dynamicMatch

Match two note matrices

Syntax

```
M = dynamicmatch(pmat, nmat, title (optional));
```

Description

M = dynamicMatch(pmat, nmat) matches one note matrix to another using dynamic programming.

If `pmat` is a note matrix that holds a performance and `nmat` is a note matrix that holds the notation for that performance, the algorithm find the match that optimizes a scoring function. The scoring function is built in to the algorithm and its parameters are not tunable. It matches individual notes in the performance to chords (called groups) in the notation file, and the scoring function uses both pitch and timing information. Based on the match, the program then determines which notes in the performance should be grouped into chords.

A match structure is returned. The fields of the structure are as follows:

title	Title of the piece
Nn	Note matrix of the notation (input to program)
Np	Note matrix of the performance (input to program)
GIn	Group indices for notation (chords). Before the match, this array is created. Any notes that have the same onset beat are grouped into chords. Each row contains a chord (maximum of 10 notes).
GIp	Group indices for performance (chords), before match. Before the match nothing is grouped, so only the first column contains indices into Np.
indN	Indices (into GIn) of notated chords that match performance. Contains only chords that match. If the chord was not performed (it was a deleted in performance), it is not in this vector.
indP	Indices (into Np) of performednotes that match notated chords. Contains only chords that match. If the chord was not in the notation (it was added in performance), it is not in this vector.
mtbl	Match table is a 2-D array that describes the match of notated chords to performed notes. Each non-zero value is the matching score of a performed note. 10 = exact match, 3 = non-exact match.
realGIp	Group indices for performance (chords), after match. The grouping is done after the match, and depends not only on temporal informa-

	tion, but also on how well notes match notated chords.
Tn	Onset times of notated chords that match performance. Contains only chords that match. If the chord was not performed (it was a deleted in performance), it is not in this vector.
Bn	Onset beats of notated chords that match performance. Contains only chords that match. If the chord was not performed (it was a deleted in performance), it is not in this vector.
Tp	Onset times of performed chords that match performance. Contains only chords that match. If the chord was not in the notation (it was added in performance), it is not in this vector. The onset time of the performed chord is calculated as the mean onset time of the notes in the chord.
Bp	Onset beats of performed chords that match performance. Contains only chords that match. If the chord was not in the notation (it was added in performance), it is not in this vector. For this reason `Bp = Bn`, always.
tempoMap	Tempo map based on match – beats per second (Hz). The tempo map is `diff(Bp) ./ diff(Tp)`. The first value is appended to the beginning, so it is the same length as Bn.
periodMap	Period map based on match – inter-beat intervals (ms). The period map is `diff(Tp) ./ diff(Bp)`. The first value is appended to the beginning, so it is the same length as Bn.
velocMap	Velocity map for matched chords. MIDI velocities of chords that match. It is calculated as the mean velocity of the notes in the chord.
Nm	Note matrix containing the match. This can be played via play midi. You will hear a time-quantized version of the performance, minus any notes that were added in performance (not in the notation). It does not contain any notes that were deleted in performance either.
M	Match report, a table containing information on the performace, which notes match, which were substituted (wrong notes), which were added and which were deleted. The complete strcutre of the table is described below.

The format of the match report is

	nota ted	perform	nota ted	perform	nota ted	perform	nota ted	perform	nota ted	perform
status	index	index	note #	note #	onset	onset	velocity	velocity	duration	duration

Status is one of

`'M'` – match: a performed note matched the notated note exactly.

`'sub'` – substitution: a performed note was substituted for one in the score

`'add'`, – addition: a note that did not appear in the score was added in performance

`'del'` – deletion: a note that appeared in the score was not performed

Example

```
nmat = readmidi('twinkleNotation.mid');
pmat = readmidi('twinklePerformance.mid');

M = dynamicmatch(pmat, nmat, 'Twinkle, Twinkle Little Star')


M =

    title: 'Twinkle, Twinkle Little Star'
       Nn: [42x7 double]
       Np: [42x7 double]
      GIn: [42x10 double]
      GIp: [42x10 double]
     indN: [1x42 double]
     indP: [1x42 double]
     mtbl: [42x42 double]
  realGIp: [41x10 double]
       Tn: [41x1 double]
       Bn: [41x1 double]
       Tp: [41x1 double]
       Bp: [41x1 double]
 tempoMap: [41x1 double]
periodMap: [41x1 double]
 velocMap: [41x1 double]
       Nm: [42x7 double]
        M: {43x11 cell}
```

