Supplementary Data

Pairwise comparisons of xSE and cRQA measures for the simulated data (Tables S1-5).

Note: A red box indicate manipulations that significantly affected the ratio of 1:1 to other ratios. A blue box indicates manipulations that significantly affected the ratio of 1:2 to other ratios. A yellow box indicates manipulations that significantly affected the ratio of 1:3 to other ratios. A green box indicates manipulations that had an effect on the majority of ratio comparisons.

Table S1. Adjusted p-values for the pairwise comparisons of fluctuation ratio within each level of manipulation and signal fluctuation type for the simulated data for cRQA radius. Table excludes the periodic signal fluctuation type. Significantly different ($p<0.05$) comparisons are shown in bold print. Note: M = manipulation and T = signal fluctuation type.

Comparison	M: Frequency T: Chaotic	M: Frequency T: Random	M: Amplitude T: Chaotic	M: Amplitude T: Random	M: Freq/Amp T: Chaotic	M: Freq/Amp T: Random
1:1 to 1:2	**<.0001**	0.14	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:3	**<.0001**	1.0	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:4	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:3	**<.0001**	1.0	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:5	**<.0001**	0.27	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:7	**<.0001**	0.19	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:2 to 1:3	**<.0001**	0.07	**<.0001**	**<.0001**	**<.0001**	1.0
1:2 to 1:4	0.11	**<.0001**	**0.04**	1.0	0.86	1.0
1:2 to 2:3	0.63	0.11	**0.01**	**0.001**	**<.0001**	**0.004**
1:2 to 2:5	**0.03**	1.0	1.0	1.0	**<.0001**	1.0
1:2 to 2:7	**0.04**	**<.0001**	1.0	1.0	**0.04**	0.38
1:3 to 1:4	0.06	**<.0001**	**<.0001**	**<.0001**	**0.0004**	0.90
1:3 to 2:3	**0.003**	1.0	**<.0001**	**<.0001**	0.07	**0.01**
1:3 to 2:5	0.20	0.14	**<.0001**	**<.0001**	1.0	1.0
1:3 to 2:7	0.14	0.35	**<.0001**	**<.0001**	0.44	0.18
1:4 to 2:3	0.94	**<.0001**	**<.0001**	**0.0005**	**<.0001**	**0.0003**
1:4 to 2:5	1.0	**<.0001**	0.10	1.0	**0.0008**	0.75
1:4 to 2:7	1.0	**0.02**	**0.01**	1.0	0.12	0.84
2:3 to 2:5	0.68	0.21	**0.003**	**0.0001**	**0.04**	**0.03**
2:3 to 2:7	0.78	0.24	**0.04**	**0.0008**	**0.0002**	**<.0001**
2:5 to 2:7	1.0	**0.0003**	0.97	1.0	0.58	0.10

Table S2. Adjusted p-values for the pairwise comparisons of fluctuation ratio within each level of manipulation and signal fluctuation type for the simulated data for cRQA %DET. Table excludes the periodic signal fluctuation type. Significantly different ($p<0.05$) comparisons are shown in bold print. Note: M = manipulation and T = signal fluctuation type.

Comparison	M: Frequency T: Chaotic	M: Frequency T: Random	M: Amplitude T: Chaotic	M: Amplitude T: Random	M: Freq/Amp T: Chaotic	M: Freq/Amp T: Random
1:1 to 1:2	**<.0001**	0.14	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:3	**<.0001**	1.0	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:4	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:3	**<.0001**	1.0	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:5	**<.0001**	0.27	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:7	**<.0001**	0.19	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:2 to 1:3	**<.0001**	0.07	**<.0001**	**<.0001**	**<.0001**	1.0
1:2 to 1:4	0.11	**<.0001**	**0.04**	1.0	0.86	1.0
1:2 to 2:3	0.63	0.11	**0.01**	**0.001**	**<.0001**	**0.004**
1:2 to 2:5	**0.03**	1.0	1.0	1.0	**<.0001**	1.0
1:2 to 2:7	**0.04**	**<.0001**	1.0	1.0	**0.004**	0.38
1:3 to 1:4	0.06	**<.0001**	**<.0001**	**<.0001**	**0.0004**	0.90
1:3 to 2:3	**0.003**	1.0	**<.0001**	**<.0001**	0.10	0.01
1:3 to 2:5	0.20	0.14	**<.0001**	**<.0001**	1.0	1.0
1:3 to 2:7	0.14	0.35	**<.0001**	**<.0001**	0.44	0.18
1:4 to 2:3	0.94	**<.0001**	**<.0001**	**0.0005**	**<.0001**	**0.0003**
1:4 to 2:5	1.0	**<.0001**	0.10	1.0	**0.0008**	0.75
1:4 to 2:7	1.0	**0.02**	**0.009**	1.0	0.12	0.84
2:3 to 2:5	0.68	0.21	**0.003**	**0.0001**	**0.04**	**0.03**
2:3 to 2:7	0.78	0.24	**0.04**	**0.0008**	**0.0002**	**<.0001**
2:5 to 2:7	1.0	**0.0003**	1.0	1.0	0.58	0.10

Table S3. Adjusted p-values for the pairwise comparisons of fluctuation ratio within each level of manipulation and signal fluctuation type for the simulated data for cRQA Max Line. Table excludes the periodic signal fluctuation type. Significantly different ($p<0.05$) comparisons are shown in bold print. Note: M = manipulation and T = signal fluctuation type.

Comparison	M: Frequency T: Chaotic	M: Frequency T: Random	M: Amplitude T: Chaotic	M: Amplitude T: Random	M: Freq/Amp T: Chaotic	M: Freq/Amp T: Random
1:1 to 1:2	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:4	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:5	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:7	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:2 to 1:3	0.81	1.0	0.28	0.09	**<.0001**	**<.0001**
1:2 to 1:4	0.94	1.0	1.0	1.0	**0.01**	**<.0001**
1:2 to 2:3	**<.0001**	**<.0001**	0.55	0.89	**0.0002**	**<.0001**
1:2 to 2:5	0.84	1.0	1.0	1.0	**0.004**	**<.0001**
1:2 to 2:7	0.91	1.0	0.90	0.84	**0.0003**	**<.0001**
1:3 to 1:4	1.0	1.0	0.76	0.23	0.66	1.0
1:3 to 2:3	**<.0001**	**<.0001**	**0.03**	**0.003**	**<.0001**	**<.0001**
1:3 to 2:5	1.0	1.0	0.64	0.16	0.83	0.49
1:3 to 2:7	1.0	1.0	0.93	0.74	1.0	1.0
1:4 to 2:3	**<.0001**	**<.0001**	0.15	0.64	**<.0001**	**<.0001**
1:4 to 2:5	1.0	1.0	1.0	1.0	1.0	0.33
1:4 to 2:7	1.0	1.0	1.0	0.98	0.92	1.0
2:3 to 2:5	**<.0001**	**<.0001**	0.22	0.76	**<.0001**	**<.0001**
2:3 to 2:7	**<.0001**	**<.0001**	0.06	0.17	**<.0001**	**<.0001**
2:5 to 2:7	1.0	1.0	1.0	0.94	0.98	0.50

Table S4. Adjusted p-values for the pairwise comparisons of fluctuation ratio within each level of manipulation and signal fluctuation type for the simulated data for cRQA Mean Line. Table excludes the periodic signal fluctuation type. Significantly different ($p<0.05$) comparisons are shown in bold print. Note: M = manipulation and T = signal fluctuation type.

