50%"
[1] "Reading data: 67%"
[1] "Reading data: 83%"
[1] "Reading data: 100%"
[1] "Finalizing..."
```

64 After collation, the structure of the collated data table is displayed and the collated
65 data table is automatically saved to the same file path as the input file.

```
Classes 'data.table' and 'data.frame': 69003396 obs. of 9 variables:  
$ Time    : num 0 0.00133 0.00267 0.004 0.00533 ...  
$ Channel A: num -0.002198 -0.00058 -0.001129 -0.000061 -0.001648 ...  
$ Channel B: num 0.00369 0.00314 0.00424 0.00314 0.00314 ...  
$ Channel C: num 0.00424 0.00479 0.00424 0.00372 0.00479 ...  
$ Channel D: num -0.000488 -0.000488 -0.001007 -0.001007 -0.001007 ...  
$ Channel E: num -0.0266 -0.0277 -0.0282 -0.0277 -0.0304 ...  
$ Channel F: num -0.023 -0.0246 -0.0246 -0.023 -0.0241 ...  
$ Channel G: num 0.00906 0.00906 0.01337 0.00748 0.008 ...  
$ Channel H: num -0.00534 -0.00375 -0.00641 -0.00693 -0.00693 ...  
- attr(*, ".internal.selfref")=<externalptr>
```

```
[1] "Collated data table saved as 20210518A.csv"
```

- 66 With the collated data table, the heart rate computation analyses are conducted
67 channel by channel and the progress will be indicated.

```
[1] "Calculating heart rates: Channel A..."  
[1] "Generating output..."  
[1] "Calculating heart rates: Channel B..."  
...
```

- 68 The user can then access the output by the channel name. `finalsubseq` is a list
69 showing the positions and durations of the final sub-sequences determined for each
70 sequence. They are presented as data tables (`s` = start index of the sub-sequence; `e` =
71 end index of the sub-sequence; `p` = which of the initial population the sub-sequence
72 is derived from; and `f` = duration of the sub-sequence), the rows of which represent
73 separate final sub-sequences. The example below shows the first five sequences (e.g.
74 `[[4]]` indicates at the 4th minute there are three final sub-sequences).

```
# positions (in indices) and durations of the final sub-sequences  
output [[“finalsubseq”]] [[“Channel A”]]
```

```
[[1]]  
  s   e p   f  
1: 1 6430 1 6429  
  
[[2]]  
  s   e p   f  
1: 1 4387 1 4386  
  
[[3]]  
  s   e p   f
```

```

1: 1 5775 1 5774

[[4]]
      s   e p     f
1: 2197 3856 6 1659
2: 3669 6430 8 2761
3: 3570 6363 8 2793

[[5]]
      s   e p     f
1:    1 4977 2 4976
2: 780 5785 9 5005
3: 1028 6430 9 5402

...

```

75 The corresponding candidate heart rates per sub-sequence can be found in
 76 `candidateHR` (ACF = autocorrelation value; lag = time lag; and hr = heart rate). Taking
 77 the 4th minute again as an example, there are two candidate heart rates for the first
 78 sub-sequence (as shown in [[4]][[1]]), and only one candidate heart rate for the
 79 other two sub-sequences (as shown in [[4]][[2]] and [[4]][[3]]).

```

# candidate heart rates of the final sub-sequences
output [[“candidateHR”]] [[“Channel A”]]

```

```

[[1]]
[[1]][[1]]
      ACF lag      hr
1: 0.1438675 88 73.05174
2: 0.5748986 180 35.71418

[[2]]
[[2]][[1]]
      ACF lag      hr
1: 0.5141086 170 37.81502

[[3]]
[[3]][[1]]
      ACF lag      hr
1: 0.5002503 164 39.19849

[[4]]
[[4]][[1]]
      ACF lag      hr
1: 0.4296444 156 41.20867
2: 0.5028381 307 20.93991