If we examine `M` in the array viewer, we find that

'M'	1	1	60	60	0	1.3906	82	82	0.5	0.2998
'M'	2	2	60	60	0.5	1.8989	72	83	0.5	0.5479
'M'	3	3	67	67	1	2.4443	91	81	0.5	0.3389
'M'	4	4	67	67	1.5	2.9624	84	90	0.5	0.3364
'M'	5	5	69	69	2	3.478	89	73	0.5	0.0864
'M'	6	6	69	69	2.5	3.75	91	64	0.5	0.25
'add'	6	7	NaN	69	NaN	4	NaN	83	NaN	0.4819
'M'	7	8	67	67	3	4.46	94	81	1	0.9731
'M'	8	9	65	65	4	5.4893	88	81	0.5	0.3457

'M'	9	10	65	65	4.5	6	87	83	0.5	0.5029
'M'	10	11	64	64	5	6.5103	86	80	0.5	0.8384
'del'	11	NaN	64	NaN	5.5	NaN	76	NaN	0.5	NaN
'M'	12	12	62	62	6	7.4854	87	81	0.5	0.2988
'M'	13	13	62	62	6.5	8	72	90	0.5	0.5
'M'	14	14	60	60	7	8.4673	93	72	1	1.0815
'M'	15	15	67	67	8	9.5469	86	90	0.5	0.2998
'M'	16	16	67	67	8.5	10.0444	76	83	0.5	0.4854
'M'	17	17	65	65	9	10.5068	89	73	0.5	0.3594
'M'	18	18	65	65	9.5	10.9985	86	83	0.5	0.5439
'M'	19	19	64	64	10	11.502	91	82	0.5	0.3301
'M'	20	20	64	64	10.5	12	80	82	0.5	0.4614
'M'	21	21	62	62	11	12.4707	93	81	1	0.8799
'M'	22	22	67	67	12	13.4819	84	83	0.5	0.3096
'M'	23	23	67	67	12.5	14	75	90	0.5	0.5122
'M'	24	24	65	65	13	14.478	91	80	0.5	0.4165
'M'	25	25	65	65	13.5	14.979	86	82	0.5	0.5093
'M'	26	26	64	64	14	15.4863	91	81	0.5	0.355
'M'	27	27	64	64	14.5	16	84	82	0.5	0.481
'sub'	28	28	62	59	15	16.5	93	64	1	1
'M'	29	29	60	60	16	17.4468	82	81	0.5	0.3403
'M'	30	30	60	60	16.5	17.9287	72	83	0.5	0.54
'M'	31	31	67	67	17	18.4351	91	90	0.5	0.3325
'M'	32	32	67	67	17.5	18.9268	84	90	0.5	0.3262
'M'	33	33	69	69	18	19.3955	89	82	0.5	0.3457
'M'	34	34	69	69	18.5	19.8936	91	82	0.5	0.5127
'M'	35	35	67	67	19	20.3735	94	81	1	1.0493
'M'	36	36	65	65	20	21.3926	88	81	0.5	0.2998
'M'	37	37	65	65	20.5	21.8643	87	81	0.5	0.5166
'M'	38	38	64	64	21	22.3579	86	81	0.5	0.3291
'M'	39	39	64	64	21.5	22.8477	76	82	0.5	0.5352
'M'	40	40	62	62	22	23.3433	87	80	0.5	0.3198
'M'	41	41	62	62	22.5	23.8047	72	82	0.5	0.5439
'M'	42	42	60	60	23	24.2979	93	81	1	1.104

We can also view and edit the match in the gui match window, using `gwm.m`.