Comparison	M: Frequency T: Chaotic	M: Frequency T: Random	M: Amplitude T: Chaotic	M: Amplitude T: Random	M: Freq/Amp T: Chaotic	M: Freq/Amp T: Random
1:1 to 1:2	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:4	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**1.0000**
1:1 to 2:5	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:7	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:2 to 1:3	**0.002**	0.41	**<.0001**	**<.0001**	**<.0001**	0.32
1:2 to 1:4	0.97	0.67	**0.01**	**0.007**	0.25	1.0
1:2 to 2:3	**<.0001**	**0.0002**	**0.002**	**<.0001**	**<.0001**	**<.0001**
1:2 to 2:5	0.07	0.97	**0.001**	**0.003**	**<.0001**	0.05
1:2 to 2:7	0.98	0.41	**0.001**	**<.0001**	**<.0001**	0.99
1:3 to 1:4	**0.04**	1.0	**0.005**	**<.0001**	**<.0001**	0.45
1:3 to 2:3	**<.0001**	0.09	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:3 to 2:5	0.89	0.92	0.05	**<.0001**	0.87	0.97
1:3 to 2:7	**0.03**	1.0	0.05	**0.0005**	0.21	0.75
1:4 to 2:3	**<.0001**	**0.03**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:4 to 2:5	0.42	0.99	0.99	1.0	**0.005**	0.08
1:4 to 2:7	1.0	1.0	0.99	0.84	0.11	1.0
2:3 to 2:5	**<.0001**	**0.004**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
2:3 to 2:7	**<.0001**	0.09	**<.0001**	**<.0001**	**<.0001**	**<.0001**
2:5 to 2:7	0.39	0.91	1.0	0.95	0.90	0.22

Table S5. Adjusted p-values for the pairwise comparisons of fluctuation ratio within each level of manipulation and signal fluctuation type for the simulated data for cRQA Entropy. Table excludes the periodic signal fluctuation type. Significantly different ($p<0.05$) comparisons are shown in bold print. Note: M = manipulation and T = signal fluctuation type.

Comparison	M: Frequency T: Chaotic	M: Frequency T: Random	M: Amplitude T: Chaotic	M: Amplitude T: Random	M: Freq/Amp T: Chaotic	M: Freq/Amp T: Random
1:1 to 1:2	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 1:4	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	0.51
1:1 to 2:5	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:1 to 2:7	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:2 to 1:3	**<.0001**	0.62	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:2 to 1:4	0.29	0.91	**0.0014**	**0.006**	**0.005**	0.06
1:2 to 2:3	**<.0001**	**<.0001**	0.29	**<.0001**	**<.0001**	**<.0001**
1:2 to 2:5	**<.0001**	0.94	**<.0001**	**0.005**	**<.0001**	**<.0001**
1:2 to 2:7	**0.004**	0.88	**<.0001**	**<.0001**	**<.0001**	0.06
1:3 to 1:4	**<.0001**	1.0	**<.0001**	**<.0001**	**<.0001**	0.26
1:3 to 2:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:3 to 2:5	0.16	1.0	**<.0001**	**<.0001**	**0.01**	0.94
1:3 to 2:7	**<.0001**	1.0	**<.0001**	**<.0001**	**<.0001**	0.26
1:4 to 2:3	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
1:4 to 2:5	**<.0001**	1.0	0.95	1.0	**<.0001**	**0.02**
1:4 to 2:7	0.63	1.0	0.98	0.55	**<.0001**	1.00
2:3 to 2:5	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
2:3 to 2:7	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**	**<.0001**
2:5 to 2:7	**<.0001**	1.0	1.0	0.58	0.29	**0.02**

Matlab script used to calculate Cross Sample Entropy (xSE)

```
function xSE = xSamp_Ent(x,y,m,R,norm)
% xSE = xSamp_Ent(x,y,m,R,norm)

% Function to calculate cross sample entropy for 2 data series using the
% method described by Richman and Moorman (2000). J McCamley - Sept, 2015;

% Inputs
% x - first data series
% y - second data series
% m - vector length for matching (usually 2 or 3)
% R - R tolerance to find matches (as a proportion of the average of the
%     SDs of the data sets, usually between 0.15 and 0.25)
% norm - normalization to perform
%       1 = max rescale/unit interval (data ranges in value from 0 - 1)
%       2 = mean/Zscore (used when data is more variable or has outliers)
%       normalized data has SD = 1. This is best for cross sample entropy.
%       Set to any value other than 1 or 2 to not normalize/rescale the
%       data

% Check both sets of data are the same length
xl = length(x);
yl = length(y);
if xl ~= yl
    disp('The data series need to be the same length!')
end
N = length(x);
% normalize the data ensure data fits in the same "space"
```

```
55  if norm == 1 %normalize data to have a range 0 - 1
56      xn = (x - min(x))/(max(x) - min(x));
57      yn = (y - min(y))/(max(y) - min(y));
58      r = R * ((std(xn)+std(yn))/2);
59  elseif norm == 2 % normalize data to have a SD = 1, and mean = 0
60      xn = (x - mean(x))/std(x);
61      yn = (y - mean(y))/std(y);
62      r = R;
63  else disp('These data will not be normalized')
64  end
65
66  fprintf(' 00.00%%');
67  for i = 1:N-m
68      for j = 1:N-m
69          for k = 1:m+1;
70              dij(k) = abs(xn(i+k-1)-yn(j+k-1));
71          end
72          di(j) = max(dij(1:m));
73          di1(j) = max(dij(1:m+1));
74      end
75      d = find(di<=r); % find the vectors of length 'm' that are less than "r"
76  distant from one another
77      d1 = find(di1<=r); % find the vectors of length 'm+1' that are less than
78  "r" distant from one another
79      nm = length(d);
80      Bm(i) = nm/(N-m);
81      nm1 = length(d1);
82      Am(i) = nm1/(N-m);
```

```
83        if mod(i,1000) == 0
84            fprintf(repmat('\b',1,7));
85            fprintf('%3.0d',floor(i/(N-m)*100));
86            fprintf('.%1.0f',floor(mod((i/(N-m))*100,1)*10));
87            fprintf('%1.0f',floor(mod((i/(N-m))*1000,1)*10));
88            fprintf('%%');
89        end
90    end
91    fprintf(repmat('\b',1,7));
92    fprintf('100.00%%');
93    fprintf('\r');
94
95    Bmr = sum(Bm)/(N-m);
96    Amr = sum(Am)/(N-m);
97
98    xSE = -log(Amr/Bmr);
99    end
100
101
102
103
104
105
106
107
```

