PHONMAT

Dinoj Surendran, dinoj@cs.uchicago.edu

Download the following zip files: phonmat.zip, matutils.zip, phonvec.zip, labels.zip. Unzip them in the same director say blah. This causes the files in matutils.zip to appear in blah, and the directories @phonmat, @phonvec and @labels t appear in blah. You need to add blah to the Matlab search path.

PHONMAT is a Matlab class that I wrote and used in early 2003. It was never written with speed in mind, so even on a 1. GHz linux box with 1 Gb RAM it takes just over two seconds to display a 16x16 matrix.

This class helps you manipulate data of the following form: suppose you have a finite set of symbols S and for each pair s,t in S have an associated value M(s,t). Pair - unordered pair or ordered pair? PHONMAT can deal with both, but assume the former for now. In other words M(s,t) and M(t,s) should be treated differently. Also, PHONMAT can deal with the cas where diagonal entries M(s,s) aren't meaningful, but assume they are for now.

An example of such a matrix is a confusion matrix like that of Miller and Nicely, 1955. This is a stimulus-response matrix with M(s,r) having the number of times subjects faced with stimulus s gave response r.

Constructing a PHONMAT object

Suppose the file toyexample.dat (the extension can be anything you like, not just .dat; you don't even need one) looks like this:

```
% Toy example to illustrate use of PHONMAT class
% abcdefg
% a ay aa f F c see
341   43    431    85     95    31    5
90   531     53    91     38    24   21
71    38    493    12    102    43   89
31    49     11   643     32    13   95
93    14     58    10    488   120   41
23    49     59    28     82   710   12
83    23     58    39     20    43  501
```

The first line is the title of the matrix, and is required. The second line, also required, is a list of the symbols in S. All these symbols MUST be 1-character long, and are called 1-char labels as a result. The third line, which is optional, has a list of alternative labels (altlabels) for some of the 1-char labels. For example, a can be called ay or aa, c can be called see, and f can be called F.

To read this file, type

```
>> toy = phonmat ('toyexample.dat');
```

There are other ways of creating PHONMAT objects, such as copying

```
toy2 = toy;
```

Or by initializing from a file with the first line having the matrix's title (optional) and the rest having individual entries For example, suppose this is the file choochoo.dat.

```
% chitty chitty choo choo
xx 134
yy 155
xz 10
zx 214
yx 24
yz 120
zy 31
xy 43
zz 14
```

Then

```
>> choochoo = phonmat ('choochoo.dat')

 choochoo (object of type PHONMAT) =

    title:
    Phones involved: 3, namely  x y z

      x     y     z
    x 134   43    10      x
    y 24    155   120     y
    z 214   31    14      z
```

Looking at a PHONMAT object

To view it, just type its name (or leave out the semicolons when you read it in the previous line).

```
>> toy

 toy (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class
    Phones involved: 7, namely  a (ay aa) b c (see) d e f (F) g

      a     b     c     d     e     f     g
    a 341   43    431   85    95    31    5       a
    b 90    531   53    91    38    24    21      b
    c 71    38    493   12    102   43    89      c
    d 31    49    11    643   32    13    95      d
    e 93    14    58    10    488   120   41      e
    f 23    49    59    28    82    710   12      f
    g 83    23    58    39    20    43    501     g
```

To display toy with row totals, type

```
>> total (toy)
```

or

```
>> toy.total;

 pm (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class
    Phones involved: 7, namely a (ay aa) b c (see) d e f (F) g

       a    b    c    d    e    f    g      Total
    a  341  43   431  85   95   31   5    a 1031
    b  90   531  53   91   38   24   21   b 848
    c  71   38   493  12   102  43   89   c 848
    d  31   49   11   643  32   13   95   d 874
    e  93   14   58   10   488  120  41   e 824
    f  23   49   59   28   82   710  12   f 963
    g  83   23   58   39   20   43   501  g 767
```

Don't worry about the fact that 'pm' appears when you use the second command; that's a display bug that I didn't consider worth fixing.

Individual element access

Suppose you want to know the entry corresponding to 'a' and 'f' in toy.

```
>> toy('af')

ans =

    31
```

However, having 1-char labels isn't always convenient. This is why alternative labels can be useful; they prevent you from having to remember 1-char labels. Recall that 'ay' and 'aa' are altlabels for 'a' and 'F' for 'f'. Any of the following commands are equivalent to toy ('af') :

```
 toy ('a','f')
 toy ('aa','f')
 toy ('ay','f')
 toy ('a','F')
 toy ('aa','F')
 toy ('ay','F')
```

There is no limit on how long altlabels can be.