Algorithm

The matcher utilizes dynamic programming techniques, runs in polynomial time and is fully described in (Large, 1993). The algorithm for this task was developed in the context of a study of music production errors (Palmer & van de Sande, 1993). The dynamic programming algorithm finds an optimal match between the two sequences, given a scoring function. The scoring function is hard coded into this program: exact match (note-for-note) = 10 points, substitution (wrong note) = 3 points. No points are given for additions or deletions. The original program written for (Palmer & van de Sande, 1993)

required that the two performances be grouped into chords before matching the sequences. This was found to cause difficulties for very long performances, because often no adequate temporal criterion can be found. In this implementation, the notation matrix is grouped into chords (notes beginning on the same beat are chords); it does not group the performance before the match. It also uses an additional scoring function for timing information, and optimizes the sum of the two scoring functions. Thus, performed notes are matched to notated chords, yielding a many-to-one match. This match is summarized in mtbl, indN and indP (see above). Based on this information, the program goes on to group the performance into chords. Because of this added complexity, the matcher may occasionally match incorrectly (this almost always has to do with identifying chords incorrectly). Thus a graphical match inspector / editor is provided to inspect and correct the output (`gmw.m`).

References

Large, E. W. (1993). Dynamic programming for the analysis of serial behaviors. *Behavior Research Methods, Instruments, and Computers, 25 (2)*, 238-241.

Palmer, C. & van de Sande, C. (1993). Units of knowledge in music performance. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 19*, 457-470.

See Also

`gmw.m`

gmw

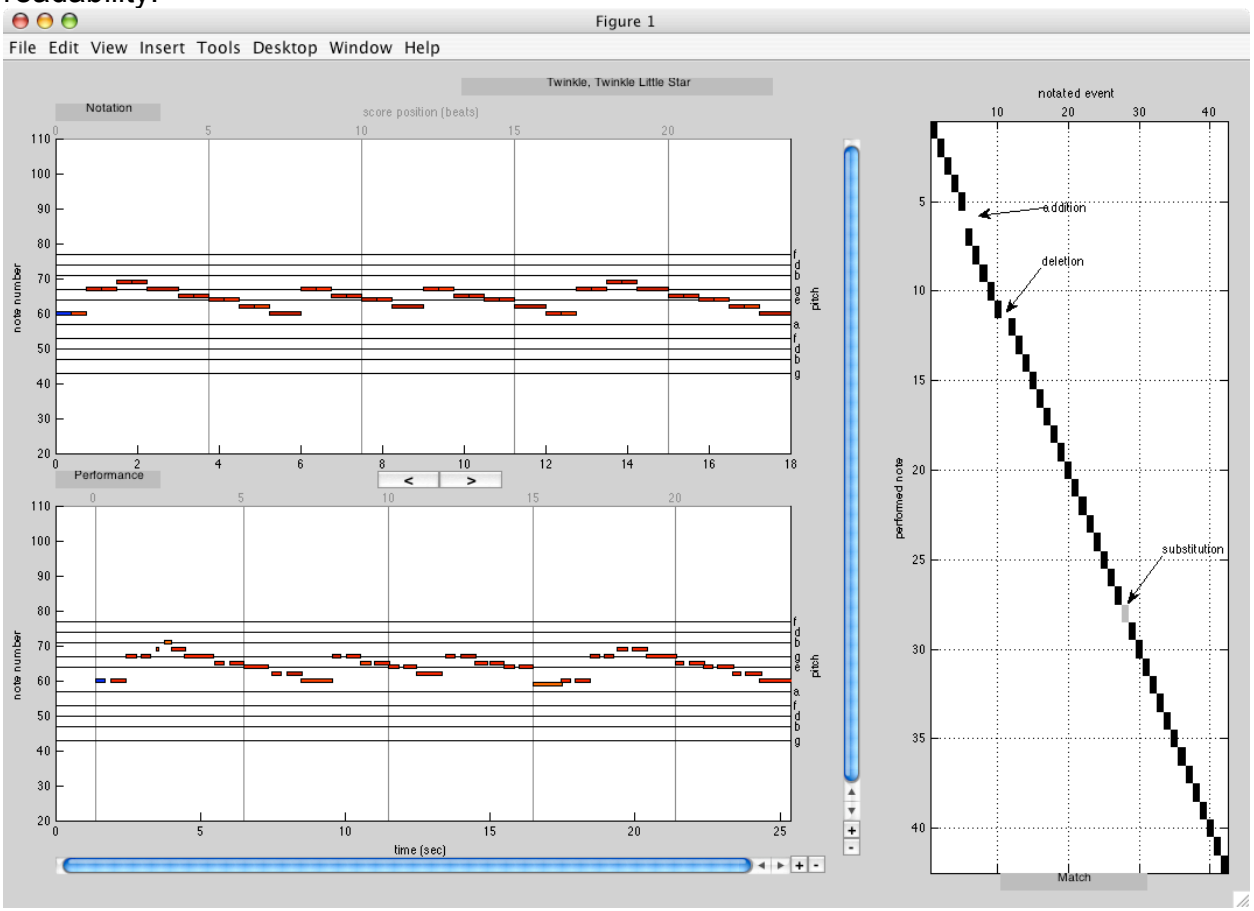
Gui Match Window: Match inspector & Editor