108 Matlab script used to calculate cross recurrence quantification analysis (cRQA)

```
function [RP, RESULTS]=RQA(DATA,TYPE,EMB,DEL,ZSCORE,NORM,SETPARA,SETVALUE)
%  [RP, RESULTS]=RQA(DATA,TYPE,EMB,DEL,ZSCORE,NORM,SETPARA,SETVALUE)
%
%  Computes a recurrence plot for either recurrence quantification analysis
%  (RQA), cross recurrence quantification analysis (cRQA), jount recurrence
%  quantification analysis (jRQA), or multidimensional recurrence
%  quantification analysis (mdRQA). Either radius or target recurrence can
%  be set.
%
%
%  Inputs:
%
%  DATA is a double-variable with each dimension of the to-be-analyzed
%  signal as a row of numbers in a separate column. If too many columns are
%  present for the TYPE of analysis selected, the other columns will be
%  ignored (i.e. for 'cRQA' only the first two columns will be used).
%
%  TYPE is a string indicating which type of RQA to run (i.e. 'RQA',
%  'cRQA', 'jRQA', 'mdRQA').
%  The default value is TYPE = 'RQA'.
%
%  EMB is the number of embedding dimensions (i.e., EMB = 1 would be no
%  embedding via time-delayed surrogates, just using the provided number of
%  colums as dimensions.
%  The default value is EMB = 1.
%
%  DEL is the delay parameter used for time-delayed embedding (if EMB > 1).
%  The default value is DEL = 1.
%
```

```
138  %  ZSCORE indicates, whether the data (i.e., the different columns of DATA,
139  %  being the different signals or dimensions of a signal) should be
140  %  z-scored before performing MdRQA:
141  %    0 - no z-scoring of DATA
142  %    1 - z-score columns of DATA
143  %  The default value is ZSCORE = 0.
144  %
145  %  NORM is the type of norm by with the phase-space is normalized. The
146  %  following norms are available:
147  %    'euc' - Euclidean distance norm
148  %    'max' - Maximum distance norm
149  %    'min' - Minimum distance norm
150  %    'non' - no normalization of phase-space
151  %  The default value is NORM = 'non'.
152  %
153  %  SETPARA is the parameter which you would like to set a target value for
154  %  the recurrence plot (i.e. 'radius' or 'recurrence').
155  %  The default value is SETPARA = 'radius'.
156  %
157  %  SETVALUE sets the value of the selected parameter. If SETVALUE = 1, then
158  %  the radius will be set to 1 if SETPARA = 'radius' or the radius will be
159  %  adjusted until the recurrence is equal to 1 if SETPARA = 'recurrence'.
160  %  The default value if SETPARA = 'radius' is 1.
161  %  The default value if SETPARA = 'recurrence' is 2.5.
162  %
163  %
164  % Outputs:
165  %
166  %  RP is a matrix holding the resulting recurrence plot.
```

```
%
%  RESULTS is a structure holding the following recurrence variables:
%    1.  DIM    - dimension of the input data (used for mdRQA)
%    2.  EMB    - embedding dimension used in the calculation of the
%                 distance matrix
%    3.  DEL    - time lag used in the calculation of the distance matrix
%    4.  RADIUS - radius used for the recurrence plot
%    5.  NORM   - type of normilization used for the distance matrix
%    6.  ZSCORE - whether or not zscore was used
%    7.  Size   - size of the recurrence plot
%    8.  %REC   - percentage of recurrent points
%    9.  %DET   - percentage of diagonally adjacent recurrent points
%    10. MeanL  - average length of adjacent recurrent points
%    11. MaxL   - maximum length of diagonally adjacent recurrent points
%    12. EntrL  - Shannon entropy of distribution of diagonal lines
%    13. %LAM   - percentage of vertically adjacent recurrent points
%    14. MeanV  - average length of diagonally adjacent recurrent points
%    15. MaxV   - maximum length of vertically adjacent recurrent points
%    16. EntrV  - Shannon entropy of distribution of vertical lines
%    17. EntrW  - Weighted entropy of distribution of vertical weighted sums
%
%
% Reference:
%
% Wallot, S., Roepstorff, A., & Monster, D. (2016). Multidimensional
% Recurrence Quantification Analysis (MdRQA) for the analysis of
% multidimensional time-series: A software implementation in MATLAB and its
% application to group-level data in joint action. Frontiers in Psychology,
% 7, 1835. http://dx.doi.org/10.3389/fpsyg.2016.01835
```

```
196 %
197 % Eroglu, D., Peron, T. K. D., Marwan, N., Rodrigues, F. A., Costa, L. D.
198 % F., Sebek, M., ... & Kurths, J. (2014). Entropy of weighted recurrence
199 % plots. Physical Review E, 90(4), 042919.
200
201
202 % Version:
203 %
204 % v1.0, 28. July 2016
205 % by Sebastian Wallot, Max Planck Insitute for Empirical Aesthetics,
206 Frankfurt, Germany
207 %  & Dan M?nster, Aarhus University, Aarhus, Denmark
208 %
209 % v1.1, 06. July 2017
210 % by Will Denton (21denton@gmail.com), Troy Rand (troyrand@gmail.com), and
211 %  Casey Wiens (cwiens32@gmail.com), Biomechanics Research Building,
212 %  University of Nebraska at Omaha. Changes include cleaning up some
213 %  errors, making the default input arguments function correctly,
214 %  incorporating other types of RQA (e.g. RQA, CRQA, JRQA), allowing %REC
215 %  to be set instead of radius, incorporating weighted recurrence plots,
216 %  and adding weighted entropy.
217
218 %% Set default parameters if no input exists
219 % If SETPARA is not specified, set to 'radius'
220 if nargin < 7 || isempty(SETPARA)
221     SETPARA = 'radius';
222 end
223
```

```
% If SETVALUE is not specified, set to 1 if radius is set or 2.5 if perRec is
set
if nargin < 8 || isempty(SETVALUE)
    switch lower(SETPARA)
        case {'radius','rad', 1}
            radius = 1;