[[4]][[2]]

```

```

      ACF lag      hr
1: 0.5149 188 34.19443

[[4]][[3]]
      ACF lag      hr
1: 0.5145251 186 34.56211

[[5]]
[[5]][[1]]
      ACF lag      hr
1: 0.5236133 180 35.71418

[[5]][[2]]
      ACF lag      hr
1: 0.5018653 174 36.94571

[[5]][[3]]
      ACF lag      hr
1: 0.5023476 172 37.37531

...

```

80 The final results after evaluating the candidate heart rates, checking for resolution
 81 and weighting for durations can eventually be obtained from `results_ACF` and
 82 `results_TI`. These results refer to evaluating the candidate heart rates by
 83 autocorrelation values (i.e. the “ACF + GA” approach) and the tracking index (i.e. the
 84 “ACF + GA +TI” approach) respectively. Each of them consists of 1) the details of the
 85 sub-sequences (`ix` = sequence (in minute), `res` = time interval (resolution) used in the
 86 final analysis, and the remaining variables are defined as above; `subseqHR`); 2) the
 87 weighted heart rate per sequence (`weightedHR`); and 3) a plot of weighted heart rate
 88 against time (`plot`).

```
# results obtained from evaluating the candidate heart rates by autocorrelation values
output [[“results_ACF”]] [[“Channel A”]]
```

	\$subseqHR									
	ix	win	s	e	p	f	ACF	lag	hr	res
1:	1	1	1	6430	10	6429	0.5748986	180	35.71418	0.00933336
2:	2	1	1	4201	1	4200	0.5240022	170	37.81502	0.00933336
3:	3	1	1	5725	1	5724	0.5000011	164	39.19849	0.00933336
4:	4	1	3215	3856	6	641	0.5241728	156	41.20867	0.00933336
5:	4	2	3548	6430	8	2882	0.5040846	188	34.19443	0.00933336

227:	166	1	NA	NA	NA	NA		NA	NA	NA 0.00933336
228:	167	1	NA	NA	NA	NA		NA	NA	NA 0.00933336
229:	168	1	NA	NA	NA	NA		NA	NA	NA 0.00933336

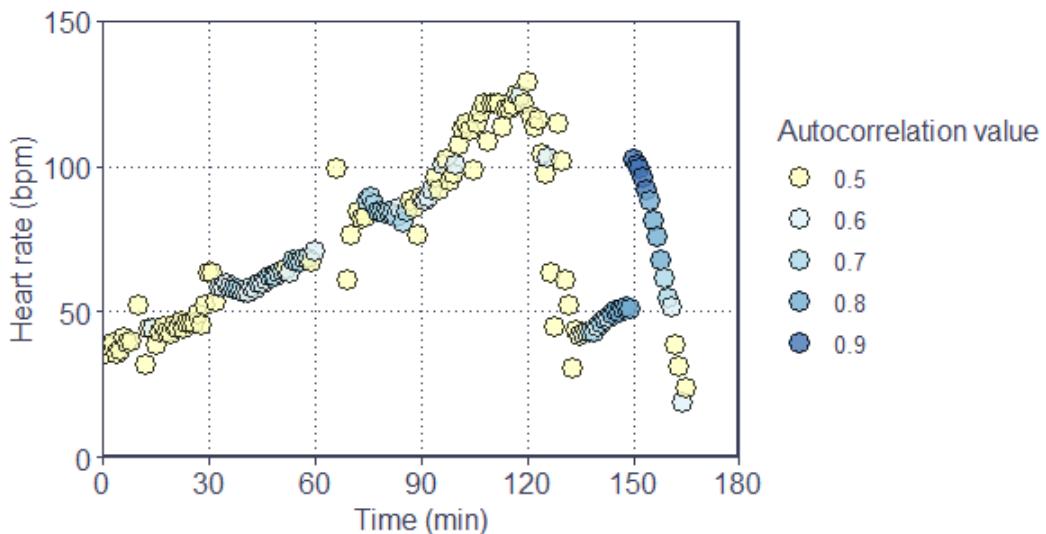
```