Using curly brackets (Americans call them braces, oui?) for access in any of the above commands returns a 2x2 matrix involving the two symbols involved e.g.

```
>> toy('af')

ans =

    31

>> toy('fa')

ans =

    23

>> toy{'fa'}                       % <--- note that {} used instead of ()

ans =

   710    23
    31   341
```

Reordering and taking submatrices

Now suppose you want to reorder the matrix so that the order isn't abcdefg but fadgceb.

```
>> reorder (toy,'fadgceb')

 ans (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class
    Phones involved: 7, namely  f a d g c e b

      f     a     d     g     c     e     b
    f 710   23    28    12    59    82    49      f
    a 31    341   85    5     431   95    43      a
    d 13    31    643   95    11    32    49      d
    g 43    83    39    501   58    20    23      g
    c 43    71    12    89    493   102   38      c
    e 120   93    10    41    58    488   14      e
    b 24    90    91    21    53    38    531     b
```

Suppose you wanted to just have a look at how c,g and b compared. You could say

```
>> reorder (toy,'cdg')

 ans (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class
    Phones involved: 7, namely  c d g a b e f

      c     d     g     a     b     e     f
    c 493   12    89    71    38    102   43      c
    d 11    643   95    31    49    32    13      d
    g 58    39    501   83    23    20    43      g
    a 431   85    5     341   43    95    31      a
    b 53    91    21    90    531   38    24      b
    e 58    10    41    93    14    488   120     e
    f 59    28    12    23    49    82    710     f
```

Which places 'cdg' at the start of the matrix and places the other labels at the end in the original order. Or you could say

```
>> sub (toy,'cdg')

 ans (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class (EXTRACTED FROM MATRIX INVOLVING abcdefg)
    Phones involved: 3, namely  c d g

      c     d     g
    c 493   12    89      c
    d 11    643   95      d
    g 58    39    501     g
```

Both the 'sub' and 'reorder' functions return PHONMAT objects. For example:

```
>> babytoy = sub (toy,'cdg');
>> babytoy

 babytoy (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class (EXTRACTED FROM MATRIX INVOLVING abcdefg)
    Phones involved: 3, namely  c d g

      c     d     g
    c 493   12    89      c
    d 11    643   95      d
    g 58    39    501     g
```

You can convert the entries of 'toy' to a vector:

```
>> toy.list

ans =

  Columns 1 through 27

   341    43   431    85    95    31     5    90   531    53    91
   38     24    21    71    38   493    12   102    43    89    31
   49    11   643    32    13

  Columns 28 through 49

    95    93    14    58    10   488   120    41    23    49    59
    28    82   710    12    83    23    58    39    20    43   501
```

This is useful in comparing matrices. For example, suppose you have a second PHONMAT toy2 (intentionally created as perturbation of toy).

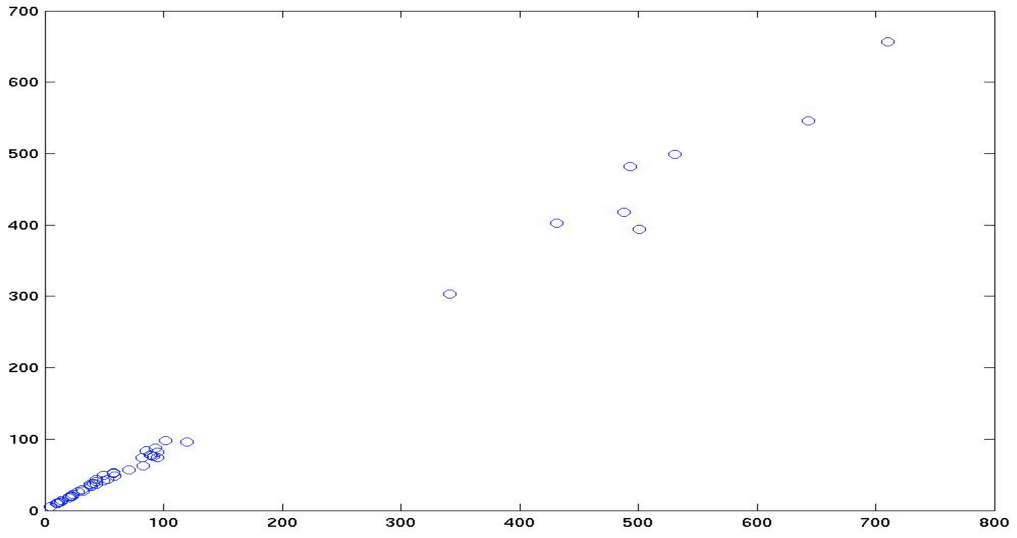
```
toy2 (object of type PHONMAT) =

    title: Another toy example to illustrate use of PHONMAT class
    Phones involved: 7, namely  a (ay aa) b c (see) d e f (F) g

      a     b     c     d     e     f     g
    a 303   37    403   84    82    29    5       a
    b 77    499   43    76    35    22    20      b
    c 57    37    482   11    98    41    78      c
    d 29    49    10    546   27    12    74      d
    e 87    14    53    10    418   96    38      e
    f 20    42    48    26    74    657   11      f
    g 63    21    52    34    18    43    394     g
```