            runSetRad = 0;
        case {'perrec','recurrence', 2}
            radiusStart = 0.01;
            radiusEnd = 0.5;
            runSetRad = 1;
            SETVALUE = 2.5;
    end
else
    switch lower(SETPARA)
        case {'radius','rad', 1}
            radius = SETVALUE;
            runSetRad = 0;
        case {'perrec','recurrence', 2}
            radiusStart = 0.01;
            radiusEnd = 0.5;
            runSetRad = 1;
    end
end

% If NORM is not specified, set to 'non'
if nargin < 6 || isempty(NORM)
    NORM = 'non';
```

```
252 end
253
254 % If ZSCORE is not specified, set to 0
255 if nargin < 5 || isempty(ZSCORE)
256     ZSCORE = 0;
257 end
258
259 % If DEL is not specified, set to 1
260 if nargin < 4 || isempty(DEL)
261     DEL = 1;
262 end
263
264 % If EMB is not specified, set to 1
265 if nargin < 3 || isempty(EMB)
266     EMB = 1;
267 end
268
269 % If EMB is not specified, set to 1
270 if nargin < 2 || isempty(TYPE)
271     TYPE = 'RQA';
272 end
273
274 % If z score is selected then z score the data
275 if ZSCORE
276     DATA = zscore(DATA);
277 end
278
```

```
%% Begin code
% Set DIM and select proper column(s) of data if too many exist
c = size(DATA,2);
switch upper(TYPE)
    case 'RQA'
        DIM = 1;
        if c > 1
            DATA = DATA(:,1);
            warning('More than one column of data. Only using first
column.');
        end
    case {'CRQA','CROSS'}
        DIM = 2;
        if c > 2
            DATA = DATA(:,1:2);
            warning('More than two columns of data. Only using first two
columns.');
        end
    case {'JRQA','JOINT'}
        DIM = c;
        if c < 2
            error('Input data must have at least two columns.');
        end
    case {'MDRQA','MD','MULTI'}
        DIM = c;
end

% Embed the data
```

```
307 if EMB > 1
308     for i = 1:EMB
309         tempDATA(1:length(DATA)-(EMB-1)*DEL,1+DIM*(i-1):DIM*i) = DATA(1+(i-
310 1)*DEL:length(DATA)-(EMB-i)*DEL,:);
311     end
312     DATA = tempDATA;
313     clear tempDATA
314 end
315
316 % Calculate distance matrix based on the type of RQA
317 switch upper(TYPE)
318     case 'RQA'
319         a{1}=pdist2(DATA,DATA);
320         a{1}=abs(a{1})*-1;
321     case {'CRQA','CROSS'}
322         a{1}=pdist2(DATA(:,1:DIM:end),DATA(:,2:DIM:end));
323         a{1}=abs(a{1})*-1;
324     case {'JRQA','JOINT'}
325         for i = 1:c
326             a{i}=pdist2(DATA(:,i:DIM:end),DATA(:,i:DIM:end));
327             a{i}=abs(a{i})*-1;
328         end
329     case {'MDRQA','MD','MULTI'}
330         a{1}=pdist2(DATA,DATA);
331         a{1}=abs(a{1})*-1;
332 end
333
334 % Normalize distance matrix
```

```
335 if contains(NORM, 'euc')
336     for i = 1:length(a)
337         b = mean(a{i}(a{i}<0));
338         b = -sqrt(abs(((b^2)+2*(DIM*EMB))));
339         a{i} = a{i}/abs(b);
340     end
341 elseif contains(NORM, 'min')
342     for i = 1:length(a)
343         b = max(a{i}(a{i}<0));
344         a{i} = a{i}/abs(b);
345     end
346 elseif contains(NORM, 'max')
347     for i = 1:length(a)
348         b = min(a{i}(a{i}<0));
349         a{i} = a{i}/abs(b);
350     end
351 elseif contains(NORM, 'non')
352     % do nothing
353 else
354     error('No appropriate norm parameter specified.');
355 end
356
357 % Create weighted recurrence plot
358 wrp = a;
359 for i = 1:size(a,2)-1
360     wrp{i+1} = wrp{i}.*wrp{i+1};
361 end
362 if i
```

```
363         wrp = -(abs(wrp{i+1})).^(1/(i+1));
364     end
365     if iscell(wrp)
366         wrp = wrp{1};
367     end
368
369     % Calculate recurrence plot
370     switch lower(SETPARA)
371         case {'radius','rad', 1}
372             [perRec, diag_hist, vertical_hist,A] = recurrenceMethod(a,radius);
373         case {'perrec','recurrence', 2}
374             [perRec, diag_hist, vertical_hist, radius, A] =
375     setRadius(radiusStart);
376     end
377
378     %% Calculate RQA variabes
379     RESULTS.DIM = DIM;
380     RESULTS.EMB = EMB;
381     RESULTS.DEL = DEL;
382     RESULTS.RADIUS = radius;
383     RESULTS.NORM = NORM;
384     RESULTS.ZSCORE = ZSCORE;
385     RESULTS.Size=length(A);
386     RESULTS.REC = perRec;
387     if RESULTS.REC > 0
388         RESULTS.DET=100*sum(diag_hist(diag_hist>1))/sum(diag_hist);
389         RESULTS.MeanL=mean(diag_hist(diag_hist>1));
390         RESULTS.MaxL=max(diag_hist);
```

```
391
392 [count,bin]=hist(diag_hist(diag_hist>1),min(diag_hist(diag_hist>1)):max(diag_
393 hist));
394     total=sum(count);
395     p=count./total;
396     del=find(count==0); p(del)=[];
397     RESULTS.EntrL=-sum(p.*log2(p));
398     RESULTS.LAM=100*sum(vertical_hist(vertical_hist>1))/sum(vertical_hist);
399     RESULTS.MeanV=mean(vertical_hist(vertical_hist>1));
400     RESULTS.MaxV=max(vertical_hist);
401
402 [count,bin]=hist(vertical_hist(vertical_hist>1),min(vertical_hist(vertical_hi
403 st>1)):max(vertical_hist));
404     total=sum(count);
405     p=count./total;
406     del=find(count==0); p(del)=[];
407     RESULTS.EntrV=-sum(p.*log2(p));
408     RESULTS.EntrW=RQA_WeightedEntropy( wrp );
409 else
410     RESULTS.DET=NaN;
411     RESULTS.MeanL=NaN;
412     RESULTS.MaxL=NaN;
413     RESULTS.EntrL=NaN;
414     RESULTS.LAM=NaN;
415     RESULTS.MeanV=NaN;
416     RESULTS.MaxV=NaN;
417     RESULTS.EntrV=NaN;
418     RESULTS.EntrW=NaN;
419 end
```

```
RP=imrotate(1-A,90);

%% PLOT
scrsz = get(0,'ScreenSize');
f = figure('Position',[scrsz(3)/4 scrsz(4)/4 scrsz(3)/3 scrsz(4)/2]);tabgp =
uitabgroup(f);
binary = uitab(tabgp,'Title','Binary');
heatmap = uitab(tabgp,'Title','Heatmap');
% Binary Plot (Tab 1)