230: 169   1   NA   NA NA   NA           NA   NA   NA  0.00933336
231: 170   1   NA   NA NA   NA           NA   NA   NA  0.00933336

$weightedHR
  ix      wACF      whr
1: 1 0.5748986 35.71418
2: 2 0.5240022 37.81502
3: 3 0.5000011 39.19849
4: 4 0.5071297 35.25769
5: 5 0.5134750 36.53428
---
166: 166      NA      NA
167: 167      NA      NA
168: 168      NA      NA
169: 169      NA      NA
170: 170      NA      NA

$plot

```



```

# results obtained from evaluating the candidate heart rates by the tracking index
output [[“results_TI”]] [[“Channel A”]]

```

```

$subseqHR
  ix win      s      e      p      f          ACF lag      hr      res
1: 1   1      1 6430 10 6429 0.5748986 180 35.71418 0.00933336
2: 2   1      1 4201 1 4200 0.5240022 170 37.81502 0.00933336
3: 3   1      1 5725 1 5724 0.5000011 164 39.19849 0.00933336
4: 4   1 3215 3856 6 641 0.5241728 156 41.20867 0.00933336
5: 4   2 3548 6430 8 2882 0.5040846 188 34.19443 0.00933336
---
227: 166   1   NA   NA NA   NA           NA   NA   NA  0.00933336
228: 167   1   NA   NA NA   NA           NA   NA   NA  0.00933336
229: 168   1   NA   NA NA   NA           NA   NA   NA  0.00933336
230: 169   1   NA   NA NA   NA           NA   NA   NA  0.00933336
231: 170   1   NA   NA NA   NA           NA   NA   NA  0.00933336