Syntax

```
gmw(M);
```

Description

`gmw(M)` creates a graphical user interface that allows you to view and edit the match structure calculated by **`dynamicMatch.`** On the left side of the window, the piano roll graph of the notation and performance note matrices is shown. On the piano roll graphs, time in seconds is displayed on the bottom x-axis; time in beats is shown on the top of the x-axis. MIDI note number is displayed on the y-axis left side; the pitch or note name is displayed on the right side of the y-axis. A grand staff is also displayed for readability.



By clicking on a note in the notation graph, one can select all the notes in that chord, and also the notes in the performance that match the chord. By clicking on the forward and backward arrow buttons between the two piano roll graphs one can move backward and forward to inspect the output of the matching process.

As described in `dynamicMatch.m`, each note in the performance is matched to chord in the notation and the result is returned on `mtbl`. On the right hand side of the window the match table is presented graphically. This shows in another way which notes of the performance were matched to which chords in the notation.A black square denotes a "match", the performed note matches a note in the notated chord exactly. A gray square denotes a "substitution". A missing square for a notated chord denotes a "deletion", and a missing square for a performed note indicates an "addition". As one steps through the match in the piano roll windows, the corresponding entry in the match table is also highlighted.

Due to complexities of the matching process and data analysis requirements, the matches may need to be corrected slightly. This can be achieved in the match window. Clicking on any square changes the match For a *match* or *substitution* (black or gray square) a click removes it from the match. Clicking on a square that is filled will delete the match and that note in the performance will not be matched to anything in the notated. This is what Palmer and van de Sande (1993) called an *addition*. This will result in an empty row, in the match window. A deletion (Palmer & van de Sande (1993)) occurrs when there is a missing note in the performance. Deleting the match for a chord in the notation will leave an empty column. By clicking on a white square, the notated chord/performed note combination is added to the match. A black or gray square will appear there, and any other match in the same row is replaced. You can match multiple notes in the performance to one chord in the notation.