a1 = axes('Parent',binary,'Position', [0 0 1 1], 'Visible', 'off');
ax(1) = axes('Parent',binary,'Position',[.375 .35 .58 .6], 'FontSize', 8);
imagesc(ax(1),RP); colormap(gray);
title(['DIM = ', num2str(DIM), '; EMB = ',num2str(EMB), '; DEL = ',
num2str(DEL), '; RAD = ', num2str(radius), '; NORM = ',num2str(NORM), ';
ZSCORE = ',num2str(ZSCORE)],'FontSize',8)
xlabel('X(i)','Interpreter','none', 'FontSize', 10);
ylabel('Y(j)','Interpreter','none', 'FontSize', 10);
set(gca,'XTick',[ ]);
set(gca,'YTick',[ ]);
switch upper(TYPE)
    case {'RQA','MDRQA','MD','MULTI'}
        ax(2) = axes('Parent',binary,'Position',[.375 .1 .58 .15],
'FontSize', 8);
        plot(1:length(DATA(:,1)), DATA(:,1), 'k-');
        xlim([1 length(DATA(:,1))]);
        ax(3) = axes('Parent',binary,'Position',[.09 .35 .15 .6], 'FontSize',
8);
        plot(DATA(:,1), 1:length(DATA(:,1)), 'k-');
```

```
448             ylim([1 length(DATA(:,1))]);
449         case {'CRQA','CROSS'}
450             ax(2) = axes('Parent',binary,'Position',[.375 .1 .58 .15],
451     'FontSize', 8);
452             plot(1:length(DATA(:,1)), DATA(:,1), 'k-');
453             xlim([1 length(DATA(:,1))]);
454             ax(3) = axes('Parent',binary,'Position',[.09 .35 .15 .6], 'FontSize',
455     8);
456             plot(DATA(:,2), 1:length(DATA(:,2)), 'k-');
457             ylim([1 length(DATA(:,1))]);
458         case {'JRQA','JOINT'}
459             for i = 1:c
460                 ax(2) = axes('Parent',binary,'Position',[.375 .1 .58 .15],
461     'FontSize', 8);
462                 plot(1:length(DATA(:,1)), DATA(:,i),'k-');
463                 xlim([1 length(DATA(:,1))]);
464                 ax(3) = axes('Parent',binary,'Position',[.09 .35 .15 .6],
465     'FontSize', 8);
466                 plot(DATA(:,1), 1:length(DATA(:,i)),'k-');
467                 ylim([1 length(DATA(:,1))]);
468             end
469     end
470     set(gcf, 'CurrentAxes', a1);
471     str(1) = {['%REC = ', sprintf('%.2f',RESULTS.REC)]};
472     text(.1, 0.27, str, 'FontSize', 8, 'Color', 'k');
473     str(1) = {['%DET = ', sprintf('%.2f',RESULTS.DET)]};
474     text(.1, .24, str, 'FontSize', 8, 'Color', 'k');
475     str(1) = {['MaxL = ', sprintf('%.0f',RESULTS.MaxL)]};
476     text(.1, .21, str, 'FontSize', 8, 'Color', 'k');
```

```
str(1) = {['MeanL = ', sprintf('%.2f',RESULTS.MeanL)]};
text(.1, .18, str, 'FontSize', 8, 'Color', 'k');
str(1) = {['EntrL = ', sprintf('%.2f',RESULTS.EntrL)]};
text(.1, .15, str, 'FontSize', 8, 'Color', 'k');
str(1) = {['%LAM = ', sprintf('%.2f',RESULTS.LAM)]};
text(.1, .12, str, 'FontSize', 8, 'Color', 'k');
str(1) = {['MaxV = ', sprintf('%.0f',RESULTS.MaxV)]};
text(.1, .09, str, 'FontSize', 8, 'Color', 'k');
str(1) = {['MeanV = ', sprintf('%.2f',RESULTS.MeanV)]};
text(.1, .06, str, 'FontSize', 8, 'Color', 'k');
str(1) = {['EntrV = ', sprintf('%.2f',RESULTS.EntrV)]};
text(.1, .03, str, 'FontSize', 8, 'Color', 'k');

% Heatmap Plot (Tab 2)
a2 = axes('Parent',heatmap,'Position', [0 0 1 1], 'Visible', 'off');
ax(4) = axes('Parent',heatmap,'Position',[.375 .35 .58 .6], 'FontSize', 8);
imagesc(ax(4),imrotate(-1*wrp,90));
title(['DIM = ', num2str(DIM), '; EMB = ',num2str(EMB), '; DEL = ',
num2str(DEL), '; RAD = ', num2str(radius), '; NORM = ',num2str(NORM), ';
ZSCORE = ',num2str(ZSCORE)],'FontSize',8)
xlabel('X(i)','Interpreter','none', 'FontSize', 10);
ylabel('Y(j)','Interpreter','none', 'FontSize', 10);
set(gca,'XTick',[ ]);
set(gca,'YTick',[ ]);
switch upper(TYPE)
    case {'RQA','MDRQA','MD','MULTI'}
        ax(5) = axes('Parent',heatmap,'Position',[.375 .1 .58 .15],
'FontSize', 8);
```

```
505             plot(1:length(DATA(:,1)), DATA(:,1), 'k-');
506             xlim([1 length(DATA(:,1))]);
507             ax(6) = axes('Parent',heatmap,'Position',[.09 .35 .15 .6],
508     'FontSize', 8);
509             plot(DATA(:,1), 1:length(DATA(:,1)), 'k-');
510             ylim([1 length(DATA(:,1))]);
511         case {'CRQA','CROSS'}
512             ax(5) = axes('Parent',heatmap,'Position',[.375 .1 .58 .15],
513     'FontSize', 8);
514             plot(1:length(DATA(:,1)), DATA(:,1), 'k-');
515             xlim([1 length(DATA(:,1))]);
516             ax(6) = axes('Parent',heatmap,'Position',[.09 .35 .15 .6],
517     'FontSize', 8);
518             plot(DATA(:,2), 1:length(DATA(:,2)), 'k-');
519             ylim([1 length(DATA(:,1))]);
520
521         case {'JRQA','JOINT'}
522             for i = 1:c
523                 ax(5) = axes('Parent',heatmap,'Position',[.375 .1 .58 .15],
524     'FontSize', 8);
525                 plot(1:length(DATA(:,1)), DATA(:,i),'k-');
526                 xlim([1 length(DATA(:,1))]);
527                 ax(6) = axes('Parent',heatmap,'Position',[.09 .35 .15 .6],
528     'FontSize', 8);
529                 plot(DATA(:,1), 1:length(DATA(:,i)),'k-');
530                 ylim([1 length(DATA(:,1))]);
531             end
532     end
```

```
set(gcf, 'CurrentAxes', a2);
str(1) = {['EntrW = ', sprintf('%.2f',RESULTS.EntrW)]};
text(.1, 0.27, str, 'FontSize', 8, 'Color', 'k');
linkaxes(ax([1,4]),'xy');
linkaxes(ax([1,2,4,5]),'x');
linkaxes(ax([1,3,4,6]),'y');

%% Function for setting radius to achieve a certain recurrence
    function [perRec, diag_hist, vertical_hist, radiusFinal,A] =
setRadius(radius)
        % Find the radius to provide target % recurrence
        [perRec, ~, ~, ~] = recurrenceMethod(a,radius);
        while perRec == 0 || perRec > 2.5
            % if radius is too small
            display('Minimum radius has been adjusted...');
            if perRec == 0