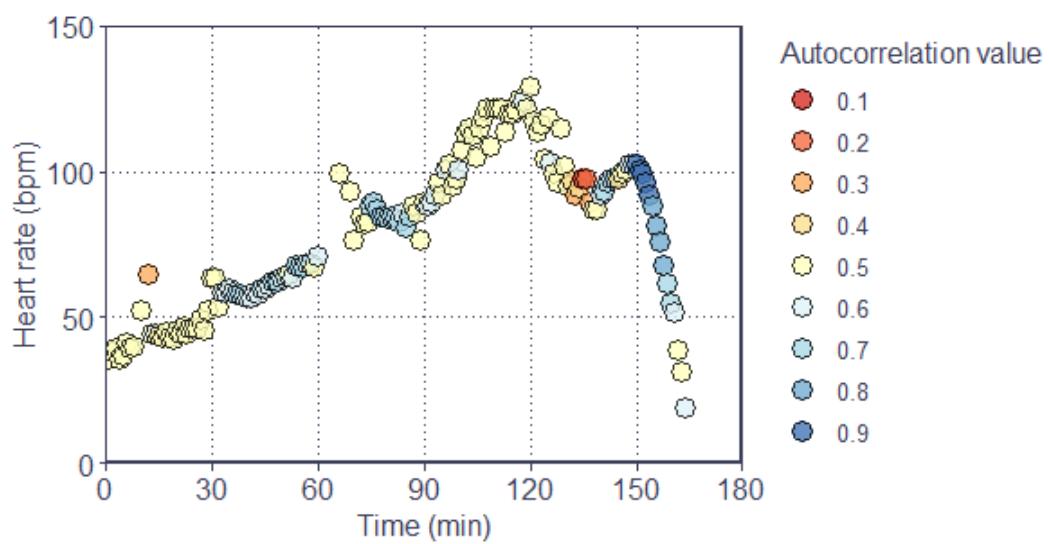
```

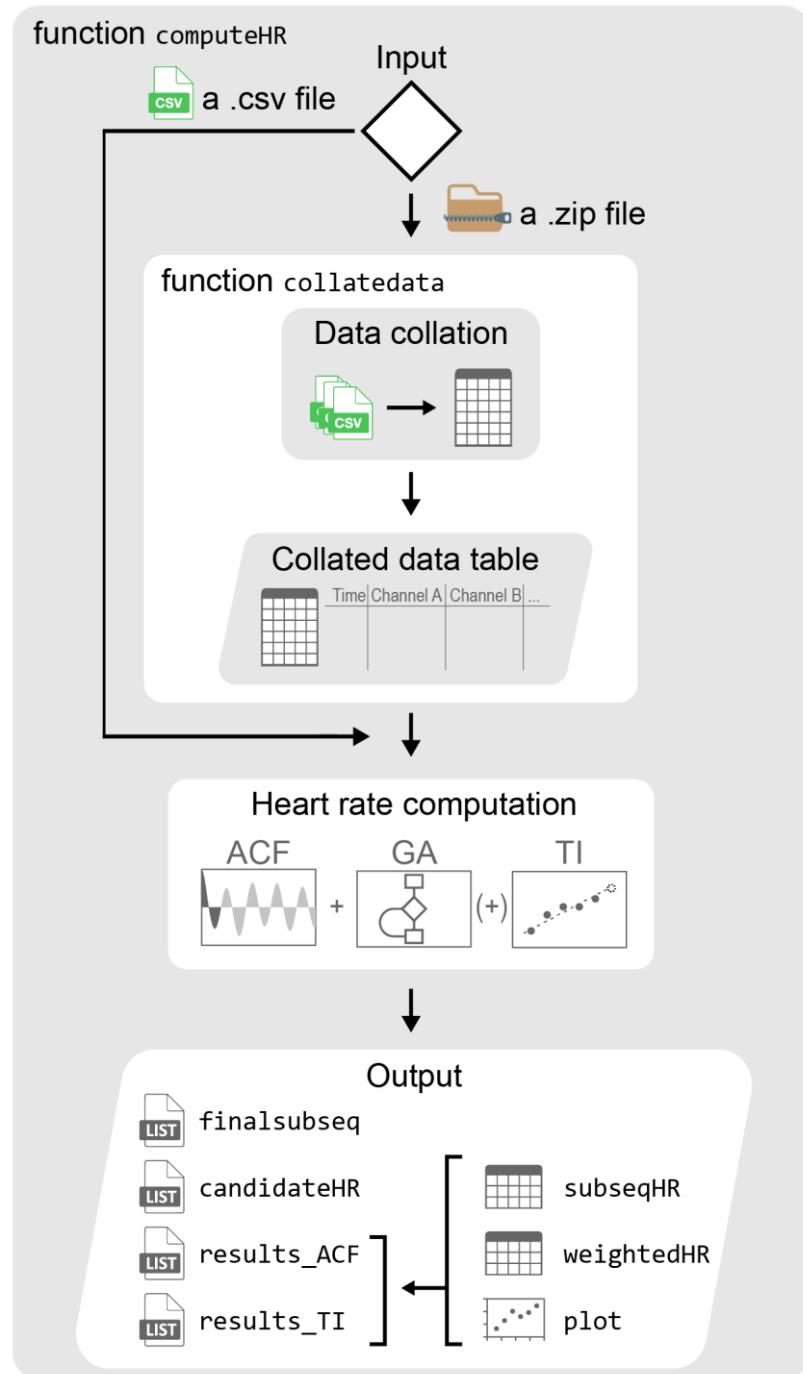
```

$weightedHR
  ix      wACF      whr
1: 1 0.5748986 35.71418
2: 2 0.5240022 37.81502
3: 3 0.5000011 39.19849
4: 4 0.5071297 35.25769
5: 5 0.5134750 36.53428
...
166: 166      NA      NA
167: 167      NA      NA
168: 168      NA      NA
169: 169      NA      NA
170: 170      NA      NA