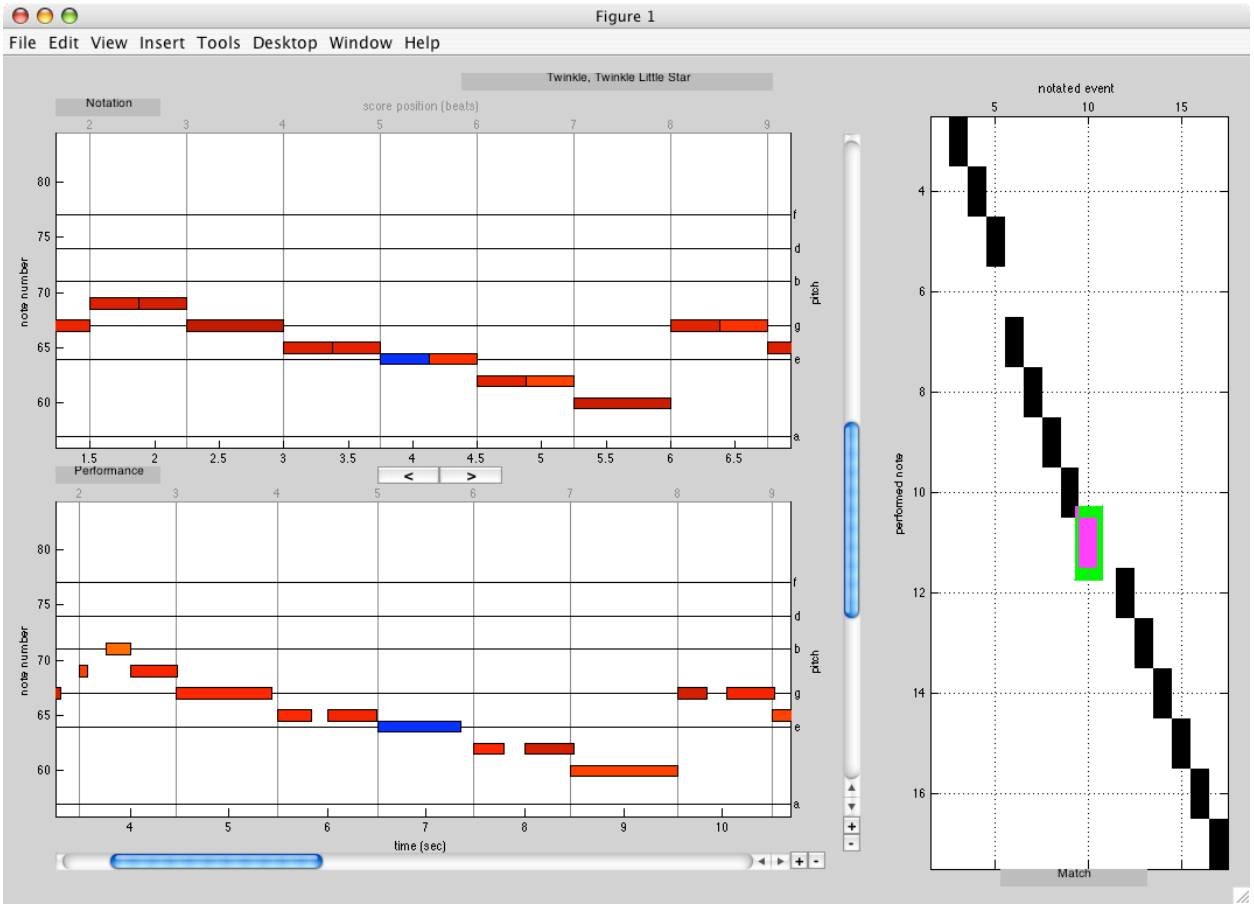
Important: Every time you make a change, `gmw` saves it in the base workspace as the variable M. If your original match structure was named M, it will be overwritten. This could be a good thing of a bad thing, depending on how you want to work. We recommend naming the match variable M, because as you edit the match, you will also be able to view the results fof the changes in by keeping the array editor open and viewing `M.M`. We do this in the example below.

Example

```
nmat = readmidi('twinkleNotation.mid');
pmat = readmidi('twinklePerformance.mid');

M = dynamicmatch(pmat, nmat, 'Twinkle, Twinkle Little Star')


M =

    title: 'Twinkle, Twinkle Little Star'
       Nn: [42x7 double]
       Np: [42x7 double]
      GIn: [42x10 double]
      GIp: [42x10 double]
     indN: [1x42 double]
     indP: [1x42 double]
     mtbl: [42x42 double]
  realGIp: [41x10 double]
       Tn: [41x1 double]
```

```
        Bn: [41x1 double]
        Tp: [41x1 double]
        Bp: [41x1 double]
 tempoMap: [41x1 double]
periodMap: [41x1 double]
 velocMap: [41x1 double]
        Nm: [42x7 double]
         M: {43x11 cell}
```

```
gmw(m);
```