                radius = radius*2;
            elseif perRec > SETVALUE
                radius = radius / 1.5;
            end
            [perRec, ~, ~, ~] = recurrenceMethod(a,radius);
        end

        [perRec, ~, ~, ~] = recurrenceMethod(a,radiusEnd);
        while perRec < SETVALUE
            % if radiusEnd is too large
            display('Maximum radius has been increased...');
            radiusEnd = radiusEnd*2;
```

```
561             [perRec, ~, ~, ~] = recurrenceMethod(a,radiusEnd);
562         end
563
564         % Search for radius with target % recurrence
565         wb = waitbar(0,['Finding radius to give %REC = ',num2str(SETVALUE), '
566 Please wait...']);    % create wait bar to display progress
567         lv = radius;    % set low value
568         hv = radiusEnd; % set high value
569         target = SETVALUE;  % designate what percent recurrence is wanted
570         iter = 20; % Number of iterations to find radius
571         for  i1 = 1:iter
572             mid(i1) = (lv(i1)+hv(i1))/2;   % find midpoint between hv and lv
573             rad(i1) = mid(i1);    % new radius for this iteration
574             %Compute recurrence matrix
575             [perRec, diag_hist, vertical_hist,A] = recurrenceMethod(a,
576 rad(i1));
577
578             perRecIter(i1) = perRec;  % set percent recurrence
579
580             if perRecIter(i1) < target
581                 % if percent recurrence is below target percent recurrence
582                 hv(i1+1) = hv(i1);
583                 lv(i1+1) = mid(i1);
584             else
585                 % if percent recurrence is above or equal to target percent
586 recurrence
587                 lv(i1+1) = lv(i1);
588                 hv(i1+1) = mid(i1);
```

```
589             end
590             waitbar(i1/iter,wb); % update wait bar
591         end
592
593         close(wb)
594         perRecFinal = perRecIter(end);  % set final percent recurrence
595         radiusFinal = rad(end);      % set radius for final percent
596 recurrence
597         disp(['% recurrence = ',num2str(perRecFinal),', radius =
598 ',num2str((radiusFinal))])
599     end
600
601     function [perRec, diag_hist, vertical_hist, A] = recurrenceMethod(A,
602 radius)
603         if ~iscell(A)
604             A = {A};
605         end
606         for i2 = 1:length(A)
607             A{i2} = A{i2}+radius;
608             A{i2}(A{i2} >= 0) = 1;
609             A{i2}(A{i2} < 0) = 0;
610         end
611
612         if length(A) > 1
613             for i3 = 1:length(A)-1
614                 A{i3+1} = A{i3}.*A{i3+1};
615             end
616             A = A{i3+1};
```

```
617         else
618             A = A{1};
619         end
620
621         diag_hist = [];
622         vertical_hist = [];
623         for i4 = -(length(DATA)-1):length(DATA)-1 % caluculate diagonal line
624 distribution
625             C=diag(A,i4);
626             % bwlabel is taking each diagonal line and looking for the 1's,
627 it will
628             % return increasing numbers for each new instance of 1's, for
629 example
630             % the input vector [0 1 1 0 1 0 1 1 0 0 1 1 1] will return
631             %                  [0 1 1 0 2 0 3 3 0 0 4 4 4]
632             d=bwlabel(C,8);
633             % tabulate counts the instances of each integer, therefore the
634 line
635             % lengths
636             d=tabulate(d);
637             if d(1,1)==0
638                 d=d(2:end,2);
639             else
640                 d=d(2);
641             end
642             % diag_hist is creating one long array of all of the line lengths
643 for
644             % all of the diagonals
```

```
645             diag_hist(length(diag_hist)+1:length(diag_hist)+length(d))=d;
646         end
647
648         % This removes the line of identity in RQA, jRQA, and mdRQA
649         if ~contains(upper(TYPE),{'CROSS','CRQA'})
650             diag_hist=diag_hist(diag_hist<max(diag_hist));
651             if isempty(diag_hist)
652                 diag_hist=0;
653             end
654         end
655
656         for i5=1:length(DATA) % calculate vertical line distribution
657             C=(A(:,i5));
658             v=bwlabel(C,8);
659             v=tabulate(v);
660             if v(1,1)==0
661                 v=v(2:end,2);
662             else
663                 v=v(2);
664             end
665
666 vertical_hist(length(vertical_hist)+1:length(vertical_hist)+length(v))=v;
667         end
668
669         % Calculate percent recurrence
670         if ~contains(upper(TYPE),{'CROSS','CRQA'})
671             perRec = 100*(sum(sum(A))-length(A))/(length(A)^2-length(A));
672         else
```

```
673                 perRec = 100*(sum(sum(A)))/(length(A)^2);
674             end
675         end
676     %% Calculate entropy of weighted recurrence plot
677         function [ Swrp ] = RQA_WeightedEntropy( WRP )
678             N = length(WRP);
679             for j = 1:N
680                 si(j) = sum(WRP(:,j));
681             end
682             mi = min(si);
683             ma = max(si);
684             m = (ma - mi)/49;
685             I = 1;
686             S = sum(si);
687             for s = mi:m:ma
688                 P = sum( si(si >= s & si < (s+m)) );
689                 p1( I ) = P / S;
690                 I = I+1;
691             end
692             for I = 1:length(p1)
693                 pp(I) = (p1(I)*log(p1(I)));
694             end
695             pp(isnan(pp)) = 0;
696             Swrp = -1*(sum(pp));
697         end
698     end
```