$plot

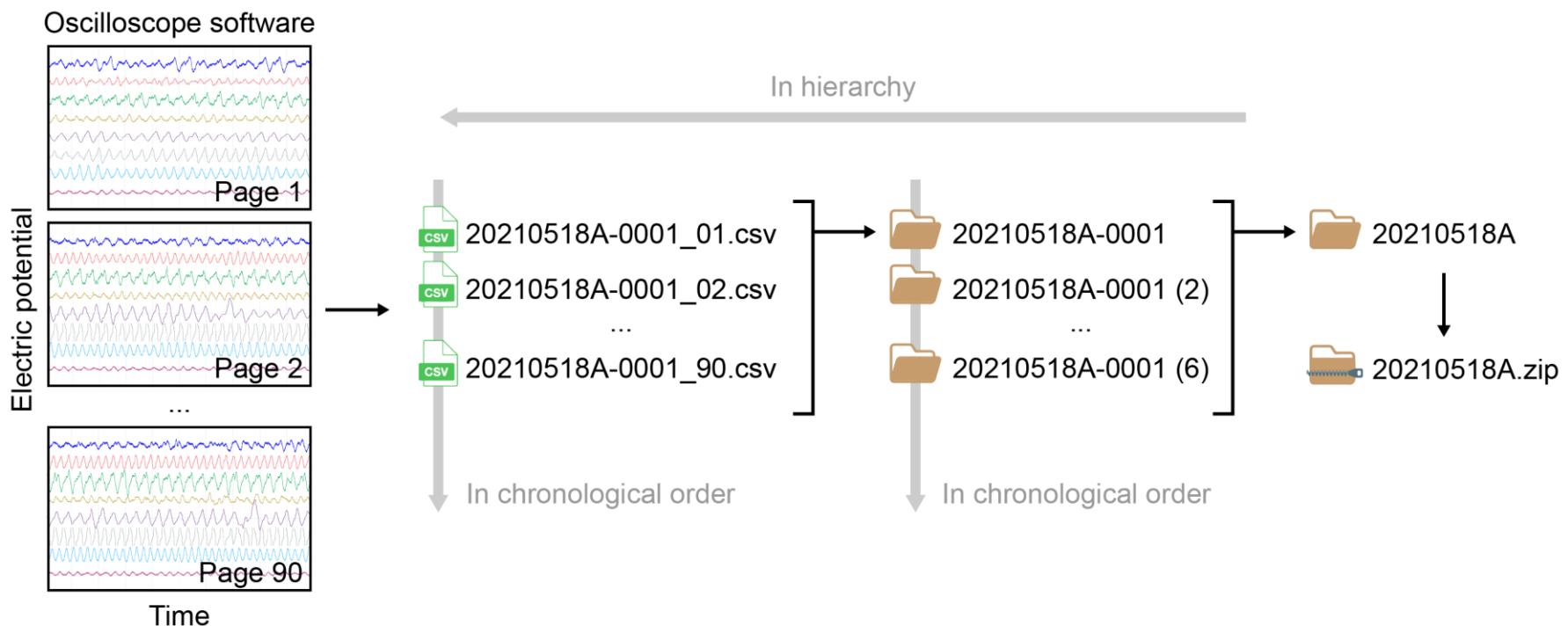
```





90

91 **Figure 1** Workflow of the R package CardiacDP. In function `computeHR`, depending on
92 the input file format, either the function `collatedata` will be automatically called to
93 read and collate separate files into a single data table in case of a `.zip` file, or the `.csv`
94 file will proceed directly to the heart rate computation using autocorrelation
95 function (ACF), genetic algorithm (GA) and tracking index (TI). Details of the output
96 are summarized in **Table 3**.



97

98 **Figure 2** An example data structure of the data files produced by the oscilloscope software Picoscope (v6, Pico Technology, UK). Each ‘page’ of
 99 the software is saved as separate .csv files nested within a folder. After a defined number of pages, the software resets and creates another
 100 folder in which ‘pages’ are again saved as separate .csv files. The files and folders are automatically named in chronological order and are
 101 finally nested within a single folder which is compressed to a .zip file for the function `collatedata`. This function reads the data in chronological
 102 order and in hierarchy (as indicated by the vertical and horizontal grey arrows respectively) and collates them to give a single data table for
 103 subsequent analyses.

104 **Table 1** Variables used in heart rate computation that allow user customization.

Variable	Definition	Default value
an_in	As in ‘analysis interval’: the length of a sequence to be analyzed in parallel (minute).	1
reduce_res	The time interval of cardiac data to be analyzed at a reduced resolution (seconds).	0.01
pop_size [#]	As in ‘population size*’: the number of sub-sequences to analyze within each sequence in the genetic algorithm.	10
max_gen [#]	As in ‘maximum generations*’: the maximum number of rounds to repeat the selection and mutation procedures in the genetic algorithm.	20
patience [#]	The number of rounds to repeat when there are no further changes in the sub-sequence durations before terminating the genetic algorithm.	2
acf_thres	The threshold used in ACF to classify periodic and aperiodic data. Increasing this threshold may increase the power to screen off noises but also lead to greater data loss.	0.5