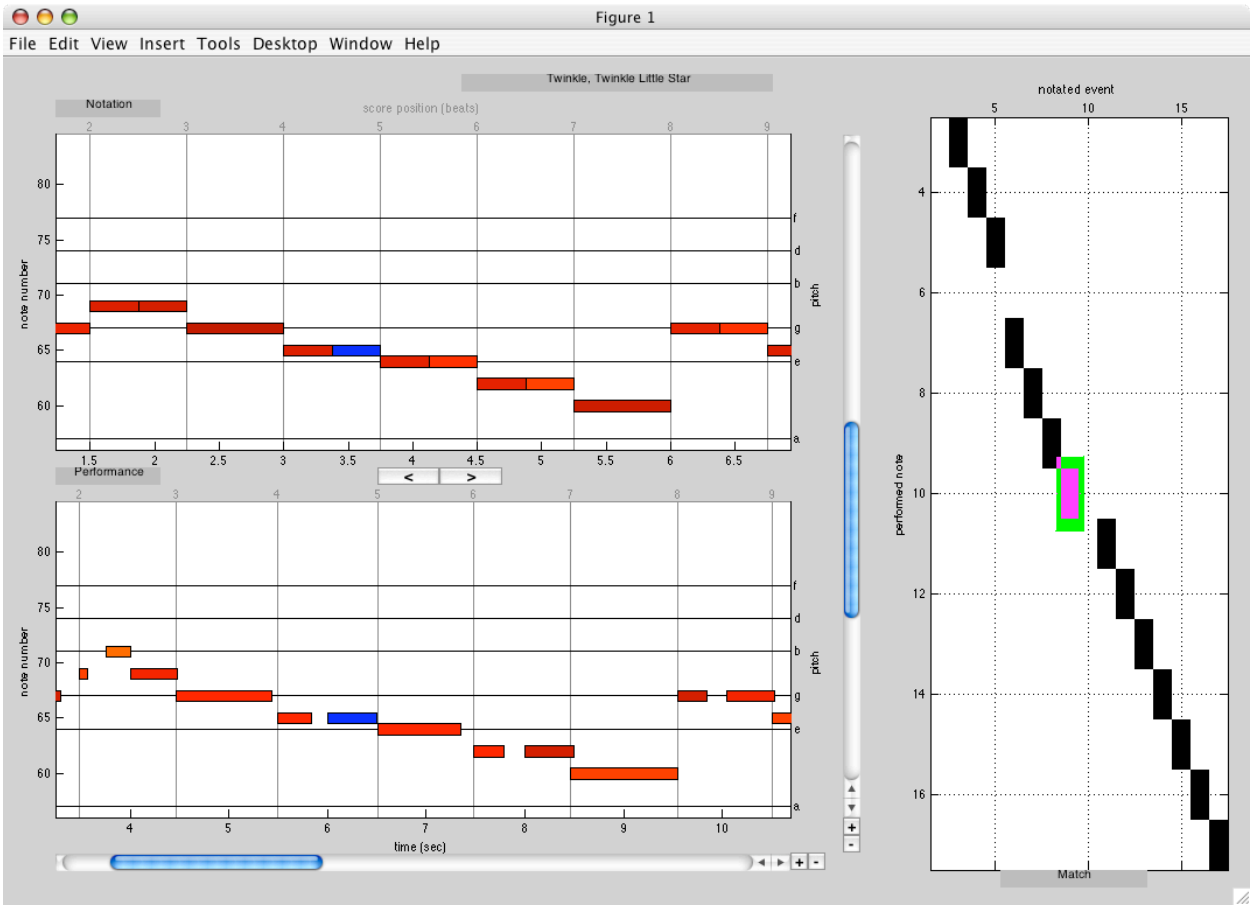


Opening M.M in Matlab's Array Editor, we see:

'M'	8	9	65	65	4	5.4893	88	81	0.5	0.3457
'M'	9	10	65	65	4.5	6	87	83	0.5	0.5029
'M'	10	11	64	64	5	6.5103	86	80	0.5	0.8384
'del'	11	NaN	64	NaN	5.5	NaN	76	NaN	0.5	NaN
'M'	12	12	62	62	6	7.4854	87	81	0.5	0.2988
'M'	13	13	62	62	6.5	8	72	90	0.5	0.5
'M'	14	14	60	60	7	8.4673	93	72	1	1.0815

In this example the two E's in the notated version are not both performed, instead a single E is played. Thus, the program registers a *deletion*. To change the performed E to match the second notated E, 1) unselect this event by moving the selection to the previous event, and then 2) click on square 11,11 in the match window.

Then the graphics window will look like this:



and the array editor will display this:

'M'	8	9	65	65	4	5.4893	88	81	0.5	0.3457
'M'	9	10	65	65	4.5	6	87	83	0.5	0.5029
'del'	10	NaN	64	NaN	5	NaN	86	NaN	0.5	NaN
'M'	11	11	64	64	5.5	6.5103	76	80	0.5	0.8384
'M'	12	12	62	62	6	7.4854	87	81	0.5	0.2988
'M'	13	13	62	62	6.5	8	72	90	0.5	0.5
'M'	14	14	60	60	7	8.4673	93	72	1	1.0815

References

Large, E. W. (1993). Dynamic programming for the analysis of serial behaviors. Behavior Research Methods, Instruments, and Computers, 25 (2), 238-241.

Palmer, C. & van de Sande, C. (1993). Units of knowledge in music performance. Journal of Experimental Psychology: Learning, Memory, & Cognition, 19, 457-470.