Matlab script used to calculate number of right heel strikes in each breath

```
function [ DRPss,freq ] = drp( B, EI, W, saveFigs, outputFilename )
%   DRP is a function to give frequency and phasic relationships between
%   two time series.
%
%   Last updated by Will Denton: 2016-04-12
%   Contact information: 21denton@gmail.com
%
%   [ DRPss,freq ] = drp( B, W, saveFigs, outputFilename ) reads vectors
%   B and W, two time series. DRP figures are output if saveFigs = 'save'
%   using the name input as outputFilename. DRPss outputs t, n, and T
%   values as well as the DRP value calculated from the equation t/T*360°,
%   where t is the distance from the peak of the low frequency oscillator to
%   a previous peak of the higher frequency oscillator within the same
%   cycle and T is the distance between the same peak of the higher
%   frequency oscillator and the following peak of the higher frequency
%   oscillator. The output variable freq gives the percentage of
%   frequencies spent at integer and half integer ratios.
%
%   Two consecutive cycles (of the lower frequency oscillator) must be at
%   the same frequency to be counted as an integer ratio. To be counted as
%   a half integer ratio, the frequency must bounce back and forth between
%   two frequencies for at least three cycles (ex. 2 1 2 ==> 1.5 1.5 1.5).
%   If the frequency is only seen in one consecutive cycle, it is
%   considered as non-coupled and given a value of 0.

    [~,HS] = findpeaks(W);
```

```
727
728         %%
729         F1 = figure('units','normalized','outerposition',[0 0 1 1]);
730         tgroup = uitabgroup('Parent', F1);
731         tab1 = uitab('Parent', tgroup, 'Title', 'DRP');
732         tab2 = uitab('Parent', tgroup, 'Title', 'Return Map');
733         tab3 = uitab('Parent', tgroup, 'Title', 'Rose Plot');
734
735         %% Tab 1
736         a1 = axes('parent', tab1);
737         subplot(3,2,1); plot(B,'k'); hold on;
738         for i = 1:length(HS)
739             plot([HS(i),HS(i)],[min(B),max(B)],'k');
740         end
741         set(gca,'XLim',[0 length(B)]);
742         set(gca,'YLim',[mean(B)-0.5*std(B) max(B)]);
743         scatter(EI,B(EI),'k','filled');
744
745         %% DRP calculation:
746         %% Compute tiR(p)
747         clc; i = 2; back = 0;
748         for i1 = 1:length(HS)
749             if HS(i1) < EI(1)
750                 display('Discrete point discarded ...');
751                 back = back+1;
752             elseif HS(i1) < EI(i)
753                 tiRp(i1-back) = EI(i) - HS(i1);
754             else
```

```
            if i < length(EI)
                i = i+1;
                tiRp(i1-back) = EI(i) - HS(i1);
            end
        end
    end
    HS = HS(back+1:end);
    HS = HS(HS <= EI(end));

    %% Compute Ti
    Ti = diff(HS);

    %% Find n's
    for i = 1:length(EI) - 1
        holder = 0;
            for f = 1:length(HS)-1
                if HS(f) >= EI(i) && HS(f) < EI(i+1)
                    holder = holder + 1;
                end
            end
        n(i,1) = holder-1;
    end

    %% Discrete relative phase
    i = 1;
    DRPss(1,:) = {'t','n','T','DRP'};
    for i1 = 1:length(Ti)
        DRP(i1,1) = tiRp(i1)/Ti(i1)*360;
```

```
783             DRPss{i+1,1} = tiRp(i1);
784             DRPss{i+1,3} = Ti(i1);
785             DRPss{i+1,4} = DRP(i);
786             i = i+1;
787         end
788
789         i = 1;
790         for k = 1:length(n)
791             for i1 = 0:n(k)
792                 DRPss{i+1,2} = n(k) - i1;
793                 i = i+1;
794             end
795         end
796
797         subplot(3,2,3:4); plot(DRP,'k-o'); hold on;
798         maxDRP = round(max(DRP)/360)+1;
799         for i = 1:maxDRP
800             plot([0,length(DRP)],[360*(i-1),360*(i-1)],'k');
801             for i1 = 1:length(DRP)
802                 if DRP(i1) > 360*(i-1) && DRP(i1) <= 360*i
803                     count(i1,1) = i;
804                 else
805                 end
806             end
807         end
808         set(gca,'XLim',[1 length(DRP)]); set(gca,'YTick',0:360:360*DRP);
809     set(gca,'YLim',[0 360*maxDRP]);
810
```

```
811         %% Find half integers
812         DRPfirst(1,1) = DRPss{2,4};
813         i1 = 2;
814         for i = 3:length(DRPss)-1
815             if DRPss{i,2} == 0
816                 DRPfirst(i1,1) = DRPss{i+1,4};
817                 DRProse(i1,1) = DRPss{i,4};
818                 i1 = i1+1;
819             else
820             end
821         end
822
823         %% Get percentage of coupling frequencies
824         for i = 1:length(DRPfirst)
825             for i1 = 1:maxDRP
826                 if DRPfirst(i) > (i1-1) * 360 && DRPfirst(i) <= i1*360
827                     freqFromDRP(i) = i1;
828                 else
829                 end
830             end
831         end
832
833         if freqFromDRP(1) == freqFromDRP(3) && freqFromDRP(2) == freqFromDRP(4)
834     && abs( freqFromDRP(1)-freqFromDRP(2) ) == 1
835             DRPfirstHalfInt(1) = ( freqFromDRP(1) + freqFromDRP(2) ) / 2;
836         else
837             DRPfirstHalfInt(1) = freqFromDRP(1);
838         end
```

```
839
840       last=length(freqFromDRP); counter = 0;
841       for i = 2:last-1
842           if freqFromDRP(i-1) == freqFromDRP(i+1) && abs(freqFromDRP(i)-
843   freqFromDRP(i+1)) == 1
844               DRPfirstHalfInt(i) = ( freqFromDRP(i)+freqFromDRP(i+1) ) / 2;
845               counter = counter+1;
846           else
847               if mod(counter,2) == 1 && counter<i && counter > 1
848                   DRPfirstHalfInt(i-counter-1) = DRPfirstHalfInt(i-counter);
849               else
850               end
851               DRPfirstHalfInt(i) = freqFromDRP(i);
852               counter = 0;
853           end
854       end
855       DRPfirstHalfInt(last) = freqFromDRP(last);
856
857       %First
858       if DRPfirstHalfInt(1) == DRPfirstHalfInt(2)
859           DRPwithNC(1) = DRPfirstHalfInt(1);
860       else
861           DRPwithNC(1) = 0; %NC
862       end
863       %Middle
864       for i = 2:length(DRPfirstHalfInt)-1;
865           if DRPfirstHalfInt(i) == DRPfirstHalfInt(i+1) || DRPfirstHalfInt(i)
866   == DRPfirstHalfInt(i-1)
```

```
867                 DRPwithNC(i) = DRPfirstHalfInt(i);
868             else
869                 DRPwithNC(i) = 0; %NC
870             end
871         end
872         %End
873         last = length(DRPfirstHalfInt);
874         if DRPfirstHalfInt(last) == DRPfirstHalfInt(last-1)
875             DRPwithNC(last) = DRPfirstHalfInt(last);
876         else
877             DRPwithNC(last) = 0; %NC
878         end
879
880         subplot(3,2,5:6); plot(DRPfirst,'k-o'); hold on;
881         maxDRP = round(max(DRPfirst)/360)+1;