lr_thres	The threshold used in the linear extrapolation of when establishing the tracking index.	0.7

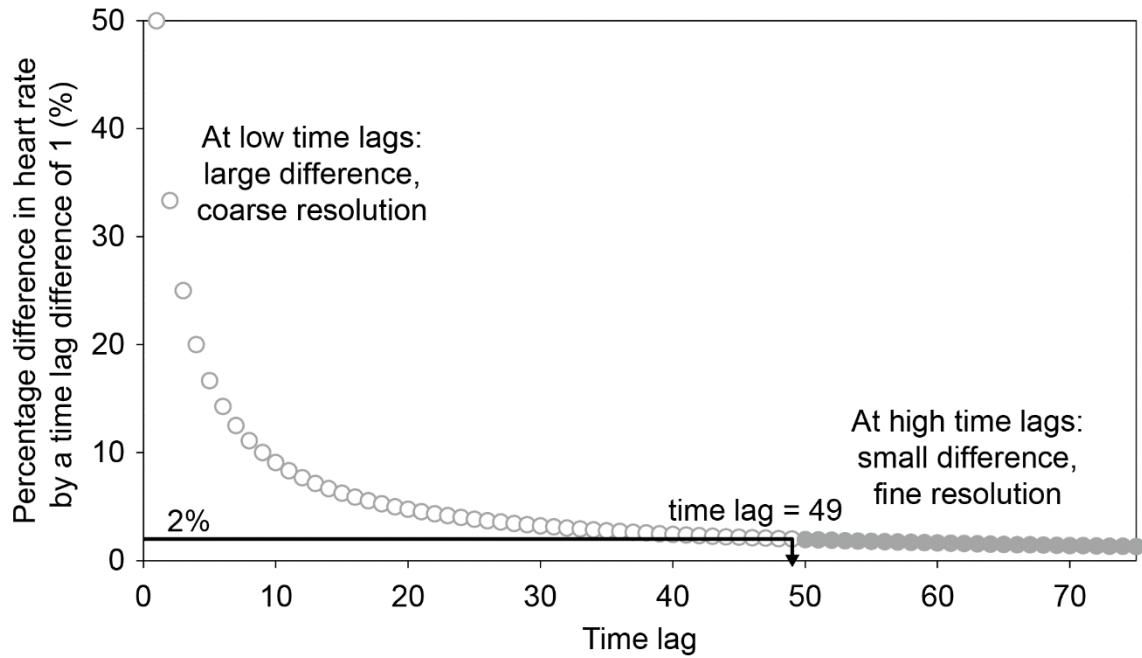
*see **Supplementary S2 of the publication** for the equivalent terms used in the typical genetic algorithm and the present study

[#]variables that are in potential tradeoff with computation time (i.e. increasing these numbers may facilitate the optimization process of the genetic algorithm but increase the computation time)

106 **Table 2** The relationship between time lag, the corresponding heart rate and
 107 resolution (as implied by the difference in heart rate by a time lag difference of 1)
 108 when analyzing cardiac data with the time interval = 0.01s. The resolution increases
 109 with time lag and the relationship is not linear (see also **Figure 3**): for the same time
 110 lag difference of 1, the difference in heart rate computed is large at small time lags,
 111 whereas the difference is small at large time lags. As a result, in this package all data
 112 are first analyzed at a reduced time interval to minimize computation time and are
 113 only re-analyzed at the finest time interval when the corresponding percentage
 114 difference is $\geq 2\%$ (i.e. for results with time lag ≤ 49 in this example).

Time lag	Period (s / beat)	Heart rate (bpm)	Difference in heart rate between time lag t and $t+1$ (bpm)	% difference
0	NA	NA	NA	NA
1	0.01	6000	3000	50.0
2	0.02	3000	1000	33.3
3	0.03	2000	500	25.0
4	0.04	1500	300	20.0
5	0.05	1200	200	16.7
6	0.06	1000	143	14.3
7	0.07	857	107	12.5
8	0.08	750	83	11.1
9	0.09	667	67	10.0
10	0.10	600	55	9.1
...				
46	0.46	130.4	2.8	2.13
47	0.47	127.7	2.7	2.08
48	0.48	125.0	2.6	2.04
49	0.49	122.4	2.4	2.00
50	0.50	120.0	2.4	1.96
51	0.51	117.6	2.3	1.92
52	0.52	115.4	2.2	1.89

115



116

117 **Figure 3** The relationship between the percentage difference in heart rate computed
 118 across a time lag difference of 1 with time lag. All data are first analyzed at the
 119 reduced resolution to minimize computation time (as indicated by the closed dots)
 120 and are only re-analyzed at the finest time interval when the percentage difference
 121 is $\geq 2\%$ (when time lag ≤ 49 ; as indicated by the open dots).