Now you want to compare corresponding entries of both matrices. You could do that by saying

```
>> figure; plot (toy.list, toy2.list, 'bo');
```



Making diagonal entries invisible

Let's suppose now that you wanted to only compare, for whatever reason, off-diagonal entries. To do this, use the 'removediag' command, which prevents you from accessing the diagonal entries.

```
>> removediag(toy)
>> toy

 toy (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class
    Phones involved: 7, namely  a (ay aa) b c (see) d e f (F) g

      a     b     c     d     e     f     g
    a ....  43    431   85    95    31    5       a
    b 90    ....  53    91    38    24    21      b
    c 71    38    ....  12    102   43    89      c
    d 31    49    11    ....  32    13    95      d
    e 93    14    58    10    ....  120   41      e
    f 23    49    59    28    82    ....  12      f
    g 83    23    58    39    20    43    ....    g
```

Note that the diagonal entries have not been removed, they are just invisible for now. To get them back, type 'removediag (toy,1)'. Let's assume we don't do that now however. The 'list' command only returns visible ('meaningful') matrix elements, in this case only offdiagonal ones.

```
>> toy.list

ans =

  Columns 1 through 27

    43   431    85    95    31     5    90    53    91    38    24
    21    71    38    12   102    43    89    31    49    11    32
    13    95    93    14    58

  Columns 28 through 42

    10   120    41    23    49    59    28    82    12    83    23
    58    39    20    43
```

Where were we? Oh right, we wanted to compare offdiagonal elements of toy and toy2.

```
>> removediag(toy2)
>> figure; plot (toy.list, toy2.list, 'bo');
```

We won't bother showing the picture, for lack-of-insight reasons. Let's put back the diagonal entries though.

```
>> removediag (toy,1)
>> removediag (toy2,1)
>> toy

 toy (object of type PHONMAT) =

    title: Toy example to illustrate use of PHONMAT class
    Phones involved: 7, namely  a (ay aa) b c (see) d e f (F) g

      a     b     c     d     e     f     g
    a 341   43    431   85    95    31    5       a
    b 90    531   53    91    38    24    21      b
    c 71    38    493   12    102   43    89      c
    d 31    49    11    643   32    13    95      d
    e 93    14    58    10    488   120   41      e
    f 23    49    59    28    82    710   12      f
    g 83    23    58    39    20    43    501     g
```

The innards of a PHONMAT object

There are (at last count) 8 fields of any PHONMAT object, say pm.

- pm.mat has the underlying matrix.
- pm.labels has the labels of the symbols involved.
- pm.title has the name of this matrix
- pm.symmetric specified whether the matrix is symmetric
- pm.hasdiag specified whether the matrix has meaningful diagonal entries
- pm.default -- don't worry about it.
- pm.smallest. Any value below smallest in magnitude is assumed to be 0.
- pm.dp is the number of decimal places to be used in displaying this object.

To get access to any object use the 'get' and 'set' commands. For example:

```
>> choochoo

 choochoo (object of type PHONMAT) =
    title:
    Phones involved: 3, namely  x y z

      x     y     z
    x 134   43    10      x
    y 24    155   120     y
    z 214   31    14      z

>> M = get (choochoo, 'mat')

M =

   134    43    10
    24   155   120
   214    31    14

>> M(2,2) = 130

M =

   134    43    10
    24   130   120
   214    31    14

>> set (choochoo, 'mat', M)
>> choochoo

 choochoo (object of type PHONMAT) =

    title:
    Phones involved: 3, namely  x y z

      x     y     z
    x 134   43    10      x
    y 24    130   120     y
    z 214   31    14      z
```

Now to explain each of these fields in more detail.