882         for i = 1:maxDRP
883             plot([0,length(DRPfirst)],[360*i,360*i],'k');
884             for i1 = 1:length(DRPfirst)
885                 if DRPfirst(i1) > 360*(i-1) && DRPfirst(i1) <= 360*i
886                     count(i1,1) = i;
887                     text(i1,DRPfirst(i1),['
888 ',num2str(DRPwithNC(i1))],'VerticalAlignment','bottom');
889                 else
890                 end
891             end
892         end
893         set(gca,'XLim',[1 length(DRPfirst)]); set(gca,'YTick',0:360:360*maxDRP);
894 set(gca,'YLim',[0 360*maxDRP]);
```

```
895
896         %%
897         maxn = max(DRPwithNC);
898         freq(1,:) = {'Ratio','Occurances','PercentageofTotal'};
899         freq{2,1} = 0; %NC
900         freq{2,2} = length(DRPwithNC(DRPwithNC==0));
901         freq{2,3} = freq{2,2}/length(DRPwithNC)*100;
902         for i = [1:0.5:maxn]
903             freq{i*2+1,1} = i;
904             freq{i*2+1,2} = length(DRPwithNC(DRPwithNC==i));
905             freq{i*2+1,3} = freq{i*2+1,2}/length(DRPwithNC)*100;
906         end
907
908         %%
909         [B,~] = size(freq);
910         subplot(3,2,2);
911         for i = 2:B
912             bar(freq{i,1},freq{i,3},0.5,'k'); hold on;
913             if freq{i,3} > 0
914                 text(freq{i,1}-0.025*B,-
915 1,sprintf('%2.2f%%',(freq{i,3})),'VerticalAlignment','bottom','Color','w','Fo
916 ntSize',12,'FontWeight','bold');
917             end
918         end
919         holder = freq{i,1}+0.5;
920
921         set(gca,'XLim',[-0.5 maxn+1]);
922         set(gca,'XTick',[(0:0.5:maxn+0.5)]);
```

```
923         set(gca,'XTickLabel',{'NC',(0.5:0.5:maxn),'PC'});
924         set(gca,'YTick',[0:10:100]);
925         set(gca,'YLim',[0 100]);
926
927         %%
928         [a,~] = size(freq);
929         for i = 2:a
930             c(i) = freq{i,3};
931         end
932         [MCFRpercent,MCFRindex] = max(c(2:length(c)));
933         MCFR = freq{MCFRindex+1,1};
934
935         %% Check which frequencies are present;
936         clear f;
937         for i = 3:length(freq)
938             if freq{i,3} ~= 0 && freq{i,3} <= 100
939                 if mod(freq{i,1},1) ~= 0
940                     mult = 2;
941                 else
942                     mult = 1;
943                 end
944                 if exist('f')
945                     f = [f,freq{i,1}*mult];
946                 else
947                     f = freq{i,1}*mult;
948                 end
949             end
950         end
```

```
951         f = sort(f);
952
953         [~,location] = max([freq{3:length(freq),2}]);
954         location = location+1;
955         maxLag = location+1;
956         if mod(location,2) == 0
957           location = location/2;
958         end
959         maxLag4PC = round(freq{maxLag,1});
960
961         %% Return maps
962         n=1;
963         for lag = location
964             original = DRP(1:length(DRP)-lag);
965             lagged = DRP(1+lag:length(DRP));
966
967             %% For each 360-multiple
968             for k = 1:maxLag4PC
969                 original_dn = original(original <= 360*k & lagged <=360*k);
970                 lagged_dn = lagged(original <= 360*k & lagged <=360*k);
971                 dn = ( (original_dn - lagged_dn) * cosd(45) );
972
973                 for i = 1:length(dn)
974                     if abs(dn(i)) <= 40 %*cosd(45)
975                         wdn(i) = 1 - ( abs(dn(i)) / (40)); %*cosd(45)) );
976                     else
977                         wdn(i) = 0;
978                     end
```

```
            end
            PC = sum(wdn) / length(wdn) * 100;
            LagPC(k,:) = [lag,PC];
            n = n+1;
        end
    end

    [~,lag4maxPC] = max(LagPC(:,2));
    index = LagPC(lag4maxPC,1);
    [r,~] = size(freq);
    freq{r+1,1} = '---';
    freq{r+1,2} = '---';
    freq{r+1,3} = '---';
    freq{r+2,1} = 'PC';
    freq{r+2,2} = index;
    freq{r+2,3} = max(LagPC(:,2));
    subplot(3,2,2); hold on; bar(holder,LagPC(lag4maxPC,2),0.5,'k');
    text(holder-0.025*B,-
1,sprintf('%2.2f%%',LagPC(lag4maxPC,2)),'VerticalAlignment','bottom','Color',
'w','FontSize',12,'FontWeight','bold');
    original = DRP(1:length(DRP)-index);
    lagged = DRP(1+index:length(DRP));

    %% Tab 2
    a2 = axes('parent', tab2);
    scatter(original,lagged,'k'); hold on;
    plot([0 max(DRP)],[0 max(DRP)],'k'); plot([0+40 max(DRP)+40],[0
max(DRP)],'r--'); plot([0-40 max(DRP)-40],[0 max(DRP)],'r--');
```

```
1007        title(['Lag = ',num2str(index)]); set(gca,'XLim',[0,max(DRP)]);
1008    set(gca,'YLim',[0,max(DRP)]); set(gca,'XTick',0:360:max(DRP));
1009    set(gca,'YTick',0:360:max(DRP)); grid on;
1010        for i = 1:round(MCFR)
1011            annotation(tab2,'textbox',[0.1025+(0.775)/(max(DRP))*(180+360*(i-1))
1012    , 0.075 , 0.0325 ,
1013    0.02],'String',num2str(LagPC(i,2)),'FontSize',20,'EdgeColor','none');
1014        end
1015
1016        %% Tab 3
1017        a3 = axes('parent', tab3); rose(DRProse/180*pi,20);
1018        [r,c] = size(freq);
1019        fprintf('\r');
1020        for i = 1:r
1021            for i1 = 1:c
1022                if i == 1
1023                    %Do nothing
1024                elseif i == r-1
1025                    fprintf('%s\t\t',freq{i,i1});
1026                elseif i == r
1027
1028                else
1029                    fprintf('%2.1f\t\t',freq{i,i1});
1030                end
1031            end
1032            if i == 1
1033                fprintf('%s\t\t%s\t%s',freq{i,:});
1034            elseif i == r
```

```
            fprintf('%s\t\t%2.2f\t\t%2.2f',freq{i,:});
        end
        fprintf('\r');
    end
    fprintf('\r');

    filename = outputFilename;
    if iscell(filename)
        filename = filename{1};
    end

    if  saveFigs == 1
        savefig(F1,[filename '_DRP.fig']);
        savefig(f2,[[newSub_dir filename '_' ] 'returnMap.fig'])
        save([filename '_DRP.mat'],'DRPss')
        save([filename '_freq.mat'],'freq')
    else
        display('Figures not saved ...');
    end

    match = 0;
    if exist('DRPresults.mat') ~= 0
        load('DRPresults.mat');
        [~,l] = size(DRPresults);
        for i = 1:l
            if strfind(DRPresults(i).name, filename) == 1
                l = i;
                match = 1;
```

```
            else
                if i == l && match == 0
                    l = l+1;
                end
            end
        end
    else
        l = 1;
    end
    DRPresults(l).name = filename;
    DRPresults(l).MCFR = MCFR;
    DRPresults(l).MCFRpercent = MCFRpercent;
    DRPresults(l).PC = LagPC(:,2);
    DRPresults(l).PClag = LagPC(lag4maxPC,1);
    DRPresults(l).freq = freq;
    save('DRPresults','DRPresults');
end

```