122 **Table 3** The output from function `computeHR` per channel consisting of four items,
123 with `finalsubseq` and `candidateHR` indicating the results from the genetic algorithm,
124 and `results_ACF` and `results_TI` indicating the final heart rates after evaluating the
125 candidate heart rates, checking for resolution and weighting for durations.

Variable	Content
<code>finalsubseq</code>	A list of positions and durations of the final periodic sub-sequences
<code>candidateHR</code>	A list of candidate heart rates extracted from ACF for each sub-sequence
<code>results_ACF</code>	Results obtained from evaluating the candidate heart rates of each sub-sequence based on autocorrelation values (i.e. following the “ACF + GA” approach as described in the main manuscript). Consisted of three items: 1) <code>subseqHR</code> : a list of sub-sequences and the corresponding heart rates and durations; 2) <code>weightedHR</code> : a list of final heart rates per sequence after weighting; and 3) <code>plot</code> : a plot of final heart rates against time
<code>results_TI</code>	Results obtained from evaluating the candidate heart rates of each sub-sequence using a tracking index (i.e. following the “ACF + GA + TI” approach as described in the main manuscript). Consisted of three items: 1) <code>subseqHR</code> : a list of sub-sequences and the corresponding heart rates and durations; 2) <code>weightedHR</code> : a list of final heart rates per sequence after weighting; and 3) <code>plot</code> : a plot of final heart rates against time

126

Literature cited

- Angilletta MJ (2009). *Thermal adaptation: a theoretical and empirical synthesis.* Oxford University Press.
- Hellicar AD, Rahman A, Smith DV, Smith G, McCulloch J, Andrewartha S and Morash A (2015). An algorithm for the automatic analysis of signals from an oyster heart rate sensor. *IEEE Sensors Journal* 15(8): 4480-4487.
- Huey RB and Kingsolver JG (1989). Evolution of thermal sensitivity of ectotherm performance. *Trends in Ecology & Evolution* 4(5): 131-135.
- Lau SLY (2021). Thermal extremes and variability shape the performance of rocky shore littorinids. PhD thesis, The University of Hong Kong, Hong Kong.
- Venables WN and Ripley BD (2002). *Modern applied statistics with S.* Springer.