- pm.mat is a nxn array of numbers. The pm.mat(i,j) corresponds to the entry for the i-th and j-th label.

  One can read, but not directly change, individual elements of pm. The only way to change the matrix elements is t get (, change) and set the whole 'mat' field.
- pm.labels is of type LABELS and has the n labels involved here

  The LABELS class is explained in the next section.
- pm.title is a string with the name of this matrix
- pm.symmetric is 1 if the matrix is symmetric and 0 (default) otherwise.
- pm.hasdiag is 1 (default) matrix has meaningful diagonal entries, else 0.

  When you called 'removediag (toy)' earlier, what you actually did was set toy.hasdiag to 0. Calling 'removediag (toy,1)' set toy.hasdiag to 1 again.
- pm.default is 0 (default)

  This is the value returned if you ask for a diagonal entry when such entries have been marked as meaningless, e.g. saying "toy('aa')" just after you say "removediag (toy)".

  The way the class handles default values should be redone, since I wanted pm.default to always be returned for underspecified entries of pm. By underspecified I mean this -- a matrix can be underspecified during construction. For example, if 'choochoo_part.dat' was the file

  ```
  % chitty chitty choo choo
  xx 134
  yy 155
  xz 10
  zx 214
  ```

  Then these entries are placed in the matrix and all others are set to 0. This is fine if 0 is your default value, but can be problematic at other times.

  ```
  >> choochoo2 = phonmat ('choochoo_part.dat');
  >> choochoo2

  choochoo2 (object of type PHONMAT) =

  title: chitty chitty choo choo
  Phones involved: 3, namely  x y z
  ```

```
  x     y     z
x 134   0     10      x
y 0     155   0       y
z 214   0     0       z
```

- pm.smallest = 1e-10 (default). Any value below smallest is assumed to be 0.
- pm.dp is the number of decimal places to be used in displaying this object.

```
   >> choochoo

   choochoo (object of type PHONMAT) =

   title:
   Phones involved: 3, namely  x y z

     x     y     z
   x 134   43    10      x
   y 24    130   120     y
   z 214   31    14      z

   >> set (choochoo, 'dp', 2)
   >> choochoo

   choochoo (object of type PHONMAT) =

     title:
     Phones involved: 3, namely  x y z

       x     y     z

     All entries seen should be multiplied by 1e2

     x 1.34  0.43  0.10    x
     y 0.24  1.30  1.20    y
     z 2.14  0.31  0.14    z

  >> set (choochoo, 'dp', -1)
  >> choochoo

   choochoo (object of type PHONMAT) =

      title:
      Phones involved: 3, namely  x y z

        x      y      z

      All entries seen should be multiplied by 1e-1

      x 1340   430    100      x
      y 240    1300   1200     y
      z 2140   310    140      z
```

The LABELS class

```
>> help labels

 --- help for labels/labels.m ---

 LABELS  user-defined class to deal with labels for phone manipulation,
         especially in the context of classes PHONMAT and CONFMAT.

        Their use is best explained by example. Supposed you have an
        experiment with the English phones /t/, /d/, /th/ and /dh/.
        For convenience, define 1-character labels (called ONELABELS)
        for each, say t,d,T and D. This is the approach followed by
        the DISC format in CELEX for example. You would still like to
        remember that T stands for /th/ of course.

        blah = labels ('tdTD','T th D dh');
        blah{'th'}  --> 'T'
        blah('th')  --> 3    (position in original string definition)
        blah{'D'}   --> 'D'
        blah('D')   --> 4
        blah{'dh'}  --> 'D'
        blah{'zh'}  --> ''
        blah('zh')  --> 0
        blah.phones --> 'tdTD'
        blah.allphones  --> 't d T (th) D (dh)'

        The second argument in the initialization is optional. If you have
        no labels other than your onelabels, "blah = labels('tdTD')" is ok.

        At the moment there is no functionality to modify or delete labels as
        I have not needed it. Feel free to add and even freer to email
        me and tell me about your improved version.

        It is your responsibility to make sure that labels don't contradict
        each other. Saying addlabels (blah, 'T th D th') is definitely a
```

"Bad command! Go sit in the corner" kind of thing.